DATA BASE MANAGEMENT SYSTEMS (19CS203)

Regulation – R19
Lab manual for the Academic Year (2020-21) Semester- I

DEPARTMENT OF COMPUTER SCIENCE & ENGINEERING

VIGNAN'S
Foundation for Science, Technology & Research
(Deemed to be University)
-Estd. u/s 3 of UGC Act 1956

Vadlamudi, 522 213, AP, India

PREFACE

About the course / Lab

This course is aimed at offering the SQL Programming to the students. It starts with the basics of creating a relational database. The objective of this course is to enable the students to understand database design, expressing queries using SQL, query optimization and transaction processing.

Upon completion of this course,

- The student will be able design a Relational Database for real world applications,
- Evaluate and Optimize Queries.
- Express Queries using MYSQL WORKBENCH

INDEX

Lab Evaluation Guidelines

I. Guidelines for Lab teacher:

A. Before commencement of lab class work:

1. The detailed information consisting of experiments, batch formations, experiment schedules, etc., will be displayed / informed to the students one week before the commencement of the semester so that the student comes prepared for the session.
2. Before commencement of laboratory classes, the teachers and instructor handling the courses should practice the experiments.
3. Copies of the lab manual (soft copy) will be made available to the students along with the schedule. The lab manual will consist of the list of equipment, detailed procedure to conduct the experiment, format for record writing, algorithm or procedure, circuit diagrams, list of equipment/tools needed, sample calculations, sample graphs, outcomes for each experiment and possible set of short questions to help student's gain critical understanding. Three hardcopies are to be made available during the lab sessions one is master copy and the other two copies are for student's ready reference during lab sessions.

B. During Laboratory Sessions:

1. Conduct a demo session of all the experiments in that cycle to students. Explain them about other guidelines to be followed in the lab.
2. A logbook is to be maintained by the instructor in which student will sign every time when he visits the lab to do the experiment.
3. Verify the student record before allowing him to perform experiment. Previous experiment completion and preparedness for performing the experiment scheduled on that day has to be checked.
4. Evaluate the performance as per the table given below. The set of parameters may slightly differ from one laboratory to the other and will be announced before the commencement of the lab session. These parameters are to be assessed for each laboratory session. The lab teacher has to use red pen to give evaluation results on record.

Continuous Evaluation of Laboratory

S.No	Component	Marks
1	Preparedness	2M
2	Coding	4M
3	Testing	2M
4	Viva	2M
Total		10M

5. These marks are to be marked in record, component wise for each experiment and total marks are to be entered into section wise lab sheet.

This assessment is to be carried out for each practical session and the average mark of all the sessions is to be considered and finally scaled down to 30 marks. An internal laboratory examination will be conducted for another 20 marks.

Absent students / not performed students:

Students can do the experiments after class hrs.

Internal Laboratory Examination:

The internal laboratory examination shall be conducted around the middle of the semester. The examination is to be conducted, by a team of two examiners, one who conducts the laboratory sessions and the other appointed by the Deputy HoD. The scheme of evaluation is given below.

Component	**Marks**		
	Internal Examiner (Lab Teacher)	**External Laboratory Examiner (Appointed by Deputy HoD)**	**Total**
Objective & Procedure write up including outcomes	5	5	10
Coding	10	10	20
Testing	5	5	10
Viva voce	0	10	10
Total Marks	**20**	**30**	**50**

The teacher shall make a comparative statement of these lab marks with the student's previous aggregate percentage of marks. When the difference is more than 20%, the remarks of the teacher are to be endorsed with full justification and displayed at notice boards. A committee appointed by the Dean Evaluation will scrutinize all such comparisons and take necessary action.

C. End – semester Laboratory Evaluation:

End semester examination for each practical course is conducted jointly by both internal and external examiners. The examiners are appointed by Dean, Evaluation from the panel of examiners suggested by the respective Heads of the Department. To maintain the objectivity and seriousness of the students towards the lab curriculum and lab examinations, a panel of large number of examiners, four times to the actual requirement shall be suggested by the HoD, at least one month in advance and submit the details to the Dean Evaluation. The Dean Evaluation will select the examiners on a random basis. After examination, at least 10% of the scripts are to be audited by a three-member committee appointed by the Dean-Evaluation. For every class, the mean of the lab marks and theory marks are to be compared and if there is a deviation of more than 20% the reasons are to be analyzed and documented. The scheme of evaluation may vary depending on the nature of laboratory, which shall be shared with the student by the laboratory in-charge and also stamped on the answer scripts. The general scheme of evaluation is given in

Table below.

End Semester Evaluation Pattern of Labs

Component	Marks		
	Internal Examiner	**External Laboratory Examiner**	**Total**
Objective & Procedure write up including outcomes	5	5	10
Coding	5	5	10
Testing	10	10	20
Viva voce	0	10	10
Total Marks	**20**	**30**	**50**

II. Guidelines to lab instructor / programmers:

A) Before commencement of semester:

1. Along with lab teacher do all the experiments offered for students in that laboratory.
2. Familiarize yourself about purpose of the experiment, outcomes, result analysis, probable sources of errors in readings etc...
3. Collect the following information
   a) List of experiments
   b) Instructions related to that lab
   c) Lab manual – 3 hard copies and one softcopy
   d) Sample lab record – 3 hard copies
   e) Logbook / File
   f) Format of stamp to be printed on record sheets
4. Make the stamp and stamp pad ready for experiments. Before commencement of lab session check the stamp and stamp pad ready for the use by lab teachers.

B) One or two days before the lab session:

1. Check the functioning of systems, instruments and take action if found defective.
2. Maintain cleanness in the laboratory

C) During lab session:

1. Allow students having appropriate dress code and ID cards into lab.

2. Ask the student to make entry into the logbook. Verify with reference to the batch wise planning of experiments. Deviations if any inform to faculty.
3. Make rounds in the laboratory, help the students to perform the experiment and clarify their doubts.
4. While helping the students, please remember that the students should be able to do the experiment on their own.
5. After class work, clean the system and make the lab ready for next session.

D) During examination:

1. Obtain the student wise experiment schedules from the lab teacher and make the lab ready for examination.
2. Collect necessary stationary from the examination section.
3. Follow the instructions given by lab teacher during examinations.
4. After evaluation, return the examination records to examination section.

