KG REDDY College of Engineering & Technology
AN AUTONOMOUS INSTITUTION

DEPARTMENT OF COMPUTER SCIENCE ENGINEERING

DATABASE MANAGEMENT SYSTEMS LABORATORY

Subject Code : KGR23ACD222

Regulation : KGR23

Academic Year : 2025-2026

II B. TECH I SEMESTER

COMPUTER SCIENCE AND ENGINEERING

KG REDDY COLLEGE OF ENGINEERING AND TECHNOLOGY

Affiliated o JNTUH, Chilkur,(V), Moinabad(M) R. R Dist, TS-501504

KG REDDY College of Engineering & Technology
AN AUTONOMOUS INSTITUTION

VISION AND MISSION OF THE INSTITUTION

VISION:

To become an institution which is internationally recognized for its holistic approach to engineering, innovative teaching and learning culture, research and entrepreneurial ecosystem, and sustainable social impact in the community.

MISSION:

• To offer undergraduate and post-graduate programs which are supported through industry relevant curriculum and innovative teaching and learning processes that would help students succeed in their professional careers.
• To provide faculty and students with an ecosystem that fosters innovation, research, entrepreneurship, and international exposure through strategic partnerships with governmentorganizations and collaboration with industries.
• To provide holistic learning environment to students which will contribute to their personal and professional growth and enable them to become leaders in their respective fields.
• To contribute to the development of the region by using our technological expertise to work with nearby communities and support them in their social and economic development.

KG REDDY College of Engineering & Technology
AN AUTONOMOUS INSTITUTION

DEPARTMENT OF COMPUTER SCIENCE AND ENGINEERING

VISION:

To be recognized as a department of excellence by stimulating a learning environment in which students and faculty will thrive and grow to achieve their professional, institutional and societal goals.

MISSION:

• To provide high quality technical education to students that will enable life-long learning and build expertise in advanced technologies in Computer Science and Engineering.
• To promote research and development by providing opportunities to solve complex engineering problems in collaboration with industry and government agencies.
• To encourage professional development of students that will inculcate ethical values and leadership skills through entrepreneurship while working with the community to address societalissues.

PROGRAM EDUCATIONAL OBJECTIVES

PEO 1: Graduates will provide solutions to difficult and challenging issues in their profession by applying computer science and engineering theory and principles.
PEO 2: Graduates have successful careers in computer science and engineering fields or will be able to successfully pursue advanced degrees.
PEO 3: Graduates will communicate effectively, work collaboratively and exhibit high levels of professionalism, moral and ethical responsibility.
PEO 4: Graduates will develop the ability to understand and analyze engineering issues in a broader perspective with ethical responsibility towards sustainable development.

KG REDDY College of Engineering & Technology
AN AUTONOMOUS INSTITUTION

DEPARTMENT OF COMPUTER SCIENCE AND ENGINEERING

PROGRAM OUTCOMES

PO I: Engineering knowledge: Apply the knowledge of mathematics, science, engineering fundamentals, and an engineering specialization to the solution of complex engineering problems.
PO II: 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.
PO III: 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.
PO IV: Conduct investigations of complex problems: Use research-based knowledge and research methods including design of experiments, analysis and interpretation of data, and synthesis of the information to provide valid conclusions.
PO V: 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.
PO VI: 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.
PO VII: 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.
PO VIII: Ethics: Apply ethical principles and commit to professional ethics and responsibilities and norms of the engineering practice.
PO IX: Individual and team work: Function effectively as an individual, and as a member or leader in diverse teams, and in multidisciplinary settings.
PO X: 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.
PO XI: 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.
PO XII: 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.

KG REDDY College of Engineering & Technology
AN AUTONOMOUS INSTITUTION

DEPARTMENT OF COMPUTER SCIENCE AND ENGINEERING

PROGRAM SPECIFIC OUTCOMES

PSO1: The Computer Science and Engineering graduates are able to analyze, design, develop, test and apply management principles, mathematical foundations in the development of intelligentsystems with computational solutions, make them to expert in designing the secure application and hardware prototype
PSO2: The graduating student will be analyze the contemporary research issues in different areasof computer science & engineering and explore research gaps, analyze and carry out research in the specialized/emerging areas.
PSO3: Develop their skills to solve problems in the broad area of programming concepts and appraise environmental and social issues with ethics and manage different projects in multi- disciplinary field to conducive in cultivating skills for successful career, entrepreneurship and higher studies.

KG REDDY College of Engineering & Technology
AN AUTONOMOUS INSTITUTION

DATABASE MANAGEMENT SYSTEMS LABORATORY

Course Code: KG23ACD222 **B. Tech. II Year I -Semester**

Prerequisites: Courses on "Data Structures" and "Database .

Course Objectives:

The objectives of this course for the student are to:

1. Understand the basics of ER data model.
2. Gain skills on database design and normalization.
3. Familiarize SQL basics for data definition and data manipulation.
4. Enhance the knowledge on Triggers and its usage.
5. Familiarize the construction of PL/SQL.

Course Outcomes: After completion of this course, the students will be able to

CO1: Analyze the requirements of a given database problem and design ER-Model for implementation of database. (K4)
CO2: Implement the basic knowledge of SQL queries and relational algebra. (K3)
CO3: Construct database models for different database applications. (K6)
CO4: Apply normalization techniques for refining of databases. (K3)
CO5: Develop solutions for database applications using triggers, procedures, and cursors. (K3)

LIST OF EXPERIMENTS:

1. Concept design with E-R Model
2. Relational Model
3. Normalization
4. Practicing DDL commands
5. Practicing DML commands
6. Querying (using ANY, ALL, IN, Exists, NOT EXISTS, UNION, INTERSECT, Constraints etc.)
7. Queries using Aggregate functions, GROUP BY, HAVING and Creation and dropping of Views.
8. Triggers (Creation of insert trigger, delete trigger, update trigger)
9. Procedures
10. Usage of Cursors
11. Construct PL/SQL block for the following.

a. To determine whether a number is palindrome
b. To determine whether a number is an Armstrong number
c. To find greatest of three numbers
d. To display Fibonacci series.

ADDITIONAL EXPERIMENTS:

12. Consider the following schema for Order Database:

a. SALESMAN (Salesman_id, Name, City, Commission)
b. CUS TOMER (Customer_id, Cust_Name, City, Grade, Salesman_id)
c. ORDERS (Ord_No, Purchase_Amt, Ord_Date, Customer_id, Salesman_id)

Write SQL queries to:

1. Count the Customers with grade above Bangalore's average.
2. Find the name and numbers of all salesmen who had more than one customer
3. List all salesman and Indicate those who have and don't have Customers in Their cities (Use UNION operation.)
4. Create a view that finds the salesman who has the customer with the highest order of a day

5. Demonstrate the DELETE operation by removing salesman with id 1000. All his orders must also be deleted.
6. Consider the schema for College Database:
STUDENT (USN, SName, Address, Phone, Gender)
SEMSEC (SSID, Sem, Sec)
CLASS (USN, SSID)
SUBJECT (Subcode, Title, Sem, Credits)
IAMARKS (USN, Subcode, SSID, Test1, Test2, Test3, FinalIA)

Write SQL queries to

1. List all the student section.
2. Compute the total number of male and female students in each semester and in each section.
3. Create a view of test1 marks of student USN "1B115CS101' in All subjects.
4. Calculate the FinalIA (average of best two test marks) and update the Corresponding table for all students.
5. Categorize students based on the following criterion: If FinalIA = 17 to 20 then CAT = Give these details only for 8th semester A, B, and C section students.

TEXT BOOKS:

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

REFERENCE BOOKS:

1. Database Systems design, Implementation, and Management, Peter Rob & Carlos Coronel 7th Edition. KGR23 B. Tech – CSE Syllabus KGRCET Hyderabad
2. Fundamentals of Database Systems, Elmasri Navrate, Pearson Education
3. Introduction to Database Systems, C.J. Date, Pearson Education
4. Oracle for Professionals, The X Team, S. Shah and V. Shah, SPD.
5. Database Systems Using Oracle: A Simplified guide to SQL and PL/SQL, Shah, PHI. 6. Fundamentals of Database Management Systems, M. L. Gillenson, Wiley Student Edition.

DEPARTMENT OF COMPUTER SCIENCE AND ENGINEERING

EXPERIMENT- 1

CONCEPT DESIGN WITH E-R MODEL

AIM: To Relate the entities appropriately. Apply cardinalities for each relationship. Identify strong and weak entities. Indicate the type of relationships (total/partial). Incorporate generalization, aggregation and specialization etc. wherever required.

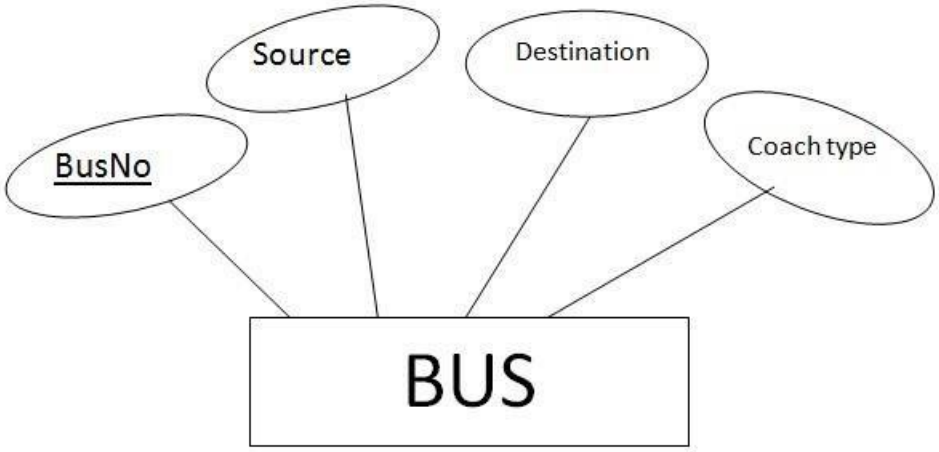
E-R Model

Bus

- BusNo
- Source
- Destination
- CoachType

SCHEMA

Bus: Bus (BusNo: String,Source: String, Destination: String, Coach Type: String)



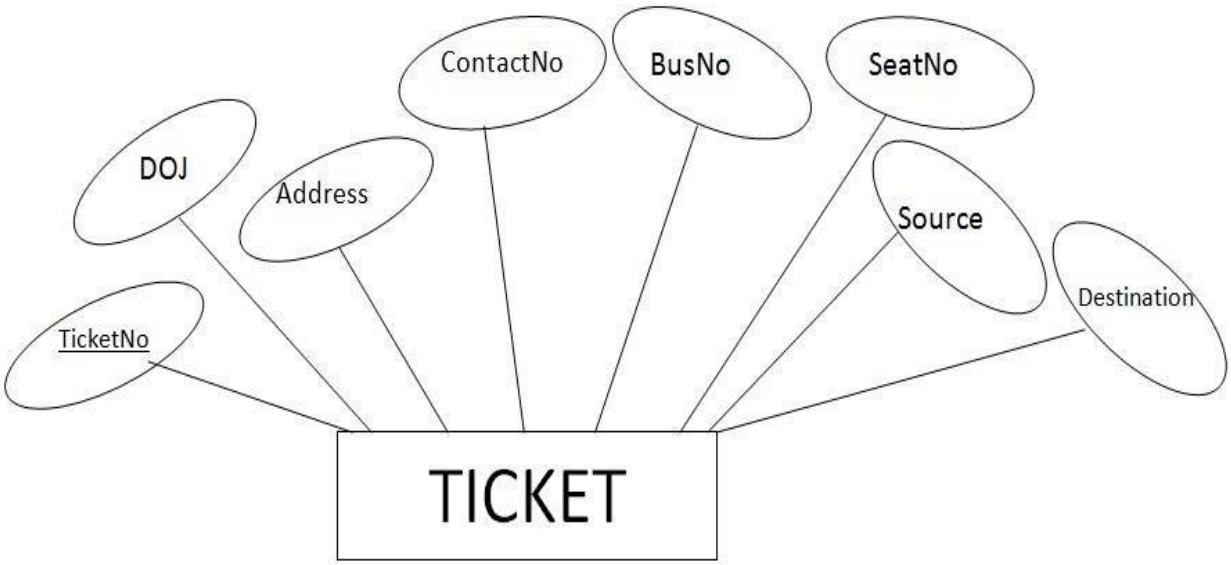
Ticket

- TicketNo

- DOJ
- Address
- ContactNo
- BusNo
- SeatNo
- Source
- Destination

SCHEMA

Ticket (TicketNo: string, DOJ: date, Address: string, ContactNo : string, BusNo:StringSeatNo : Integer, Source: String, Destination: String)

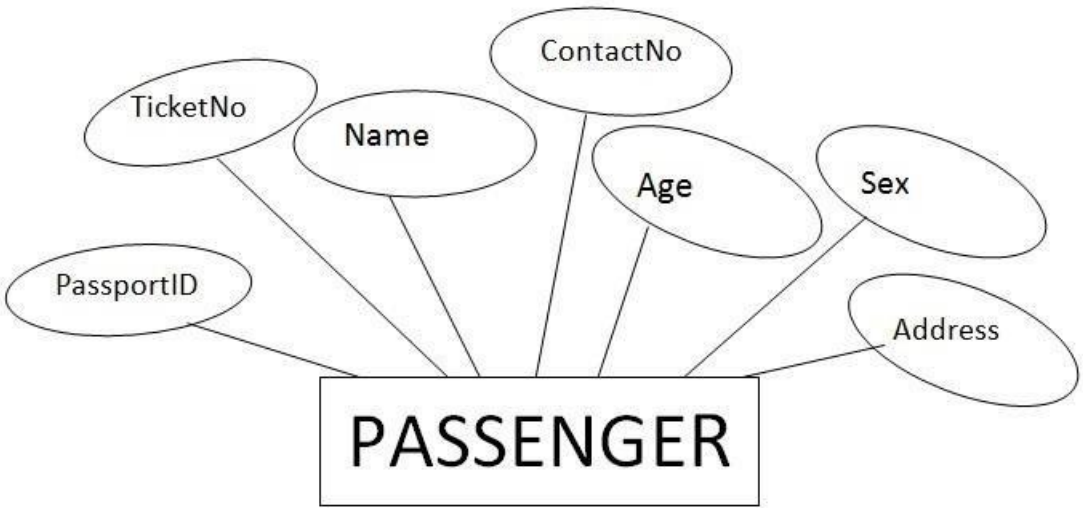


Passenger

- PassportID
- TicketNo
- Name
- ContactNo
- Age
- Sex
- Address

SCHEMA

Passenger (PassportID: String, TicketNo :string, Name: String, ContactNo: string, Age: integer, Sex: character, Address: String)

