BALAJI INSTITUTE OF TECHNOLOGY & SCIENCES

DEPARTMENT OF COMPUTER SCIENCE & ENGINEERING

DBMS LAB MANUAL

ISO 9001:2015 Certified Institution

Estd. : 2001

Balaji Institute of Technology & Science

Laknepally, NARSAMPET, Warangal (Rural) – 506331

Accredited by NBA (UG – CE, ME, ECE & CSE) & NAAC A+ Grade

(Affiliated to JNTUH, Hyderabad and Approved by the AICTE, New Delhi)

BITS NARSAMPET

www.bitswgl.ac.in, email: principal@bitswgl.ac.in :: Ph. 98660 50044, Fax 08718-230521

DEPARTMENT OF COMPUTER SCIENCE AND ENGINEERING

LAB MANUAL FOR THE ACADEMIC YEAR: 2023-24

COURSE	**:**	**B. TECH**
YEAR	**:**	**II**
SEMESTER	**:**	**I - SEM**
DEPARTMENT	**:**	**CSE**
SUBJECT	**:**	**DATABASE MANAGEMENT SYSTEMS LAB**

FACULTY **HOD**

CS407PC: DATABASE MANAGEMENT SYSTEMS LAB

Co-requisites: Co-requisite of course "Database Management Systems"

Course Objectives:

- Introduce ER data model, database design and normalization
- Learn SQL basics for data definition and data manipulation

Course Outcomes:

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

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

INTRODUCTION TO DBMS

A Database Management System (DBMS) is software designed to store, retrieve, define, and manage data in a database.

DBMS allows users to create their own databases as per their requirement. The term “DBMS” includes the user of the database and other application programs. It provides an interface between the data and the software application.

- 1960 - Charles Bachman designed first DBMS system
- 1970 - Codd introduced IBM'S Information Management System (IMS)
- 1976- Peter Chen coined and defined the Entity-relationship model also know as the ER model
- 1980 - Relational Model becomes a widely accepted database component
- 1985- Object-oriented DBMS develops.
- 1990s- Incorporation of object-orientation in relational DBMS.
- 1991- Microsoft ships MS access, a personal DBMS and that displaces all other personal DBMS products.
- 1995: First Internet database applications
- 1997: XML applied to database processing. Many vendors begin to integrate XML into DBMS products.

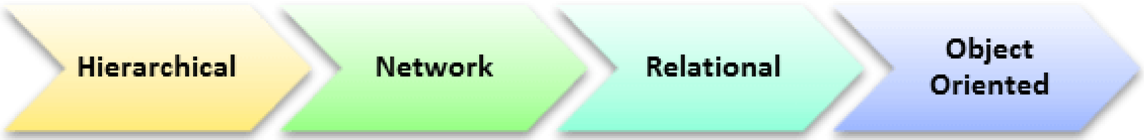
Characteristics of Database Management System

- Provides security and removes redundancy
- Self-describing nature of a database system
- Insulation between programs and data abstraction
- Support of multiple views of the data
- Sharing of data and multiuser transaction processing
- DBMS allows entities and relations among them to form tables.
- It follows the ACID concept (Atomicity, Consistency, Isolation, and Durability).
- DBMS supports multi-user environment that allows users to access and manipulate data in parallel.

list of some popular DBMS system:

- MySQL
- Microsoft Access
- Oracle
- PostgreSQL
- Microsoft SQL Server etc.
- IBM DB2

Types of DBMS



Four Types of DBMS systems are:

- Hierarchical database
- Network database
- Relational database
- Object-Oriented database

Advantages of DBMS

- DBMS offers a variety of techniques to store & retrieve data
- DBMS serves as an efficient handler to balance the needs of multiple applications using the same data
- Uniform administration procedures for data
- Application programmers never exposed to details of data representation and storage.
- A DBMS uses various powerful functions to store and retrieve data efficiently.
- Offers Data Integrity and Security
- The DBMS implies integrity constraints to get a high level of protection against prohibited access to data.
- A DBMS schedules concurrent access to the data in such a manner that only one user can access the same data at a time
- Reduced Application Development Time

Disadvantage of DBMS

DBMS may offer plenty of advantages but, it has certain flaws-

- Cost of Hardware and Software of a DBMS is quite high which increases the budget of your organization.
- Most database management systems are often complex systems, so the training for users to use the DBMS is required.
- In some organizations, all data is integrated into a single database which can be damaged because of electric failure or database is corrupted on the storage media
- Use of the same program at a time by many users sometimes lead to the loss of some data.
- DBMS can't perform sophisticated calculation**s**

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

Analyze the problem carefully and come up with entities in it. Identify what data has to bepersisted in the database. This contains the entities, attributes etc.

Identify the primary keys for all the entities. Identify the other keys like candidate keys, partialkeys, if any.

Definitions:

Entity: the object in the **ER** Model represents is an entity which is thing in the real world with anindependent existence.

ER-Model:

Describes data as entities, relationships and attributes .The ER-Model is important preliminary for its role in database design. ER Model is usually shown pictorially using entity relationship diagrams.

Attributes:

The properties that characterize an entity set are called its attributes. An attribute is referred to bythe terms data items, data element, data field item.

Candidate key:

It can be defined as minimal super key or irreducible super key. In other words an attribute or combination of attributes that identifies the record uniquely but none of its proper subsets can identify the record uniquely.

Candidate key:

It can be defined as minimal super key or irreducible super key. In other words an attribute or combination of attributes that identifies the record uniquely but none of its proper subsets can identify the record uniquely.

Partial key:

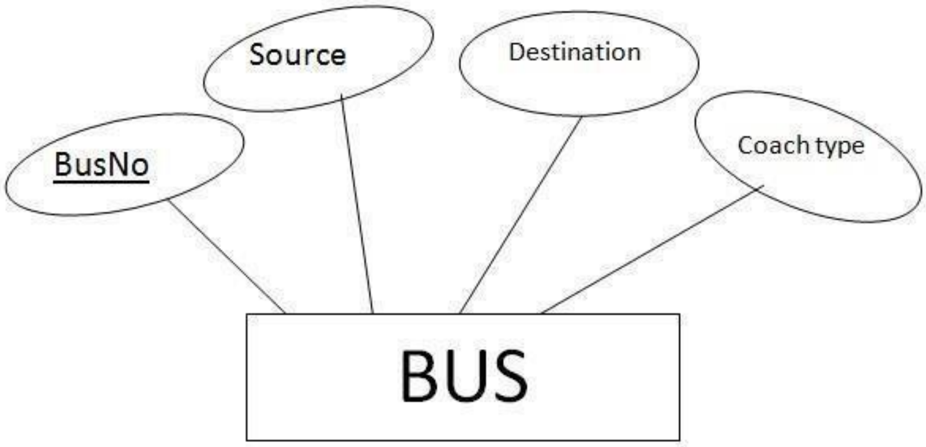
A weak entity type normally has a partial key which is the set of attributes that can uniquelyidentify weak entity that are related to the same owner entity.

Bus

- BusNo
- Source
- Destination
- CoachType

SCHEMA

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

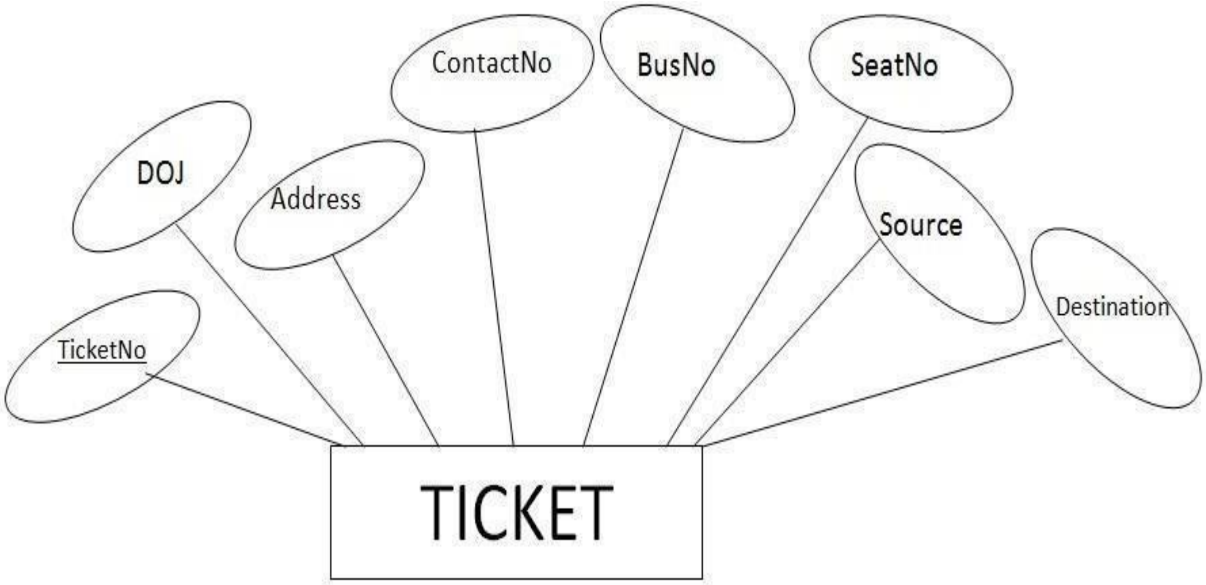


Ticket

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

SCHEMA

Ticket (<u>TicketNo:</u> string, DOJ: date, Address: string, ContactNo : string, BusNo:String
SeatNo : Integer, Source: String, Destination: String)

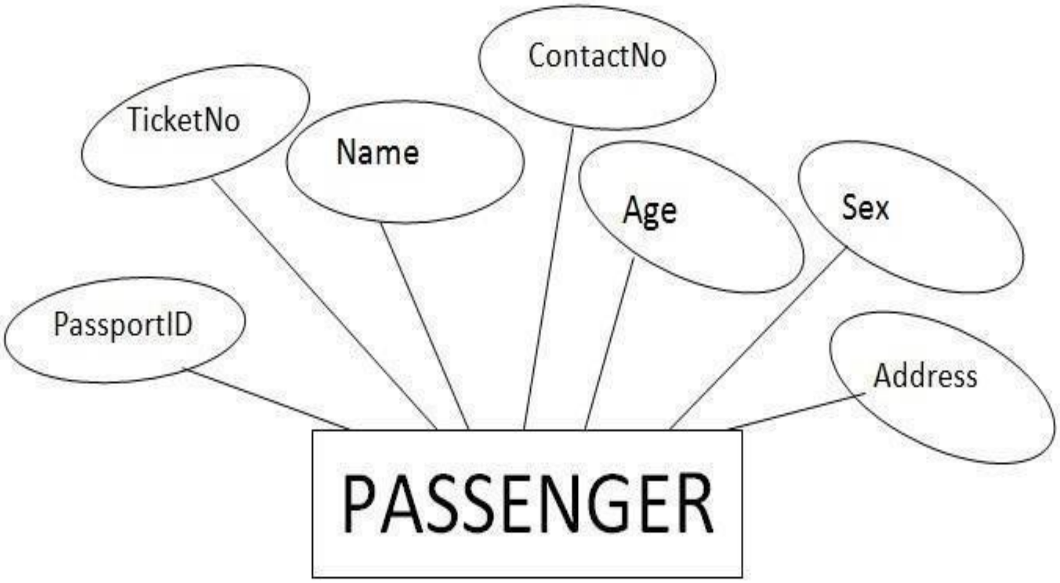


Passenger

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

SCHEMA

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



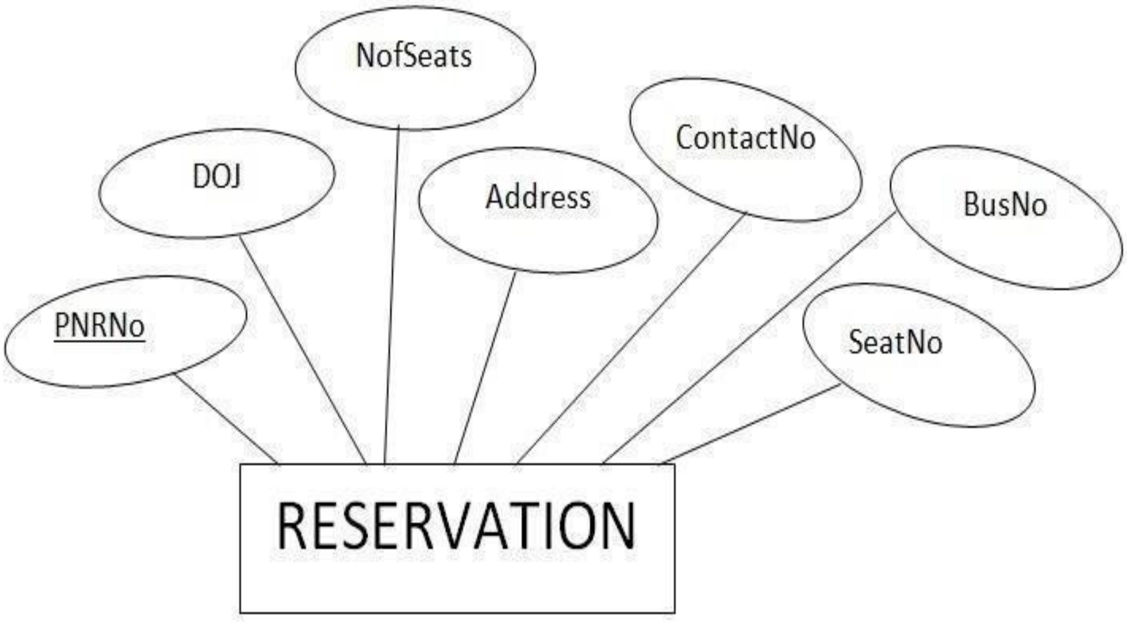
Reservation

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

SCHEMA

Reservation(PNRNo: String, DOJ: Date, NoofSeats: integer , Address: String ,ContactNo: String, ,

BusNo: String,SeatNo:Integer)

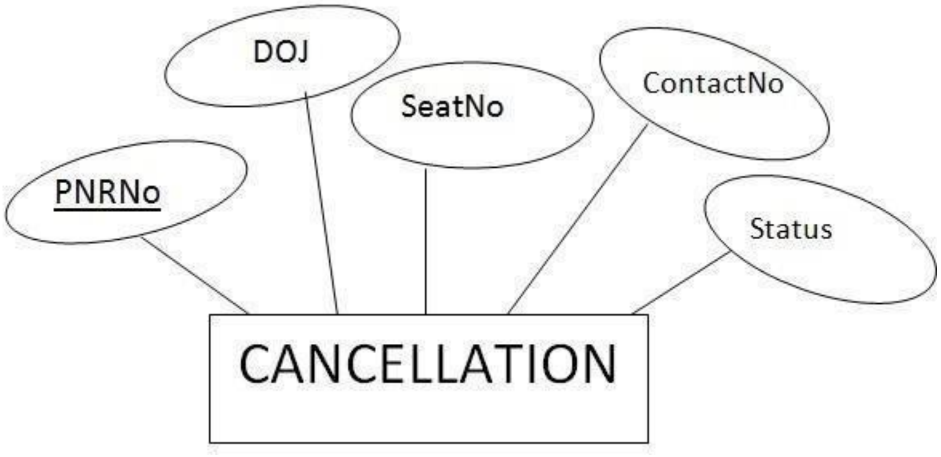


Cancellation

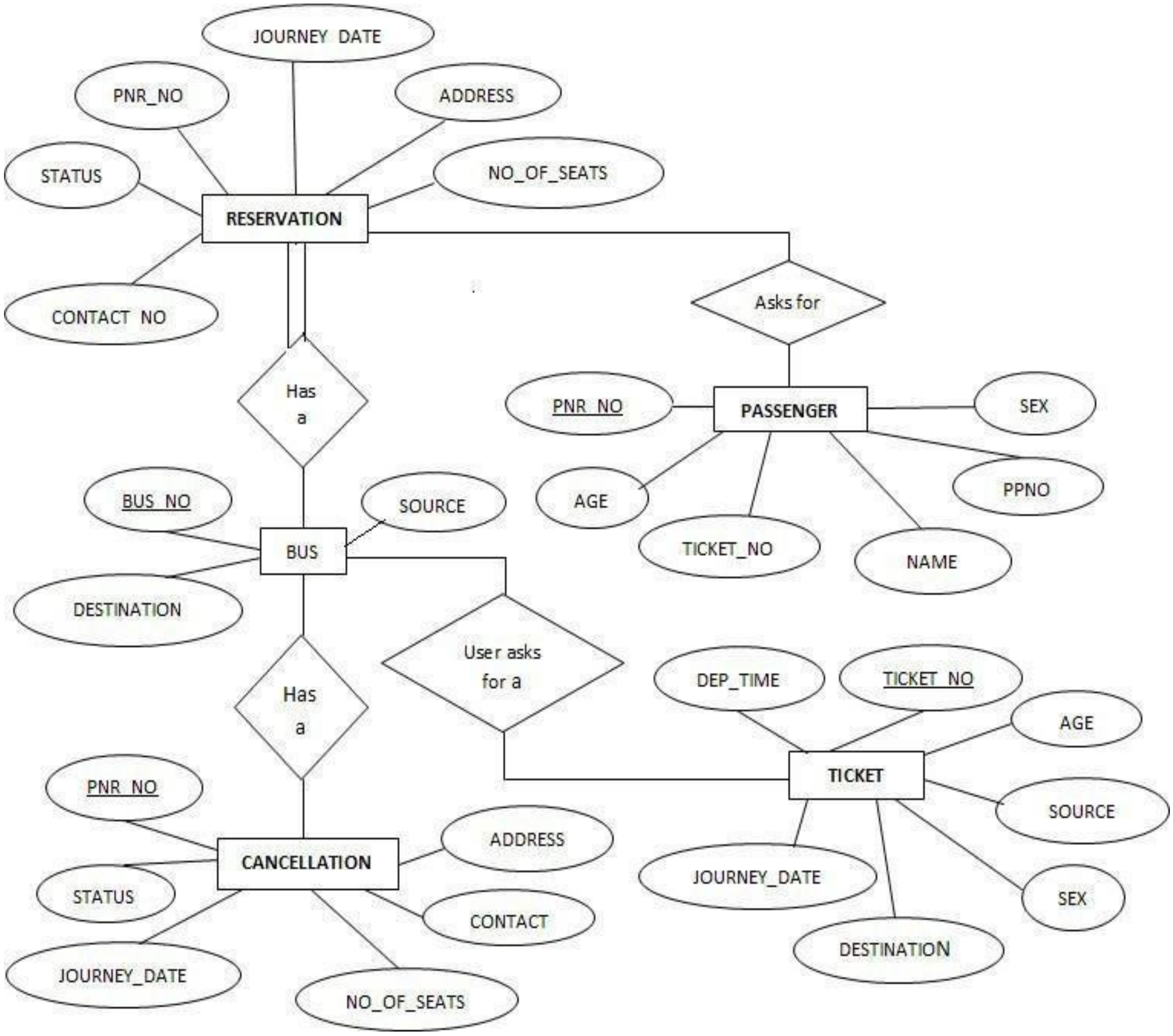
- PNRNo
- DOJ
- SeatNo
- ContactNo
- Status

SCHEMA

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



CONCEPT DESIGN WITH E-R MODEL



EXPERIMENT – 2
RELATIONAL MODEL

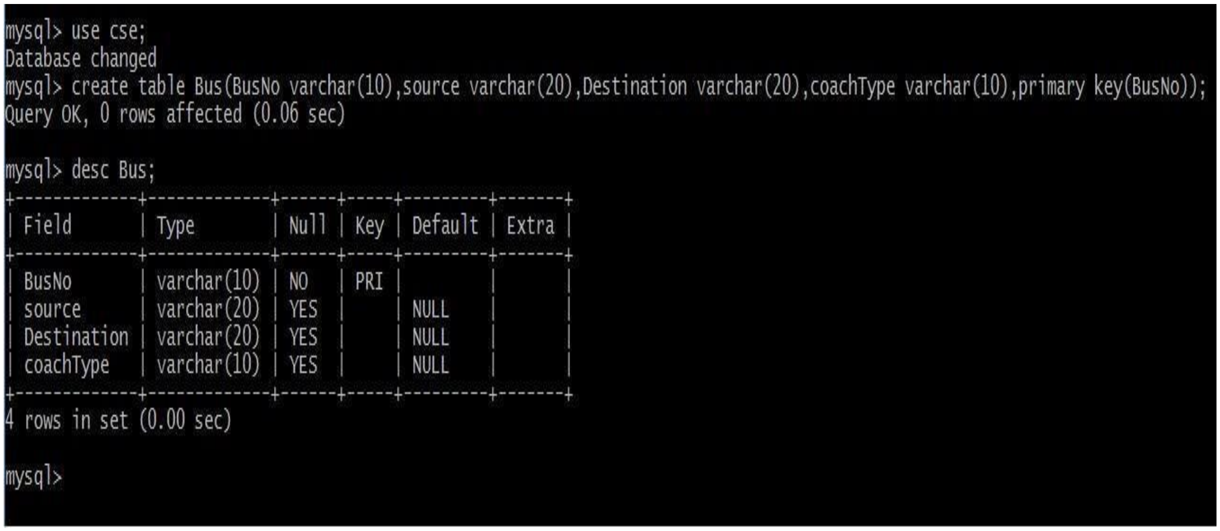
AIM: To Represent all the entities (Strong, Weak) in tabular fashion. Represent relationships in a tabular fashion.

1. **Bus:** Bus(BusNo: String, Source: String, Destination: String, CoachType: String)

ColumnName	Datatype	Constraints	Type of Attributes
BusNo	Varchar(10)	Primarykey	Single-value
Source	Varchar(20)		Single-value
Destination	Varchar(20)		Simple
CoachType	Varchar(10)		Simple

Mysql>create table Bus(BusNo varchar(10),source varchar(20),Destination varchar(20),coachType varchar(10),primary key(BusNo));

Mysql>desc Bus;



```
mysql> use cse;
Database changed
mysql> create table Bus(BusNo varchar(10),source varchar(20),Destination varchar(20),coachType varchar(10),primary key(BusNo));
Query OK, 0 rows affected (0.06 sec)

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

mysql>
```

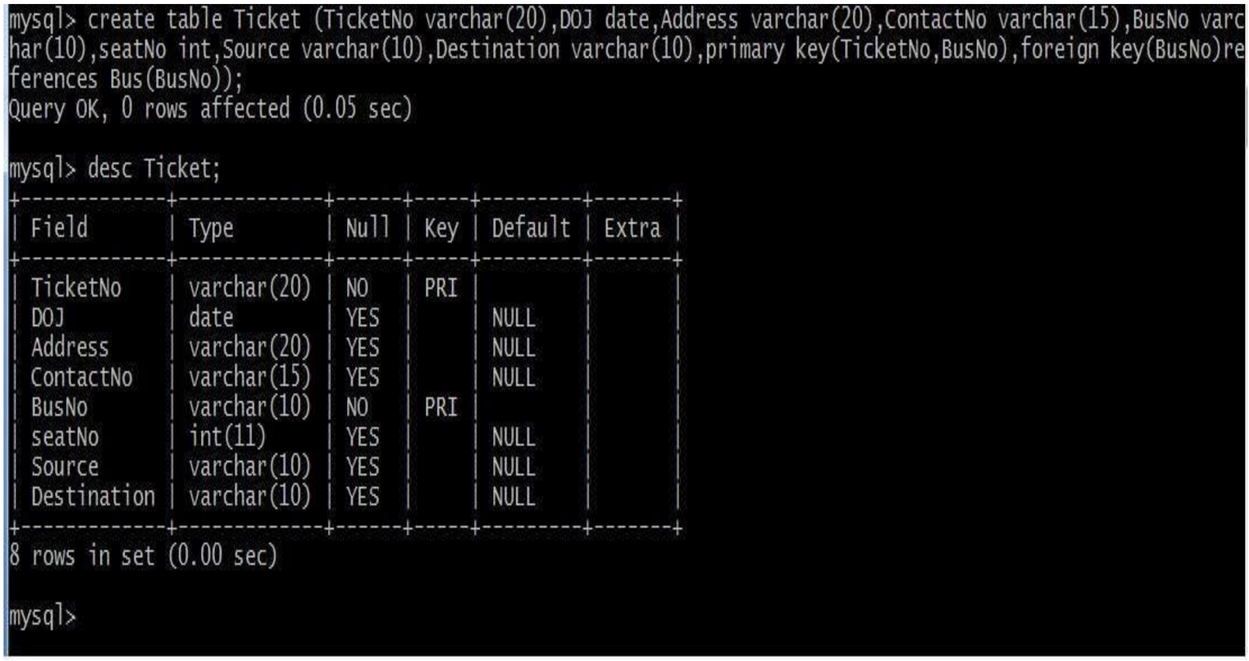
Ticket:

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

ColumnName	Datatype	Constraints	Type of Attributes
TicketNo	Varchar(20)	Primary Key	Single-valued
DOJ	Date		Single-valued
Address	Varchar(20)		Composite
ContactNo	Integer		Multi-valued
BusNo	Varchar(10)	Foreign Key	Single-valued
SeatNo	Integer		Simple
Source	Varchar(10)		Simple
Destination	Varchar(10)		Simple

Mysql> create table ticket(ticketno varchar(20), doj date,address varchar(20),contactno int, busno varchar(20),seatno int,source varchar(10),destination varchar(10),primary key(ticketno,busno) foreign key(busno)references bus(busno);

Mysql>desc Ticket;



```
mysql> create table Ticket (TicketNo varchar(20),DOJ date,Address varchar(20),ContactNo varchar(15),BusNo varc
har(10),seatNo int,Source varchar(10),Destination varchar(10),primary key(TicketNo,BusNo),foreign key(BusNo)re
ferences Bus(BusNo));
Query OK, 0 rows affected (0.05 sec)

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(15) | 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)

mysql>
```

