# CS407PC: DATABASE MANAGEMENT SYSTEMS LAB

B.Tech. II Year I Sem. L T P C 0 0 3 1.5

**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

### EXPERIMENT- 1

**CONCEPT DESIGN WITH E-R MODEL**

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

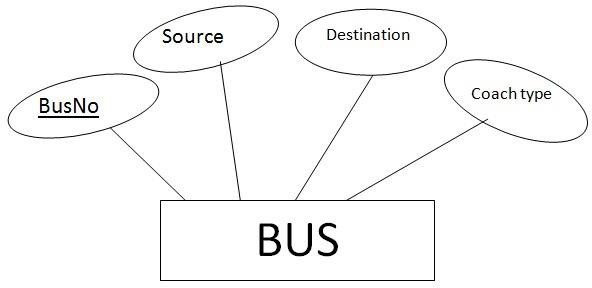
**E-R Model**

# Bus

* BusNo
* Source
* Destination
* CoachType

# SCHEMA

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

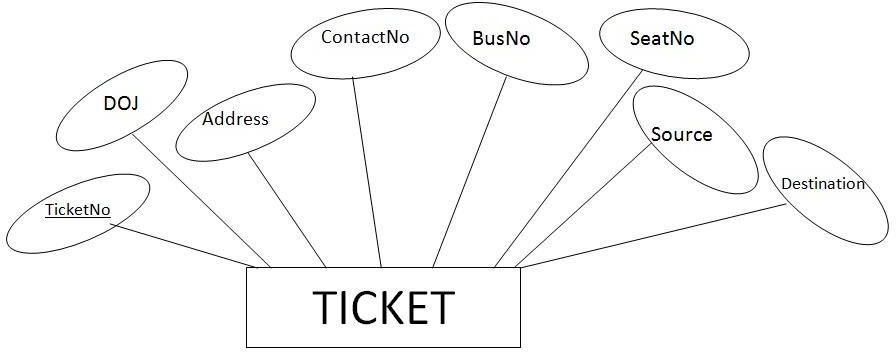


### Ticket

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

### SCHEMA

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

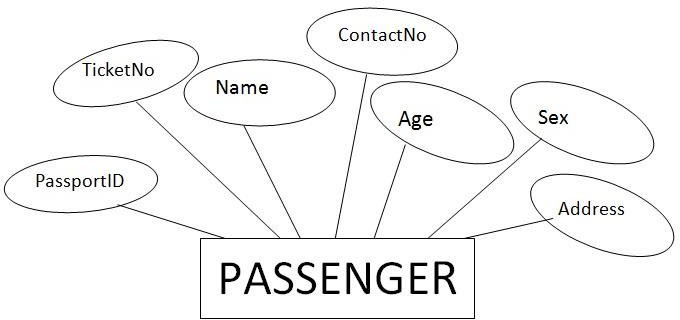


**Passenger**

* + PassportID
  + TicketNo
  + Name
  + ContactNo
  + Age
  + Sex
  + Address

## SCHEMA

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

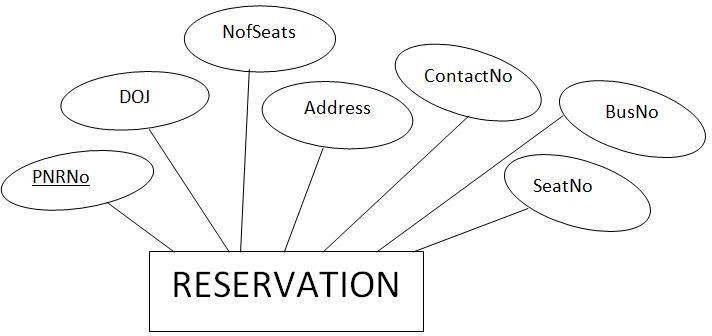


## Reservation

* + 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)