E) Records to be maintained by lab instructor:

1. Log sheet for each lab.

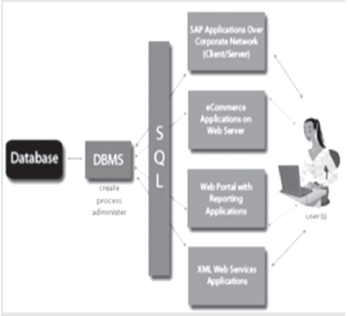
19CS203 DATABASE MANAGEMENT SYSTEMS

Hours Per Week :

L	T	P	C
3	-	2	4

Total Hours :

L	T	P	CS	WA/RA	SSH	SA	S	BS
45	-	30	5	5	30	20	5	5



Database
DBMS
create
process
administer
S
Q
L
SAP Applications Over Corporate Network (Client/Server)
eCommerce Applications on Web Server
Web Portal with Reporting Applications
XML Web Services Applications
user (s)

Source: *https:cdn.sparkfun.com/ assets/*

COURSE DESCRIPTION AND OBJECTIVES:

This course presents an introduction to database management systems with an emphasis on how to organize, maintain and retrieve data efficiently from a relational database. It also focuses on requirements gathering and conceptual, logical, physical database design. The objective of the course is to enable the student to understand database design, exprssing queries using SQL, query optimization and transaction processing.

Course Outcomes:

Upon completion of the course, the student will be able to achieve the following outcomes:

COs	Course Outcomes	POs
1	Develop an E-R model for real life applications.	2,10
2	Design and normalize databases for real time applications.	1, 3
3	Devise queries using Relational Algebra, Relational Calculus and SQL.	1
4	Evaluate and optimize queries	4
5	Express queries using database tools like Oracle, DB2, MYSQL, Mongo DB.	5, 10

SKILLS:

- ✓ *Understand functional components of the DBMS.*
- ✓ *Devise q*ueries using Relational Algebra and SQL.
- ✓ *Develop E-R model for real life applications.*
- ✓ *Design of relational databases for real world applications.*
- ✓ *Evaluate and optimize queries.*
- ✓ *Understand transaction processing, Concurrency control and recovery techniques.*

UNIT– I **L- 8**

DATABASES AND DATABASE USERS: Introduction; Characteristics of the database approach; Actors on the scene; Advantages of using DBMS approach.

DATABASE SYSTEM CONCEPTS AND ARCHITECTURE: Data models, Schemas and instances; Three-Schema architecture and data Independence; Database languages and interfaces; The database system environment; Centralized and Client - Server architectures for DBMS.

CONCEPTUAL DATA MODELING AND DATABASE DESIGN: Entity types, Entity sets, Attributes and keys; Relationship types, Relationship sets, Roles and structural constraints; Weak entity types; Relationship types of degree higher than two.

ENHANCED ENTITY RELATIONSHIP (EER) MODEL: Subclasses, Superclasses and inheritance, Specialization and generalization, Constraints and characteristics of specialization and generalization hierarchies.

UNIT – II **L- 12**

RELATIONAL DATABASE DESIGN BY ER- AND EER-TO-RELATIONAL MAPPING: Relational database design using ER-to-Relational mapping; Mapping EER model constructs to relations.

THE RELATIONAL DATA MODEL AND RELATIONAL DATABASE CONSTRAINTS: Relational model concepts; Relational model constraints and Relational database schemas. **Basic SQL:** SQL data definition and data types; Specifying constraints in SQL, Basic retrieval queries in SQL; INSERT, DELETE, and UPDATE statements in SQL.

MORE SQL: COMPLEX QUERIES, TRIGGERS, VIEWS: More complex SQL retrieval queries; Specifying constraints as assertions and actions as triggers; Views (virtual tables) in SQL.

RELATIONAL ALGEBRA : Unary relational operations - SELECT and PROJECT; Relational algebra operations from set theory; Binary relational operations - JOIN and DIVISION.

UNIT – III **L- 9**

BASICS OF FUNCTIONAL DEPENDENCIES AND NORMALIZATION FOR RELATIONAL DATABASES: Informal design guidelines for relation schemas; Functional dependencies-inference rules, equivalence and minimal cover; Normal forms based on primary keys; Boyce-Codd normal form; multivalued dependency and 4NF; Join dependencies and 5NF; Properties of relational decompositions.

QUERY PROCESSING AND OPTIMIZATION: Phases of query processing; Translating SQL Queries into Relational Algebra and other operators; Query trees and Heuristics for query optimization.

UNIT – IV **L- 10**

INTRODUCTION TO TRANSACTION PROCESSING CONCEPTS AND THEORY: Introduction to transaction processing; Transaction and system concepts; Desirable properties of transactions; Characterizing schedules based on serializability.

CONCURRENCY CONTROL TECHNIQUES: Two-phase locking techniques for concurrency control; Concurrency control based on timestamp ordering.

DATABASE RECOVERY TECHNIQUES: Recovery concepts; Shadow paging; The ARIES recovery algorithm.

UNIT - V **L- 6**

INDEXING STRUCTURES FOR FILES AND PHYSICAL DATABASE DESIGN: Single level and multi level indexing; Dynamic multi level indexing using B trees and B+ trees.

NOSQL DATABASES AND BIG DATA STORAGE SYSTEMS: Introduction to NoSQL systems; Document-based NoSQL systems and MongoDB.

LABORATORY EXPERIMENTS

LIST OF EXPERIMENTS **TOTAL HOURS: 30**

1. Design Conceptual database schema using ER Modeling Software Tools.
2. Development of Relational Database schemas for Company/Student/Sailors/ using DDL constructs of SQL.
3. Specifying various DML Commands such as select, insert, update etc. of SQL on Relational Database.
4. Specifying various DCL and TCL constructs of SQL on Relational Database.
5. Development of Relational Database schemas by specifying different types of Constraints
6. Specifying queries using Relational Database operators (Arithmetic, Logical & comparison) and string matching constructs of SQL.
7. Expressing queries using Aggregate Functions of SQL on Relational Database.
8. Queries on Relational Database using GROUP BY, HAVING and ORDER BY clauses of SQL.
9. Design and Development of company database and expressing Nested queries using SQL.
10. Design and Development of sailors database and specifying queries using different types of JOINs.
11. Creation and dropping of VIEWS.
12. Implementation of PL/SQL programs with Control Structures.
13. Implementation of PL/SQL programs with Procedures.
14. Implementation of PL/SQL programs with Function.
15. Implementation of PL/SQL programs with Triggers.

TEXT BOOK:

1. Ramez, Elmasri and Shamkant B. Navathe, “Fundamentals of Database Systems”, 7th edition, Pearson Education, 2016.

REFERENCE BOOKS:

1. Raghu Rama Krishnan and Johannes Gehrke, “Database Management Systems”, 3rd edition, Tata McGraw Hill, 2013.
2. Abraham Silberschatz, Henry F.Korth and S.Sudarshan, “Database System Concepts”, 6th edition, Tata McGraw Hill, 2010.