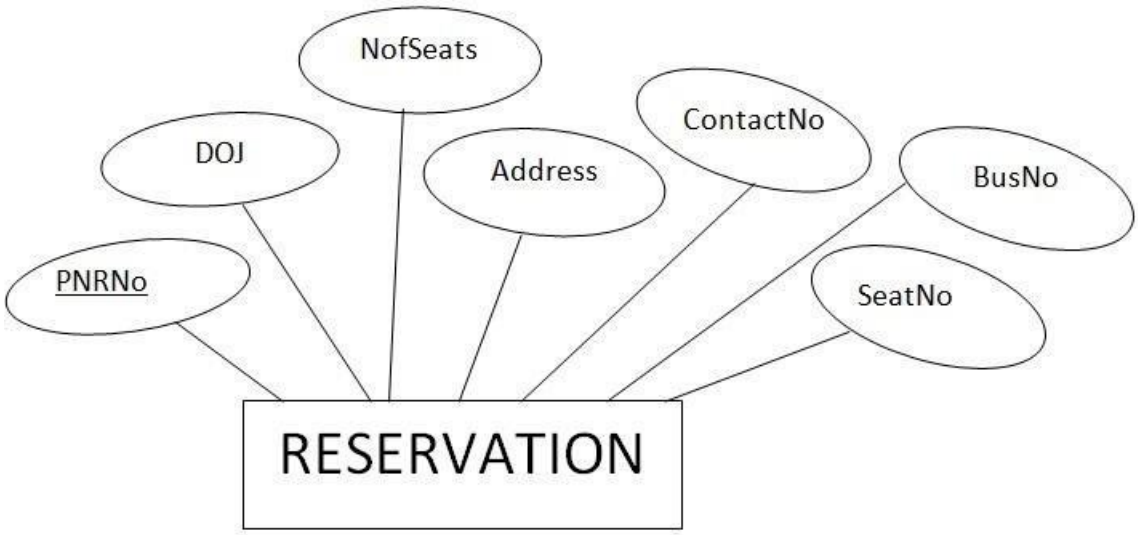


Reservation

- PNRNo
- DOJ
- No_of_seats
- Address
- ContactNo
- BusNo
- SeatNo

SCHEMA

Reservation(PNRNo: String, DOJ: Date, No_of_Seats: integer , Address: String ,ContactNo: String, , BusNo: String,SeatNo:Integer)

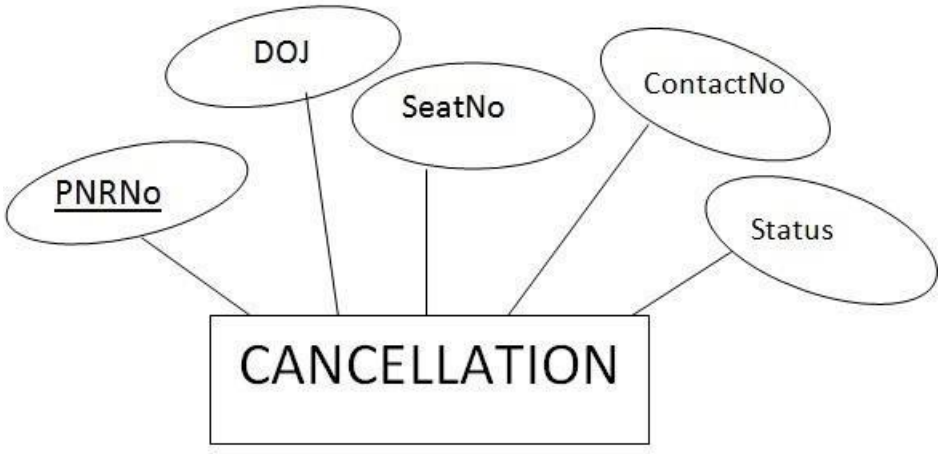


Cancellation

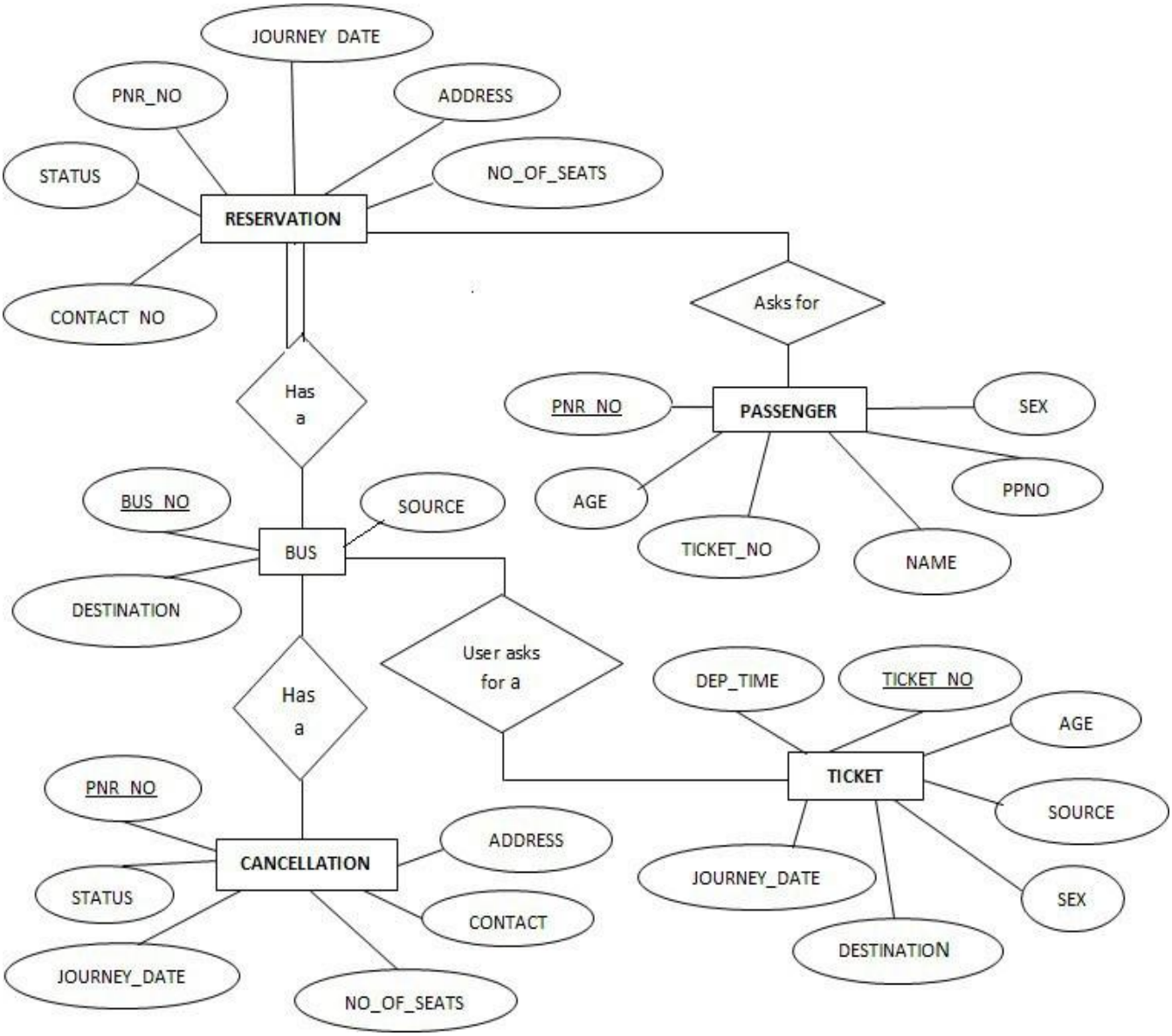
- PNRNo
- DOJ
- SeatNo
- ContactNo
- Status

SCHEMA

Cancellation (PNRNo: String, DOJ: Date, SeatNo: integer, ContactNo: String, Status: String)



CONCEPT DESIGN WITH E-R MODEL



EXPERIMENT 2
RELATIONAL MODEL

AIM: To Represent all the entities (Strong, Weak) in tabular fashion. Represent relationships in a tabular fashion.

1. **Bus:** Bus(BusNo: varchar(10), Source: varchar(20), Destination: varchar(20), CoachType:varchar(10))

mysql> create database csd;
Query OK, 1 row affected (0.00 sec)

mysql> use csd;
Database changed
mysql> create table Bus(BusNo varchar(10),Source varchar(20),Destination varchar(20),CoachType varchar(10),Primary key(BusNo);
mysql> desc Bus;

```
mysql> desc Bus;
+-------------+-------------+------+-----+---------+-------+
| Field       | Type        | Null | Key | Default | Extra |
+-------------+-------------+------+-----+---------+-------+
| BusNo       | varchar(10) | NO   | PRI | NULL    |       |
| Source      | varchar(20) | YES  |     | NULL    |       |
| Destination | varchar(20) | YES  |     | NULL    |       |
| CoachType   | varchar(10) | YES  |     | NULL    |       |
+-------------+-------------+------+-----+---------+-------+
4 rows in set (0.00 sec)
```

Ticket:

Ticket(TicketNo:varchar(20),DOJ:date,Address:varchar(20),ContactNo:varchar(10),BusNo:varchar(10),SeatNo:Integer,Source:varchar(10), Destination: varchar(10))

mysql> create table Ticket(TicketNo varchar(20),DOJ date,Address varchar(20),ContactNo varchar(10),BusNo varchar(10),SeatNoint,Source varchar(10),Destination varchar(10),Primary key(TicketNo,BusNO),foreign key(BusNo) references Bus(BusNo));
Query OK, 0 rows affected (0.01 sec)
mysql> desc Ticket;

```
mysql> desc Ticket;
+-------------+-------------+------+-----+---------+-------+
| Field       | Type        | Null | Key | Default | Extra |
+-------------+-------------+------+-----+---------+-------+
| TicketNo    | varchar(20) | NO   | PRI |         |       |
| DOJ         | date        | YES  |     | NULL    |       |
| Address     | varchar(20) | YES  |     | NULL    |       |
| ContactNo   | varchar(10) | YES  |     | NULL    |       |
| BusNo       | varchar(10) | NO   | PRI |         |       |
| SeatNo      | int(11)     | YES  |     | NULL    |       |
| Source      | varchar(10) | YES  |     | NULL    |       |
| Destination | varchar(10) | YES  |     | NULL    |       |
+-------------+-------------+------+-----+---------+-------+
8 rows in set (0.00 sec)
```

Passenger:

Passenger(PassportID:varchar(10), TicketNo:varchar(20),Name: varchar(15), ContactNo:varchar(15),Age: integer, Sex: character, Address: varchar(20));

mysql> create table Passenger(PassportId varchar(10),TicketNo varchar(20),Name varchar(15),ContactNo varchar(15),Age int,Sex char(2),Address varchar(20),primary key(PassportId,TicketNo),foreign key(TicketNo) references Ticket(TicketNo));

Query OK, 0 rows affected (0.00 sec)

mysql> desc Passenger;

```
mysql> desc Passenger;
+------------+-------------+------+-----+---------+-------+
| Field      | Type        | Null | Key | Default | Extra |
+------------+-------------+------+-----+---------+-------+
| PassportId | varchar(10) | NO   | PRI |         |       |
| TicketNo   | varchar(20) | NO   | PRI |         |       |
| Name       | varchar(15) | YES  |     | NULL    |       |
| ContactNo  | varchar(15) | YES  |     | NULL    |       |
| Age        | int(11)     | YES  |     | NULL    |       |
| Sex        | char(2)     | YES  |     | NULL    |       |
| Address    | varchar(20) | YES  |     | NULL    |       |
+------------+-------------+------+-----+---------+-------+
7 rows in set (0.00 sec)
```

Reservation:

Reservation(PNRNo: varchar(20), DOJ: Date, NoofSeats: integer, Address: varchar(20), ContactNo: varchar(20), BusNo: varchar(20),SeatNo:Integer)

mysql> Create table Reservation(PNRNo varchar(20),DOJ date,NoofSeatesinteger,Addressvarchar(20),ContactNo varchar(20),BusNo varchar(20),SeatNo int, primary key(PNRNo,BusNo),foreign key(BusNo) references Bus(BusNo));

Query OK, 0 rows affected (0.00 sec)

mysql> desc Reservation;

```
mysql> desc Reservation;
+------------+------------+------+-----+---------+-------+
| Field      | Type       | Null | Key | Default | Extra |
+------------+------------+------+-----+---------+-------+
| PNRNo      | varchar(20) | NO   | PRI |         |       |
| DOJ        | date       | YES  |     | NULL    |       |
| NoofSeates | int(11)    | YES  |     | NULL    |       |
| Address    | varchar(20) | YES  |     | NULL    |       |
| ContactNo  | varchar(20) | YES  |     | NULL    |       |
| BusNo      | varchar(20) | NO   | PRI |         |       |
| SeatNo     | int(11)    | YES  |     | NULL    |       |
+------------+------------+------+-----+---------+-------+
7 rows in set (0.00 sec)
```

Cancellation:

Cancellation (PNRNo: varchar(10),DOJ: Date, SeatNo: integer,ContactNo: varchar(15),Status: String:varchar(10))

mysql> create table Cancellation(PNRNo varchar(10),DOJ date,SeatNo integer, ContactNo varchar(15),Status varchar(10), primary key(PNRNo), foreign key(PNRNo) references Reservation(PNRNo));

Query OK, 0 rows affected (0.00 sec)

mysql> desc Cancellation;

```
mysql> desc Cancellation;
+-----------+------------+------+-----+---------+-------+
| Field     | Type       | Null | Key | Default | Extra |
+-----------+------------+------+-----+---------+-------+
| PNRNo     | varchar(10) | NO   | PRI |         |       |
| DOJ       | date       | YES  |     | NULL    |       |
| SeatNo    | int(11)    | YES  |     | NULL    |       |
| ContactNo | varchar(15) | YES  |     | NULL    |       |
| Status    | varchar(10) | YES  |     | NULL    |       |
+-----------+------------+------+-----+---------+-------+
5 rows in set (0.00 sec)
```

EXPERIMENT – 3
NORMALIZATION

AIM: Apply the database Normalization techniques for designing relational database tables to minimize duplication of information like 1NF, 2NF, 3NF, BCNF.

