

# DR. D. Y. PATIL INSTITUTE OF ENGINEERING, MANAGEMENT & RESEARCH, AKURDI

Approved by A.I.C.T.E, New Delhi, Maharashtra State Government, Affiliated to Savitribai Phule Pune University Sector No. 29, PCNTDA, Nigdi Pradhikaran, Akurdi Pune 411044. Phone: 020-27654470, Fax: 020-27656566 Website: [www.dypiemr.ac.in](http://www.dypiemr.ac.in) Email: principal@dypiemr.ac.in

Ref.No: DYPIEMR/Admin/2021-22

Date: 2 /6 /2022

## NAAC 1.4.1 Curriculum Feedback Report

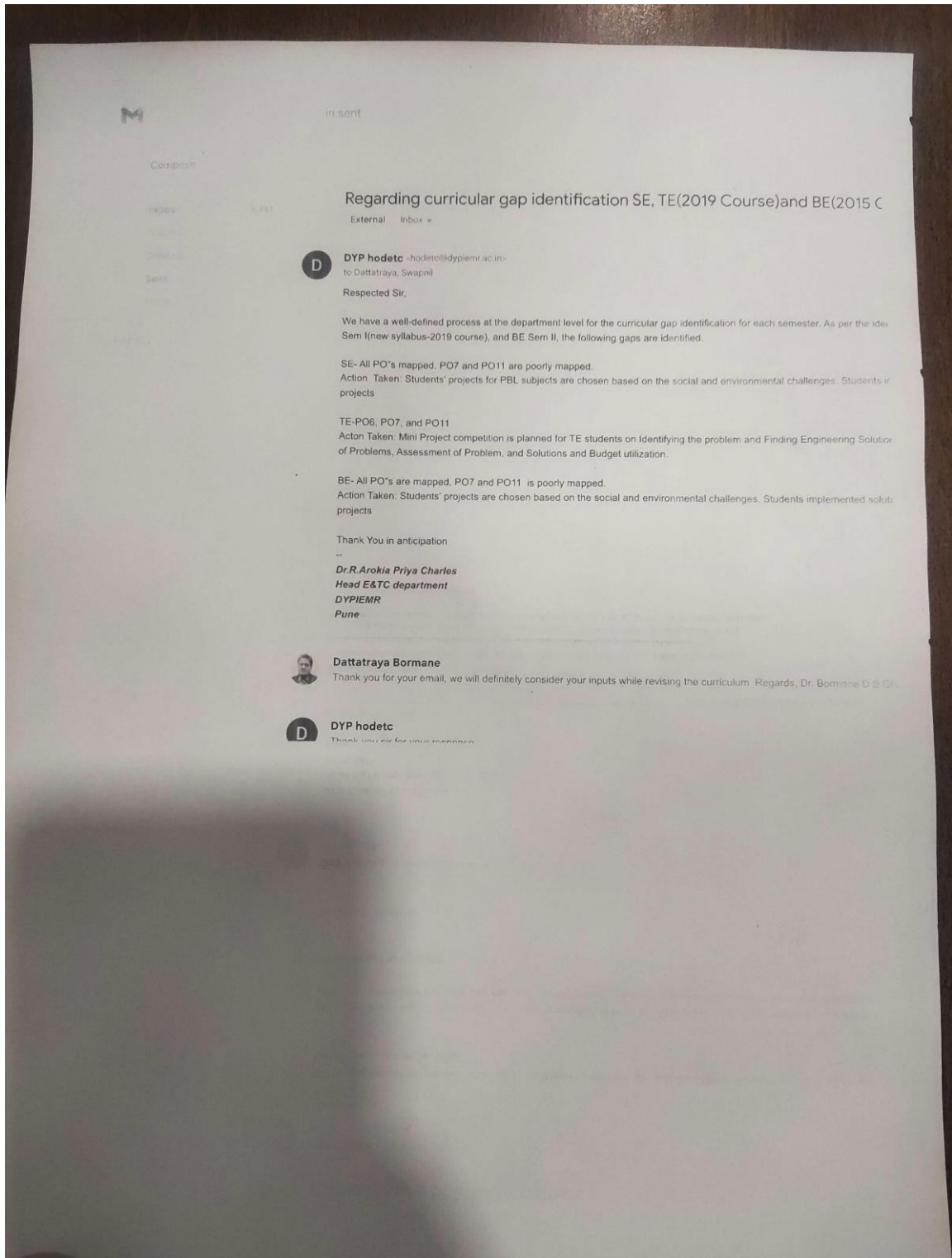
(A.Y- 2021-2022)

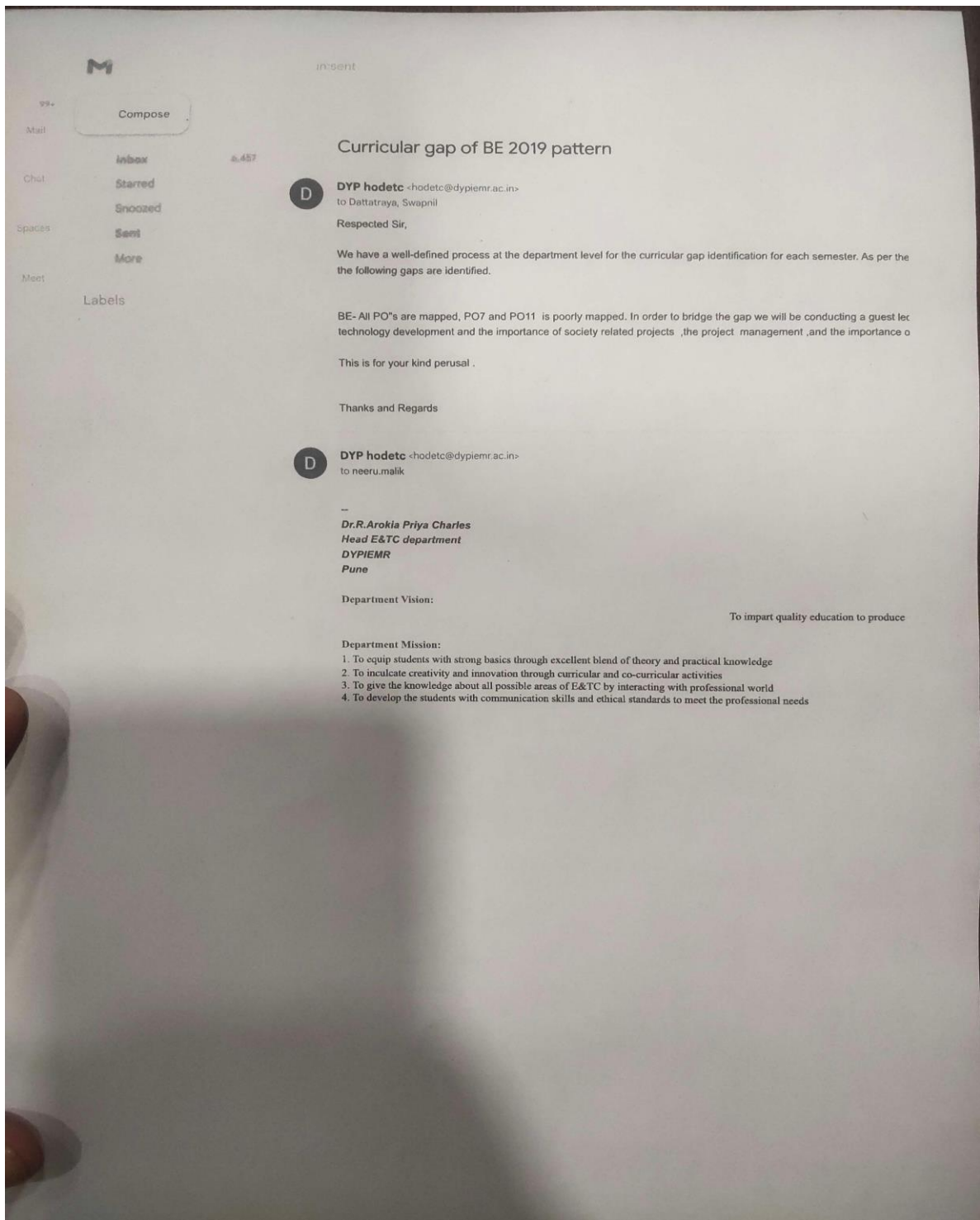
Sr.No	Stakeholder	Feedback Collected
1	Students	Syllabus should based on practical basis
		Industrial Visit and expert lecture from industry person required, virtual visits should me more
		Software's like ANSIS, VHDL,MATLAB, MODFLOW is required in addition languages like R, Python and SQL also included in syllabus
		The efficient online practical platform should be provided.
		Faculty should upload their courses on e-platform like Udemmy and coursera.
2	Teachers	The curriculum should be design a such way that it fulfill industry demand
		For practical's more time should be given in academic
3	Alumni	Give printed manual for submission
		Lectures from Industry experts must be there
		To Improve communication skill, class should be there
		Take add-on courses on new techniques as per industry requirements
4	Parents	Overall improvement should be there
		The softskills workshop should be there for students
		The placement should be there for each student
		Motivation should be there for student to have higher degree




*[Signature]*  
**PRINCIPAL**  
 Dr. D. Y. Patil Institute of Engineering,  
 Management & Research  
 Akurdi, Pune - 411 044

1. Mail to BOS regarding curricular gap of SE, TE, BE (2019 course) of E&TC.





	<b>Dr D Y Patil Pratishthan's</b> <b>Dr. D.Y. Patil Institute of Engineering, Management and Research, Akurdi, Pune</b>	DI No.: <b>ACAD/DI/56B</b>
<b>Academic Year:</b> 2021-22	<b>Report of Event Organized</b>	<b>Revision : 00</b> <b>Dated : 20/11/2019</b>
<b>Term –I</b>	<b>Department of AI&amp;DS and Computer Engineering</b>	<b>Date of Preparation</b> <b>: 09.09.2021</b>

**Dr. D. Y. Patil Pratishthan's**

**DR. D. Y. PATIL INSTITUTE OF ENGINEERING, MANAGEMENT & RESEARCH**

Approved by A.I.C.T.E, New Delhi , Maharashtra State Government, Affiliated to Savitribai Phule Pune University  
Sector No. 29, PCNTDA , Nigidi Pradhikaran, Akurdi, Pune 411044. Phone: 020-27654470, Fax: 020-27656566  
Website : [www.dypiemr.ac.in](http://www.dypiemr.ac.in) Email : [principal.dypiemr@gmail.com](mailto:principal.dypiemr@gmail.com)

**AI&DS and COMPUTER ENGINEERING DEPARTMENT**



## **GUEST LECTURE**

**on**

## **Devops Tool**

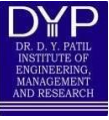
<b>Participants</b>	: TE students of DYPIEMR
<b>Venue</b>	: Online MS Teams Platform
<b>Date</b>	: 08/09/2021
<b>Organizing Team</b>	: Mrs. Sandhya Gundre, Mrs. Ketaki Bhoyar, Mrs. Suvarna Patil, Program Chair, ACM.



## Table of Content

Sr. No	Content	Page No.
1.	Guest Lecture on “Guidance for Higher Studies (M.S.)”	
2.	Appendix	
i.	Notice	
ii.	Invitation to the Guest	
iii.	Attendance Record	
iv.	Feedback Forms	
v.	Analysis of Feedback	
vi.	Letter of Conduction	
vii.	Speakers Feedback	

# 1. Notice

	<b>Dr D Y Patil Pratishthan's Dr. D.Y. Patil Institute of Engineering, Management and Research, Akurdi, Pune</b>	<b>DI No.: ACAD/DI/72</b>
<b>Academic Year: 2021-22</b>	<b>Expert Lecture Notice</b>	<b>Revision : 00 Dated : 20/11/2019</b>
<b>Term – I</b>	<b>Computer Engineering Department</b>	<b>Date of Preparation : 1/09/2021</b>

## EXPERT LECTURE

All TE Computer students are hereby informed that Expert Lecture on “**DevopsTools**” will be conducted in association with ACM, Students Chapter. The Expert Lecture is organized to enable the understanding about the techniques, and best practices to create cleaner, more readable, more efficient code with minimal errors.

**Attendance is mandatory.**

**TOPIC:** “DevopsTools”

**Date & Time:** 08<sup>th</sup> September 2021, 10. 00 am

**Speaker:** Mr. Jayant Nandurkar, Technical Architect , Whirlpool Asia LLP

**Microsoft Teams Link:**

[https://teams.microsoft.com/l/meetup-join/19%3ameeting\\_ODFIN2YyYWEtN2UyMi00NWRhLWI0YjgtNjA5YTg3NTgzNzU2%40thread.v2/0?context=%7b%22Tid%22%3a%223a3cd3f3-9917-40dc-91e0-85146eaf5d55%22%2c%22Oid%22%3a%226fcd1471-1d3a-4e7e-88f2-ea499dd7a7e5%22%7d](https://teams.microsoft.com/l/meetup-join/19%3ameeting_ODFIN2YyYWEtN2UyMi00NWRhLWI0YjgtNjA5YTg3NTgzNzU2%40thread.v2/0?context=%7b%22Tid%22%3a%223a3cd3f3-9917-40dc-91e0-85146eaf5d55%22%2c%22Oid%22%3a%226fcd1471-1d3a-4e7e-88f2-ea499dd7a7e5%22%7d)

Mrs. Sandhya Gundre

Mrs Ketaki Bhoyar

Mrs P.P Shevatekar

Mrs. Suvarna Patil

Mrs. Pallavi Yevale

Guest Lecture Coordinator

ACM Coordinator

HOD Computer

HOD AI-DS

## 2. Objectives

---

- To motivate students for higher studies in foreign countries.
- To aware students about real scenario in foreign countries while doing M.S.

## 3. Information about Speaker

---

Mr. Jayant Nandurkar  
Technical Architect , Whirlpool Asia LLP


## 4. Report

---

**Title:** “DevopsTools”

**Day & Date:** 08/09/2021

**Highlights of the Talk:**

- To demonstrate the current situation in IT industry
  - How to prepare for GRE & TOEFL
  - Living situation in foreign countries while doing M.S.
  - Motivating students to prepare themselves for higher studies
- 

The Session was organized on MS Teams platform for the SE, TE and BE Computer Engineering Students.

**Details of the session:**

The speaker itself is an alumni of DYPIEMR. He opted for higher education (M.S.) immediately after graduation from DYPIEMR, SPPU University, Pune. He shared experience right from preparation of GRE & TOFEL till the complete graduation and working environment there. He also shared his experience regarding spending money on course as well as living expenses. He motivated students for opting higher education and also cleared out their confusions regarding many issues living outside the country. He is working at well known position in the well known industry. At the end of session he gave his contact details to students for any further queries.

## 5. Glimpses of the session

---



10:06



VoWiFi 2

VoLTE 1



Sandhya G started recording



# DevOps

teams.microsoft.com is sharing your screen. [Stop sharing](#) [Hide](#)

Snapshot 1: Glimpse on ongoing session by Jayant Nandurkar

Snapshot 2: Participants attending guest session

## Learning Outcomes

- Students understood current situation in foreign countries.
- Students were motivated to go for higher studies.
- Students understood placement scenario after M.S.

## Attendance Record



meetingAttendanceList (38) - Excel

Ketaki Bhoyar

File Home Insert Page Layout Formulas Data Review View Developer Help Tell me what you want to do

Clipboard Font Alignment Number Styles Cells Editing

Calibri 11

General

Conditional Formatting Table Cell Insert Delete Format AutoSum Fill Sort & Find & Filter Select

113

	A	B	C	D	E	F	G	H	I	J	K	L	M	N	O	P	Q	R	S	T
59	TEA15 Tanmay Gangurde (Guest)	Joined	4/24/2021, 10:04:56 AM																	
60	suyash jamdade	Joined	4/24/2021, 10:04:58 AM																	
61	suyash jamdade	Left	4/24/2021, 10:13:28 AM																	
62	suyash jamdade	Joined	4/24/2021, 10:13:31 AM																	
63	TEB-21KSHITU KANAKE (Guest)	Joined	4/24/2021, 10:05:10 AM																	
64	TEB-21KSHITU KANAKE (Guest)	Left	4/24/2021, 10:05:40 AM																	
65	TEB 76 Ruchira (Guest)	Joined	4/24/2021, 10:05:25 AM																	
66	TEA_27_Sampada Jondhale (Guest)	Joined	4/24/2021, 10:05:40 AM																	
67	Sahil Temgire (Guest)	Joined	4/24/2021, 10:06:02 AM																	
68	Akhay Gaikwad SBCO19183 (Guest)	Joined	4/24/2021, 10:06:02 AM																	
69	TE-B-65 Anjali Vedpathak (Guest)	Joined	4/24/2021, 10:06:05 AM																	
70	TE-B-65 Anjali Vedpathak (Guest)	Left	4/24/2021, 10:08:23 AM																	
71	B_19130_Nandini (Guest)	Joined	4/24/2021, 10:06:19 AM																	
72	TEA 22 Manasi Wagh (Guest)	Joined	4/24/2021, 10:06:46 AM																	
73	TEA34 Rahul Patil (Guest)	Joined	4/24/2021, 10:06:46 AM																	
74	Swarupa Patil (Guest)	Joined	4/24/2021, 10:07:00 AM																	
75	kolageprasads0	Joined	4/24/2021, 10:07:09 AM																	
76	Soham Kulkarni (Guest)	Joined	4/24/2021, 10:07:17 AM																	
77	B,19-Avanti (Guest)	Joined	4/24/2021, 10:07:24 AM																	
78	Vijay ganesh_B15	Joined	4/24/2021, 10:07:25 AM																	
79	Vijay ganesh_B15	Left	4/24/2021, 10:07:51 AM																	
80	19161 B Rohan	Joined	4/24/2021, 10:07:37 AM																	
81	19161 B Rohan	Left	4/24/2021, 10:08:54 AM																	
82	Vaishnavi Sakunde (Guest)	Joined	4/24/2021, 10:07:38 AM																	
83	Vaishnavi Sakunde (Guest)	Left	4/24/2021, 10:08:15 AM																	
84	B_19167_Mahima_Sahu	Joined	4/24/2021, 10:07:44 AM																	
85	B-58 Jyoti Murge (Guest)	Joined	4/24/2021, 10:07:50 AM																	
86	B-58 Jyoti Murge (Guest)	Left	4/24/2021, 10:10:47 AM																	
87	SACO19186_Sapana Pande (Guest)	Joined	4/24/2021, 10:07:52 AM																	

meetingAttendanceList (38)

Type here to search

3:43 PM 4/26/2021

meetingAttendanceList (38) - Excel

Ketaki Bhoyar

File Home Insert Page Layout Formulas Data Review View Developer Help Tell me what you want to do

Clipboard Font Alignment Number Styles Cells Editing

Calibri 11

General

Conditional Formatting Table Cell Insert Delete Format AutoSum Fill Sort & Find & Filter Select