Branch: Computer Science & Engineering
Year & Semester: II Year I Semester
Lab Name: DATA BASE MANAGEMENT SYSTEMS (19CS203)

II. LIST OF EXPERIMENTS

Exp.No	NAME OF THE EXPERIMENTS	CO's	PO's
1	Development of Relational Database Schemas for Sailors using DDL constructs of SQL	2	1,3
2	Specifying various DML Commands such as select, insert, update etc of SQL on Relational Database.	2	1,3
3	Specifying DCL constructs of SQL on Relational Database	5	5,10
4	Development of Relational Database Schemas by specifying different types of Constraints.	2	1,3
5	Specifying queries using Relational Database operators (Arithmetic; Logical & Comparison) and string-matching constructs of SQL	5	5,10
6	Expressing Queries using Aggregate Functions of SQL on Relational Database.	5	5,10
7	Queries on Relational Database using GROUP BY, HAVING and ORDER BY clauses of SQL.	5	5,10
8	Design and Development of databases and expressing Nested queries using SQL.	5	5,10
9	Design and Development of databases and specifying queries using different types of JOINS.	5	5,10
10	Creation and dropping of VIEWS	5	5,10

III. EXPERIMENTS

EXPERIMENT-1: Develop Relational Database Schema for **Boat Club** database using DDL.

DESCRIPTION:

SQL (Structured Query Language):
Structured Query Language is a database computer language designed for managing data in relational database management systems (RDBMS), and originally based upon Relational Algebra. Its scope includes data query and update, schema creation and modification, and data access control. SQL was one of the first languages for Edgar F. Codd's relational model and became the most widely used language for relational databases.

SQL language is sub-divided into several language. *They are*:

1. **Data Definition Language (DDL)**
2. **Data Manipulation Language (DML)**
3. **Transactional Control Language (TCL)**
4. **Data Control Language (DCL)**

DATA TYPES:

- **CHAR (Size):** This data type is used to store character strings values of fixed length. The maximum number of characters is 255characters.
- **VARCHAR (Size) / VARCHAR2 (Size)**: This data type is used to store variable length alphanumeric data. The maximum character can hold is 2000character.
- **NUMBER (P, S):** The NUMBER data type is used to store number (fixed or floating point). Number of virtually any magnitude may be stored up to 38 digits of precision. Number as large as $9.99 * 10^{124}$.
- **DATE:** This data type is used to represent date and time. The standard format is DD- MM-YY as in17-SEP-2009.Date time stores date in the 24-Hours format.
- **LONG:** This data type is used to store variable length character strings containing up to 2GB. Long data can be used to store arrays of binary data in ASCII format. LONG values cannot be indexed, and the normal character functions such as SUBSTR cannot be applied.
- **RAW:** The RAW data type is used to store binary data, such as digitized picture or image. Data loaded into columns of these data types are stored without any further conversion. RAW data type can have a maximum length of 255 bytes. LONG RAW data type can contain up to 2GB.

DATA DEFINITION LANGUAGE (DDL): The Data Definition Language (DDL) is used to create and destroy databases and database objects. These commands will primarily be used by database administrators during the setup and removal phases of a database project.

1. CREATE
2. ALTER
3. DROP
4. RENAME

COMMANDS:

1. CREATE: This is used to create a new relation (table)

SYNTAX: CREATE TABLE <relation name/table name> (field_1data_type(size), field_2 datatype(size), .. .);
EXAMPLE: Create table student (SID varchar (20), sname char(30));

2.ALTER:

- **ALTER TABLE ...ADD:** This is used to add some extra fields into existing relation.
  SYNTAX: ALTER TABLE relation name ADD (new field_1name data type(size));
  EXAMPLE: ALTER TABLE std ADD (Address CHAR (10));

- **ALTER TABLE...MODIFY...:** This is used to change the width as well as data type of fields of existing relations.
  SYNTAX: ALTER TABLE table_name MODIFY column_name column_type;
  EXAMPLE: ALTER TABLE sailors MODIFY age int(10);

- **ALTER TABLE - DROP:** This is used to remove any field of existing relations.
  SYNTAX*:* ALTER TABLE relation_name DROP COLUMN field_name;
  EXAMPLE: ALTER TABLE student DROP column sname;

- **ALTER TABLE- RENAME:** This is used to change the name of fields in existing relations.
  SYNTAX: ALTER TABLE relation_name RENAME COLUMN (OLD field_name) to (NEW field_name);
  EXAMPLE: ALTER TABLE student RENAME COLUMN sname to stu_name;

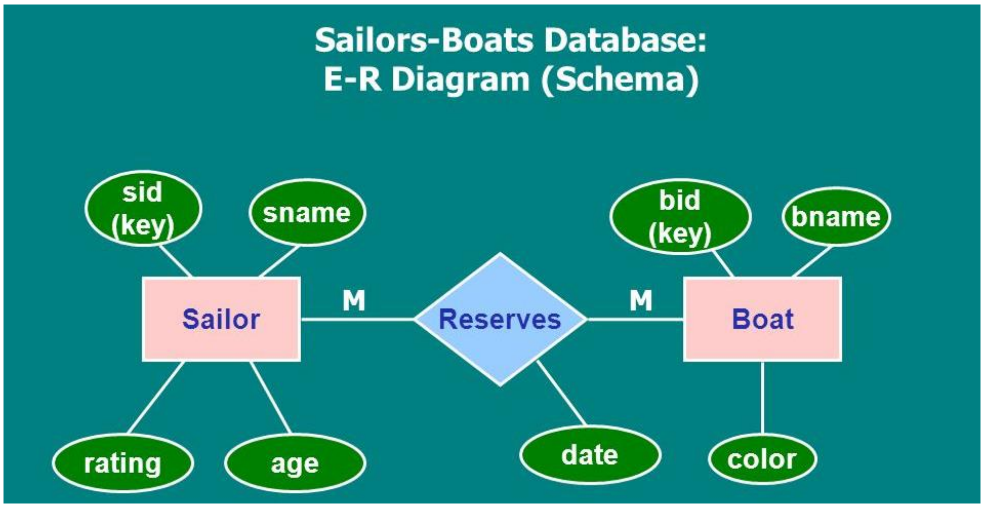
3.DROP TABLE: This is used to delete the structure of a relation. It permanently deletes the records in the table.
SYNTAX: DROP TABLE relation_name;
EXAMPLE: DROP TABLE std***;***

4.RENAME: It is used to modify the name of the existing database object.
SYNTAX: RENAME table old_relation_name TO new_relation_name;
EXAMPLE: RENAME table std TO std1***;***

5.TRUNCATE TABLE: This is used to truncate thetable.
SYNTAX: TRUNCATE TABLErelation_name;
EXAMPLE: TRUNCATE TABLE std;

EXERCISES:
Refer below ER- Model of Boat Club Database to translate into Relational Model using DDL Commands.