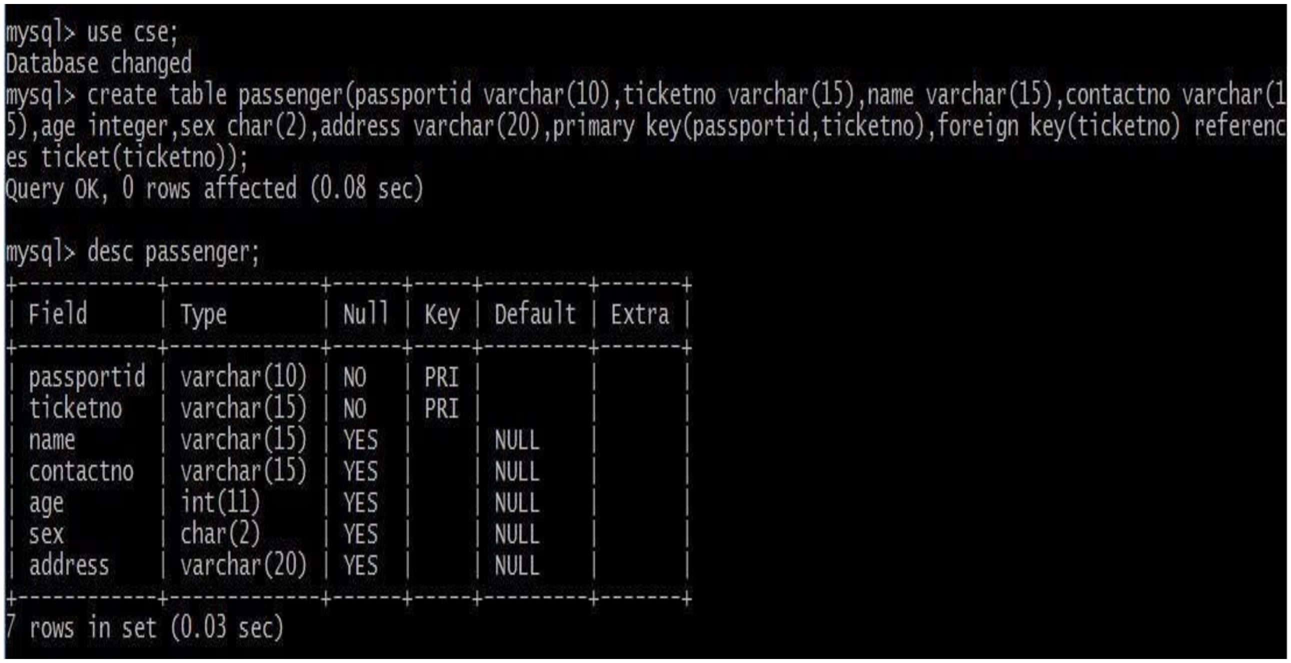
Passenger:

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

ColumnName	**Datatype**	**Constraints**	**Type of Attributes**
PassportID	Varchar(15)	Primary Key	Single-valued
TicketNo	Varchar(20)	Foreign Key	Single-valued
Name	Varchar(20)		Composite
ContactNo	Varchar(20)		Multi-valued
Age	Integer		Single-valued
Sex	character		Simple
Address	Varchar(20)		Composite

Mysql> Create table passenger(passportID varchar(15) ,TicketNo varchar(15),Name varchar(15),ContactNo varchar(20),Age integer, sex char(2),address varchar(20), primary key(passportID,TicketNo),foreign key(TicketNo) references Ticket(TicketNo));

Mysql> desc passenger;



```
mysql> use cse;
Database changed
mysql> create table passenger(passportid varchar(10),ticketno varchar(15),name varchar(15),contactno varchar(1
5),age integer,sex char(2),address varchar(20),primary key(passportid,ticketno),foreign key(ticketno) referenc
es ticket(ticketno));
Query OK, 0 rows affected (0.08 sec)

mysql> desc passenger;
+------------+-------------+------+-----+---------+-------+
| Field      | Type        | Null | Key | Default | Extra |
+------------+-------------+------+-----+---------+-------+
| passportid | varchar(10) | NO   | PRI |         |       |
| ticketno   | varchar(15) | 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 (0.03 sec)
```

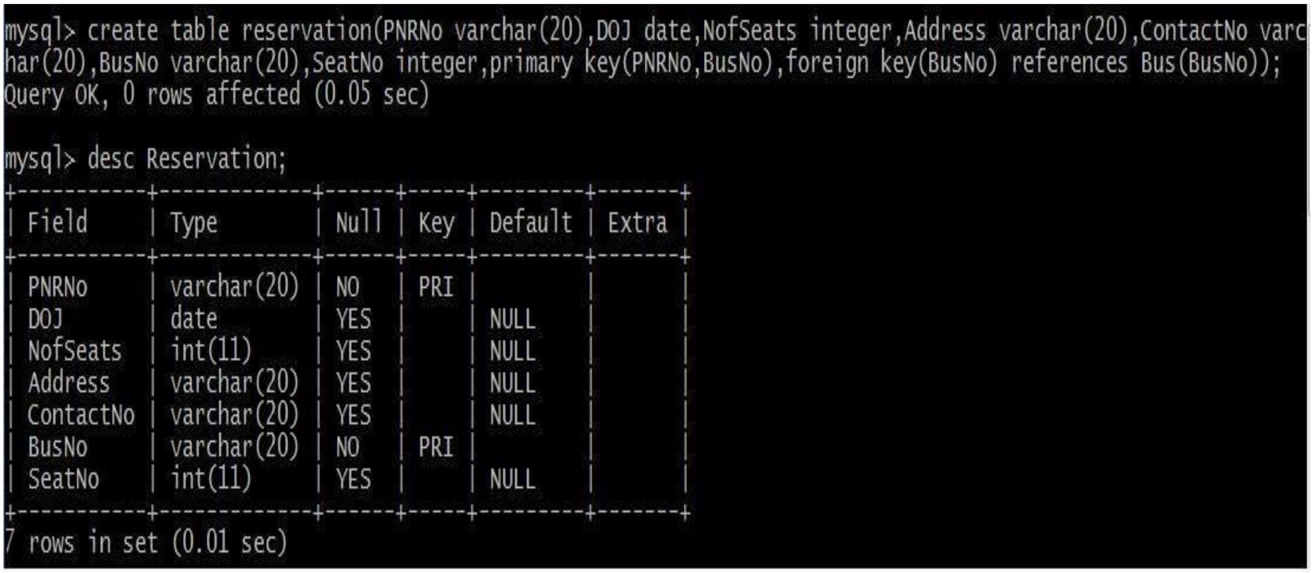
Reservation:

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

ColumnName	Datatype	Constraints	Type of Attributes
PNRNo	Varchar(20)	Primary Key	Single-valued
DOJ	date		Single-valued
No_of_Seats	Integer		Simple
Address	Varchar(20)		Composite
ContactNo	Varchar(10)		Multi-valued
BusNo	Varchar(10)	ForeignKey	Single-valued
SeatNo	Integer		Simple

```
Mysql> Create table Resevation(PNRNo varchar(20),DOJ date,NoofSeates integer,Address varchar(20),ContactNovarchar(20),BusNo varchar(20),SeatNo integer, primary key(PNRNo,BusNo),foreign key(BusNo) references Bus(BusNo));
```

```
Mysql> desc  reservation;
```



```
mysql> create table reservation(PNRNo varchar(20),DOJ date,NofSeats integer,Address varchar(20),ContactNo varc
har(20),BusNo varchar(20),SeatNo integer,primary key(PNRNo,BusNo),foreign key(BusNo) references Bus(BusNo));
Query OK, 0 rows affected (0.05 sec)

mysql> desc Reservation;
+-----------+-------------+------+-----+---------+-------+
| Field     | Type        | Null | Key | Default | Extra |
+-----------+-------------+------+-----+---------+-------+
| PNRNo     | varchar(20) | NO   | PRI |         |       |
| DOJ       | date        | YES  |     | NULL    |       |
| NofSeats  | 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 (0.01 sec)
```

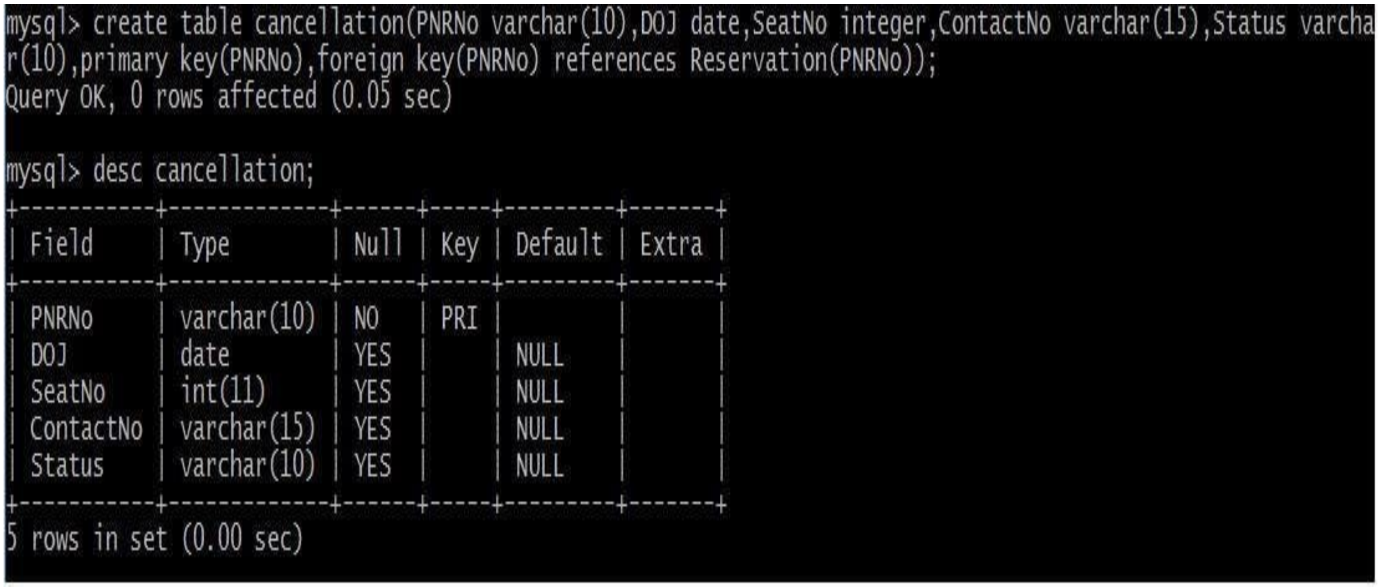
Cancellation:

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

ColumnName	**Datatype**	**Constraints**	**Type of Attributes**
PNRNo	Varchar(10)	Primary Key	Single-valued
DOJ	date		Single-valued
SeatNo	Integer		Simple
ContactNo	Varchar(15)		Multi-valued
Status	Varchar(10)		Simple

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));

Mysql> desc cancellation;



```
mysql> create table cancellation(PNRNo varchar(10),DOJ date,SeatNo integer,ContactNo varchar(15),Status varcha
r(10),primary key(PNRNo),foreign key(PNRNo) references Reservation(PNRNo));
Query OK, 0 rows affected (0.05 sec)

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.

Until now we have created table without using any constraint, Hence the tables have not been givenany instructions to filter what is being stored in the table.
The following are the types of integrity constraints

1. ***Domain Integrity constraints***
2. **Entity Integrity constraints**
3. ***Referential Integrity constraint***
4. **Oracle allows programmers to define constraints**
5. ***Column Level***
6. **Table Level**

Column Level constraints:

If data constraints are defined along with the column definition when creating or altering a table structure, they are column level constraints. Column level constraints are applied to the current column. The current column is the column that immediately precedes the constraints i.e. they are local to a specific column. Column level constraints cannot be applied if the data constraints span across the multiple columns in a table.

Table Level Constraint:

If the data constraints are defined after defining all the table columns when creating or altering a table structure, it is a table level constraint. Table Level constraints mostly used when data constraints spans across multiple columns in a table.

Domain Integrity Constraints:

These constraints set a range and any violations that take place will prevent the user from performing the manipulations that caused the breached.

Entity Integrity Constraints:

This type of constraints are further classified into

1. **Unique Constraint**
2. **Primary Key Constraint**

Unique Constraint:

The purpose of unique key is to ensure that information in the column(s) is unique i.e. the value entered in column(s) defined in the unique constraint must not be repeated across the column. A table may have many unique keys. If unique constraint is defined in more than one column (combination of columns), it is said to be composite unique key. Maximum combination of columns that a composite unique key can contain is 16.

Primary Key Constraint:

A primary key is one or on more columns(s) in a table to uniquely identify each row in the table. A primary key column in a table has a special attribute. It defines the column, as a mandatory column i.e. the column cannot be left blank and should have a unique value. Here by default not null constraint is attached with the column. A multicolumn primary key is called a Composite primary key. The only function of a primary key in a table is to uniquely identify a row. A table can have only one primary key.

Referential Integrity Constraint:

In this category there is only one constraint and it is Foreign Key & References to establish a Parent- child_ or a Master-detail_ relationship between two tables having a common column, we make use of referential integrity constraint. Foreign key represent relationships between tables. A foreign key is a column whose values are derived from the primary key or unique key. The table in which the foreign key is defined is called a foreign table or Detail table. The table that defines the primary or unique keys and is referenced by the foreign key is called the Primary table or Master table. The master table can be referenced in the foreign key definition by using references keyword. If the column name is not specified, by default, Oracle references the primary key in the master table.

The existence of a foreign key implies that the table with the foreign key is related to the master table from which the foreign key is derived. A foreign key must have a corresponding primary key or a unique key value in a master table.

Principles of Foreign Key Constraint:

Rejects an insert or update of a value in a particular column, if a corresponding value does not exist inthe master table.

Deletion of rows from the Master table is not possible if detail table having corresponding values.

Primary key or unique key must in Master table.

Requires that the foreign key column(s) and reference column(s) have same data type

References constraint defined at column level

Normalization is a process of converting a relation to be standard form by decomposition a larger relation intosmaller 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 – valuedattributes are not permitted.
- 2NF: A Relation scheme is said to be in 2NF,iff and every Non-key attribute is fully functionally dependent On primary Key.
- 3NF: A Relation scheme is said to be in 3NF,iff and does not have transitivity dependencies. A Relation is Said to be 3NF if every determinant is a key for each & every functional dependency.

- 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 & Qis a single attribute of R.

The given FD is a trival

P is a super key.

Normalized tables are:-

```
Mysql> create table Bus2(BusNo varchar(20) primary key,Source varchar(20),Destination varchar(20));
```

```
Mysql>Create table passenger4(PPN varchar(15) Primary key,Name varchar(20),Age integer,Sex
char,Addressvarchar(20));
```

```
Mysql> Create table PassengerTicket(PPN varchar(15) Primary key,TicketNo integer);
```

```
Mysql> Create table Reservation2(PNRNO integer Primary key, JourneyDate DateTime,NoofSeats int,Address
varchar(20),ContactNo Integer);
```

```
Mysql> create table Cancellation2(PNRNO Integer primary key,JourneyDate DateTime,NoofSeats
Integer,Address varchar(20),ContactNo Integer,foreign key(PNRNO) references Reservation2(PNRNO));
```

```
Mysql> Create table Ticket2(TicketNo Integer Primary key,JourneyDate DateTime, Age Int(4),Sex char(2),Source
varchar(20),Destination varchar(20),DeptTime varchar(2));
```

EXPERIMENT – 4
PRACTICING DDL COMMANDS

AIM: **Installation of Mysql in Ubuntu.**

MySQL is a fast, easy to use relational database. It is currently the most popular open- source database. It is very commonly used in conjunction with PHP scripts to create powerful and dynamic server-side applications.

MySQL is used for many small and big businesses. It is developed, marketed and supported by MySQL AB, a Swedish company. It is written in C and C++.

- Relational Database Management System (RDBMS): MySQL is a relational database management system.
- Easy to use: MySQL is easy to use. You have to get only the basic knowledge of SQL. You can build and interact with MySQL with only a few simple SQL statements.
- It is secure: MySQL consist of a solid data security layer that protects sensitive data from intruders. Passwords are encrypted in MySQL.
- Client/ Server Architecture: MySQL follows a client /server architecture. There is a database server (MySQL) and arbitrarily many clients (application programs), which communicate with the server; that is, they query data, save changes, etc.
- Free to download: MySQL is free to use and you can download it from MySQL official website.
- It is scalable: MySQL can handle almost any amount of data, up to as much as 50 million rows or more. The default file size limit is about 4 GB. However, you can increase this number to a theoretical limit of 8 TB of data.

Installation of Mysql. In this week you will learn creating databases. How to create table, altering the database, dropping table and databases if not required. You will also try truncate, rename commands etc…

RESOURCE:

Ubuntu (Linux) / My Sql database

PROCEDURE:

Installation of MySql:

Follow these steps on to install MySql in Ubuntu:

1. Open Terminal and run below command.sudo

   apt-get install mysql-server

2. Give the root password.
3. Wait for the installation to finish.
4. The installer itself start the MySql server. To check whether MySql server is running or not, runbelow command.

```
sudo netstat-tap | grep mysql
```

5. To make sure. Your MySql installation works fine with Apache and PHP, run below command.It will install necessary modules to connect to a MySql database through PHP using Apache.

```
sudo apt-get install libapache2-mod-auth-mysql php5-mysql
```

6. Installation is completed.

SQL

- SQL stands for Structured Query Language. It is used for storing and managing data in relational database management system (RDMS).
- It is a standard language for Relational Database System. It enables a user to create,read, update and delete relational databases and tables.
- All the RDBMS like MySQL, Informix, Oracle, MS Access and SQL Server use SQLas their standard database language.
- SQL allows users to query the database in a number of ways, using English-like statements.

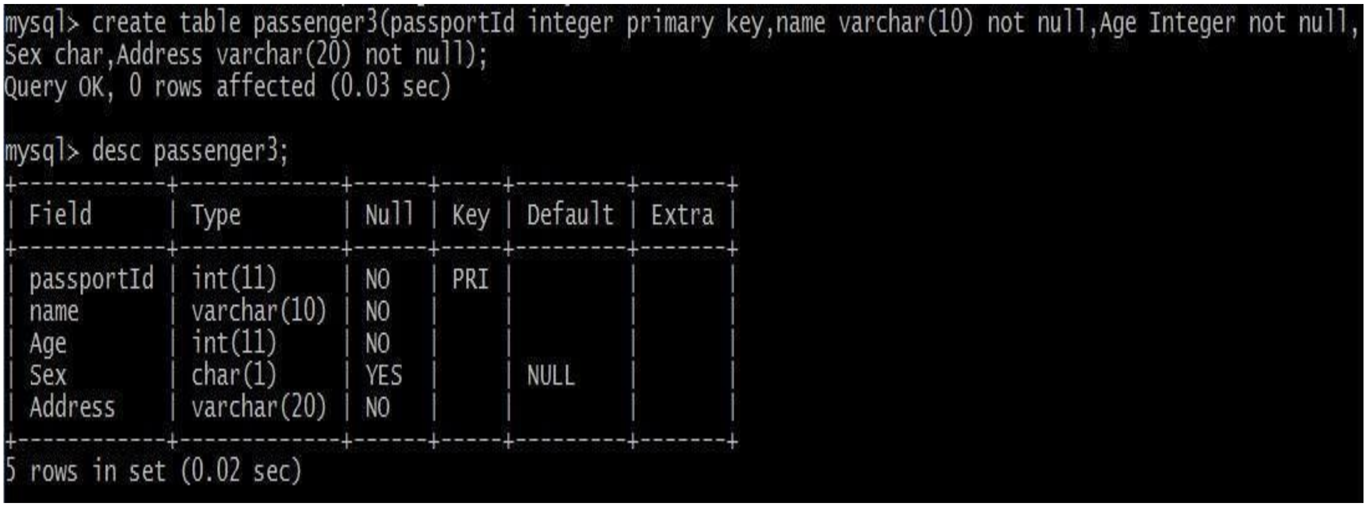
AIM : Creating Tables and altering the Tables

Creation of databases:

```
mysql> show databases;
+------------------------------+
| Database              |
+------------------------------+
| information_schema |
| mysql                  |
| test                   |
+------------------------------+
3 rows in set (0.09 sec)
mysql> create database groupa; Query OK, 1 row
affected (0.01 sec)mysql> use groupa;
Database changed
```

Mysql>Create table passenger2(passportId Integer Primary Key,Name varchar(10) Not Null,Age Integer Not Null,Sex char,Address varchar(20) Not Null);

Mysql> desc passenger2;

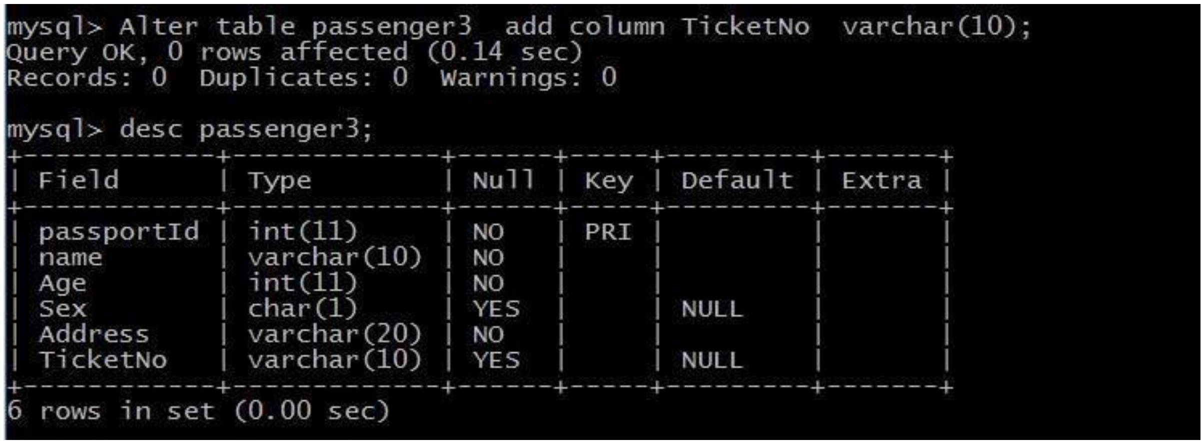


```
mysql> create table passenger3(passportId integer primary key,name varchar(10) not null,Age Integer not null,
Sex char,Address varchar(20) not null);
Query OK, 0 rows affected (0.03 sec)

mysql> desc passenger3;
+------------+-------------+------+-----+---------+-------+
| Field      | Type        | Null | Key | Default | Extra |
+------------+-------------+------+-----+---------+-------+
| passportId | int(11)     | NO   | PRI |         |       |
| name       | varchar(10) | NO   |     |         |       |
| Age        | int(11)     | NO   |     |         |       |
| Sex        | char(1)     | YES  |     | NULL    |       |
| Address    | varchar(20) | NO   |     |         |       |
+------------+-------------+------+-----+---------+-------+
5 rows in set (0.02 sec)
```

USING ALTER COMMAND

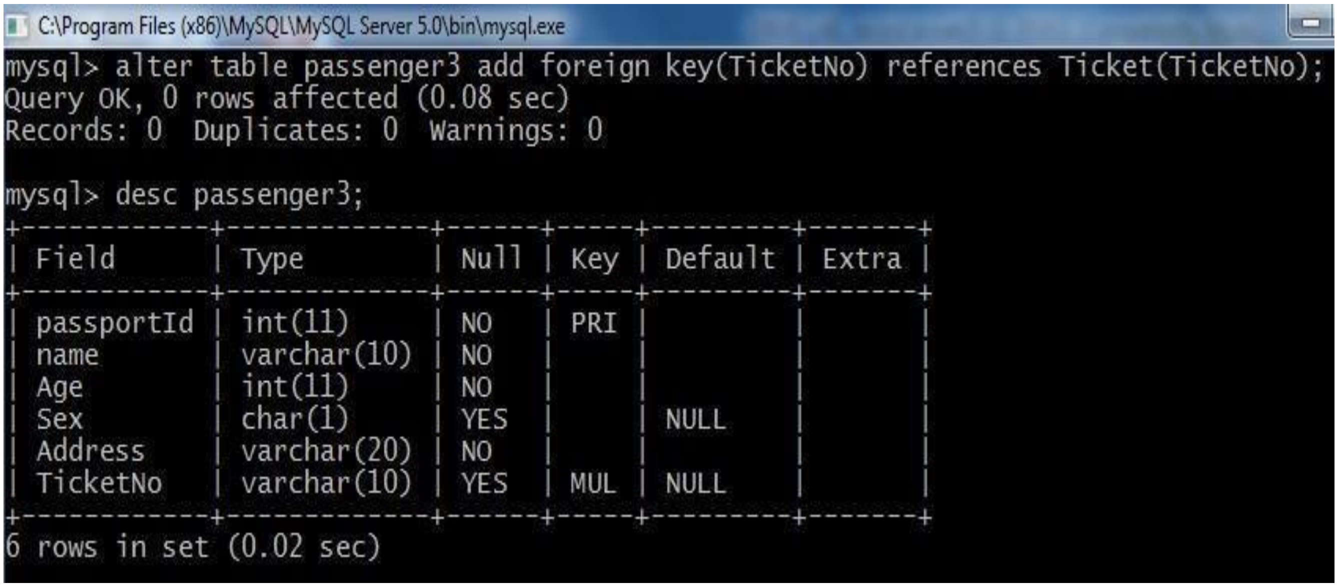
Adding Extra column to Existing Table

Mysql>Alter table passenger3 add column TicketNo varchar(10);



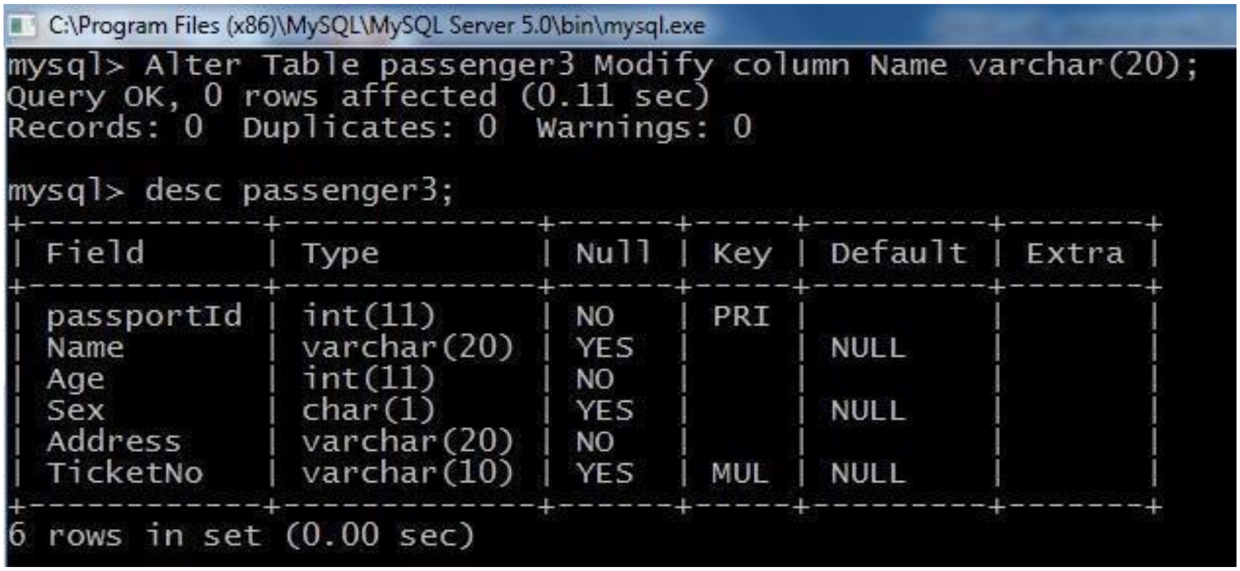
```
mysql> Alter table passenger3  add column TicketNo  varchar(10);
Query OK, 0 rows affected (0.14 sec)
Records: 0  Duplicates: 0  Warnings: 0