113

	A	B	C	D	E	F	G	H	I	J	K	L	M	N	O	P	Q	R	S	T
88	SE_B_19181_Kadam_Vaishnavi (Guest)	Joined	4/24/2021, 10:07:58 AM																	
89	A_19165_Bhame Vijay Balasaheb (Guest)	Joined	4/24/2021, 10:08:06 AM																	
90	Satyam Raut (Guest)	Joined	4/24/2021, 10:08:35 AM																	
91	SBCO19182 Aarti (Guest)	Joined	4/24/2021, 10:08:53 AM																	
92	SBCO19182 Aarti (Guest)	Left	4/24/2021, 10:14:28 AM																	
93	SACO19113_NISHANT (Guest)	Joined	4/24/2021, 10:09:06 AM																	
94	B 19154 Krishnesh (Guest)	Joined	4/24/2021, 10:09:10 AM																	
95	TE-A-18143-DEEPESH DESALE (Guest)	Joined	4/24/2021, 10:09:12 AM																	
96	BE A29 Saurabh Pawar (Guest)	Joined	4/24/2021, 10:09:18 AM																	
97	19116 A Wajid (Guest)	Joined	4/24/2021, 10:09:37 AM																	
98	TE_A_54_Tamvi Dube	Joined	4/24/2021, 10:09:44 AM																	
99	SE B 19101 gayatri (Guest)	Joined	4/24/2021, 10:09:45 AM																	
100	SE B 19101 gayatri (Guest)	Left	4/24/2021, 10:10:56 AM																	
101	SBCO19188 Omkar (Guest)	Joined	4/24/2021, 10:09:59 AM																	
102	Devika_B_89 (Guest)	Joined	4/24/2021, 10:11:16 AM																	
103	INSTRU SE 263 Patil Gaurav	Joined	4/24/2021, 10:11:17 AM																	
104	SBCO19108 (Guest)	Joined	4/24/2021, 10:11:35 AM																	
105	SCOA191031 Prasad Shewale (Guest)	Joined	4/24/2021, 10:11:49 AM																	
106	TEA 25 Reeva Prasad (Guest)	Joined	4/24/2021, 10:13:10 AM																	
107	TE B 01 Neha Kale (Guest)	Joined	4/24/2021, 10:13:31 AM																	
108	Vishal Patil	Joined	4/24/2021, 10:14:05 AM																	
109	Vishal Patil	Left	4/24/2021, 10:14:46 AM																	
110	SBCO19182 Aarti (Guest)	Joined	4/24/2021, 10:15:08 AM																	
111	TE_B_12 Sanket (Guest)	Joined	4/24/2021, 10:15:29 AM																	
112	TEB 16 Avaneesh Yadav	Joined	4/24/2021, 10:16:01 AM																	
113	Sayali Mundlik_DYPIEMR_BE	Joined	4/24/2021, 10:16:03 AM																	
114	Vishwajeet Rahul Jadhav	Joined	4/24/2021, 10:16:12 AM																	
115	Vishwajeet Rahul Jadhav	Left	4/24/2021, 10:16:50 AM																	
116	SBCO19186 Rohit (Guest)	Joined	4/24/2021, 10:16:23 AM																	

meetingAttendanceList (38)

Type here to search

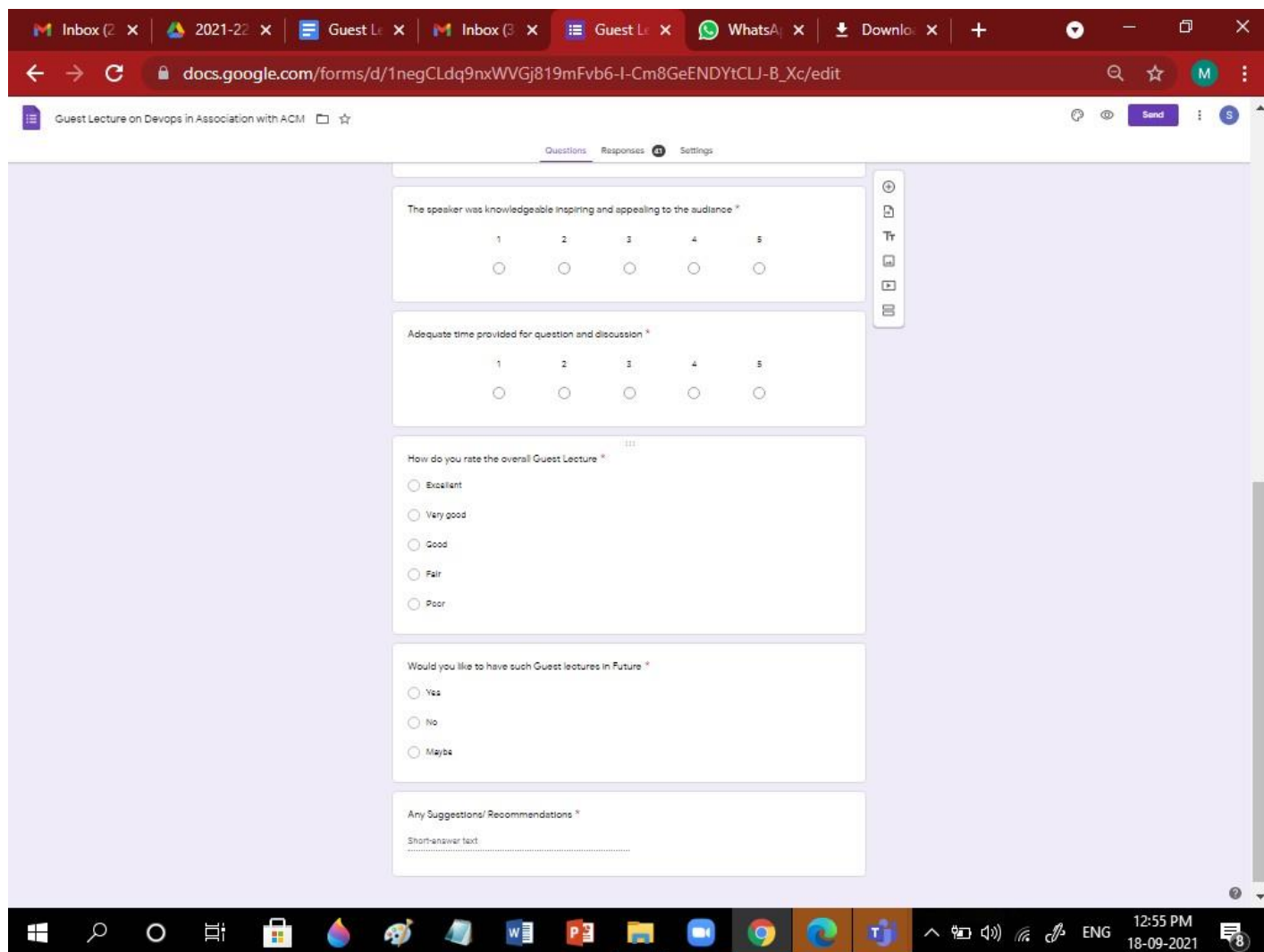
3:44 PM 4/26/2021

# Feedback Form and Analysis

The screenshot shows a Google Forms interface for a feedback form titled "Guest Lecture on Devops in Association with ACM". The form is currently in edit mode, as indicated by the "Questions" tab being active. The form contains the following questions:

- Form description:** A text field for the form description.
- Email:** A short-answer text question.
- Name Of Participant:** A short-answer text question.
- Class & Division:** A multiple-choice question with two options: TE A and TE B.
- The Lecture Has met my expectation:** A scale question with five radio buttons labeled 1, 2, 3, 4, and 5.
- The Lecture was Well Organised:** A scale question with five radio buttons labeled Strongly disagree, Disagree, Neutral, Agree, and Strongly agree.

The form is displayed on a web browser with multiple tabs open, including "Inbox (2)", "2021-22", "Guest L...", "Inbox (3)", "Guest L...", "WhatsApp", and "Downlo...". The browser address bar shows the URL "docs.google.com/forms/d/1negCLdq9nxWVGj819mFvb6-I-Cm8GeENDYtCLJ-B\_Xc/edit". The Windows taskbar at the bottom shows the time as 12:55 PM on 18-09-2021.



Inbox (2)

2021-22

Guest L

Inbox (3)

Guest L

WhatsApp

Downlo

+

docs.google.com/forms/d/1negCLdq9nxWVGj819mFvb6-I-Cm8GeENDYtCLJ-B\_Xc/edit#responses

Search

Star

M

Guest Lecture on Devops in Association with ACM

Send

S

Questions

Responses 11

Settings

shubham45sonar@gmail.com

ghyargaurav90@gmail.com

Name Of Participant

41 responses

Vedant Munj

Bharat choudhary

Aaryan Joshi

Gayatri. R. Pillai

Vedant Bhosale

Nachiket Gulve

Shamali Sandip Deshmukh

Bhama Vijay Balasaneb

Wajid Sayyad

Class & Division

41 responses

53.7%

46.3%

10 A

10 B

The Lecture Has met my expectation

41 responses

20

15

10

18 (43.9%)

18 (46.3%)

Windows

Search

Calendar

Task View

File Explorer

Microsoft Word

PowerPoint

Google Chrome

Edge

Teams

System Tray

12:56 PM

18-09-2021

ENG

8

Inbox (2)2021-22Guest LInbox (3)Guest LWhatsAppDownload

docs.google.com/forms/d/1negCLdq9nxWVGj819mFvb6-I-Cm8GeENDYtCLJ-B\_Xc/edit#responses

Guest Lecture on Devops in Association with ACM

QuestionsResponsesSettings

41 responses

46.3%

38%

15.8%

0%

Strongly disagree

Disagree

Neutral

Agree

Strongly agree

The speaker was knowledgeable inspiring and appealing to the audience

41 responses

1 (2.4%)

0 (0%)

3 (7.3%)

18 (43.9%)

21 (51.2%)

1

2

3

4

5

Adequate time provided for question and discussion

41 responses

1 (2.4%)

1 (2.4%)

2 (4.9%)

18 (43.9%)

19 (46.3%)

1

2

3

4

5

How do you rate the overall Guest Lecture

41 responses

41.0%

Excellent

Very good

Good

WindowsTaskbar

12:56 PM18-09-2021

A blue triangle pointing upwards, located at the bottom right of the page.



Inbox (2)2021-22Guest LInbox (3)Guest LWhatsAppDownload+

docs.google.com/forms/d/1negCLdq9nxWVGj819mFvb6-I-Cm8GeENDYtCLJ-B\_Xc/edit#responses

Guest Lecture on Devops in Association with ACM

QuestionsResponses (1)Settings

How do you rate the overall Guest Lecture

21 responses

41.9%

53.7%

Excellent

Very good

Good

Fair

Poor

Would you like to have such Guest lectures in Future

21 responses

97.6%

Yes

No

Maybe

Any Suggestions/ Recommendations

21 responses

10 (47.6%)

4 (19%)

3 (14.3%)

2 (9.5%)

1 (4.8%)

0 (0%)

Excellent

Very good

Good

Fair

Poor

No suggestion

Good but more time

None as of now. I ...

We need more st...

WindowsTaskbar

12:56 PM18-09-2021

# Letter of Conduction

**Dr. D. Y. Patil Pratishthan's**



**DR. D. Y. PATIL INSTITUTE OF ENGINEERING, MANAGEMENT & RESEARCH**

Approved by A.I.C.T.E, New Delhi, Maharashtra State Government, Affiliated to Savitribai Phule Pune University  
Sector No. 29, PCNTDA, Nigdi Pradhikaran, Akurdi, Pune 411044. Phone: 020-27654470, Fax: 020-27656566

Website: [www.dypiemr.ac.in](http://www.dypiemr.ac.in) Email: [principal.dypiemr@gmail.com](mailto:principal.dypiemr@gmail.com)

**Department of Computer Engineering**

Date: 11/09/2021

To

Mr Raju Masand  
Module Lead - FIX protocol,  
FinIQ Consulting India Pvt. Ltd.

Respected Sir

On behalf of the Computer Engineering Department, Dr. D. Y. Patil Institute of Engineering, Management and Research, I would like to extend our heartfelt gratitude for conducting the Expert Lecture on "Coding Standards in IT" on 11th September 2021. It was an honor for DYPIEMR to have you as one of our experts.

Feedback from the audience was extremely positive. All responses were enthusiastic about the content and the overall quality of the lecture. We believe that the knowledge you have shared with students will help immensely in their technical development.

We appreciate you for sharing your time, talent, and expertise with us.

Once again, Thank you for joining, and all the best for your professional endeavors.

Yours sincerely

Mrs. P. P. Shevatekar

HOD Computer

Dept. of Computer Engineering,  
DYPIEMR

# Letter of Appreciation



Dr. D. Y. Patil Institute of Engineering, Management  
and Research, Akurdi, Pune – 44  
Department of Computer Engineering

Year : 2020-21

Date: 24/04/2021

## Appreciation Letter

Dear Mr. Shubham Kale,

On behalf of Computer Engineering Department, DYPIEMR, I would like to extend our heartfelt gratitude for your participation as a speaker on "Higher studies Guidance (M.S.)" on 24<sup>th</sup> April 2021. It was an honor to have you as our speaker.

We believe the guidance and experience you shared will help immensely to our students participants in preparation for higher studies.

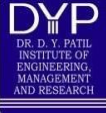
Thank you once again for sparing some time for us from your busy schedule. We look forward to your participation on future events.

Yours sincerely,

Prof. P.P. Shevatekar  
H.O.D Computer Engineering  
DYPIEMR, Akurdi


**Report Prepared by**

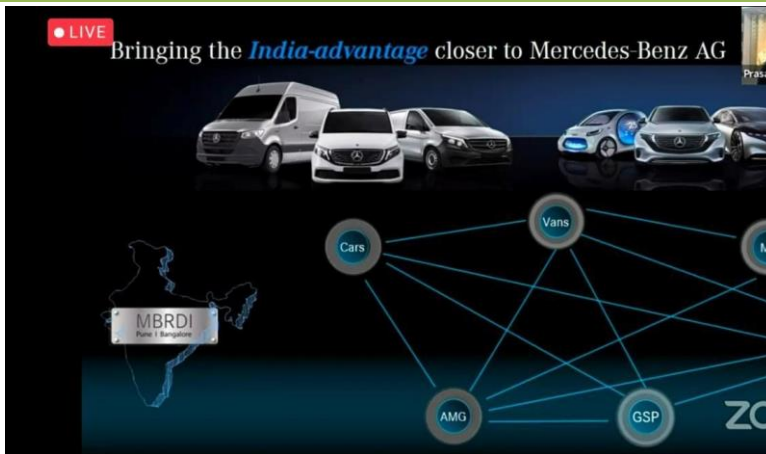
**Mrs. Sandhya Gundre  
Mrs. Suvarna Patil  
Mrs. Shivganga Gavhane  
Mrs. Ketaki Bhoyar  
Asst. Prof,  
DYPIEMR**

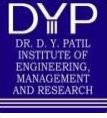
	<b>Dr D Y Patil Pratishthan's Dr. D.Y. Patil Institute of Engineering, Management and Research, Akurdi, Pune</b>	DI No.: <b>ACAD/DI/56 A</b>
<b>Academic Year:</b> 2021-22	<b>Industrial Visit One Page Report</b>	<b>Revision : 00 Dated : 20/11/2019</b>
<b>Term – I</b>	<b>Computer Engineering Department</b>	<b>Date of Preparation : 19/12/2021</b>

### Activity Report

<b>Activity</b>	<b>Industrial Visit</b>
<b>Department</b>	<b>Computer Engineering</b>
<b>Title</b>	Virtual Industrial Visit- IIT Bombay Virtual Industry Visits 2021-22 with top companies like <b>Mercedes Benz, Godrej &amp; Boyce.</b>
<b>Date</b>	19/12/21
<b>Name of Speaker</b>	Company representatives. Mr.Prasanna Gonugatla
<b>Objectives</b>	<ul style="list-style-type: none"> <li>• How the companies are working?</li> <li>• Which area company is looking more productive performance?</li> </ul>
<b>Brief Description</b>	<p>Mercedes Benz Research &amp; Development India, is the largest R&amp;D center outside Germany for Mercedes Benz AG. MBRDI taps into India's engineering and IT talent to develop innovative products both locally and globally. Here at MBRDI, with a mix of interdisciplinary team players, working on the future of mobility, we focus on topics ranging from computer-aided design and simulations (CAD and CAE) for powertrain, chassis and exteriors to embedded systems, telematics and on developing a host of IT applications and tools. The virtual tour will give you a glimpse of the Digital process chain that enables digital product development and Manufacturing.</p> <p>Godrej &amp; Boyce Mfg. Co. Ltd., the flagship company of the Godrej Group, has played a key role in India's economic history by driving excellence in design and manufacturing, and delivering sustainable value for its stakeholders and communities. Godrej Process Equipment, a strategic business unit of Godrej &amp; Boyce is one of the leading global manufacturers of critical Process Equipment for Oil &amp; Gas, Fertilizer &amp; Chemicals and Power Sector. Having manufactured</p>

	<b>Dr D Y Patil Pratishthan's</b> <b>Dr. D.Y. Patil Institute of Engineering, Management and Research, Akurdi, Pune</b>	DI No.: <b>ACAD/DI/56 A</b>
<b>Academic Year:</b> 2021-22	<b>Industrial Visit One Page Report</b>	<b>Revision : 00</b> <b>Dated : 20/11/2019</b>
<b>Term – I</b>	<b>Computer Engineering Department</b>	<b>Date of Preparation :</b> 19/12/2021

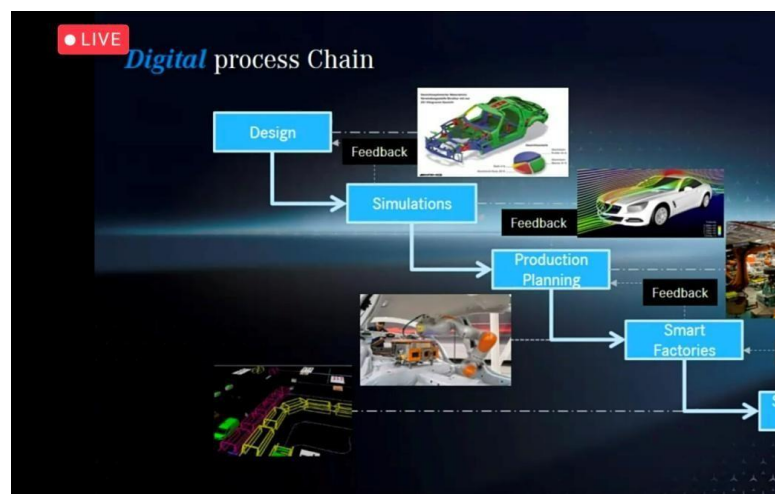
	<p>more than 3000 equipment and catered to more than 400 global customers in 35 countries, Godrej Process Equipment has world class manufacturing facilities in Mumbai &amp; Dahej.</p>
<b>Outcome</b>	<p>Students will be able to</p> <ul style="list-style-type: none"> <li>• Understand the working of company plants and their production strategies.</li> <li>• Information regarding designs.</li> </ul>
<b>CO/PO/PSO Mapping</b>	PO1, PO6, PO7
<b>Student Benefited</b>	SE ( A & B),TE(A & B),BE(A & B) Computer 233 students
<b>Glimpses (Mercedez &amp; Godrej)</b>	

	<b>Dr D Y Patil Pratishthan's</b> <b>Dr. D.Y. Patil Institute of Engineering, Management and Research, Akurdi, Pune</b>	DI No.: <b>ACAD/DI/56 A</b>
<b>Academic Year:</b> 2021-22	<b>Industrial Visit One Page Report</b>	<b>Revision : 00</b> <b>Dated : 20/11/2019</b>
<b>Term – I</b>	<b>Computer Engineering Department</b>	<b>Date of Preparation : 19/12/2021</b>

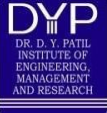
**LIVE** Leveraging *Digital* India


- R&D** India continues to be a chosen *R&D destination* with the number of GICs being setup exhibiting a steady growth YoY
- Global** A total of *1,150* multinationals have *global in-house centers* who employ close 0.8 million people
- AI** India ranks among top 3 in terms of niche skills such as : *ML/AI, Analytics, UI*
- Academia** The *academia system in India* provides GIC's an opportunity to engage in struc
- Indian GICs** Indian GICs are enabling *Digital Transformation* and continue to leverage Dig in India. GIC's in India today focus on :
  - IP Creation
  - Engineering Ownership
  - Competency Building
  - Strengthen Eco-system Connec


Courtesy : Zinnov

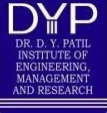





	<b>Dr D Y Patil Pratishthan's</b> <b>Dr. D.Y. Patil Institute of Engineering, Management and Research, Akurdi, Pune</b>	DI No.: <b>ACAD/DI/56 A</b>
<b>Academic Year:</b> 2021-22	<b>Industrial Visit One Page Report</b>	<b>Revision : 00</b> <b>Dated : 20/11/2019</b>
<b>Term – I</b>	<b>Computer Engineering Department</b>	<b>Date of Preparation :</b> 19/12/2021






	<b>Dr D Y Patil Pratishthan's</b> <b>Dr. D.Y. Patil Institute of Engineering, Management and Research, Akurdi, Pune</b>	DI No.: <b>ACAD/DI/56 A</b>
<b>Academic Year:</b> 2021-22	<b>Industrial Visit One Page Report</b>	<b>Revision : 00</b> <b>Dated : 20/11/2019</b>
<b>Term – I</b>	<b>Computer Engineering Department</b>	<b>Date of Preparation :</b> 19/12/2021