*Note: use **DESC** command to view the schema after every commands execution.*
SYNTAX:** desc tablename/relationname**;



Attributes list

Sailors (Attributes)	**Description**
SID	A sailor—each sailor is assigned a unique ID
name	The sailor's name
rating	The sailor's rating, ranging from 1 (low) to 10 (high)
age	The sailor's age
Boat (Attributes)	**Description**
BID	A boat ID—each boat is assigned a unique ID (painted on the bow)
name	The name of the boat (also painted on the bow)
color	The color of the boat
Reservation (Attributes)	**Description**
SID	Sailor ID
BID	Boat ID
date	The date and time of the reservation
duration	The expected duration of the reservation in minutes

TOOLS TO USE:

1. Online Compilers: **https://paiza.io/en/languages/mysql**
2. My sql desktop version***:*** **https://www.professionalcipher.com/p/downloads_22.html**

EXPERIMENT-2: Use DML commands to insert and manipulate the data in **Boat Club** database.

DESCRIPTION:

DATA MANIPULATION LANGUAGE (DML): The Data Manipulation Language (DML) is used to retrieve, insert, and modify database information. These commands will be used by all database users during the routine operation of the database.

1. **INSERT**
2. **UPDATE**
3. **DELETE**
4. **SELECT**

COMMANDS:

1. INSERT: This is used to add records into a relation. These are three type of INSERT INTO queries which areas.

a) Inserting a single record

SYNTAX: INSERT INTO < relation/table name> (field_1,field_2……field_n)VALUES (data_1,data_2,……. data_n);

EXAMPLE: INSERT INTO student(sno, sname, class, address) VALUES (1,'Ravi','M. Tech','Palakol');

b) Inserting a single record

SYNTAX: INSERT INTO < relation/tablename>VALUES (data_1,data_2,.data_n);

EXAMPLE: INSERT INTO student VALUES (1,'Ravi','M. Tech','Palakol');

c) Inserting all records from another relation

SYNTAX: INSERT INTO relation_name_1 SELECT Field_1,field_2,field_n FROM relation_name_2 WHERE field_x=data;

EXAMPLE: INSERT INTO std SELECT sno,sname FROM student WHERE name = 'Ramu';

d) Inserting multiple records

SYNTAX: INSERT INTO relation_namefield_1,field_2,. field_n) VALUES (&data_1,&data_2,……. &data_n);

EXAMPLE: SQL>INSERT INTO student (sno, sname, class,address) VALUES (&sno,'&sname','&class','&address');

```
Enter value for sno: 101
Enter value for name: Ravi
Enter value for class: M.Tech
Enter value for name: Palakol
```

2. UPDATE: This is used to update the content of a record in a relation.

SYNTAX: UPDATE relation name SET Field_name1=data,field_name2=data, WHERE field_name=data;

EXAMPLE: UPDATE student SET sname = 'kumar' WHEREsno=1;

3. DELETE: This is used to delete all the records of a relation but it will retain the structure of that relation.

a) DELETE-FROM: This is used to delete all the records of relation.

SYNTAX*:* SQL>DELETE FROM relation_name;
EXAMPLE*:* SQL>DELETE FROM std;

b) DELETE -FROM-WHERE: This is used to delete a selected record from a relation.
SYNTAX*:* SQL>DELETE FROM relation_name WHERE condition;
EXAMPLE: SQL>DELETE FROM student WHERE sno = 2;

DIFFERENCE BETWEEN TRUNCATE & DELETE: -

- ✓ By using truncate command data will be removed permanently & will not get back where as by using delete command data will be removed temporally & get back by using roll back command.
- ✓ By using delete command data will be removed based on the condition where as by using truncate command there is no condition.
- ✓ Truncate is a DDL command & delete is a DML command.

4. SELECT: It retrieves the records from the table

a. **SELECT FROM**: To display fields for all records.
SYNTAX: SELECT * FROM relation_name;
EXAMPLE*:* select * from dept;

```
DEPTNO      DNAME           LOC
 --------   -----------     ----------
10          ACCOUNTING      NEWYORK
20          RESEARCH        DALLAS
30          SALES           CHICAGO
40          OPERATIONS      BOSTON
```

b. **SELECT - FROM -WHERE:** This query is used to display a selected set of fields for a selected set of records of a relation.
SYNTAX: SELECT a set of fields FROM relation_name WHERE condition;
EXAMPLE: select * FROM dept WHERE deptno<=20;

```
DEPTNO      DNAME           LOC
------      -----------     ------------
10         ACCOUNTING       NEWYORK
20           RESEARCH       DALLAS
```

EXERCISE:

1. Enter the sample data into the "**Sailor, Boats & Reserve"** table.
2. Retrieve the "**Sailor, Boats & Reserve"** to show sample data.
3. Update Brutus rating to 2 and display the table with changes.
4. Display sailors whose rating is 3.
5. Display RID, SID and date from reservation table.
6. Delete sailor whose id is 1.

SAMPLE DATA for “BOAT CLUD “ DATABASE:

SID	name	rating	age
1	Dustin	7	45.0
2	Brutus	1	33.0
3	Laura	8	55.5
4	Andy	8	25.5
5	Rusty	10	35.5
6	Horatio	7	35.0
7	Zelda	10	16.0
8	Horatio	9	35.0
9	Amy	3	25.5
10	Bob	3	63.5

BID	name	color
1	Interlake	blue
2	Interlake	red
3	Clipper	green
4	Marine	red

RID	SID	BID	date	duration
1	1	1	10/10/07	60
2	1	2	10/10/07	60
3	1	3	10/08/07	60
4	1	4	10/07/07	90
5	3	2	11/10/07	90
6	3	3	11/06/07	60
7	3	4	11/12/07	90
8	6	1	9/05/07	60
9	6	2	9/08/07	60
10	7	3	9/08/07	60

EXPERIMENT-3: Practicing DCL commands of SQL on Relational Database Schemas.

DESCRIPTION:
DCL (Data Control Language) includes commands like GRANT and REVOKE, which are useful to give "rights & permissions." Other permission controls parameters of the database system.

COMMANDS:
Commands that come under DCL:

- Grant
- Revoke

GRANT: This command is use to give user access privileges to a database.
SYNTAX: GRANT SELECT, UPDATE ON MY_TABLE TO SOME_USER, ANOTHER_USER;
EXAMPLE: GRANT SELECT ON Users TO 'Tom'@'localhost;

REVOKE: It is useful to back permissions from the user.
SYNTAX: REVOKE privilege_nameON object_nameFROM {user_name |PUBLIC |role_name}
EXAMPLE: REVOKE SELECT, UPDATE ON student FROM BCA, MCA;

EXERCISE:

1. Open MySQL Workbench.
2. The Users and Privileges tab of MySQL Workbench allows you to administer users and their respective privileges for the MySQL instance.
3. To display the Users and Privileges tab, click `Users and Privileges` on the `Management` tab:
4. The Users and Privileges screen displays a list of all users, along with relevant details for each account. Here, an administrator can create new user accounts, set their password, authentication type, their administrative roles, and schema privileges, if any.
5. Clicking on the various buttons will display their respective tabs.
6. The Account Limit Tab shows
   i. The number of queries that an account can issue per hour
   ii. The number of updates that an account can issue per hour
   iii. The number of times an account can connect to the server per hour
   iv. The number of simultaneous connections to the server by an account

   NOTE: Any statement that a client can issue counts against the query limit, unless its results are served from the query cache. Only statements that modify databases or tables count against the update limit.

