Department of Computer Science and Engineering

II B. Tech - II Semester

Sub: Data Base Management Systems Laboratory Manual

Subject Code : 50520

Academic Year 2021-22

Regulations: MR18

Malla Reddy Engineering College

(Autonomous)
(An UGC Autonomous Institution, Approved by AICTE and Affiliated to JNTUH Hyderabad,
Recognized under 2(f) &12 (B) of UGC Act 1956, Accredited by NAAC with 'A' Grade (II Cycle)
Maisammaguda, Dhulapally (Post Via Kompally), Secunderabad-500 100
www.mrec.ac.in E-mail: principal@mrec.ac.in

2021-2022

Malla Reddy Engineering College
(Autonomous)
Maisammaguda, Dhulapally (Post Via Kompally), Secunderabad-500100

Institute Vision

To be a premier centre of professional education and research, offering quality programs in a socio - economic and ethical ambience.

Institute Mission

1. **To impart knowledge of advanced technologies using state-of-the-art infrastructural facilities.**
2. **To inculcate innovation and best practices in education, training and research.**
3. **To meet changing socio-economic needs in an ethical ambience.**

Malla Reddy Engineering College
(Autonomous)
Department of Computer Science and Engineering

Department Vision

To attain global standards in Computer Science and Engineering education, training and research to meet thegrowing needs of the industry with socio-economic and ethical considerations.

Department Mission

1. **To impart knowledge of advanced technologies using state-of-the-art infrastructural facilities.**
2. **To inculcate innovation and best practices in education, training and research.show tables**
3. **To meet changing socio-economic needs in an ethical ambience.**

Programme Educational Objectives (PEOs)

PEO I:

Impart with a sound knowledge in scientific and engineering technologies necessary to formulate, analyze, design and implement solutions to computer technology related problems.

PEO II:

Carry out research in frontier areas of computer science and engineering with the capacity to learn independently throughout life to develop new technologies.

PEO III:

Train to exhibit effective technical, communication and project management skills in their profession and follow ethical practices.

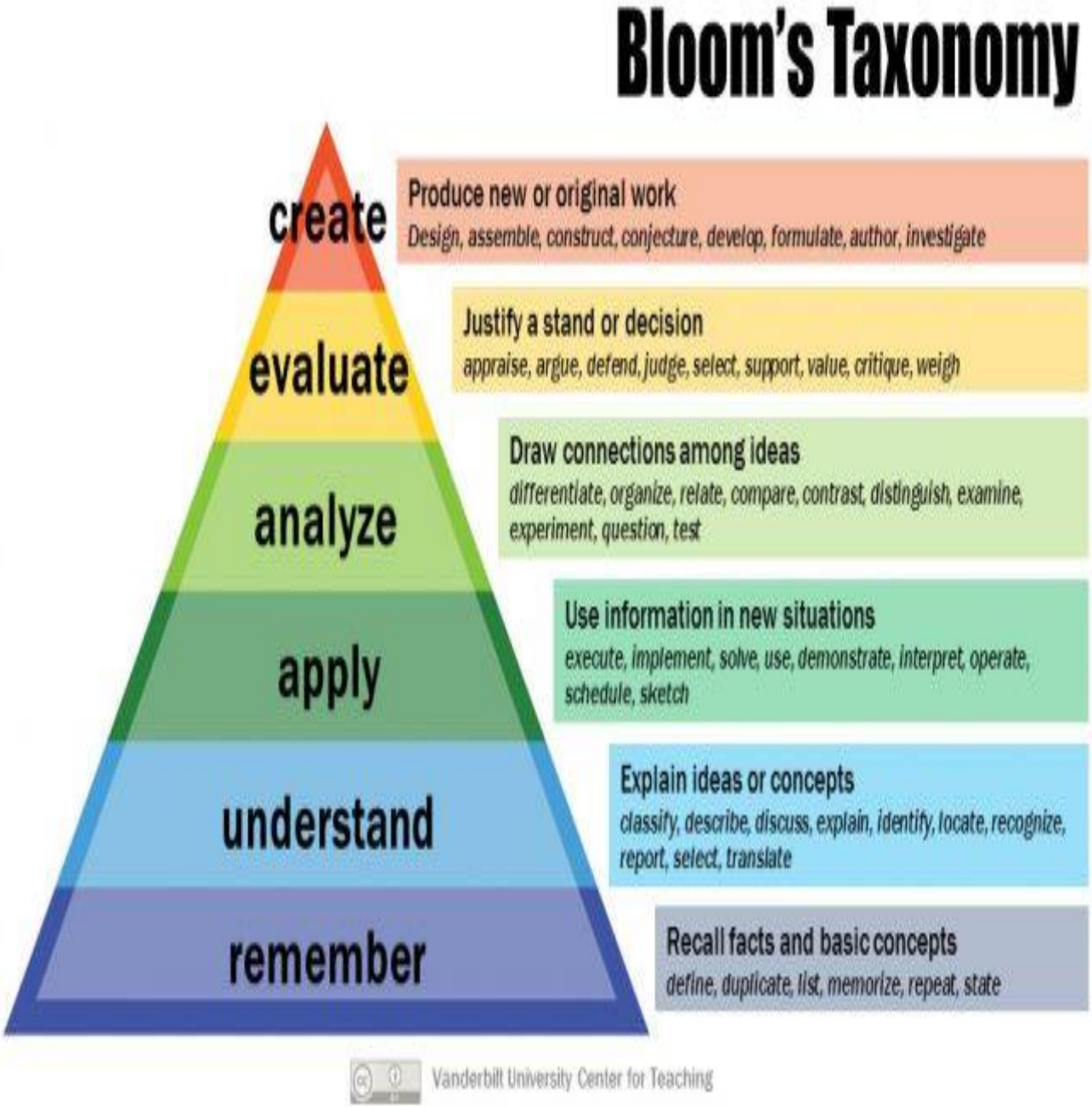
PEO IV:

Posses leadership and team working skills to become a visionary and an inspirational leader and entrepreneur.

REVISED Bloom's Taxonomy Action Verbs

Definitions	I. Remembering	II. Understanding	III. Applying	IV. Analyzing	V. Evaluating	VI. Creating
Bloom's Definition	Exhibit memory of previously learned material by recalling facts, terms, basic concepts, and answers.	Demonstrate understanding of facts and ideas by organizing, comparing, translating, interpreting, giving descriptions, and stating main ideas.	Solve problems to new situations by applying acquired knowledge, facts, techniques and rules in a different way.	Examine and break information into parts by identifying motives or causes. Make inferences and find evidence to support generalizations.	Present and defend opinions by making judgments about information, validity of ideas, or quality of work based on a set of criteria.	Compile information together in a different way by combining elements in a new pattern or proposing alternative solutions.
Verbs	• Choose • Define • Find • How • Label • List • Match • Name • Omit • Recall • Relate • Select • Show • Spell • Tell • What • When • Where • Which • Who • Why	• Classify • Compare • Contrast • Demonstrate • Explain • Extend • Illustrate • Infer • Interpret • Outline • Relate • Rephrase • Show • Summarize • Translate	• Apply • Build • Choose • Construct • Develop • Experiment with • Identify • Interview • Make use of • Model • Organize • Plan • Select • Solve • Utilize	• Analyze • Assume • Categorize • Classify • Compare • Conclusion • Contrast • Discover • Dissect • Distinguish • Divide • Examine • Function • Inference • Inspect • List • Motive • Relationships • Simplify • Survey • Take part in • Test for • Theme	• Agree • Appraise • Assess • Award • Choose • Compare • Conclude • Criteria • Criticize • Decide • Deduct • Defend • Determine • Disprove • Estimate • Evaluate • Explain • Importance • Influence • Interpret • Judge • Justify • Mark • Measure • Opinion • Perceive • Prioritize • Prove • Rate • Recommend • Rule on • Select • Support • Value	• Adapt • Build • Change • Choose • Combine • Compile • Compose • Construct • Create • Delete • Design • Develop • Discuss • Elaborate • Estimate • Formulate • Happen • Imagine • Improve • Invent • Make up • Maximize • Minimize • Modify • Original • Originate • Plan • Predict • Propose • Solution • Solve • Suppose • Test • Theory

Anderson, L. W., & Krathwohl, D. R. (2001). A taxonomy for learning, teaching, and assessing, Abridged Edition. Boston, MA: Allyn and Bacon.


Bloom's Taxonomy
create
Produce new or original work
Design, assemble, construct, conjecture, develop, formulate, author, investigate
evaluate
Justify a stand or decision
appraise, argue, defend, judge, select, support, value, critique, weigh
analyze
Draw connections among ideas
differentiate, organize, relate, compare, contrast, distinguish, examine, experiment, question, test
apply
Use information in new situations
execute, implement, solve, use, demonstrate, interpret, operate, schedule, sketch
understand
Explain ideas or concepts
classify, describe, discuss, explain, identify, locate, recognize, report, select, translate
remember
Recall facts and basic concepts
define, duplicate, list, memorize, repeat, state
Vanderbilt University Center for Teaching

Malla Reddy Engineering College
(Autonomous)

Department of Computer Science and Engineering

PROGRAM OUTCOMES (POs)

PO1. **Engineering knowledge**: Apply the knowledge of mathematics, science, engineering fundamentals, and an engineering specialization to the solution of complex engineering problems.

PO2. **Problem analysis**: Identify, formulate, review research literature, and analyze complex engineering problems reaching substantiated conclusions using first principles of mathematics, natural sciences, and engineering sciences.

PO3. **Design/development of solutions**: Design solutions for complex engineering problems and design system components or processes that meet the specified needs with appropriate consideration for the public health and safety, and the cultural, societal, and environmental considerations.

PO4. **Conduct investigations of complex problems**: Useresearch-based knowledge and research methods including design of experiments, analysis and interpretation of data, and synthesis of the information to provide valid conclusions.

PO5. **Modern tool usage**: Create, select, and apply appropriate techniques, resources, and modern engineering and IT tools including prediction and modeling to complex engineering activities with an understanding of the limitations.

PO6. **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 engineering practice.

PO7. **Environment and sustainability**: Understand the impact of the professional engineering solutions in societal and environmental contexts, and demonstrate the knowledge of, and need for sustainable development.

PO8. **Ethics**: Apply ethical principles and commit to professional ethics and responsibilities and norms of the engineering practice.

PO9. **Individual and team work**: Function effectively as an individual, and as a member or leader in diverse teams, and in multidisciplinary settings.

PO10. **Communication**: Communicate effectively on complex engineering activities with the engineering community and with society at large, such as, being able to comprehend and write effective reports and design documentation, make effective presentations, and give and receive clear instructions.

PO11. **Project management and finance**: Demonstrate knowledge and understanding of the engineering and management principles and apply these to one's own work, as a member and leader in a team, to manage projects and in multidisciplinary environments.

PO12. **Life-long learning**: Recognize the need for, and have the preparation and ability to engage in independent and life-long learning in the broadest context of technological change.

PROGRAMME SPECIFIC OUTCOMES (PSOs)

PSO1: Apply the knowledge gained during the course of the program from mathematics, basics Computing, Basic Sciences and all computer science courses in particular to identify, formulate and solve real life complex engineering problems faced in industries and/or during research work with due consideration for the public health and safety, in the context of cultural, societal, and environmental situations.

PSO2: provide socially acceptable technical solutions to complex computer science engineering problem with the application of modern and appropriate techniques for sustainable development relevant to professional engineering practice.

PSO3: Comprehend and write effective project in multidisciplinary environment in the context of changing technologies

Department of Computer Science and Engineering
LESSON PLAN
Academic Year 2021-2022

Department :	**Computer Science and Engineering**		
Course Title :	**Data Base Management Systems Lab**	**Course Code**	A0520
Year / Semester :	**II / II**	**Sections**	CSE(A,B,C,D)
Category :	Core Course		
Prerequisites Knowledge	Nil		
Duration :	One semester	**Credit Units :**	2
Class / Laboratory Schedule :	0/1/2 [L T P]		
Curriculum gap :		Matching POs and PSOs	
		PO: PO1, PO2, PO3, PO4, PSO: PSO1, PSO2,PSO3	
Course Objectives:	This course enables the students to practice the concepts learnt in the subject DBMS by developing a database for an example project. 1. The student is expected to practice the designing a database system using ER diagram, developing and querying a relational database using normalization techniques in the context of example database. 2. Students are expected to Learn SQL basics for data definition and data manipulation using "MySQL" database. 3. Students are expected to apply the learn developing database applications using procedures, cursors and triggers.		
Course Outcomes:	At the end of the course, students will be able to 1. Design database schema for a given application and apply normalization 2. Acquire skills in using SQL commands for data definition and data manipulation. 3. Develop solutions for database applications using procedures, cursors and triggers		
Texts & References: ***(*recommended text book(s))***	**Textbooks:** 1. Database Management Systems, Raghurama Krishnan, Johannes Gehrke, Tata Mc Graw Hill 3rd Edition 2. Database System Concepts, Silberschatz, Korth, Mc Graw hill, V edition. **References:** 1. Database Systems design, Implementation, and Management, Peter Rob & Carlos Coronel, 7th Edition. 2. SQL The Complete Reference, James R. Groff, Paul N. Weinberg, 3rd Edition, 3. Oracle for Professionals, The X Team, S. Shah and V. Shah, SPD. 4. Database Systems Using Oracle: A Simplified guide to SQL and PL/SQL, Shah, PHI.		
	E-RESOURCES:		
Student Assessments:	• Lab Record • Day to day evaluation		
Outcome Assessment:	• Internal I and II • Final examination		