mysql> desc passenger3;
+------------+-------------+------+-----+---------+-------+
| Field      | Type        | Null | Key | Default | Extra |
+------------+-------------+------+-----+---------+-------+
| passportId | int(11)     | NO   | PRI |         |       |
| name       | varchar(10) | NO   |     |         |       |
| Age        | int(11)     | NO   |     |         |       |
| Sex        | char(1)     | YES  |     | NULL    |       |
| Address    | varchar(20) | NO   |     |         |       |
| TicketNo   | varchar(10) | YES  |     | NULL    |       |
+------------+-------------+------+-----+---------+-------+
6 rows in set (0.00 sec)
```

Mysql>Alter Table passenger3 add Foreign key(TicketNo) references Ticket(TicketNo);



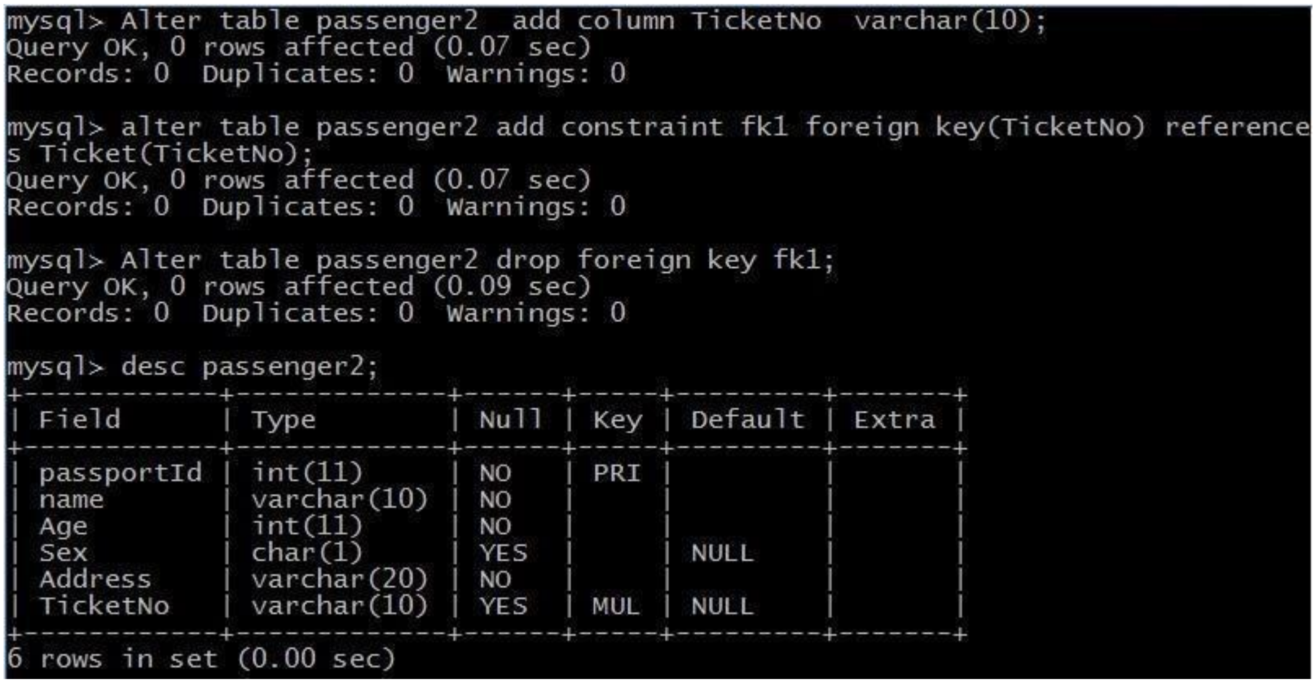
```
mysql> alter table passenger3 add foreign key(TicketNo) references Ticket(TicketNo);
Query OK, 0 rows affected (0.08 sec)
Records: 0  Duplicates: 0  Warnings: 0

mysql> desc passenger3;
+------------+-------------+------+-----+---------+-------+
| Field      | Type        | Null | Key | Default | Extra |
+------------+-------------+------+-----+---------+-------+
| passportId | int(11)     | NO   | PRI |         |       |
| name       | varchar(10) | NO   |     |         |       |
| Age        | int(11)     | NO   |     |         |       |
| Sex        | char(1)     | YES  |     | NULL    |       |
| Address    | varchar(20) | NO   |     |         |       |
| TicketNo   | varchar(10) | YES  | MUL | NULL    |       |
+------------+-------------+------+-----+---------+-------+
6 rows in set (0.02 sec)
```

Mysql>Alter Table passenger3 Modify column Name varchar(20);



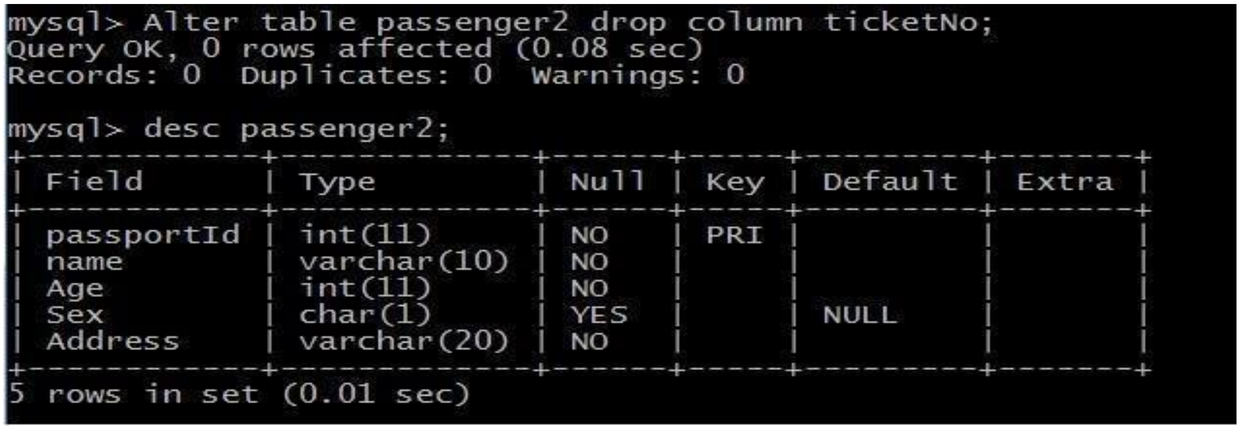
```
mysql> Alter Table passenger3 Modify column Name varchar(20);
Query OK, 0 rows affected (0.11 sec)
Records: 0  Duplicates: 0  Warnings: 0

mysql> desc passenger3;
+------------+-------------+------+-----+---------+-------+
| Field      | Type        | Null | Key | Default | Extra |
+------------+-------------+------+-----+---------+-------+
| passportId | int(11)     | NO   | PRI |         |       |
| Name       | varchar(20) | YES  |     | NULL    |       |
| Age        | int(11)     | NO   |     |         |       |
| Sex        | char(1)     | YES  |     | NULL    |       |
| Address    | varchar(20) | NO   |     |         |       |
| TicketNo   | varchar(10) | YES  | MUL | NULL    |       |
+------------+-------------+------+-----+---------+-------+
6 rows in set (0.00 sec)
```

Mysql>Alter table passenger drop foreign key fk1;



```
mysql> Alter table passenger2  add column TicketNo  varchar(10);
Query OK, 0 rows affected (0.07 sec)
Records: 0  Duplicates: 0  Warnings: 0

mysql> alter table passenger2 add constraint fk1 foreign key(TicketNo) reference
s Ticket(TicketNo);
Query OK, 0 rows affected (0.07 sec)
Records: 0  Duplicates: 0  Warnings: 0

mysql> Alter table passenger2 drop foreign key fk1;
Query OK, 0 rows affected (0.09 sec)
Records: 0  Duplicates: 0  Warnings: 0

mysql> desc passenger2;
+------------+-------------+------+-----+---------+-------+
| Field      | Type        | Null | Key | Default | Extra |
+------------+-------------+------+-----+---------+-------+
| passportId | int(11)     | NO   | PRI |         |       |
| name       | varchar(10) | NO   |     |         |       |
| Age        | int(11)     | NO   |     |         |       |
| Sex        | char(1)     | YES  |     | NULL    |       |
| Address    | varchar(20) | NO   |     |         |       |
| TicketNo   | varchar(10) | YES  | MUL | NULL    |       |
+------------+-------------+------+-----+---------+-------+
6 rows in set (0.00 sec)
```

Mysql> Alter table passenger2 Drop column TicketNo;



```
mysql> Alter table passenger2 drop column ticketNo;
Query OK, 0 rows affected (0.08 sec)
Records: 0  Duplicates: 0  Warnings: 0

mysql> desc passenger2;
+------------+-------------+------+-----+---------+-------+
| Field      | Type        | Null | Key | Default | Extra |
+------------+-------------+------+-----+---------+-------+
| passportId | int(11)     | NO   | PRI |         |       |
| name       | varchar(10) | NO   |     |         |       |
| Age        | int(11)     | NO   |     |         |       |
| Sex        | char(1)     | YES  |     | NULL    |       |
| Address    | varchar(20) | NO   |     |         |       |
+------------+-------------+------+-----+---------+-------+
5 rows in set (0.01 sec)
```

EXPERIMENT – 5
PRACTICING DML COMMANDS

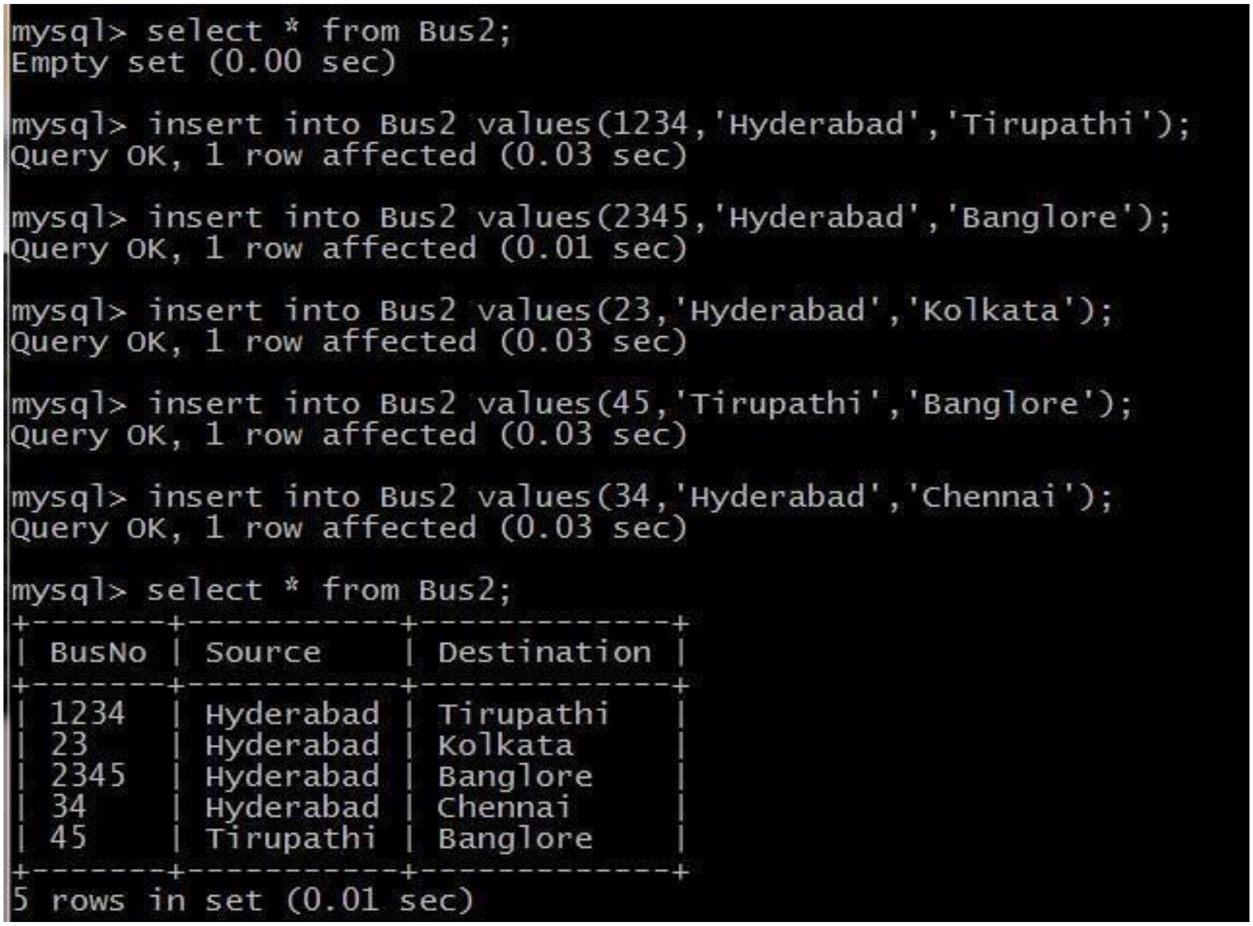
AIM: Create a DML Commands are used to manage data within the scheme objects.

DML Commands:

INSERT COMMAND ON BUS2 & PASSENGER2 RELATIONS

```
mysql> select * from Bus2; Empty set (0.00 sec)

mysql> insert into Bus2 values(1234,'Hyderabad','Tirupathi');

Query OK, 1 row affected (0.03 sec)

mysql> insert into Bus2 values(2345,'Hyderabad','Banglore');

Query OK, 1 row affected (0.01 sec)

mysql> insert into Bus2 values(23,'Hyderabad','Kolkata');

 Query OK, 1 row affected (0.03 sec)

mysql> insert into Bus2 values(45,'Tirupathi','Banglore');

Query OK, 1 row affected (0.03 sec)

mysql> insert into Bus2 values(34,'Hyderabad','Chennai');

Query OK, 1 row affected (0.03 sec)
```

mysql> select * from Bus2;



```
mysql> select * from Bus2;
Empty set (0.00 sec)

mysql> insert into Bus2 values(1234,'Hyderabad','Tirupathi');
Query OK, 1 row affected (0.03 sec)

mysql> insert into Bus2 values(2345,'Hyderabad','Banglore');
Query OK, 1 row affected (0.01 sec)

mysql> insert into Bus2 values(23,'Hyderabad','Kolkata');
Query OK, 1 row affected (0.03 sec)

mysql> insert into Bus2 values(45,'Tirupathi','Banglore');
Query OK, 1 row affected (0.03 sec)

mysql> insert into Bus2 values(34,'Hyderabad','Chennai');
Query OK, 1 row affected (0.03 sec)

mysql> select * from Bus2;
+-------+-----------+-------------+
| BusNo | Source    | Destination |
+-------+-----------+-------------+
| 1234  | Hyderabad | Tirupathi   |
| 23    | Hyderabad | Kolkata     |
| 2345  | Hyderabad | Banglore    |
| 34    | Hyderabad | Chennai     |
| 45    | Tirupathi | Banglore    |
+-------+-----------+-------------+
5 rows in set (0.01 sec)
```

mysql> select * from Passenger2;

Empty set (0.00 sec)

mysql> insert into Passenger2 values(145,'Ramesh',45,'M','abc123');

Query OK, 1 row affected (0.05 sec)

mysql> insert into Passenger2 values(278,'Geetha',36,'F','abc124');

Query OK, 1 row affected (0.02 sec)

mysql> insert into Passenger2 values(4590,'Ram',30,'M','abc12');

Query OK, 1 row affected (0.03 sec)

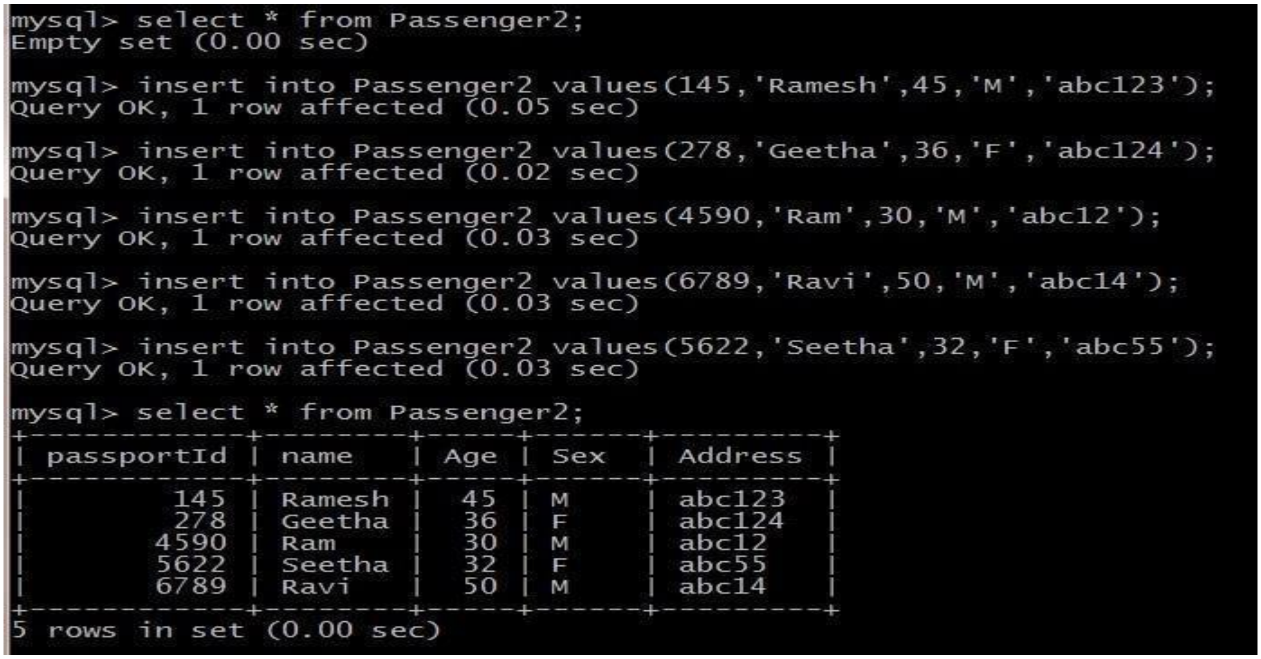
mysql> insert into Passenger2 values(6789,'Ravi',50,'M','abc14');

Query OK, 1 row affected (0.03 sec)

mysql> insert into Passenger2 values(5622,'Seetha',32,'F','abc55');

Query OK, 1 row affected (0.03 sec)

mysql> select * from Passenger2;



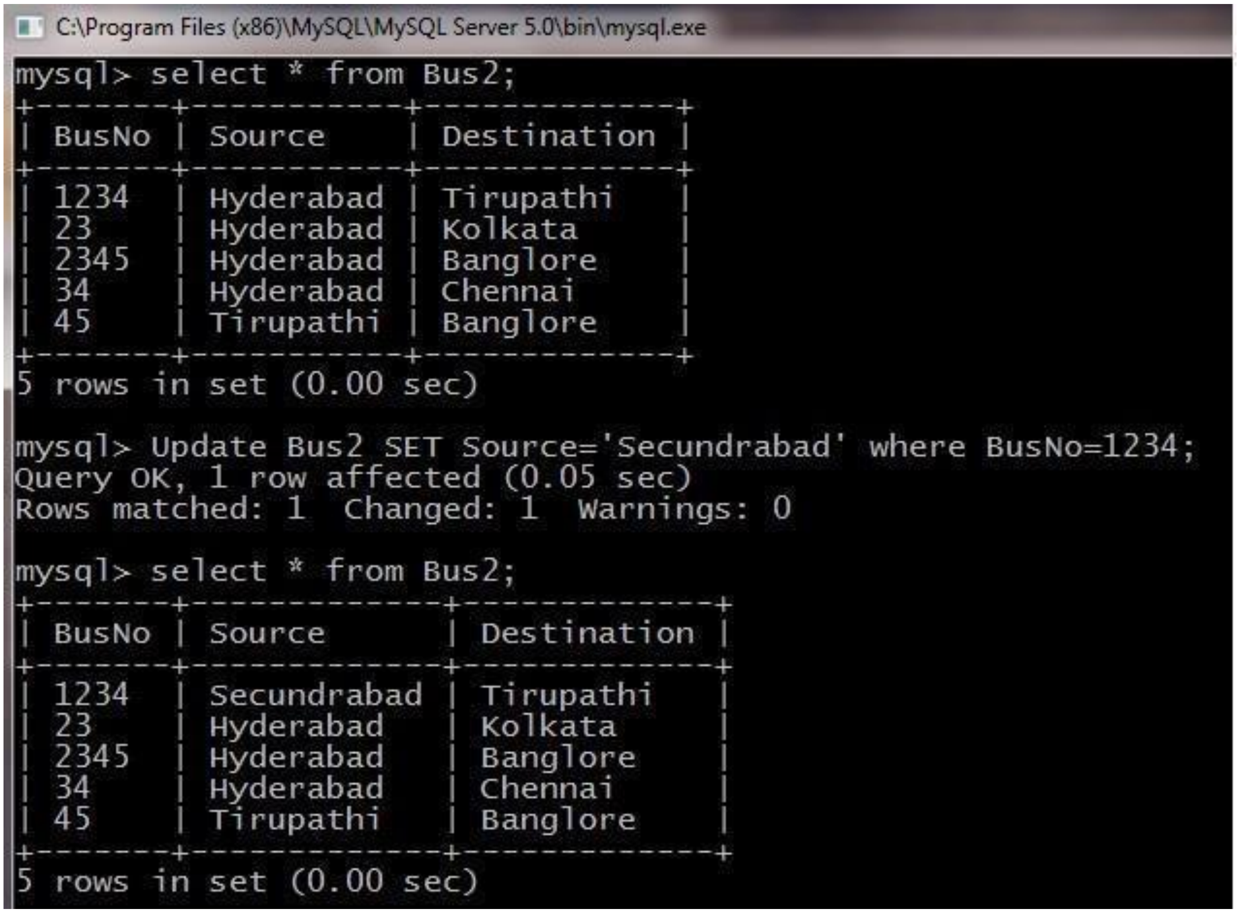
```
mysql> select * from Passenger2;
Empty set (0.00 sec)

mysql> insert into Passenger2 values(145,'Ramesh',45,'M','abc123');
Query OK, 1 row affected (0.05 sec)

mysql> insert into Passenger2 values(278,'Geetha',36,'F','abc124');
Query OK, 1 row affected (0.02 sec)

mysql> insert into Passenger2 values(4590,'Ram',30,'M','abc12');
Query OK, 1 row affected (0.03 sec)

mysql> insert into Passenger2 values(6789,'Ravi',50,'M','abc14');
Query OK, 1 row affected (0.03 sec)

mysql> insert into Passenger2 values(5622,'Seetha',32,'F','abc55');
Query OK, 1 row affected (0.03 sec)

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

UPDATE COMMAND ON BUS2 RELATION

UPDATE Selected Rows & Multiple Rows

mysql> Update Bus2 SET Source='Secundrabad' where BusNo=1234; Query OK, 1 row affected (0.05 sec)

Rows matched: 1 Changed: 1 Warnings: 0



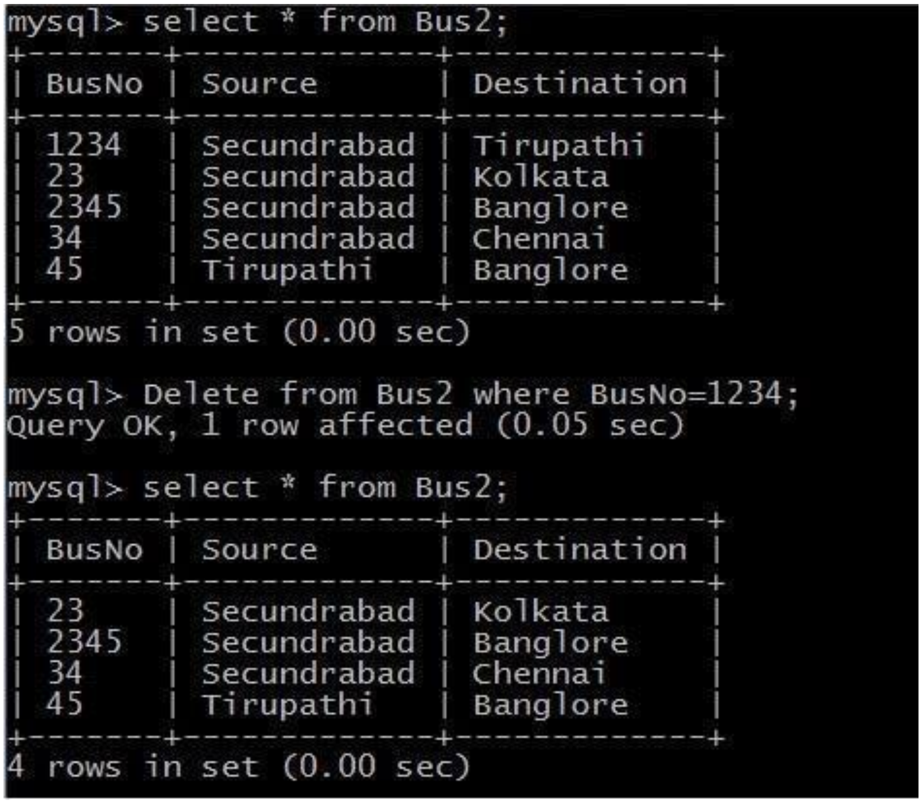
```
mysql> select * from Bus2;
+-------+-----------+-------------+
| BusNo | Source    | Destination |
+-------+-----------+-------------+
| 1234  | Hyderabad | Tirupathi   |
| 23    | Hyderabad | Kolkata     |
| 2345  | Hyderabad | Banglore    |
| 34    | Hyderabad | Chennai     |
| 45    | Tirupathi | Banglore    |
+-------+-----------+-------------+
5 rows in set (0.00 sec)

mysql> Update Bus2 SET Source='Secundrabad' where BusNo=1234;
Query OK, 1 row affected (0.05 sec)
Rows matched: 1  Changed: 1  Warnings: 0

mysql> select * from Bus2;
+-------+-------------+-------------+
| BusNo | Source      | Destination |
+-------+-------------+-------------+
| 1234  | Secundrabad | Tirupathi   |
| 23    | Hyderabad   | Kolkata     |
| 2345  | Hyderabad   | Banglore    |
| 34    | Hyderabad   | Chennai     |
| 45    | Tirupathi   | Banglore    |
+-------+-------------+-------------+
5 rows in set (0.00 sec)
```