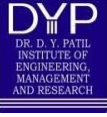
### MANUFACTURING..... Process flow



```

graph LR
    A[DISHED END CAP FORMING] --> B[PLATE MARKING & CUTTING]
    B --> C[PLATE ROLLING]
    C --> D[LONGITUDINAL SEAM WELDING & NDT]
    D --> E[CIRCULAR]
    E --> F[NOZZLE C]
    F --> G[STRESS RELIEVING OF THE EQUIPMENT]
    G --> H[FINAL NDT & HYDRO TEST]
    H --> I[READINESS FOR SHIPMENT]
  
```




	<b>Dr D Y Patil Pratishthan's</b> <b>Dr. D.Y. Patil Institute of Engineering, Management and Research, Akurdi, Pune</b>	DI No.: <b>ACAD/DI/56 A</b>
<b>Academic Year:</b> 2021-22	<b>Industrial Visit One Page Report</b>	<b>Revision : 00</b> <b>Dated : 20/11/2019</b>
<b>Term – I</b>	<b>Computer Engineering Department</b>	<b>Date of Preparation :</b> 19/12/2021



Dr. Amol Ramrao Dhakne  
Ms.Raji Ajith Panickar  
Mr.Shivaji Vasekar  
(Faculty Coordinator)

Mrs P.P Shevatekar  
(Head of Department)


4. Virtual Lab report of E&TC department

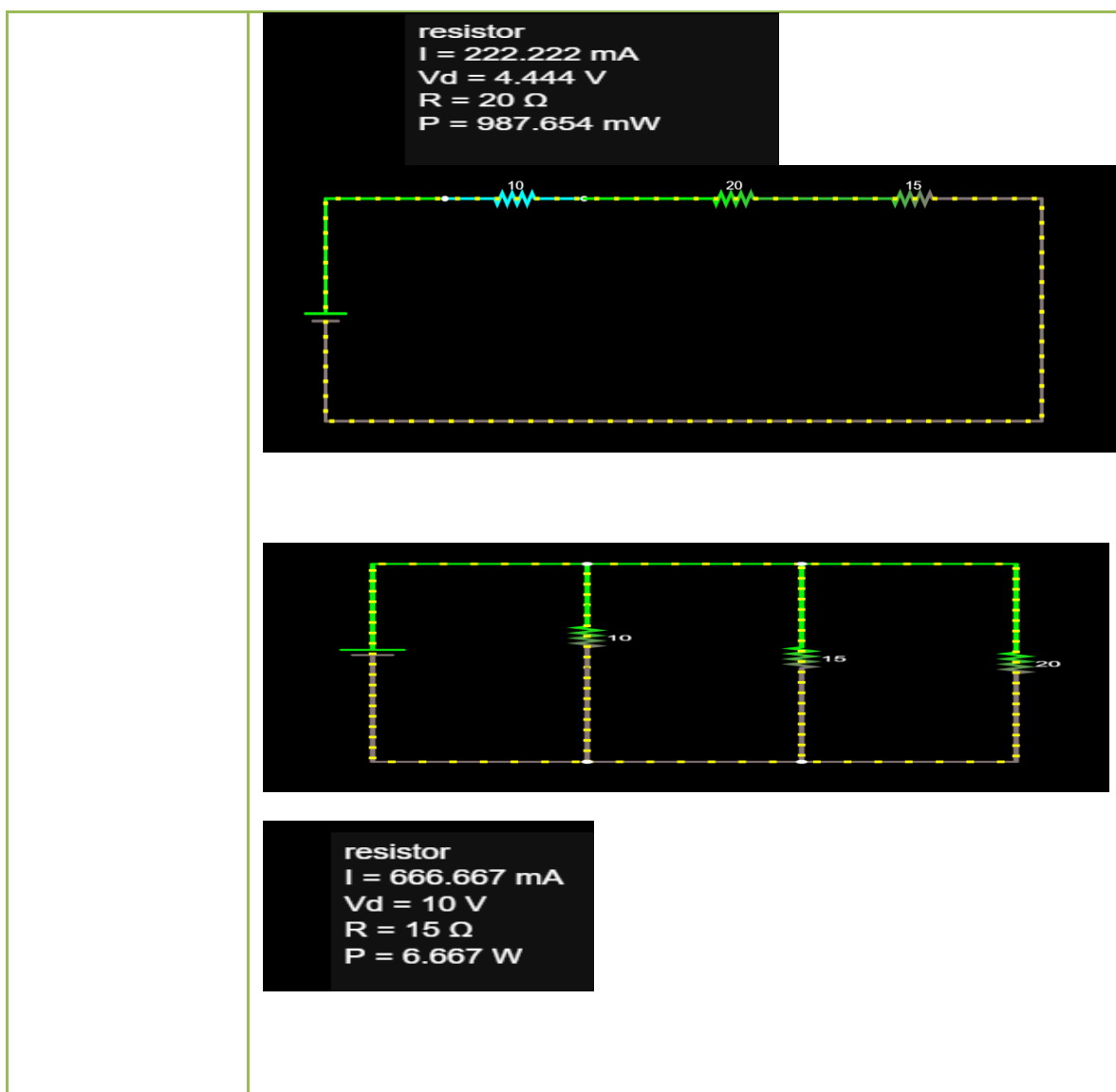
	<b>Dr D Y Patil Pratishthan's Dr. D.Y. Patil Institute of Engineering, Management and Research, Akurdi, Pune</b>	DI No.: <b>ACAD/DI/56 A</b>
<b>Academic Year:</b> 2021-22	<b>One Page Activity Report</b>	<b>Revision : 00 Dated : 20/11/2019</b>
<b>Term – I</b>	<b>Department of Electronics and Telecommunication Engineering</b>	<b>Date of Preparation : 14/9/21</b>

### Activity Report

<b>Activity</b>	<b>Virtual Lab</b>
<b>Department</b>	<b>Electronics and Telecommunication Engineering</b>

<b>Title</b>	Virtual Lab on Electrical Circuits
<b>Date</b>	13/9/2021 and 14/09/2021
<b>Name of Speaker</b>	-
<b>Objectives</b>	Understand the practical knowledge about Electrical Circuits using Virtual lab
<b>Brief Description</b>	Practice on Virtual Lab for Electrical Circuits subject was conducted on 13/9/21 and 14/9/21 at 1.40 pm. Here we performed One experiments ie KCL, KVL and Ohms Law through virtual lab, where we calculated and observed 1. Currents through various given branches. 2. Voltages across the given branches. 3. Power absorbed or delivered by a given component.
<b>Outcome</b>	Understood the Ohms Law, KCL and KVL
<b>CO/PO/PSO Mapping</b>	CO1
<b>Student Benefited</b>	23
<b>Glimpses</b>	

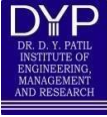
	<b>Dr D Y Patil Pratishthan's</b> <b>Dr. D.Y. Patil Institute of Engineering, Management and Research, Akurdi, Pune</b>	DI No.: <b>ACAD/DI/56 A</b>
<b>Academic Year:</b> 2021-22	<b>One Page Activity Report</b>	<b>Revision : 00</b> <b>Dated : 20/11/2019</b>
<b>Term – I</b>	<b>Department of Electronics and Telecommunication Engineering</b>	<b>Date of Preparation : 14/9/21</b>



**Mrs. Munmun Kakkar**  
Faculty Coordinator

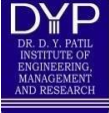
**Dr. Priya Charles**  
Head of Department

## 5. Summary of Add on courses of E&TC department of semester I

	<b>Dr D Y Patil Pratishthan's Dr. D.Y. Patil Institute of Engineering, Management and Research, Akurdi, Pune</b>		<b>DI No.: ACAD/DI/57</b>
<b>Academic Year:</b> 2021-22	<b>Summary of Add on Course</b>		<b>Revision : 00 Dated : 20/11/2019</b>
<b>Term – I</b>	<b>Department : Electronics And Telecommunication Engineering</b>		<b>Date of Preparation : 31/12/21</b>

**A**  
**Report On**  
**Add On Course**



	<b>Dr D Y Patil Pratishthan's</b> <b>Dr. D.Y. Patil Institute of Engineering, Management and Research, Akurdi, Pune</b>	DI No.: <b>ACAD/DI/57</b>
<b>Academic Year:</b> 2021-22	<b>Summary of Add on Course</b>	<b>Revision : 00</b> <b>Dated : 20/11/2019</b>
<b>Term – I</b>	<b>Department : Electronics And Telecommunication Engineering</b>	<b>Date of Preparation : 31/12/21</b>

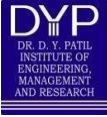
**Academic**  
**Year: 2021-22**

<b>Sr. No.</b>	<b>Course Title</b>	<b>Year/ Branch./ Div.</b>	<b>Duration of Course (Hrs.)</b>	<b>Total No. of Students</b>	<b>Training Agency</b>
1	C/C++	S.E.	30 hrs	79	VAP

**Training Coordinator**  
**Mr. Lokesh Giripunje**

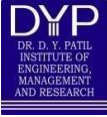
**Head of Department**  
**Dr. Priya Charles**

6. Summary of Add on courses of E&TC department of semester II

	<b>Dr D Y Patil Pratishthan's Dr. D.Y. Patil Institute of Engineering, Management and Research, Akurdi, Pune</b>		<b>DI No.: ACAD/DI/57</b>
<b>Academic Year:</b> 2021-22	<b>Summary of Add on Course</b>		<b>Revision : 00 Dated : 20/11/2019</b>
<b>Term – II</b>	<b>Department : Electronics And Telecommunication Engineering</b>		<b>Date of Preparation : 25/04/22</b>

**A**  
**Report On**  
**Add On Course**



	<b>Dr D Y Patil Pratishthan's</b> <b>Dr. D.Y. Patil Institute of Engineering, Management and</b> <b>Research, Akurdi, Pune</b>		DI No.: <b>ACAD/DI/57</b>
<b>Academic Year:</b> 2021-22	<b>Summary of Add on Course</b>		<b>Revision : 00</b> <b>Dated : 20/11/2019</b>
<b>Term – II</b>	<b>Department : Electronics And Telecommunication</b> <b>Engineering</b>		<b>Date of Preparation</b> <b>: 25/04/22</b>

**Academic Year: 2021-22**

<b>Sr. No.</b>	<b>Course Title</b>	<b>Year/ Branch./ Div.</b>	<b>Duration of Course (Hrs.)</b>	<b>Total No. of Students</b>	<b>Training Agency</b>
1	<b>Aptitude and GDPI Training</b>	T.E.	80 hrs.	25	<b>Campus Credentials</b>
2	PYTHON	S.E.	30 hrs	79	VAP

**Training Coordinator**  
**Mr. Lokesh Giripunje**

**Head of Department**  
**Dr.. Priya Charles**

7. Printed lab manual of DBMS(E&TC department) provided to students



**DR. D. Y. PATIL INSTITUTE OF ENGINEERING, MANAGEMENT AND  
RESEARCH, AKURDI, PUNE-44**

**Department of  
Electronics & Telecommunication**

**2021-2022**

**LAB MANUAL**

**Subject – Data Base Management Lab**

**Subject code: 304187**

**Class – TE**



## Program Outcomes

### Engineering Graduates will be able to:

1. **Engineering Knowledge:** Apply the knowledge of mathematics, science, engineering fundamentals and E&TC engineering specialization to the solution of complex E&TC engineering problems
  - PO1.a – Apply the knowledge of mathematics
  - PO1.b – Apply the knowledge of science
  - PO1.c – Apply the knowledge of engineering fundamentals
2. **Problem Analysis:** Identify and analyze complex engineering problems using first principles of mathematics, natural sciences, and E&TC engineering science
  - PO2.a – Identify the engineering problem
  - PO2.b – analyze the engineering problem
  - PO3.c – reaching the conclusion for the problem
3. **Design /development of Solution:** Design solutions for E&TC engineering problems and design system components for real life
  - PO3.a – Design solution for engineering problems
  - PO3.b – Design system components for real life solution
4. **Conduct investigations of complex problems:** Use Engineering knowledge for analysis and interpretation of data, and synthesis of the information to provide valid conclusions.
  - PO4.a – analysis of data
  - PO4.b – Interpretation of data
  - PO4.c – synthesis of data for valid conclusion
5. **Modern tool Usage:** select and apply appropriate techniques, using IT tools to model E&TC engineering problems with an understanding of the limitations.
  - PO5.a – Select and apply appropriate technique
  - PO5.b – knowledge of various IT tools
  - PO5.b – Use IT tools to model E&TC engineering problems
6. **The engineer and society:** Apply reasoning informed by the contextual knowledge to assess societal, health, safety, legal and cultural issues and the consequent responsibilities relevant to the professional E&TC engineering practice.
  - PO6.a – ability to identify the problem
  - PO6.b – assess the problem
  - PO6.c – apply the engineering solution
7. **Environment and sustainability:** Understand the impact of the E&TC engineering solutions in societal and environmental contexts, and demonstrate the knowledge of, and need for sustainable development.
  - PO7.a – understand the impact of E&TC engineering solutions
  - PO7.b – demonstrate the knowledge for sustainable development
8. **Ethics:** Apply ethical principles and commit to professional ethics and responsibilities and norms of the E&TC engineering practice.
  - PO8.a – have awareness of ethical principles
  - PO8.b – be committed to professional ethics
9. **Individual and team work:** Function effectively as an individual , and as a member or leader in a team
  - PO9.a – ability to function effectively as an individual
  - PO9.b – ability to function as a leader in a team
10. **Communication:** Communicate effectively ,comprehend and write effective reports and make effective presentations
  - PO10.a – ability to communicate effectively
  - PO10.b – ability to comprehend and write effective reports
  - PO10.c – ability to make effective presentations
11. **Project management and finance:** Have knowledge and understanding of the E&TC engineering and management principles and apply these to one's own work, as a member and leader in a team, to manage projects
  - PO11.a – Ability to have the knowledge and understanding of Engineering and Management principles
  - PO11.b – apply managerial skills effectively as a leader
  - PO11.c – Apply the E&TC engineering skills as a team member
12. **Life-long learning:** Ability of self-education and understand the technological changes
  - PO12.a – Inculcate the habit of self-learning and understanding
  - PO12.b – ability to adapt to technological changes

**Vision:**

To impart quality education to produce competent E&TC Engineers

**Mission:**

1. To equip students with strong basics through excellent blend of theory and practical knowledge
2. To inculcate creativity and innovation through curricular and co-curricular activities
3. To give the knowledge about all possible areas of E&TC by interacting with professional world
4. To develop the students with communication skills and ethical standards to meet the professional needs

**PSOs:**

The E&TC engineering graduates should be able to

- 1) Apply principles of Electronics and communication , digital systems, signal processing, software programming in the field of Embedded, Telecommunication & Software services for real world applications
- 2) Comprehend the technological advancements, demonstrate the proficiency in the usage of engineering tools to analyze and design systems for variety of applications.
- 3) Demonstrate professional ethics , apply communication skills for successful career and higher studies

**PEOs:**

- 1) The graduate shall utilize the basic knowledge to address the Engineering problems
- 2) The graduate shall attain the qualities of professional leadership with ethical and moral standards
- 3) The graduate shall develop their capabilities for lifelong learning throughout their professional career and higher education
- 4) The graduate shall explore engineering capabilities through creativity and innovation.

**Course Outcomes:**

University course Code	SAR course code	COURSE OUTCOMES
<b>304187</b>	C304.1	Ability to learn and understand various DDL queries like create, drop, truncate, DML queries like insert, select, update, delete. all types of Join and Sub-Query, and to demonstrate creating and dropping SQL objects like table, view, sequence, index etc.
	C304.2	Ability to learn and understand PL/SQL, to implement all types of Cursors(All types: Implicit, Explicit, Cursor FOR Loop, Parameterized Cursor), Stored Procedure and function, for writing Database Triggers(Row level and Statement level triggers, Before and After Triggers).
	C304.3	Ability to Implement MYSQL/Oracle database connectivity with PHP/python/Java Implement Database navigation operations (add, delete, edit,) using ODBC/JDBC.
	C304.4	Ability to design and develop database application as a mini project

**CO and PO Mapping for DBMS Lab:**

CO	PO1	PO2	PO3	PO4	PO5	PO6	PO7	PO8	PO9	PO10	PO11	PSO1	PSO2	PSO3	PO12
<b>C304.1</b>	2	1	1	1	1	1	-	-	2	2	2	2	2	2	2
<b>C304.2</b>	1	2	2	1	1	1	-	-	2	2	2	2	2	2	2
<b>C304.3</b>	1	1	2	1	1	1	-	-	2	2	2	2	1	1	2
<b>C304.4</b>	1	1	2	1	1	1	-	-	2	2	2	2	1	1	2

**Savitribai Phule Pune University, Pune**  
**T.E. (Electronics & Telecommunication Engineering) 2019 Course**  
 (With effect from Academic Year 2021-22)

**Semester-V**

Course Code	Course Name	Teaching Scheme (Hours/Week)			Examination Scheme and Marks						Credit			
		Theory	Practical	Tutorial	In-Sem	End-Sem	TW	PR	OR	Total	TH	PR	TUT	Total
304181	Digital Communication	03	-	-	30	70	-	-	-	100	03	-	-	03
304182	Electromagnetic Field Theory	03	-	01	30	70	25	-	-	125	03	-	01	04
304183	Database Management	03	-	-	30	70	-	-	-	100	03	-	-	03
304184	Microcontrollers	03	-	-	30	70	-	-	-	100	03	-	-	03
304185	Elective - I	03	-	-	30	70	-	-	-	100	03	-	-	03
304186	Digital Communication Lab	-	02	-	-	-	-	50	-	50	-	01	-	01
304187	Database Management Lab	-	02	-	-	-	-	-	25	25	-	01	-	01
304188	Microcontroller Lab	-	02	-	-	-	-	50	-	50	-	01	-	01
304189	Elective I Lab	-	02	-	-	-	-	25	-	25	-	01	-	01
304190	Skill Development	-	02	-	-	-	25	-	-	25	-	01	-	01
304191A	Mandatory Audit Course 5 &	-	-	-	-	-	-	-	-	-	-	-	-	-
<b>Total</b>		<b>15</b>	<b>10</b>	<b>01</b>	<b>150</b>	<b>350</b>	<b>50</b>	<b>125</b>	<b>25</b>	<b>700</b>	<b>-</b>			
<b>Total Credit</b>											<b>15</b>	<b>05</b>	<b>01</b>	<b>21</b>

### **Guidelines for Student's Lab Journal**

- The laboratory assignments/experiments are to be submitted by student in the form of journal.
- Journal consists of Certificate, table of contents, and handwritten write-up for each experiment.
- Each experiment should consist of:
  - ✓ Assignment No
  - ✓ Title of Assignment
  - ✓ Date of Performance
  - ✓ Date of Submission
  - ✓ Aims & Objectives
  - ✓ Theory
  - ✓ Description of data used
  - ✓ Results
  - ✓ Conclusion.