S. No	Date	Exp. No.	Details of experiment	Actual Date of completion	Remarks
1		1	Railway Reservation System -(Redesigning IRCTC database) **Train** (train Integer, name, source, destination, start_time, reach_time, traveltime, distance, class, days, type) **Ticket** (PNRNo, Transactionid, from_station, To_station, date_of_journey, class date_of_booking, total_ticket_fare, train integer) **Passenger** (PNR No, Serial no, Name, Age, Reservation_status) **Train_Route**(Train_No, route_no, station_code, name, arrival_time, depart_time, distance, day) **Train_Ticket_fare**(Train_No, class, base_fare, reservation_charge, superfast_charge, other_charge, tatkal_charge, service_tax)		
			Create all the tables specified above. Make underlined columns as primary key.(use integer, integer(m,n), varchar(n), date, time, timestamp data types appropriately) Insert atleast 5 rows to each table. (Check www.irctc.co.in website for actual data) 1. Use Interactive insertion for inserting rows to the table. 2. Use ADT (varray) for class and days column in Train table.		
2		2	Write simple DDL/DML Queries to 1. Remove all the rows from Passenger table permanently. 2. Change the name of the Passenger table to Passenger_Details. 3. List all train details. 4. List all passenger details. 5. Give a list of trains in ascending order of integer. 6. List the senior citizen passengers details. 7. List the station names where code starts with 'M'. 8. List the trains details within a range of integers. 9. Change the super fast charge value in train fare as zero, if it is null. 10. List the passenger names whose tickets are not confirmed. 11. List the base_fare of all AC coaches available in each train. Find the ticket details where transaction id is not known. 1) Use Interactive updation for updating the seat no for particular PNR NO. 2) Find the train names that are from Secunderabad to Mumbai, but do not have the source or destination in its name. 3) Find the train details that are on Thursday (Use the ADT column created).		
3		3	Create (Alter table to add constraint) the necessary foreign keys by identifying the relationships in the table. 1) Add a suitable constraint to train table to always have train no in the range 10001 to 99999. 2) Add a suitable constraint for the column of station name, so that does not take duplicates. 3) Change the data type of arrival time, depart time (date -> timestamp or timestamp to date), and do the necessary process for updating the table with new values. 4) Add a suitable constraint for the class column that it should take values only as 1A, 2A, 3A, SL, C.		

			5) Add a not null constraint for the column distance in train_route.		
4		4	Use SQL PLUS functions to. 1. Find the passengers whose date of journey is one month from today. 2. Print the train names in upper case. 3. Print the passenger names with left padding character. 4. Print the station codes replacing K with M. 5. Translate all the LC in class column (Train_fare) to POT and display. 6. Display the fare details of all trains, if any value is ZERO, print as NULL value. 7. Display the pnrno and transaction id, if transaction id is null, print 'not generated'. 8. Print the date_of_jounrney in the format '27th November 2010'. 9. Find the maximum fare (total fare). 10. Find the average age of passengers in one ticket. 11. Find the maximum length of station name available in the database. 12. Print the fare amount of the passengers as rounded value. 13. Add the column halt time to train route. 14. Update values to it from arrival time and depart time. High Level: 15. Update values to arrival time and depart time using conversion functions. 16. Display the arrival time, depart time in the format HH:MI (24 hours and minutes).		
5		5	Write Queries to. Use SET Operators 1. Find the train integers for which reservation have not yet been made. 2. Find the train names that donot have a first AC class coach. 3. Print all the PNR nos available in the database. 4. Find passenger names who have booked to 'Pune'. Use Nested Query(in Operators) 1. Find the train names that stop in 'Warangal'. 2. Find the train names that are superfast and the service tax is zero. 3. Find the Passenger name who have booked for the train that starts from 'Secunderabad'. 4. Find the trains names that have all the AC coaches and the base fare is less than 3000 for each case.		
6		6	Use Join Query 1. Find the train names that stop in 'Warangal'. 2. Find the train names that are superfast and the service tax is zero. 3. Find the Passenger name (and train name) who have booked for the train that starts from 'Secunderabad'. 4. Display the trains names, each type of class and the total fare for each type of class. 5. Display all the train details and the ticket details (if booked any). 6. Create a sequence to provide values for the PNR no. 7. Write a query for full outer join using any of the tables above. Write Queries to.		

			Use Coorelated (and nested) Query 1. Find the train names for which ten tickets have been reserved. 2. Find the trains that have more than ten substations. 3. Find the passengers who do not pass through 'Kachiguda'. 4. Find passengers who have booked for super fast trains.		
7			**Lab Internal-1**		
8		7	Complex queries (use group by /group by having/join/nested) 1. Take the start station code and end station code and display the train details. 2. List the train names and the integer of sub stations it has. 3. List the stations where all types of trains stop. 4. List the trains names that has atleast four bookings. 5. Create a table cancellation history (Insert values from ticket and passenger table). 6. Create a table for all the train integers and class available in train_ticket_fare with total seats. 1. Find the station name that has highest integer of trains stopping at.		
9		8	Write a simple PL/SQL block to. 1. Print the fibonacci series. 2. Print the factorial of a given integer. 3. Print 'NOT confirmed' based on the reservation status, of a particular passenger. 4. Print the total seats available for a particular train and for a particular class.		
10		9	Write a cursor for the following. 1. Retrieve the passenger details for —X train integer and given journey date. 2. Display the train name (once) and the substation names. 3. Display the fare details of a particular train(use basic exceptions) 4. Write a cursor to update the reservation status of the passengers (generate seat integer, if seats have reached maximum, put waiting list integer (30% of total seats), if waiting list integer reaches maximum, put PQWL (10%of total seats), RAC-20%)		
11		10	1. Write a PL/SQL procedure to. a. List the details of passengers who has reserved next to —Mr. X. b. PNR No. of a passengers for a given source and a destination. 2. Write a PL/SQL function to. a. Get the PNRNo and return the total ticket fare. b. Get the Passenger name, train no and return the total journey time in hours and minutes.		
12		11	Write a Trigger for the following: 1) When a passenger cancels a ticket, do the necessary process and update the cancellation history table. 2) When train integer is changed, update it in referencing tables. 3) When a passenger record is inserted reservation status should be automatically updated.		
13		12	1) Use TCL commands for your transactions. (commit, rollback, savepoint)		

			2) Create a role named 'clerk', and give permission for him to select only the trains starting from 'Warangal' along with fare details. 3) Create a nested table containing trainno, name, source, destination and passengers who have booked for it (PNR no, sno, name, age). Find the passengers whose name start with 'S' and train starts from 'Warangal'		
14			**Lab Internal-2**		

FACULTY IN-CHARGE **HOD/CSE**

CO- PO, PSO Mapping (3/2/1 indicates strength of correlation) 3-Strong, 2-Medium, 1-Weak															
COs	Programme Outcomes (POs)												PSOs		
	PO1	PO2	PO3	PO4	PO5	PO6	PO7	PO8	PO9	PO10	PO11	PO12	PSO1	PSO2	PSO3
CO1	2	-	-												
CO2	-	2	2										1	2	
CO3	2	2	2	3										2	2

2020-21 Onwards (MR-20)	MALLA REDDY ENGINEERING COLLEGE (Autonomous)	B.Tech. IV Semester		
Code: A0520	Database Management Systems Lab (Common for CSE, CSE (Cyber Security), CSE (AI and ML), CSE (DS), CSE (IOT) and IT)	L	T	P
Credits: 2		-	1	2

Co-requisites

1. Co-requisite of course "Database Management Systems"

Course Objectives:

This course enables the students to practice the concepts learnt in the subject DBMS by developing a database for an example project.

4. The student is expected to practice the designing a database system using ER diagram, developing and querying a relational database using normalization techniques in the context of example database.
5. Students are expected to Learn SQL basics for data definition and data manipulation using "MySQL" database.
6. Students are expected to apply the learn developing database applications using procedures, cursors and triggers.

List of experiments:

1	Railway Reservation System -(Redesigning IRCTC database) **Train** (train Integer, name, source, destination, start_time, reach_time, traveltime, distance, class, days, type) **Ticket** (PNRNo, Transactionid, from_station, To_station, date_of_journey, class date_of_booking, total_ticket_fare, train integer) **Passenger** (PNR No, Serial no, Name, Age, Reservation_status) **Train_Route**(Train_No, route_no, station_code, name, arrival_time, depart_time, distance, day) **Train_Ticket_fare**(Train_No, class, base_fare, reservation_charge, superfast_charge, other_charge, tatkal_charge, service_tax)
	Create all the tables specified above. Make underlined columns as primary key.(use integer, integer(m,n), varchar(n), date, time, timestamp data types appropriately) Insert atleast 5 rows to each table. (Check www.irctc.co.in website for actual data) 1. Use Interactive insertion for inserting rows to the table. 2. Use ADT (varray) for class and days column in Train table.
2	Write simple DDL/DML Queries to 1. Remove all the rows from Passenger table permanently. 2. Change the name of the Passenger table to Passenger_Details. 3. List all train details. 4. List all passenger details. 5. Give a list of trains in ascending order of integer. 6. List the senior citizen passengers details. 7. List the station names where code starts with 'M'. 8. List the trains details within a range of integers. 9. Change the super fast charge value in train fare as zero, if it is null. 10. List the passenger names whose tickets are not confirmed. 11. List the base_fare of all AC coaches available in each train. Find the ticket details where transaction id is not known. 1) Use Interactive updation for updating the seat no for particular PNR NO.

	2) Find the train names that are from Secunderabad to Mumbai, but do not have the source or destination in its name. 3) Find the train details that are on Thursday (Use the ADT column created).
3	Create (Alter table to add constraint) the necessary foreign keys by identifying the relationships in the table. 1) Add a suitable constraint to train table to always have train no in the range 10001 to 99999. 2) Add a suitable constraint for the column of station name, so that does not take duplicates. 3) Change the data type of arrival time, depart time (date -> timestamp or timestamp to date), and do the necessary process for updating the table with new values. 4) Add a suitable constraint for the class column that it should take values only as 1A, 2A, 3A, SL, C. 5) Add a not null constraint for the column distance in train_route.
4	Use SQL PLUS functions to. 1. Find the passengers whose date of journey is one month from today. 2. Print the train names in upper case. 3. Print the passenger names with left padding character. 4. Print the station codes replacing K with M. 5. Translate all the LC in class column (Train_fare) to POT and display. 6. Display the fare details of all trains, if any value is ZERO, print as NULL value. 7. Display the pnrno and transaction id, if transaction id is null, print 'not generated'. 8. Print the date_of_jounrney in the format '27th November 2010'. 9. Find the maximum fare (total fare). 10. Find the average age of passengers in one ticket. 11. Find the maximum length of station name available in the database. 12. Print the fare amount of the passengers as rounded value. 13. Add the column halt time to train route. 14. Update values to it from arrival time and depart time. High Level: 15. Update values to arrival time and depart time using conversion functions. 16. Display the arrival time, depart time in the format HH:MI (24 hours and minutes).
5	Write Queries to. Use SET Operators 1. Find the train integers for which reservation have not yet been made. 2. Find the train names that do not have a first AC class coach. 3. Print all the PNR nos available in the database. 4. Find passenger names who have booked to 'Pune'. Use Nested Query(in Operators) 1. Find the train names that stop in 'Warangal'. 2. Find the train names that are superfast and the service tax is zero. 3. Find the Passenger name who have booked for the train that starts from 'Secunderabad'. 4. Find the trains names that have all the AC coaches and the base fare is less than 3000 for each case.
6	Use Join Query 1. Find the train names that stop in 'Warangal'. 2. Find the train names that are superfast and the service tax is zero. 3. Find the Passenger name (and train name) who have booked for the train that starts from 'Secunderabad'.

	4. Display the trains names, each type of class and the total fare for each type of class. 5. Display all the train details and the ticket details (if booked any). 6. Create a sequence to provide values for the PNR no. 7. Write a query for full outer join using any of the tables above. 1. Write Queries to. Use Coorelated (and nested) Query 1. Find the train names for which ten tickets have been reserved. 2. Find the trains that have more than ten substations. 3. Find the passengers who do not pass through 'Kachiguda'. 4. Find passengers who have booked for super fast trains.
7	Complex queries (use group by / group by having/join/nested) 1. Take the start station code and end station code and display the train details. 2. List the train names and the integer of sub stations it has. 3. List the stations where all types of trains stop. 4. List the trains names that has at least four bookings. 5. Create a table cancellation history (Insert values from ticket and passenger table). 6. Create a table for all the train integers and class available in train_ticket_fare with total seats. 1. Find the station name that has highest integer of trains stopping at.
8	Write a simple PL/SQL block to. 1. Print the fibonacci series. 2. Print the factorial of a given integer. 3. Print 'NOT confirmed' based on the reservation status, of a particular passenger. 4. Print the total seats available for a particular train and for a particular class.
9	Write a cursor for the following. 1. Retrieve the passenger details for —X train integer and given journey date. 2. Display the train name (once) and the substation names. 3. Display the fare details of a particular train(use basic exceptions) 4. Write a cursor to update the reservation status of the passengers (generate seat integer, if seats have reached maximum, put waiting list integer (30% of total seats), if waiting list integer reaches maximum, put PQWL (10%of total seats), RAC-20%)
10	1. Write a PL/SQL procedure to. a. List the details of passengers who has reserved next to —Mr. X. b. PNR No. of a passengers for a given source and a destination. 2. Write a PL/SQL function to. a. Get the PNRNo and return the total ticket fare. b. Get the Passenger name, train no and return the total journey time in hour's and minutes.
11	Write a Trigger for the following: 1) When a passenger cancels a ticket, do the necessary process and update the cancellation history table. 2) When train integer is changed, update it in referencing tables. 3) When a passenger record is inserted reservation status should be automatically updated.
12	1) Use TCL commands for your transactions. (commit, rollback, savepoint) 2) Create a role named 'clerk', and give permission for him to select only the trains starting from 'Warangal' along with fare details.