DELETE COMMAND ON BUS2 RELATION

DELETES Selected Rows and Multiple Rows

mysql> Delete from Bus2 where BusNo=1234; Query OK, 1 row affected (0.05 sec)

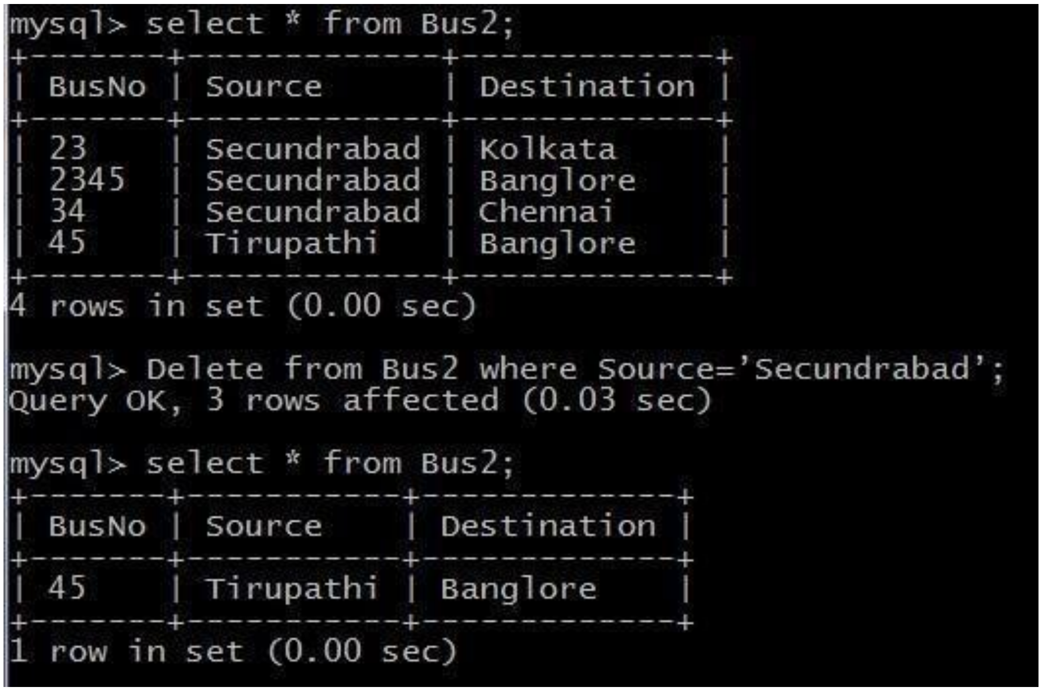
mysql> select * from Bus2;



```
mysql> select * from Bus2;
+-------+------------+------------+
| BusNo | Source     | Destination |
+-------+------------+------------+
| 1234  | Secundrabad | Tirupathi  |
| 23    | Secundrabad | Kolkata    |
| 2345  | Secundrabad | Banglore   |
| 34    | Secundrabad | Chennai    |
| 45    | Tirupathi  | Banglore   |
+-------+------------+------------+
5 rows in set (0.00 sec)

mysql> Delete from Bus2 where BusNo=1234;
Query OK, 1 row affected (0.05 sec)

mysql> select * from Bus2;
+-------+------------+------------+
| BusNo | Source     | Destination |
+-------+------------+------------+
| 23    | Secundrabad | Kolkata    |
| 2345  | Secundrabad | Banglore   |
| 34    | Secundrabad | Chennai    |
| 45    | Tirupathi  | Banglore   |
+-------+------------+------------+
4 rows in set (0.00 sec)
```

mysql> Delete from Bus2 where Source=’Secundrabad’; Query OK, 1 row affected (0.05 sec)

mysql> select * from Bus2;



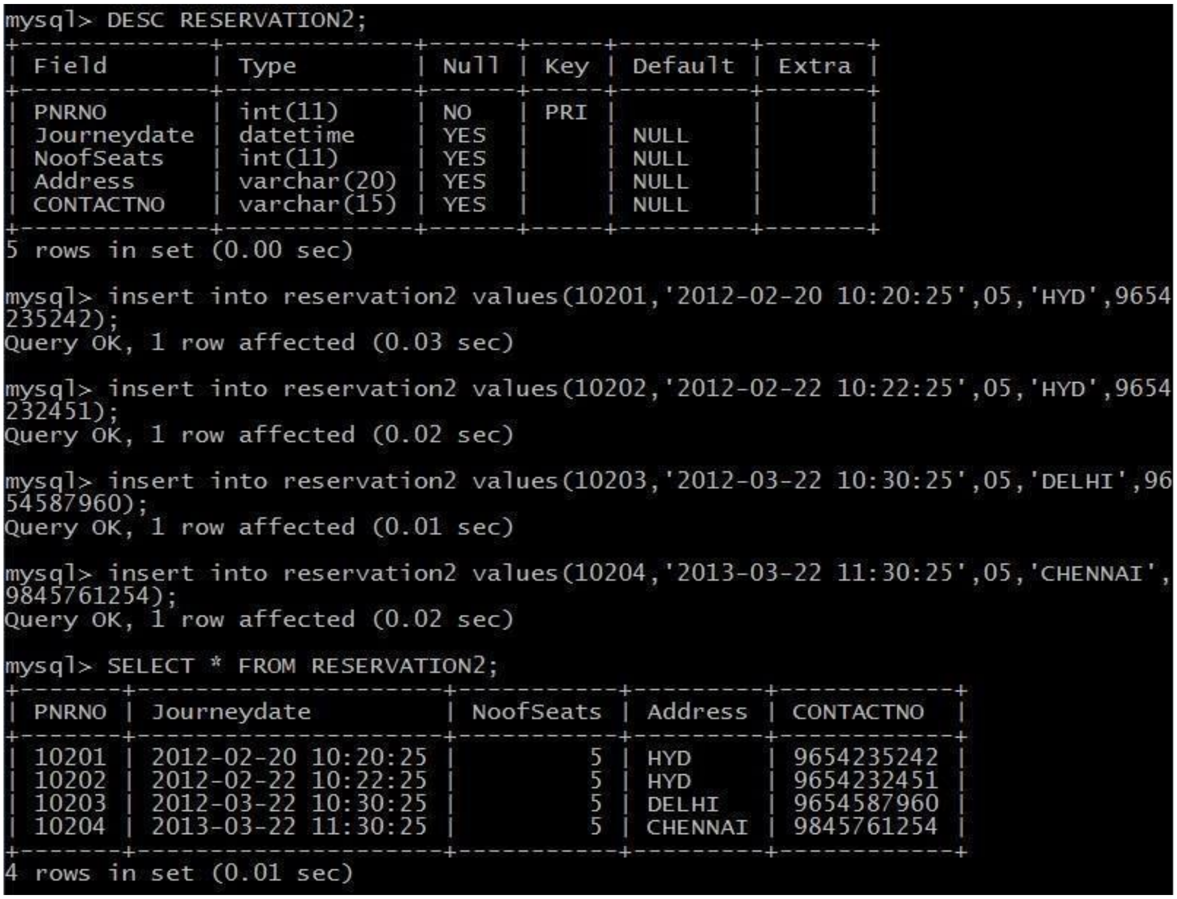
```
mysql> select * from Bus2;
+-------+-------------+-------------+
| BusNo | Source      | Destination |
+-------+-------------+-------------+
| 23    | Secundrabad | Kolkata     |
| 2345  | Secundrabad | Banglore    |
| 34    | Secundrabad | Chennai     |
| 45    | Tirupathi   | Banglore    |
+-------+-------------+-------------+
4 rows in set (0.00 sec)

mysql> Delete from Bus2 where Source=’Secundrabad’;
Query OK, 3 rows affected (0.03 sec)

mysql> select * from Bus2;
+-------+-----------+-------------+
| BusNo | Source    | Destination |
+-------+-----------+-------------+
| 45    | Tirupathi | Banglore    |
+-------+-----------+-------------+
1 row in set (0.00 sec)
```

EXPERIMENT – 6

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

Aim: Practice the following Queries:

1. Display unique PNR_NO of all passengers
2. Display all the names of male passengers.
3. Display the ticket numbers and names of all the passengers.
4. Find the ticket numbers of the passengers whose name start with ‘r’ and ends with ‘h’.
5. Find the names of Passengers whose age is between 30 and 45.
6. Display all the passengers names beginning with ‘A’.
7. Display the sorted list of Passengers names



```
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 passenger2 values(82302,'Smith',23,'M','Hyderabad');

Query OK, 1 row affected (0.02 sec)


mysql> insert into passenger2 values(82303,'Neha',23,'F','Hyderabad');

Query OK, 1 row affected (0.01 sec)


mysql> insert into passenger2 values(82304,'Neha',35,'F','Hyderabad');

Query OK, 1 row affected (0.03 sec)


mysql> insert into passenger2 values(82306,'Ramu',40,'M','Hyderabad');

Query OK, 1 row affected (0.02 sec)


mysql> insert into passenger2 values(82308,'Aakash',40,'M','Hyderabad');

Query OK, 1 row affected (0.02 sec)


mysql> insert into passenger2 values(82402,'Aravind',42,'M','Hyderabad');

Query OK, 1 row affected (0.02 sec)


mysql> insert into passenger2 values(82403,'Avinash',42,'M','Hyderabad');

Query OK, 1 row affected (0.02 sec)


mysql> insert into passenger2 values(82502,'Ramesh',23,'M','Hyderabad');

Query OK, 1 row affected (0.02 sec)


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',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)

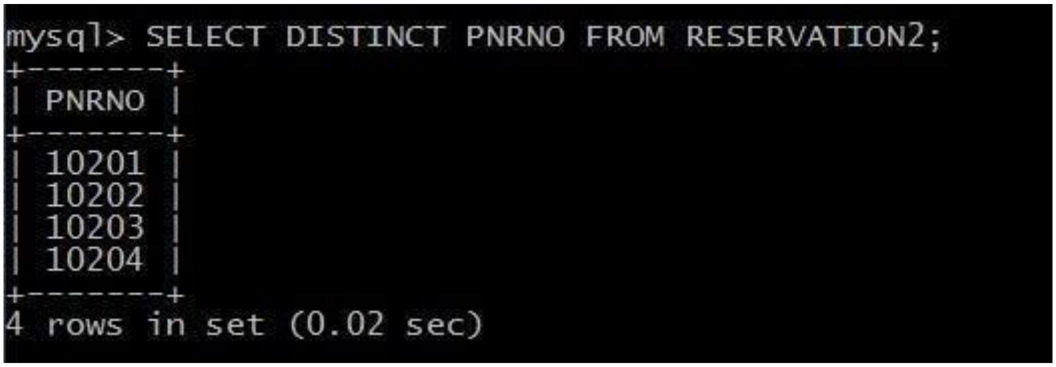
mysql> insert into reservation2 values(10204,'2013-03-22 11:30:25',05,'CHENNAI', 9845761254);

Query OK, 1 row affected (0.02 sec)

1. Display unique PNR_NO of all reservation Mysql>Select

   DISTINCT PNR_NO from Reservation;

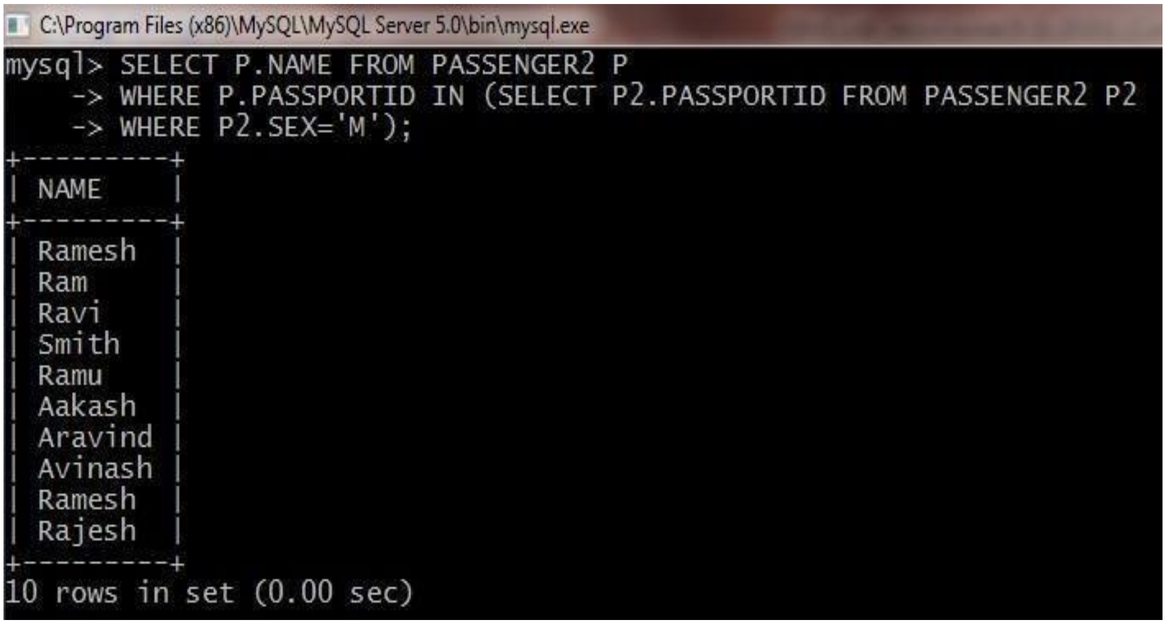
PNR_No
10201
10202
10203
10204



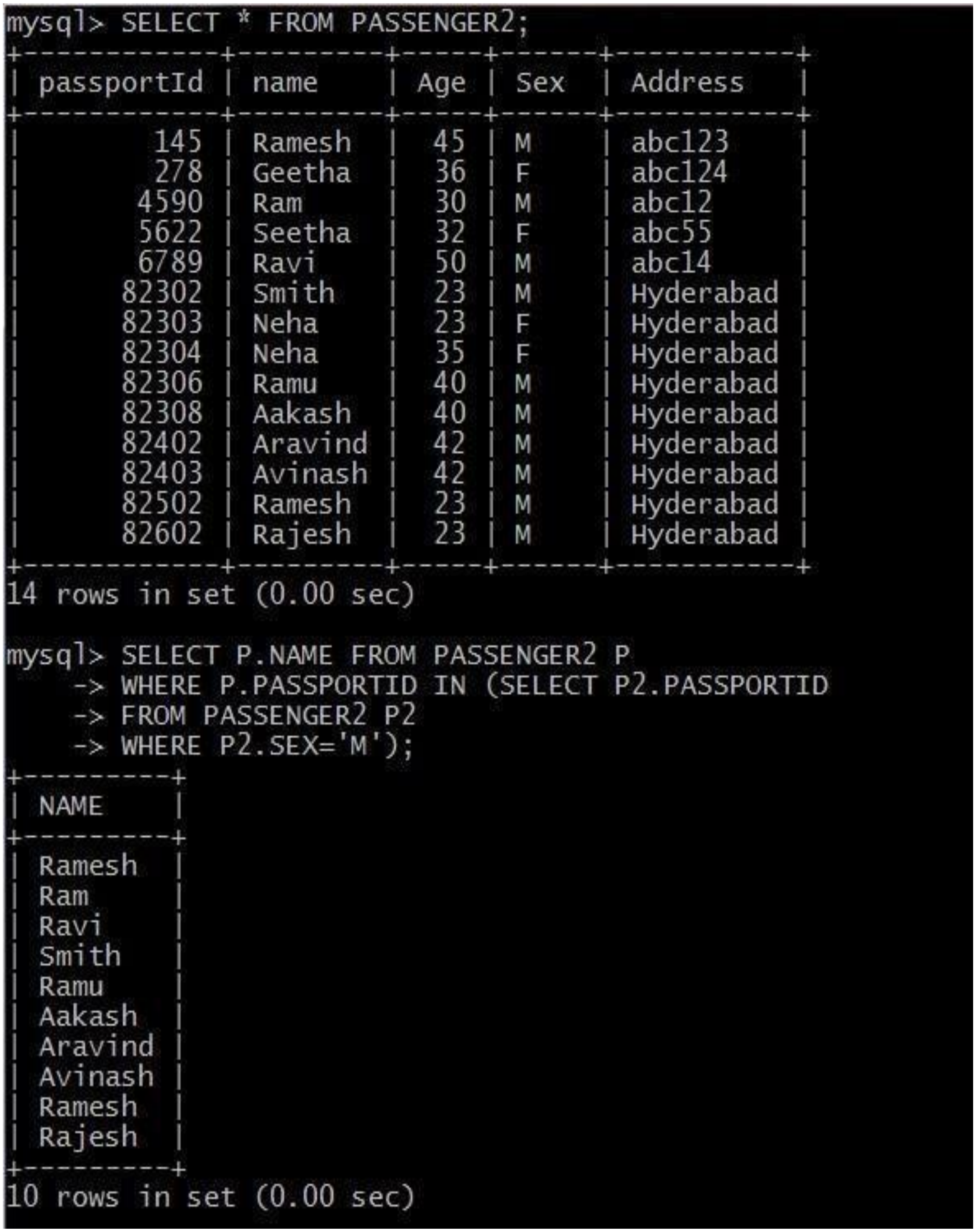
```
mysql> SELECT DISTINCT PNRNO FROM RESERVATION2;
+-------+
| PNRNO |
+-------+
| 10201 |
| 10202 |
| 10203 |
| 10204 |
+-------+
4 rows in set (0.02 sec)
```

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');
```

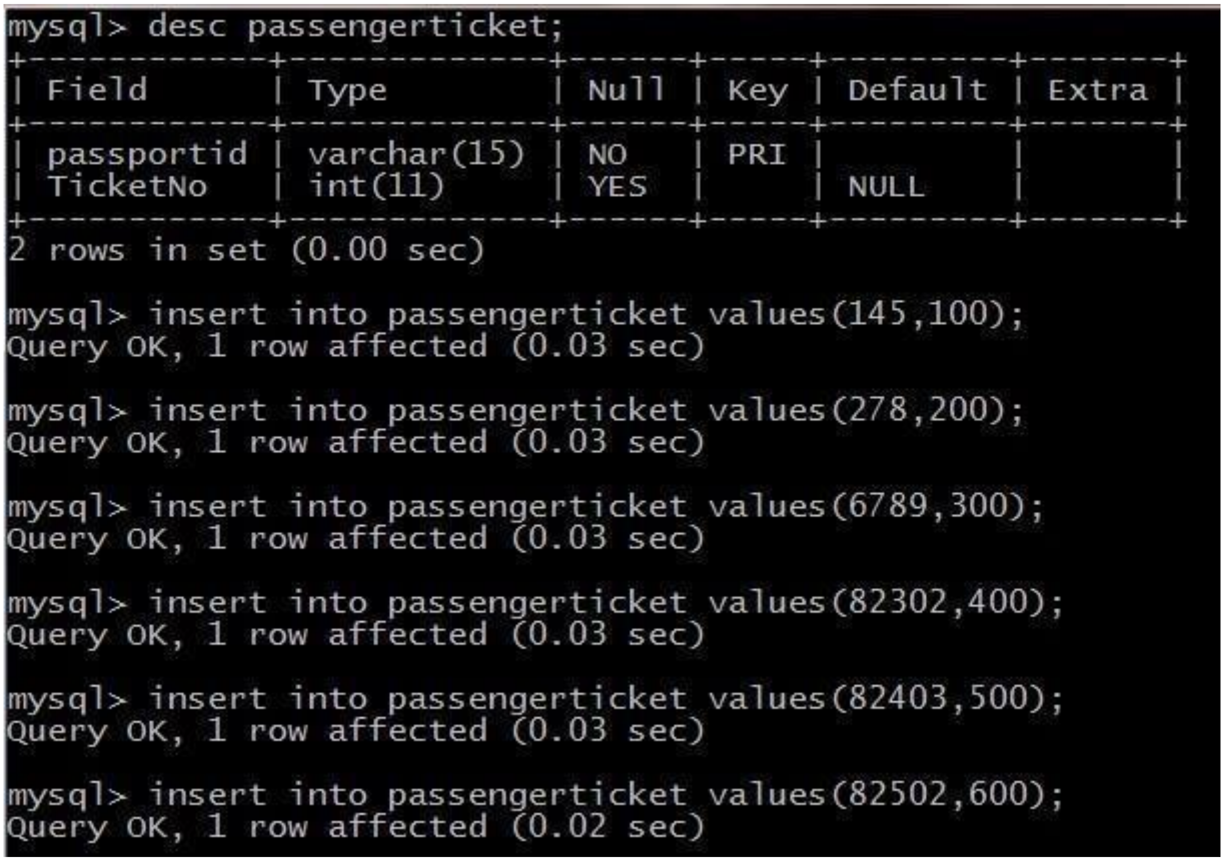


```
mysql> SELECT P.NAME FROM PASSENGER2 P
    -> WHERE P.PASSPORTID IN (SELECT P2.PASSPORTID FROM PASSENGER2 P2
    -> WHERE P2.SEX='M');
+---------+
| NAME    |
+---------+
| Ramesh  |
| Ram     |
| Ravi    |
| Smith   |
| Ramu    |
| Aakash  |
| Aravind |
| Avinash |
| Ramesh  |
| Rajesh  |
+---------+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



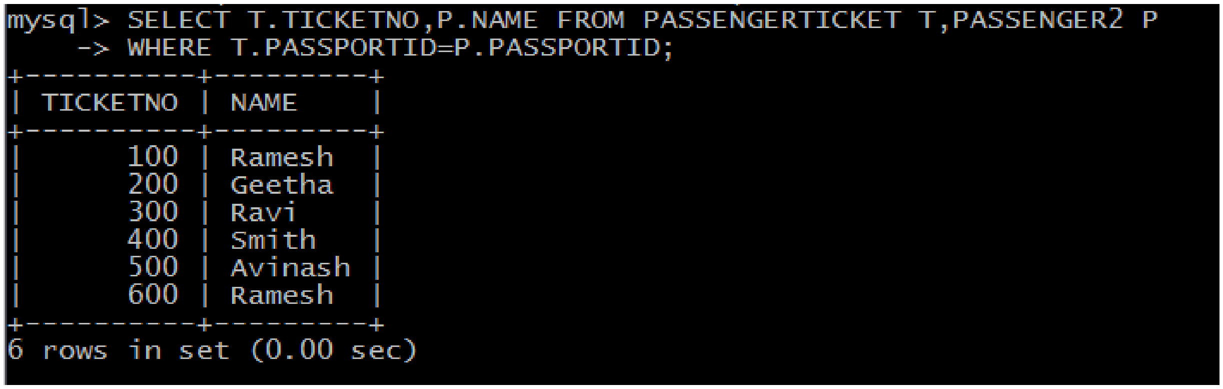
```
mysql> SELECT * FROM PASSENGER2;
+------------+---------+-----+------+-----------+
| passportId | name    | Age | Sex  | Address   |
+------------+---------+-----+------+-----------+
|        145 | Ramesh  |  45 | M    | abc123    |
|        278 | Geetha  |  36 | F    | abc124    |
|       4590 | 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 the passengers.



```
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;

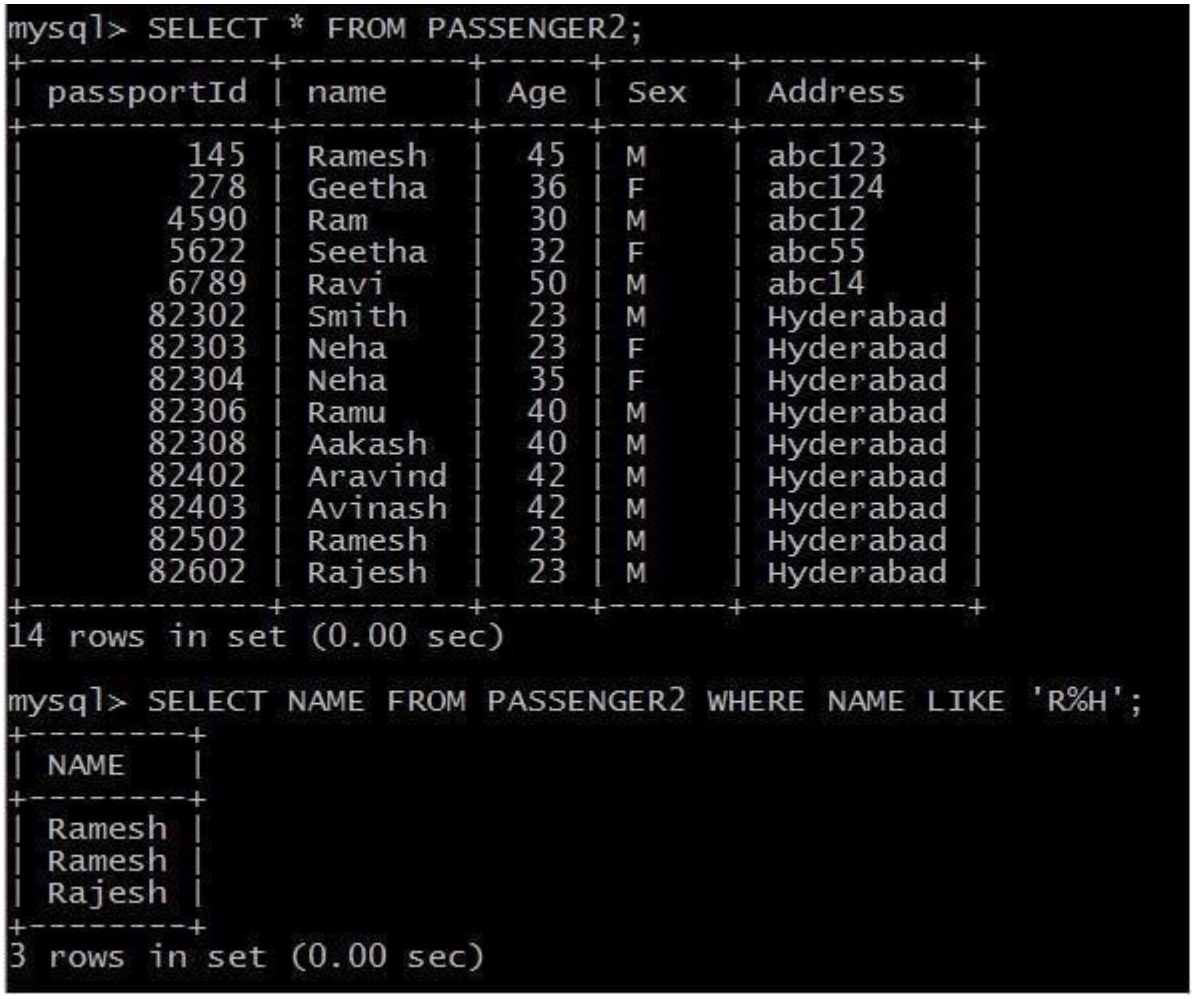