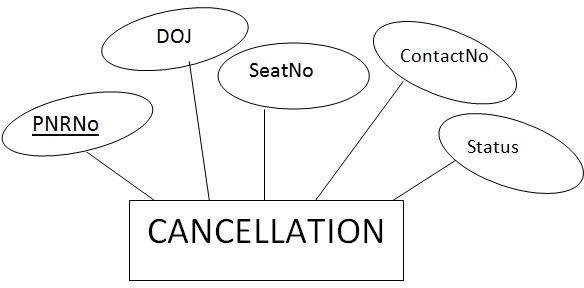


## Cancellation

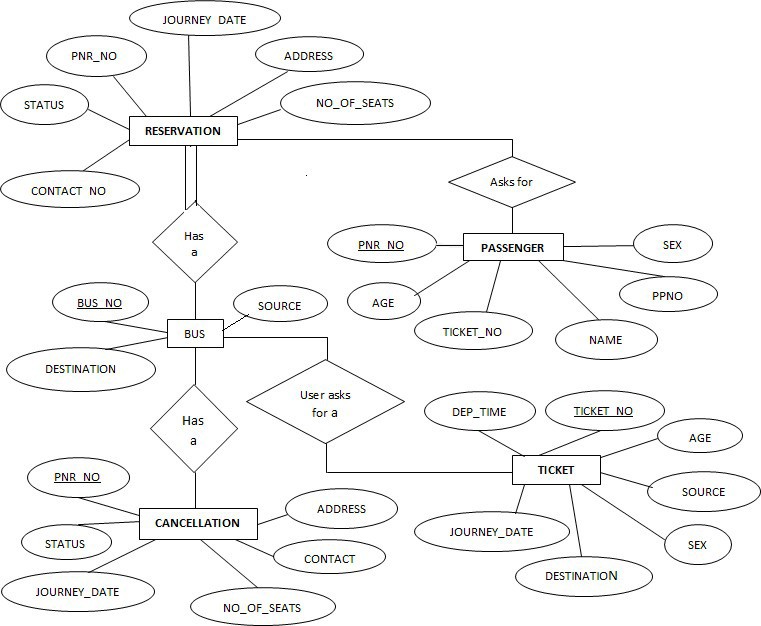
* + - PNRNo
    - DOJ
    - SeatNo
    - ContactNo
    - Status

## SCHEMA

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



# CONCEPT DESIGN WITH E-R MODEL



### EXPERIMENT – 2 RELATIONAL MODEL

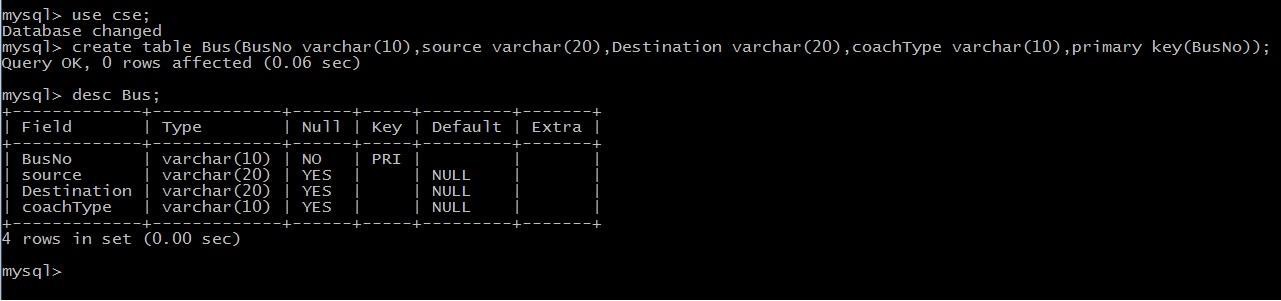
**AIM:** To Represent all the entities (Strong, Weak) in tabular fashion. Represent relationships in a tabular fashion.

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

|  |  |  |  |
| --- | --- | --- | --- |
| **ColumnName** | **Datatype** | **Constraints** | **Type of Attributes** |
| **BusNo** | **Varchar(10)** | **Primary key** | **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;