7. The `Administrative Roles` tab allows you to apply a role (or roles) for each user. DBA is the highest role — a DBA can perform all tasks.
8. Other roles are more specific to a particular task (or set of tasks) that a user might be allowed to perform. You can also set the Global Privileges in this screen.
9. When you create a new user, you might only allow a small number of options for that user. For example, you might restrict one user to only being able to perform `SELECT`, `INSERT`, and `UPDATE` statements, while another might be able to perform `DELETE` and even `DROP`

statements in addition to those.

10. It's good practice to limit user privileges and roles to the bare minimum that the user needs. Never grant a user account more access than it needs. Privileges can always be added later if required.
11. The `Schema Privileges` tab is used for adding schema privileges to a user account. This enables that user to perform the given actions based on the rights given to them in this screen.

With Queries:

1. Display databases
   show databases;
2. Enable the user: Root with Grant access.
3. Show users
   select user from mysql.user;
4. Create users
   create user auth@localhost identified by 'sql';
5. Show users (This should show new user here)
6. *Show grants;*
7. Grant all privileges to the created user
   grant all privileges on boatclubdb. to auth@localhost;*
8. View grants once again to observe the changes
   show grants for auth@localhost;
9. Revoke all the grants from the user.
   REVOKE ALL PRIVILEGES, GRANT OPTION FROM auth@localhost;
10. Observe the changes by viewing grants. *show grants for auth@localhost;*
11. Drop the user and view if he is deleted or not.
    drop user auth@localhost;

EXPERIMENT-4: Develop Relational Database Schemas by specifying different types of Constraints.

DESCRIPTION:

Constraints in SQL: Rules which are enforced on data being entered and prevents the user from entering invalid data into tables are called constraints.

TYPES OF CONSTRAINTS: There are six types of constraints.

1. NOT NULL
2. UNIQUE
3. PRIMARY KEY
4. FOREIGN KEY
5. CHECK
6. DEFAULT

1. **NOT NULL:** NOT NULL means that column never empty.

   EX:
   ```
   CREATE TABLE CUSTOMER (CNO INT NOTNULL,
                          CNAME VARCHAR(20) NOTNULL,
                          ADDRESS VARCHAR(20),
                          PHONENO INT(10));
   ```
2. **UNIQUE:** It is used to uniquely identify each value in a column.

   EX:
   ```
   CREATE TABLE CUSTOMER (CNO INT UNIQUE NOTNULL,
                          CNAME VARCHAR(20) NOTNULL,
                          ADDRESS VARCHAR(20),
                          PHONENO INT(10));
   ```
3. **PRIMARY KEY:** A primary key is one or more columns in a table used to uniquely identify each row in a table.

   EX:
   ```
   CREATE TABLE CUSTOMER (CNO INT(4),
                          CNAME VARCHAR(20),
                          ADDRESS VARCHAR(20)
                          PRIMARY KEY(CNO,CNAME));
   ```
4. **FOREIGN KEY:** A foreign key is an attribute in one relation which acts as a primary key in another relation of the same database. It is used to establish the relationship between two tables.

   EX:
   ```
   CREATE TABLE EMP (EMPNO   INT(4) PRIMARY KEY,
                     NAME VARCHAR2(20),
                     SAL FLOAT (7,2),
                     JDATE DATE,
                     DEPTNO INT(4) REFERENCES
                     DEPT(DEPTNO));
   CREATE TABLE DEPT (DEPTNO INT(4)PRIMARY KEY,
                      DNAME VARCHAR(20),
                      LOC VARCHAR(20));
   ```
5. **CHECK:** Many columns must have values that are within a certain range or that satisfy certain condition with a CHECK constraint .A condition can be specified by the value entered for the column.

   EX:
   ```
   CREATE TABLE CUSTOMER (CNO NUMBER (4),
                          CNAME VARCHAR (20),
                          ADDRESS VARCHAR (20) CHECK
   ```

```
                                                ADDRESS IN (‘GNT’,’TNL’,’VJA’),
                                                PHONE INT (12));
```

6. DEFAULT: While inserting a row into a table without having values for every column SQL must insert a default value to fill in the executed columns. The most common default value is NULL.

EX:
```
CREATE TABLE CUSTOMER (CNO INT(4),
                       CNAME VARCHAR(20),
                       ADDRESS VARCHAR(20),
                       PHONE   INT (12),
                      PINCODE  INT(6) DEFAULT
                       PINCODE IS NULL);
```

EXERCISE: Create a new database with name “Constraints” to show above all.

EXPERIMENT-5: Working with different types of operators (Arithmetic & Logical and Comparison) and String-matching Constraints on relational database.

DESCRIPTION:

ARITHMETIC OPERATORS: The usual arithmetic operators are available. The result is determined according to the following rules:

- In the case of -, +, and *, the result is calculated with BIGINT (64-bit) precision if both operands are integers.
- If both operands are integers and any of them are unsigned, the result is an unsigned integer
- If any of the operands of a +, -, /, *, % is a real or string value, the precision of the result is the precision of the operand with the maximum precision.

EXAMPLES:

1. **Addition:** mysql>SELECT 3+5;
2. **Subtraction:** mysql>SELECT3-5;
3. **Unary minus.** This operator changes the sign of the operand.
   mysql>SELECT-2;
4. **Multiplication:** mysql>SELECT3*5;
5. **Division:** mysql>SELECT3/5; mysql>SELECT 102/(1-1);\
6. **Integer division:** Discards from the division result any fractional part to the right of the decimal point.
   mysql>SELECT5DIV2,-5DIV2,5DIV-2,-5DIV-2;
7. **Modulo operation.** Returns the remainder of **N** divided by **M**.

COMPARISON OPERATORS: Comparison operators are used in the WHERE clause to determine which records to select. Here is a list of the comparison operators that you can use in MySQL:

Comparison Operator	**Description**
=	Equal
<=>	Equal (Safe to compare NULL values)
<>	Not Equal
!=	Not Equal
>	Greater Than
>=	Greater Than or Equal
<	Less Than
<=	Less Than or Equal

Let us consider below table as input table

contact_id	last_name	website1	website2
1	Johnson	techonthenet.com	<NULL>
2	Anderson	<NULL>	<NULL>
3	Smith	TBD	TDB
4	Jackson	checkyourmath.com	digminecraft.com

EQUAL OPERATOR:

SELECT * FROM contacts WHERE website1 = website2;

OUTPUT:

contact_id	last-named	website1	website2
3	Smith	TBD	TDB

SELECT * FROM contacts WHERE website1 <=>website2;

Because we used the <=> operator, we would get the following results:

contact_id	last_name	website1	website2
2	Anderson	<NULL>	<NULL>
3	Smith	TBD	TDB

Now our query returns all rows from the contacts table where website1 is equal to website2, including those records where website1 and website2 are NULL values.