```
mysql> SELECT T.TICKETNO,P.NAME FROM PASSENGERTICKET T,PASSENGER2 P
    -> WHERE T.PASSPORTID=P.PASSPORTID;
+----------+---------+
| TICKETNO | NAME    |
+----------+---------+
|      100 | Ramesh  |
|      200 | Geetha  |
|      300 | Ravi    |
|      400 | Smith   |
|      500 | Avinash |
|   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

4. Find the ticket numbers of the passengers whose name start with ‘r’ and ends with ‘h’.

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

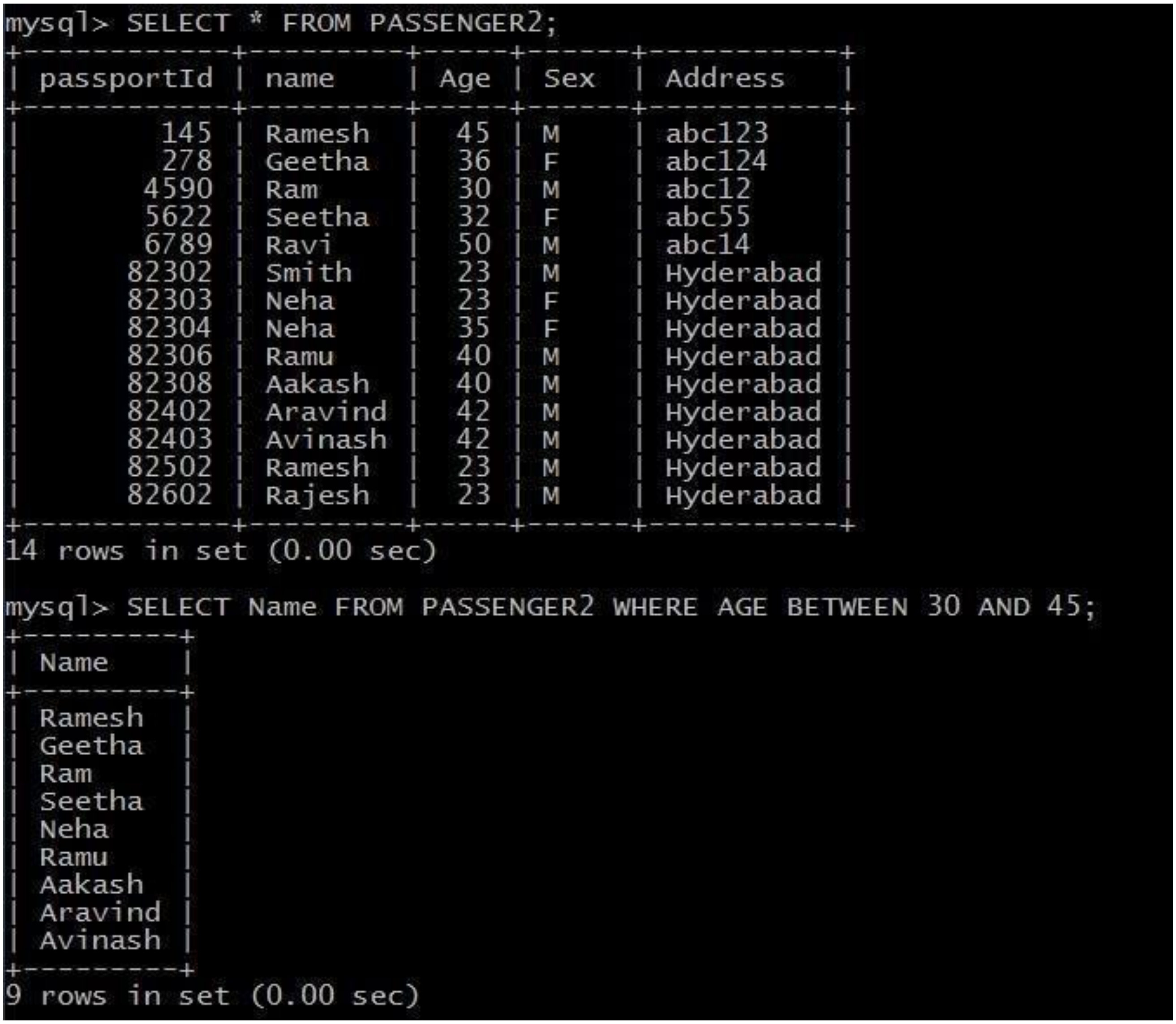
Name
Rajesh
Ramesh
Ramesh



```
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. Find the names of Passengers whose age is between 30 and 45.

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

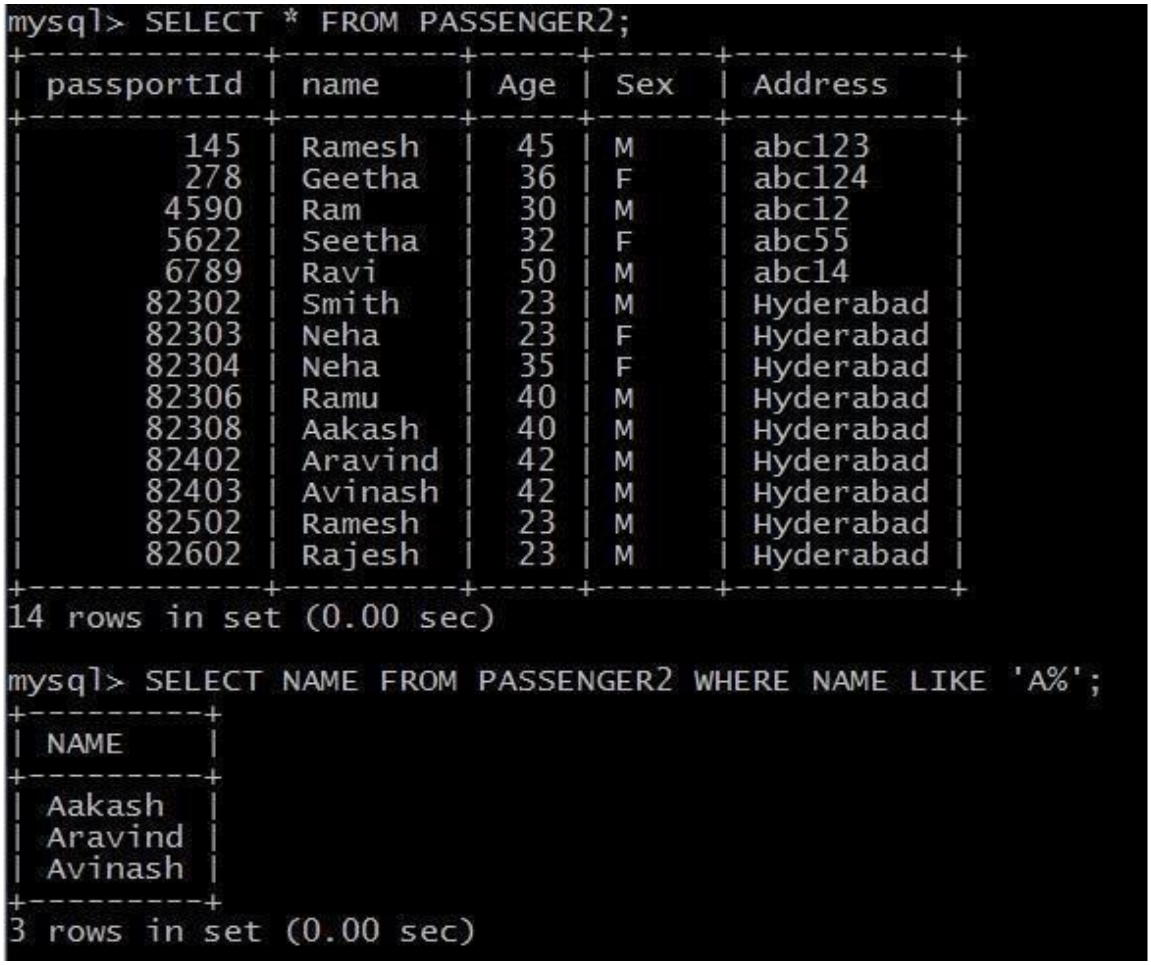


```
mysql> SELECT * FROM PASSENGER2;
+-----------+---------+-----+------+-----------+
| passportId | name    | Age | Sex  | Address   |
+-----------+---------+-----+------+-----------+
|       145 | Ramesh  |  45 | M    | abc123    |
|       278 | Geetha  |  36 | F    | abc124    |
|      4590 | Ram     |  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%’;

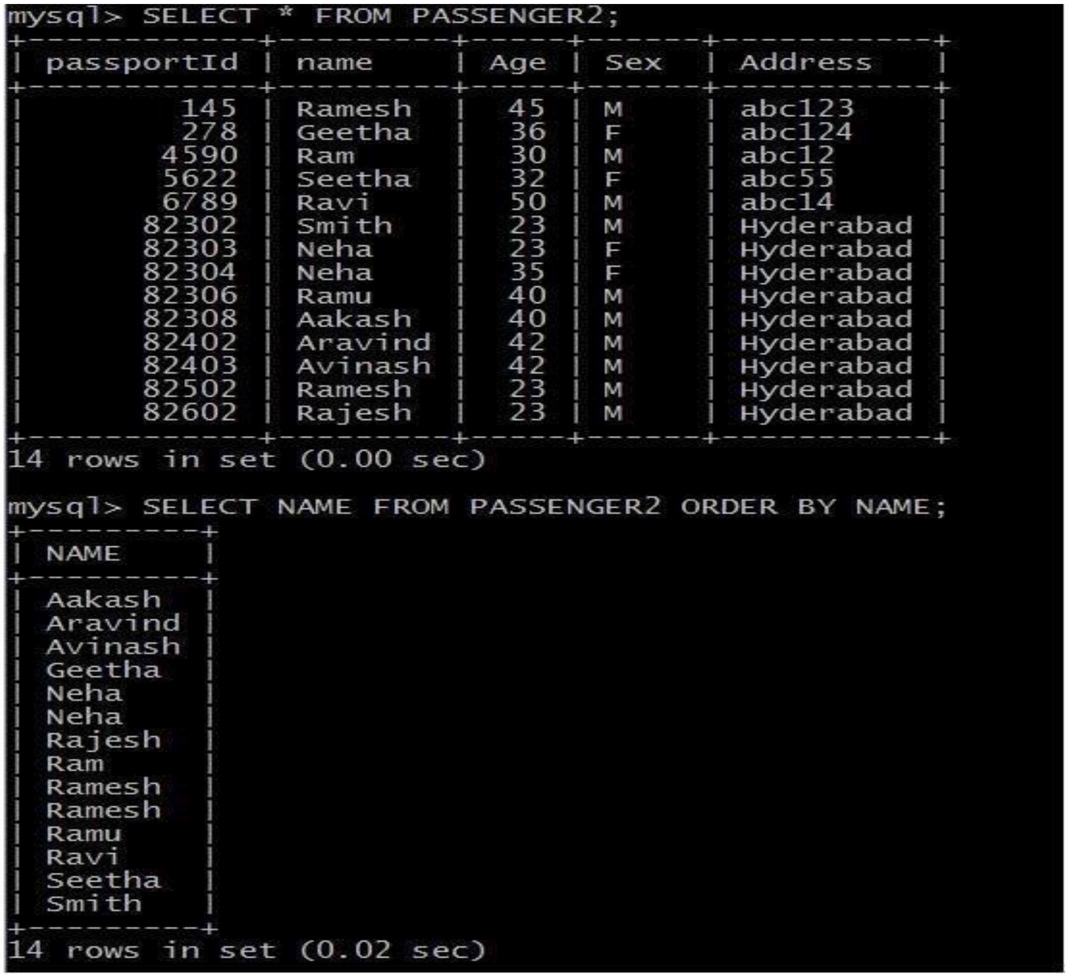
Name
Akash
Arivind
Avinash



```
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 Passengers names

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 and cancellation tables
2. Display the number of days in a week on which the AP123 bus is available
3. Find number of tickets booked for each PNR_No using GROUP BY CLAUSE
4. Find the distinct PNR Numbers that are present.

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

```
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));
```

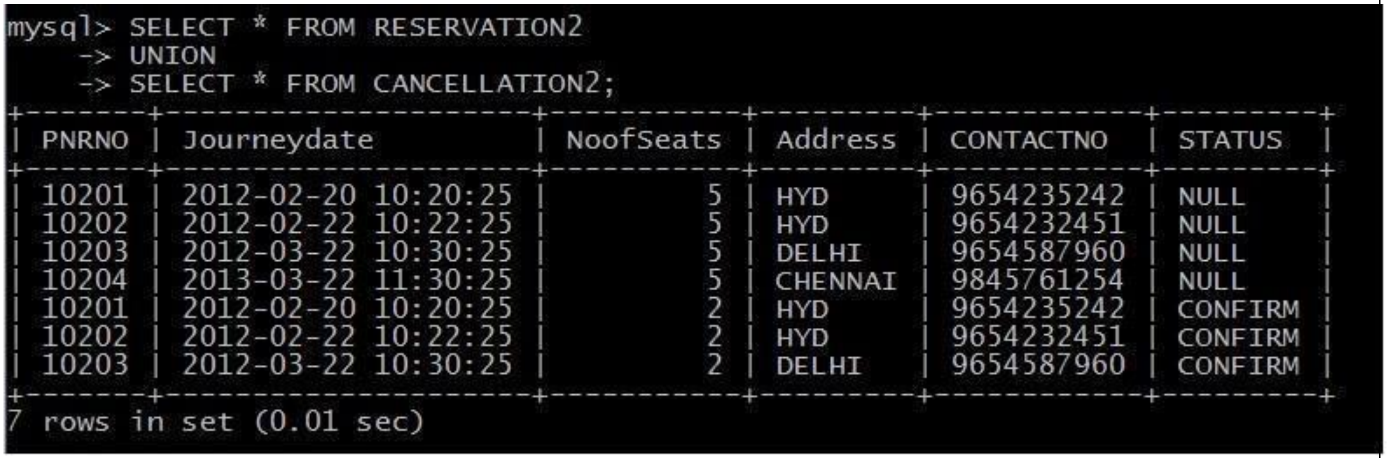
```
mysql> INSERT INTO CANCELLATION2 VALUES(10201,'2012-02-20
10:20:25',2,'HYD',9654235242,'CONFIRM');
```

```
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

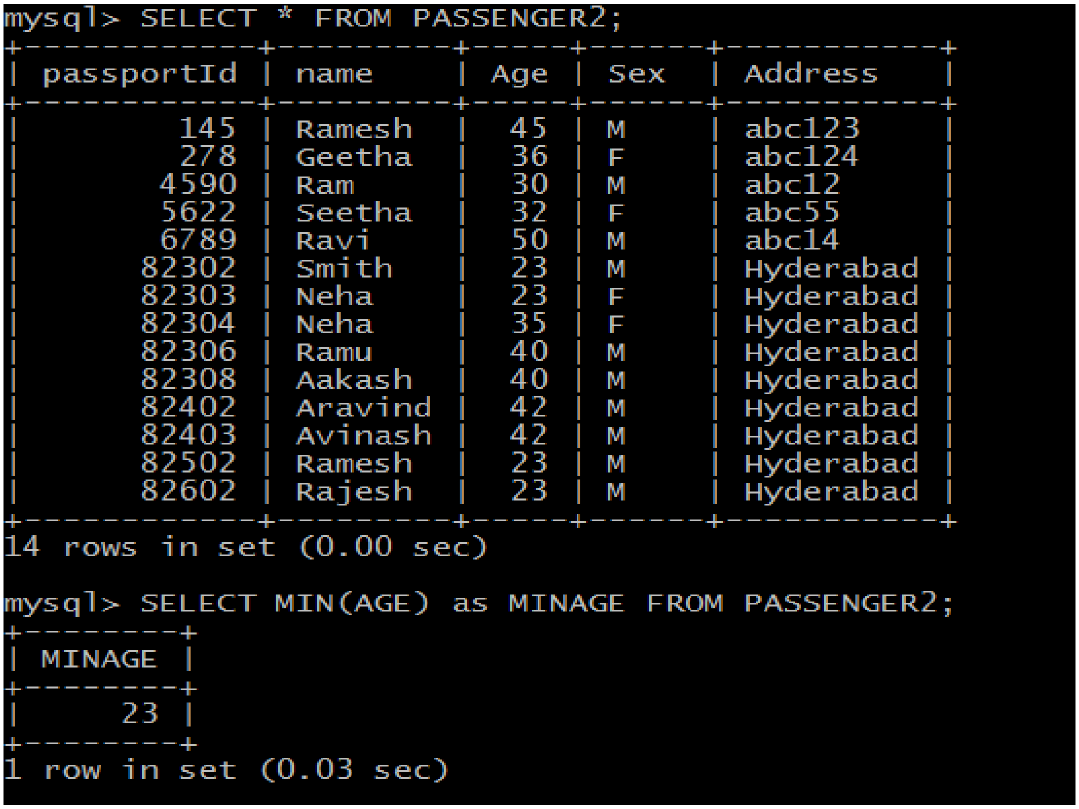
SELECT * FROM CANCELLATION;



```
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. Display the Minimum age of the Passenger

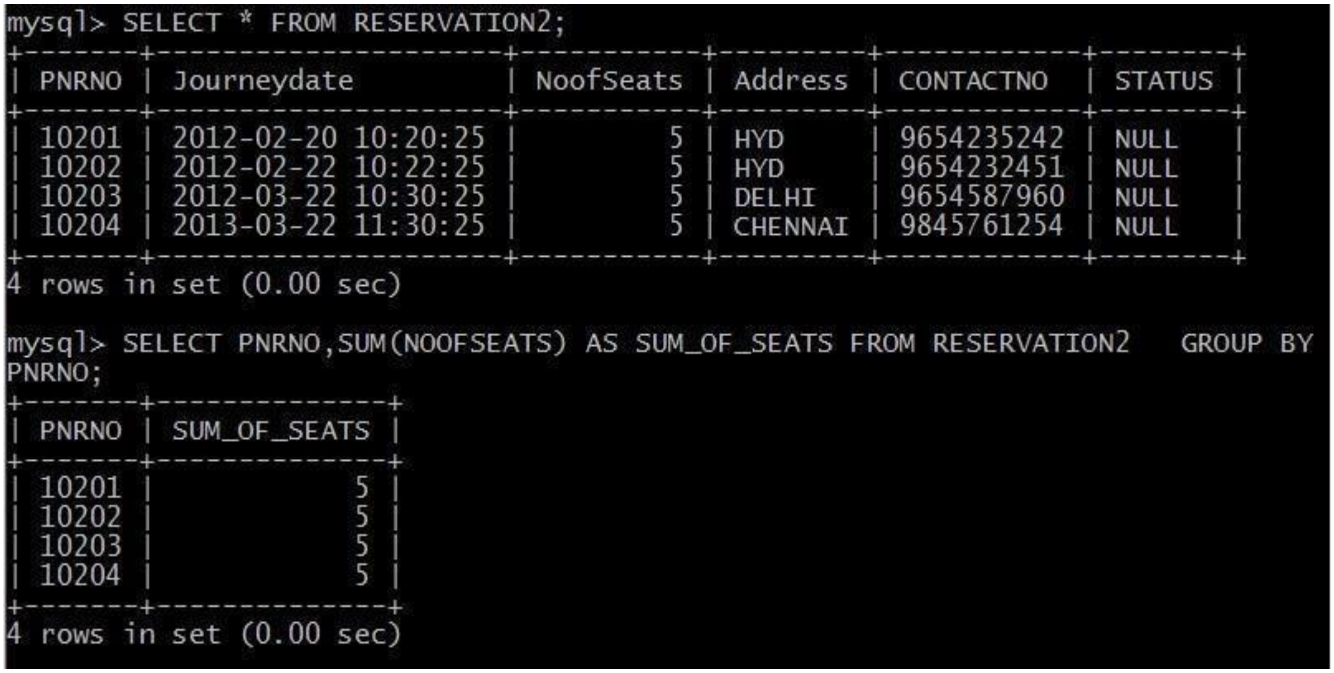
MySQL> SELECT MIN(AGE) as MINAGE FROM PASSENGER;



```
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. Find number of tickets booked for each PNR_No using GROUP BY CLAUSE

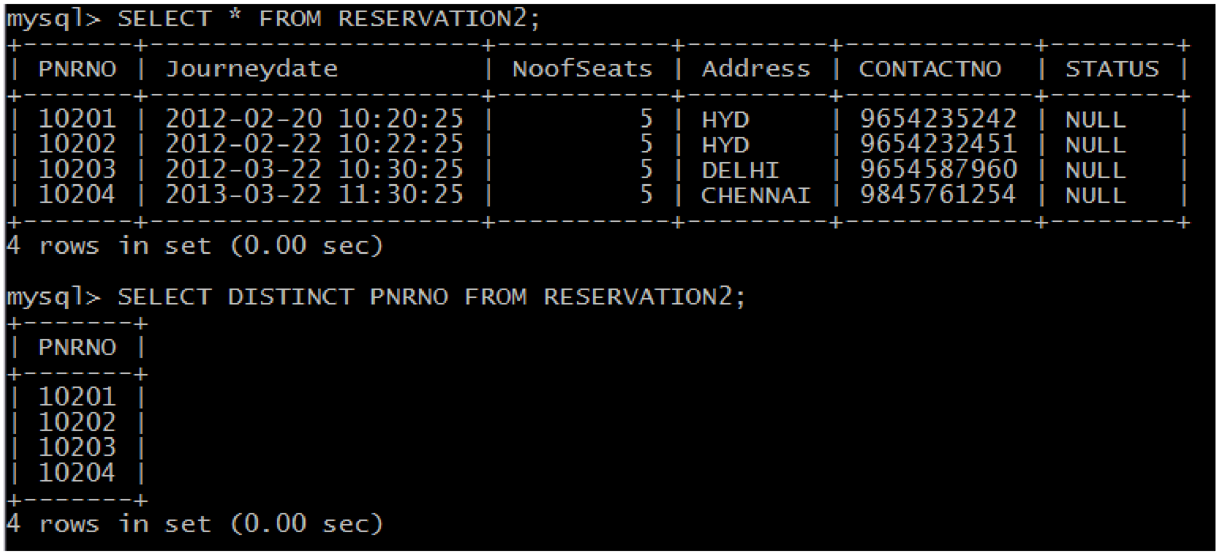
```
MySQL> SELECT PNRNO,SUM(No_of_SEATS) AS SUM_OF_SEATS FROM
RESERVATION2           GROUP BY PNRNO;
```



```
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)
```

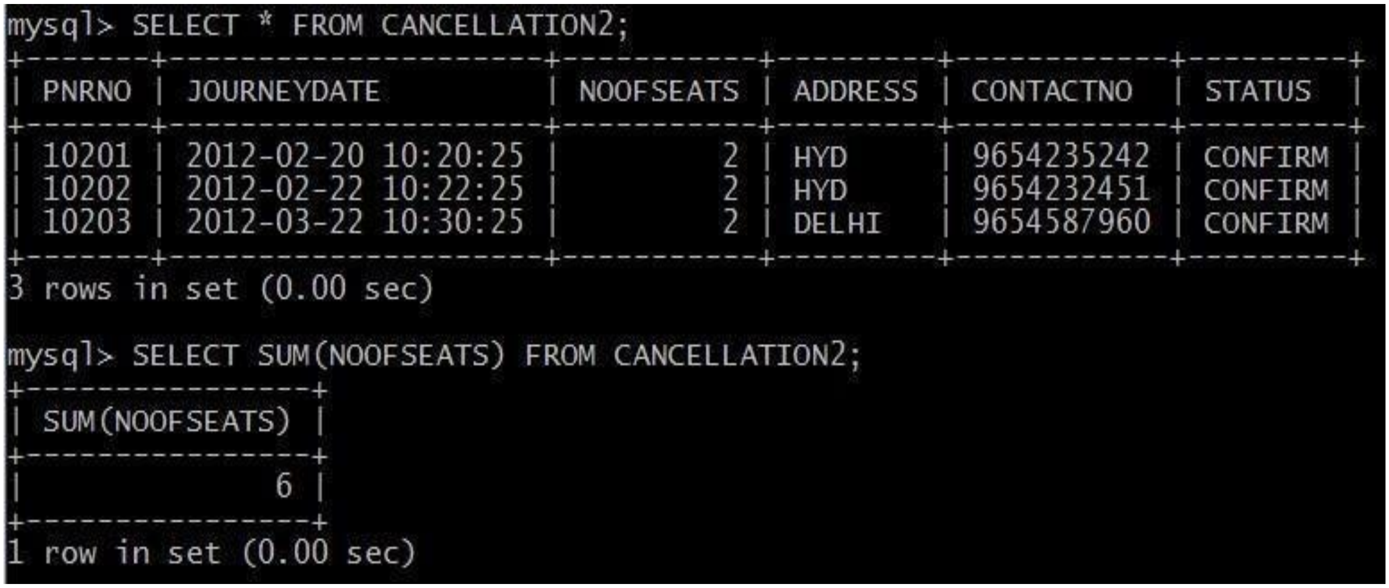
4 Find the distinct PNR Numbers that are present.

```
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)
```