Normalization is a process of converting a relation to be standard form by decomposition a larger relation into smaller efficient relation that depicts a good database design.

- 1NF: A Relation scheme is said to be in 1NF if the attribute values in the relation are atomic.i.e., Mutli –valued attributes are notpermitted.
- 2NF: A Relation scheme is said to be in 2NF,if and every Non-key attribute is fully functionally dependent on primaryKey.
- 3NF: A Relation scheme is said to be in 3NF,if and does not have transitivity dependencies. A Relation is said to be 3NF if every determinant is a key for each & every functionaldependency.
- BCNF: A Relation scheme is said to be BCNF if the following statements are true for eacg FD P->Q in set F of FDs that holds for each FD. P->Q in set F of FD's that holds over R. Here P is the subset of attributes of R & Q is a single attribute ofR.

First Normal Form:
Relation EMPLOYEE is not in 1NF because of multi-valued attribute EMP_PHONE.
Employee Table:

```
mysql> create table Employee(Emp_Id varchar(10),Emp_Name varchar(20),Phone varchar(30),State
varchar(10));
Query OK, 0 rows affected (0.01 sec)
mysql> insert into Employee values('14','John','9848892333,7608934521','UP');
Query OK, 1 row affected (0.00 sec)
mysql> insert into Employee values('20','Harry','8574783832','Bihar');
Query OK, 1 row affected (0.00 sec)

mysql> insert into Employee values('12','Sam','7390372385,8589830302','Punjab');
Query OK, 1 row affected (0.00 sec)
mysql> insert into Employee values('22','Romi','9876543210','Delhi');
Query OK, 1 row affected (0.01 sec)
mysql> select * from Employee;
```

```
mysql> select * from Employee;
+--------+----------+----------------------+--------+
| Emp_Id | Emp_Name | Phone                | State  |
+--------+----------+----------------------+--------+
| 14     | John     | 9848892333,7608934521 | UP     |
| 20     | Harry    | 8574783832           | Bihar  |
| 12     | Sam      | 7390372385,8589830302 | Punjab |
| 22     | Romi     | 9876543210           | Delhi  |
+--------+----------+----------------------+--------+
4 rows in set (0.00 sec)
```

- The above Relation can be converted into 1NF as follows

Employee-DetailsTable:

```
mysql> create table Employee_Details(Emp_Id varchar(10),Emp_Name varchar(20),Phone varchar(20),State varchar(10));
Query OK, 0 rows affected (0.01 sec)
mysql> insert into Employee_Details values('14','John','9848892333','UP');
Query OK, 1 row affected (0.00 sec)
mysql> insert into Employee_Details values('14','John','7608934521','UP');
Query OK, 1 row affected (0.00 sec)
mysql> insert into Employee_Details values('20','Harry','8574783832','Bihar');
Query OK, 1 row affected (0.00 sec)
mysql> insert into Employee_Details values('12','Sam','7390372385','Punjab');
Query OK, 1 row affected (0.01 sec)
mysql> insert into Employee_Details values('12','Sam','8589830302','Punjab');
Query OK, 1 row affected (0.00 sec)
mysql> insert into Employee_Details values('22','Romi','9876543210','Delhi');
Query OK, 1 row affected (0.00 sec)
mysql> select * from Employee_Details;
```

```
mysql> select * from Employee_Details;
+--------+----------+------------+--------+
| Emp_Id | Emp_Name | Phone      | State  |
+--------+----------+------------+--------+
| 14     | John     | 9848892333 | UP     |
| 14     | John     | 7608934521 | UP     |
| 20     | Harry    | 8574783832 | Bihar  |
| 12     | Sam      | 7390372385 | Punjab |
| 12     | Sam      | 8589830302 | Punjab |
| 22     | Romi     | 9876543210 | Delhi  |
+--------+----------+------------+--------+
6 rows in set (0.00 sec)
```

Second Normal Form:

- In the 2NF, relational must be in 1NF.
- In the second normal form, all non-key attributes are fully functional dependent on the primary key

Teacher Table: Relation not in 2nf

```
mysql> create table Teacher(Teacher_Idint,Subject varchar(10),Teacher_age int);
Query OK, 0 rows affected (0.00 sec)
mysql> insert into Teacher values(25,'Chemistry',30);
Query OK, 1 row affected (0.01 sec)
mysql> insert into Teacher values(25,'Biology',30);
Query OK, 1 row affected (0.00 sec)

mysql> insert into Teacher values(47,'English',35);
Query OK, 1 row affected (0.00 sec)
mysql> insert into Teacher values(83,'Maths',38);
Query OK, 1 row affected (0.01 sec)
mysql> insert into Teacher values(83,'Computer',38);
Query OK, 1 row affected (0.00 sec)
mysql> select * from Teacher;
```

```
mysql> select * from Teacher;
+------------+-----------+-------------+
| Teacher_Id | Subject   | Teacher_age |
+------------+-----------+-------------+
|         25 | Chemistry |          30 |
|         25 | Biology   |          30 |
|         47 | English   |          35 |
|         83 | Maths     |          38 |
|         83 | Computer  |          38 |
+------------+-----------+-------------+
5 rows in set (0.00 sec)
```

- Convert the above relation into 2NF as follows
- Create the Teacher_Detail table with Teacher-Id as primary key
- Create the Subject_Details table with Teacher-Id as foreign key

Teacher_DetailTable:

```
mysql> create table Teacher_Details(Teacher_Idint,Teacher_ageint,primary key(Teacher_Id));
Query OK, 0 rows affected (0.02 sec)
mysql> insert into Teacher_Details values(25,30);
Query OK, 1 row affected (0.00 sec)
mysql> insert into Teacher_Details values(47,35);
Query OK, 1 row affected (0.00 sec)
mysql> insert into Teacher_Details values(83,38);
Query OK, 1 row affected (0.00 sec)
mysql> select * from Teacher_Details;
```

```
mysql> select * from Teacher_Details;
+------------+-------------+
| Teacher_Id | Teacher_age |
+------------+-------------+
|         25 |          30 |
|         47 |          35 |
|         83 |          38 |
+------------+-------------+
3 rows in set (0.00 sec)
```

Subject_DetailsTable:

mysql> create table Subject_Details(Teacher_Idint,Subject varchar(10),foreign key(Teacher_Id) references Teacher_Details(Teacher_Id));
Query OK, 0 rows affected (0.02 sec)
mysql> insert into Subject_Details values(25,'Chemistry');
Query OK, 1 row affected (0.00 sec)
mysql> insert into Subject_Details values(25,'Biology');
Query OK, 1 row affected (0.01 sec)
mysql> insert into Subject_Details values(47,'English');
Query OK, 1 row affected (0.01 sec)
mysql> insert into Subject_Details values(83,'Maths');
Query OK, 1 row affected (0.01 sec)
mysql> insert into Subject_Details values(83,'Computer');
Query OK, 1 row affected (0.00 sec)
mysql> select * from Subject_Details;

```
mysql> select * from Subject_Details;
+------------+-----------+
| Teacher_Id | Subject   |
+------------+-----------+
|         25 | Chemistry |
|         25 | Biology   |
|         47 | English   |
|         83 | Maths     |
|         83 | Computer  |
+------------+-----------+
```

Third Normal Form:

A relation will be in 3NF if it is in 2NF and not contain any transitive partial dependency.

o 3NF is used to reduce the data duplication. It is also used to achieve the data integrity.

o If there is no transitive dependency for non-prime attributes, then the relation must be in third normal form.

A relation is in third normal form if it holds atleast one of the following conditions for every non-trivial function dependency X → Y.

1. X is a super key.
2. Y is a prime attribute, i.e., each element of Y is part of some candidate key.

Employee_DetailsTable:

mysql> create table Employee_Details1(Emp_Id varchar(10),Emp_Name varchar(20),Emp_Zip varchar(10),Emp_State varchar(10),
Emp_City varchar(10));

Query OK, 0 rows affected (0.02 sec)
mysql> insert into Employee_Details1 values('222','Harry','201010','UP','Noida');
Query OK, 1 row affected (0.00 sec)
mysql> insert into Employee_Details1 values('333','Stephan','02228','US','Boston');
Query OK, 1 row affected (0.02 sec)
mysql> insert into Employee_Details1 values('444','Lan','60007','US','Chicago');
Query OK, 1 row affected (0.00 sec)
mysql> insert into Employee_Details1 values('555','Katharine','06389','UK','Norwich');
Query OK, 1 row affected (0.00 sec)
mysql> insert into Employee_Details1 values('666','John','462007','MP','Bhopal');
Query OK, 1 row affected (0.00 sec)
mysql> Select * from Employee_Details1;

```
mysql> Select * from Employee_Details1;
+--------+-----------+---------+-----------+----------+
| Emp_Id | Emp_Name  | Emp_Zip | Emp_State | Emp_City |
+--------+-----------+---------+-----------+----------+
| 222    | Harry     | 201010  | UP        | Noida    |
| 333    | Stephan   | 02228   | US        | Boston   |
| 444    | Lan       | 60007   | US        | Chicago  |
| 555    | Katharine | 06389   | UK        | Norwich  |
| 666    | John      | 462007  | MP        | Bhopal   |
+--------+-----------+---------+-----------+----------+
5 rows in set (0.00 sec)
```

The above table can be converted into 3NF as follows

- Create the employee table with emp_id as primary key
- Create employee_zip table with emp_zip as primary key

EmployeeTable:
mysql> create table Employee1(Emp_Id varchar(10),Emp_Name varchar(20),Emp_Zip varchar(10),primary key(Emp_Id));
Query OK, 0 rows affected (0.01 sec)

mysql> insert into Employee1 values('222','Harry','201010');
Query OK, 1 row affected (0.00 sec)
mysql> insert into Employee1 values('333','Stephan','02228');
Query OK, 1 row affected (0.00 sec)
mysql> insert into Employee1 values('444','Lan','60007');
Query OK, 1 row affected (0.01 sec)
mysql> insert into Employee1 values('555','Katharine','06389');
Query OK, 1 row affected (0.00 sec)
mysql> insert into Employee1 values('666','John','462007');
Query OK, 1 row affected (0.02 sec)
mysql> select * from Employee1;

```
mysql> select * from Employee1;
+--------+-----------+---------+
| Emp_Id | Emp_Name  | Emp_Zip |
+--------+-----------+---------+
| 222    | Harry     | 201010  |
| 333    | Stephan   | 02228   |
| 444    | Lan       | 60007   |
| 555    | Katharine | 06389   |
| 666    | John      | 462007  |
+--------+-----------+---------+
5 rows in set (0.00 sec)
```

Employee _Zip Table:

mysql> create table Employee_Zip(Emp_Zip varchar(10),Emp_State varchar(10),Emp_City varchar(10),primary key(Emp_Zip));
Query OK, 0 rows affected (0.01 sec)

mysql> insert into Employee_Zip values('201010','UP','Noida');
Query OK, 1 row affected (0.00 sec)
mysql> insert into Employee_Zip values('02228','US','Boston');
Query OK, 1 row affected (0.00 sec)
mysql> insert into Employee_Zip values('60007','US','Chicago');
Query OK, 1 row affected (0.00 sec)
mysql> insert into Employee_Zip values('06389','UK','Norwich');
Query OK, 1 row affected (0.02 sec)
mysql> insert into Employee_Zip values('462007','MP','Bhopal');
Query OK, 1 row affected (0.00 sec)
mysql> select * from Employee_Zip;

```
mysql> select * from Employee_Zip;
+---------+-----------+----------+
| Emp_Zip | Emp_State | Emp_City |
+---------+-----------+----------+
| 02228   | US        | Boston   |
| 06389   | UK        | Norwich  |
| 201010  | UP        | Noida    |
| 462007  | MP        | Bhopal   |
| 60007   | US        | Chicago  |
+---------+-----------+----------+
5 rows in set (0.00 sec)
```

EXPERIMENT – 4
PRACTICING DDL COMMANDS

Aim: Creating the Tables and altering the tables

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.

The Following the DDL Commands:

1. Create
2. Alter
    a) Alter-add
    b) Alter-modify
    c) Alter-drop
    d) Alter-rename
3. Drop
4. Rename

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

Syntax: Create table table-name (column-name1 datatype(size), column-name2 datatype(size)----------

---------- ---------- columnn datatype(size));

2.Alter: This is used to add, modify, drop and rename the extra fields into existing relations.

(a). Alter-add: This is used to add some extra fields into existing relations.

Syntax: alter table table-name add(column-name1 datatype(size),column-name2 datatype(size));

(b). Alter-modify: This is used to change the data type of a column in existingrelations.

Syntax: alter table table-name modify column-name datatype(size);

(c). Alter-drop: This is used to remove any field of existing relations.

Syntax: alter table table-name drop column-name;

(d). Alter-rename: This is used to change the name of fields in existing relations.

Syntax: alter table table-name change old-column-name new-column-name;

3.Rename: It is used to change the name of the table.

Syntax: rename table old-table-name to new-table-name;

4.Drop: This is used to delete the structure of a relation. It permanently deletes the records in the table.

Syntax: drop table table-name;

1. **Create the Student table for the following schema**

(Std_id:varchar(10),Std_Name:varchar(20), Std_Phone: varchar(10));

2. Add a new columns Email_id:varchar(20),Address varchar(20) to the Student table.
3. Change the datatype of the Std-Id from varchar(10) to varchar(20).
4. Change the column-name Email_id to Std-Email.
5. Delete the Address column in the Student Table.
6. Change the table name Student to Student1.
7. Delete the Student1 table in Database.