### **Guidelines for Lab Assessment:**

- Continuous assessment of laboratory work is done based on overall performance.
- Each lab assignment/ experiment assessment will assign grade / marks based on parameters with appropriate weightage.
- Suggested parameters for overall assessment as well as each lab assignment / experiment assessment include:
  - ✓ Timely completion.
  - ✓ Performance.
  - ✓ Punctuality and neatness.
- The parameters for assessment are to be known to the students at the beginning of the course

# INDEX

EXP. NO.	List of Laboratory Experiments / Assignments
<b>Group A- Database Programming Languages – SQL</b>	
1	Study of Open Source Relational Databases: MySQL
2	Design and develop at SQL DDL statements which demonstrate the use of SQL objects such as Table, View, Index, Sequence and Synonym
3	Design and develop at least 5SQL queries for suitable database application using SQL DML statements: Insert and Select with operators and functions.
4	Design and develop at least 5 SQL queries for suitable database application using SQL DML statements: Update and Delete with operators and functions.
5	Design and develop at least 5 SQL queries for suitable database application using SQL DML statements: all types of Join and Sub-Query.
<b>Group B- Database Programming Languages – PL / SQL</b>	
6	<p>Write a PL/SQL block of code for the following requirements:-</p> <p><b>Schema:</b> 1. Borrower (Roll no., Name, Date of Issue, Name of Book, Status) 2. Fine (Roll no, Date, Amt.) • Accept roll no. &amp; name of book from user. • Check the number of days (from date of issue), if days are between 15 to 30 then fine amount will be Rs 5per day. • If no. of days&gt;30, per day fine will be Rs 50 per day &amp; for days less than 30, Rs. 5 per day. • After submitting the book, status will change from I to R. • If condition of fine is true, then details will be stored into fine table.</p> <p>Frame the problem statement for writing PL/SQL block in line with above statement.</p>
7	<p><b>Cursors: (All types: Implicit, Explicit, Cursor FOR Loop, Parameterized Cursor).</b></p> <p>Write a PL/SQL block of code using parameterized Cursor that will merge the data available in the newly created table N_RollCall with the data available in the table O_RollCall. If the data in the first table already exist in the second table then that data should be skipped. Frame the separate problem statement for writing PL/SQL block to implement all types of Cursors in line with above statement. The problem statement should clearly state the requirements.</p>
8	<p><b>PL/SQL Stored Procedure and Stored Function.</b></p> <p>Write a Stored Procedure namely proc_Grade for the categorization of student. If marks scored by students in examination is &lt;=1500 and marks&gt;=990 then student will be placed in distinction category if marks scored are between 989 and900 category is first class, if marks 899 and 825 category is Higher Second Class Write a PL/SQL block for using procedure created with above requirement. Stud_Marks(name, total_marks) Result(Roll,Name, Class). Frame the separate problem statement for writing PL/SQL Stored Procedure and function, in line with above statement. The problem statement should clearly state the requirements.</p>

9	<p><b>Database Trigger (All Types: Row level and Statement level triggers, Before and After Triggers):</b></p> <p>Write a database trigger on Library table. The System should keep track of the records that are being updated or deleted. The old value of updated or deleted records should be added in Library_Audit table. Frame the problem statement for writing Database Triggers of all types, in-line with above statement. The problem statement should clearly state the requirements.</p>
<b>Group C- Mini Project: Database Project Life Cycle</b>	
10	Implement MYSQL/Oracle database connectivity with PHP/python/Java Implement Database navigation operations (add, delete, edit,) using ODBC/JDBC.
11	<p>Using the database concepts covered in Group A &amp; Group B &amp; connectivity concepts covered in Group C, students in group are expected to design and develop database application with following details: Requirement Gathering and Scope finalization Database Analysis and Design: • Design Entity Relationship Model, Relational Model, Database Normalization • Implementation • Front End : Java/Perl/PHP/Python/Ruby/.net • Backend : MYSQL/Oracle • Database Connectivity : ODBC/JDBC</p> <p>Testing: Data Validation Group of students should submit the Project Report which will be consist of documentation related to different phases of Software Development Life Cycle: Title of the Project, Abstract, Introduction, scope, Requirements, Data Modeling features, Data Dictionary, Relational Database Design, Database Normalization, Graphical User Interface, Source Code, Testing document, Conclusion. Instructor should maintain progress report of mini project throughout the semester from project group and assign marks as a part of the term work.</p>
<b>Link of the Virtual Lab:</b> <a href="http://vlabs.iitb.ac.in/vlabs-dev/labs/dblab/index.php">http://vlabs.iitb.ac.in/vlabs-dev/labs/dblab/index.php</a>	

### Learning Resources:

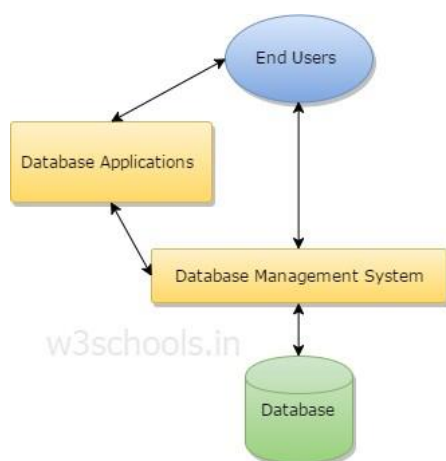
Learning Resources
<p><b>Text Books:</b></p> <ol style="list-style-type: none"> <li>1. A. Silberschatz, H.F. Korth and S. Sudarshan , “Database System Concepts”, McGraw Hill, 6<sup>th</sup> Edition.</li> <li>2. C.J. Date, A. Kannan, S. Swamynathan “An introduction to Database Systems”, Pearson, 8<sup>th</sup> Edition.</li> </ol>
<p><b>Reference Books:</b></p> <ol style="list-style-type: none"> <li>1. Martin Gruber, “Understanding SQL”, Sybex Publications.</li> <li>2. Ivan Bayross, “SQL- PL/SQL”, BPB Publications, 4<sup>th</sup> Edition.</li> <li>3. S.K. Singh, “Database Systems: Concepts, Design and Application”, Pearson, Education, 2<sup>nd</sup> Edition.</li> </ol>
<p><b>MOOC / NPTEL Courses:</b></p> <ol style="list-style-type: none"> <li>1. NPTEL Course “Database Management System”</li> </ol> <p><b>Link of the Course:</b> <a href="https://nptel.ac.in/courses/106/106/106106220/">https://nptel.ac.in/courses/106/106/106106220/</a></p>

**AIM:**

Study of Open Source Relational Databases : MySQL

**OBJECTIVES:**

Study of Open Source Relational Databases : MySQL

**Theory:**

MySQL is a Relational Management Database System(RDBMS), and ships with no GUI tools to administer MySQL databases or manage data contained within the databases. Users may use the included Command line tools, or use MySQL "front-ends", desktop software and web applications that create and manage MySQL databases, build database structures, back up data, inspect status, and work with data records. The official set of MySQL front-end tools, MySQL workbench is actively developed by Oracle, and is freely available for use.

Client server system has one or more client process and one or more server processes, and a client process can send a query to any one server process. Clients are responsible for user-interface issues, and servers manage data and execute transactions. Thus, a client process could run on a personal computer and send queries to a server running on a mainframe.

## Introduction to DBMS

A database management system (DBMS) refers to the technology for creating and managing databases. DBMS is a software tool to organize (create, retrieve, update, and manage) data in a database.

**Component of DBMS:**

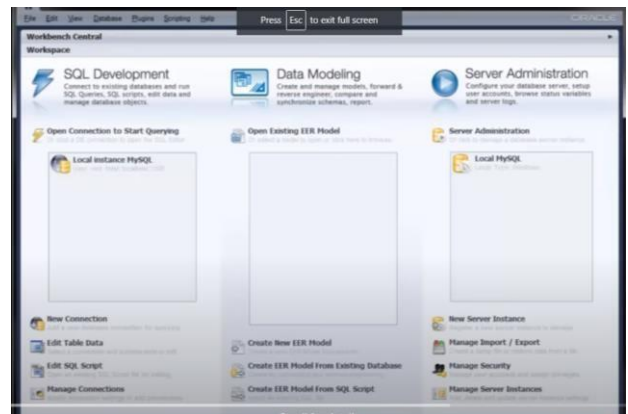
- **Users:** Users may be of any kind such as DB administrator, System developer, or database users.
- **Database application:** Database application may be Departmental, Personal, organization's and / or Internal.
- **DBMS:** Software that allows users to create and manipulate database access,
- **Database:** Collection of logical data as a single unit.



## INTRODUCTION TO SQL:

Pronounced as SEQUEL: Structured English QUERY Language

- Pure non-procedural query language
- Designed and developed by IBM, Implemented by Oracle
- 1978 System/R IBM- 1st Relational DBMS
- 1979 Oracle and Ingres
- 1982 SQL/DS and DB2 IBM
- Accepted by both ANSI + ISO as **Standard Query Language for any RDBMS**
- SQL86 (SQL1) : first by ANSI and ratified by ISO (SQL-87), minor revision on 89
- (SQL-89)
- SQL92 (SQL2) : major revision
- SQL99 (SQL3) : add recursive query, trigger, some OO features, and non-scholar type
- SQL2003 : XML, Window functions, and sequences (Not free)
- Supports all the three sublanguages of DBMS: **DDL, DML, DCL**



## MYSQL Installation Process:

### Installation Process –

Step 1: double click on this first software file

1.mysql-essential-5.1.67-win32.msi

Use following credentials:

port no: 3306

username: root

password: root

hostname: localhost

Step 2: After installation of mysql essential,

Double click this second software file

2.mysql-gui-tools-5.0-r17-win32.msi

Step 3: After installation of mysql-gui-tools  
Double click this third software file  
3.mysql-workbench-gpl-5.2.44-win32.msi

## Data Definition in SQL

### CREATE, ALTER and DROP

table .....relation  
row ..... tuple  
column... attribute

## DATA TYPES

- Numeric: NUMBER, NUMBER(s,p), INTEGER, INT, FLOAT, DECIMAL
- Character: CHAR(n), VARCHAR(n), VARCHAR2(n), CHAR VARYING(n)
- Bit String: BLOB, CLOB
- Boolean: true, false, and null

### Components of SQL:

#### 1) DDL (Data Definition Language)

-This SQL syntax is used to create, modify and delete database structures.DDL syntax cannot be applied to the manipulation of business data.DDL is almost always used by the database administrator, a database schema implementer or an application developer. Every DDL command implicitly issues a COMMIT making permanent all changes in the database.

#### Examples:

- **CREATE:** Create objects in database schema.
- **ALTER:** Alters the structure of objects that exist within the database schema.
- **DROP:** Drops objects that exist within database schema.
- **TRUNCATE:** Removes all records from a table, including all space allocated for the records.
- **COMMENT:**Adds comments ,generally used for proper documentation of a database schema.

#### 2)DML(Data Manipulation Language)

-It is the SQL syntax that allows manipulating data within database tables.

**Examples:** INSERT, UPDATE, DELETE.

#### 3)DCL(Data Control Language)

-It controls access to the database and table data. Occasionally DCL statements are grouped with DML statements.

#### Examples:

COMMIT,SAVEPOINT,ROLLBACK,SET TRANSACTION

- **The commands used in MySQL are:**

#### 1) CREATE :

The CREATE command is used to create a database or create a table in a particular database.

#### i) FOR CREATING A DATABASE :

##### Syntax:

create database database\_name;                      //for creating a database

##### Example:

create database Student\_info                      //Student\_info database is created

## ii)FOR CREATING A TABLE:

The table creation command requires:

Name of the table

Names of fields

Definitions for each field

### Syntax:

```
CREATE TABLE table_name (column_name1 column_type, column_name2 column_type,..... );
```

### Example:

```
create table Student(Roll_no tinyint PRIMARY KEY,Fname varchar(20) NOT NULL,Lname  
varchar(20),Mob_no char(10));
```

**2)ALTER:**MySQL**ALTER** command is used to change a name of your table, any table field or if you want to add or delete an existing column in a table.

### Syntax:

**i)DROP-** Used to delete a particular column.

**Syntax:** mysql> ALTER TABLE table\_name DROP i; //i is the row you want to delete.

**ii)ADD-**Used to add a particular column to an existing table.

**Syntax:**mysql> ALTER TABLE table\_name ADD i int;

**iii)CHANGE-** Used to change a column's definition, use **MODIFY** or **CHANGE** clause along with ALTER command. After the CHANGE keyword, you name the column you want to change, then specify the new definition, which includes the new name

### Syntax:

For example, to change column **c** from CHAR(1) to CHAR(10), do this:

```
mysql> ALTER TABLE table_name MODIFY c CHAR(10);
```

```
mysql> ALTER TABLE testalter_tbl CHANGE i j BIGINT;
```

**3)DELETE:**used to delete a record from any MySQL table, then you can use SQL command **DELETE FROM**.

### Syntax:

```
DELETE FROM table_name [WHERE Clause]
```

**INDEX:** Indexing is the way of keeping table column data sorted so that searching and locating data consumes less time.Hence indexes essentially improve the speed at which records can be located and retrieved from a table.

Types of Index:

**SIMPLE INDEX:** An index created on single column data is called Simple index.

**COMPOSITE INDEX:** An index created on multiple column data is called a composite index.

1) **CREATE INDEX**: A database index is a data structure that improves the speed of operations in a table. Indexes can be created using one or more columns, providing the basis for both rapid random lookups and efficient ordering of access to records.

**Syntax:**

```
CREATE UNIQUE INDEX index_name ON table_name ( column1, column2,...);
```

2) **DELETE INDEX**: Used to delete any index.

**Syntax:**

```
mysql> ALTER TABLE table_name DROP INDEX (c);
```

**VIEWS**: A view is a table whose rows are not explicitly stored in the database but are computed as needed from a view definition. To reduce redundant data to the minimum possible, MySQL allows creation of an object called a view. A view is mapped to a SELECT statement. This technique offers a simple, effective way of hiding columns of a table.

```
S.name,S.sid,S.cid
```

```
FROM stud S,Enrolled E
```

1) **CREATE VIEW**: Used to create a view.

**Syntax:**

```
mysql> CREATE VIEW database_name.view_name AS SELECT * FROM table_name;
```

**Example:**

```
CREATE VIEW stud_info(name,sid,course)
AS SELECT S.name,S.sid,S.cid
FROM stud S,Enrolled E WHERE S.sid AND E.grade='B'
```

2) **DELETE VIEW**: DROP view removes one or more views.

**Syntax:**

```
mysql> DROP view viewname;
```

**Example:**

```
DROP view stud_info;
```

**Conclusion:**

Thus we studied Open Source Relational Databases: MySQL successfully.

**AIM:**

Study of Design and Develop SQL DDL statements which demonstrate the use of SQL objects such as Table, View, Index, Sequence, Synonym

**OBJECTIVES:**

Study of Design and Develop SQL DDL statements which demonstrate the use of SQL objects such as Table, View, Index, Sequence, Synonym

**Problem Statement:**

(Create following tables with constraints, alter table, insert, drop table , rename table, view, index, synonym, sequence/AUTO\_INCREMENT)

Account(Acc\_no, branch\_name,balance)  
branch(branch\_name,branch\_city,assets)  
customer(cust\_name,cust\_street,cust\_city)  
Depositor(cust\_name,acc\_no)  
Loan(loan\_no,branch\_name,amount)  
Borrower(cust\_name,loan\_no)

**Theory:**

A schema is the collection of multiple database objects, which are known as schema objects. These objects have direct access by their owner schema. Below table lists the schema objects.

- Table - to store data
- View - to project data in a desired format from one or more tables
- Sequence - to generate numeric values
- Index - to improve performance of queries on the tables
- Synonym - alternative name of an object

One of the first steps in creating a database is to create the tables that will store an organization's data.Database design involves identifying system user requirements for various organizational systems such as order entry, inventory management, and accounts receivable. Regardless of database size and complexity, each database is comprised of tables.

**Table of Contents**

- [1. DDL](#)
- [2. DML](#)
- [3. DCL](#)
- [4. TCL](#)

**DDL**

DDL is short name of Data Definition Language, which deals with database schemas and descriptions, of how the data should reside in the database.

- [CREATE](#) - to create a database and its objects like (table, index, views, store procedure, function, and triggers)

- ALTER - alters the structure of the existing database
- DROP - delete objects from the database
- TRUNCATE - remove all records from a table, including all spaces allocated for the records are removed
- COMMENT - add comments to the data dictionary
- RENAME - rename an object

### **DML**

DML is short name of Data Manipulation Language which deals with data manipulation and includes most common SQL statements such SELECT, INSERT, UPDATE, DELETE, etc., and it is used to store, modify, retrieve, delete and update data in a database.

- [SELECT](#) - retrieve data from a database
- [INSERT](#) - insert data into a table
- [UPDATE](#) - updates existing data within a table
- [DELETE](#) - Delete all records from a database table
- MERGE - UPSERT operation (insert or update)
- CALL - call a PL/SQL or Java subprogram
- EXPLAIN PLAN - interpretation of the data access path
- LOCK TABLE - concurrency Control

### **DCL**

DCL is short name of Data Control Language which includes commands such as GRANT and mostly concerned with rights, permissions and other controls of the database system.

- GRANT - allow users access privileges to the database
- REVOKE - withdraw users access privileges given by using the GRANT command

### **TCL**

TCL is short name of Transaction Control Language which deals with a transaction within a database.

- COMMIT - commits a Transaction
- ROLLBACK - rollback a transaction in case of any error occurs
- SAVEPOINT - to rollback the transaction making points within groups
- SET TRANSACTION - specify characteristics of the transaction

---

### **SQL Statements For Tables**

char(n).	Fixed length character string, with user-specified length n.
varchar(n).	Variable length character strings, with user-specified maximum length n.
int.	Integer (a finite subset of the integers that is machine-dependent).
smallint.	Small integer (a machine-dependent subset of the integer domain type).
numeric(p,d).	Fixed point number, with user-specified precision of p digits, with n digits to the right of decimal point.
real, double precision.	Floating point and double-precision floating point numbers, with machine-dependent precision.
float(n).	Floating point number, with user-specified precision of at least n digits.

### **Constraints**

Constraints are the set of rules defined in Oracle tables to ensure data integrity. These rules are enforced placed for each column or set of columns. Whenever the table participates in data action, these rules are validated and raise exception upon violation. The available constraint types are NOT NULL, Primary Key, Unique, Check, and Foreign Key.

The below syntax can be used to impose constraint at the column level.

Syntax:

column [data type] [CONSTRAINT constraint\_name] constraint\_type

All constraints except NOT NULL, can also be defined at the table level. Composite constraints can only be specified at the table level.

### **NOT NULL Constraint**

A NOT NULL constraint means that a data row must have a value for the column specified as NOT NULL. If a column is specified as NOT NULL, the Oracle RDBMS will not allow rows to be stored to the employee table that violate this constraint. It can only be defined at column level, and not at the table level.

Syntax:

COLUMN [data type] [NOT NULL]

### **UNIQUE constraint**

Sometimes it is necessary to enforce uniqueness for a column value that is not a primary key column. The UNIQUE constraint can be used to enforce this rule and Oracle will reject any rows that violate the unique constraint. Unique constraint ensures that the column values are distinct, without any duplicates.

Syntax:

Column Level:

COLUMN [data type] [CONSTRAINT <name>] [UNIQUE]

Table Level: CONSTRAINT [constraint name] UNIQUE (column name)

Note: Oracle internally creates unique index to prevent duplication in the column values. Indexes would be discussed later in PL/SQL.

CREATE TABLE TEST

( ... ,

NAME VARCHAR2(20)

CONSTRAINT TEST\_NAME\_UK UNIQUE,

... );

In case of composite unique key, it must be defined at table level as below.

CREATE TABLE TEST

( ... ,

NAME VARCHAR2(20),

STD VARCHAR2(20) ,

CONSTRAINT TEST\_NAME\_UK UNIQUE (NAME, STD)

);

### **Primary Key**

Each table must normally contain a column or set of columns that uniquely identifies rows of data that are stored in the table. This column or set of columns is referred to as the primary key. Most tables have a single column as the primary key. Primary key columns are restricted against NULLs and duplicate values.

Points to be noted -

- A table can have only one primary key.
- Multiple columns can be clubbed under a composite primary key.
- Oracle internally creates unique index to prevent duplication in the column values. Indexes would be discussed later in PL/SQL.

Syntax:

Column level:

COLUMN [data type] [CONSTRAINT <constraint name> PRIMARY KEY]

Table level:

CONSTRAINT [constraint name] PRIMARY KEY [column (s)]

The following example shows how to use PRIMARY KEY constraint at column level.

```
CREATE TABLE TEST ( ID NUMBER CONSTRAINT TEST_PK PRIMARY KEY, ... );
```

The following example shows how to define composite primary key using PRIMARY KEY constraint at the table level.

```
CREATE TABLE TEST ( ..., CONSTRAINT TEST_PK PRIMARY KEY (ID) );
```

## Foreign Key

**When two tables share the parent child relationship based on specific column, the joining column** in the child table is known as Foreign Key. This property of corresponding column in the parent table is known as Referential integrity. Foreign Key column values in the child table can either be null or must be the existing values of the parent table. Please note that only primary key columns of the referenced table are eligible to enforce referential integrity.