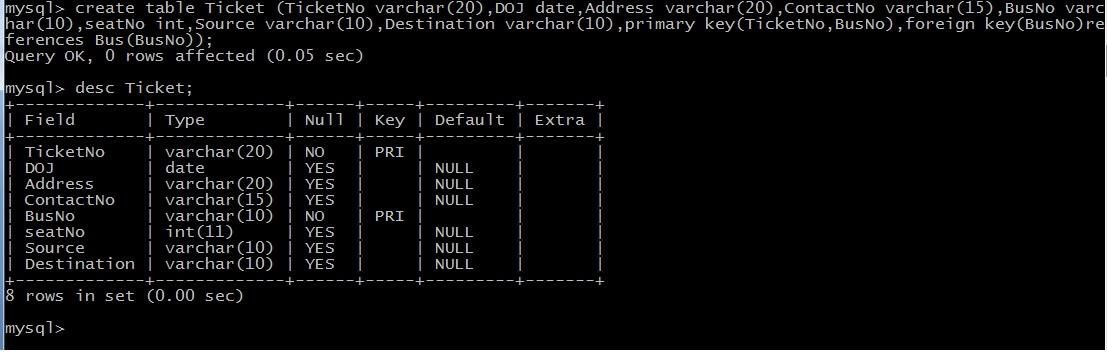
# Ticket:

**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);

**Mysq**l>desc Ticket;



# 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;



### Reservation:

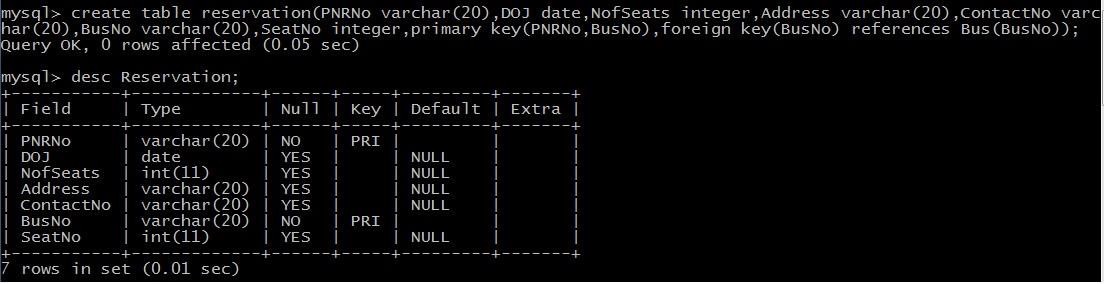
**Reservation**(PNRNo: String, DOJ: Date, NoofSeats: integer , Address: String ,ContactNo: String, , BusNo: String,SeatNo:Integer)

|  |  |  |  |
| --- | --- | --- | --- |
| **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)** | **Foreign Key** | **Single-valued** |
| **SeatNo** | **Integer** |  | **Simple** |

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

Mysql> desc reservation;



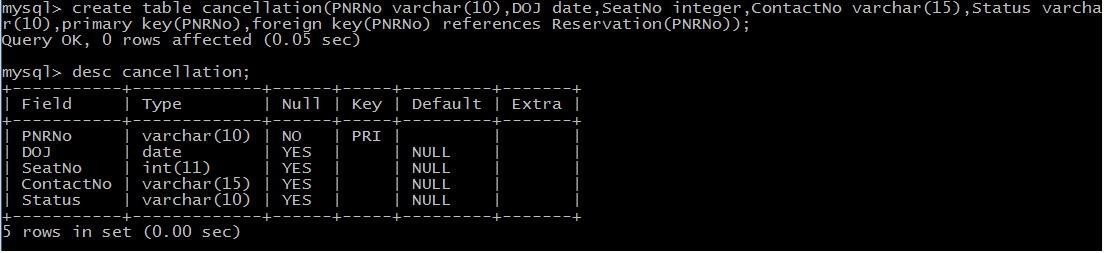
### Cancellation:

**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;



### EXPERIMENT – 3 NORMALIZATION

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

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

* 1NF: A Relation scheme is said to be in 1NF if the attribute values in the relation are atomic.i.e., Mutli –valued attributes are 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 & Q is 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,Address varchar(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));

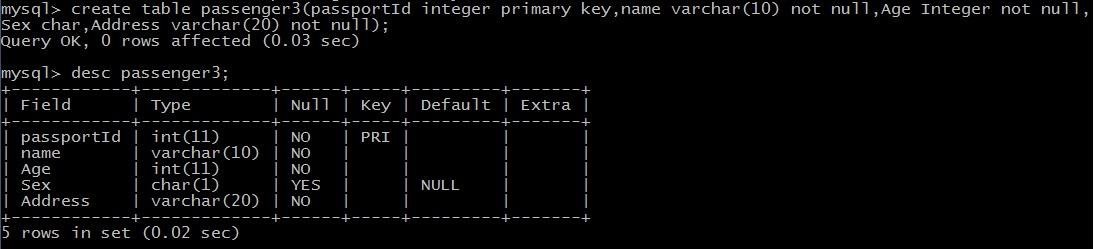
Mysql> Create table Ticket2(TicketNo Integer Primary key,JourneyDate DateTime, Age Int(4),Sex char(2),Source varchar(20),Destination varchar(20),DeptTime varchar(2));