1. Create the Student table for the following schema

(Std_id: varchar(10),Std_Name:varchar(20), Std_Phone: varchar(10));

```
mysql> create table Student(Std_Id varchar(10),Std_Name varchar(20),Std_Phone varchar(10),
primary key(Std_Id));
Query OK, 0 rows affected (0.01 sec)
mysql> desc Student;
```

```
mysql> desc Student;
+-----------+-------------+------+-----+---------+-------+
| Field     | Type        | Null | Key | Default | Extra |
+-----------+-------------+------+-----+---------+-------+
| Std_Id    | varchar(10) | NO   | PRI |         |       |
| Std_Name  | varchar(20) | YES  |     | NULL    |       |
| Std_Phone | varchar(10) | YES  |     | NULL    |       |
+-----------+-------------+------+-----+---------+-------+
3 rows in set (0.00 sec)
```

2. Add a new columns Email_id:varchar(20),Address varchar(20) to the Student table.

```
mysql> alter table Student add(Email_id varchar(20),Address varchar(20));
       Query OK, 0 rows affected (0.02 sec)
mysql> desc Student;
```

```
mysql> desc Student;
+-----------+-------------+------+-----+---------+-------+
| Field     | Type        | Null | Key | Default | Extra |
+-----------+-------------+------+-----+---------+-------+
| Std_Id    | varchar(10) | NO   | PRI |         |       |
| Std_Name  | varchar(20) | YES  |     | NULL    |       |
| Std_Phone | varchar(10) | YES  |     | NULL    |       |
| Email_id  | varchar(20) | YES  |     | NULL    |       |
| Address   | varchar(20) | YES  |     | NULL    |       |
+-----------+-------------+------+-----+---------+-------+
5 rows in set (0.01 sec)
```

3. Change the datatype of the Std-Id from varchar(10) to varchar(20).

mysql> alter table Student modify Std_Id varchar(20);

Query OK, 0 rows affected (0.00 sec)

mysql> desc student;

```
mysql> desc student;
+-----------+-------------+------+-----+---------+-------+
| Field     | Type        | Null | Key | Default | Extra |
+-----------+-------------+------+-----+---------+-------+
| Std_Id    | varchar(20) | NO   | PRI |         |       |
| Std_Name  | varchar(20) | YES  |     | NULL    |       |
| Std_Phone | varchar(10) | YES  |     | NULL    |       |
| Email_id  | varchar(20) | YES  |     | NULL    |       |
| Address   | varchar(20) | YES  |     | NULL    |       |
+-----------+-------------+------+-----+---------+-------+
5 rows in set (0.00 sec)
```

4. Change the column-name Email_id to Std-Email.

mysql> alter table Student change column Email_id Std_Email varchar(20);

Query OK, 0 rows affected (0.01 sec)

mysql> desc Student;

```
mysql> desc Student;
+-----------+-------------+------+-----+---------+-------+
| Field     | Type        | Null | Key | Default | Extra |
+-----------+-------------+------+-----+---------+-------+
| Std_Id    | varchar(20) | NO   | PRI |         |       |
| Std_Name  | varchar(20) | YES  |     | NULL    |       |
| Std_Phone | varchar(10) | YES  |     | NULL    |       |
| Std_Email | varchar(20) | YES  |     | NULL    |       |
| Address   | varchar(20) | YES  |     | NULL    |       |
+-----------+-------------+------+-----+---------+-------+
5 rows in set (0.02 sec)
```

5. Delete the Address column in the Student Table.

mysql> alter table Student drop Address;

Query OK, 0 rows affected (0.02 sec)

mysql> desc Student;

```
mysql> desc Student;
+-----------+-------------+------+-----+---------+-------+
| Field     | Type        | Null | Key | Default | Extra |
+-----------+-------------+------+-----+---------+-------+
| Std_Id    | varchar(20) | NO   | PRI |         |       |
| Std_Name  | varchar(20) | YES  |     | NULL    |       |
| Std_Phone | varchar(10) | YES  |     | NULL    |       |
| Std_Email | varchar(20) | YES  |     | NULL    |       |
+-----------+-------------+------+-----+---------+-------+
4 rows in set (0.02 sec)
```

6. Change the table name Student to Student1.

mysql> rename table Student to Student1;

Query OK, 0 rows affected (0.00 sec)

mysql> desc Student1;

```
mysql> desc Student1;
+-----------+-------------+------+-----+---------+-------+
| Field     | Type        | Null | Key | Default | Extra |
+-----------+-------------+------+-----+---------+-------+
| Std_Id    | varchar(20) | NO   | PRI |         |       |
| Std_Name  | varchar(20) | YES  |     | NULL    |       |
| Std_Phone | varchar(10) | YES  |     | NULL    |       |
| Std_Email | varchar(20) | YES  |     | NULL    |       |
+-----------+-------------+------+-----+---------+-------+
4 rows in set (0.03 sec)
```

mysql> desc Student;

```
mysql> desc Student;
ERROR 1146 (42S02): Table 'csd.student' doesn't exist
```

7. **Delete the Student1 table in Database.**

mysql> drop table Student1;

Query OK, 0 rows affected (0.00 sec)

mysql> desc Student1;

```
mysql> desc Student1;
ERROR 1146 (42S02): Table 'csd.student1' doesn't exist
```

EXPERIMENT – 5
PRACTICING DML COMMANDS

AIM: Create a DML Commands are used to manage data within the scheme objects.

Data Manipulation Language(DML): The DML commands in Structured Query Language change the data present in the SQL database. We can easily access, store, modify, update and delete the existing records from the database using DML commands.

The Following are the DML Commands:

1. Insert
2. Select
3. Update
4. Delete

1. **Insert:** This is used to insert the data in existing table.

Syntax1: insert into table-name values(column1-value,column2-value-------- columnn-valuen);

Syntax2:insert into table-name(column-name1,column-name2----- column-nameN)
values(value1,value2--------- valuen);

2. **Select:**The SELECT command shows the records of the specified table. It also shows the particular record of a particular column by using the WHERE clause.

Syntax1: select * from table-name;

Syntax1: select column-name1,column-name2------- column-nameN from table-name;

Syntax1: select * from table-name where condition;

3. **Update:**which allows users to update or modify the existing data in database tables.

Syntax: update table-name set (column-name1=value1,column-name2=value2--------column-nameN=valueN) where condition;

4. **Delete:**which allows SQL users to remove single or multiple existing records from the database tables and We use the WHERE clause with the DELETE command to select specific rows from the table.

Syntax: delete from table-name where condition;

1.Create and insert the data in Student table.

Std_Id	**Name**	**Email_Id**	**CGPA**	**Address**
101	John	John101@gmail.com	9	Hyderabad
102	Bobby	Bobby102@gmail.com	10	Mumbai

103	Cheery	Cheery103@gmail.com	8	Delhi
104	Ramu	Ramu104@gmail.com	8	Hyderabad
105	Teju	Teju105@gmail.com	9	Chenai
109	Srija	Srija109@gmail.com	7	Karela
110	Swathi	Swathi110@gmail.com	9	Bengaluru
113	Soumya	Soumya113@gmail.com	8	Delhi
120	Charan	Charan120@gmail.com	9	Mumbai

2.Diaplay the Student Name and CGPA from the student table?

3.Diaplay all the details about student where Student id is 104?

4.Display the Name, Email_id and address of the students where CGPA is 8?

5. Write a SQL Query to change the city as Hyderabad and CGPA as 8 for the student109?

6. Write a SQL Query to delete students with student id is 105?

7. Write a SQL Query to delete students with CGPA is 8?

8.Write a SQL Query to delete all data in the student table?

1. **Creating a Table:**

mysql> create table Student(Std_Id int primary key,Name varchar(10),Email_Id varchar(20),CGPA varchar(10),Address varchar(10));

Query OK, 0 rows affected (0.00 sec)

mysql> desc student;

```
mysql> desc student;
+----------+-------------+------+-----+---------+-------+
| Field    | Type        | Null | Key | Default | Extra |
+----------+-------------+------+-----+---------+-------+
| Std_Id   | int(11)     | NO   | PRI | NULL    |       |
| Name     | varchar(10) | YES  |     | NULL    |       |
| Email_Id | varchar(20) | YES  |     | NULL    |       |
| CGPA     | varchar(10) | YES  |     | NULL    |       |
| Address  | varchar(10) | YES  |     | NULL    |       |
+----------+-------------+------+-----+---------+-------+
5 rows in set (0.01 sec)
```

Insert the Data:

mysql> insert into student values('101','John','John101@gmail.com','9','Hyderabad');
Query OK, 1 row affected (0.00 sec)
mysql> insert into student values('102','Bobby','Bobby102@gmail.com','10','Mumbai');
Query OK, 1 row affected (0.01 sec)
mysql> insert into student values('103','Cheery','Cheery103@gmail.com','8','Delhi');

```
Query OK, 1 row affected (0.00 sec)
mysql> insert into student values('104','Ramu','Ramu104@gmail.com','8',' Hyderabad ');
Query OK, 1 row affected (0.01 sec)
mysql> insert into student values('105',' Teju ','Teju105@gmail.com','9','Chenai');
Query OK, 1 row affected (0.00 sec)
mysql> insert into student values('120','Charan','Charan120@gmail.com','9','Mumbai');
Query OK, 1 row affected (0.00 sec)
mysql> insert into student values('109','Srija','Srija109@gmail.com','7','Kerala');
Query OK, 1 row affected (0.00 sec)
mysql> insert into student values('110','Srija','Swathi110@gmail.com','9','Bengaluru');
Query OK, 1 row affected (0.01 sec)
mysql> insert into student values('113','Soumya','Soumya113@gmail.com','8','Delhi');
Query OK, 1 row affected (0.00 sec)
mysql> insert into student values('120','Charan','Charan120@gmail.com','9','Mumbai');
Query OK, 1 row affected (0.00 sec)
mysql> Select * from Student;
```

```
mysql> Select * from Student;
+--------+--------+--------------------+------+-----------+
| Std_Id | Name   | Email_Id           | CGPA | Address   |
+--------+--------+--------------------+------+-----------+
|    101 | John   | John101@gmail.com  | 9    | Hyderabad |
|    102 | Bobby  | Bobby102@gmail.com | 10   | Mumbai    |
|    103 | Cheery | Cheery103@gmail.com| 8    | Delhi     |
|    104 | Ramu   | Ramu104@gmail.com  | 8    | Hyderabad |
|    105 | Teju   | Teju105@gmail.com  | 9    | Chenai    |
|    109 | Srija  | Srija109@gmail.com | 7    | Kerala    |
|    110 | Srija  | Swathi110@gmail.com| 9    | Bengaluru |
|    113 | Soumya | Soumya113@gmail.com| 8    | Delhi     |
|    120 | Charan | Charan120@gmail.com| 9    | Mumbai    |
+------+--+--------+--------------------+------+-----------+
9 rows in set (0.00 sec)
```

2.**Diaplay the Student Name and CGPA from the student table?**

```
mysql> select Name,CGPA from student;
```

```
mysql> select Name,CGPA from student;
+--------+------+
| Name   | CGPA |
+--------+------+
| John   | 9    |
| Bobby  | 10   |
| Cheery | 8    |
| Ramu   | 8    |
| Teju   | 9    |
| Srija  | 7    |
| Srija  | 9    |
| Soumya | 8    |
| Charan | 9    |
+--------+------+
9 rows in set (0.00 sec)
```

3.Diaplay all the details about student where Student id is 104?

mysql> select * from student where Std_Id=104;

```
mysql> select * from student where Std_Id=104;
+--------+------+------------------+------+-----------+
| Std_Id | Name | Email_Id         | CGPA | Address   |
+--------+------+------------------+------+-----------+
|    104 | Ramu | Ramu104@gmail.com | 8    | Hyderabad |
+--------+------+------------------+------+-----------+
1 row in set (0.00 sec)
```

4.Display the Name, Email_id and address of the students where CGPA is 8?

mysql> select Name,Email_Id,Address from student where CGPA='8';

```
mysql> select Name,Email_Id,Address from student where CGPA='8';
+--------+--------------------+-----------+
| Name   | Email_Id           | Address   |
+--------+--------------------+-----------+
| Cheery | Cheery103@gmail.com | Delhi     |
| Ramu   | Ramu104@gmail.com   | Hyderabad |
| Soumya | Soumya113@gmail.com | Delhi     |
+--------+--------------------+-----------+
3 rows in set (0.00 sec)
```

5. **Write a SQL Query to change the city as Hyderabad and CGPA as 8 for the student 109?**

mysql> update student set Address='Hyderabad',CGPA='8' where Std_Id=109;
Query OK, 1 row affected (0.02 sec)
mysql> select * from student;

```
mysql> select * from student;
+--------+--------+--------------------+------+-----------+
| Std_Id | Name   | Email_Id           | CGPA | Address   |
+--------+--------+--------------------+------+-----------+
|    101 | John   | John101@gmail.com  | 9    | Hyderabad |
|    102 | Bobby  | Bobby102@gmail.com | 10   | Mumbai    |
|    103 | Cheery | Cheery103@gmail.com| 8    | Delhi     |
|    104 | Ramu   | Ramu104@gmail.com  | 8    | Hyderabad |
|    105 | Teju   | Teju105@gmail.com  | 9    | Chenai    |
|    109 | Srija  | Srija109@gmail.com | 8    | Hyderabad |
|    110 | Srija  | Swathi110@gmail.com| 9    | Bengaluru |
|    113 | Soumya | Soumya113@gmail.com| 8    | Delhi     |
|    120 | Charan | Charan120@gmail.com| 9    | Mumbai    |
+--------+--------+--------------------+------+-----------+
9 rows in set (0.00 sec)
```

6. Write a SQL Query to delete students with student id is 105?

mysql> delete from student where Std_Id=105;

Query OK, 1 row affected (0.02 sec)

mysql> select * from Student;

```
mysql> select * from Student;
+--------+--------+--------------------+------+-----------+
| Std_Id | Name   | Email_Id           | CGPA | Address   |
+--------+--------+--------------------+------+-----------+
|    101 | John   | John101@gmail.com  | 9    | Hyderabad |
|    102 | Bobby  | Bobby102@gmail.com | 10   | Mumbai    |
|    103 | Cheery | Cheery103@gmail.com| 8    | Delhi     |
|    104 | Ramu   | Ramu104@gmail.com  | 8    | Hyderabad |
|    109 | Srija  | Srija109@gmail.com | 8    | Hyderabad |
|    110 | Srija  | Swathi110@gmail.com| 9    | Bengaluru |
|    113 | Soumya | Soumya113@gmail.com| 8    | Delhi     |
|    120 | Charan | Charan120@gmail.com| 9    | Mumbai    |
+--------+--------+--------------------+------+-----------+
8 rows in set (0.00 sec)
```

7. Write a SQL Query to delete students with CGPA is 8?

mysql> delete from student where CGPA='8';
Query OK, 4 rows affected (0.00 sec)

mysql> select * from Student;

```
mysql> select * from Student;
+--------+--------+--------------------+------+-----------+
| Std_Id | Name   | Email_Id           | CGPA | Address   |
+--------+--------+--------------------+------+-----------+
|    101 | John   | John101@gmail.com  | 9    | Hyderabad |
|    102 | Bobby  | Bobby102@gmail.com | 10   | Mumbai    |
|    110 | Srija  | Swathi110@gmail.com| 9    | Bengaluru |
|    120 | Charan | Charan120@gmail.com| 9    | Mumbai    |
+--------+--------+--------------------+------+-----------+
4 rows in set (0.00 sec)
```

8.Write a SQL Query to delete all data in the student table?

mysql> delete from student;
Query OK, 4 rows affected (0.02 sec)
mysql> select * from student;

```
mysql> select * from student;
Empty set (0.00 sec)
```

EXPERIMENT – 6

Querying (using ANY, ALL, IN, Exists, NOT EXISTS, UNION, INTERSECT, Constraints etc.)