If a foreign key is defined on the column in child table then Oracle does not allow the parent row to be deleted, if it contains any child rows. However, if ON DELETE CASCADE option is given at the time of defining foreign key, Oracle deletes all child rows while parent row is being deleted. Similarly, ON DELETE SET NULL indicates that when a row in the parent table is deleted, the foreign key values are set to null.

Syntax:

Column Level:

COLUMN [data type] [CONSTRAINT] [constraint name] [REFERENCES] [table name (column name)]

Table level:

CONSTRAINT [constraint name] [FOREIGN KEY (foreign key column name) REFERENCES] [referenced table name (referenced column name)]

The following example shows how to use FOREIGN KEY constraint at column level.

```
CREATE TABLE TEST (ccode varchar2(5) CONSTRAINT TEST_FK REFERENCES PARENT_TEST(ccode), ... );
```

Usage of ON DELETE CASCADE clause

```
CREATE TABLE TEST (ccode varchar2(5) CONSTRAINT TEST_FK REFERENCES PARENT_TEST (ccode) ON DELETE CASCADE, ... );
```

## Check constraint

Sometimes the data values stored in a specific column must fall within some acceptable range of values. A CHECK constraint requires that the specified check condition is either true or unknown for each row stored in the table. Check constraint allows to impose a conditional rule on a column, which must be validated before data is



inserted into the column. The condition must not contain a sub query or pseudo column CURRVAL, NEXTVAL, LEVEL, ROWNUM, or SYSDATE.

Oracle allows a single column to have more than one CHECK constraint. In fact, there is no practical limit to the number of CHECK constraints that can be defined for a column.

Syntax:

Column level:

COLUMN [data type] CONSTRAINT [name] [CHECK (condition)]

Table level:

CONSTRAINT [name] CHECK (condition)

The following example shows how to use CHECK constraint at column level.

```
CREATE TABLE TEST ( ..., GRADE char (1) CONSTRAINT TEST_CHK CHECK (upper (GRADE) in ('A','B','C')), ... );
```

The following example shows how to use CHECK constraint at table level.

```
CREATE TABLE TEST ( ..., CONSTRAINT TEST_CHK CHECK (stdate <= enddate), );
```

```
create database Student;
```

```
show databases;
```

```
use Student;
```

```
drop database Student;
```

DDL : create, desc, alter, drop, rename,

```
create table <table_name> (column_name1 dat_type(size) [constraint], column_name2 data_type(size) [constraint], ..column_nameN dat_type(size) [constraint]);
```

```
create table Student_info(RollNO integer(10) NOT NULL, Name varchar(30), MobNo integer(10));
```

```
desc student_info;
```

```
show tables;
```

```
drop table student_info;
```

```
alter table student_info add(emailid varchar(30));
```

```
alter table student_info modify(emailid char(10));
```

```
rename Student_info to Stud_data;
```

```
create index idx on student_info(rollno);
```

```
alter table student_info drop index idx;
```

index::

```
show databases;
```

```
use student;
```

```
show tables;
```

```
desc student_info;
```

### **Create table using subquery**

A table can be created from an existing table in the database using a subquery option. It copies the table structure as well as the data from the table. Data can also be copied based on conditions. The column data type definitions including the explicitly imposed NOT NULL constraints are copied into the new table.

The below CTAS script creates a new table EMP\_BACKUP. Employee data of department 20 gets copied into the new table

```
.  
CREATE TABLE EMP_BACKUP  
AS  
SELECT * FROM EMP_TEST  
WHERE department_id=20;
```

### **SQL CREATE INDEX Statement**

The CREATE INDEX statement is used to create indexes in tables.

Indexes are used to retrieve data from the database very fast. The users cannot see the indexes, they are just used to speed up searches/queries.

Note: Updating a table with indexes takes more time than updating a table without (because the indexes also need an update).

So, only create indexes on columns that will be frequently searched against.

#### **CREATE INDEX Syntax**

Creates an index on a table. Duplicate values are allowed:

```
CREATE INDEX index_name  
ON table_name (column1, column2, ...);
```

Example:

```
create index roll_no on student_info(rollno);  
alter table student_info drop index roll_no;
```

#### **CREATE UNIQUE INDEX Syntax**

Creates a unique index on a table. Duplicate values are not allowed:

```
CREATE UNIQUE INDEX index_name  
ON table_name (column1, column2, ...);
```

Note: The syntax for creating indexes varies among different databases. Therefore: Check the syntax for creating indexes in your database.

#### **CREATE INDEX Example**

The SQL statement below creates an index named "idx\_lastname" on the "LastName" column in the "Persons" table:

```
CREATE INDEX idx_lastname  
ON Persons (LastName);
```

If you want to create an index on a combination of columns, you can list the column names within the parentheses, separated by commas:

```
CREATE INDEX idx_pname  
ON Persons (LastName, FirstName);  
DROP INDEX Statement
```

The DROP INDEX statement is used to delete an index in a table.

MS Access:

```
DROP INDEX index_name ON table_name;
```

SQL Server:

```
DROP INDEX table_name.index_name;
```

DB2/Oracle:

```
DROP INDEX index_name;
```

MySQL:

```
ALTER TABLE table_name  
DROP INDEX index_name;
```

### **SOL Sequence**

Sequence is a feature supported by some database systems to produce unique values on demand. Some DBMS like MySQL supports **AUTO\_INCREMENT** in place of Sequence. AUTO\_INCREMENT is applied on columns, it automatically increments the column value by 1 each time a new record is entered into the table. Sequence is also some what similar to AUTO\_INCREMENT but it has some extra features.

```
create table Student_info(RollNO integer(10) Primary key AUTO_INCREMENT, Name varchar(30),  
MobNo integer(10));
```

```
insert into Student_info (Name, MobNo) values ('Amol', 9049417616);
```

```
insert into Student_info (Name, MobNo) values ('Amol', 9049417616);
```

### **Creating Sequence**

Syntax to create sequences is,

```
CREATE Sequence sequence-name  
start with initial-value  
increment by increment-value  
maxvalue maximum-value  
cycle|nocycle
```

Initial-value specifies the starting value of the Sequence, increment-value is the value by which sequence will be incremented and maxvalue specifies the maximum value until which sequence will increment itself. Cycle specifies that if the maximum value exceeds the set limit, sequence will restart its cycle from the beginning. No cycle specifies that if sequence exceeds maxvalue an error will be thrown.

### Example to create Sequence

The sequence query is following

```
CREATE Sequence seq_1  
start with 1  
increment by 1  
maxvalue 999  
cycle ;
```

### Example to use Sequence

The class table,

ID	NAME
1	abhi
2	adam
4	alex

The sql query will be,

```
INSERT into class value(seq_1.nextval,'anu');
```

Result table will look like,

ID	NAME
1	abhi
2	adam
4	alex
1	anu

Once you use nextval the sequence will increment even if you don't Insert any record into the table.

### **CREATE SYNONYM:**

Examples To define the synonym offices for the table locations in the schema hr, issue the following statement:

```
CREATE SYNONYM offices  
FOR hr.locations;
```

To create a PUBLIC synonym for the employees table in the schema hr on the remote database, you could issue the following statement:

```
CREATE PUBLIC SYNONYM emp_table  
FOR hr.employees@remote.us.oracle.com;
```

A synonym may have the same name as the underlying object, provided the underlying object is contained in another schema.

Oracle Database Resolution of Synonyms: Example Oracle Database attempts to resolve references to objects at the schema level before resolving them at the PUBLIC synonym level. For example, the schemas oe and sh both

contain tables named customers. In the next example, user SYSTEM creates a PUBLIC synonym named customers for customers:

```
CREATE PUBLIC SYNONYM customers FOR oe.customers;
```

If the user sh then issues the following statement, then the database returns the count of rows from sh.customers:

```
SELECT COUNT(*) FROM customers;
```

To retrieve the count of rows from oe's.customers, the user sh must preface customers with the schema name. (The user must have select permission on oe's.customers as well.)

```
SELECT COUNT(*) FROM oe.customers;
```

If the user hr's schema does not contain an object named customers, and if hr has select permission on .customers, then hr can access the customers table in oe's schema by using the public synonym customers:

```
SELECT COUNT(*) FROM customers;
```

### **SQL CREATE VIEW Statement**

In SQL, a view is a virtual table based on the result-set of an SQL statement.

A view contains rows and columns, just like a real table. The fields in a view are fields from one or more real tables in the database.

You can add SQL functions, WHERE, and JOIN statements to a view and present the data as if the data were coming from one single table.

**CREATE VIEW Syntax**

```
CREATE VIEW view_name AS
```

```
SELECT column1, column2, ...
```

```
FROM table_name
```

```
WHERE condition;
```

Note: A view always shows up-to-date data! The database engine recreates the data, using the view's SQL statement, every time a user queries a view.

**SQL CREATE VIEW Examples**

If you have the Northwind database you can see that it has several views installed by default.

The view "Current Product List" lists all active products (products that are not discontinued) from the "Products" table. The view is created with the following SQL:

```
CREATE VIEW [Current Product List] AS
```

```
SELECT ProductID, ProductName
```

```
FROM Products
```

```
WHERE Discontinued = No;
```

Then, we can query the view as follows:

```
SELECT * FROM [Current Product List];
```

Another view in the Northwind sample database selects every product in the "Products" table with a unit price higher than the average unit price:

```
CREATE VIEW [Products Above Average Price] AS
```

```
SELECT ProductName, UnitPrice
FROM Products
WHERE UnitPrice > (SELECT AVG(UnitPrice) FROM Products);
```

We can query the view above as follows:

```
SELECT * FROM [Products Above Average Price];
```

Another view in the Northwind database calculates the total sale for each category in 1997. Note that this view selects its data from another view called "Product Sales for 1997":

```
CREATE VIEW [Category Sales For 1997] AS
SELECT DISTINCT CategoryName, Sum(ProductSales) AS CategorySales
FROM [Product Sales for 1997]
GROUP BY CategoryName;
```

We can query the view above as follows:

```
SELECT * FROM [Category Sales For 1997];
```

We can also add a condition to the query. Let's see the total sale only for the category "Beverages":

```
SELECT * FROM [Category Sales For 1997]
WHERE CategoryName = 'Beverages';
```

SQL Updating a View

You can update a view by using the following syntax:

```
SQL CREATE OR REPLACE VIEW Syntax
CREATE OR REPLACE VIEW view_name AS
SELECT column1, column2, ...
FROM table_name
WHERE condition;
```

Now we want to add the "Category" column to the "Current Product List" view. We will update the view with the following SQL:

```
CREATE OR REPLACE VIEW [Current Product List] AS
SELECT ProductID, ProductName, Category
FROM Products
WHERE Discontinued = No;
```

SQL Dropping a View

You can delete a view with the DROP VIEW command.

```
SQL DROP VIEW Syntax
DROP VIEW view_name;
```

**Examples:**

```
insert into student_info values(3,'amol',2154455,'abc@gmai');
insert into student_info values(1,'vijay',21543255,'asvs@gmai');
select * from student_info;
```

view::

```
create view myview as select rollno,name from student_info;
select * from myview;
create view myview2 as select rollno,name from student_info where rollno>1;
```

(Create following tables with constraints, alter table, insert, drop table , rename table, view, index, synonym, sequence/AUTO\_INCREMENT)

```
Account(Acc_no, branch_name,balance)
branch(branch_name,branch_city,assets)
customer(cust_name,cust_street,cust_city)
Depositor(cust_name,acc_no)
Loan(loan_no,branch_name,amount)
Borrower(cust_name,loan_no)
```

**Solve following queries:**

- Q1.Create Depositor table with foreign key with on delete cascade constraint on columns cust\_name and acc\_no.
- Q2. Create Borrower table with foreign key with on delete cascade constraint on columns cust\_name,loan\_no.
- Q3. Create Account table with primary key and AUTO\_INCREMENT constraint on Acc\_no column
- Q4. Create Loan table with primary key and AUTO\_INCREMENT constraint on loan\_no column.
- Q5. Create Customer table with primary key constraint on cust\_name column.
- Q6. Create View on Account table and Loan Table.
- Q7. Insert following Data into above tables
- Q.8. Create synonym for customer table as cust.
- Q.9. Create sequence acc\_seq and use in Account table for acc\_no column.
- Q.10 Insert following data into all above tables.

\*\*\*\*\***Problem Statements**\*\*\*\*\*

- Q1.** Q1.Create Depositor table with foreign key with on delete cascade constraint on columns cust\_name and acc\_no.

Column Level:

```
SQL> create table depositor (cust_name varchar(20) CONSTRAINT FK_1 REFRENECS
customer(cust_name) ON DELETE CASCADE , acc_no integer(10) CONSTRAINT FK_2
REFRENECS account(acc_no) ON DELETE CASCADE);
```

**Q2.** Create Borrower table with foreign key with on delete cascade constraint on columns cust\_name,loan\_no

Table level :

```
SQL> create table borrower (cust_name varchar(20), loan_no integer(10) , CONSTRAINT FK_1
FOREIGN KEY (cust_name) REFRENECS customer(cust_name) ON DELETE CASCADE,
CONSTRAINT FK_2 FOREIGN KEY (loan_no) FK2 REFRENECS loan(loan_no) ON DELETE
CASCADE);
```

**Q3.** Create Account table with primary key and AUTO\_INCREMENT constraint on Acc\_no column

```
SQL> create table account (acc_no integer(10) primary key AUTO_INCREMENT, branch_name
varchar(20), balance integer(10));
```

**Q4.** Create Loan table with primary key and AUTO\_INCREMENT constraint on loan\_no column.

```
SQL> create table loan (loan_no integer(10) primary key AUTO_INCREMENT, branch_name
varchar(20), amount integer(10));
```

**Q5.** Create Customer table with primary key constraint on cust\_name column.

```
SQL> create table customer (cust_name varchar(20) primary key, cust_street varchar(20), city
varchar(20));
```

**Q6.** Create View on Account table and Loan Table.

```
SQL> create view ac1 AS (select acc_no, balance from account);
```

```
SQL> create view ln1 AS (select loan_no, amount from loan);
```

**Q7.** Create synonym for customer table as cust.

```
SQL> create public synonym cust2 for customer1;
Synonym created.
```

**Q8.** Create sequence acc\_seq and use in Account table for acc\_no column.

```
CREATE Sequence seq_1 start with 1 increment by 1 maxvalue 100000 no cycle ;
```

**Q.9** Insert following data into all above tables.

\*\*\*\*\* **Table Structure** \*\*\*\*\*

```
create table Account(Acc_no, branch_name,balance) :
```



SQL> select \* from account;

ACC_NO	BRANCH_NAME	BALANCE
1001	Akurdi	15000
1002	Nigdi	11000
1003	Chinchwad	20000
1004	Wakad	10000
1005	Akurdi	14000
1006	Nigdi	17000

6 rows selected.

**Create table branch(branch\_name,branch\_city,assets) :**

SQL> select \* from branch;

BRANCH_NAME	BRANCH_CITY	ASSETS
Akurdi	Pune	200000
Nigdi	Pimpri_chinchwad	300000
Wakad	Pune	100000
Chinchwad	Pimpri_chinchwad	400000
Sangvi	Pune	230000

**create table customer(cust\_name,cust\_street,cust\_city) :**

SQL> select \* from customer1;

CUST_NAME	CUST_STREET	CUST_CITY
Rutuja	JM road	Pune
Alka	Senapati road	Pune
Samiksha	Savedi road	Pimpri_chinchwad
Trupti	Lakshmi road	Pune
Mahima	Pipeline road	Pimpri_chinchwad
Ayushi	FC road	pune
Priti	Camp road	Pimri_chinchwad

7 rows selected.

**Create table Depositor(cust\_name,acc\_no):**

SQL> select \* from depositer;

CUST_NAME	ACC_NO
Rutuja	1005
Trupti	1002
Samiksha	1004

**Loan(loan\_no,branch\_name,amount) :**

SQL> select \* from loan;

LOAN_NO	BRANCH_NAME	AMMOUNT
2001	Akurdi	2000
2002	Nigdi	1200
2003	Akurdi	1400
2004	Wakad	1350
2005	Chinchwad	1490
2006	Akurdi	12300
2007	Akurdi	14000

7 rows selected.

**Create table borrower(cust\_name,loan\_no) :**

SQL> select \* from borrower;

CUST_NAME	LOAN_NO
Mahima	2005
Trupti	2002
Rutuja	2004
Ayushi	2006
Priti	2007

**Conclusion:**

Thus we successfully implemented Table, View, Index, Sequence, Synonym MySQL queries.

**Expt. No: 3 &4**

**Design and develop at least 5SQL queries for suitable database application using SQL DML statements: Insert and Select with operators and functions, Update and Delete with operators and functions.**

**AIM:**

To Design at least 10 SQL queries for suitable database application using SQL DML statements: Insert, Select, Update, Delete with operators, functions, and set operator.

**OBJECTIVES:**

To Study and Design SQL queries for suitable database application using SQL DML statements: Insert, Select, Update, Delete with operators, functions, and set operator.

**Problem Statement:**

(Insert, Select, Update, Delete, operators, functions, setoperator, all constraints, view, index, synonym, sequence)

Account(Acc\_no, branch\_name,balance)

branch(branch\_name,branch\_city,assets)

customer(cust\_name,cust\_street,cust\_city)

Depositor(cust\_name,acc\_no)

Loan(loan\_no,branch\_name,amount)

Borrower(cust\_name,loan\_no)

**Input:** insert Data into above tables and fire queries on databases;

**Theory:****Set Operators:**

The set operations union, intersect, and except operate on relations and correspond to the relational algebra operations  $\cup$ ,  $\cap$ ,  $-$ .

Each of the above operations automatically eliminates duplicates; to retain all duplicates use the corresponding multiset versions union all, intersect all and except all.

Suppose a tuple occurs m times in r and n times in s, then, it occurs:

m + n times in r **union all** s

min(m,n) times in r **intersect all** s

max(0, m – n) times in r **except all** s

**Aggregate Functions:**

These functions operate on the multiset of values of a column of a relation, and return a value

avg: average value

min: minimum value

max: maximum value

sum: sum of values

count: number of values

**Solve following queries:**

- Q1. Find the names of all branches in loan relation.
- Q2. Find all loan numbers for loans made at Akurdi Branch with loan amount > 12000.
- Q3. Find all customers who have a loan from bank. Find their names, loan\_no and loan amount.
- Q4. List all customers in alphabetical order who have loan from Akurdi branch.
- Q5. Find all customers who have an account or loan or both at bank.
- Q6. Find all customers who have both account and loan at bank.
- Q7. Find all customer who have account but no loan at the bank.
- Q8. Find average account balance at Akurdi branch.
- Q9. Find the average account balance at each branch
- Q10. Find no. of depositors at each branch.
- Q11. Find the branches where average account balance > 12000.
- Q12. Find number of tuples in customer relation.
- Q13. Calculate total loan amount given by bank.
- Q14. Delete all loans with loan amount between 1300 and 1500.
- Q15. Delete all tuples at every branch located in Nigdi.
- Q.16. Create synonym for customer table as cust.
- Q.17. Create sequence roll\_seq and use in student table for roll\_no column.

Create above tables with appropriate constraints like primary key, foreign key, check constraints, not null etc.

\*\*\*\*\* **Table Structure** \*\*\*\*\*

**create table Account(Acc\_no, branch\_name,balance) :**

SQL> select \* from account;

ACC_NO	BRANCH_NAME	BALANCE
1001	Akurdi	15000
1002	Nigdi	11000
1003	Chinchwad	20000
1004	Wakad	10000
1005	Akurdi	14000
1006	Nigdi	17000

6 rows selected.

**Create table branch(branch\_name,branch\_city,assets) :**

SQL> select \* from branch;

BRANCH_NAME	BRANCH_CITY	ASSETS
Akurdi	Pune	200000
Nigdi	Pimpri_chinchwad	300000
Wakad	Pune	100000
Chinchwad	Pimpri_chinchwad	400000
Sangvi	Pune	230000

**create table customer(cust\_name,cust\_street,cust\_city) :**

SQL> select \* from customer1;

CUST_NAME	CUST_STREET	CUST_CITY
Rutuja	JM road	Pune
Alka	Senapati road	Pune
Samiksha	Savedi road	Pimpri_chinchwad
Trupti	Lakshmi road	Pune
Mahima	Pipeline road	Pimpri_chinchwad
Ayushi	FC road	pune
Priti	Camp road	Pimri_chinchwad

7 rows selected.