INEQUALITY OPERATOR:

SELECT * FROM contacts WHERE last_name<> 'Johnson';

(OR)

SELECT * FROM contacts WHERE last_name != 'Johnson';

Both queries would return the same results.

GREATER THAN OPERATOR

SELECT * FROM contacts WHERE contact_id> 50;

The SELECT statement would return all rows from the *contacts* table where the *contact_id* is greater than 50. A *contact_id* equal to 50 would not be included in the result set.

GREATER THAN OR EQUAL OPERATOR

SELECT * FROM contacts WHERE contact_id>= 50;

The SELECT statement would return all rows from the *contacts* table where the *contact_id* is greater than or equal to 50.In this case, *contact_id* equal to 50 would be included in the result set.

LESS THAN OPERATOR

SELECT * FROM inventory WHERE product_id< 300;

The SELECT statement would return all rows from the *inventory* table where the *product_id* is less than 300. A *product_id* equal to 300 would not be included in the result set.

LESS THAN OR EQUAL OPERATOR

```
SELECT * FROM inventory WHERE product_id<= 300;
```

The SELECT statement would return all rows from the *inventory* table where the *product_id* is less than or equal to 300. In this case, *product_id* equal to 300 would be included in the result set.

LOGICAL OPERATORS: Logical operators enable us to use more than one condition in WHERE clause.

Operator	**Meaning**
NOT	Negate a condition to its opposite. NOT TRUE => FALSE NOT FALSE => TRUE NOT UNKNOWN => UNKNOWN
AND	Returns TRUE if both conditions are TRUE.
&&	The same as AND operator. Returns TRUE if both conditions are TRUE.
OR	Returns TRUE if either condition is TRUE.
||	The same as OR operator. Returns TRUE if either condition is TRUE.
XOR	Equivalent to AND NOT.

AND (&&) operator requires both conditions to be TRUE.

Find products in Beverages category and their number of units in stock is less than reorder level.

```
-- Query 1: use AND operator
SELECT ProductID, ProductName FROM products
WHERE UnitsInStock < ReorderLevel  AND CategoryID=1;
                    (OR)
-- Query 2: use && operator
SELECT ProductID, ProductName  FROM products
WHERE UnitsInStock < ReorderLevel   && CategoryID=1;
```

OR (||) operator requires either condition to be TRUE.

Find products in Beverages or Seafood category.

```
-- Query 1: use OR operator
SELECT ProductID, ProductName FROM products
WHERE CategoryID=1 OR CategoryID=8;
                  (OR)
-- Query 2: use || operator
SELECT ProductID, ProductName FROM products
WHERE CategoryID=1 || CategoryID=8;
```

The following table shows the result of comparing two expressions with OR operator:

	TRUE	**FALSE**	**UNKNOWN**
TRUE	TRUE	TRUE	TRUE
FALSE	TRUE	FALSE	UNKNOWN

UNKNOWN	TRUE	UNKNOWN	UNKNOWN

NOT IN:
List products which reorder level is not in the list of 0, 10, 15, 20, 25, and 30.

```
SELECT ProductID, ProductName, ReorderLevel FROM products
WHERE ReorderLevel NOT IN (0,10,15,20,25,30);
```

XOR: XOR is equivalent to AND NOT.
Get all employees who report to Vice President and the result should exclude Nancy if it exists in FirstName column.

```
SELECT EmployeeID, FirstName, LastName, ReportsTo
FROM employees
WHERE ReportsTo=2 XOR FirstName='Nancy';
                    (OR)
-- Query 2 returns the same result as Query 1.
SELECT EmployeeID, FirstName, LastName, ReportsTo
FROM employees
WHERE ReportsTo=2 AND NOT FirstName='Nancy';
```

STRING COMPARISON OPERATORS:

Name	**Description**
LIKE	Simple pattern matching
NOT LIKE	Negation of simple pattern matching
STRCMP()	Compare two strings

STRCMP (EXPR1, EXPR2): using the STRCMP () function, two strings are compared, and since it is found that the strings are same, it returns 0.

```
mysql> SELECT STRCMP('mytesttext', 'mytesttext');
OUTPUT: 0
```

LIKE: The LIKE operator is used in a WHERE clause to search for a specified pattern in a column. There are two wildcards often used in conjunction with the LIKE operator:
% - The percent sign represents zero, one, or multiple characters
_ - The underscore represents a single character
The percent sign and the underscore can also be used in combinations!

SYNTAX:

```
SELECT column1, column2, ...
FROM table_name
WHERE column LIKE pattern;
```

LIKE Operator	**Description**
WHERE CustomerName LIKE 'a%'	Finds any values that start with "a"

WHERE CustomerName LIKE '%a'	Finds any values that end with "a"
WHERE CustomerName LIKE '%or%'	Finds any values that have "or" in any position
WHERE CustomerName LIKE '_r%'	Finds any values that have "r" in the second position
WHERE CustomerName LIKE 'a_%'	Finds any values that start with "a" and are at least 2 characters in length
WHERE CustomerName LIKE 'a__%'	Finds any values that start with "a" and are at least 3 characters in length
WHERE ContactName LIKE 'a%o'	Finds any values that start with "a" and ends with "o"

EXERCISE:

Consider the below table to perform **Arithmetic operations**

employee_id	employee_name	salary
1	Ajay	25000
2	Greshma	55000
3	Pooja	52000
4	Bhargav	12312

1. Add 100 to each Employee's salary i.e, addition operation on single column .
2. Add 2 columns with each other i.e, each employee's employee_id is added with its salary.
3. Subtract 100 from each Employee's salary i.e, subtraction operation on single column.
4. Subtract 2 columns with each other i.e, each employee's employee_id is subtracted from its salary.
5. Multiply 2 columns with each other i.e, each employee's employee_id is multiplied with its salary.
6. Perform modulus on a single column with a value 100.

Consider the below table to perform **Comparison operators:**

Supplier-id	Supplier-name	city	state
100	Microsoft	Redmond	Washington
200	Google	Mountain View	California
300	Oracle	Redwood City	California
400	Kimberly-Clark	Irving	Texas
500	Tyson Foods	Springdale	Arkansas
600	SC Johnson	Racine	Wisconsin
600	Dole Food Company	Westlake Village	California

Supplier-id	**Supplier-name**	**city**	**state**
800	Flowers Foods	Thomasville	Georgia
900	Electronic Arts	Redwood City	California

1. Use SELECT statements to return all rows from the suppliers table where the supplier_name is not equal to Microsoft.
2. Use SELECT statements to return all rows from the suppliers table where the supplier-name is equal to Microsoft.
3. Use SELECT statement to return all rows from the supplier table where the supplier_id is greater than 600.
4. Return all rows from the supply table where supplier-id is greater than equal to 600.
5. Return all rows from the supply table where supplier-id less than and equal to 300.
6. Retrieve the rows by applying following conditions: supplier-id must be 600 and state must be California.
7. Retrieve the rows by applying following conditions: supplier-id must be 600 or state must be California.
8. Retrieve the rows where city name does not start with R.