Aim: Practice the following Queries:

1. DisplayuniquePNR_NOofallpassengers
2. Displayallthenamesofmalepassengers.
3. Displaytheticketnumbersandnamesofallthepassengers.
4. Findtheticketnumbersofthepassengerswhosenamestartwith‘r’andendswith‘h’.
5. FindthenamesofPassengerswhoseageisbetween30and45.
6. Displayallthepassengersnamesbeginningwith‘A’.
7. DisplaythesortedlistofPassengersnames

```
mysql> DESC RESERVATION2;
+-------------+-------------+------+-----+---------+-------+
| Field       | Type        | Null | Key | Default | Extra |
+-------------+-------------+------+-----+---------+-------+
| PNRNO       | int(11)     | NO   | PRI |         |       |
| Journeydate | datetime    | YES  |     | NULL    |       |
| NoofSeats   | int(11)     | YES  |     | NULL    |       |
| Address     | varchar(20) | YES  |     | NULL    |       |
| CONTACTNO   | varchar(15) | YES  |     | NULL    |       |
+-------------+-------------+------+-----+---------+-------+
5 rows in set (0.00 sec)

mysql> insert into reservation2 values(10201,'2012-02-20 10:20:25',05,'HYD',9654
235242);
Query OK, 1 row affected (0.03 sec)

mysql> insert into reservation2 values(10202,'2012-02-22 10:22:25',05,'HYD',9654
232451);
Query OK, 1 row affected (0.02 sec)

mysql> insert into reservation2 values(10203,'2012-03-22 10:30:25',05,'DELHI',96
54587960);
Query OK, 1 row affected (0.01 sec)

mysql> insert into reservation2 values(10204,'2013-03-22 11:30:25',05,'CHENNAI',
9845761254);
Query OK, 1 row affected (0.02 sec)

mysql> SELECT * FROM RESERVATION2;
+-------+---------------------+-----------+---------+------------+
| PNRNO | Journeydate         | NoofSeats | Address | CONTACTNO  |
+-------+---------------------+-----------+---------+------------+
| 10201 | 2012-02-20 10:20:25 |         5 | HYD     | 9654235242 |
| 10202 | 2012-02-22 10:22:25 |         5 | HYD     | 9654232451 |
| 10203 | 2012-03-22 10:30:25 |         5 | DELHI   | 9654587960 |
| 10204 | 2013-03-22 11:30:25 |         5 | CHENNAI | 9845761254 |
+-------+---------------------+-----------+---------+------------+
4 rows in set (0.01 sec)
```

mysql> insert into passenger2values(82302,'Smith',23,'M','Hyderabad'); Query OK, 1 row affected (0.02sec)

mysql> insert into passenger2values(82303,'Neha',23,'F','Hyderabad'); Query OK, 1 row affected (0.01sec)

mysql> insert into passenger2values(82304,'Neha',35,'F','Hyderabad'); Query OK, 1 row affected (0.03sec)

mysql> insert into passenger2values(82306,'Ramu',40,'M','Hyderabad'); Query OK, 1 row affected (0.02sec)

mysql> insert into passenger2values(82308,'Aakash',40,'M','Hyderabad'); Query OK, 1 row affected (0.02sec)

mysql>insertintopassenger2values(82402,'Aravind',42,'M','Hyderabad'); Query OK, 1 row affected (0.02sec)

mysql>insertintopassenger2values(82403,'Avinash',42,'M','Hyderabad'); Query OK, 1 row affected (0.02sec)

mysql>insertintopassenger2values(82502,'Ramesh',23,'M','Hyderabad'); Query OK, 1 row affected (0.02sec)

mysql> insert into passenger2 values(82602,'Rajesh',23,'M','Hyderabad'); Query OK, 1 row affected (0.02 sec)

RESERVATION2

mysql> insert into reservation2 values(10201,'2012-02-20 10:20:25',05,'HYD',9654235242); Query OK, 1 row affected (0.03sec)

mysql> insert into reservation2 values(10202,'2012-02-22 10:22:25',05,'HYD',9654232451); Query OK, 1 row affected (0.02sec)

mysql> insert into reservation2 values(10203,'2012-03-22 10:30:25',05,'DELHI',96 54587960); Query OK, 1 row affected (0.01sec)

mysql> insert into reservation2 values(10204,'2013-03-22 11:30:25',05,'CHENNAI',9845761254); Query OK, 1 row affected (0.02sec)

1. Display unique PNR_NO of all reservationMysql>Select

   DISTINCT PNR_NO fromReservation;

PNR_No
10201
10202
10203
10204

2. display all the names of male passengers.

```
mysql> Select p.name from passenger2 p
        where     p.passportid IN (select p2.passportid from passenger2 p2
         where      p2.sex='M');
```

```
mysql> SELECT * FROM PASSENGER2;
+------------+---------+-----+------+-----------+
| passportId | name    | Age | Sex  | Address   |
+------------+---------+-----+------+-----------+
|        145 | Ramesh  |  45 | M    | abc123    |
|        278 | Geetha  |  36 | F    | abc124    |
|       4590 | Ram     |  30 | M    | abc12     |
|       5622 | Seetha  |  32 | F    | abc55     |
|       6789 | Ravi    |  50 | M    | abc14     |
|      82302 | Smith   |  23 | M    | Hyderabad |
|      82303 | Neha    |  23 | F    | Hyderabad |
|      82304 | Neha    |  35 | F    | Hyderabad |
|      82306 | Ramu    |  40 | M    | Hyderabad |
|      82308 | Aakash  |  40 | M    | Hyderabad |
|      82402 | Aravind |  42 | M    | Hyderabad |
|      82403 | Avinash |  42 | M    | Hyderabad |
|      82502 | Ramesh  |  23 | M    | Hyderabad |
|      82602 | Rajesh  |  23 | M    | Hyderabad |
+------------+---------+-----+------+-----------+
14 rows in set (0.00 sec)

mysql> SELECT P.NAME FROM PASSENGER2 P
    -> WHERE P.PASSPORTID IN (SELECT P2.PASSPORTID
    -> FROM PASSENGER2 P2
    -> WHERE P2.SEX='M');
+---------+
| NAME    |
+---------+
| Ramesh  |
| Ram     |
| Ravi    |
| Smith   |
| Ramu    |
| Aakash  |
| Aravind |
| Avinash |
| Ramesh  |
| Rajesh  |
+---------+
10 rows in set (0.00 sec)
```

3. Display the ticket numbers and names of all thepassengers.

```
mysql> desc passengerticket;
+------------+-------------+------+-----+---------+-------+
| Field      | Type        | Null | Key | Default | Extra |
+------------+-------------+------+-----+---------+-------+
| passportid | varchar(15) | NO   | PRI |         |       |
| TicketNo   | int(11)     | YES  |     | NULL    |       |
+------------+-------------+------+-----+---------+-------+
2 rows in set (0.00 sec)

mysql> insert into passengerticket values(145,100);
Query OK, 1 row affected (0.03 sec)

mysql> insert into passengerticket values(278,200);
Query OK, 1 row affected (0.03 sec)

mysql> insert into passengerticket values(6789,300);
Query OK, 1 row affected (0.03 sec)

mysql> insert into passengerticket values(82302,400);
Query OK, 1 row affected (0.03 sec)

mysql> insert into passengerticket values(82403,500);
Query OK, 1 row affected (0.03 sec)

mysql> insert into passengerticket values(82502,600);
Query OK, 1 row affected (0.02 sec)
```

mysql> select t.ticketno,p.name from passengerticket t,passenger2 p where t.passportid = p.passportid;

```
mysql> SELECT T.TICKETNO,P.NAME FROM PASSENGERTICKET T,PASSENGER2 P
    -> WHERE T.PASSPORTID=P.PASSPORTID;
+----------+---------+
| TICKETNO | NAME    |
+----------+---------+
|      100 | Ramesh  |
|      200 | Geetha  |
|      300 | Ravi    |
|      400 | Smith   |
|      500 | Avinash |
|      600 | Ramesh  |
+----------+---------+
6 rows in set (0.00 sec)
```

4. Findtheticketnumbersofthepassengerswhosenamestartwith‘r’andendswith‘h’.

MySQL> SELECT Name FROM Passenger WHERE name LIKE ‘R%H’

Name
Rajesh
Ramesh
Ramesh

```
mysql> SELECT * FROM PASSENGER2;
+------------+---------+-----+------+-----------+
| passportId | name    | Age | Sex  | Address   |
+------------+---------+-----+------+-----------+
|        145 | Ramesh  |  45 | M    | abc123    |
|        278 | Geetha  |  36 | F    | abc124    |
|       4590 | Ram     |  30 | M    | abc12     |
|       5622 | Seetha  |  32 | F    | abc55     |
|       6789 | Ravi    |  50 | M    | abc14     |
|      82302 | Smith   |  23 | M    | Hyderabad |
|      82303 | Neha    |  23 | F    | Hyderabad |
|      82304 | Neha    |  35 | F    | Hyderabad |
|      82306 | Ramu    |  40 | M    | Hyderabad |
|      82308 | Aakash  |  40 | M    | Hyderabad |
|      82402 | Aravind |  42 | M    | Hyderabad |
|      82403 | Avinash |  42 | M    | Hyderabad |
|      82502 | Ramesh  |  23 | M    | Hyderabad |
|      82602 | Rajesh  |  23 | M    | Hyderabad |
+------------+---------+-----+------+-----------+
14 rows in set (0.00 sec)

mysql> SELECT NAME FROM PASSENGER2 WHERE NAME LIKE 'R%H';
+--------+
| NAME   |
+--------+
| Ramesh |
| Ramesh |
| Rajesh |
+--------+
3 rows in set (0.00 sec)
```

5. FindthenamesofPassengerswhoseageisbetween30and45.

MySQL> SELECT Name FROM PASSENGER WHERE AGE BETWEEN 30 AND 45

```
mysql> SELECT * FROM PASSENGER2;
+------------+---------+-----+------+-----------+
| passportId | name    | Age | Sex  | Address   |
+------------+---------+-----+------+-----------+
|        145 | Ramesh  |  45 | M    | abc123    |
|        278 | Geetha  |  36 | F    | abc124    |
|       4590 | Ram     |  30 | M    | abc12     |
|       5622 | Seetha  |  32 | F    | abc55     |
|       6789 | Ravi    |  50 | M    | abc14     |
|      82302 | Smith   |  23 | M    | Hyderabad |
|      82303 | Neha    |  23 | F    | Hyderabad |
|      82304 | Neha    |  35 | F    | Hyderabad |
|      82306 | Ramu    |  40 | M    | Hyderabad |
|      82308 | Aakash  |  40 | M    | Hyderabad |
|      82402 | Aravind |  42 | M    | Hyderabad |
|      82403 | Avinash |  42 | M    | Hyderabad |
|      82502 | Ramesh  |  23 | M    | Hyderabad |
|      82602 | Rajesh  |  23 | M    | Hyderabad |
+------------+---------+-----+------+-----------+
14 rows in set (0.00 sec)

mysql> SELECT Name FROM PASSENGER2 WHERE AGE BETWEEN 30 AND 45;
+---------+
| Name    |
+---------+
| Ramesh  |
| Geetha  |
| Ram     |
| Seetha  |
| Neha    |
| Ramu    |
| Aakash  |
| Aravind |
| Avinash |
+---------+
9 rows in set (0.00 sec)
```

6. Display all the passengers names beginning with‘A’.

MySQL> SELECT * FROM PASSENGER WHERE NAME LIKE ‘A%’;

Name
Akash
Arivind
Avinash

```
mysql> SELECT * FROM PASSENGER2;
+------------+---------+-----+------+-----------+
| passportId | name    | Age | Sex  | Address   |
+------------+---------+-----+------+-----------+
|        145 | Ramesh  |  45 | M    | abc123    |
|        278 | Geetha  |  36 | F    | abc124    |
|       4590 | Ram     |  30 | M    | abc12     |
|       5622 | Seetha  |  32 | F    | abc55     |
|       6789 | Ravi    |  50 | M    | abc14     |
|      82302 | Smith   |  23 | M    | Hyderabad |
|      82303 | Neha    |  23 | F    | Hyderabad |
|      82304 | Neha    |  35 | F    | Hyderabad |
|      82306 | Ramu    |  40 | M    | Hyderabad |
|      82308 | Aakash  |  40 | M    | Hyderabad |
|      82402 | Aravind |  42 | M    | Hyderabad |
|      82403 | Avinash |  42 | M    | Hyderabad |
|      82502 | Ramesh  |  23 | M    | Hyderabad |
|      82602 | Rajesh  |  23 | M    | Hyderabad |
+------------+---------+-----+------+-----------+
14 rows in set (0.00 sec)

mysql> SELECT NAME FROM PASSENGER2 WHERE NAME LIKE 'A%';
+---------+
| NAME    |
+---------+
| Aakash  |
| Aravind |
| Avinash |
+---------+
3 rows in set (0.00 sec)
```

7. Display the sorted list of Passengersnames

MySQL> SELECT NAME FROM PASSENGER ORDER BY NAME;

```
mysql> SELECT * FROM PASSENGER2;
+------------+---------+-----+------+-----------+
| passportId | name    | Age | Sex  | Address   |
+------------+---------+-----+------+-----------+
|        145 | Ramesh  |  45 | M    | abc123    |
|        278 | Geetha  |  36 | F    | abc124    |
|       4590 | Ram     |  30 | M    | abc12     |
|       5622 | Seetha  |  32 | F    | abc55     |
|       6789 | Ravi    |  50 | M    | abc14     |
|      82302 | Smith   |  23 | M    | Hyderabad |
|      82303 | Neha    |  23 | F    | Hyderabad |
|      82304 | Neha    |  35 | F    | Hyderabad |
|      82306 | Ramu    |  40 | M    | Hyderabad |
|      82308 | Aakash  |  40 | M    | Hyderabad |
|      82402 | Aravind |  42 | M    | Hyderabad |
|      82403 | Avinash |  42 | M    | Hyderabad |
|      82502 | Ramesh  |  23 | M    | Hyderabad |
|      82602 | Rajesh  |  23 | M    | Hyderabad |
+------------+---------+-----+------+-----------+
14 rows in set (0.00 sec)

mysql> SELECT NAME FROM PASSENGER2 ORDER BY NAME;
+---------+
| NAME    |
+---------+
| Aakash  |
| Aravind |
| Avinash |
| Geetha  |
| Neha    |
| Neha    |
| Rajesh  |
| Ram     |
| Ramesh  |
| Ramesh  |
| Ramu    |
| Ravi    |
| Seetha  |
| Smith   |
+---------+
14 rows in set (0.02 sec)
```

EXPERIMENT – 7

Querying Aggregate Functions(COUNT,SUM,AVG,MAX and MIN)

Aim: To Practice Queries using Aggregate functions for the following

1. Write a Query to display the information present in the passenger andcancellation tables
2. Display the number of days in a week on which the AP123 bus isavailable
3. Find number of tickets booked for each PNR_No using GROUP BYCLAUSE
4. Find the distinct PNR Numbers that arepresent.