### EXPERIMENT – 4 PRACTICING DDL COMMANDS

**AIM : Creating Tables and altering the Tables**

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

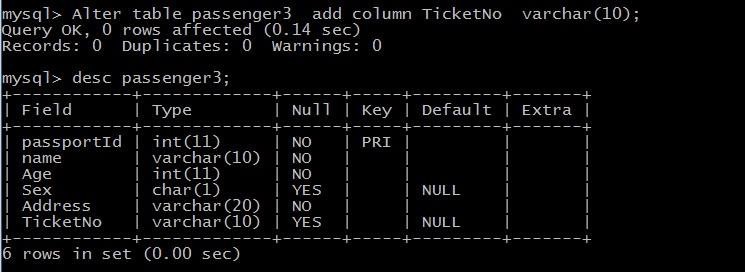
Mysql> desc passenger2;



USING ALTER COMMAND

Adding Extra column to Existing Table

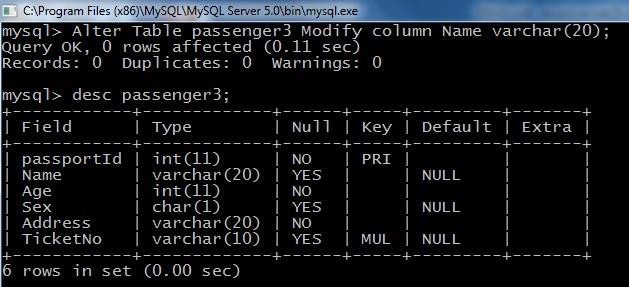
Mysql>Alter table passenger3 add column TicketNo varchar(10);



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



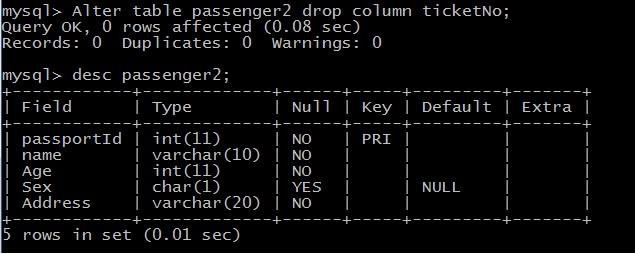
Mysql>Alter Table passenger3 Modify column Name varchar(20);



Mysql>Alter table passenger drop foreign key fk1;



Mysql> Alter table passenger2 Drop column TicketNo;



### 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)

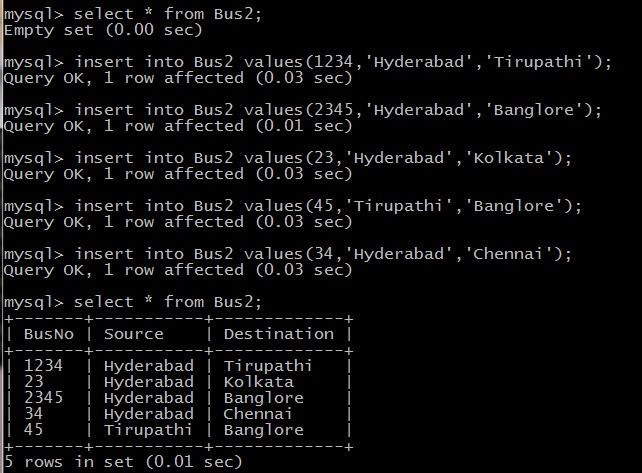
mysql> insert into Bus2 values(2345,'Hyderabad','Banglore'); Query OK, 1 row affected (0.01 sec)

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 Passenger2; Empty set (0.00 sec)