	3) Create a nested table containing trainno, name, source, destination and passengers who have booked for it (PNR no, sno, name, age). Find the passengers whose name start with 'S' and train starts from 'Warangal'

Textbooks:

1. Database Management Systems, Raghurama Krishnan, Johannes Gehrke, Tata Mc Graw Hill 3rd Edition
2. Database System Concepts, Silberschatz, Korth, Mc Graw hill, V edition.

References:

1. Database Systems design, Implementation, and Management, Peter Rob & Carlos Coronel, 7th Edition.
2. SQL The Complete Reference, James R. Groff, Paul N. Weinberg, 3rd Edition,
3. Oracle for Professionals, The X Team, S.Shah and V. Shah, SPD.
4. Database Systems Using Oracle: A Simplified guide to SQL and PL/SQL, Shah, PHI.

Outcomes:

1. Design database schema for a given application and apply normalization
2. Acquire skills in using SQL commands for data definition and data manipulation.
3. Develop solutions for database applications using procedures, cursors and triggers.

CO- PO, PSO Mapping
(3/2/1 indicates strength of correlation) 3-Strong, 2-Medium, 1-Weak

COs	Programme Outcomes (POs)												PSOs		
	PO1	PO2	PO3	PO4	PO5	PO6	PO7	PO8	PO9	PO10	PO11	PO12	PSO1	PSO2	PSO3
CO1	2	-	-												
CO2	-	2	2										1	2	
CO3	2	2	2	3										2	2

Week-1
MySQL
(Structured Query Language)

SQL is a standard language for storing, manipulating and retrieving data in databases.
SQL in: MySQL, SQL Server, MS Access, Oracle, Sybase, Informix, Postgres, and other database systems.

Introduction to SQL
SQL is a standard language for accessing and manipulating databases.

What is SQL?
- SQL stands for Structured Query Language
- SQL lets you access and manipulate databases
- SQL became a standard of the American National Standards Institute (ANSI) in 1986, and of the International Organization for Standardization (ISO) in 1987

What Can SQL do?
- SQL can execute queries against a database
- SQL can retrieve data from a database
- SQL can insert records in a database
- SQL can update records in a database
- SQL can delete records from a database
- SQL can create new databases
- SQL can create new tables in a database
- SQL can create stored procedures in a database
- SQL can create views in a database
- SQL can set permissions on tables, procedures, and views

SQL is a Standard - BUT....
Although SQL is an ANSI/ISO standard, there are different versions of the SQL language.
However, to be compliant with the ANSI standard, they all support at least the major commands (such as SELECT, UPDATE, DELETE, INSERT, WHERE) in a similar manner.
Note: Most of the SQL database programs also have their own proprietary extensions in addition to the SQL standard!
Using SQL in Your Web Site
To build a web site that shows data from a database, you will need:
- An RDBMS database program (i.e. MS Access, SQL Server, MySQL)
- To use a server-side scripting language, like PHP or ASP
- To use SQL to get the data you want
- To use HTML / CSS to style the page

RDBMS
RDBMS stands for Relational Database Management System.
RDBMS is the basis for SQL, and for all modern database systems such as MS SQL Server, IBM DB2, Oracle, MySQL, and Microsoft Access.
The data in RDBMS is stored in database objects called tables. A table is a collection of related data entries and it consists of columns and rows.
Look at the "Customers" table:
Example
```
SELECT * FROM Customers;
```
Try it Yourself »

Every table is broken up into smaller entities called fields. The fields in the Customers table consist of CustomerID, CustomerName, ContactName, Address, City, PostalCode and Country. A field is a column in a table that is designed to maintain specific information about every record in the table.
A record, also called a row, is each individual entry that exists in a table. For example, there are 91 records in the above Customers table. A record is a horizontal entity in a table.
A column is a vertical entity in a table that contains all information associated with a specific field in a table.

SQL Syntax

Database Tables

A database most often contains one or more tables. Each table is identified by a name (e.g. "Customers" or "Orders"). Tables contain records (rows) with data.
In this tutorial we will use the well-known Northwind sample database (included in MS Access and MS SQL Server).
Below is a selection from the "Customers" table:

CustomerID	CustomerName	ContactName	Address	City	Postal Code	Country
1	Alfreds Futterkiste	Maria Anders	Obere Str. 57	Berlin	12209	Germany
2	Ana Trujillo Emparedados y helados	Ana Trujillo	Avda. de la Constitución 2222	México D.F.	05021	Mexico
3	Antonio Moreno Taquería	Antonio Moreno	Mataderos 2312	México D.F.	05023	Mexico
4	Around the Horn	Thomas Hardy	120 Hanover Sq.	London	WA1 1DP	UK
5	Berglunds snabbköp	Christina Berglund	Berguvsvägen 8	Luleå	S-958 22	Sweden

The table above contains five records (one for each customer) and seven columns (CustomerID, CustomerName, ContactName, Address, City, PostalCode, and Country).

SQL Statements
Most of the actions you need to perform on a database are done with SQL statements.

The following SQL statement selects all the records in the "Customers" table:

Example

```
SELECT * FROM Customers;
```

Try it Yourself »

In this tutorial we will teach you all about the different SQL statements.

Keep in Mind That...

- SQL keywords are NOT case sensitive: select is the same as SELECT

In this tutorial we will write all SQL keywords in upper-case.

Semicolon after SQL Statements?

Some database systems require a semicolon at the end of each SQL statement.
Semicolon is the standard way to separate each SQL statement in database systems that allow more than one SQL statement to be executed in the same call to the server.
In this tutorial, we will use semicolon at the end of each SQL statement.

Some of The Most Important SQL Commands

1. DDL: Data Definition Language (DDL) statements are used to define the database structure or schema.

- **CREATE DATABASE** - creates a new database
- **ALTER DATABASE** - modifies a database
- **CREATE TABLE** - creates a new table
- **ALTER TABLE** - modifies a table
- **DROP TABLE** - deletes a table
- **CREATE INDEX** - creates an index (search key)
- **DROP INDEX** - deletes an index
- **RENAME -** rename an object

2. DML: Data Manipulation Language (DML) statements are used for managing data within schema objects and to manipulate data of a database objects.

- **SELECT** - extracts data from a database
- **UPDATE** - updates data in a database
- **DELETE** - deletes data from a database
- **INSERT INTO** - inserts new data into a database
- **TRUNCATE** - remove all records from a table, including all spaces allocated for the records are removed

3. DCL: Data Control Language (DCL) statements are used to create roles, permissions, and referential integrity as well it is used to control access to database by securing it. To control the data of a database.

- DCL Commands: Grant, Revoke
- GRANT - gives user's access privileges to database
- REVOKE -withdraw access privileges given with the GRANT command

4. TCL: Transaction Control (TCL) statements are used to manage the changes made by DML statements. It allows statements to be grouped together into logical transactions.

- TCL Commands: Commit, Rollback, Save point
- COMMIT - save work done
- SAVEPOINT - identify a point in a transaction to which you can later roll back
- ROLLBACK - restore database to original since the last COMMIT

MySQL –u Username –p Password;
mysql>QUIT [OR] mysql>\q

mysql>select version();
mysql>select now();
mysql>select current_date;
mysql>select current_time;
mysql>show databases;
mysql>show tables;

CREATE DATABASE Reservations;
mysql>create database reservations;

use Reservations;

SHOW TABLES;

Syntax with examples
1. DDL (Data Definition Language) Commands: CREATE, ALTER and DROP.

CREATE: This command useful for creating creating table.

Syntax:
create table [table name] (column1 datatype[size], column 2 datatype[size],… column n datatype[size]);
Ex:

```
Mysql >CREATE TABLE president (
        last_name varchar(15) not null,
        first_name varchar(15) not null,
        state varchar(2) not null,
        city varchar(20) not null,
        birth date not null default '0000-00-00',
        death date null
        );

mysql> SHOW tables;
mysql> DESCRIBE president;

INSERT INTO president values ('Washington', 'George', 'VA', 'Westmoreland County', '17320212',
'17991214');

SELECT * FROM president;
SELECT * FROM president WHERE state="VA";
SELECT state, first_name, last_name FROM president;

DELETE FROM president WHERE first_name="George";
INSERT INTO president values ('Washington', 'George', 'VA', 'Westmoreland County', '17320212',
'17991214');

UPDATE president set city= “hyd”  WHERE first_name="George";
```

Ex:
SQL >create table student (s_rollno integer(10) primary key, s_name varchar(10), gender varchar(5), dob date, addr1 varchar(10), addr2 varchar(10), city varchar(10), percentage integer(3));

MYSQL> DESC STUDENT;

Name	Type	Null	Key	Default	Extra
s_rollno	integer(10)	NO	PRI	NULL	
s_name	varchar(10)	YES		NULL	
gender	varchar(5)	YES		NULL	
dob	date	YES		NULL	
addr1	varchar(10)	YES		NULL	
addr2	varchar(10)	YES		NULL	
city	varchar(10)	YES		NULL	
percentage	integer (3)	YES		NULL	

MYSQL > select s_rollno, s_name from student;
Empty set (0.02 sec)

Week-2

Create table by using Constraints: Constraints are two types:
1. Table Level Constraints.
2. Column Level Constraints.

1. NOT NULL: **a)** ***Not null constraint at column level.***
Syntax:
<col> <datatype> (size) not null
MYSQL > create table emp(e_id varchar(5) NOT NULL, e_name varchar(10), e_design varchar(10), dept varchar(10), mgr varchar(10), salary integer(10));

2. UNIQUE:

Unique constraint at column level.

Syntax: <col> <datatype> (size) unique
Ex:-

MYSQL > create table depositor(customer_name varchar(10), acc_no integer(15) UNIQUE, brach_name varchar(10));
Query OK, 0 rowa affected (0.19 sec)

Unique constraint at table level: **Syntax:**

Create table tablename(col=format, col=format, unique(<col1>,<col2>));
Ex:-

MYSQL > create table depositor1(customer_name varchar(10), acc_no integer(15), brach_name varchar(10),UNIQUE(acc_no));
Query OK, 0 rowa affected (0.19 sec)

3. PRIMARY KEY:

Primary key constraint at column level

Syntax:
<col><datatype>(size)primary key;
Ex:-

MYSQL> create table customer(customer_id integer(5) PRIMARY KEY, customer_name varchar(10), customer_street varchar(10), brach_name varchar(10));

Primary key constraint at table level.

Syntax:
Create table tablename(col=format, col=format primary key(col1>, <col2>);
Ex:-

SQL > create table customer1(customer_id integer(5), customer_name varchar(10), customer_street varchar(10), brach_name varchar(10), PRIMARY KEY(customer_id));

4. CHECK:

Check constraint at column level.

Syntax: <col><datatype>(size) check(<logical expression>)
Ex:-create table loan(loan_no varchar(10), customer_name varchar(10), balance integer (10) CHECK(balance>1000));

Check constraint constraint at table level.

Syntax: check(<logical expression>)
Ex:- create table loan1(loan_no varchar(10), customer_name varchar(10), balance integer(10), CHECK(balance>1000));

5. FOREIGN KEY:

Foreign key constraint at column level.
Syntax: Column_name Datatype(size) REFERENCES parent_table_name (parent_column_name)

Ex:- CREATE TABLE books (book_id INTEGER(3), book_title VARCHAR(30), book_price INTEGER(3), book_author_id INTEGER(3) REFERENCES author(author_id));

Foreign key constraint at table level

Syntax: CONSTRAINT constraint_name FOREIGN KEY(child_table_column) REFERENCES Parent_table_name(parent_table_column)

Ex:-CREATE TABLE books1 (
book_id INTEGER,
book_title VARCHAR(30),
book_price INTEGER(3),
book_author_id INTEGER(3),
constraint book_bi_pk PRIMARY KEY (book_id),
CONSTRAINT bok_ai_fk FOREIGN KEY (book_author_id) REFERENCES authors(author_id));

Create table authors (
author_id integer,
author_name varchar(30),
constraint author_id primary key (author_id)
);

```
CREATE TABLE FLTAVAIL
        (
        FLIGHT_ID CHAR(6) NOT NULL,
        SEGMENT_NUMBER INT NOT NULL,
        FLIGHT_DATE DATE NOT NULL,
        ECONOMY_SEATS_TAKEN INT,
        BUSINESS_SEATS_TAKEN INT,
        FIRSTCLASS_SEATS_TAKEN INT,
        CONSTRAINT FLTAVAIL_PK PRIMARY KEY (FLIGHT_ID, SEGMENT_NUMBER)
        );
```

WEEK-3

CREATION OF TABLES

1) Create a table called Employee with the following structure.

Name	Type
Empno	Integer
Ename	Varchar(10)
Job	Varchar(10)
Mgr	Integer
Sal	Integer

a. Add a column commission with domain to the Employee table.
b. Insert any five records into the table.
c. Update the column details of job
d. Rename the column of Employ table using alter command.
e. Delete the employee whose Empno is 105.

SOLUTION:

MYSQL> create table employee(empno integer, ename varchar(10), job varchar(10),mgr integer, sal integer);

Table created.

MYSQL> desc employee;

Name	Type	Null?	Key	Default	Extra
EMPNO	INTEGER	YES		NULL	
ENAME	VARCHAR(10)	YES		NULL	
JOB	VARCHAR(10)	YES		NULL	
MGR	INTEGER	YES		NULL	
SAL	INTEGER	YES		NULL	

a. Add a column commission with domain to the Employee table.