1. Write a Query to display the information present in the passenger and cancellationtables

MYSQL> CREATE TABLE CANCELLATION2(PNRNO INT PRIMARY KEY,JOURNEYDATE DATETIME, NOOFSEATS INT,ADDRESS VARCHAR(20),CONTACTNO INT,STATUS VARCHAR(10),FOREIGN KEY(PNRNO) REFERENCES RESERVATION2(PNRNO));

mysql> INSERT INTO CANCELLATION2 VALUES(10201,'2012-02-20 10:20:25',2,'HYD',9654235242,'CONFIRM');

mysql> INSERT INTO CANCELLATION2 VALUES(10202,'2012-02-22 10:22:25',2,'HYD',9654232451,'CONFIRM');

mysql> INSERT INTO CANCELLATION2 VALUES(10203,'2012-03-22 10:30:25',2,'DELHI',9654587960,'CONFIRM');

MySQL> SELECT * FROM RESERVATION UNION

SELECT * FROM CANCELLATION;

```
mysql> SELECT * FROM RESERVATION2
    -> UNION
    -> SELECT * FROM CANCELLATION2;
+-------+---------------------+-----------+---------+------------+---------+
| PNRNO | Journeydate         | NoofSeats | Address | CONTACTNO  | STATUS  |
+-------+---------------------+-----------+---------+------------+---------+
| 10201 | 2012-02-20 10:20:25 |         5 | HYD     | 9654235242 | NULL    |
| 10202 | 2012-02-22 10:22:25 |         5 | HYD     | 9654232451 | NULL    |
| 10203 | 2012-03-22 10:30:25 |         5 | DELHI   | 9654587960 | NULL    |
| 10204 | 2013-03-22 11:30:25 |         5 | CHENNAI | 9845761254 | NULL    |
| 10201 | 2012-02-20 10:20:25 |         2 | HYD     | 9654235242 | CONFIRM |
| 10202 | 2012-02-22 10:22:25 |         2 | HYD     | 9654232451 | CONFIRM |
| 10203 | 2012-03-22 10:30:25 |         2 | DELHI   | 9654587960 | CONFIRM |
+-------+---------------------+-----------+---------+------------+---------+
7 rows in set (0.01 sec)
```

2. DisplaytheMinimumageofthePassenger

MySQL> SELECT MIN(AGE) as MINAGE FROM PASSENGER;

```
mysql> SELECT * FROM PASSENGER2;
+-------------+---------+------+------+-------------+
| passportId  | name    | Age  | Sex  | Address     |
+-------------+---------+------+------+-------------+
|         145 | Ramesh  |   45 | M    | abc123      |
|         278 | Geetha  |   36 | F    | abc124      |
|        4590 | Ram     |   30 | M    | abc12       |
|        5622 | Seetha  |   32 | F    | abc55       |
|        6789 | Ravi    |   50 | M    | abc14       |
|       82302 | Smith   |   23 | M    | Hyderabad   |
|       82303 | Neha    |   23 | F    | Hyderabad   |
|       82304 | Neha    |   35 | F    | Hyderabad   |
|       82306 | Ramu    |   40 | M    | Hyderabad   |
|       82308 | Aakash  |   40 | M    | Hyderabad   |
|       82402 | Aravind |   42 | M    | Hyderabad   |
|       82403 | Avinash |   42 | M    | Hyderabad   |
|       82502 | Ramesh  |   23 | M    | Hyderabad   |
|       82602 | Rajesh  |   23 | M    | Hyderabad   |
+-------------+---------+------+------+-------------+
14 rows in set (0.00 sec)

mysql> SELECT MIN(AGE) as MINAGE FROM PASSENGER2;
+--------+
| MINAGE |
+--------+
|     23 |
+--------+
1 row in set (0.03 sec)
```

3. FindnumberofticketsbookedforeachPNR_NousingGROUPBYCLAUSE

MySQL> SELECT PNRNO,SUM(No_of_SEATS) AS SUM_OF_SEATS FROM RESERVATION2 GROUP BYPNRNO;

```
mysql> SELECT * FROM RESERVATION2;
+-------+---------------------+-----------+---------+------------+--------+
| PNRNO | Journeydate         | NoofSeats | Address | CONTACTNO  | STATUS |
+-------+---------------------+-----------+---------+------------+--------+
| 10201 | 2012-02-20 10:20:25 |         5 | HYD     | 9654235242 | NULL   |
| 10202 | 2012-02-22 10:22:25 |         5 | HYD     | 9654232451 | NULL   |
| 10203 | 2012-03-22 10:30:25 |         5 | DELHI   | 9654587960 | NULL   |
| 10204 | 2013-03-22 11:30:25 |         5 | CHENNAI | 9845761254 | NULL   |
+-------+---------------------+-----------+---------+------------+--------+
4 rows in set (0.00 sec)

mysql> SELECT PNRNO,SUM(NOOFSEATS) AS SUM_OF_SEATS FROM RESERVATION2   GROUP BY
PNRNO;
+-------+--------------+
| PNRNO | SUM_OF_SEATS |
+-------+--------------+
| 10201 |            5 |
| 10202 |            5 |
| 10203 |            5 |
| 10204 |            5 |
+-------+--------------+
4 rows in set (0.00 sec)
```

4 FindthedistinctPNRNumbersthatarepresent.

MySQL> SELECT DISTINCT PNR_NO FROM RESERVATION2;

```
mysql> SELECT * FROM RESERVATION2;
+-------+---------------------+-----------+---------+------------+--------+
| PNRNO | Journeydate         | NoofSeats | Address | CONTACTNO  | STATUS |
+-------+---------------------+-----------+---------+------------+--------+
| 10201 | 2012-02-20 10:20:25 |         5 | HYD     | 9654235242 | NULL   |
| 10202 | 2012-02-22 10:22:25 |         5 | HYD     | 9654232451 | NULL   |
| 10203 | 2012-03-22 10:30:25 |         5 | DELHI   | 9654587960 | NULL   |
| 10204 | 2013-03-22 11:30:25 |         5 | CHENNAI | 9845761254 | NULL   |
+-------+---------------------+-----------+---------+------------+--------+
4 rows in set (0.00 sec)

mysql> SELECT DISTINCT PNRNO FROM RESERVATION2;
+-------+
| PNRNO |
+-------+
| 10201 |
| 10202 |
| 10203 |
| 10204 |
+-------+
4 rows in set (0.00 sec)
```

5 Mysql>selectsum(Noofseats)fromCancellation2;

```
mysql> SELECT * FROM CANCELLATION2;
+-------+---------------------+-----------+---------+------------+---------+
| PNRNO | JOURNEYDATE         | NOOFSEATS | ADDRESS | CONTACTNO  | STATUS  |
+-------+---------------------+-----------+---------+------------+---------+
| 10201 | 2012-02-20 10:20:25 |         2 | HYD     | 9654235242 | CONFIRM |
| 10202 | 2012-02-22 10:22:25 |         2 | HYD     | 9654232451 | CONFIRM |
| 10203 | 2012-03-22 10:30:25 |         2 | DELHI   | 9654587960 | CONFIRM |
+-------+---------------------+-----------+---------+------------+---------+
3 rows in set (0.00 sec)

mysql> SELECT SUM(NOOFSEATS) FROM CANCELLATION2;
+----------------+
| SUM(NOOFSEATS) |
+----------------+
|              6 |
+----------------+
1 row in set (0.00 sec)
```

6 Findthetotalnumberofcancelledseats.

MySQL> select sum(noofseats) as canceled_seats from cancellation2;

```
mysql> SELECT * FROM CANCELLATION2;
+-------+---------------------+-----------+---------+------------+---------+
| PNRNO | JOURNEYDATE         | NOOFSEATS | ADDRESS | CONTACTNO  | STATUS  |
+-------+---------------------+-----------+---------+------------+---------+
| 10201 | 2012-02-20 10:20:25 |         2 | HYD     | 9654235242 | CONFIRM |
| 10202 | 2012-02-22 10:22:25 |         2 | HYD     | 9654232451 | CONFIRM |
| 10203 | 2012-03-22 10:30:25 |         2 | DELHI   | 9654587960 | CONFIRM |
+-------+---------------------+-----------+---------+------------+---------+
3 rows in set (0.00 sec)

mysql> select sum(noofseats) as canceled_seats from  cancellation2;
+----------------+
| canceled_seats |
+----------------+
|              6 |
+----------------+
1 row in set (0.00 sec)
```

Creation and Droping of Views

mysql> create table students(sid int primary key,name varchar(15),login varchar(15), age int,gpa real); mysql> create table Enrolled(sidint,cidint,grade varchar(5),primary key(sid,cid), foreign key(sid) references students(sid));

mysql>create view BStudents(name,sid,course) AS SELECT

s.name,s.sid,E.cid from students s,enrolled E where s.sid=e.sid AND

E.grade='B';

```
mysql> create view BStudents(name,sid,course) AS SELECT s.name,s.sid,E.cid from
students s,enrolled E where s.sid=e.sid AND E.grade='B';
Query OK, 0 rows affected (0.00 sec)

mysql> select * from Bstudents;
+-------+-------+--------+
| name  | sid   | course |
+-------+-------+--------+
| jones | 53666 |      3 |
| Guldu | 53832 |      2 |
+-------+-------+--------+
2 rows in set (0.03 sec)
```

Syntax: Drop view viewname;

Mysql> Drop view Bstudents; Mysql> Drop view Goodstudents;

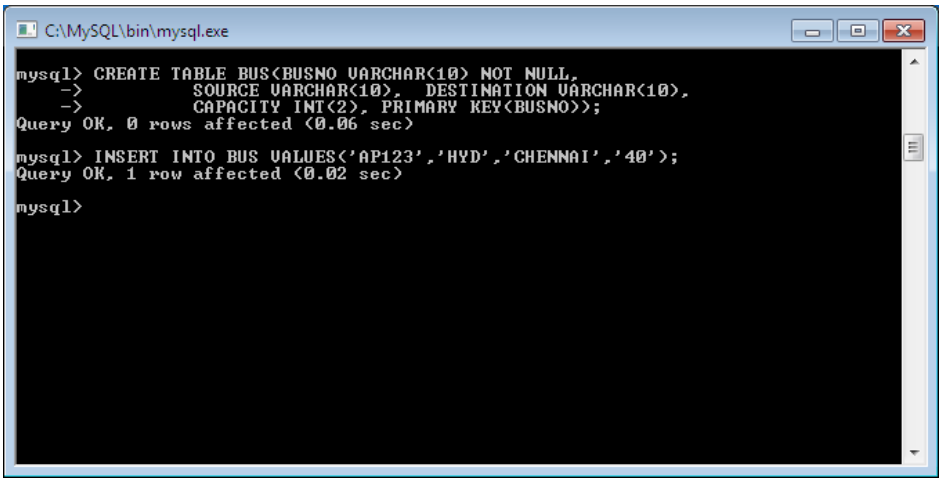
```
mysql> Drop view Bstudents;
Query OK, 0 rows affected (0.00 sec)

mysql> Drop view Goodstudents;
Query OK, 0 rows affected (0.00 sec)
```

EXPERIMENT – 8
TRIGGERS

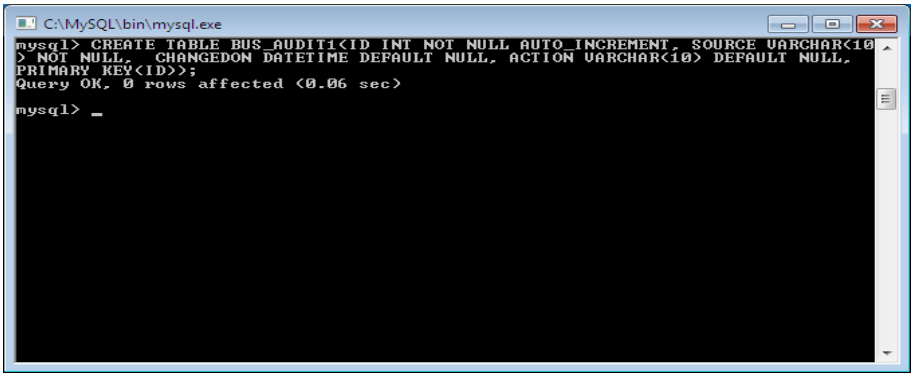
Aim: Creation of insert trigger, delete trigger and update trigger.