**Create table Depositor(cust\_name,acc\_no):**

SQL> select \* from depositer;

CUST_NAME	ACC_NO
Rutuja	1005
Trupti	1002
Samiksha	1004

**Loan(loan\_no,branch\_name,amount) :**

SQL> select \* from loan;

LOAN_NO	BRANCH_NAME	AMMOUNT
2001	Akurdi	2000
2002	Nigdi	1200
2003	Akurdi	1400
2004	Wakad	1350
2005	Chinchwad	1490
2006	Akurdi	12300
2007	Akurdi	14000

7 rows selected.

**Create table borrower(cust\_name,loan\_no) :**

SQL> select \* from borrower;

CUST_NAME	LOAN_NO
Mahima	2005
Trupti	2002
Rutuja	2004
Ayushi	2006
Priti	2007

\*\*\*\*\***Problem Statements**\*\*\*\*\*

**Q1. Find the names of all branches in loan relation.**

SQL>select branch\_name from loan;

BRANCH_NAME
Akurdi
Nigdi
Akurdi
Wakad
Chinchwad
Akurdi
Akurdi

7 rows selected.

**Q2. Find all loan numbers for loans made at Akurdi Branch with loan amount >12000.**

SQL> select loan\_no from loan where branch\_name='Akurdi' and amount>12000;

LOAN\_NO

-----

2006

2007

**Q3. Find all customers who have a loan from bank. Find their names, loan\_no and loan amount.**

SQL> select b.cust\_name,b.loan\_no,l.amount from borrower b inner join loan l on  
b.loan\_no=l.loan\_no;

CUST_NAME	LOAN_NO	AMOUNT
-----	-----	-----
Trupti	2002	1200
Rutuja	2004	1350
Mahima	2005	1490
Ayushi	2006	12300
Priti	2007	14000

**Q4. List all customers in alphabetical order who have loan from Akurdi branch.**

SQL> select b.cust\_name from borrower b inner join loan l on b.loan\_no=l.loan\_no  
where l.branch\_name='Akurdi' order by b.cust\_name;

CUST\_NAME

-----

Ayushi

Priti

**Q5. Find all customers who have an account or loan or both at bank.**

SQL>select cust\_name from depositer union select cust\_name from borrower;

CUST\_NAME

-----

Ayushi

MahimaPriti

Rutuja

Samiksha

Trupti

6 rows selected.

**Q6. Find all customers who have both account and loan at bank.**

SQL> select cust\_name from depositer intersect select cust\_name from borrower;

CUST\_NAME

-----  
Rutuja  
Trupti

**Q7. Find all customer who have account but no loan at the bank.**

SQL> select cust\_name from depositer minus select cust\_name from borrower;

CUST\_NAME

-----  
Samiksha

**Q8. Find average account balance at Akurdi branch.**

SQL> select avg(balance) from account where branch\_name='Akurdi';

AVG(BALANCE)

-----  
14500

**Q9. Find the average account balance at each branch**

SQL> select branch\_name,avg(balance) from account group by branch\_name;

BRANCH_NAME	AVG(BALANCE)
-------------	--------------

Chinchwad	20000
Nigdi	14000
Wakad	10000
Akurdi	14500

**10. Find no. of depositors at each branch.**

SQL> select branch\_name,count(branch\_name) from account a inner join depositer d on a.acc\_no=d.acc\_no group by branch\_name;

BRANCH_NAME	COUNT(BRANCH_NAME)
-------------	--------------------

Nigdi	1
Wakad	1
Akurdi	1



**Q11. Find the branches where average account balance > 12000.**

SQL> select branch\_name from account group by branch\_name having avg(balance)>1200;

BRANCH\_NAME

-----

Chinchwad

Nigdi

Wakad

Akurdi

**Q12. Find number of tuples in customer relation.**

SQL> select count(cust\_name) no\_of\_tuples from customer1;

NO\_OF\_TUPLES

-----

7

**Q13. Calculate total loan amount given by bank.**

SQL> select sum(amount) total\_loan\_amount from loan;

TOTAL\_LOAN\_AMOUNT

-----

33740

**Q14. Delete all loans with loan amount between 1300 and 1500.**

SQL> delete from loan where amount>1300 and amount<1500;

LOAN_NO	BRANCH_NAME	AMOUNT
---------	-------------	--------

-----

2001	Akurdi	2000
------	--------	------

2002	Nigdi	1200
------	-------	------

2006	Akurdi	12300
------	--------	-------

2007	Akurdi	14000
------	--------	-------

**Q15. Delete all tuples at every branch located in Nigdi.**

SQL>delete from branch where branch\_name='Nigdi';

**Q.16. Create synonym for customer table as cust.**

SQL> create public synonym cust2 for customer1;

Synonym created.

**Q.17. Create sequence roll\_seq and use in student table for roll\_no column.**

**Conclusion:**

Thus we successfully implemented MySQL queries.

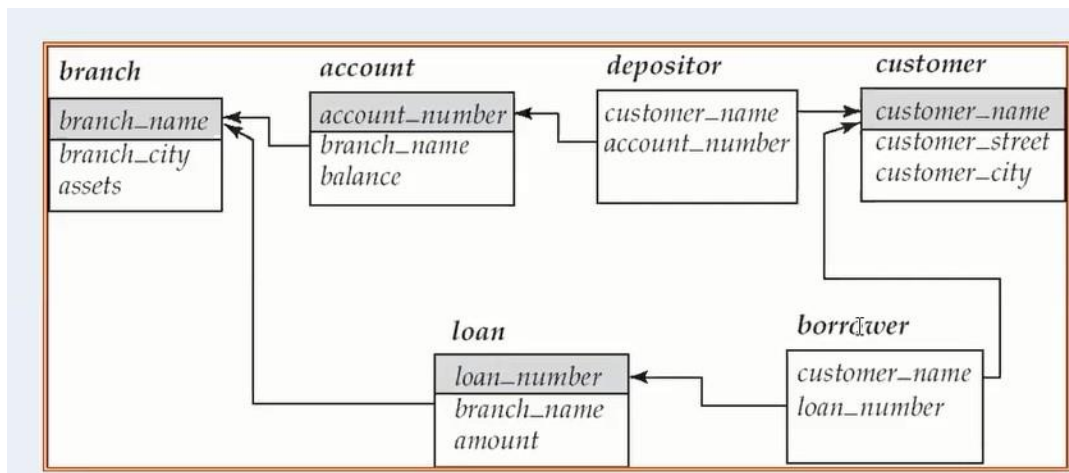
**Expt. No: 5**

**Design and develop at least 5 SQL queries for suitable database application using SQL DML statements: all types of Join and Sub-Query.**

**AIM:** To Design at least 5 SQL queries for suitable database application using SQL DML statements: all types of Join and Sub-Query.

**Problem Statement:** Design SQL queries for suitable database application using SQL DML statements: all types of Join, Sub-Query and View.

```
create database pune_bank;
use pune_bank;
#branch(branch_name,branch_city,assets)
#Account(Acc_no, branch_name,balance)
#Loan(loan_no,branch_name,amount)
#customer(cust_name,cust_street,cust_city)
#Depositor(cust_name,acc_no)
#Borrower(cust_name,loan_no)
```



### 1. Create following Tables

cust\_mstr(cust\_no,fname,lname)

add\_dets(code\_no,add1,add2,state,city,pincode)

**Retrieve the address of customer Fname as 'xyz' and Lname as 'pqr'**

### 2. Create following Tables

cust\_mstr(custno,fname,lname)

acc\_fd\_cust\_dets(codeno,acc\_fd\_no)

fd\_dets(fd\_sr\_no,amt)

**List the customer holding fixed deposit of amount more than 5000**

### 3. Create following Tables

emp\_mstr(e\_mpnno,f\_name,l\_name,m\_name,dept,desg,branch\_no)

branch\_mstr(name,b\_no)

**List the employee details along with branch names to which they belong**

#### 4. Create following Tables

emp\_mstr(emp\_no,f\_name,l\_name,m\_name,dept)

cntc\_dets(code\_no,cntc\_type,cntc\_data)

**List the employee details along with contact details using left outer join & right join**

#### 5. Create following Tables

cust\_mstr(cust\_no,fname,lname)

add\_dets(code\_no,pincode)

**List the customer who do not have bank branches in their vicinity.**

**6. a) Create View on borrower table by selecting any two columns and perform insert update delete operations**

b) Create view on borrower and depositor table by selecting any one column from each table perform insert update delete operations

c) create updateable view on borrower table by selecting any two columns and perform insert, update and delete operations.

#### Solutions:

##### 1. Create following Tables

cust\_mstr(cust\_no,fname,lname)

add\_dets(code\_no,add1,add2,state,city,pincode)

**Retrieve the address of customer Fname as 'Rutuja' and Lname as 'Deshmane'**

SQL> select \* from cust\_mstr;

CUSTNO	FNAME	LNAME
C101	Rutuja	Deshmane
C102	Trupti	Bargaje
C103	Samiksha	Dharmadhikari
C104	Mahima	Khandelwal

SQL> select add1,add2 from add\_dets where code\_no in(select custno from cust\_mstr where fname='Rutuja' and lname='Deshmane');

ADD1	ADD2
venu nagar	dange chowk

##### 2. Create following Tables

cust\_mstr(custno,fname,lname)

acc\_fd\_cust\_dets(codeno,acc\_fd\_no)

fd\_dets(fd\_sr\_no,amt)

### List the customer holding fixed deposit of amount more than 5000

```
SQL> select fname,lname from cust_mstr where custno in(select codeno from acc_fd_cust_dets where acc_fd_no in(select fd_sr_no from fd_dets where amt>5000));
```

FNAME	LNAME
Rutuja	Deshmane
Samiksha	Dharmadhikari

### 3. Create following Tables

```
emp_mstr(e_mpnno,f_name,l_name,m_name,dept,desg,branch_no)
branch_mstr(name,b_no)
```

### List the employee details along with branch names to which they belong

```
SQL> select emp_no,fname,lname,mname,dept,desg,branch_no,b.name from emp_mstr e inner join branch_tb b on e.branch_no=b.b_no;
```

EMP_NO	FNAME	LNAME	MNAME	DEPT	DESG	BRANCH_NO	NAME
1011	Samarth	Deshmane	Suryakant	sports	trainer	2011	Akurdi
1012	Alka	Choudhari	Rohitash	comp	tester	2012	nigdi
1013	Shriyash	Shingare	Santosh	comp	coder	2013	chinchwad

### 4. Create following Tables

```
emp_mstr(emp_no,f_name,l_name,m_name,dept)
cntc_dets(code_no,cntc_type,cntc_data)
```

### List the employee details along with contact details using left outer join & right join

```
SQL> select emp_no,fname,lname,mname,dept,c.code_no,c.cntc_type,c.cntc_data from emp_mstr e left outer join cntc_dets c on e.emp_no=c.code_no;
```

EMP_NO	FNAME	LNAME	MNAME	DEPT	CODE_NO	CNTC_TYPE	CNTC_DATA
1011	Samarth	Deshmane	Suryakant	sports	1011	phno	9689349523
1012	Alka	Choudhari	Rohitash	comp	1012	email	rutu@gmail.com
1013	Shriyash	Shingare	Santosh	comp			

```
SQL> select emp_no,fname,lname,mname,dept,c.code_no,c.cntc_type,c.cntc_data from emp_mstr e right outer join cntc_dets c on e.emp_no=c.code_no;
```

EMP_NO	FNAME	LNAME	MNAME	DEPT	CODE_NO	CNTC_TYPE	CNTC_DATA
1011	Samarth	Deshmane	Suryakant	sports	1011	phno	9689349523
1012	Alka	Choudhari	Rohitash	comp	1012	email	rutu@gmail.com
1014						email	shrink@gmail.com

### 5. Create following Tables

```
cust_mstr(cust_no,fname,lname)
add_dets(code_no,pincode)
```

### List the customer who do not have bank branches in their vicinity.

```
SQL> select * from cust_mstr where cust_no in (select code_no from add_dets where code_no like 'C%' and pincode not in (select pincode from add_dets where code_no like 'B%'));
```

CUST_NO	FNAME	LNAME
C102	Trupti	Bargaje

**6. A)** Create View on borrower table by selecting any two columns and perform insert update delete operations

SQL> select \* from borrower;

ACC_NO	NAME	AMOUNT
101	Aish	10000
102	Adi	10000
103	Swati	45216

SQL> create view b1 as select name, amount from borrower;  
View created.

SQL> select \* from b1;

NAME	AMOUNT
Aish	10000
Adi	10000
Swati	45216

SQL> update b1 set amount=7845 where name='swati';  
1 row updated.

SQL> select \* from borrower;

ACC_NO	NAME	AMOUNT
101	Aish	10000
102	Adi	10000
103	Swati	7845

SQL> delete from b1 where name='swati';  
1 row deleted.

SQL> select \* from borrower;

ACC_NO	NAME	AMOUNT
101	Aish	10000

**B)** Create view on borrower and depositor table by selecting any one column from each table  
perform insert update delete operations

SQL> select \* from borrower;

ACC_NO	NAME	AMOUNT
101	Aish	10000
102	Adi	10000

SQL> select \* from depositor;

DACC_NO	DNAME	DAMOUNT
102	Adi	45789
104	Sneha	7895
103	Swati	79854

SQL> create view b3 as select amount loan, damount deposit from borrower, depositor;  
View created.

SQL> select \* from b3;

LOAN	DEPOSIT
10000	45789
10000	7895
10000	79854

C) create updateable view on borrower table by selecting any two columns and perform insert, Update and delete operations.

SQL> create table borrower(acc\_no number(10) primary key,name varchar(10),amount number(10));  
Table created.

SQL> insert into borrower values(&acc,'&name',&amount);  
Enter value for acc: 101  
Enter value for name: Aish  
Enter value for amount: 10000  
old 1: insert into borrower values(&acc,'&name',&amount)  
new 1: insert into borrower values(101,'aish',10000)  
1 row created.

SQL> /  
Enter value for acc: 102  
Enter value for name: Adi  
Enter value for amount: 4500  
old 1: insert into borrower values(&acc,'&name',&amount)  
new 1: insert into borrower values(102,'adi',4500)  
1 row created.

SQL> /  
Enter value for acc: 103  
Enter value for name: Swati  
Enter value for amount: 45216.  
old 1: insert into borrower values(&acc,'&name',&amount)  
new 1: insert into borrower values(103,'swati',45216.)  
1 row created.

SQL> create view bview as select acc\_no, amount from borrower;  
View created.

```
SQL> insert into bview values(&acc,&amount);
Enter value for acc: 104
Enter value for amount: 58901
old 1: insert into bview values(&acc,&amount)
new 1: insert into bview values(104,58901)
1 row created.
```

```
SQL> select * from borrower;
```

ACC_NO	NAME	AMOUNT
101	aish	10000
102	adi	4500
103	swati	45216
104		58901

```
SQL> update bview set amount=45000 where acc_no=104;
1 row updated.
```

```
SQL> select * from borrower;
```

ACC_NO	NAME	AMOUNT
101	aish	10000
102	adi	4500
103	swati	45216
104		45000

```
SQL> select * from bview;
```

ACC_NO	AMOUNT
101	10000
102	4500
103	45216
104	45000

```
SQL> delete from bview where acc_no=104;
1 row deleted.
```

```
SQL> select * from bview;
```

ACC_NO	AMOUNT
101	10000
102	4500
103	45216

### Conclusion:

Thus we successfully implemented MySQL queries.



**Aim:** Study of PL SQL Control Structures and Exception Handling.

**Input:** Student roll no and attendance is input to Procedure.

**Theory:**

- The PL/SQL programming language was developed by Oracle Corporation in the late 1980s as procedural extension language for SQL and the Oracle relational database. Following are certain notable facts about PL/SQL –
- PL/SQL is a completely portable, high-performance transaction-processing language.
- PL/SQL provides a built-in, interpreted and OS independent programming environment.
- PL/SQL can also directly be called from the command-line SQL\*Plus interface.
- Direct call can also be made from external programming language calls to database.
- PL/SQL's general syntax is based on that of ADA and Pascal programming language.
- Apart from Oracle, PL/SQL is available in TimesTen in-memory database and IBM DB2.

**Features of PL/SQL**

- PL/SQL has the following features –
- PL/SQL is tightly integrated with SQL.
- It offers extensive error checking.
- It offers numerous data types.
- It offers a variety of programming structures.
- It supports structured programming through functions and procedures.
- It supports object-oriented programming.
- It supports the development of web applications and server pages.

**Advantages of PL/SQL**

- PL/SQL has the following advantages –
- SQL is the standard database language and PL/SQL is strongly integrated with SQL. PL/SQL supports both static and dynamic SQL. Static SQL supports DML operations and transaction control from PL/SQL block. In Dynamic SQL, SQL allows embedding DDL statements in PL/SQL blocks.
- PL/SQL allows sending an entire block of statements to the database at one time. This reduces network traffic and provides high performance for the applications.

- PL/SQL gives high productivity to programmers as it can query, transform, and update data in a database.
- PL/SQL saves time on design and debugging by strong features, such as exception handling, encapsulation, data hiding, and object-oriented data types.
- Applications written in PL/SQL are fully portable.
- PL/SQL provides high security level.
- PL/SQL provides access to predefined SQL packages.
- PL/SQL provides support for Object-Oriented Programming.
- PL/SQL provides support for developing Web Applications and Server Pages.

DECLARE :- if you want to declare a variable in plsql program then it takes place in declare section

BEGIN:- is used to start the working of program and end is used to terminate the begin.

Delimiter is used to run (/)

SET SERVEROUTPUT ON ; is run before every time when you compile a program in a session.

SET ECHO ON : is optional

DBMS\_OUTPUT.PUT\_LINE command for e.g. if sal=10 and you want to print it Then it looks like  
dbms\_output.put\_line('the salary is ' || sal);

## **IF STATEMENT**

Common syntax

IF condition THEN

statement 1;

ELSE

statement 2;

END IF;

**INTO command:** is used to catch a value in variable from table under some while condition

Only one value must be returned For e.g. in the above example if there are two people whose name is john then it shows error

## Exception Handling:

An exception is an error condition during a program execution. PL/SQL supports programmers to catch such conditions using EXCEPTION block in the program and an appropriate action is taken against the error condition. There are two types of exceptions –

- System-defined exceptions
- User-defined exceptions

### Syntax for Exception Handling

The general syntax for exception handling is as follows. Here you can list down as many exceptions as you can handle. The default exception will be handled using *WHEN others THEN* –

DECLARE

<declarations section>

BEGIN

<executable command(s)>

EXCEPTION

<exception handling goes here >

WHEN exception1 THEN

exception1-handling-statements

WHEN exception2 THEN

exception2-handling-statements

WHEN exception3 THEN

exception3-handling-statements

.....

WHEN others THEN

exception3-handling-statements

END;

**ORACLE :**

<http://127.0.0.1:8080/apex/f?p=4500:1000:2849714591695263>

### Problem Statement:

Use of Control structure and Exception handling is mandatory. Write a PL/SQL block of code for the following requirements: Schema:

1. Borrower(Roll\_no, Name, DateofIssue, NameofBook, Status)
2. Fine(Roll\_no,Date,Amt)
  - a) Accept roll\_no & name of book from user.

- b) Check the number of days (from date of issue), if days are between 15 to 30 then fine amount will be Rs 5per day.
- c) If no. of days>30, per day fine will be Rs 50 per day & for days less than 30, Rs. 5 per day.
- d) After submitting the book, status will change from I to R.
- e) If condition of fine is true, then details will be stored into fine table.