MYSQL> alter table employee add(commission integer);
Table altered.

MYSQL> desc employee;

Name	Type	Null?	Key	Default	Extra
EMPNO	INTEGER	YES		NULL	
ENAME	VARCHAR(10)	YES		NULL	
JOB	VARCHAR(10)	YES		NULL	
MGR	INTEGER	YES		NULL	
SAL	INTEGER	YES		NULL	
COMMISSION	INTEGER	YES		NULL	

b. Insert any five records into the table.

SQL> insert into employee values(101,' abhi ',' manager', 1234,10000,70);
Query OK, 1 row affected(0.02 sec).

SQL> select * from employee;

EMPNO	ENAME	JOB	MGR	SAL	COMMISSION
101	abhi	manager	1234	10000	70
102	rohith	analyst	2345	9000	65
103	david	analyst	3456	9000	65
104	rahul	clerk	4567	7000	55
105	pramod	salesman	5678	5000	50

d. Rename the column of Employ table using alter command.

MYSQL> alter table employee change column mgr manager_no int;
Query OK, 0 rows affected (0.33 sec)
Records: 0 Duplicates: 0 Warnings: 0

MYSQL> desc employee;

Name	Type	Null?	Key	Default	Extra
EMPNO	INTEGER	YES		NULL	
ENAME	VARCHAR(10)	YES		NULL	
JOB	VARCHAR(10)	YES		NULL	
MANAGER_NO	INTEGER	YES		NULL	
SAL	INTEGER	YES		NULL	
COMMISSION	INTEGER	YES		NULL	

e. Delete the employee whose Empno is 105.

MYSQL> delete from employee where empno=105;
Query OK, 2 rows affected (0.15 sec)

MYSQL> select * from employee;

EMPNO	ENAME	JOB	MANAGER_NO	SAL	COMMISSION
101	abhi	manager	1234	10000	70
102	rohith	analyst	2345	9000	65
103	david	trainee	3456	9000	65
104	rahul	clerk	4567	7000	55

2) Create department table with the following structure.

Name	**Type**
Deptno	Integer
Deptname	Varchar(10)
location	Varchar(10)

a. Add column designation to the department table.
b. Insert values into the table.
c. List the records of dept table grouped by deptno.
d. Update the record where deptno is 9.
e. Delete any column data from the table.

SOLUTION:

MYSQL> create table department(deptno integer,deptname varchar(10),location varchar(10));
Query OK, 0 rows affected (0.08 sec)

MYSQL> desc department;
desc department;

Field	Type	Null	Key	Default	Extra
deptno	int(11)	YES		NULL	
deptname	varchar(10)	YES		NULL	
location	varchar(10)	YES		NULL	

3 rows in set (0.00 sec)

a. Add column designation to the department table.

MYSQL> alter table department add(designation varchar(10));
Query OK, 0 rows affected (0.08 sec)
Records: 0 Duplicates: 0 Warnings: 0

MYSQL> desc department;

Field	Type	Null	Key	Default	Extra
deptno	int(11)	YES		NULL	
deptname	varchar(10)	YES		NULL	
location	varchar(10)	YES		NULL	
designation	varchar(10)	YES		NULL	

4 rows in set (0.02 sec)

b. Insert values into the table.

MYSQL> insert into department values(9,'accounting','hyderabad','manager');
Query OK, 1 row affected (0.05 sec)

Insert 4 more rows using data in below table.

insert into department values(9,' research','hyderabad','professor');
insert into department values(9,' sales',' chennai','salesman');
insert into department values(9,' operation',' banglore','operator');
insert into department values(9,' operation',' mumbai','manager');

MYSQL> select * from department ;

deptno	deptname	location	designation
9	accounting	Hyderabad	accountant
9	research	hyderabad	professor
9	sales	chennai	salesman
9	operation	banglore	operator
9	operation	mumbai	manager

5 rows in set (0.00 sec)

c. List the records of dept table grouped by deptno.

SQL> select deptno,deptname from department group by deptno, deptname;

```
+--------        +------------    +
| deptno         | deptname       |
+--------        +------------    +
|    9           |  operation     |
|    9           |  research      |
|    9           |  sales         |
|    9           | accounting     |
+--------        +------------    +
```

4 rows in set (0.00 sec)

d. Update the record where deptno is 9.

SQL> update department set designation='accountant' where deptno=9;
Query OK, 4 rows affected (0.02 sec)
Rows matched: 5 Changed: 4 Warnings: 0

MYSQL> select * from department;

```
+--------        +------------    +-----------     +-------------   +
| deptno         | deptname       | location        | designation   |
+--------        +------------    +-----------     +-------------   +
|    9           | accounting     | hyderabad       | accountant    |
|    9           |  research      | hyderabad       | accountant    |
|    9           |  sales         |  chennai        | accountant    |
|    9           |  operation     |  banglore       | accountant    |
|    9           |  operation     |  mumbai         | accountant    |
+--------        +------------    +-----------     +-------------   +
```

5 rows in set (0.00 sec)

e. Delete any column data from the table.

MYSQL> alter table department drop column designation;
Query OK, 5 rows affected (0.14 sec)
Records: 5 Duplicates: 0 Warnings: 0

MYSQL> select * from department;

```
+--------        +------------    +-----------     +
| deptno         | deptname       | location        |
+--------        +------------    +-----------     +
|    9           | accounting     | hyderabad       |
|    9           |  research      | hyderabad       |
|    9           |  sales         |  chennai        |
|    9           |  operation     |  banglore       |
|    9           |  operation     |  mumbai         |
+--------        +------------    +-----------     +
```

Creating Sailors Table:

The table has following relation schema:
Sailors(sid : integer, sname : string, rating : integer, age : real)

```
mysql > CREATE TABLE Sailors(
        sid INT,
        sname VARCHAR(20),
        rating INT,
        age FLOAT,
        PRIMARY KEY(sid) );

        Query OK, 0 rows affected (0.03 sec)

mysql> INSERT INTO Sailors VALUES(22,'Dustin',7,45);
mysql> INSERT INTO Sailors VALUES(29,'Brutus',1,33);
mysql> INSERT INTO Sailors VALUES(31,'Lubber',8,56);
mysql> INSERT INTO Sailors VALUES(32,'Andy',8,26);
mysql> INSERT INTO Sailors VALUES(58,'Rusty',10,35);
mysql> INSERT INTO Sailors VALUES(74,'Horatio',9,35);
mysql> INSERT INTO Sailors VALUES(64,'Horatio',7,35);
mysql> INSERT INTO Sailors VALUES(95,'Bob',3,64);
mysql> INSERT INTO Sailors VALUES(85,'Art',3,26);
mysql> INSERT INTO Sailors VALUES(71,'Zorba',10,16);

SELECT * FROM Sailors;
```

Creating Boats Table:
The table has following relation schema: Boats(bid : integer, bname : string, color : string)

```
mysql > CREATE TABLE Boats(
        bid INT,
        bname VARCHAR(10),
        color VARCHAR(10),
        PRIMARY KEY(bid));
        Query OK, 0 rows affected (0.05 sec)

mysql> DESC Boats;

mysql> INSERT INTO Boats VALUES(101,'Interlake','blue');
mysql> INSERT INTO Boats VALUES(102,'Interlake','red');
mysql> INSERT INTO Boats VALUES(103,'Clipper','green');
mysql> INSERT INTO Boats VALUES(104,'Marine','red');
SELECT * FROM Boats;
```

Creating Reserves Table:
The table has following relation schema: Reserve(sid : integer, bid : integer, day : date)

```
mysql > CREATE TABLE Reserves(
        sid INT,
        bid INT,
        day DATE NOT NULL,
        PRIMARY KEY(sid,bid),
        FOREIGN KEY(sid) REFERENCES Sailors(sid) ON DELETE CASCADE,
        FOREIGN KEY(bid) REFERENCES Boats(bid) ON DELETE CASCADE);

        Query OK, 0 rows affected (0.05 sec)
mysql> DESC Reserves;

mysql> INSERT INTO Reserves VALUES(22,101,'2012/10/10');
mysql> INSERT INTO Reserves VALUES(22,102,'2012/10/9');
mysql> INSERT INTO Reserves VALUES(22,103,'2012/08/10');
mysql> INSERT INTO Reserves VALUES(22,104,'2012/07/10');
mysql> INSERT INTO Reserves VALUES(31,102,'2012/11/10');
mysql> INSERT INTO Reserves VALUES(31,103,'2012/06/11');
mysql> INSERT INTO Reserves VALUES(31,104,'2012/12/11');
mysql> INSERT INTO Reserves VALUES(64,101,'2012/05/09');
mysql> INSERT INTO Reserves VALUES(64,102,'2012/08/09');
mysql> INSERT INTO Reserves VALUES(74,103,'2012/08/09');
```

LAB ASSIGNMENT:

1. Create a table called Customer table

Name	Type
Cust name	Varchar(20)
Cust street	Varchar(20)
Cust city	Varchar(20)

   a. Insert records into the table.
   b. Add salary column to the table.
   c. Alter the table column domain.
   d. Drop salary column of the customer table.
   e. Delete the rows of customer table whose cust_city is „hyd".

2. Create a table called branch table.

Name	Type
Branch name	Varchar(20)
Branch city	Varchar(20)
asserts	Integer

   a. Increase the size of data type for asserts to the branch.
   b. Add and drop a column to the branch table.
   c. Insert values to the table.
   d. Update the branch name column
   e. Delete any two columns from the table

3. Create a table called sailor table

Name	Type
Sid	Integer
Sname	Varchar(20)
rating	Varchar(20)

a. Add column age to the sailor table.
b. Insert values into the sailor table.
c. Delete the row with rating >8.
d. Update the column details of sailor.
e. Insert null values into the table.

4. Create a table called reserves table

Name	**Type**
Boat id	Integer
sid	Integer
day	Integer

a. Insert values into the reserves table.
b. Add column time to the reserves table.
c. Alter the column day data type to date.
d. Drop the column time in the table.
e. Delete the row of the table with some condition.

WEEK -4
QUERIES USING DDL AND DML

1.

a. Create a user and grant all permissions to the user.
b. Insert the any three records in the employee table and use rollback. Check the result.
c. Add primary key constraint and not null constraint to the employee table.
d. Insert null values to the employee table and verify the result.

SOLUTION:

a) create a user and grant all permissions to the user.

CONNECT <USER-NAME>/<PASSWORD>@<DATABASE NAME>;

--Create user query
CREATE USER <USER NAME> IDENTIFIED BY <PASSWORD>;

--Provide roles
GRANT CONNECT,RESOURCE,DBA TO <USER NAME>; -

-Assigning privileges
GRANT CREATE SESSION GRANT ANY PRIVILEGE TO <USER NAME>; GRANT UNLIMITED TABLESPACE TO <USER NAME>;

--Provide access to tables.
GRANT SELECT, UPDATE, INSERT, DELETE ON <TABLE NAME> TO <USER NAME>;

b) Insert the any three records in the employee table and use rollback. Check the result.

In MySQL, all user activity occurs inside a transaction. If autocommit mode is enabled, each SQL statement forms a single transaction on its own. By default, MySQL starts the session for each new connection with autocommit enabled, so MySQL does a commit after each SQL statement if that statement did not return an error. If a statement returns an error, the commit or rollback behavior depends on the error.

A session that has autocommit enabled can perform a multiple-statement transaction by starting it with an explicit START TRANSACTION or BEGIN statement and ending it with a COMMIT or ROLLBACK statement.

If autocommit mode is disabled within a session with SET autocommit = 0, the session always has a transaction open. A COMMIT or ROLLBACK statement ends the current transaction and a new one starts.

If a session that has autocommit disabled ends without explicitly committing the final transaction, MySQL rolls back that transaction.

Some statements implicitly end a transaction, as if you had done a COMMIT before executing the statement.

A COMMIT means that the changes made in the current transaction are made permanent and become visible to other sessions. A ROLLBACK statement, on the other hand, cancels all modifications made by the current transaction. Both COMMIT and ROLLBACK release all MySQL locks that were set during the current transaction.

MYSQL> SELECT * FROM EMPLOYEE;

EMPNO	ENAME	JOB	MANAGER_NO	SAL	COMMISSION
101	abhi	manager	1234	1100	70
102	rohith	analyst	2345	9000	65
103	david	trainee	3456	9000	65
104	rahul	clerk	4567	7000	55

```
MYSQL> insert into employee values(101,'abhi',' manager',1234,1100,70);
MYSQL> insert into employee values(102,'rohith','analyst',2345,9000,65);
MYSQL> insert into employee values(103,' david ','trainee',3456,9000,65);

MYSQL> SET autocommit = 0;
MYSQL> insert into employee values(104,' rahul','clerk',4567,7000,55);

mysql> select * from employee;
+-------+---------+----------+---------+------+------------+
| empno | ename   | job      | manager | sal  | commission |
+-------+---------+----------+---------+------+------------+
|   101 | abhi    |  manager |    1234 | 1100 |         70 |
|   102 | rohith  | analyst  |    2345 | 9000 |         65 |
|   103 |  david  | trainee  |    3456 | 9000 |         65 |
|   104 |  rahul  | clerk    |    4567 | 7000 |         55 |
+-------+---------+----------+---------+------+------------+
4 rows in set (0.00 sec)

MYSQL> rollback;
Query OK, 0 rows affected (0.00 sec)

MYSQL> SELECT * FROM EMPLOYEE;
+-------+---------+----------+---------+------+------------+
| empno | ename   | job      | manager | sal  | commission |
+-------+---------+----------+---------+------+------------+
|   101 | abhi    |  manager |    1234 | 1100 |         70 |
|   102 | rohith  | analyst  |    2345 | 9000 |         65 |
|   103 |  david  | trainee  |    3456 | 9000 |         65 |
+-------+---------+----------+---------+------+------------+

MYSQL> SET autocommit = 1;
```

c) Add primary key constraint and not null constraint to the employee table.

```
MYSQL> alter table employee modify empno integer primary key;
Query OK, 1 row affected (0.17 sec)
```

```
Records: 1  Duplicates: 0  Warnings: 0

MYSQL> desc employee;
+------------  +-------------  +------+-----  +---------+-------  +
| Field        | Type         | Null | Key   | Default | Extra |
+------------  +-------------  +------+-----  +---------+-------+
| empno        | int(11)      | NO   | PRI   | NULL    |       |
| ename        | varchar(10)  | YES  |       | NULL    |       |
| job          | varchar(10)  | YES  |       | NULL    |       |
| mgr          | int(11)      | YES  |       | NULL    |       |
| sal          | int(11)      | YES  |       | NULL    |       |
| commission   | int(11)      | YES  |       | NULL    |       |
+------------  +-------------  +------+-----  +---------  +-------  +
6 rows in set (0.00 sec)
```

d) Insert null values to the employee table and verify the result.

```
MYSQL> insert into employee  values(105,'mohith','salesman',5678,null,50);
Query OK, 1 row affected (0.00 sec)
```

2.
   a. Insert values in the department table and use commit.
   b. Add constraints like unique and not null to the department table.
   c. Insert repeated values and null values into the table.