MySQL>CREATE TABLE BUS(BUSNO VARCHAR(10) NOT NULL, SOURCE VARCHAR(10), DESTINATION VARCHAR(10), CAPACITY INT(2), PRIMARY KEY(BUSNO));

MySQL>INSERT INTO BUS VALUES('AP123','HYD','CHENNAI','40');



CREATE TABLE BUS_AUDIT1(ID INT NOT NULL AUTO_INCREMENT, SOURCE VARCHAR(10) NOT NULL, CHANGEDON DATETIME DEFAULT NULL, ACTION VARCHAR(10) DEFAULT NULL, PRIMARY KEY(ID));

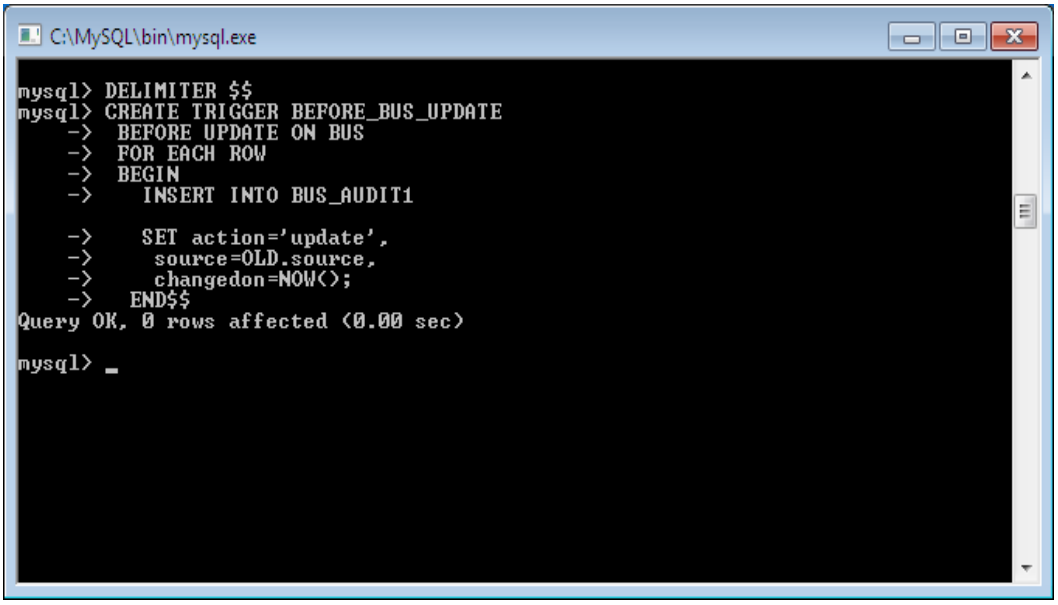


CREATE TRIGGER BEFORE_BUS_UPDATE BEFORE UPDATE ON BUS

FOR EACH ROW BEGIN

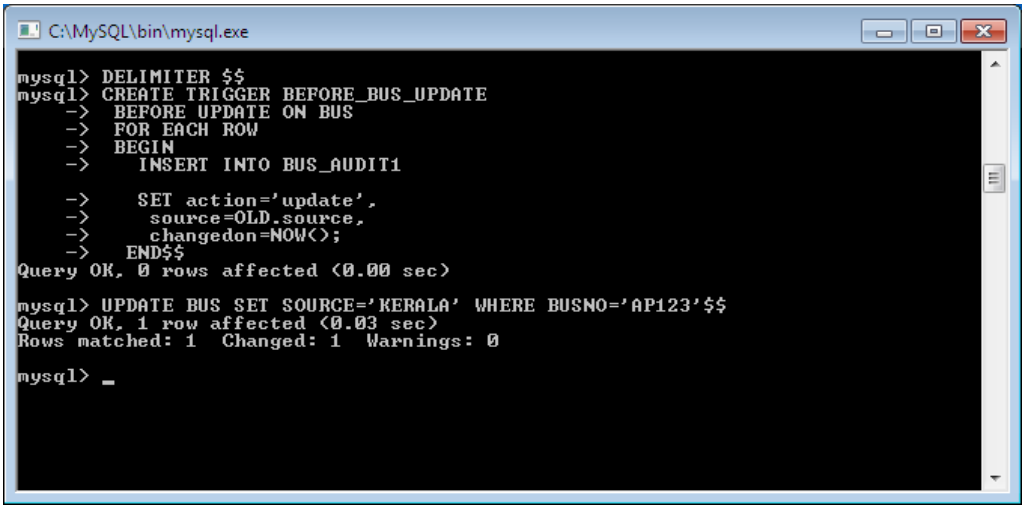
INSERT INTO BUS_AUDIT1

SET action='update', source=OLD.source, changedon=NOW(); END$$



UPDATE :

MySQL>UPDATE BUS SET SOURCE='KERALA' WHERE BUSNO='AP123'$$



SNo	Source	Changedon	Action
1	Banglore	2014:03:23 12:51:00	Insert
2	Kerela	2014:03:25:12:56:00	Update
3	Mumbai	2014:04:26:12:59:02	Delete

INSERT:

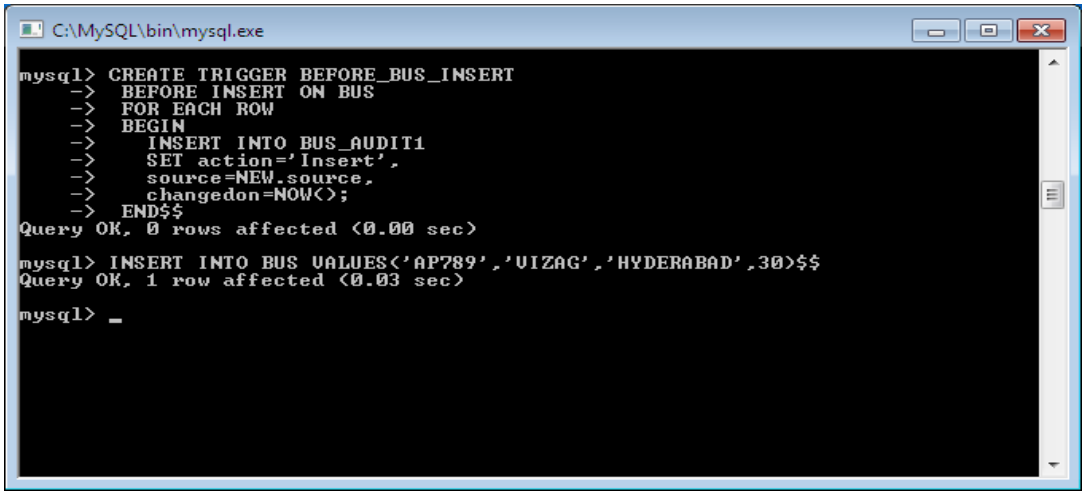
CREATE TRIGGER BEFORE_BUS_INSERT BEFORE INSERT ON BUS

FOR EACH ROW BEGIN

INSERT INTO BUS_AUDIT1

SET action='Insert', source=NEW.source, changedon=NOW(); END$$

MYSQL>INSERT INTO BUS VALUES('AP789','VIZAG','HYDERABAD',30)$$



SNo	Source	Changedon	Action
1	Banglore	2014:03:23 12:51:00	Insert
2	Kerela	2014:03:25:12:56:00	Update
3	Mumbai	2014:04:26:12:59:02	Delete

```
CREATE TRIGGER BEFORE_BUS_DELETE BEFORE DELETE ON BUS

FOR EACH ROW BEGIN

DELETE FROM BUS_AUDIT1


SET action='Insert', source=NEW.source, changedon=NOW(); END$$

DELETE FROM BUS WHERE SOURCE='HYDERABAD'$$
```

Sno	Source	Changedon	Action
1	**Banglore**	**2014:03:23 12:51:00**	**Insert**
2	**Kerela**	**2014:03:25:12:56:00**	**Update**
3	**Mumbai**	**2014:04:26:12:59:02**	**Delete**

Examples

```
CREATE TRIGGER updcheck1 BEFORE UPDATE ON passengerticket FOR EACH ROW

BEGIN

IF NEW.TicketNO> 60 THEN

SET New.TicketNo = New.TicketNo; ELSE

SET New.TicketNo = 0; END IF;

END;
```

```
mysql> select * from passengerticket;$$
+------------+----------+
| passportid | TicketNo |
+------------+----------+
| 145        |      100 |
| 278        |      200 |
| 6789       |      300 |
| 82302      |      400 |
| 82403      |      500 |
| 82502      |      600 |
+------------+----------+
6 rows in set (0.00 sec)

mysql> desc passengerticket;$$
+------------+-------------+------+-----+---------+-------+
| Field      | Type        | Null | Key | Default | Extra |
+------------+-------------+------+-----+---------+-------+
| passportid | varchar(15) | NO   | PRI |         |       |
| TicketNo   | int(11)     | YES  |     | NULL    |       |
+------------+-------------+------+-----+---------+-------+
2 rows in set (0.00 sec)
```

```
mysql> CREATE TRIGGER updcheck BEFORE UPDATE ON passengerticket
    -> FOR EACH ROW
    -> BEGIN
    -> IF NEW.TicketNO > 60 THEN
    -> SET New.TicketNo = TicketNo;
    -> ELSE
    -> SET New.TicketNo = 0;
    -> END IF;
    -> END;
    -> $$
Query OK, 0 rows affected (0.00 sec)

mysql> update passengerticket set TicketNo=TicketNo-50 where passportid=145;$$
Query OK, 1 row affected (0.03 sec)
Rows matched: 1  Changed: 1  Warnings: 0

mysql> select * from passengerticket;$$
+------------+----------+
| passportid | TicketNo |
+------------+----------+
| 145        |        0 |
| 278        |      200 |
| 6789       |      300 |
| 82302      |      400 |
| 82403      |      500 |
| 82502      |      600 |
+------------+----------+
6 rows in set (0.00 sec)
```

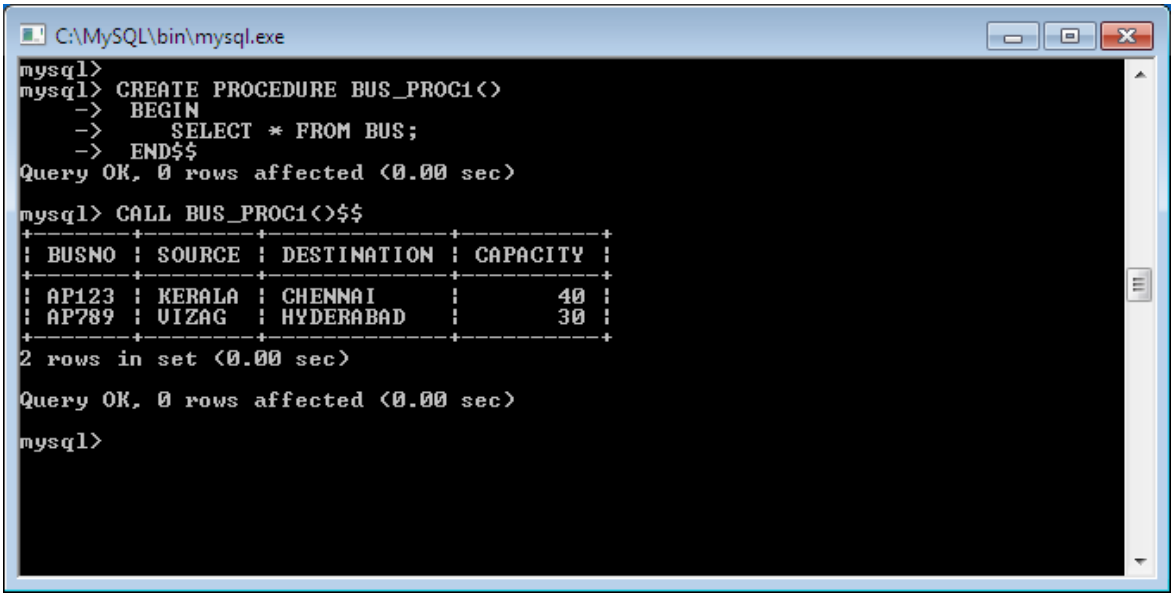
```
mysql> select * from passengerticket;$$
+------------+----------+
| passportid | TicketNo |
+------------+----------+
| 145        |        0 |
| 278        |      200 |
| 6789       |      300 |
| 82302      |      400 |
| 82403      |      500 |
| 82502      |      600 |
+------------+----------+
6 rows in set (0.00 sec)

mysql> CREATE TRIGGER updcheck BEFORE UPDATE ON passengerticket
    -> FOR EACH ROW
    -> BEGIN
    -> IF NEW.TicketNO>60 THEN
    -> SET New.TicketNo=New.TicketNo;
    -> ELSE
    -> SET New.TicketNo=0;
    -> END IF;
    -> END;
    -> $$
Query OK, 0 rows affected (0.00 sec)

mysql> update passengerticket set TicketNo=TicketNo+80 where passportid=145;$$
Query OK, 1 row affected (0.03 sec)
Rows matched: 1  Changed: 1  Warnings: 0

mysql> select * from passengerticket;$$
+------------+----------+
| passportid | TicketNo |
+------------+----------+
| 145        |       80 |
| 278        |      200 |
| 6789       |      300 |
| 82302      |      400 |
| 82403      |      500 |
| 82502      |      600 |
+------------+----------+
6 rows in set (0.00 sec)
```

EXPERIMENT –9
PROCEDURES

Aim: Creation of stored Procedures and Execution of Procedures and Modification of Procedures.