EXPERIMENT-6: Expressing queries using Aggregate Functions of SQL on Relational Database schemas.

DESCRIPTION:
In addition to simply retrieving data, we often want to perform some computation or summarization. SQL supports five aggregate operations, which can be applied on any column, to perform required computations and summarization.

1. COUNT (A): - The number of values in the A column.
Or
COUNT (DISTINCT A): The number of unique values in the A column.
SYNTAX: select count (column name) from table_name;

2. SUM (A) :- The sum of all values in the A column.
Or SUM (DISTINCT A): The sum of all unique values in the A column.
SYNTAX: Select sum(column name) from table_name;

3. AVG (A) :- The average of all values in the A column.
Or AVG (DISTINCT A): The average of all unique values in the A column.
SYNTAX: Select AVG (column name) from table_name;

4. MAX (A) :- The maximum value in the A column.
SYNTAX: Select MAX (column name) from table_name;

5. MIN (A) :- The minimum value in the A column.
SYNTAX: Select MIN(column name) from table_name;

Note that it does not make sense to specify DISTINCT in conjunction with MIN or MAX (Although SQL does not preclude this).

EXERCISE:
1. Display the numbers of boats booked in Reserves table.
2. Find age of Oldest Sailor.
3. Display the number of Boats in Boats table.
4. Display average of distinct age of Sailors (Duplicate ages are eliminated).
5. Display sum of rating from Sailors
6. Display average age of Sailors.
7. Display sum of distinct age of Sailors (Eliminate Duplicate ages).
8. Find age of Youngest Sailor.

EXPERIMENT-7: Specify queries on Relational Database schemas using GROUP BY and HAVING and ORDER BY Clauses of SQL

DESCRIPTION:

ORDER BY Clause: -
The ORDER BY keyword is used to sort the result-set by a specified column. The ORDER BY keyword sorts the records in ascending order by default (we can even use ASC keyword). If we want to sort the records in a descending order, we can use the DESC keyword. The gene

SYNTAX:

```
SELECT ATT_LIST FROM TABLE_LIST ORDER BY ATT_NAMES [ASC | DESC];
```

Examples:

1) Display all the sailors according to their ratings (topper first).

```
SQL> SELECT * FROM SAILORS ORDER BY RATING DESC;
```

2) Displays all the sailors according to rating, if rating is same then sort according to age.

```
SQL> SELECT * FROM SAILORS ORDER BY RATING, AGE;
```

GROUP BY: - Group by is used to make each a number of groups of rows in a relation, where the number of groups depends on the relation instances.
The general SYNTAX is

```
SELECT [DISTINCT] ATT_LIST
FROM TABLE_LIST
WHERE CONDITION
GROUP BY GROUPING_LIST;
```

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

HAVING: - The extension of GROUP BY is HAVING clause which can be used to specify the qualification over group.
The general SYNTAX is

```
SELECT [DISTINCT] ATT_LIST
FROM TABLE_LIST
WHERE CONDITION
GROUP BY GROUPING_LIST
HAVING GROUP_CONDITION.
```

EXERCISE:

Write the query

1) To display names of sailors according to alphabetical order.
2) Displays all the sailors according to rating (Topper First), if rating is same then sort according to age (Older First).
3) Displays all the sailors according to rating (Topper First), if rating is same then sort according to age (Younger First).
4) Find the minimum age of the youngest sailor for each rating level.
5) Find the age of youngest sailor with age >= 18 for each rating with at least 2 such sailors.
6) Find age of the youngest sailor with age ≥ 18, for each rating level with at least 2 sailors between 18 and 60.

EXPERIMENT-8: Working on Boat Club database with Nested Queries.

DESCRIPTION:

NESTED QUERIES: - One of the most powerful features of SQL is nested queries. A nested query is a query that has another query embedded within it; the embedded query is called a sub query.

IN Operator: - The IN operator allows us to test whether a value is in a given set of elements; an SQL query is used to generate the set to be tested.
SYNTAX:

```
SELECT column_name(s)
FROM table_name
WHERE column_name IN (value1, value2, ...);

Or

SELECT column_name(s)
FROM table_name
WHERE column_name IN (SELECT STATEMENT);
```

NOT IN Operator: - The NOT IN is used in a opposite manner to IN.
SYNTAX:

```
SELECT column_name(s)
FROM table_name
WHERE column_name NOT IN (value1, value2, ...);

Or

SELECT column_name(s)
FROM table_name
WHERE column_name NOT IN (SELECT STATEMENT);
```

EXISTS OPERATOR: - This is a Correlated Nested Queries operator. The EXISTS operator isanother set comparison operator, such as IN. It allows us to test whether a set is nonempty, animplicit comparison with the empty set.
SYNTAX:

```
SELECT column_name(s)
FROM table_name
WHERE EXISTS
(SELECT column_name FROM table_name WHERE condition);
```

NOT EXISTS OPERATOR: - The NOT EXISTS is used in a opposite manner to EXISTS.
SYNTAX:

```
SELECT [Column Names]
FROM [Source]
WHERE NOT EXISTS (Write Subquery to Check)
```

SET-COMPARISON OPERATORS: - SQL also supports **op ANY** and **op ALL**, where **op** is one of the arithmetic comparison operators {<, <=, =, <>, >=, >}.

ANY OPERATOR: - It is a comparison operator. It is used to compare a value with any of element in a given set.

SYNTAX:

```
SELECT column_name(s)
FROM table_name
WHERE column_name operator ANY
(SELECT column_name FROM table_name WHERE condition);
```

Note that IN and NOT IN are equivalent to = ANY and <> ALL, respectively.

ALL OPERATOR: - It is a comparison operator. It is used to compare a value with all theelements in a given set.

SYNTAX:

```
SELECT column_name(s)
FROM table_name
WHERE column_name operator ALL
(SELECT column_name FROM table_name WHERE condition);
```

EXERCISE: -

1. Find the names of sailors who have reserved boat 4.
2. Find the names of sailors who have reserved all boats.
3. Find sailors whose rating is better than Dustin.
4. Find the sailor's with the highest rating using ALL

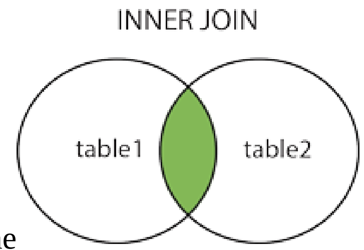
EXPERIMENT- 9 Practicing JOINS on Sailors Database.