SOLUTION:

b) Insert values in the department table and use commit.

```
MYSQL> insert into department values(13,'CSE','HYD','Professor');
Query OK, 1 row affected (0.00 sec)

MYSQL> commit;
Query OK, 0 rows affected (0.02 sec)

MYSQL> select * from department;
+--------+------------  +-----------  +-------------  +
| deptno | deptname     | location     | designation   |
+--------+------------  +-----------  +-------------  +
|    9   |  research    | hyderabad    | professor     |
|    9   |  sales       |  chennai     | salesman      |
|    9   |  operation   |  banglore    | operator      |
|    9   |  operation   |  mumbai      | manager       |
|   13   | CSE          | HYD          | Professor     |
+--------+------------  +-----------  +-------------  +
5 rows in set (0.00 sec)
```

c) Add constraints like unique and not null to the department table.

```
MYSQL> alter table department  modify  deptno integer unique;
Table altered.
MYSQL> alter table department  modify  location varchar(10) not null;
```

Query OK, 5 rows affected (0.08 sec)
Records: 5 Duplicates: 0 Warnings: 0

SQL> DESC DEPARTMENT;

Field	Type	Null	Key	Default	Extra
deptno	int(11)	YES		NULL	
deptname	varchar(10)	YES		NULL	
location	varchar(10)	NO		NULL	
designation	varchar(10)	YES		NULL	

4 rows in set (0.04 sec)

d) Insert repeated values and null values into the table.

MYSQL> insert into department values(10,'research','', 'Sales');
Query OK, 1 row affected, 1 warning (0.02 sec)

LAB ASSIGNMENT:

1 a. create a user and grant all permissions to the user.
b. Insert values into the table and use commit.
c. Delete any three records in the department table and use rollback.
d. Add constraint primary key and foreign key to the table.

2 a. create a user and grant all permissions to the user.
b. Insert records in the sailor table and use commit.
c. Add save point after insertion of records and verify save point.
d. Add constraints not null and primary key to the sailor table.

3 a. create a user and grant all permissions to the user.
b. Use revoke command to remove user permissions.
c. Change password of the user created.
d. Add constraint foreign key and not null.

4 a. create a user and grant all permissions to the user.
b. Update the table reserves and use savepoint and rollback.
c. Add constraint primary key, foreign key and not null to the reserves table
d. Delete constraint not null to the table column.

WEEK – 5

SQL Queries

SELECT * FROM Reserves;

Find the names and ages of all sailors.

mysql > SELECT sname, age FROM Sailors;

[OR]

mysql > SELECT S.sname, S.age FROM Sailors AS S; [where S is a rename for Sailors]

[OR]

mysql > SELECT S.sname, S.age FROM Sailors S; [where S is a rename for Sailors]

Find all sailors with a rating above 7.

mysql > SELECT * FROM Sailors WHERE rating>7;

[OR]

mysql > SELECT S.sid, S.sname, S.rating, S.age FROM Sailors AS S WHERE S.rating > 7;

[OR]

mysql > SELECT S.sid AS SailorId, S.sname, S.rating, S.age FROM Sailors S WHERE S.rating > 7;

Find the names of sailors who have reserved boat 103.

mysql > SELECT sname FROM Sailors, Reserves WHERE Sailors.sid=Reserves.sid AND bid=103;

[OR]

mysql > SELECT S.sname FROM Sailors AS S, Reserves AS R WHERE S.sid=R.sid AND R.bid=103;

[OR]

mysql > SELECT S.sname FROM Sailors S WHERE S.sid IN (SELECT R.sid FROM Reserves R WHERE R.bid=103);

Mysql > SELECT DISTINCT R.sid FROM Reserves R, Boats B WHERE R.bid=B.bid AND B.color='red';

Find the names of sailors who have reserved a red boat.

mysql > SELECT DISTINCT S.sname FROM Sailors S, Reserves R, Boats B WHERE S.sid=R.sid AND R.bid=B.bid AND B.color='red';

[OR]

mysql > SELECT DISTINCT S.sname FROM Sailors S WHERE S.sid IN (SELECT R.sid FROM Reserves R, Boats B WHERE R.bid=B.bid AND B.color='red');

[OR]

mysql > SELECT DISTINCT S.sname FROM Sailors S WHERE S.sid IN (SELECT R.sid FROM Reserves R WHERE R.bid IN (SELECT B.bid FROM Boats B WHERE B.color='red'));

Find the names of sailors who have reserved a red boat.

```
mysql > SELECT DISTINCT S.sname
    -> FROM Sailors S, Reserves R, Boats B
    -> WHERE S.sid=R.sid AND R.bid=B.bid AND B.color='red';
```

[OR]

```
mysql > SELECT DISTINCT S.sname
    -> FROM Sailors S
    -> WHERE S.sid IN (SELECT R.sid
    ->FROM Reserves R, Boats B
    ->WHERE R.bid=B.bid AND B.color='red');
```

[OR]

mysql > SELECT DISTINCT S.sname

```
-> FROM Sailors S
-> WHERE S.sid IN (SELECT R.sid
                    ->FROM Reserves R
                    ->WHERE R.bid IN (SELECT B.bid
                                        ->FROM Boats B
                                        ->WHERE B.color='red'));
```

Find the colors of boats reserved by Lubber.

```
mysql > SELECT DISTINCT B.color FROM Sailors S, Reserves R, Boats B WHERE S.sid=R.sid AND
R.bid=B.bid AND S.sname='Lubber';
                                        [OR]
mysql > SELECT DISTINCT B.color FROM Boats B WHERE B.bid IN (SELECT R.bid FROM Reserves
R WHERE R.sid IN (SELECT S.sid FROM Sailors S WHERE S.sname='Lubber'));
```

Find the names of sailors who have reserved at least one boat.

```
Mysql > SELECT DISTINCT S.sname FROM Sailors S, Reserves R WHERE S.sid=R.sid;
```

Find the names of sailors who have reserved at least two boats.

```
Mysql > SELECT DISTINCT S.sname FROM Sailors S, Reserves R1, Reserves R2 WHERE S.sid=R1.sid
AND R1.sid=R2.sid AND R1.bid<>R2.bid;
```

Find the names of sailors who have reserved at least three boats.

```
Mysql > SELECT DISTINCT S.sname FROM Sailors S WHERE S.sid IN (SELECT R1.sid FROM
Reserves R1, Reserves R2, Reserves R3 WHERE R1.sid=R2.sid AND R2.sid=R3.sid AND
R1.bid<>R2.bid AND R2.bid<>R3.bid AND R3.bid<>R1.bid);
```

Compute increments for the ratings of persons who have sailed two different boats on the same day.

```
Mysql > SELECT S.sname,S.rating+1 AS rating FROM Sailors S, Reserves R1, Reserves R2 WHERE
S.sid=R1.sid AND S.sid=R2.sid AND R1.day=R2.day AND R1.bid<>R2.bid;
```

Find the ages of sailors whose name begins and ends with B and has at least three characters.

```
Mysql > SELECT S.age FROM Sailors S WHERE S.sname LIKE 'B%_B'; [OR] LIKE 'B_%B';
```

Find the names of sailors who have reserved a red or a green boat.

```
Mysql > SELECT DISTINCT S.sname
        -> FROM Sailors S, Reserves R, Boats B
        -> WHERE S.sid=R.sid AND R.bid=B.bid AND
        ->(B.color='red' OR B.color='green');
                                        [OR]
mysql > SELECT S.sname
        -> FROM Sailors S, Reserves R, Boats B
        -> WHERE S.sid=R.sid AND R.bid=B.bid AND B.color='red'
        ->UNION
        -> SELECT S2.sname
        -> FROM Sailors S2, Reserves R2, Boats B2
        -> WHERE S2.sid=R2.sid AND R2.bid=B2.bid AND B2.color='green';
```

Find the names of sailors who have reserved a red and a green boat.

```
mysql > SELECT DISTINCT S.sname
        -> FROM Sailors S, Reserves R, Boats B
```

```
-> WHERE S.sid=R.sid AND R.bid=B.bid AND B.color='red' AND EXISTS
        ->( SELECT R2.bid
        -> FROM Reserves R2, Boats B2
        -> WHERE R2.bid=B2.bid AND
        -> B2.color='green'AND R.sid=R2.sid);
```

```
mysql > SELECT DISTINCT S.sname
        -> FROM Sailors S, Reserves R, Boats B
        -> WHERE S.sid=R.sid AND R.bid=B.bid AND B.color='red' AND R.sid IN
        ->( SELECT R2.bid
        -> FROM Reserves R2, Boats B2
        -> WHERE R2.bid=B2.bid AND
        -> B2.color='green'AND R.sid=R2.sid);
```

```
mysql > SELECT S.sname
        -> FROM Sailors S, Reserves R, Boats B
        -> WHERE S.sid=R.sid AND R.bid=B.bid AND B.color='red'
        ->INTERSECT
         -> SELECT S2.sname
        -> FROM Sailors S2, Reserves R2, Boats B2
        -> WHERE S2.sid=R2.sid AND R2.bid=B2.bid AND B2.color='green';
```

Find the sids of all sailors who have reserved red boats but not green boats.

```
mysql > SELECT R.sid
        -> FROM Reserves R, Boats B
        -> WHERE R.bid=B.bid AND B.color='red'
        -> AND NOT EXISTS
        ( SELECT R2.bid
        -> FROM Reserves R2, Boats B2
        -> WHERE R2.bid=B2.bid AND
        -> B2.color='green' AND R.sid=R2.sid);

mysql > SELECT R.sid
        -> FROM Reserves R, Boats B
        -> WHERE R.bid=B.bid AND B.color='red'
        ->EXCEPT
        -> SELECT R2.sid
        -> FROM Reserves R2, Boats B2
        -> WHERE R2.bid=B2.bid AND B2.color='green';
```

Find all sids of sailors who have a rating of 10 or have reserved boat 104.

```
mysql > SELECT S.sid
        -> FROM sailors S
        -> WHERE S.rating=10
        ->UNION
        -> SELECT R.sid
        -> FROM Reserves R
        -> WHERE R.bid=104;
```

Nested Query:

Find the names of sailors who have reserved boat 103.

```
mysql > SELECT S.sname
        ->FROM Sailors S
        ->WHERE S.sid IN ( SELECT R.sid
                -> FROM Reserves R
                -> WHERE R.bid = 103);
```

Correlated Nested Query:
Find the names of sailors who have reserved boat 103.

```
mysql > SELECT S.sname
        ->FROM Sailors S
        ->WHERE EXISTS ( SELECT *
        -> FROM Reserves R
        -> WHERE R.bid = 103 AND R.sid = S.sid);
```

Find the names of sailors who have not reserved a red boat.

```
mysql > SELECT DISTINCT S.sname
        -> FROM Sailors S
        -> WHERE S.sid NOT IN ( SELECT R.sid
                -> FROM Reserves R, Boats B
                - > WHERE R.bid=B.bid AND B.color='red');
```

```
mysql > SELECT DISTINCT S.sname
        -> FROM Sailors S
        -> WHERE S.sid NOT IN ( SELECT R.sid
        -> FROM Reserves R
        -> WHERE R.bid IN ( SELECT B.bid
                -> FROM Boats B
                -> WHERE B.color='red'));
```

```
mysql > SELECT DISTINCT S.sname
        -> FROM Sailors S
        -> WHERE S.sid IN (SELECT R.sid
                -> FROM Reserves R
                -> WHERE R.bid NOT IN ( SELECT B.bid
                        -> FROM Boats B -> WHERE B.color='red'));
```

Find sailors whose rating is better than some sailor called Horatio.

```
mysql > SELECT *
        -> FROM Sailors S
        -> WHERE S.rating >ANY (SELECT S2.rating
                -> FROM Sailors S2
                -> WHERE S2.sname='Horatio');
```

[OR]

```
mysql > SELECT *
        -> FROM Sailors S
        -> WHERE S.rating >SOME ( SELECT S2.rating
                -> FROM Sailors S2
                -> WHERE S2.sname='Horatio');
```