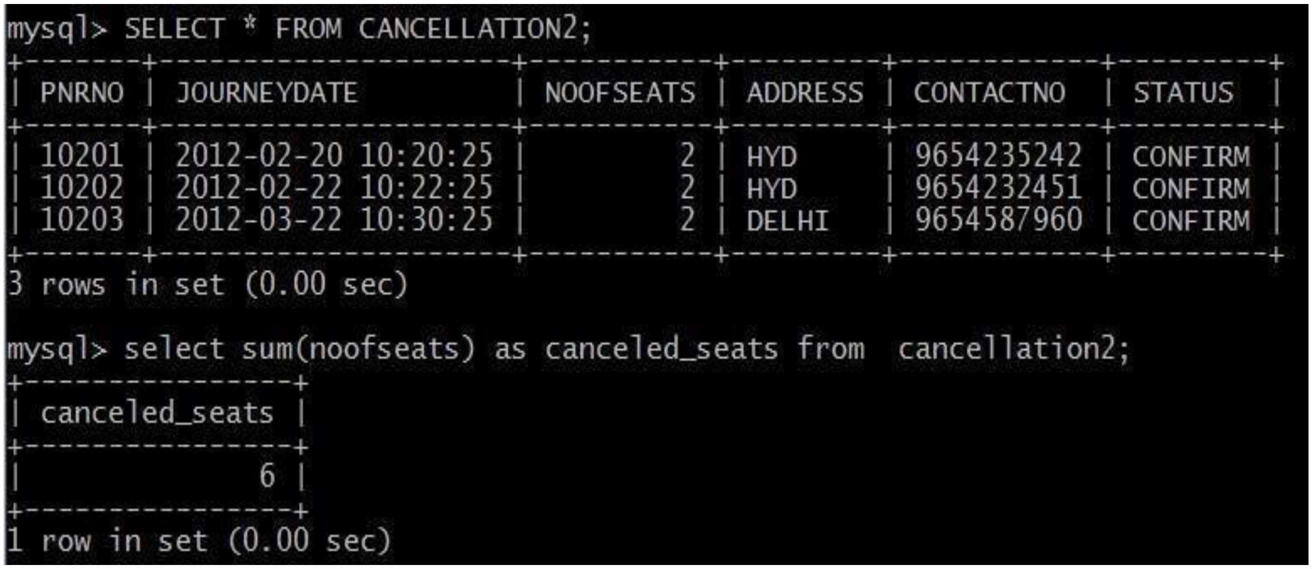
5 Mysql> select sum(Noofseats) from Cancellation2;



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

6 Find the total number of cancelled seats.

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)
```

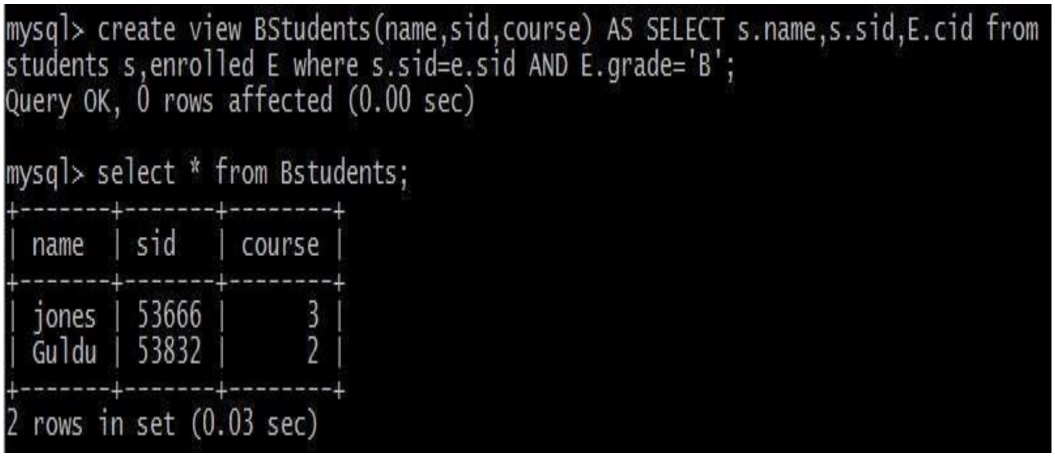
Creation and Droping of Views

mysql> create table students(sid int primary key,name varchar(15),login varchar(15), age int,gpa real); mysql> create table Enrolled(sid int,cid int,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

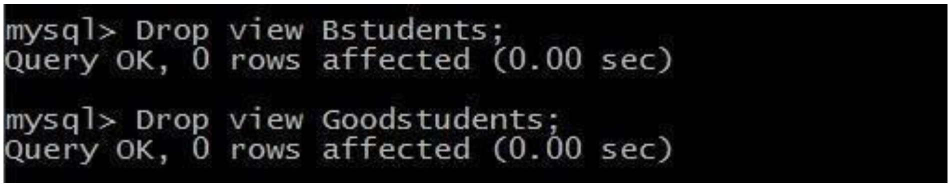
E.grade='B';



```
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.

Database Triggers:

Trigger defines an action the database should take when some database-related event occurs. Triggers may be used to supplement declarative referential integrity, to enforce complex business rules, or to audit changes to data. The code within a trigger, called a trigger body, is made up of PL/SQL blocks. It's like a stored procedure that is fired when an insert, update or delete command is issued against associated table.

Triggers can be executed, or fired, in response to the following events:
A row is inserted into a table
A row in a table is updated A
row in a table is deleted

It is not possible to define a trigger to fire when a row is selected.

.

Types of Triggers:

A trigger 's type is defined by the type of triggering transaction and by the level at which the trigger is executed. In the following sections, you will see descriptions of these classifications, along with relevant restrictions.

Row-Level Triggers:

Row-level triggers execute once for each row in a transaction. Row-level triggers are the most commontype of trigger; they are often used in data auditing applications.

Statement-Level Triggers:

Statement-level triggers execute once for each transaction. For example, if a single transaction inserted 500 rows into a table, then a statement-level trigger on that table would only be executed once.

INSTEAD OF Triggers:

You can use INSTEAD OF triggers to tell Oracle what to do instead of performing the actions that invoked the trigger. For example, you could use an INSTEAD OF trigger on a view to redirect inserts into table or to update multiple tables that are part of a view. You can use INSTEAD OF triggers on either object views or relational views.

Uses of Triggers:

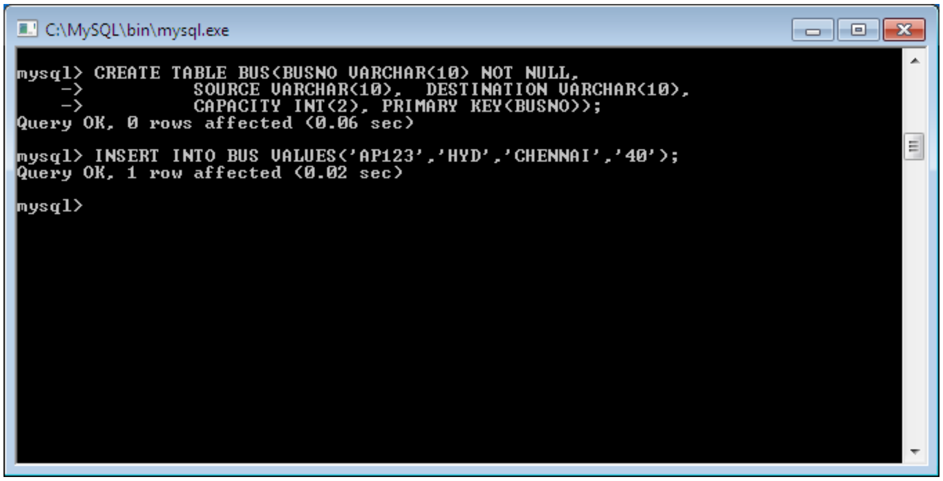
The possible uses for database triggers are varied and are limited only by your imagination. Somecommon uses are listed below:

- Enforcing business rules
- Maintaining referential integrity
- Enforcing security

- Maintaining a historical log of changes

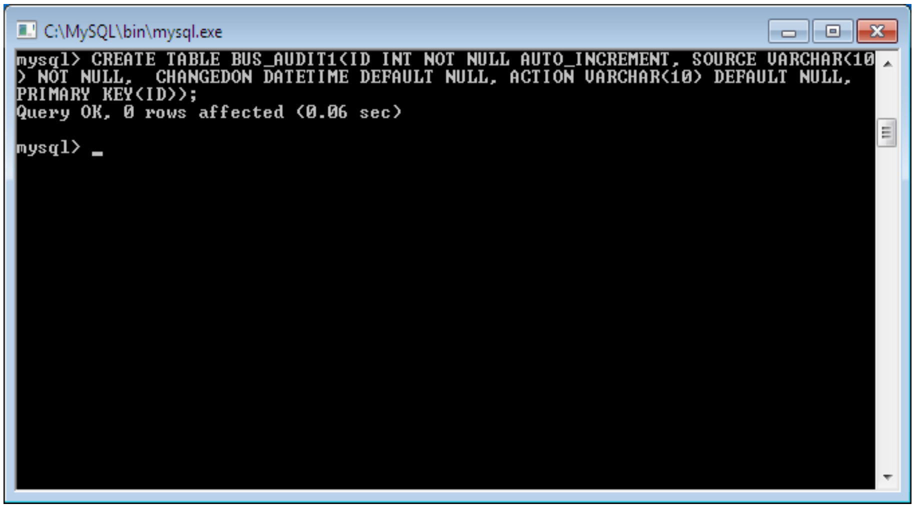
MySQL>CREATE TABLE BUS(BUSNO VARCHAR(10) NOT NULL, SOURCE VARCHAR(10), DESTINATION VARCHAR(10), CAPACITY INT(2), PRIMARY KEY(BUSNO));

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



```
mysql> CREATE TABLE BUS(BUSNO VARCHAR(10) NOT NULL,
    ->          SOURCE VARCHAR(10),  DESTINATION VARCHAR(10),
    ->          CAPACITY INT(2), PRIMARY KEY(BUSNO));
Query OK, 0 rows affected (0.06 sec)

mysql> INSERT INTO BUS VALUES('AP123','HYD','CHENNAI','40');
Query OK, 1 row affected (0.02 sec)

mysql>
```

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));



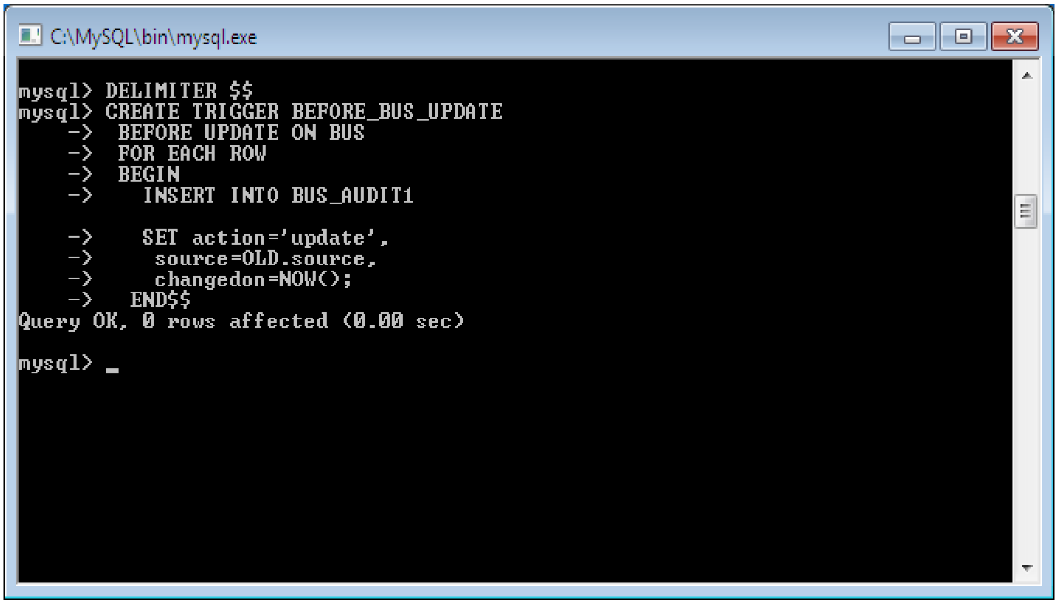
```
mysql> CREATE TABLE BUS_AUDIT1(ID INT NOT NULL AUTO_INCREMENT, SOURCE VARCHAR(10
) NOT NULL,  CHANGEDON DATETIME DEFAULT NULL, ACTION VARCHAR(10) DEFAULT NULL,
PRIMARY KEY(ID));
Query OK, 0 rows affected (0.06 sec)

mysql> _
```

CREATE TRIGGER BEFORE_BUS_UPDATE BEFORE UPDATE ON BUS

FOR EACH ROW BEGIN

INSERT INTO BUS_AUDIT1

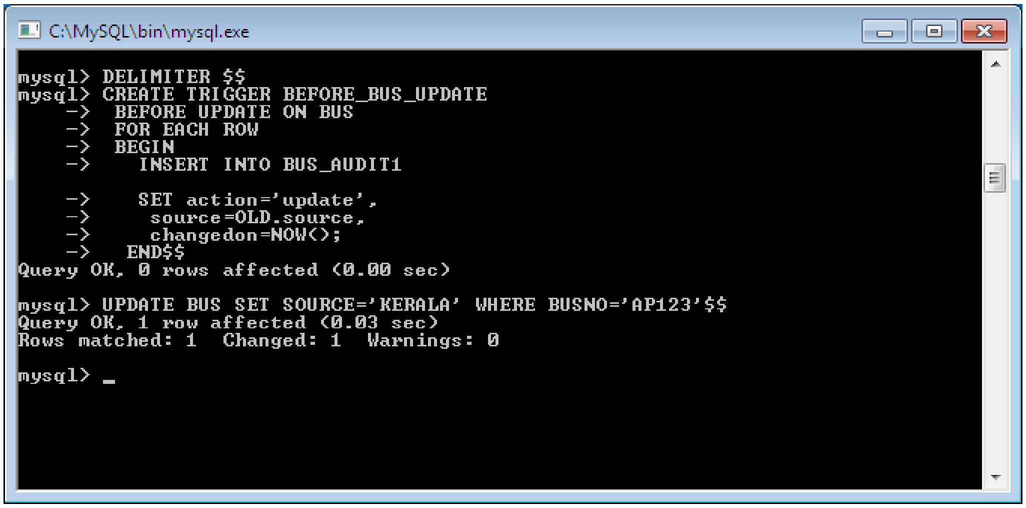
SET action='update', source=OLD.source, changedon=NOW(); END$$




```
mysql> DELIMITER $$
mysql> 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$$
Query OK, 0 rows affected (0.00 sec)

mysql> _
```

UPDATE :

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



```
mysql> DELIMITER $$
mysql> 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$$
Query OK, 0 rows affected (0.00 sec)

mysql> UPDATE BUS SET SOURCE='KERALA' WHERE BUSNO='AP123'$$
Query OK, 1 row affected (0.03 sec)
Rows matched: 1  Changed: 1  Warnings: 0

mysql> _
```

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:

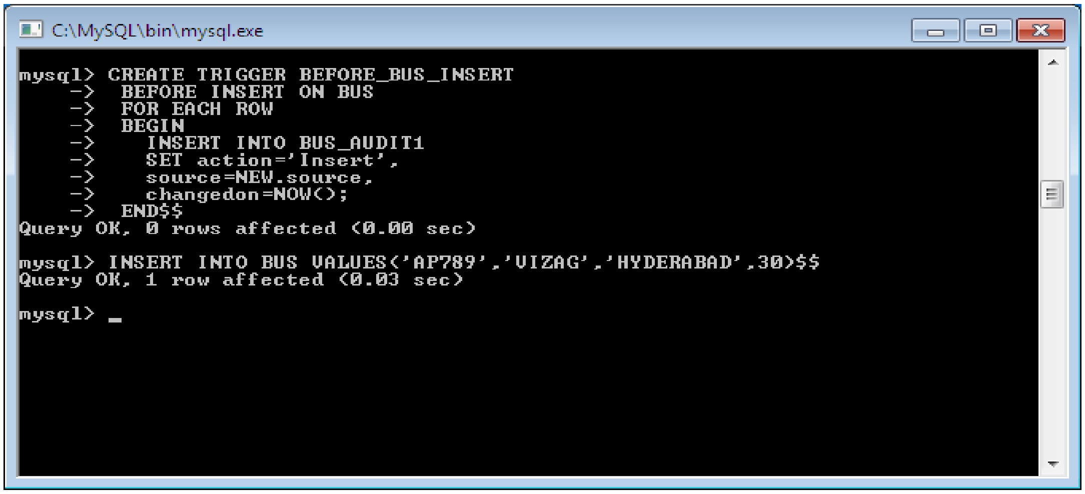
CREATE TRIGGER BEFORE_BUS_INSERT BEFORE INSERT ON BUS

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)$$



```
mysql> 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$$
Query OK, 0 rows affected (0.00 sec)

mysql> INSERT INTO BUS VALUES('AP789','VIZAG','HYDERABAD',30)$$
Query OK, 1 row affected (0.03 sec)

mysql> _
```

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

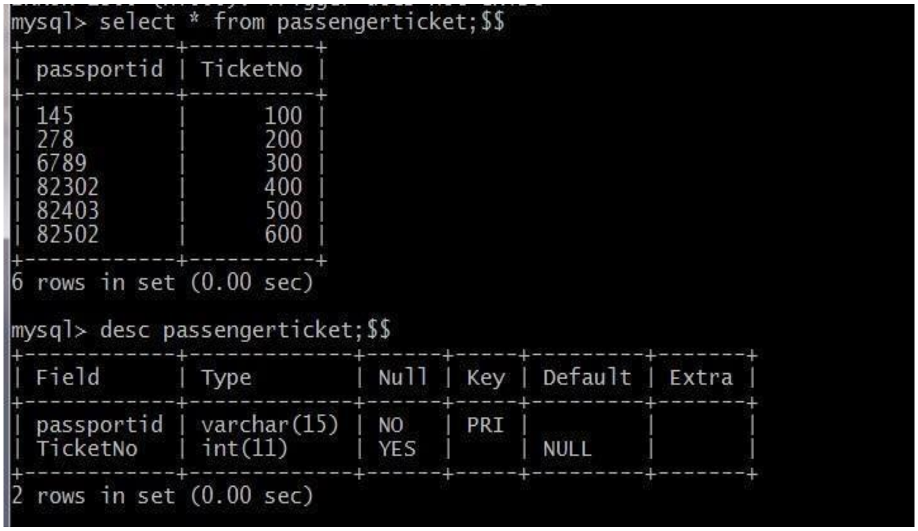
BEGIN

IF NEW.TicketNO > 60 THEN

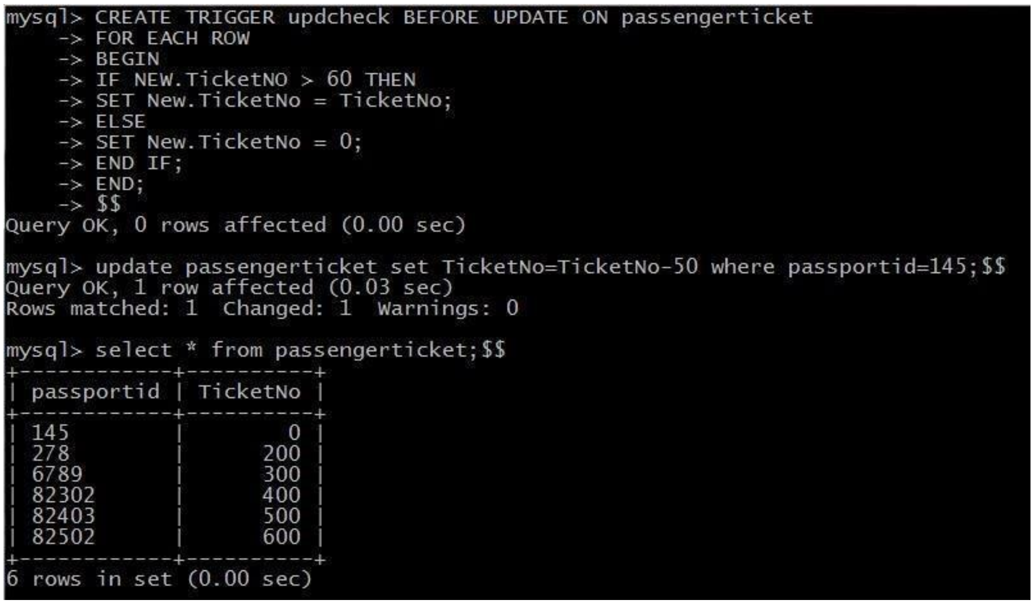
SET New.TicketNo = New.TicketNo; ELSE

SET New.TicketNo = 0; END IF;

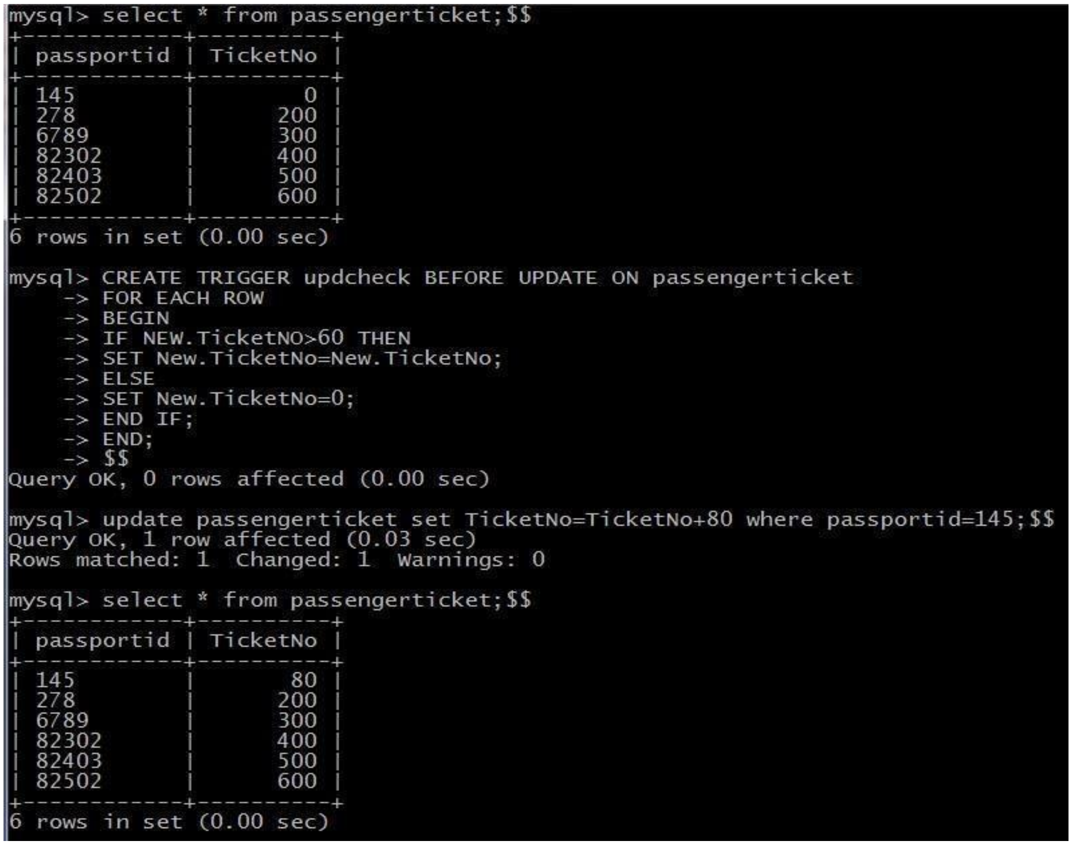
END;



```
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.

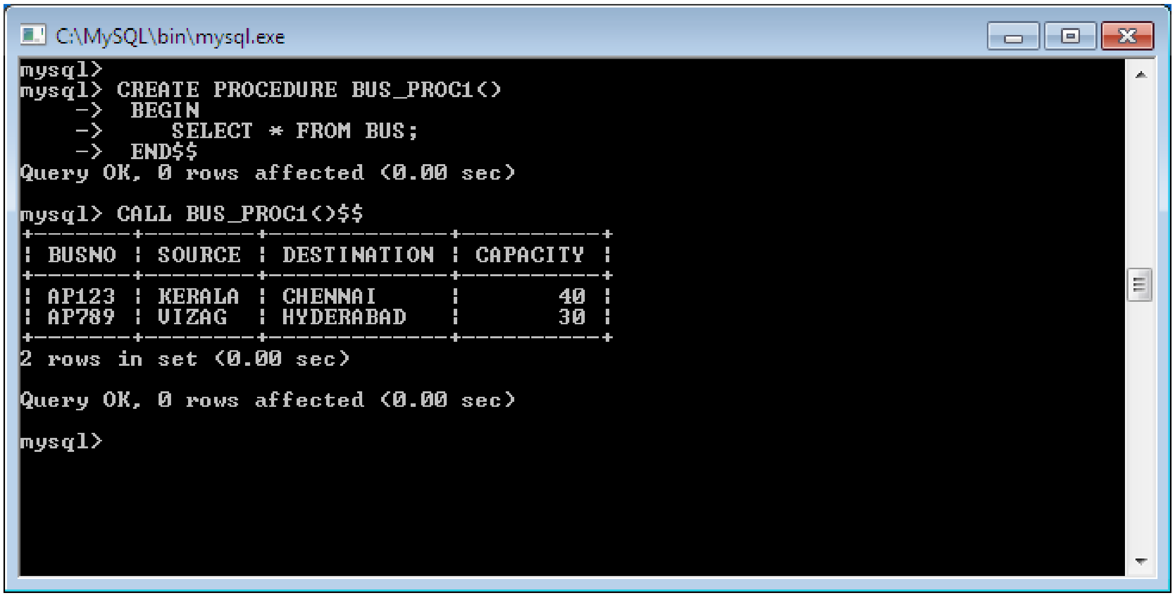
Ex1:

CREATE PROCEDURE BUS_PROC1() BEGIN

SELECT * FROM BUS;

END$$

CALL BUS_PROC1()$$

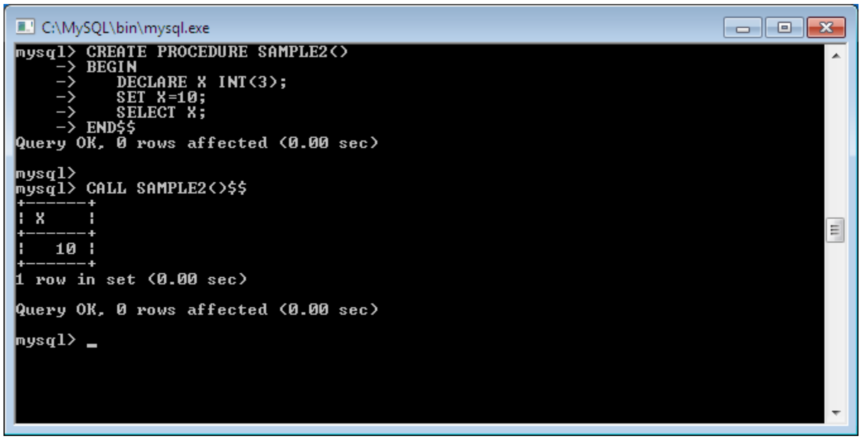