### Solution in Mysql:

Steps are:

- 1) Create borrower and fine table with primary and foreign keys
- 2) insert records in borrower table
- 3) create procedure to insert entries in fine table with exception handling
- 4) call procedure to calculate fine and display fine table.

```
mysql> create table borrower(rollin int primary key,name varchar(20),dateofissue date,nameofbook
varchar(20),status varchar(20));
Query OK, 0 rows affected (0.30 sec)
```

```
mysql> desc borrower;
```

Field	Type	Null	Key	Default	Extra
rollin	int(11)	NO	PRI	NULL	
name	varchar(20)	YES		NULL	
dateofissue	date	YES		NULL	
nameofbook	varchar(20)	YES		NULL	
status	varchar(20)	YES		NULL	

5 rows in set (0.00 sec)

```
mysql> create table fine(rollno int,foreign key(rollno) references borrower(rollin),returndate date,amount
int);
Query OK, 0 rows affected (0.38 sec)
```

```
mysql> desc fine;
```

Field	Type	Null	Key	Default	Extra
roll_no	int(11)	YES	MUL	NULL	
returndate	date	YES		NULL	
amnt	int(11)	YES		NULL	

3 rows in set (0.02 sec)

```
mysql> insert into borrower values(1,'abc','2017-08-01','SEPM','PEN')$
Query OK, 1 row affected (0.16 sec)
```

```
mysql> insert into borrower values(2,'xyz','2017-07-01','DBMS','PEN')$
Query OK, 1 row affected (0.08 sec)
```

```
mysql> insert into borrower values(3,'pqr','2017-08-15','DBMS','PEN')$
```

Query OK, 1 row affected (0.03 sec)

```
mysql> delimiter $
mysql> create procedure calc_fine_lib6(in roll int)
begin
declare fine1 int;
declare noofdays int;
declare issuedate date;
declare exit handler for SQLEXCEPTION select'create table definition';
select dateofissue into issuedate from borrower where rollin=roll;
select datediff(curdate(),issuedate) into noofdays;
if noofdays>15 and noofdays<=30 then
set fine1=noofdays*5;
insert into fine values(roll,curdate(),fine1);
elseif noofdays>30 then
set fine1=((noofdays-30)*50) + 15*5;
insert into fine values(roll,curdate(),fine1);
else
insert into fine values(roll,curdate(),0);
end if;
update borrower set status='return' where rollin=roll;
end $
```

```
mysql> call calc_fine_lib6(1)$
Query OK, 0 rows affected (0.09 sec)
mysql> call calc_fine_lib6(2)$
Query OK, 0 rows affected (0.09 sec)
mysql> call calc_fine_lib6(3)$
Query OK, 0 rows affected (0.09 sec)
```

```
mysql> select * from fine;
-> $
+-----+-----+-----+
| roll_no | returndate | amnt |
+-----+-----+-----+
| 1 | 2017-08-22 | 105 |
| 2 | 2017-08-22 | 780 |
| 3 | 2017-08-22 | 0 |
+-----+-----+-----+
3 rows in set (0.00 sec)
```

```
mysql> drop table fine$
Query OK, 0 rows affected (0.21 sec)
```

```
mysql> call calc_fine_lib6(1)$
+-----+
| create table definition |
+-----+
| create table definition |
+-----+
+-----+-----+-----+
| roll_no | returndate | amnt |
+-----+-----+-----+
```

1 row in set (0.00 sec)

Query OK, 0 rows affected (0.00 sec)

```
mysql> create table fine(rollno int,foreign key(rollno) references borrower(rollin),returndate date,amount
int)$
```

Query OK, 0 rows affected (0.34 sec)

```
mysql> call calc_fine_lib6(1)$
```

Query OK, 0 rows affected (0.09 sec)

```
mysql> select * from fine$
```

```
+-----+-----+-----+
| rollno | returndate | amount |
+-----+-----+-----+
| 1 | 2017-08-22 | 105 |
+-----+-----+-----+
1 row in set (0.00 sec)
```

---

**Problem Statement: Consider table Stud(Roll, Att,Status)**

Write a PL/SQL block for following requirement and handle the exceptions. Roll no. of student will be entered by user. Attendance of roll no. entered by user will be checked in Stud table. If attendance is less than 75% then display the message “Term not granted” and set the status in stud table as “D”. Otherwise display message “Term granted” and set the status in stud table as “ND”

**Solution in Oracle:**

```
SQL> create table stud1(roll_no number(5),attendance number(5),status varchar(7));
```

Table created.

```
SQL> select * from stud1;
```

ROLL_NO	ATTENDANCE	STATUS
-----	-----	-----
101	80	
102	65	
103	92	
104	55	
105	68	

```
SQL> set serveroutput on;
```

```
SQL>
```

```
declare
```

```
roll number(10);
```

```
att number(10);
```

```
begin
```

```
roll:=&roll;

select attendance into att from stud1 where roll_no=roll; if
att<75 then

dbms_output.put_line(roll||'is detained');

update stud1 set status='D' where roll_no=roll;

else

dbms_output.put_line(roll||'is not detained');

update stud1 set status='ND' where roll_no=roll;

end if;

exception

when no_data_found then

dbms_output.put_line(roll||'not found');

end;

/
```

Enter value for roll: 102

old 5: roll:=&roll;

new 5: roll:=102;

102is detained

PL/SQL procedure successfully completed.

SQL> /

Enter value for roll: 101

old 5: roll:=&roll;

new 5: roll:=101;

101is not detained

PL/SQL procedure successfully completed.

SQL> /

Enter value for roll: 103

old 5: roll:=&roll;

new 5: roll:=103;

103is not detained

PL/SQL procedure successfully completed.

SQL> /

Enter value for roll: 104

old 5: roll:=&roll;

new 5: roll:=104;

104is detained

PL/SQL procedure successfully completed.

SQL> /

Enter value for roll: 105

old 5: roll:=&roll;

new 5: roll:=105;

105is detained

PL/SQL procedure successfully completed.

SQL> select \* from stud1;

ROLL_NO	ATTENDANCE	STATUS
-----	-----	-----
101	80	ND
102	65	D
103	92	ND
104	55	D

**Conclusion:**

Thus we successfully implemented procedures.



**Aim:** To Study of all types of Cursor (All types: Implicit, Explicit, Cursor FOR Loop, Parameterized Cursor)

**Input:** New roll calls and old roll calls

**Theory:**

Oracle creates a memory area, known as the context area, for processing an SQL statement, which contains all the information needed for processing the statement; for example, the number of rows processed, etc. A **cursor** is a pointer to this context area. PL/SQL controls the context area through a cursor. A cursor holds the rows (one or more) returned by a SQL statement. The set of rows the cursor holds is referred to as the **active set**.

You can name a cursor so that it could be referred to in a program to fetch and process the rows returned by the SQL statement, one at a time. There are two types of cursors –

- Implicit cursors
- Explicit cursors

**Implicit Cursors**

Implicit cursors are automatically created by Oracle whenever an SQL statement is executed, when there is no explicit cursor for the statement. Programmers cannot control the implicit cursors and the information in it.

Whenever a DML statement (INSERT, UPDATE and DELETE) is issued, an implicit cursor is associated with this statement. For INSERT operations, the cursor holds the data that needs to be inserted. For UPDATE and DELETE operations, the cursor identifies the rows that would be affected.

In PL/SQL, you can refer to the most recent implicit cursor as the **SQL cursor**, which always has attributes such as **%FOUND**, **%ISOPEN**, **%NOTFOUND**, and **%ROWCOUNT**. The SQL cursor has additional attributes, **%BULK\_ROWCOUNT** and **%BULK\_EXCEPTIONS**, designed for use with the **FORALL** statement.

**%FOUND**

Returns TRUE if an INSERT, UPDATE, or DELETE statement affected one or more rows or a SELECT INTO statement returned one or more rows. Otherwise, it returns FALSE.

**%NOTFOUND**

The logical opposite of %FOUND. It returns TRUE if an INSERT, UPDATE, or DELETE statement affected no rows, or a SELECT INTO statement returned no rows. Otherwise, it returns FALSE.

**%ISOPEN**

Always returns FALSE for implicit cursors, because Oracle closes the SQL cursor automatically after executing its associated SQL statement.

**%ROWCOUNT**

Returns the number of rows affected by an INSERT, UPDATE, or DELETE statement, or returned by a SELECT INTO statement.

## Explicit Cursors

Explicit cursors are programmer-defined cursors for gaining more control over the **context area**. An explicit cursor should be defined in the declaration section of the PL/SQL Block. It is created on a SELECT Statement which returns more than one row.

The syntax for creating an explicit cursor is –

**CURSOR cursor\_name IS select\_statement;**

Working with an explicit cursor includes the following steps –

- Declaring the cursor for initializing the memory
- Opening the cursor for allocating the memory
- Fetching the cursor for retrieving the data
- Closing the cursor to release the allocated memory

### Declaring the Cursor

Declaring the cursor defines the cursor with a name and the associated SELECT statement. For example –

**CURSOR c\_customers IS**

**SELECT id, name, address FROM customers;**

### Opening the Cursor

Opening the cursor allocates the memory for the cursor and makes it ready for fetching the rows returned by the SQL statement into it. For example, we will open the above defined cursor as follows –

**OPEN c\_customers;**

### Fetching the Cursor

Fetching the cursor involves accessing one row at a time. For example, we will fetch rows from the above-opened cursor as follows –

**FETCH c\_customers INTO c\_id, c\_name, c\_addr;**

### Closing the Cursor

Closing the cursor means releasing the allocated memory. For example, we will close the above-opened cursor as follows –

**CLOSE c\_customers;**

To handle a result set inside a [stored procedure](#), you use a cursor. A cursor allows you to [iterate](#) a set of rows returned by a query and process each row individually.

MySQL cursor is read-only, non-scrollable and asensitive.

- **Read-only:** you cannot update data in the underlying table through the cursor.
- **Non-scrollable:** you can only fetch rows in the order determined by the [SELECT](#) statement. You cannot fetch rows in the reversed order. In addition, you cannot skip rows or jump to a specific row in the result set.
- **Asensitive:** there are two kinds of cursors: asensitive cursor and insensitive cursor. An asensitive cursor points to the actual data, whereas an insensitive cursor uses a temporary copy of the data. An asensitive cursor performs faster than an insensitive cursor because it does not have to make a temporary copy of data. However, any change that made to the data from other connections will affect the data that is being used by an asensitive cursor, therefore, it is safer if you do not update the data that is being used by an asensitive cursor. MySQL cursor is asensitive.

## Problem Statement :

### Cursors: (All types: Implicit, Explicit, Cursor FOR Loop, Parameterized Cursor)

Write a PL/SQL block of code using parameterized Cursor, that will merge the data available in the newly created table N\_RollCall with the data available in the table O\_RollCall. If the data in the first table already exist in the second table then that data should be skipped.

### Solution in MySQL:

Steps are:

- 1) Create new\_roll call and old\_roll call tables
- 2) Insert records in both tables with few records duplication
- 3) create procedure and use cursor to merge above two tables to finalize roll list without duplication.

Assignment code:

```
mysql>create table new_roll(roll int,name varchar(10));
```

```
Query OK, 0 rows affected (0.29 sec)
```

```
mysql> create table old_roll(roll int,name varchar(10));
```

```
Query OK, 0 rows affected (0.28 sec)
```

```
mysql> insert into new_roll values(2,'b')$
```

```
Query OK, 1 row affected (0.05 sec)
```

```
mysql> insert into old_roll values(4,'d')$
```

```
Query OK, 1 row affected (0.05 sec)
```

```
mysql> insert into old_roll values(3,'bcd')$
```

```
Query OK, 1 row affected (0.04 sec)
```

```
mysql> insert into old_roll values(1,'bc')$
```

```
Query OK, 1 row affected (0.04 sec)
```

```
mysql> insert into old_roll values(5,'bch')$
```

```
Query OK, 1 row affected (0.04 sec)
```

```
mysql> insert into new_roll values(5,'bch')$
```

```
Query OK, 1 row affected (0.05 sec)
```

```
mysql> insert into new_roll values(1,'bc')$
```

```
Query OK, 1 row affected (0.04 sec)
```

```
mysql> select * from new_roll$
```

```
+-----+-----+
```

```
| roll | name |
```

```
+-----+-----+
```

```
| 2 | b |
```

```
| 4 | d |
```

```
| 5 | bch |
```

```
| 1 | bc |
```

```
+-----+-----+
```

```
4 rows in set (0.00 sec)
```

```
mysql> select * from old_roll$
```

```
+-----+-----+
```

```
| roll | name |
```

```
+-----+-----+
```

```
| 2 | b |
```

```
| 4 | d |
```

```
| 3 | bcd |
```

```
| 1 | bc |
| 5 | bch |
+-----+-----+
5 rows in set (0.00 sec)
```

```
delimiter $
create procedure roll_list()
begin
declare oldrollnumber int;
declare oldname varchar(10);
declare newrollnumber int;
declare newname varchar(10);
declare done int default false;
declare c1 cursor for select roll,name from old_roll;
declare c2 cursor for select roll,name from new_roll;
declare continue handler for not found set done=true;
open c1;
loop1:loop
fetch c1 into oldrollnumber,oldname;
if done then
leave loop1;
end if;
open c2;
loop2:loop
fetch c2 into newrollnumber,newname;
if done then
insert into new_roll values(oldrollnumber,oldname);
set done=false;
close c2;
leave loop2;
end if;
if oldrollnumber=newrollnumber then
leave loop2;
end if;
end loop;
end loop;
close c1;
end $
```

```
mysql> call roll_list()$
Query OK, 1 row affected (0.04 sec)
```

```
mysql> select * from new_roll$
+-----+-----+
| roll | name |
+-----+-----+
| 2 | b |
| 4 | d |
| 5 | bch |
| 1 | bc |
| 3 | bcd |
+-----+-----+
5 rows in set (0.01 sec)
```

**Problem Statement 2:** The bank manager has decided to activate all those accounts which were previously marked as inactive for performing no transaction in last 365 days. Write a PL/SQ block (using implicit cursor) to update the status of account, display an approximate message based on the no. of rows affected by the update. (Use of %FOUND, %NOTFOUND, %ROWCOUNT)

**Solution in Oracle:**

```
Declare
Rows_affe number(10);
Begin
update bankcursor set status='active'where
status='inactive'; Rows_affe:=(SQL%rowcount);

dbms_output.put_line(Rows_affe||' rows are
affected...');
END;
```

**Solution :**

```
SQL> create table bankcursor(acc_no number(10),status varchar(10));
```

Table created.

```
SQL> select * from bankcursor;
```

ACC_NO	STATUS
-----	-----
101	active
102	inactive
103	inactive
104	active
105	inactive

```
SQL>
```

```
Declare
Rows_affe number(10);
Begin
update bankcursor set status='active'where status='inactive';
Rows_affe:=(SQL%rowcount);
dbms_output.put_line(Rows_affe||' rows are affected...');
END;
/
```

3 rows are affected...

PL/SQL procedure successfully completed.

```
SQL> select * from bankcursor;
```

ACC_NO	STATUS
-----	-----
101	active
102	active
103	active
104	active
105	active

**Problem Statement 3:** Organization has decided to increase the salary of employees by 10% of existing salary, who are having salary less than average salary of organization, Whenever such salary updates takes place, a record for the same is maintained in the increment\_salary table.

EMP (E\_no , Salary)  
increment\_salary(E\_no ,  
Salary) code:

**Solution in Oracle:**

```
Declare
Cursor crsr_sal is select e_no,salary from emp2 where salary<(select avg(salary) from emp2);
me_no emp2.e_no%type;
msalary emp2.salary%type;
```

```
Begin
open crsr_sal;
if crsr_sal%isopen then
loop
fetch crsr_sal into me_no,msalary;
exit when crsr_sal%notfound;
if crsr_sal%found then
update emp2 set salary=salary+(salary*0.1) where
e_no=me_no; select salary into msalary from emp2 where
e_no=me_no; insert into increament_t values(me_no,msalary);
end if;
end loop;
end if;
end;
```

```
SQL> create table emp2(e_no number(10),salary number(10));
```

Table created.

```
SQL> select * from emp2;
```

E_NO	SALARY
-----	-----
101	1000
102	2000
103	113
104	4000

```
SQL> create table increament_t(eno number(10),sal number(10));
```

Table created.

```
SQL>
```

```
Declare
```

```
Cursor crsr_sal is select e_no,salary from emp2 where salary<(select avg(salary)
from emp2);
```

```
me_no emp2.e_no%type;
```

```
msalary emp2.salary%type;
```

```
Begin
```

```
open crsr_sal;
```

```
if crsr_sal%isopen then
```

```
loop
```

```
fetch crsr_sal into me_no,msalary;
```

```
exit when crsr_sal%notfound;
```

```
if crsr_sal%found then
```

```
update emp2 set salary=salary+(salary*0.1) where e_no=me_no; 14
```

```
select salary into msalary from emp2 where e_no=me_no;
```

```
insert into increament_t values(me_no,msalary);
```

```
end if;
```

```
end loop;
```

```
end if;
```

```
end;
```

```
/
```

PL/SQL procedure successfully completed.

```
SQL> select * from emp2;
```

	E_NO	SALARY
	-----	-----
101		1100
	102	2000
	103	113
	104	4000

```
SQL> select * from increament_t;
```

	ENO	SAL
	-----	-----
•		1100
	103	113

### Conclusion:

Thus we successfully implemented procedures.

**Expt. No: 8**

**Study all types of Database Trigger (All Types: Row level and Statement level triggers, Before and After Triggers).**

**Aim:** To Study of PL/SQL Stored Procedure and Stored Function.

**Input:** Students details and marks

**Theory:**

**Procedure:**

- A **subprogram** is a program unit/module that performs a particular task. These subprograms are combined to form larger programs. This is basically called the 'Modular design'. A subprogram can be invoked by another subprogram or program which is called the **calling program**.
- A subprogram can be created –
  - At the schema level
  - Inside a package
  - Inside a PL/SQL block
- At the schema level, subprogram is a **standalone subprogram**. It is created with the CREATE PROCEDURE or the CREATE FUNCTION statement. It is stored in the database and can be deleted with the DROP PROCEDURE or DROP FUNCTION statement.
- A subprogram created inside a package is a **packaged subprogram**. It is stored in the database and can be deleted only when the package is deleted with the DROP PACKAGE statement. We will discuss packages in the chapter '**PL/SQL - Packages**'.
- PL/SQL subprograms are named PL/SQL blocks that can be invoked with a set of parameters. PL/SQL provides two kinds of subprograms –
  - **Functions** – These subprograms return a single value; mainly used to compute and return a value.
  - **Procedures** – These subprograms do not return a value directly; mainly used to perform an action.
- This chapter is going to cover important aspects of a **PL/SQL procedure**. We will discuss **PL/SQL function** in the next chapter.

**Creating a Function:**

A standalone function is created using the **CREATE FUNCTION** statement. The simplified syntax for the **CREATE OR REPLACE PROCEDURE** statement is as follows –

```
CREATE [OR REPLACE] FUNCTION function_name
[(parameter_name [IN | OUT | IN OUT] type [, ...])]
RETURN return_datatype
{IS | AS}
BEGIN
```

```
    < function_body >
```

```
END [function_name];
```

Where,

- *function-name* specifies the name of the function.
- [OR REPLACE] option allows the modification of an existing function.
- The optional parameter list contains name, mode and types of the parameters. IN represents the value that will be passed from outside and OUT represents the <sup>5</sup>e<sup>7</sup> parameter that will be used to return a value outside of the procedure.
- The function must contain a **return** statement.



- The *RETURN* clause specifies the data type you are going to return from the function.
- *function-body* contains the executable part.
- The AS keyword is used instead of the IS keyword for creating a standalone function.

## Calling a Function

While creating a function, you give a definition of what the function has to do. To use a function, you will have to call that function to perform the defined task. When a program calls a function, the program control is transferred to the called function.

A called function performs the defined task and when its return statement is executed or when the **last end statement** is reached, it returns the program control back to the main program.

To call a function, you simply need to pass the required parameters along with the function name and if the function returns a value, then you can store the returned value.

```
DECLARE
  c number(2);
BEGIN
  c := totalCustomers();
  dbms_output.put_line('Total no. of Customers: ' || c);
END;
/
```

## Problem Statement 1: PL/SQL Stored Procedure and Stored Function.

Write a Stored Procedure namely proc\_Grade for the categorization of student. If marks scored by students in examination is  $\leq 1500$  and marks  $\geq 990$  then student will be placed in distinction category if marks scored are between 989 and 900 category is first class, if marks 899 and 825 category is Higher Second Class

Write a PL/SQL block for using procedure created with above requirement.