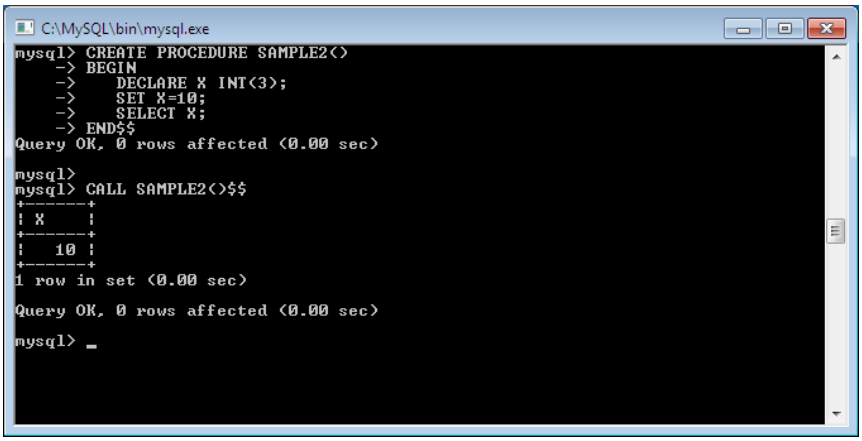
Ex1:

```
CREATE PROCEDURE BUS_PROC1() BEGIN

SELECT * FROM BUS;

END$$

CALL BUS_PROC1()$$
```



Ex2:

```
CREATE PROCEDURE SAMPLE2() BEGIN
DECLARE X INT(3); SET X=10;
SELECT X;

END$$
Mysql> CALL SAMPLE2()$$
```

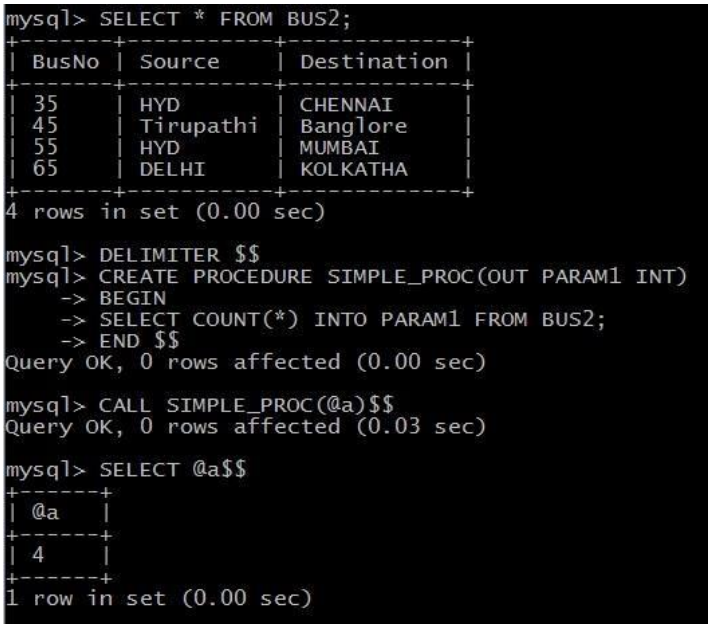


Ex3: CREATE PROCEDURE SIMPLE_PROC(OUT PARAM1 INT) BEGIN

SELECT COUNT(*) INTO PARAM1 FROM BUS;

END$$

Mysql> CALL SIMPLE_PROC(@a)$$ Mysql> select @a;



EXPERIMENT – 10

Cursors

Aim: Declare a cursor that defines a result set. Open the cursor to establish the result set. Fetch the data into local variables as needed from the cursor, one row at a time. Close the cursor when done.

Cursors

In MySQL, a cursor allows row-by-row processing of the result sets. A cursor is used for the result set and returned from a query. By using a cursor, you can iterate, or by step through the results of a query and perform certain operations on each row. The cursor allows you to iterate through the result set and then perform the additional processing only on the rows that requireit.

In a cursor contains the data in a loop. Cursors may be different from SQL commands that operate on all the rows in the returned by a query at onetime.

There are some steps we have to follow, given below :

- □ Declare acursor
- □ Open a cursorstatement
- □ Fetch thecursor
- □ Close thecursor

1 . Declaration of Cursor : To declare a cursor you must use the DECLARE statement. With the help of the variables, conditions and handlers we need to declare a cursor before we can use it. first of all we will give the cursor a name, this is how we will refer to it later in the procedure. We can have more than one cursor in a single procedure so its necessary to give it a name that will in some way tell us what its doing. We then need to specify the select statement we want to associate with the cursor. The SQL statement can be any valid SQL statement and it is possible to use a dynamic where clause using variable or parameters as we have seen previously.

Syntax : DECLARE *cursor_name*CURSOR FOR *select_statement;*

2 . Open a cursor statement : For open a cursor we must use the open statement.Ifwewanttofetchrowsfromityoumustopenthecursor.

Syntax : OPEN cursor_name;

3 . Cursor fetch statement : When we have to retrieve the next row from the cursor and movethecursor to next row then you need to fetch thecursor.

Synatx : FETCH cursor_name INTOvar_name;

If any row exists, then the above statement fetches the next row and cursor pointer moves ahead to the next row.

4 . Cursor close statement : By this statement closed the opencursor.

Syntax: CLOSE_name;

By this statement we can close the previously opened cursor. If it is not closed explicitly then a cursor is closed at the end of compound statement in which that was declared.

Delimiter $$

```
Create procedure p1(in_customer_id int) begin
declare v_id int;
declare v_name varchar(20); declare v_finished integer default 0;
declare c1 cursor for select sid,sname from students where sid=in_customer_id; declare
continue handler for NOT FOUND set v_finished=1;
open c1; std:LOOP
fetch c1 into v_id,v_name; if v_finished=1 then
leave std; end if;
select concat(v_id,v_name); end LOOP std;
close c1; end;
```

```
mysql> select * from students;
+------+---------+------+--------+
| sid  | sname   | age  | marks  |
+------+---------+------+--------+
| 1    | ravi    |   15 |     25 |
| 2    | ramu    |   20 |     30 |
| 2    | rahul   |   18 |     26 |
| 5    | kiran   |   19 |     28 |
| 6    | varun   |   21 |     32 |
| 8    | ramesh  |   22 |     33 |
| 8    | xyz     |   10 |     20 |
+------+---------+------+--------+
7 rows in set (0.00 sec)
```

```
mysql> delimiter $$
mysql> Create procedure p1(in_customer_id int)
    -> begin
    -> declare v_id int;
    -> declare v_name varchar(20);
    -> declare v_finished integer  default 0;
    -> declare c1 cursor for select sid,sname from students where sid=in_custome
r_id;
    -> declare continue handler for NOT FOUND set v_finished=1;
    -> open c1;
    -> std:LOOP
    -> fetch c1 into v_id,v_name;
    -> if v_finished=1 then
    -> leave std;
    -> end if;
    -> select concat(v_id,v_name);
    -> end LOOP std;
    -> close c1;
    -> end;$$
Query OK, 0 rows affected (0.01 sec)
```

A database is a logically coherent collection of data with some inherent meaning, representing some aspect of real world and which is designed, built and populated with data for a specific purpose.

1. What is DBMS?

It is a collection of programs that enables user to create and maintain a database. In other words it is general-purpose software that provides the users with the processes of defining, constructing and manipulating the database for various applications.

2. What is a Databasesystem?

The database and DBMS software together is called as Database system.

3. Advantages ofDBMS?

- Redundancy iscontrolled.
- Unauthorized access isrestricted.
- Providing multiple userinterfaces.
- Enforcing integrityconstraints.
- Providing backup andrecovery.

4. Disadvantage in File ProcessingSystem?

- Data redundancy &inconsistency.
- Difficult in accessingdata.
- Dataisolation.
- Dataintegrity.
- Concurrent access is notpossible.
- SecurityProblems.

5. Describe the three levels of dataabstraction?

Three levels of abstraction:

Physical level: The lowest level of abstraction describes how data are stored.

Logical level: The next higher level of abstraction, describes what data are stored in database and what relationship among those data.

View level: The highest level of abstraction describes only part of entire database.

6. Define the "integrityrules"

There are two Integrity rules.

Entity Integrity: States that Primary key cannot have NULL value

Referential Integrity: States that Foreign Key can be either a NULL value or should be Primary Key value of other relation.

7. What is extension andintension?

Extension: It is the number of tuples present in a table at any instance. This is time dependent.

Intension: It is a constant value that gives the name, structure of table and the constraints laid on it.

8. What is DataIndependence?

Data independencemeans that "The application is independent of the storage structure and access strategy of data". In other words, the ability to modify the schema definition in one level should not affect the schema definition in the next higher level.

Two types of Data Independence:

Physical Data Independence: Modification in physical level should not affect the logical level.

Logical Data Independence: Modification in logical level should affect the view level.

9. What is a view? How it is related to dataindependence?

A view may be thought of as a virtual table, that is, a table that does not really exist in its own right but is instead derived from one or more underlying base table. In other words, there is no stored file that direct represents the view instead a definition of view is stored in data dictionary. Growth and restructuring of base tables is not reflected in views. Thus the

View can insulate users from the effects of restructuring and growth in the database. Hence accounts for logical data independence.

10. What is DataModel?

A collection of conceptual tools for describing data, data relationships data semantics and constraints.

11. What is E-Rmodel?

This data model is based on real world that consists of basic objects called entities and of relationship among these objects. Entities are described in a database by a set of attributes.

12. What is Object Orientedmodel?

This model is based on collection of objects. An object contains values stored in instance variables within the object. An object also contains bodies of code that operate on the object. These bodies of code are called methods. Objects that contain same types of values and the same methods are grouped together into classes.

13. What is anEntity?

It is a 'thing' in the real world with an independent existence.

14. What is an Entitytype?

It is a collection (set) of entities that have same attributes.

15. What is an Entityset?

It is a collection of all entities of particular entity type in the database.

16. What is an Extension of entitytype?

The collections of entities of a particular entity type are grouped together into an entity set.

17. What is Weak Entityset?

An entity set may not have sufficient attributes to form a primary key, and its primary key compromises of its partial key and primary key of its parent entity, then it is said to be Weak Entity set.

18. What is anattribute?

It is a particular property, which describes the entity.

19. What is aRelation?

A relation is defined as a set of tuples.

20. What is degree of aRelation?

It is the number of attribute of its relation schema.

21. What isRelationship?

It is an association among two or more entities.

22. What is Relationshipset?

The collection (or set) of similar relationships.

23. What is Relationshiptype?

Relationship type defines a set of associations or a relationship set among a given set of entity types.

24. What is degree of Relationship type?

It is the number of entity type participating.

25. What is DDL (Data DefinitionLanguage)?

A data base schema is specifies by a set of definitions expressed by a special language called DDL.

26. What is VDL (View DefinitionLanguage)?

It specifies user views and their mappings to the conceptual schema.

27. What is DML (Data ManipulationLanguage)?

This language that enable user to access or manipulate data as organized by appropriate data model.

28. What is DMLCompiler?

It translates DML statements in a query language into low-level instruction that the query evaluation engine canunderstand.

29. What is Query evaluationengine?

It executes low-level instruction generated by compiler.

30. What is DDLInterpreter?

It interprets DDL statements and records them in tables containing metadata.

31. What is aquery?

A query with respect to DBMS relates to user commands that are used to interact with a data base. The query language can be classified into data definition language and data manipulation language.

32. What do you mean by Correlated subquery?

A correlated sub query can be easily identified if it contains any references to the parent sub querycolumns in its WHERE clause. Columns from the sub query cannot be referenced anywhere else in the parent query.

33. Are the resulting relations of PRODUCT and JOIN operation the same?

No.

PRODUCT: Concatenation of every row in one relation with every row in another.

JOIN: Concatenation of rows from one relation and related rows from another.

34. What is databaseTrigger?

A database trigger is a PL/SQL block that can defined to automatically execute for insert, update, and delete statements against a table. The trigger can be defined to execute once for the entire statement or once for every row that is inserted, updated, or deleted. For any one table, there are twelve events for which you can define database triggers. A database trigger can call database procedures that are also written inPL/SQL.

35. What are stored-procedures? What are the advantages of usingthem?

Stored procedures are database objects that perform a user defined operation. A stored procedure can have a set of compound SQL statements. A stored procedure executes the SQL commands and returns the result tothe client. Stored procedures are used to reduce networktraffic.

36. Define super key and give example to illustrate the superkey?

Set of one or more attributes taken collectively, allowing to identify uniquely an entity in the entity set.Eg1. {SSN} and {SSN, Cust_name} of customer table are super keys.Eg2. {Branch name}and
{Branch name, Branch city} of Branch table re super keys.

37. Define candidate key and give example to illustrate the candidatekey?

Super keys with no proper subset are called the candidate keys. Otherwise it is called minimal super key. Candidate key is nothing but the primary key used in SQL. Eg1. {SSN} is the candidate keyfor the super keys {SSN} and {SSN, Cust_name} of customer table.Eg2. {Branch name} is the candidate key for the super keys {Branch name} and {Branch name, Branch city} of Branch table.

38. What is Primarykey?

A key chosen to act as the means by which to identify tuples in a relation.

39. What is foreignkey?

A foreign key of relation R is a set of its attributes intended to be used (by each tuple in R) for identifying/referring to a tuples in some relation S. (R is called the referencing relation and S the referenced relation.) For this to make sense, the set of attributes of R forming the foreign key should "correspond to" some superkeyof S. Indeed, by definition we require this superkeyto be the primary key of S.

40. What is a Cursor?

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 activeset.

41. What is Functional Dependency?

A Functional dependency is denoted by X-> Y between two sets of attributes X and Y that are subsets of R specifies a constraint on the possible tuples that can form a relation state r of R. The constraint is for any two tuples t1 and t2 in r if t1[X] = t2[X] then they have t1[Y] = t2[Y].

42. What is 1 NF (Normal Form)?

The domain of attribute must include only atomic (simple, indivisible) values.

43. What is Fully Functional dependency?

It is based on concept of full functional dependency. A functional dependency X-> Y is fully functional dependency if removal of any attribute A from X means that the dependency does not hold any more.

44. What is2NF?

A relation schema R is in 2NF if it is in 1NF and every non-prime attribute A in R is fully functionally dependent on primary key.

45. What is3NF?

A relation schema R is in 3NF if it is in 2NF and for every FD X A either of the following is true X is a Super-key of R.

46. What is BCNF (Boyce-Codd NormalForm)?

A relation schema R is in BCNF if it is in 3NF and satisfies an additional constraint that for every FD X A, X must be a candidate key.

47. What is 4NF?

A relation schema R is said to be in 4NF if for every multivalued dependency X Y that holds over R, one of following is true X is subset or equal to (or) XY = R. X is a superkey.

48. What is 5NF?

A Relation schema R is said to be 5NF if for every join dependency {R1, R2... Rn} that holds R, one the following is true

i) Ri = R for somei.

ii) The join dependency is implied by the set of FD, over R in which the left side is key of R.

49.What is dependencypreservation?

Dependency Preservation Property enables us to enforce a constraint on the original relation from corresponding instances in the smaller relations.

50. What is Lossless joinproperty?

Lossless join property enables us to find any instance of the original relation from corresponding instances in the Smaller relations