Find sailors whose rating is better than every sailor called Horatio.

```
mysql > SELECT *
        -> FROM Sailors S
        -> WHERE S.rating >ALL ( SELECT S2.rating
                -> FROM Sailors S2
                -> WHERE S2.sname='Horatio');
```

Find the names of sailors who have reserved all boats.

```
mysql > SELECT DISTINCT S.sname
        -> FROM Sailors S
        -> WHERE NOT EXISTS ( SELECT B.bid
                -> FROM Boats B -> WHERE NOT EXISTS
        ( SELECT R.bid
        -> FROM Reserves R
        -> WHERE R.bid=B.bid AND R.sid=S.sid));
                                        [OR]
mysql > SELECT DISTINCT S.sname
        -> FROM Sailors S
        -> WHERE NOT EXISTS ( SELECT B.bid
                -> FROM Boats B
                -> EXCEPT ( SELECT R.bid
                        -> FROM Reserves R
                        -> WHERE R.sid=S.sid));
```

Find the names of sailors who have reserved at least two boats.

```
mysql > SELECT DISTINCT S.sname
        -> FROM Sailors S, Reserves R1, Reserves R2
        -> WHERE S.sid=R1.sid AND R1.sid=R2.sid AND R1.bid<>R2.bid;
```

Find the names of sailors who have reserved all boats called Interlake.

```
mysql > SELECT DISTINCT S.sname
        -> FROM Sailors S
        -> WHERE NOT EXISTS (SELECT B.bid
        ->FROM Boats B
        ->WHERE B.bname='Interlake' AND
        ->NOT EXISTS (SELECT R.sid
                ->FROM Reserves R
                ->WHERE R.bid=B.bid AND R.sid=S.sid));
```

Find sailors who have reserved all red boats.

```
mysql > SELECT *
        -> FROM Sailors S
        -> WHERE NOT EXISTS (SELECT B.bid
                ->FROM Boats B
                ->WHERE B.color='red' AND
                ->NOT EXISTS (SELECT R.sid
                        ->FROM Reserves R
                        ->WHERE R.bid=B.bid AND R.sid=S.sid));
```

Find the sailor name, boat id, and reservation date for each reservation.

```
mysql > SELECT S.sname, R.bid, R.day
        -> FROM Sailors S, Reserves R
        -> WHERE S.sid=R.sid
        -> GROUP BY R.day;
```

Find the sids of sailors with age over 20 who have not reserved a red boat.

```
mysql > SELECT S.sid
        -> FROM Sailors S
        -> WHERE S.age>20 AND S.sid NOT IN ( SELECT R.sid
                -> FROM Reserves R, Boats B
                -> WHERE R.bid=B.bid AND B.color='red');
```

WEEK – 6

QUERIES USING AGGREGATE FUNCTIONS

AIM :- Queries using aggregate functions(COUNT,AVG,MIN,MAX,SUM), Group by, Order by, Having.

Aggregate Operators:

COUNT The integer of values in the column
SUMThe sum of all values in the column
AVGThe average of all values in the column
MAXThe maximum value in the column
MINThe minimum value in the column
Variations of COUNT, SUM, AVG using DISTINCT

E_id	E_name	Age	Salary
101	Anu	22	9000
102	Shane	29	8000
103	Rohan	34	6000
104	Scott	44	10000
105	Tiger	35	8000
106	Alex	27	7000
107	Abhi	29	8000

(i) Create Employee table containing all Records.

MYSQL> create table emp(eid integer,ename varchar(10),age integer,salary integer);

(ii)Count integer of employee names from employee table.

MYSQL> select count(ename) from employee;

```
+--------------      +
| count(ename) |
+--------------      +
|         2          |
+--------------      +
1 row in set (0.03 sec)
```

(iii) Find the Maximum age from employee table.

MYSQL> select max(age) from employee;

```
+----------  +
| max(sal) |
+----------  +
|   10000 |
+----------  +
1 row in set (0.01 sec)
```

(iv) Find the Minimum age from employee table.

MYSQL> select min(sal) from employee;

```
+----------+
| min(sal) |
+----------+
|   10000 |
+----------+
1 row in set (0.00 sec)
```

(v)Display the Sum of age employee table.

```
MYSQL> select sum(sal) from employee;
+----------+
| sum(sal) |
+----------+
|   10000 |
+----------+
1 row in set (0.02 sec)
```

(vi)Display the Average of age from Employee table.

```
MYSQL> select avg(sal) from employee;
+------------+
| avg(sal)   |
+------------+
| 10000.0000 |
+------------+
1 row in set (0.01 sec)
```

(vii)Create a View for age in employee table.

```
MYSQL> create or replace view A as select sal from employee where sal>5000;
Query OK, 0 rows affected (0.06 sec)
```

(viii)Display views

```
MYSQL> select * from A;
+-------+
| sal   |
+-------+
| 10000 |
+-------+
1 row in set (0.00 sec)
```

(ix)Find grouped salaries of employees.(group by clause)

```
MYSQL> select salary from employee group by salary;
+-------+
|   sal   |
+-------+
| 10000 |
|  9000 |
|  8000 |
|  7000 |
|  6000 |
+------- +
```

(x).Find salaries of employee in Ascending Order.(order by clause)

```
MYSQL> select ename, sal from employee order by sal;
        +--------          +-------  +
        | ename            | sal     |
        +--------          +-------  +
        | mohith           |  NULL |
        |  abhi            |   6000 |
        |  kiran           |   7000 |
        |  joshi           |   8000 |
```

```
| raj          |  9000 |
| verma        | 10000 |
+--------      +-------  +
6 rows in set (0.03 sec)
```

(xi) Find salaries of employee in Descending Order.

```
MYSQL> select ename,sal from employee order by salary desc;
+--------      +-------  +
| ename        | sal     |
+--------      +-------  +
| verma        | 10000 |
| raj          |  9000 |
| joshi        |  8000 |
| kiran        |  7000 |
| abhi         |  6000 |
| mohith       |  NULL |
+--------      +-------  +
6 rows in set (0.03 sec)
```

(xii)Having Clause.

```
MYSQL> select ename, sal from employee where commission > 4000 group by ename, sal having sal<10000;
+--------      +-------  +
| ename        | sal     |
+--------      +-------  +
| verma        | 10000 |
| raj          |  9000 |
| joshi        |  8000 |
| kiran        |  7000 |
| abhi         |  6000 |
+--------      +-------  +
```

Find the average age of all sailors.
mysql > SELECT AVG(S.age) AS avgage FROM Sailors S;

Find the average age of sailors with a rating of 10.
mysql > SELECT AVG(S.age) FROM Sailors S WHERE S.rating=10;

Find the name and age of the oldest sailor.
mysql > SELECT S.sname, MAX(S.age) FROM Sailors S; [We got wrong answer, below one is correct]

It is not good practice; if SELECT clause uses an aggregate operation, then it must use only aggregate operations unless the query contains a GROUP BY clause.

mysql > SELECT S.sname, S.age FROM Sailors S WHERE S.age = (SELECT MAX(S2.age) FROM Sailors S2);

Count the integer of sailors.
mysql > SELECT COUNT(*) FROM Sailors;

Find the names of sailors who are older than the oldest sailor with a rating of 10.

```
mysql > SELECT S.sname FROM Sailors S WHERE S.age (SELECT MAX(S2.age) FROM Sailors S2
WHERE S2.rating=10);
```

[OR]

```
mysql > SELECT S.sname FROM Sailors S WHERE S.age > ALL ( SELECT S2.age FROM Sailors S2
WHERE S2.rating=10);
```

Count the integer of different sailor names.

```
mysql > SELECT COUNT(DISTINCT S.sname) FROM Sailors S;
```

Find the sailors with the highest rating.

```
mysql > SELECT * FROM Sailors S WHERE S.rating = ( SELECT MAX(S2.rating) FROM Sailors S2);
```

WEEK – 7

GROUP BY and HAVING Clauses:

```
mysql > SELECT [DISTINCT]select-list
        -> FROM from-list
        -> WHERE qualification
        -> GROUP BY grouping-list
        -> HAVING group-qualification;
```

Find the names of sailors who have reserved at least one boat.

Removing duplication by using GROUP BY clause

```
mysql > SELECT S.sname FROM Sailors S, Reserves R WHERE S.sid=R.sid GROUP BY R.sid;
                                    [OR]
mysql > SELECT S.sname FROM Sailors S WHERE S.sid IN (SELECT R.sid FROM Reserves R GROUP
BY R.sid HAVING COUNT(R.sid) >= 1);
```

Find the names of sailors who have reserved at least two boats.

```
mysql > SELECT S.sname  FROM Sailors S WHERE S.sid IN (SELECT R.sid FROM Reserves R GROUP
BY R.sid HAVING COUNT(R.sid) >= 2);
```

Find the names of sailors who have reserved at least two boats.

```
mysql > SELECT S.sname  FROM Sailors S WHERE S.sid IN (SELECT R.sid FROM Reserves R GROUP
BY R.sid HAVING COUNT(R.sid) >= 2);
```

Find the names of sailors who have reserved at least three boats.

```
Mysql > SELECT S.sname FROM Sailors S WHERE S.sid IN (SELECT R.sid FROM Reserves R GROUP
BY R.sid HAVING COUNT(R.sid) >= 3);
```

Find the names of sailors who have reserved all boats.

```
Mysql > SELECT S.sname FROM Sailors S WHERE S.sid IN ( SELECT R.sid FROM Reserves R
GROUP BYR.sid HAVING COUNT(R.sid)=( SELECT COUNT(B.bid) FROM Boats B));
```

Find the names of sailors who have reserved at least two boats.

```
mysql > SELECT S.sname FROM Sailors S, Reserves R WHERE S.sid=R.sid GROUP BY R.sid
HAVING COUNT(R.sid)>=2;
```

Find the age of the youngest sailor for each rating level.

```
Mysql > SELECT S.rating, MIN(S.age) FROM Sailors S GROUP BY S.rating;
```

Find the age of the youngest sailor who is eligible to vote (i.e., is at least 18 years old) for each rating level with at least two such sailors.

```
mysql > SELECT S.rating, MIN(S.age) FROM Sailors S WHERE S.age >= 18 GROUP BY S.rating
HAVING COUNT(*) > 1;
```

Find the average age of sailors for each rating level that has at least two sailors.

```
mysql > SELECT S.rating, AVG(S.age) FROM Sailors S GROUP BY S.rating HAVING COUNT(*) > 1;
                                    [OR]
```

```
mysql > SELECT S.rating, AVG(s.age) AS avgage FROM Sailors S GROUP BY S.rating HAVING 1 < ( SELECT COUNT(*) FROM Sailors S2 WHERE S.rating = S2.rating);
```

For each red boat, find the integer of reservations for this boat.

```
mysql > SELECT B.bid, COUNT(*) AS ResCount FROM Boats B, Reserves R WHERE B.bid=R.bid AND B.color='red'  GROUP BY B.bid;

SELECT B.bid, COUNT(*) AS ResCount FROM Boats B, Reserves R WHERE B.bid=R.bid GROUP BY B.bid HAVING B.color="red";
```

What is the output? IT"s an error, only columns that appear in the GROUP BY clause can appear in the HAVING clause, unless they appear as arguments to an aggregate operator in the HAVING clause.

Find the average age of sailors who are of voting age (i.e., at least 18 years old for each rating level that has at least two sailors.

```
SELECT S.rating, AVG(S.age) AS avgage FROM Sailors S WHERE S.age >= 18 GROUP BY S.rating HAVING 1 < ( SELECT COUNT(*) FROM Sailors S2 WHERE S.rating=S2.rating);
```

Find the average age of sailors who are of voting age (i.e., at least 18 years old) for each rating level that has at least two such sailors.

```
SELECT S.rating, AVG(S.age) AS avgage FROM Sailors S WHERE S.age > 18 GROUP BY S.rating HAVING 1 < ( SELECT COUNT(*) FROM Sailors S2 WHERE S.rating=S2.rating AND S2.age >=18 );
```

Find those ratings for which the average age of sailors is the minimum overall ratings

Creating VIEW:

```
mysql> show tables;  +------------------------+
                     | Tables_in_reservations |
                     +------------------------+
                     | boats                  |
                     | reserves               |
                     | sailors                |
                     +------------------------+
                     3 rows in set (0.08 sec)
```

```
mysql> CREATE VIEW IT AS (SELECT * FROM Sailors WHERE rating>5);
mysql> SHOW TABLES;
+------------------------+
| Tables_in_reservations |
+------------------------+
| boats                  |
| IT                     |
| reserves               |
| sailors                |
+------------------------+
 4 rows in set (0.02 sec)

mysql> SELECT * FROM IT;
```

LAB ASSIGNMENT:

Case Study 1:

a) By using the group by clause, display the enames who belongs to deptno 10 along with average salary.

b) Display lowest paid employee details under each department.

c) Display integer of employees working in each department and their department integer.

d) Using built in functions, display integer of employees working in each department and their department name from dept table. Insert deptname to dept table and insert deptname for each row, do the required thing specified above.

e) List all employees which start with either B or C.

f) Display only these ename of employees where the maximum salary is greater than or equal to 5000.

Case Study 2:

a) Calculate the average salary for each different job.

b) Show the average salary of each job excluding manager.

c) Show the average salary for all departments employing more than three people.

d) Display employees who earn more than the lowest salary in department 30

e) Show that value returned by sign (n) function.

f) How many days between day of birth to current date.

Case Study 3:

a) Show that two substring as single string.

b) List all employee names, salary and 15% rise in salary.

c) Display lowest paid emp details under each manager

d) Display the average monthly salary bill for each deptno.

e) Show the average salary for all departments employing more than two people.

f) By using the group by clause, display the eid who belongs to deptno 05 along with average salary.