```
mysql>
mysql> CREATE PROCEDURE BUS_PROC1()
    ->  BEGIN
    ->     SELECT * FROM BUS;
    ->  END$$
Query OK, 0 rows affected (0.00 sec)

mysql> CALL BUS_PROC1()$$
+-------+--------+-------------+----------+
| BUSNO | SOURCE | DESTINATION | CAPACITY |
+-------+--------+-------------+----------+
| AP123 | KERALA | CHENNAI     |       40 |
| AP789 | VIZAG  | HYDERABAD   |       30 |
+-------+--------+-------------+----------+
2 rows in set (0.00 sec)

Query OK, 0 rows affected (0.00 sec)

mysql>
```

Ex2:

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

END$$
Mysql> CALL SAMPLE2()$$



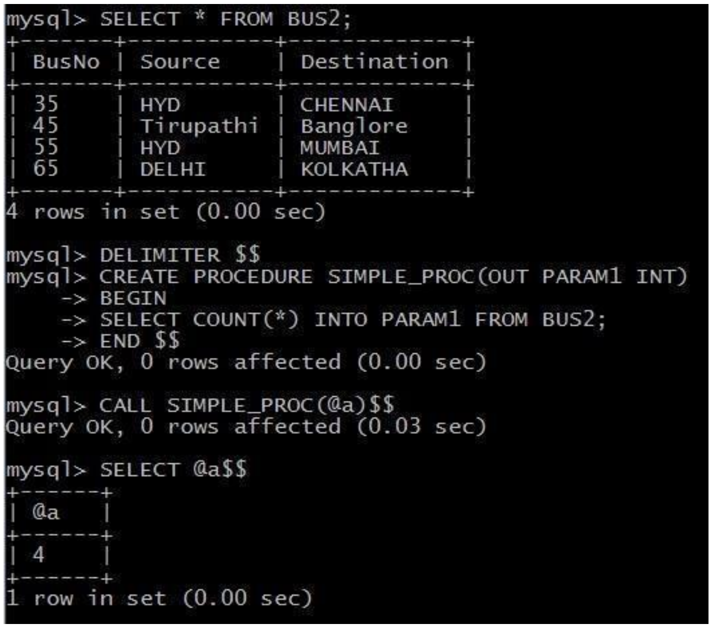
```
mysql> CREATE PROCEDURE SAMPLE2()
    -> BEGIN
    ->    DECLARE X INT(3);
    ->    SET X=10;
    ->    SELECT X;
    -> END$$
Query OK, 0 rows affected (0.00 sec)

mysql>
mysql> CALL SAMPLE2()$$
+------+
| X    |
+------+
|   10 |
+------+
1 row in set (0.00 sec)

Query OK, 0 rows affected (0.00 sec)

mysql> _
```

Ex3: CREATE PROCEDURE SIMPLE_PROC(OUT PARAM1 INT) BEGIN

SELECT COUNT(*) INTO PARAM1 FROM BUS;

END$$

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



```
mysql> SELECT * FROM BUS2;
+-------+-----------+-------------+
| BusNo | Source    | Destination |
+-------+-----------+-------------+
| 35    | HYD       | CHENNAI     |
| 45    | Tirupathi | Banglore    |
| 55    | HYD       | MUMBAI      |
| 65    | DELHI     | KOLKATHA    |
+-------+-----------+-------------+
4 rows in set (0.00 sec)

mysql> DELIMITER $$
mysql> CREATE PROCEDURE SIMPLE_PROC(OUT PARAM1 INT)
    -> BEGIN
    -> SELECT COUNT(*) INTO PARAM1 FROM BUS2;
    -> END $$
Query OK, 0 rows affected (0.00 sec)

mysql> CALL SIMPLE_PROC(@a)$$
Query OK, 0 rows affected (0.03 sec)

mysql> SELECT @a$$
+------+
| @a   |
+------+
| 4    |
+------+
1 row in set (0.00 sec)
```

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 require it.

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 one time.

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

- □ Declare a cursor
- □ Open a cursor statement
- □ Fetch the cursor
- □ Close the cursor

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.If we want to fetch rows from it you must open thecursor.

Syntax : OPEN cursor_name;

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

Synatx : FETCH cursor_name INTO var_name;

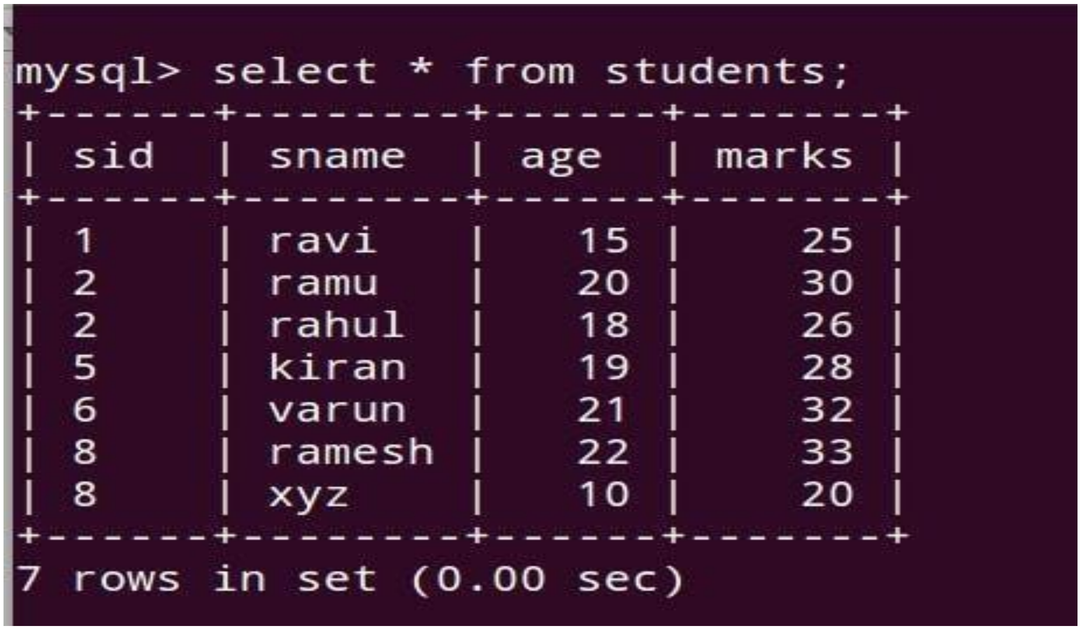
If any row exists, then the above statement fetches the next row and cursor pointer moves ahead to the next row.

4 . Cursor close statement : By this statement closed the open cursor.

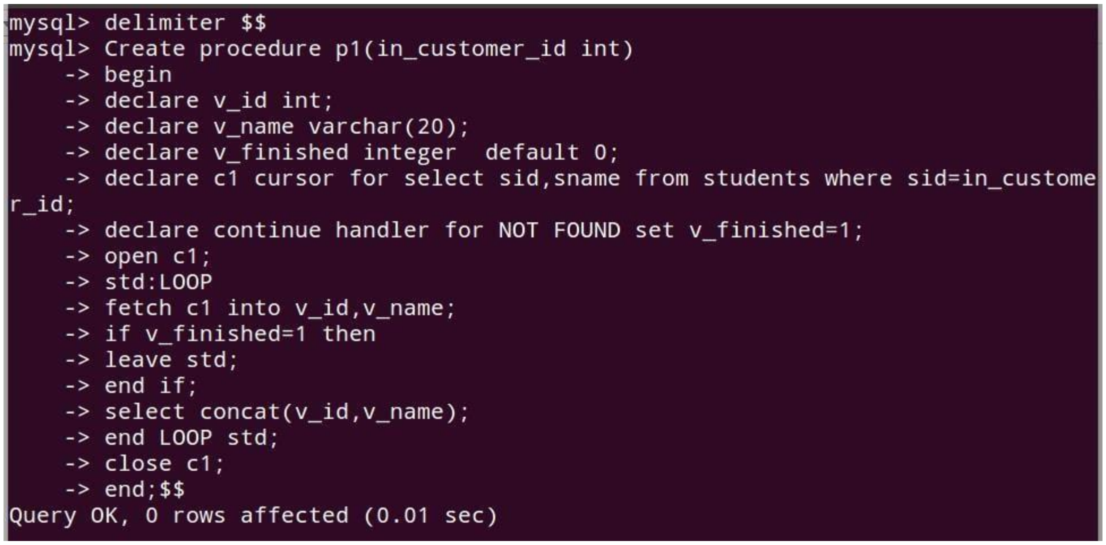
Syntax: CLOSE_name;

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)
```



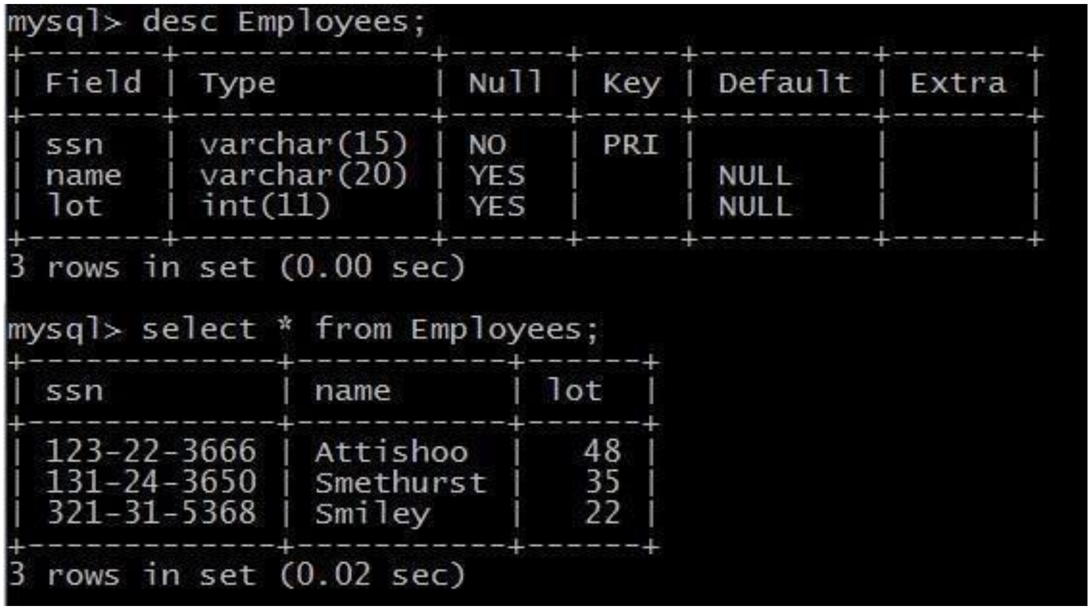
```
mysql> delimiter $$
mysql> Create procedure p1(in_customer_id int)
    -> begin
    -> declare v_id int;
    -> declare v_name varchar(20);
    -> declare v_finished integer  default 0;
    -> declare c1 cursor for select sid,sname from students where sid=in_custome
r_id;
    -> declare continue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

ADDITIONAL PROGRAMMS

EMPLOYEES TABLE

mysql> create table Employees(ssn varchar(15),name varchar(20),lot int,PRIMARY KEY(ssn)); mysql> insert into Employees values('123-22-3666','Attishoo',48);

mysql> insert into Employees values('321-31-5368','Smiley',22); mysql> insert into Employees values('131-24-3650','Smethurst',35);



```
mysql> desc Employees;
+-------+-------------+------+-----+---------+-------+
| Field | Type        | Null | Key | Default | Extra |
+-------+-------------+------+-----+---------+-------+
| ssn   | varchar(15) | NO   | PRI |         |       |
| name  | varchar(20) | YES  |     | NULL    |       |
| lot   | int(11)     | YES  |     | NULL    |       |
+-------+-------------+------+-----+---------+-------+
3 rows in set (0.00 sec)

mysql> select * from Employees;
+-------------+-----------+------+
| ssn         | name      | lot  |
+-------------+-----------+------+
| 123-22-3666 | Attishoo  |   48 |
| 131-24-3650 | Smethurst |   35 |
| 321-31-5368 | Smiley    |   22 |
+-------------+-----------+------+
3 rows in set (0.02 sec)
```

DEPARTMENT TABLE

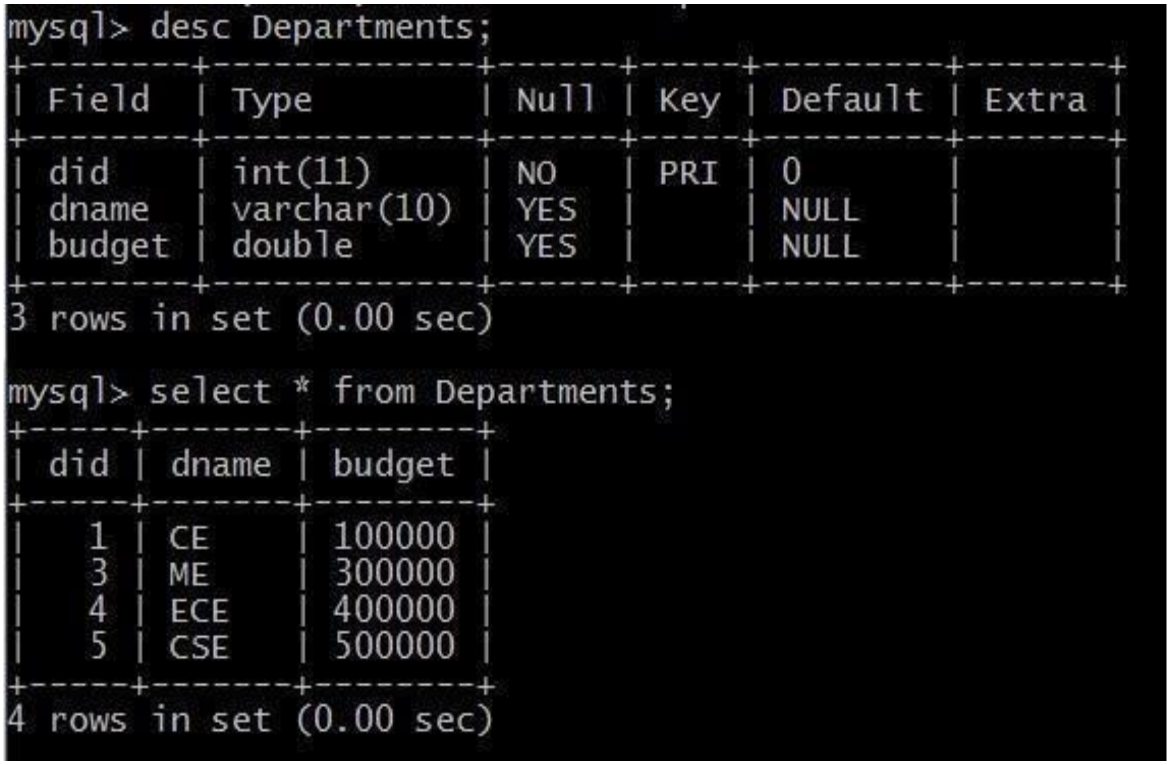
mysql> create table Departments(did int,dname varchar(10),budget real, PRIMARY KEY(did));

mysql> insert into Departments values(05,'CSE',500000);

mysql> insert into Departments values(04,'ECE',400000);

mysql> insert into Departments values(03,'ME',300000);

mysql> insert into Departments values(01,'CE',100000);

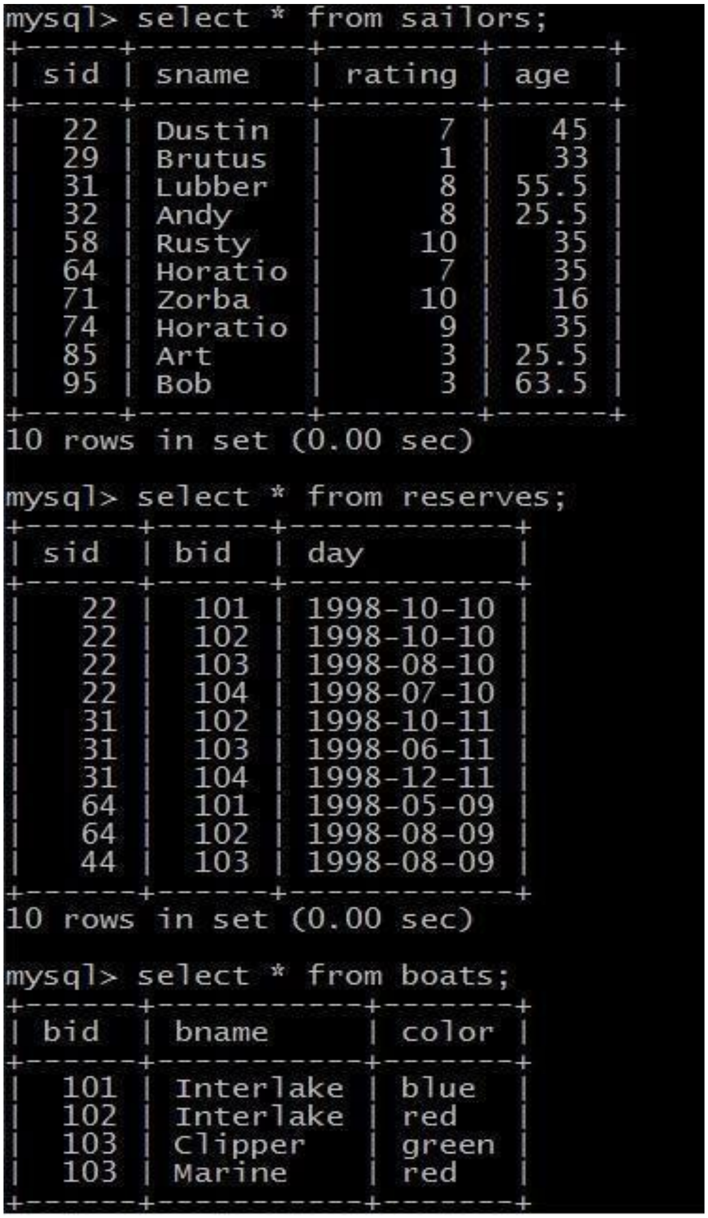


```
mysql> desc Departments;
+--------+-------------+------+-----+---------+-------+
| Field  | Type        | Null | Key | Default | Extra |
+--------+-------------+------+-----+---------+-------+
| did    | int(11)     | NO   | PRI | 0       |       |
| dname  | varchar(10) | YES  |     | NULL    |       |
| budget | double      | YES  |     | NULL    |       |
+--------+-------------+------+-----+---------+-------+
3 rows in set (0.00 sec)

mysql> select * from Departments;
+-----+-------+--------+
| did | dname | budget |
+-----+-------+--------+
|   1 | CE    | 100000 |
|   3 | ME    | 300000 |
|   4 | ECE   | 400000 |
|   5 | CSE   | 500000 |
+-----+-------+--------+
4 rows in set (0.00 sec)
```

Sailors , Reserves , Boats Tables

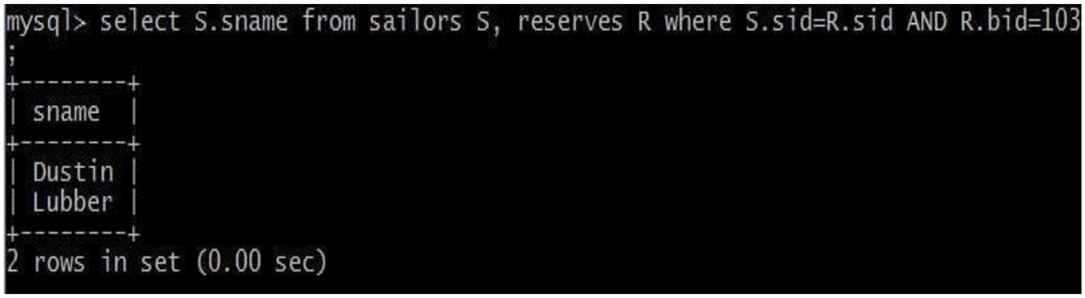
Mysql> Create table Sailors(Sid integer PRIMARY KEY,sname varchar(15), rating int,age real); Mysql>Create table Reserves(Sid int,Bid int,Day Date);

Mysql>Create table Boats(Bid int,Bname varchar(15),Color varchar(15);



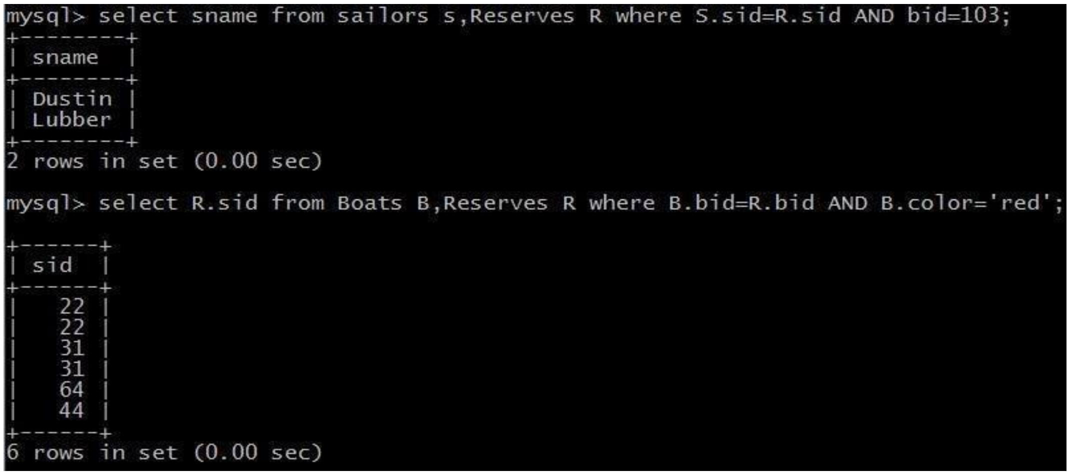
```
mysql> select * from sailors;
+-----+---------+--------+------+
| sid | sname   | rating | age  |
+-----+---------+--------+------+
|  22 | Dustin  |      7 |   45 |
|  29 | Brutus  |      1 |   33 |
|  31 | Lubber  |      8 | 55.5 |
|  32 | Andy    |      8 | 25.5 |
|  58 | Rusty   |     10 |   35 |
|  64 | Horatio |      7 |   35 |
|  71 | Zorba   |     10 |   16 |
|  74 | Horatio |      9 |   35 |
|  85 | Art     |      3 | 25.5 |
|  95 | Bob     |      3 | 63.5 |
+-----+---------+--------+------+
10 rows in set (0.00 sec)

mysql> select * from reserves;
+------+------+------------+
| sid  | bid  | day        |
+------+------+------------+
|   22 |  101 | 1998-10-10 |
|   22 |  102 | 1998-10-10 |
|   22 |  103 | 1998-08-10 |
|   22 |  104 | 1998-07-10 |
|   31 |  102 | 1998-10-11 |
|   31 |  103 | 1998-06-11 |
|   31 |  104 | 1998-12-11 |
|   64 |  101 | 1998-05-09 |
|   64 |  102 | 1998-08-09 |
|   44 |  103 | 1998-08-09 |
+------+------+------------+
10 rows in set (0.00 sec)

mysql> select * from boats;
+------+-----------+-------+
| bid  | bname     | color |
+------+-----------+-------+
|  101 | Interlake | blue  |
|  102 | Interlake | red   |
|  103 | Clipper   | green |
|  103 | Marine    | red   |
+------+-----------+-------+
```

mysql> select S.sname from sailors S, reserves R where S.sid=R.sid AND R.bid=103;



```
mysql> select S.sname from sailors S, reserves R where S.sid=R.sid AND R.bid=103
;
+--------+
| sname  |
+--------+
| Dustin |
| Lubber |
+--------+
2 rows in set (0.00 sec)
```

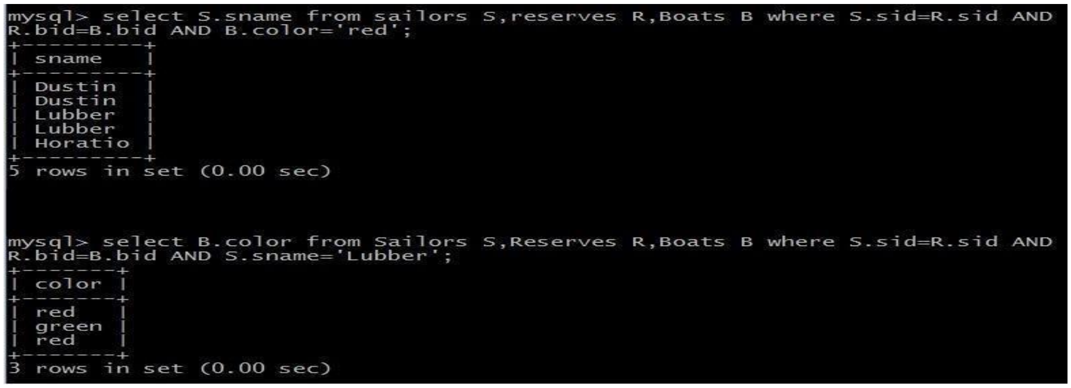
mysql> select sname from sailors s,Reserves R where S.sid=R.sid AND bid=103; mysql> select R.sid from Boats B,Reserves R where B.bid=R.bid AND B.color='red';



```
mysql> select sname from sailors s,Reserves R where S.sid=R.sid AND bid=103;
+--------+
| sname  |
+--------+
| Dustin |
| Lubber |
+--------+
2 rows in set (0.00 sec)

mysql> select R.sid from Boats B,Reserves R where B.bid=R.bid AND B.color='red';