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

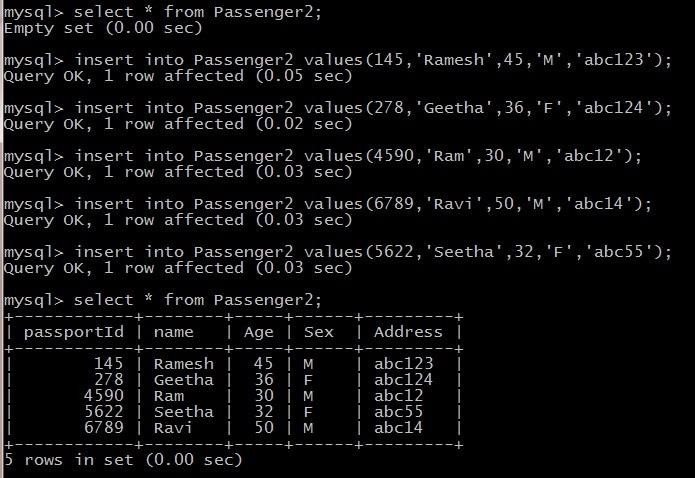
mysql> insert into Passenger2 values(278,'Geetha',36,'F','abc124'); Query OK, 1 row affected (0.02 sec)

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)

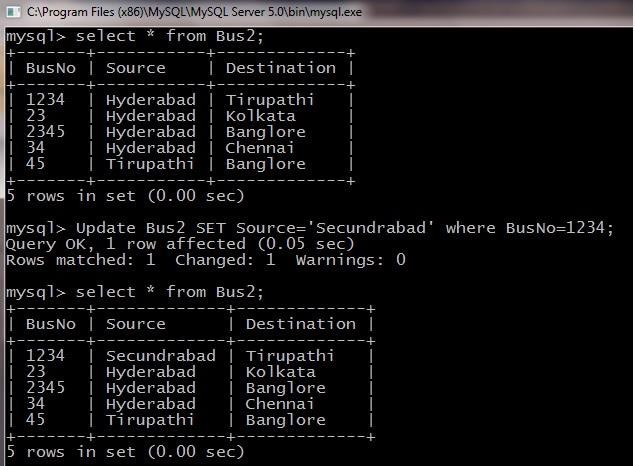
### mysql> select \* from Passenger2;



**UPDATE COMMAND ON BUS2 RELATION**

UPDATE Selected Rows & Multiple Rows

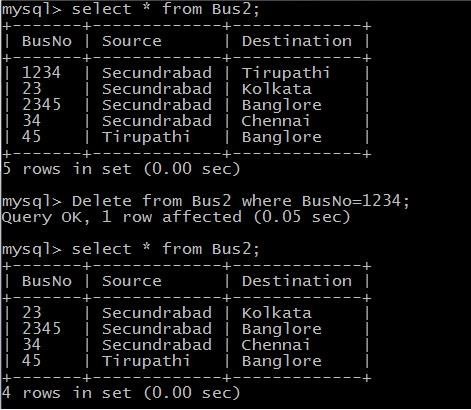
mysql> Update Bus2 SET Source='Secundrabad' where BusNo=1234; Query OK, 1 row affected (0.05 sec) Rows matched: 1 Changed: 1 Warnings: 0