Case Study 4:

a) Count the integer of employees in department 20

b) Find the minimum salary earned by clerk.

c) Find minimum, maximum, average salary of all employees.

d) List the minimum and maximum salaries for each job type.

e) List the employee names in descending order.

f) List the employee id, names in ascending order by empid.

Case Study 5:

a) Find the sids ,names of sailors who have reserved all boats called “INTERLAKE

b) Find the age of youngest sailor who is eligible to vote for each rating level with at least two such sailors.

c) Find the sname , bid and reservation date for each reservation.

d) Find the ages of sailors whose name begin and end with B and has at least 3 characters.

e) List in alphabetic order all sailors who have reserved red boat.

f) Find the age of youngest sailor for each rating level.

Case Study 6:

a) List the Vendors who have delivered products within 6 months from order date.

b) Display the Vendor details who have supplied both Assembled and Sub parts.

c) Display the Sub parts by grouping the Vendor type (Local or Non Local).

d) Display the Vendor details in ascending order.

e) Display the Sub part which costs more than any of the Assembled parts.

f) Display the second maximum cost Assembled part.

WEEK – 8

Removing /deleting the database with total data.
Syntax: DROP DATABASE Database_Name;

Removing /deleting the table with structure.
Syntax: DROP TABE Table_Name;

Deleting the all rows in a table with data.
Syntax: DELETE FROM Table_Name;
Ex: DELETE FROM Sailors; [delete all rows in a Sailors table]

Truncate: It is used to delete all rows in a table at once. It is faster than delete command
Syntax: TRUNCATE Table_Name;
Ex: TRUNCATE Sailors; [delete all rows in a Sailors table]

Delete the records of sailors who have rating 8 (deleting some rows in a table).
Syntax: DELETE from Table_Name WHERE Condition; [delete all the rows which satisfies the condition]
Ex: DELETE from Sailors WHERE rating=8; [delete all the rows in a sailors tables which have rating value 8]

Loading data which is present in the text into the table.
Syntax: LOAD DATA LOCAL INFILE „File Path" INTO TABLE Table_Name;
Ex: LOAD DATA LOCAL INFILE „F:\\Boats.txt" INTO TABLE Boats;

Changing the table name
Syntax: RENAME TABLE old_name TO new_name;
Ex: RENAME TABLE Catalog to Bill; [Rename table name from Catalog to Bill]
Syntax: RENAME TABLE old_1 TO new_1, old_2 TO new_2, ……;
Ex: RENAME TABLE Catalog TO Bill, Parts TO PartId; [Rename tables names from Catalog to Bill and Parts to PartId]

ALTER Command:

Changing the table name from Suppliers to Supply using the ALTER
Syntax: ALTER TABLE old_name RENAME TO new_name;
Example: alter table cancel add date date;
Query OK, 0 rows affected (0.30 sec)
Records: 0 Duplicates: 0 Warnings: 0

ADD is used to add an attribute to the table.
Syntax: ALTER TABLE Table_name ADD Attri_name Data Type;
Example: alter table cancel add date date;
Query OK, 0 rows affected (0.30 sec)
Records: 0 Duplicates: 0 Warnings: 0

To add an attribute to the table at particular position.
Example: alter table cancel add color integer after bid;
Query OK, 0 rows affected (0.74 sec)
Records: 0 Duplicates: 0 Warnings: 0
DROP is used to remove an attribute from the table.

Syntax: ALTER TABLE Table_name DROP Attri_name;
Example: alter table cancel drop color;
Query OK, 0 rows affected (0.20 sec)
Records: 0 Duplicates: 0 Warnings: 0

CHANGE is used to change the attribute name or data type or both.
Example: ALTER TABLE cancel CHANGE date bdate DATE;
Query OK, 0 rows affected (0.27 sec)
Records: 0 Duplicates: 0 Warnings: 0

[attribute date is changed to bdate with type DATE]

TO change only type of an attribute using **MODIFY**
Ex: ALTER TABLE cancel modify bid integer(15);
Query OK, 0 rows affected (0.07 sec)
Records: 0 Duplicates: 0 Warnings: 0;

SET is used to change the values of the table.
ALTER TABLE sailors SET rating = rating +1 [rating is increased by 1]

Ex: ALTER TABLE sailors MODIFY rating integer DEFAULT 5;
Query OK, 0 rows affected (0.04 sec)
Records: 0 Duplicates: 0 Warnings: 0
[if rating value is not entered then default value is set to 5]

Copy one table data in to another table.
Ex: CREATE TABLE Table2 LIKE Table1; [Table1 data is copied into table2]
mysql> create table cancel2 like cancel;
Query OK, 0 rows affected (0.09 sec)

Ex: create table sailors2 as (select * from sailors);
Query OK, 19 rows affected (0.23 sec)
Records: 19 Duplicates: 0 Warnings: 0
[Sailors table data is copied into Table2]

WEEK – 9

Creation of tables for Roadway Travels:

Bus Table:

```
mysql> CREATE TABLE Bus(BusNo VARCHAR(20),
       Source VARCHAR(20),
       Destination VARCHAR(20),
       CoachType VARCHAR(20),
       noofseats int(8),
       PRIMARY KEY(BusNo));
Query OK, 0 rows affected (0.10 sec)
```

Reservation Table:

```
mysql> CREATE TABLE Reservation(PNRNO INTEGER(9),
       JournyDate DATE NOT NULL,
       NoofSeats INTEGER(8),
       Address VARCHAR(50),
        ContactNo INTEGER(10),
       BusNo VARCHAR(20),
       SeatNo INTEGER,
       PRIMARY KEY(PNRNO),
       FOREIGN KEY(BusNo) REFERENCES bus(BusNO) ON DELETE CASCADE);
Query OK, 0 rows affected (0.12 sec)
```

Ticket Table:

```
mysql> CREATE TABLE Ticket(TicketNO INTEGER(9),
       JournyDate DATE NOT NULL,
       Age INTEGER(4),
       GENDER CHAR(10),
        Source VARCHAR(20),
       Destination VARCHAR(20),
       DeptTime VARCHAR(20),
       BusNo VARCHAR(20),
       PRIMARY KEY(TicketNo),
       FOREIGN KEY(BusNo) REFERENCES bus(BusNo) ON DELETE CASCADE);
Query OK, 0 rows affected (0.08 sec)
```

Passenger Table:

```
mysql> CREATE TABLE Passenger(PNRNO INT(9),
       TicketNO INTEGER(9),
       AadharNo INT(9),
       Name VARCHAR(20),
       Age INT(4),
       Gender  CHAR(10),
       ContactNo INT(9),
       PRIMARY KEY(PNRNO),
       FOREIGN KEY(TicketNo) REFERENCES Ticket(TicketNo));
       Query OK, 0 rows affected (0.14 sec)
```

Cancellation table:

```
CREATE TABLE Cancellation(PNRNO INT(9),
        JournyDate DATE NOT NULL,
        SeatNo INTEGER,
        ContactNo INT(9),
        PRIMARY KEY(PNRNO));
        Query OK, 0 rows affected (0.08 sec)
```

Display unique *PNR_No* of all passengers.

```
mysql> SELECT DISTINCT P.PNRNO FROM Passenger AS P;
```

Display all the *names* of male passengers.

```
mysql> SELECT P.name FROM Passenger P WHERE Gender='Male';
```

Display the *ticket integers* and *names* of all the passengers.

```
mysql> SELECT P.TicketNo, P.name FROM Passenger P;
```

Display the *source* and *destination* having journey time more than 10 hours.

```
mysql> SELECT T.Source, T.Destination FROM Ticket T WHERE DeptTime – JournyDate > 10;
```

Find the *ticket integers* of the passengers whose name start with „A‟ and ends with „H‟.

```
mysql> SELECT P.TicketNo FROM Passenger P WHERE P.name LIKE ‘A%H’;
```

Find the *names* of passengers whose age is between 30 and 45.

```
mysql> SELECT P.name FROM Passenger P WHERE Age BETWEEN 30 AND 45;
```

Display all the passengers *names* beginning with „A‟

```
mysql> SELECT P.name FROM Passenger P WHERE P.name LIKE ‘A%’;
```

Display the sorted list of passengers‟ *names.*

```
mysql> SELECT P.name FROM Passenger P ORDER BY P.name ASC;
```

Display the *Bus integers* that travel on Sunday and Wednesday.

```
mysql> SELECT T.BusNo FROM Ticket T WHERE DAYOFWEEK(JournyDate) IN (1,4);
```

Display the *details* of passengers who are traveling in AC or NON_AC

```
mysql> SELECT * FROM Bus B, Reservation R, Passenger P WHERE B.BusNo=R.BusNo AND
R.PNRNO=P.PNRNO AND (B.CoachType=’AC’ OR B.CoachType=’NON_AC’);
```

Find the distinct *PNR integers* that are present

```
mysql> SELECT COUNT(DISTINCT R.PNRNO) FROM Reservation R;
```

Find the *total* integer of cancelled seats.

```
mysql> SELECT COUNT(R.NofSeats) FROM Cancellation C, Reservation R WHERE C.PNRNO =
R.PNRNO;
```

Write a Query to display the Information present in the Passenger and cancellation tables. *Hint: Use UNION Operator*

```
mysql> SELECT * FROM Passenger P  UNION SELECT * FROM Cancellation C where C.PNRNO =
P.PNRNO;
```

Display the details of passengers who are traveling either in AC or NON_AC. *Hint: Using only IN operator*
Write a Query to display different travelling options available in British Airways. Display the integer of days in a week on which the 9W01 bus is available. Find *integer* of tickets booked for each PNRNO using GROUP BY CLAUSE.
Hint: Use GROUP BY on PNRNO.

mysql> SELECT SUM(R.NoofSeats) FROM Reservation R GROUP BY PNRNO;

Find the *integer* of tickets booked in each class where the integer of seats is greater than 1.
Hint: Use GROUP BY, WHERE and HAVING CLAUSES.
mysql> SELECT SUM(R.PNRNO) FROM Reservation R, Bus B WHERE R.BusNo = B.BusNo AND R.NoofSeats>1 GROUP BY (B.CoachType);

Write a Query to count the integer of tickets for the buses, which travelled after the date '14/3/2009'.
Hint: Use HAVING CLAUSES.

1. **Display unique PNR_no of all passengers.**
   mysql> select distinct pnrno from passenger;

2. **Display all the names of male passengers.**
   mysql> select pname from passenger where gender='m';

3. **Display ticket integers and names of all the passengers.**
   mysql> select tic_no,pname from ticket t,passenger p where t.tic_no=p.ticinteger;

4. **Display the source and destination having journey time more than 10 hours.**
   mysql> select source,dest from ticket where hour(timediff(reatime,deptime))>10;

5. **Find the ticket integers of passengers whose name starts with "A" and ends with "H".**
   mysql> select tic_no from ticket where tic_no=any (select ticinteger from passenger where pname like 'a%h');

6. **Find the name of passengers whose age is between 30 and 45.**
   mysql> select pname from passenger where age between 30 and 45;

7. **Display all the passengers names beginning with "A".**
   mysql> select all pname from passenger where pname like 'a%';

8. **Display the sorted list of passengers names.**
   mysql> select pname from passenger order by pname;
9. **Display the Bus integers that travel on Sunday and Wednesday.**
   mysql> select busno from bus where busno in(select businteger from ticket where dayofweek(joudate)=1 or dayofweek(joudate)=4);
10. **Display the details of passengers who are traveling either in AC or NON_AC.**
    mysql> select pname,pnrno,age,gender from passenger where ticinteger in(select tic_no from ticket where businteger in(select busno from bus555 where bustype='ac' or bustype='nonac'));

You are going to practice the queries using Aggregate functions (COUNT, SUM, AVG, MAX and MIN), GROUP BY, HAVING AND Creation of Views.

1. Write a query to display the information present in the Passenger and Cancellation tables.
mysql> select pnrno from passenger union select pnrinteger from cancel;

2. Write a query to display the businteger with source and destination available in Roadway Travels.
mysql> select busno,source,dest from bus,ticket where busno=businteger group by businteger;

3. Display the integer of days in a week on which TS444 bus is available.
mysql> select count(joudate) from ticket where businteger in(select busno from bus where busno='ts444') and joudate between '2020-08-14' and '2020-08-20';

4. Find the ticket integers booked for each PNR_no using Group By clause.
mysql> select sum(noofseats) as reserved_seats,pnrinteger from reserve where status='yes' group by pnrinteger;

5. Find the distinct PNR integers that are present.
mysql> select distinct pnrinteger from reserve;

6. Find the integer of tickets booked for each bus with bustype where the integer of seats is greater than 1.
mysql> select busno,bustype,sum(noofseats) as booked_seats from bus,reserve,ticket,passenger where busno=businteger and tic_no=ticinteger and pnrno=pnrnumbe r and status='yes'group by tic_no having count(*)>=1;

7. **Find the total integer of cancelled seats.**
   mysql> select sum(noofseats) as cancelled_seats from cancel where status='yes';
8. **Write a query to count the integer of tickets for the buses which traveled after the date „2020-08-06".**
   mysql> select busno,bustype,sum(noofseats) as booked_seats from bus,reserve,t icket,passenger where busno=businteger and tic_no=ticinteger and pnrno=pnrnumbe r and status='yes'and joudate>'2020-8-6' group by tic_no having count(*)>=1;

Creation of Views:

mysql> create view takes1 as
-> select tic_no,pname from ticket555,passenger where tic_no=ticinteger;
mysql> select tic_no from takes1;

Dropping of Views:mysql> drop view takes1;

WEEK – 10
PROGRAMS ON PL/SQL
stored procedure

- A stored procedure is nothing more than prepared SQL code that you save so you can reuse the code over and over again.
- So, if you think about a query that you write over and over again, instead of having to write that query each time you would save it as a stored procedure and then just call the stored procedure to execute the SQL code that you saved as part of the stored procedure.
- In addition to running the same SQL code over and over again you also have the ability to pass parameters to the stored procedure, so depending on what the need is the stored procedure can act accordingly based on the parameter values that were passed.
- There are various options that can be used to create stored procedures.
- Before you create a stored procedure you need to know what your end result is, whether you are selecting data, inserting data, etc.