+------+
| sid  |
+------+
|   22 |
|   22 |
|   31 |
|   31 |
|   64 |
|   44 |
+------+
6 rows in set (0.00 sec)
```

mysql> select S.sname from sailors S,reserves R,Boats B where S.sid=R.sid AND R.bid=B.bid AND B.color='red';

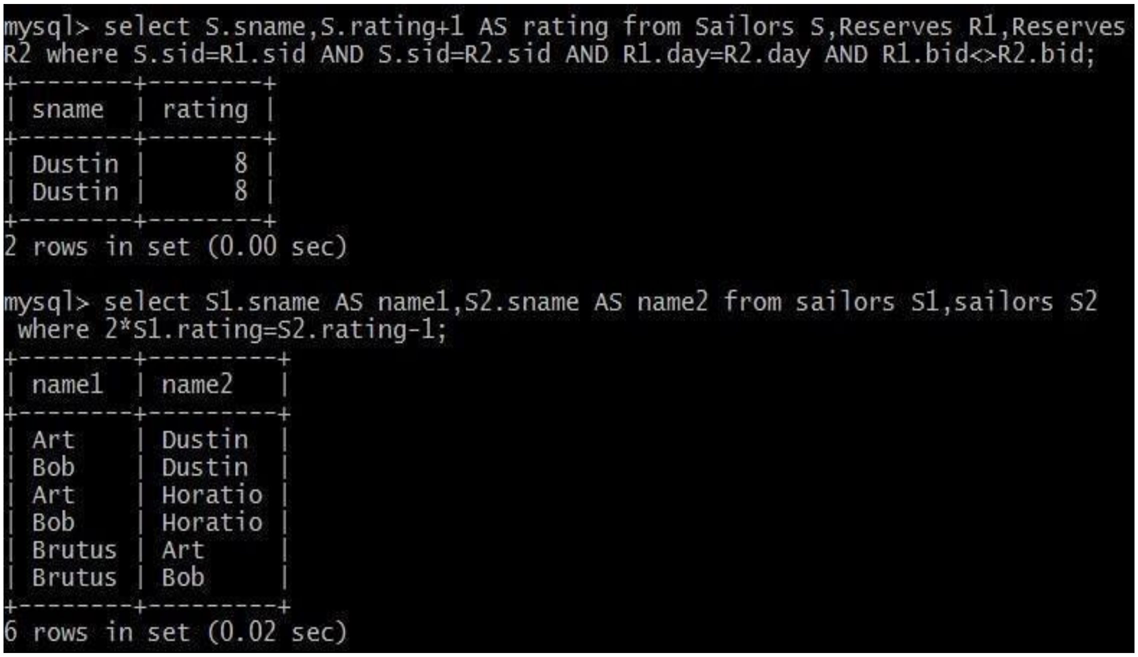
mysql> select B.color from Sailors S,Reserves R,Boats B where S.sid=R.sid AND R.bid=B.bid AND S.sname='Lubber';



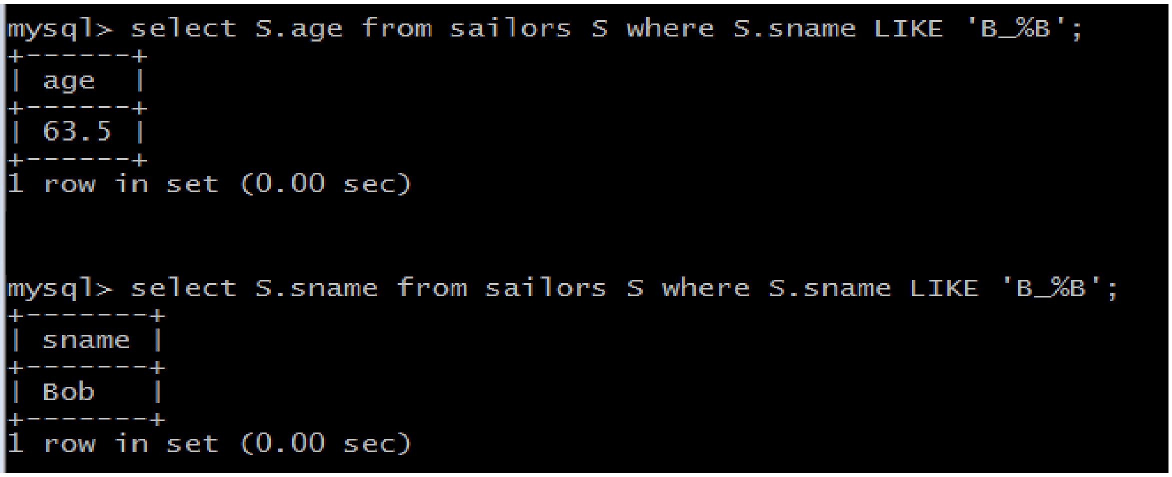
```
mysql> select S.sname from sailors S,reserves R,Boats B where S.sid=R.sid AND
R.bid=B.bid AND B.color='red';
+---------+
| sname   |
+---------+
| Dustin  |
| Dustin  |
| Lubber  |
| Lubber  |
| Horatio |
+---------+
5 rows in set (0.00 sec)


mysql> select B.color from Sailors S,Reserves R,Boats B where S.sid=R.sid AND
R.bid=B.bid AND S.sname='Lubber';
+-------+
| color |
+-------+
| red   |
| green |
| red   |
+-------+
3 rows in set (0.00 sec)
```

mysql> select S.sname,S.rating+1 AS rating from Sailors S,Reserves R1,Reserves R2 where S.sid=R1.sid AND S.sid=R2.sid AND R1.day=R2.day AND R1.bid<>R2.bid;

mysql> select S1.sname AS name1,S2.sname AS name2 from sailors S1,sailors S2 where 2*S1.rating=S2.rating-1;



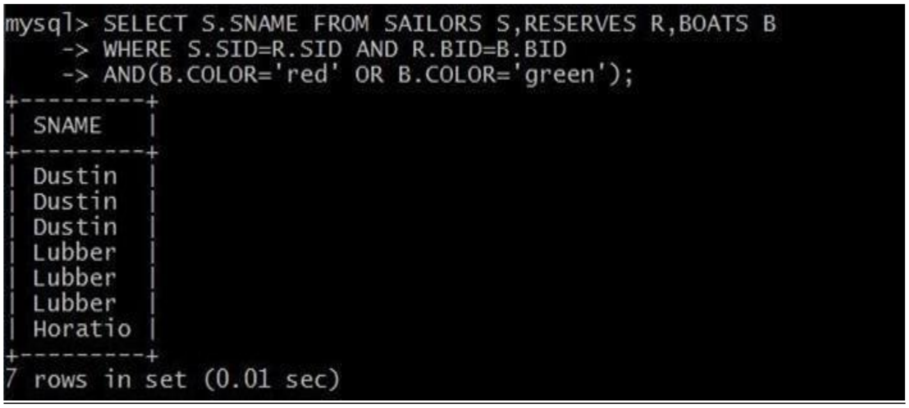
```
mysql> select S.sname,S.rating+1 AS rating from Sailors S,Reserves R1,Reserves
R2 where S.sid=R1.sid AND S.sid=R2.sid AND R1.day=R2.day AND R1.bid<>R2.bid;
+--------+--------+
| sname  | rating |
+--------+--------+
| Dustin |      8 |
| Dustin |      8 |
+--------+--------+
2 rows in set (0.00 sec)

mysql> select S1.sname AS name1,S2.sname AS name2 from sailors S1,sailors S2
 where 2*S1.rating=S2.rating-1;
+--------+---------+
| name1  | name2   |
+--------+---------+
| Art    | Dustin  |
| Bob    | Dustin  |
| Art    | Horatio |
| Bob    | Horatio |
| Brutus | Art     |
| Brutus | Bob     |
+--------+---------+
6 rows in set (0.02 sec)
```



```
mysql> select S.age from sailors S where S.sname LIKE 'B_%B';
+------+
| age  |
+------+
| 63.5 |
+------+
1 row in set (0.00 sec)


mysql> select S.sname from sailors S where S.sname LIKE 'B_%B';
+-------+
| sname |
+-------+
| Bob   |
+-------+
1 row in set (0.00 sec)
```

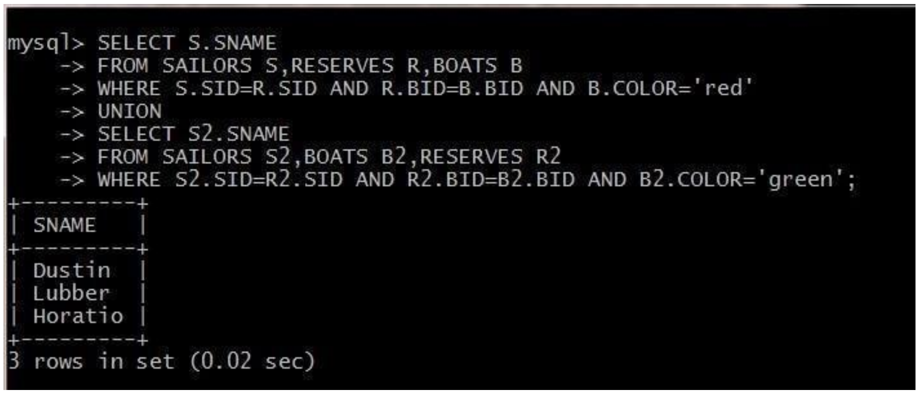
USING UNION , INTERSECT , AND EXCEPT

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



```
mysql> SELECT S.SNAME FROM SAILORS S,RESERVES R,BOATS B
    -> WHERE S.SID=R.SID AND R.BID=B.BID
    -> AND(B.COLOR='red' OR B.COLOR='green');
+---------+
| SNAME   |
+---------+
| Dustin  |
| Dustin  |
| Dustin  |
| Lubber  |
| Lubber  |
| Lubber  |
| Horatio |
+---------+
7 rows in set (0.01 sec)
```

OR



```
mysql> SELECT S.SNAME
    -> FROM SAILORS S,RESERVES R,BOATS B
    -> WHERE S.SID=R.SID AND R.BID=B.BID AND B.COLOR='red'
    -> UNION
    -> SELECT S2.SNAME
    -> FROM SAILORS S2,BOATS B2,RESERVES R2
    -> WHERE S2.SID=R2.SID AND R2.BID=B2.BID AND B2.COLOR='green';
+---------+
| SNAME   |
+---------+
| Dustin  |
| Lubber  |
| Horatio |
+---------+
3 rows in set (0.02 sec)
```

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

SELECT S.SNAME

FROM SAILORS S,RESERVES R,BOATS B

WHERE S.SID=R.SID AND R.BID=B.BID AND B.COLOR='red' INTERSECT

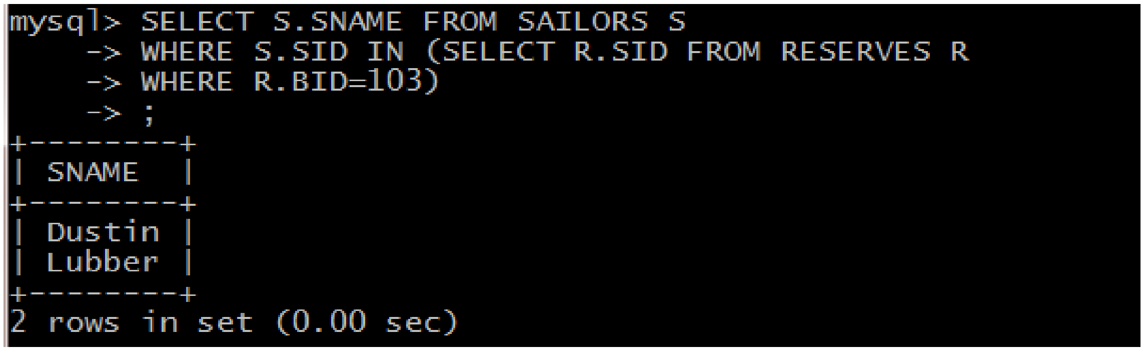
SELECT S2.SNAME

FROM SAILORS S2,RESERVES R2,BOATS B2

WHERE S2.SID=R2.SID AND R2.BID=B2.BID AND B2.COLOR='green';

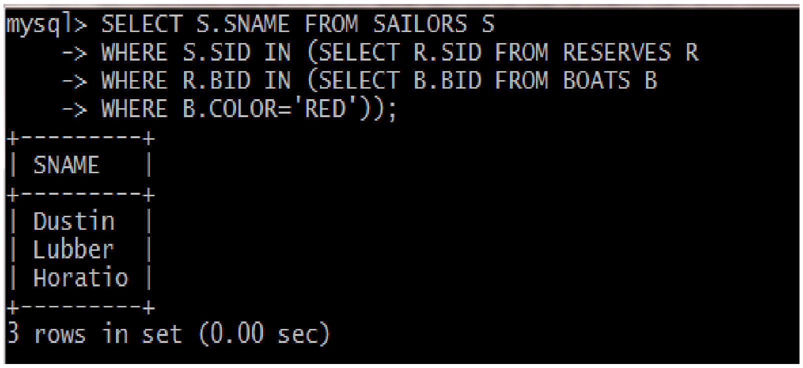
NESTED QUERIES

1) Find the Names of sailors who have reserved boat 103



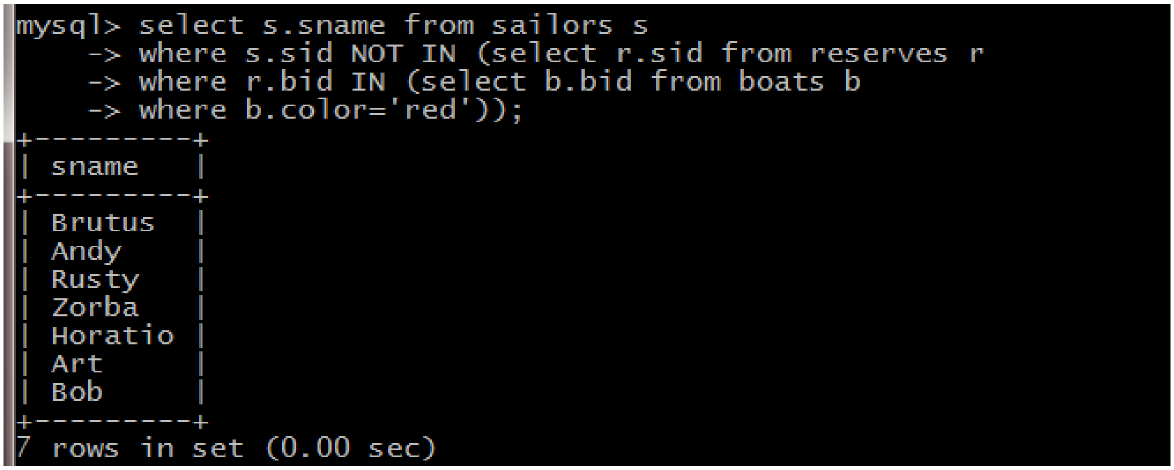
```
mysql> SELECT S.SNAME FROM SAILORS S
    -> WHERE S.SID IN (SELECT R.SID FROM RESERVES R
    -> WHERE R.BID=103)
    -> ;
+--------+
| SNAME  |
+--------+
| Dustin |
| Lubber |
+--------+
2 rows in set (0.00 sec)
```

2) Find the names of Sailors who have reserved a red Boat



```
mysql> SELECT S.SNAME FROM SAILORS S
    -> WHERE S.SID IN (SELECT R.SID FROM RESERVES R
    -> WHERE R.BID IN (SELECT B.BID FROM BOATS B
    -> WHERE B.COLOR='RED'));
+---------+
| SNAME   |
+---------+
| Dustin  |
| Lubber  |
| Horatio |
+---------+
3 rows in set (0.00 sec)
```

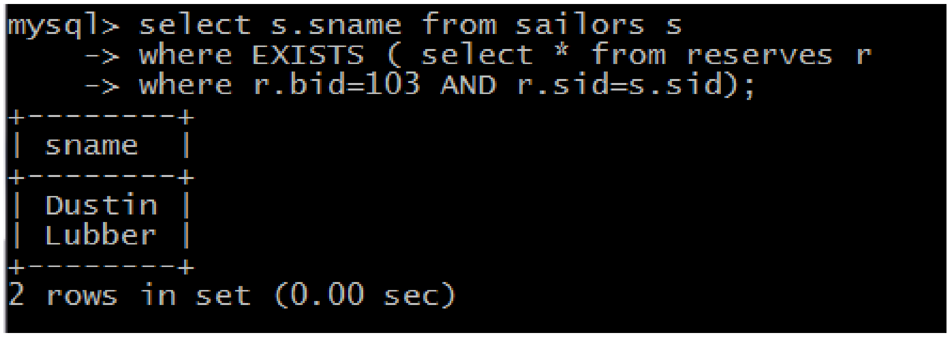
3) Find the names of Sailors who have NOT reserved a red Boat



```
mysql> select s.sname from sailors s
    -> where s.sid NOT IN (select r.sid from reserves r
    -> where r.bid IN (select b.bid from boats b
    -> where b.color='red'));
+---------+
| sname   |
+---------+
| Brutus  |
| Andy    |
| Rusty   |
| Zorba   |
| Horatio |
| Art     |
| Bob     |
+---------+
7 rows in set (0.00 sec)
```

Correlated Nested Queries:

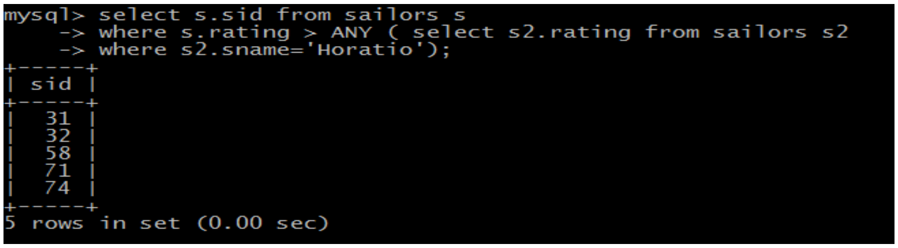
1) Find the names of Sailors who have reserved a red Boat



```
mysql> select s.sname from sailors s
    -> where EXISTS ( select * from reserves r
    -> where r.bid=103 AND r.sid=s.sid);
+--------+
| sname  |
+--------+
| Dustin |
| Lubber |
+--------+
2 rows in set (0.00 sec)
```

Set Comparison Operators:

1) Find sailors whose rating is better than some sailor called Horatio



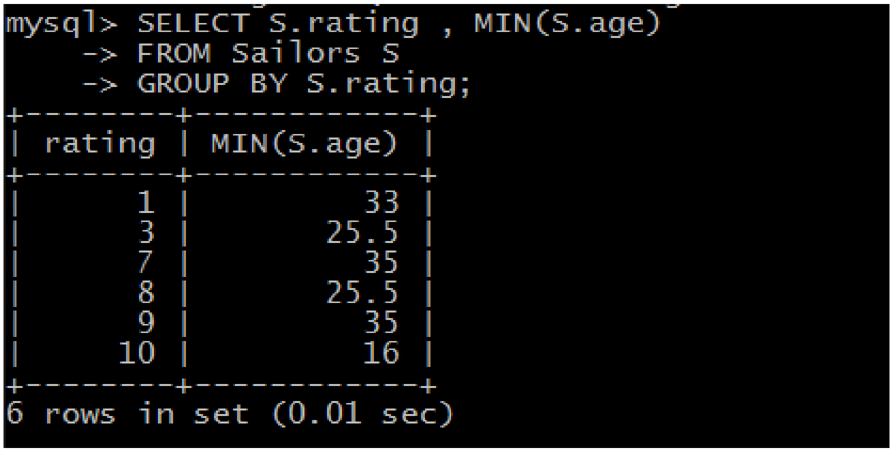
```
mysql> select s.sid from sailors s
    -> where s.rating > ANY ( select s2.rating from sailors s2
    -> where s2.sname='Horatio');
+-----+
| sid |
+-----+
|  31 |
|  32 |
|  58 |
|  71 |
|  74 |
+-----+
5 rows in set (0.00 sec)
```

2) Find the sailors with the highest rating.

mysql> SELECT S.sid FORM Sailors WHERE S.rating>=ALL(SELECT S2.rating FROM Sailors S2);

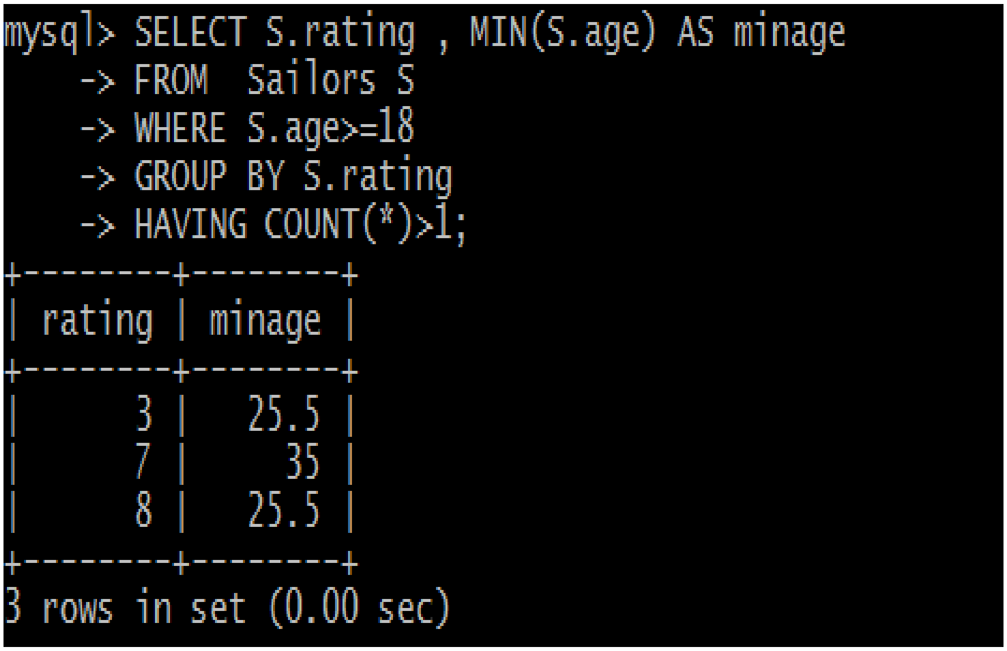
The GROUP BY and HAVING Clauses:

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



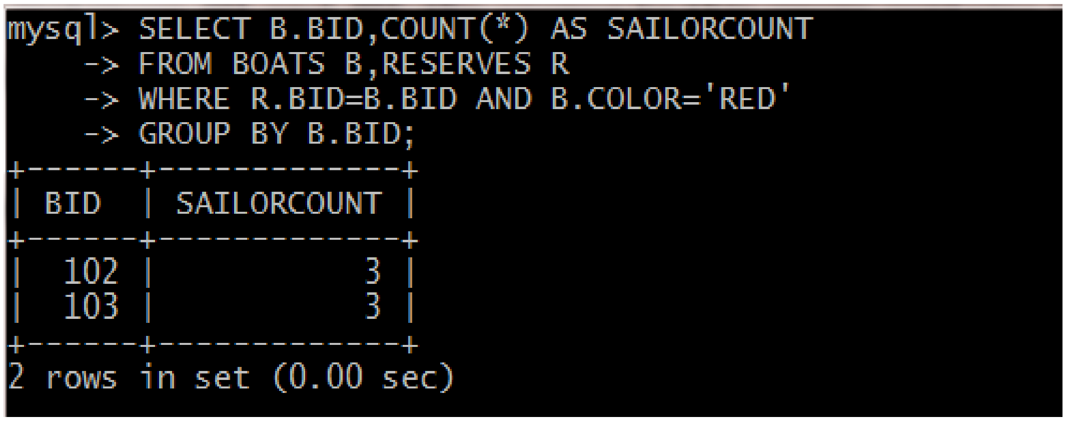
```
mysql> SELECT S.rating , MIN(S.age)
    -> FROM Sailors S
    -> GROUP BY S.rating;
+--------+------------+
| rating | MIN(S.age) |
+--------+------------+
|      1 |         33 |
|      3 |       25.5 |
|      7 |         35 |
|      8 |       25.5 |
|      9 |         35 |
|     10 |         16 |
+--------+------------+
6 rows in set (0.01 sec)
```

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



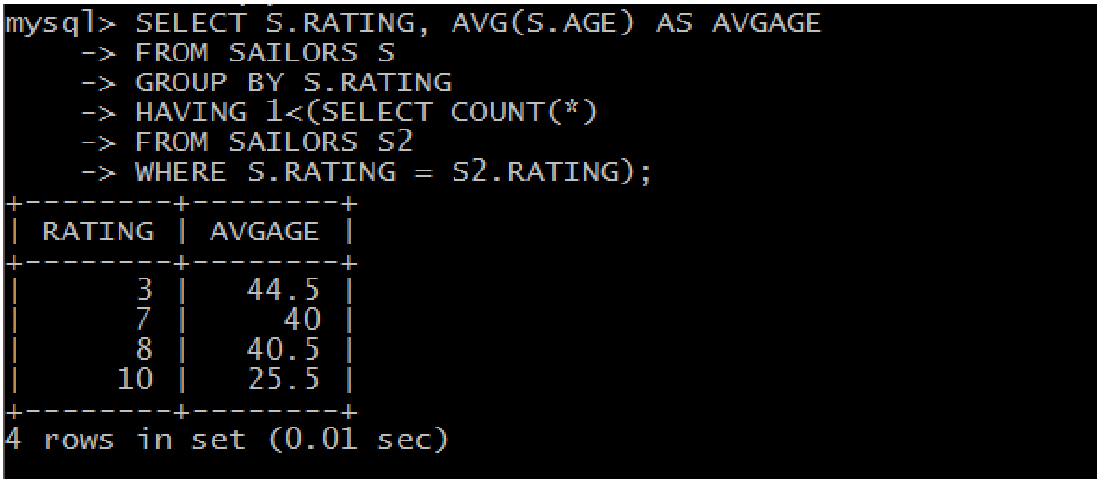
```
mysql> SELECT S.rating , MIN(S.age) AS minage
    -> FROM  Sailors S
    -> WHERE S.age>=18
    -> GROUP BY S.rating
    -> HAVING COUNT(*)>1;
+--------+--------+
| rating | minage |
+--------+--------+
|      3 |   25.5 |
|      7 |     35 |
|      8 |   25.5 |
+--------+--------+
3 rows in set (0.00 sec)
```

3) For each red boat , find the number of reservations for this boat



```
mysql> SELECT B.BID,COUNT(*) AS SAILORCOUNT
    -> FROM BOATS B,RESERVES R
    -> WHERE R.BID=B.BID AND B.COLOR='RED'
    -> GROUP BY B.BID;
+------+-------------+
| BID  | SAILORCOUNT |
+------+-------------+
|  102 |           3 |
|  103 |           3 |
+------+-------------+
2 rows in set (0.00 sec)
```

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



```
mysql> SELECT S.RATING, AVG(S.AGE) AS AVGAGE
    -> FROM SAILORS S
    -> GROUP BY S.RATING
    -> HAVING 1<(SELECT COUNT(*)
    -> FROM SAILORS S2
    -> WHERE S.RATING = S2.RATING);
+--------+--------+
| RATING | AVGAGE |
+--------+--------+
|      3 |   44.5 |
|      7 |     40 |
|      8 |   40.5 |
|     10 |   25.5 |
+--------+--------+
4 rows in set (0.01 sec)
```