DESCRIPTION:

A JOIN clause is used to combine rows from two or more tables, based on a related column between them.

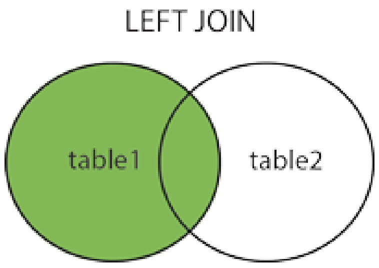
INNER JOIN: Returns records that have matching values in both tables
SYNTAX:

```
SELECT column_name(s)
FROM table1
INNER JOIN table2
ON table1.column_name = table2.column_name;
```



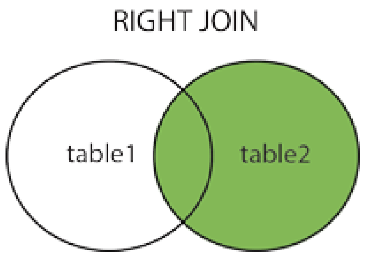
LEFT OUTER JOIN: Returns all records from the left table, and the matched records from the right table.
In some databases LEFT JOIN is called LEFT OUTER JOIN**.**
SYNTAX:

```
SELECT column_name(s)
FROM table1
LEFT JOIN table2
ON table1.column_name = table2.column_name;
```



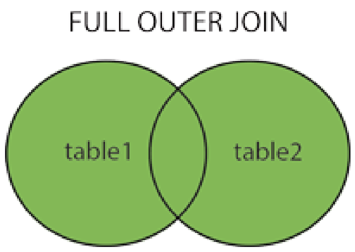
RIGHT OUTER JOIN: Returns all records from the right table, and the matched records from the left table.
SYNTAX:

```
SELECT column_name(s)
FROM table1
RIGHT JOIN table2
ON table1.column_name = table2.column_name;
```



FULL OUTER JOIN: Returns all records when there is a match in either left or right table
FULL OUTER JOIN and FULL JOIN are the same.
SYNTAX:

```
SELECT column_name(s)
FROM table1
FULL OUTER JOIN table2
ON table1.column_name = table2.column_name
WHERE condition;
```



SELF JOIN: A self-JOIN is a regular join, but the table is joined with itself.
SYNTAX:

```
SELECT column_name(s)
FROM table1 T1, table1 T2
WHERE condition;
```

EQUI JOIN: SQL EQUI JOIN performs a JOIN against equality or matching column(s) values of the associated tables. An equal sign (=) is used as comparison operator in the where clause to refer equality.

SYNTAX:

```
SELECT column_list
FROM table1, table2....
WHERE table1.column_name = table2.column_name;
```

(OR)

```
SELECT *
FROM table1
JOIN table2  [ON (join_condition)]
```

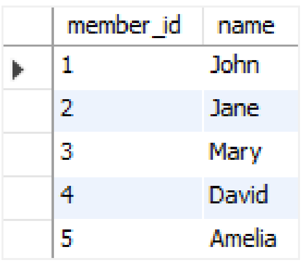
NON-EQUI JOIN: NON EQUI JOIN performs a JOIN using comparison operator other than equal (=) sign like >, <, >=, <= with conditions.

SYNTAX:

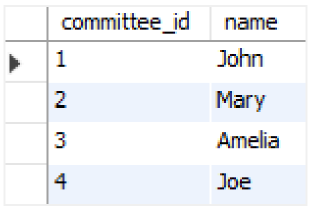
```
SELECT *
FROM table_name1, table_name2
WHERE table_name1.column [> | < |  >= | <= ] table_name2.column;
```

<u>EXERCISE:</u>

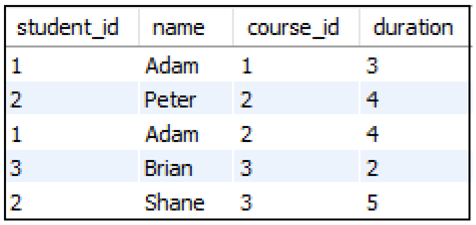
Create tables for Member (member_id, name), Committee (Committee_id, name), student(student_id, name, course_id, duration), emp(ID,NAME, CLASS,CITY), record(ID, CLASS,CITY) and insert the following data.



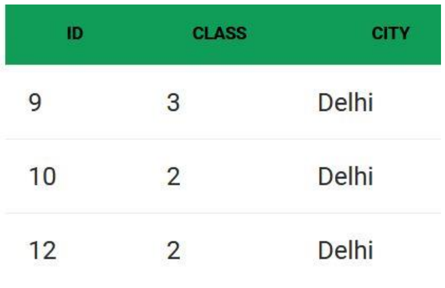
member_id	name
1	John
2	Jane
3	Mary
4	David
5	Amelia



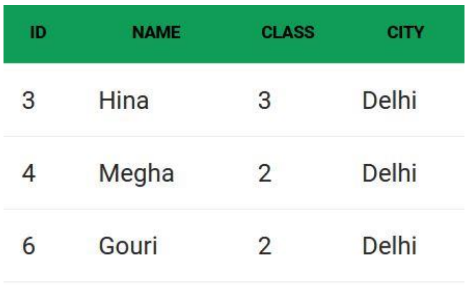
committee_id	name
1	John
2	Mary
3	Amelia
4	Joe



student_id	name	course_id	duration
1	Adam	1	3
2	Peter	2	4
1	Adam	2	4
3	Brian	3	2
2	Shane	3	5



ID	CLASS	CITY
9	3	Delhi
10	2	Delhi
12	2	Delhi



ID	NAME	CLASS	CITY
3	Hina	3	Delhi
4	Megha	2	Delhi
6	Gouri	2	Delhi

1. Finds members who are also the committee members.
2. Use the left join & Right join to view the members with the committees' table.
3. Find the committee members who are not in the members' table. (Hint Right Join)
4. Return the student(student_id,name) from the table where student_id is equal and course_id is not equal. (Hint Self Join)
5. Use equi join to display common members from both member and committee table.
6. Use non- equi join to display the rows with the condition emp id is less than record id.

EXPERIMENT-10: Creation and dropping of VIEWS.

DESCRIPTION:

A view is a table whose rows are not explicitly stored in the database but are computed as needed from a view definition.
The views are created using **CREATE VIEW** command.
SYNTAX:

```
CREATE VIEW view_name AS
SELECT column1, column2, ...
FROM table_name
WHERE condition;
```

To drop a view: `DROP VIEW view_name;`

EXAMPLE: -

1. Create a view for Expert Sailors (A sailor is a Expert Sailor if his rating is more than 7).

NOTE:
If we decide that we no longer need a view and want to destroy it (i.e. removing the definition of view) we can drop the view.
A view can be dropped using the **DROP VIEW** command.

2. Retrieve the values from Expert sailors.
3. Drop the Expert Sailor view.