Ex: SELECT * FROM student.

DELIMITER //

To create a stored procedure to do this the code would look like this:
CREATE PROCEDURE proc_name(arguments)
Begin
SELECT * FROM students
end
When creating a stored procedure you can either use CREATE PROCEDURE or CREATE PROC.
After the stored procedure name you need to use the keywords "begin" & “end” and then the rest is just the regular SQL code that you would normally execute.
One thing to note is that you cannot use the keyword "end" in the stored procedure.
Once the SQL Server compiler sees "end" it assumes it is the end of the batch.

Also, you cannot change database context within the stored procedure such as using "USE dbName" the reason for this is because this would be a separate batch and a stored procedure is a collection of only one batch of statements.

Ex:

```
create procedure p1(p_age int)
 begin
SELECT S.rating, S.age
 FROM Sailors S
 WHERE S.age >= p_age;
 end
```

mysql> SHOW PROCEDURE STATUS;

mysql> call p1(30) //

```
+--------+------+
| rating | age  |
+--------+------+
|      9 |   50 |
|      2 |   70 |
|      7 |   30 |
|     10 |   49 |
|     10 |   49 |
|      8 |   39 |
|     10 |   49 |
|      7 |   45 |
|     10 |   49 |
|      1 |   33 |
|     10 |   35 |
|      7 |   35 |
|      9 |   35 |
|      3 |   64 |
+--------+------+
14 rows in set (0.00 sec)

Query OK, 0 rows affected (0.01 sec)
```

WEEK – 11

TRIGGERS

TRIGGERS In this week you are going to work on the triggers.

Creation of insert trigger, delete trigger, update trigger. (Practice triggers using above database.)

A Trigger is a named database object which defines some action that the database should take when some databases related event occurs. Triggers are executed when you issues a data manipulation command like INSERT, DELETE, UPDATE on a table for which the trigger has been created.

They are automatically executed and also transparent to the user. But for creating the trigger the user must have the CREATE TRIGGER privilege.

CREATE TRIGGER

The general syntax of CREATE TRIGGER is :

CREATE TRIGGER trigger nametrigger timetrigger event ON tbl_name
FOR EACH ROW
trigger_statement

By using above statement we can create the new trigger. The trigger can associate only with the table name and that must be refer to a permanent table.

Trigger_time means trigger action time. It can be BEFORE or AFTER. It is used to define that the trigger fires before or after the statement that executed it.

Trigger_event specifies the statement that executes the trigger. The trigger_event can be any of the DML Statement : INSERT, UPDATE, DELETE.

We can not have the two trigger for a given table, which have the same trigger action time and event. For Instance : we cannot have two BEFORE INSERT triggers for same table. But we can have a BEFORE INSERT and BEFORE UPDATE trigger for a same table.

Trigger_statement have the statement that executes when the trigger fires but if you want to execute multiple statement the you have to use the BEGIN…END compound statement.

We can refer the columns of the table that associated with trigger by using the OLD and NEW keyword. OLD.column_name is used to refer the column of an existing row before it is deleted or updated and NEW.column_name is used to refer the column of a new row that is inserted or after updated existing row.

In INSERT trigger we can use onlyNEW.column_name because there is no old row and in a DELETE trigger we can use only OLD.column_name because there is no new row. But in UPDATE trigger we can use both, OLD.column_name is used to refer the columns of a row before it is updated and NEW.Column_name is used to refer the column of the row after it is updated.

TRIGGERS

Update Trigger:

DELIMITER //

Update Trigger:

```
mysql> create trigger t1 before update on boats
for each row
begin
if new.color='red' then
set new.color=old.color;
else
set new.color=new.color;
end if;
end//

Query OK, 0 rows affected (0.08 sec)

select * from boats //
+-----+-----------+--------+
| bid | bname     | color  |
+-----+-----------+--------+
|   1 | Simple    | red    |
|   2 | Mediam    | yellow |
|   3 | high      | blue   |
|   4 | other     | Green  |
| 101 | Interlake | blue   |
| 102 | Interlake | red    |
| 103 | Clipper   | green  |
| 104 | Marine    | red    |
+-----+-----------+--------+
8 rows in set (0.00 sec)

mysql> update boats set color= 'red' where bid=4 //
Query OK, 0 rows affected (0.02 sec)
Rows matched: 1  Changed: 0  Warnings: 0

mysql> select * from boats //
+-----+-----------+--------+
| bid | bname     | color  |
+-----+-----------+--------+
|   1 | Simple    | red    |
|   2 | Mediam    | yellow |
|   3 | high      | blue   |
|   4 | other     | Green  |
| 101 | Interlake | blue   |
| 102 | Interlake | red    |
| 103 | Clipper   | green  |
| 104 | Marine    | red    |
+-----+-----------+--------+
8 rows in set (0.00 sec)
```

```
mysql> update boats set color= 'green' where bid=4 //
Query OK, 1 row affected (0.06 sec)
Rows matched: 1  Changed: 1  Warnings: 0

mysql> select * from boats //
+-----+-----------+--------+
| bid | bname     | color  |
+-----+-----------+--------+
|   1 | Simple    | red    |
|   2 | Mediam    | yellow |
|   3 | high      | blue   |
|   4 | other     | green  |
| 101 | Interlake | blue   |
| 102 | Interlake | red    |
| 103 | Clipper   | green  |
| 104 | Marine    | red    |
+-----+-----------+--------+
8 rows in set (0.00 sec)
```

Insert Trigger:

```
mysql> create trigger t2
before insert on sailors
for each row
begin
if new.age>40 then
set new.rating='10';
else
set new.rating=new.rating;
end if;
end //
Query OK, 0 rows affected (0.14 sec)

mysql> select * from sailors //
+-----+---------+--------+------+
| sid | sname   | rating | age  |
+-----+---------+--------+------+
|   2 | murthy  |      9 |   50 |
|   3 | vamsi   |      2 |   70 |
|   4 | kishna  |      7 |   30 |
|   5 | sree    |      5 |   25 |
|  11 | Leo     |     10 |   49 |
|  13 | Leo     |     10 |   49 |
|  15 | Leo     |     10 |   49 |
|  22 | Dustin  |      7 |   45 |
|  29 | Brutus  |      1 |   33 |
|  58 | Rusty   |     10 |   35 |
+-----+---------+--------+------+
19 rows in set (0.00 sec)

mysql> insert into sailors values(23,'Leo',8, 49) //
Query OK, 1 row affected (0.36 sec)
```

```
mysql> select * from sailors //
+-----+---------+--------+------+
| sid | sname   | rating | age  |
+-----+---------+--------+------+
|   2 | murthy  |      9 |   50 |
|   3 | vamsi   |      2 |   70 |
|   4 | kishna  |      7 |   30 |
|   5 | sree    |      5 |   25 |
|  11 | Leo     |     10 |   49 |
|  13 | Leo     |     10 |   49 |
|  15 | Leo     |     10 |   49 |
|  22 | Dustin  |      7 |   45 |
|  23 | Leo     |     10 |   49 |
|  29 | Brutus  |      1 |   33 |
|  58 | Rusty   |     10 |   35 |
| +-----+---------+--------+------+
20 rows in set (0.00 sec)

mysql> insert into sailors values(14,'Leo',8, 39) //
Query OK, 1 row affected (0.07 sec)

mysql> select * from sailors //

mysql> select * from sailors //
+-----+---------+--------+------+
| sid | sname   | rating | age  |
+-----+---------+--------+------+
|   2 | murthy  |      9 |   50 |
|   3 | vamsi   |      2 |   70 |
|   4 | kishna  |      7 |   30 |
|   5 | sree    |      5 |   25 |
|  11 | Leo     |     10 |   49 |
|  13 | Leo     |     10 |   49 |
|  14 | Leo     |      8 |   39 |
|  15 | Leo     |     10 |   49 |
|  22 | Dustin  |      7 |   45 |
|  23 | Leo     |     10 |   49 |
|  29 | Brutus  |      1 |   33 |
|  58 | Rusty   |     10 |   35 |
+-----+---------+--------+------+
21 rows in set (0.00 sec)
```

Delete Trigger:

```
mysql> create trigger t3 before delete on reserves
for each row
begin
insert into cancel values(old.sid, old.bid, old.day);
end //

Query OK, 0 rows affected (0.14 sec)
```

```
mysql> select * from reserves //
+-----+-----+------------+
| sid | bid | day        |
+-----+-----+------------+
|   2 |   1 | 2021-04-29 |
|   3 |   2 | 2021-04-22 |
|  64 | 101 | 2012-05-09 |
|  64 | 102 | 2012-08-09 |
|  74 | 103 | 2012-08-09 |
+-----+-----+------------+
5 rows in set (0.00 sec)

mysql>delete from reserves where sid=64 //
Query OK, 2 rows affected (0.00 sec)

mysql>select *from reserves //
+-----+-----+------------+
| sid | bid | day        |
+-----+-----+------------+
|   2 |   1 | 2021-04-29 |
|   3 |   2 | 2021-04-22 |
|  74 | 103 | 2012-08-09 |
+-----+-----+------------+
3 rows in set (0.00 sec)

mysql>select *from cancel //
+-----+-----+------------+
| sid | bid | day        |
+-----+-----+------------+
|  64 | 101 | 2012-05-09 |
|  64 | 102 | 2012-08-09 |
+-----+-----+------------+
2 rows in set (0.00 sec)
```

WEEK – 12

Cursors

- MySQL 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.

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 −
DECLARE s1 CURSOR FOR SELECT * FROM sailors

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 s1;

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 s1 INTO s1_sid, s1_sname, s1_rating, s1_age;

Closing the Cursor

Closing the cursor means releasing the allocated memory. For example, we will close the above-opened cursor as follows −
CLOSE s1;

Ex:1

```
create procedure mycur1(sa_id int)
begin
declare v_sname varchar(30);
declare v_rating int;
declare v_age int;
declare c1 cursor for select  sname, rating, age from sailors where sid = sa_id;
open c1;
fetch c1 into v_sname,v_rating,v_age;
select v_sname,v_rating,v_age;
close c1;
end //
Query OK, 0 rows affected (0.00 sec)

call mycur10(9) //
+---------      +----------      +-------      +
| v_sname      | v_rating       | v_age        |
+---------      +----------      +-------      +
| BAb          |       5        |   25         |
+---------      +----------      +-------      +
1 row in set (0.00 sec)

Query OK, 0 rows affected (0.01 sec)
```

Ex: 2

```
create procedure mycur2(sa_rating int)
begin
declare v_sname varchar(30);
declare v_sid int;
declare v_age int;
declare c1 cursor for select  sid,sname,age from sailors where rating=sa_rating;
open c1;
fetch c1 into v_sid,v_sname,v_age;
select v_sid,v_sname,v_age;
close c1;
end //
Query OK, 0 rows affected (0.00 sec)

call mycur222(5) //
call mycur222(5) //
+-------      +---------      +-------      +
| v_sid        | v_sname      | v_age        |
+-------      +---------      +-------      +
|    5         | sree         |   25         |
+-------      +---------      +-------      +
1 row in set (0.00 sec)

Query OK, 0 rows affected (0.01 sec)
```

Ex:3

```
create procedure mycur3(sa_rating int)
begin
declare finished int default 0;
declare count int default 0;
declare v_sname varchar(30);
declare v_sid int;
declare v_age int;
declare c1 cursor for select  sid,sname,age from sailors where rating=sa_rating;
declare continue handler for not found set finished=1;
open c1;
getcur : loop
fetch c1 into v_sid,v_sname,v_age;
if finished=1 then
leave getcur;
end if;
set count =count + 1;
select v_sid,v_sname,v_age;
end loop;
close c1;
select count;
end //
```

Query OK, 0 rows affected (0.00 sec)

call mycur3(5) //

```
mysql> call mycur3(5) //
+-------        +---------      +-------        +
| v_sid         | v_sname       | v_age         |
+-------        +---------      +-------        +
|    5          | sree          |   25          |
+-------        +---------      +-------        +
1 row in set (0.00 sec)

+-------        +---------      +-------        +
| v_sid         | v_sname       | v_age         |
+-------        +---------      +-------        +
|    6          | BB            |   25          |
+-------        +---------      +-------        +
1 row in set (0.01 sec)

+-------        +---------      +-------        +
| v_sid         | v_sname       | v_age         |
+-------        +---------      +-------        +
|    7          | BAB           |   25          |
+-------        +---------      +-------        +
1 row in set (0.03 sec)
```

```
+-------        +---------      +-------        +
| v_sid         | v_sname       | v_age         |
+-------        +---------      +-------        +
|    8 | BABB   |    25 |
+-------        +---------      +-------        +
1 row in set (0.04 sec)

+-------        +---------      +-------        +
| v_sid         | v_sname       | v_age         |
+-------        +---------      +-------        +
|    9          | BAb           |    25         |
+-------        +---------      +-------        +
1 row in set (0.04 sec)

+-------+
| count |
+-------+
|    5  |
+-------+
1 row in set (0.05 sec)

Query OK, 0 rows affected, 1 warning (0.06 sec)
```