Stud\_Marks(name, total\_marks) Result(Roll, Name, Class)

## Solution in MySQL:

Steps:

- 1) create stud\_marks and result table with primary and foreign keys
- 2) insert values in stud\_marks
- 3) write and execute PL/SQL procedure for inserting grades in result table

Assignment is as follows:

```
mysql> desc stud_marks;
+-----+-----+-----+-----+-----+-----+
| Field | Type | Null | Key | Default | Extra |
+-----+-----+-----+-----+-----+-----+
| name  | varchar(20) | NO | PRI | NULL | |
| total_marks | int(11) | YES | | NULL | |
```

```
+.....+.....+.....+.....+.....+.....+
```

2 rows in set (0.01 sec)

```
mysql> desc result;
```

```
+.....+.....+.....+.....+.....+.....+
```

```
| Field | Type | Null | Key | Default | Extra |
```

```
+.....+.....+.....+.....+.....+.....+
```

```
| roll | int(11) | YES | | NULL | |
```

```
| class | varchar(10) | YES | | NULL | |
```

```
| name | varchar(20) | YES | MUL | NULL | |
```

```
+.....+.....+.....+.....+.....+.....+
```

3 rows in set (0.00 sec)

```
mysql> insert into stud_marks values('abhijit',1020)$
```

Query OK, 1 row affected (0.15 sec)

```
mysql> insert into stud_marks values('anand',979)$
```

Query OK, 1 row affected (0.04 sec)

```
mysql> insert into stud_marks values('vijay',864)$
```

Query OK, 1 row affected (0.04 sec)

```
mysql> insert into stud_marks values('vikas',755)$
```

Query OK, 1 row affected (0.03 sec)

```
Create procedure proc_grade()
```

```
begin
```

```
declare done int default false;
```

```
declare roll int;
```

```
declare totmarks int;
```

```
declare class varchar(10);
```

```
declare name1 varchar(20);
```

```
declare c1 cursor for select name,total_marks from stud_marks;
```

```
declare continue handler for not found set done=true;
```

```
open c1;
```

```
set roll=1;
```

```
readloop:loop
```

```
fetch c1 into name1,totmarks;
```

```

if done then
leave readloop;
end if;
if totmarks<=1500 and totmarks>=990 then
insert into result values(roll,'dist',name1);
elseif totmarks<=989 and totmarks>=900 then
insert into result values(roll,'first',name1);
elseif totmarks<=899 and totmarks>=825 then
insert into result values(roll,'HSC',name1);
else
insert into result values(roll,'poor',name1);
end if;
set roll=roll+1;
end loop;
end $

```

mysql> call proc\_grade()

Query OK, 0 rows affected (0.38 sec)

mysql> select \* from result\$

```
+.....+.....+.....+
```

```
| roll | class | name |
```

```
+.....+.....+.....+
```

```
| 1 | dist | abhijit |
```

```
| 2 | first | anand |
```

```
| 3 | HSC | vijay |
```

```
| 4 | poor | vikas |
```

```
+.....+.....+.....+
```

4 rows in set (0.00 sec)

---

**Problem Statement 2.** Write a PL/SQL stored Procedure for following requirements and call the procedure in appropriate PL/SQL block.

1. Borrower(Rollin, Name, DateofIssue, NameofBook, Status)

2. Fine(Roll\_no,Date,Amt)

- Accept roll\_no & name of book from user.

- Check the number of days (from date of issue), if days are between 15 to 30 then fine amount will be Rs 5per day.
- If no. of days>30, per day fine will be Rs 50 per day & for days less than 30, Rs. 5 per day.
- After submitting the book, status will change from I to R.
- If condition of fine is true, then details will be stored into fine table

**Solution :**

SQL>create or replace function cal\_fine(diffdate number) return number is

begin

if diffdate<15 then

return 0;

elsif diffdate<30 then

return (5\*(diffdate-15));

else

return (50\*(diffdate-30)+5\*(15));end if;

end;

/

-----

SQL> Declare

troll\_no varchar(5);

tdays number(5);

tdate date;

diffdate number(5);

begin

troll\_no := '&troll\_no';

select to\_date(sysdate,'DD-MM-YY')"Now" into tdate from dual;

select ((select to\_date(sysdate,'DD-MM-YY')"Now" from dual)-dateofissue) into diffdate

from Borrower

where roll\_no=troll\_no;

insert into Fine values(troll\_no,tdate,cal\_fine(diffdate));

update borrower set status = 'R' where roll\_no=troll\_no;

End;

/

```

create function cal_fss(diffdate number) return number is
begin
if diffdate<15 then
return 0;
elsif diffdate<30 then
return (5*(diffdate-15));
else
return (50*(diffdate-30)+5*(15));
end if;
end ;

```

```

create table borrower(rollno number primary key, name varchar2(20), dateofissue date, nameofbook
varchar2(20), status varchar2(20));
create table fine(rollno number, foreign key(rollno) references borrower(rollno), returndate date, amount
number);

```

```

insert into borrower values(1,'abc',date '2021-06-01','SEPM','I');
insert into borrower values(2,'xyz',date '2021-05-01','OOP','I');
insert into borrower values(3,'pqr',date '2021-06-15','DBMS','I');
insert into borrower values(4,'def',date '2021-06-30','DSA','I');
insert into borrower values(5,'lmn',date '2021-07-05','ADS','I');

```

```

create procedure calc_fine_lib3(roll number) is
troll_no number(5);
tdays number(5);
tdate date;
diffdate number(5);
begin
troll_no := roll;
select to_date(sysdate,'DD-MM-YY')"Now" into tdate from dual;
select ((select to_date(sysdate,'DD-MM-YY')"Now" from dual)-dateofissue) into diffdate from Borrower
where rollno=troll_no;
insert into Fine values(troll_no,tdate,cal_fss(diffdate));
update borrower set status = 'R' where rollno=troll_no;
End;

```

### **Conclusion:**

Thus we have successfully implemented PL/SQL stored procedures.

**Aim:** To Study all types of Database Trigger (All Types: Row level and Statement level triggers, Before and After Triggers).

**Input:** Student library books information

**Theory:**

- Triggers are stored programs, which are automatically executed or fired when some events occur. Triggers are, in fact, written to be executed in response to any of the following events –
- A **database manipulation (DML)** statement (DELETE, INSERT, or UPDATE)
- A **database definition (DDL)** statement (CREATE, ALTER, or DROP).
- A **database operation** (SERVERERROR, LOGON, LOGOFF, STARTUP, or SHUTDOWN).
- Triggers can be defined on the table, view, schema, or database with which the event is associated.

**Benefits of Triggers**

- Triggers can be written for the following purposes –
- Generating some derived column values automatically
- Enforcing referential integrity
- Event logging and storing information on table access
- Auditing
- Synchronous replication of tables
- Imposing security authorizations
- Preventing invalid transactions

**Creating Triggers**

- The syntax for creating a trigger is –  
CREATE [OR REPLACE ] TRIGGER trigger\_name  
{ BEFORE | AFTER | INSTEAD OF }  
{ INSERT [OR] | UPDATE [OR] | DELETE }  
[OF col\_name]  
ON table\_name  
[REFERENCING OLD AS o NEW AS n]  
[FOR EACH ROW]  
WHEN (condition)  
DECLARE  
    Declaration-statements  
BEGIN  
    Executable-statements  
EXCEPTION  
    Exception-handling-statements  
END;

Where,

- CREATE [OR REPLACE] TRIGGER trigger\_name – Creates or replaces an existing trigger with the *trigger\_name*.
- {BEFORE | AFTER | INSTEAD OF} – This specifies when the trigger will be executed. The INSTEAD OF clause is used for creating trigger on a view.
- {INSERT [OR] | UPDATE [OR] | DELETE} – This specifies the DML operation.
- [OF col\_name] – This specifies the column name that will be updated.
- [ON table\_name] – This specifies the name of the table associated with the trigger.
- [REFERENCING OLD AS o NEW AS n] – This allows you to refer new and old values for various DML statements, such as INSERT, UPDATE, and DELETE.
- [FOR EACH ROW] – This specifies a row-level trigger, i.e., the trigger will be executed for each row being affected. Otherwise the trigger will execute just once when the SQL statement is executed, which is called a table level trigger.
- WHEN (condition) – This provides a condition for rows for which the trigger would fire. This clause is valid only for row-level triggers.

## Problem Statement:

**Database Trigger (All Types: Row level and Statement level triggers, Before and After Triggers).**

Write a database trigger on Library table. The System should keep track of the records that are being updated or deleted. The old value of updated or deleted records should be added in Library\_Audit table.

## Solution in MySQL:

Steps are:

- 1) Create lib\_audit and lib\_audit2 tables
- 2) Insert records in lib\_audit
- 3) create trigger for before update and before delete on lib\_audit.

//Trigger for delete on lib\_audit

```
mysql> create table lib_audit(bookid int,bookname varchar(20),price int)$  
Query OK, 0 rows affected (0.58 sec)
```

```
mysql> create table lib_audit2(bookid int,bookname varchar(20),price int)$  
Query OK, 0 rows affected (0.36 sec)
```

```
mysql> Create trigger before_delete_lib_audit before delete on lib_audit for each row  
begin  
insert into lib_audit2 values(old.bookid,old.bookname,old.price);  
end$  
Query OK, 0 rows affected (0.13 sec)
```

```
mysql> insert into lib_audit values(1,'ab',100)$  
Query OK, 1 row affected (0.05 sec)  
mysql> insert into lib_audit values(2,'cd',10)$  
Query OK, 1 row affected (0.05 sec)  
mysql> insert into lib_audit values(3,'dg',101)$  
Query OK, 1 row affected (0.05 sec)
```

```
mysql> select * from lib_audit$
```

```
+-----+-----+-----+
| bookid | bookname | price |
+-----+-----+-----+
| 1 | ab | 100 |
| 2 | cd | 10 |
| 3 | dg | 101 |
+-----+-----+-----+
3 rows in set (0.00 sec)
```

```
mysql> select * from lib_audit2$
```

```
Empty set (0.00 sec)
```

```
mysql> delete from lib_audit where bookid=1$
```

```
Query OK, 1 row affected (0.14 sec)
```

```
mysql> select * from lib_audit$
```

```
+-----+-----+-----+
| bookid | bookname | price |
+-----+-----+-----+
| 2 | cd | 10 |
| 3 | dg | 101 |
+-----+-----+-----+
2 rows in set (0.00 sec)
```

```
mysql> select * from lib_audit2$
```

```
+-----+-----+-----+
| bookid | bookname | price |
+-----+-----+-----+
| 1 | ab | 100 |
+-----+-----+-----+
1 row in set (0.00 sec)
```

```
mysql> delete from lib_audit where bookid=3$
```

```
Query OK, 1 row affected (0.04 sec)
```

```
mysql> select * from lib_audit2$
```

```
+-----+-----+-----+
| bookid | bookname | price |
+-----+-----+-----+
| 1 | ab | 100 |
| 3 | dg | 101 |
+-----+-----+-----+
2 rows in set (0.00 sec)
```

```
//Trigger for update on lib_audit
```

```
mysql> Create trigger before_update_lib_audit before update on lib_audit for each row
```

```
begin
```

```
insert into lib_audit2 values(old.bookid,old.bookname,old.price);
```

```
end$
```

```
mysql> update lib_audit set bookname='xy' where bookid=2$
```

```
Query OK, 1 row affected (0.07 sec)
```

```
Rows matched: 1 Changed: 1 Warnings: 0
```



```
mysql> select * from lib_audit$
+-----+-----+-----+
| bookid | bookname | price |
+-----+-----+-----+
| 2 | xy | 10 |
+-----+-----+-----+
1 row in set (0.00 sec)
mysql> select * from lib_audit2$
+-----+-----+-----+
| bookid | bookname | price |
+-----+-----+-----+
| 1 | ab | 100 |
| 3 | dg | 101 |
| 2 | cd | 10 |
+-----+-----+-----+
3 rows in set (0.00 sec)
```

---

**Problem Statement :** Write a update, delete trigger on client mstr table. The System should keep track of the records that ARE BEING updated or deleted. The old value of updated or deleted records should be added in audit trade table. (separate implementation using both row and statement triggers).

#### **Solution in Oracle:**

##### **Row trigger:**

```
SQL> create or replace trigger t1 after update or delete on client_master 2
for each row
declare
op varchar(10);
begin
if updating then
op:='update';
end if;
if deleting then
op:='Delete'; end if;
into stat values(:old.id,op);
insert into audit_trade values(:old.id,:old.cname);
dbms_output.put_line('Details updated to stat and audit_trade table');
end;
/ Trigger created.
```

##### **Statement Trigger:**

```
SQL> create or replace trigger t1 after update or delete on client_master 2
for each row
declare
op varchar(10);
begin
if updating then
op:='update';
end if;
if deleting then
op:='Delete'; end if;
into stat values("",op);
insert into audit_trade values(:old.id,:old.cname);
dbms_output.put_line('Details updated to stat and audit_trade table'); end;
/ Trigger created.
```

**Conclusion:** Thus we have successfully implemented trigger.

## Virtual LAB Links:

### 1. Lab Name: Database Lab

**Link of the Virtual Lab:** <http://vlabs.iitb.ac.in/vlabs-dev/labs/dblab/index.php>

## Database Lab

Data Definition Language(DDL) Statements: (Create table, Alter table, Drop table)

Data Manipulation Language(DML) Statements

Data Query Language(DQL) Statements: (Select statement with operations like Where clause, Order by, Logical operators, Scalar functions and Aggregate functions)

Transaction Control Language(TCL) statements: (Commit(make changes permanent), Rollback (undo))

Describe statement: To view the structure of the table created

The screenshot displays the IIT Bombay Database Lab interface. At the top, there is a navigation bar with the IIT Bombay logo and links for HOME, LABS, and GITLAB. Below this, a breadcrumb trail reads: Database Lab > Data Definition Language(DDL) Statements: (Create table, Alter table, Drop table) > Post Test. The main content area is titled "Data Definition Language(DDL) Statements: (Create table, Alter table, Drop table)" and contains a "Post Test" section. The test instructions state: "Use simulator to complete below task." The test consists of three questions:

- Which of the following command is used to change the structure of the table?
  - ☐ Create table
  - ☐ Alter table
  - ☐ Drop table
  - ☐ All of the above
- The \_\_\_\_\_ command permanently removes the table.
  - ☐ Delete table
  - ☐ Drop table
  - ☐ Remove table
  - ☐ All of the above
- The data types supported in this database are:
  - ☐ Varchar
  - ☐ Number
  - ☐ Date
  - ☐ All of the above

A left sidebar contains a menu with the following items: Aim, Theory, Pre Test, Procedure, Simulation, Post Test (highlighted), and References.

**Aim:** To implement MYSQL/Oracle database connectivity with PHP/python/Java Implement Database navigation operations (add, delete, edit,) using ODBC/JDBC.

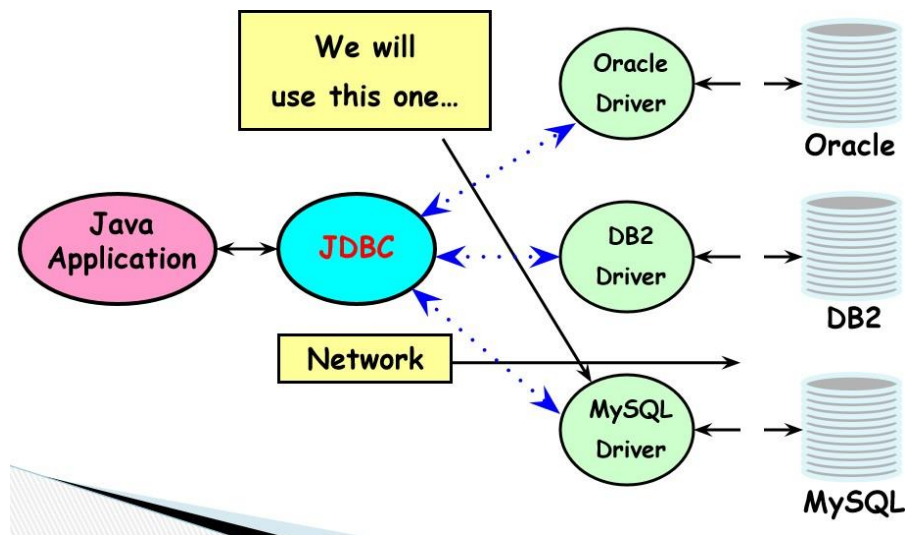
**Input:** Student library books information

**Theory:** Introduction to JDBC:

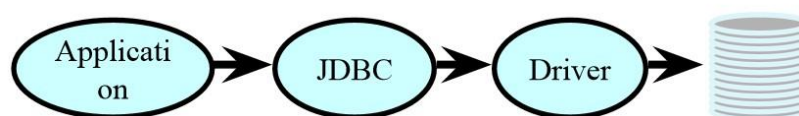
JDBC is used for accessing databases from Java applications Information is transferred from relations to objects and vice-versa

- databases optimized for searching/indexing
- objects optimized for engineering/flexibility

**JDBC architecture:**



## JDBC Architecture (cont.)



Java code calls JDBC library JDBC loads a driver Driver talks to a particular database An application can work with several databases by using all corresponding drivers Ideal: can change database engines without changing any application code (not always in practice)

## Common JDBC components:

**Driver Manager:** This class manages a list of database drivers. Matches connection requests from the java application with the proper database driver using communication subprotocol. The first driver that recognizes a certain subprotocol under JDBC will be used to establish a database Connection.

**Driver:** This interface handles the communications with the database server. You will interact directly with Driver objects very rarely. Instead, you use DriverManager objects, which manages objects of this type. It also abstracts the details associated with working with Driver objects

**Connection :** This interface with all methods for contacting a database. The connection object represents communication context, i.e., all communication with database is through connection object only.

**Statement :** You use objects created from this interface to submit the SQL statements to the database. Some derived interfaces accept parameters in addition to executing stored procedures.

**ResultSet:** These objects hold data retrieved from a database after you execute an SQL query using Statement objects. It acts as an iterator to allow you to move through its data.

**SQLException:** This class handles any errors that occur in a database application.

## JDBC SQL Syntax:

Structured Query Language (SQL) is a standardized language that allows you to perform operations on a database, such as creating entries, reading content, updating content, and deleting entries. This tutorial gives an overview of SQL, which is a pre-requisite to understand JDBC concepts. This tutorial gives you enough SQL to be able to Create, Read, Update, and Delete (often referred to as CRUD operations) data from a database.

### Create database:

```
SQL> CREATE DATABASE DATABASE_NAME;
```

Example: The following SQL statement creates a Database named EMP: SQL> CREATE DATABASE EMP;

### Drop database:

```
SQL> DROP DATABASE DATABASE_NAME;
```

### Create table:

```
SQL> CREATE TABLE table_name ( column_name column_data_type, column_name column_data_type, column_name column_data_type ... );
```

### Insert data:

```
SQL> INSERT INTO table_name VALUES (column1, column2, ...);
```

**Select data:**

SQL> SELECT column\_name, column\_name, ... FROM table\_name WHERE conditions;

**Update data:**

SQL> UPDATE table\_name SET column\_name = value, column\_name = value, ... WHERE conditions;

**Delete data:**

SQL> DELETE FROM table\_name WHERE conditions;

**Step for creating JDBC application:**

There are following steps involved in building a JDBC application:

1. Import the packages. Requires that you include the packages containing the JDBC classes needed for database programming. Most often, using `import java.sql.*` ; will suffice.
2. Register the JDBC driver. Requires that you initialize a driver so you can open a communications channel with the database.
3. Open a connection . Requires using the DriverManager. `getConnection()` method to create a Connection object, which represents a physical connection with the database.

JDBC Driver is a software component that enables java application to interact with the database. There are 4 types of JDBC drivers: 1. JDBC-ODBC bridge driver

2. Native-API driver (partially java driver)
3. Network Protocol driver (fully java driver)
4. Thin driver (fully java driver)

**Six steps:**

1. Load the driver
2. Define the connection URL
3. Establish the connection
4. Create a Statement object
5. Execute a query using the Statement, Process the result
6. Close the connection

**Conclusion: Thus through JDBC connection the operations are performed.**