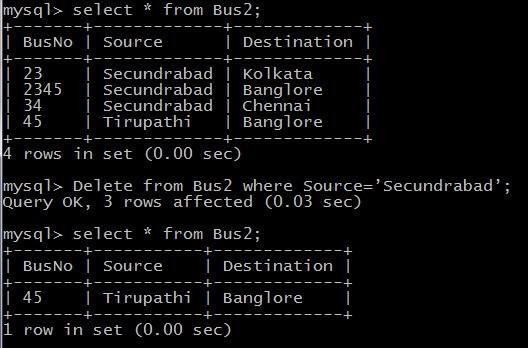
### DELETE COMMAND ON BUS2 RELATION

**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;



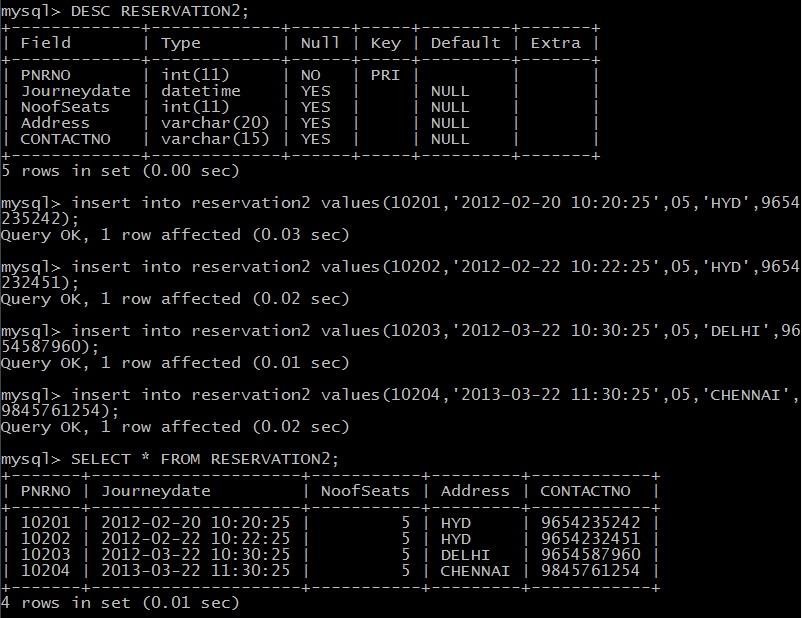
mysql> Delete from Bus2 where Source=’Secundrabad’; Query OK, 1 row affected (0.05 sec) mysql> select \* from Bus2;



### 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> 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)

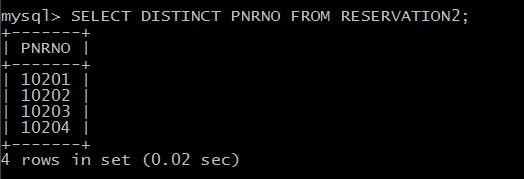
mysql> insert into reservation2 values(10202,'2012-02-22 10:22:25',05,'HYD',9654 232451); Query OK, 1 row affected (0.02 sec)

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;

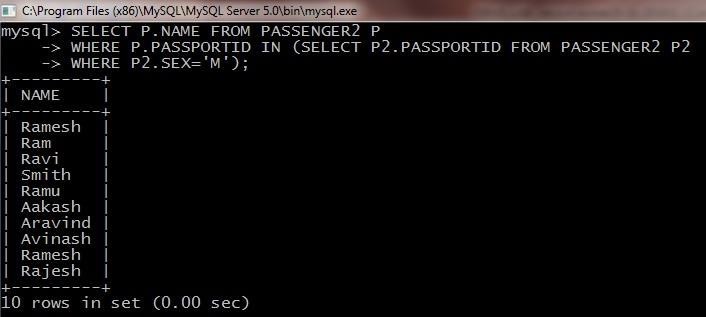
|  |
| --- |
| **PNR\_No** |
| 10201 |
| 10202 |
| 10203 |
| 10204 |

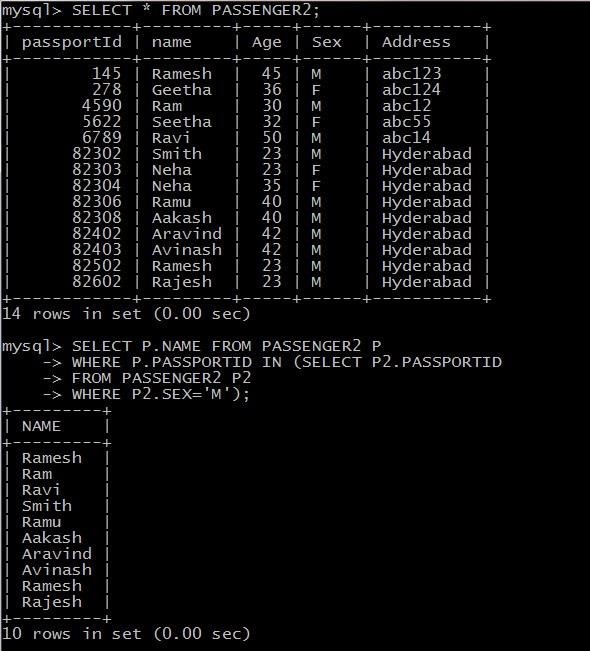


1. Display all the names of male passengers.

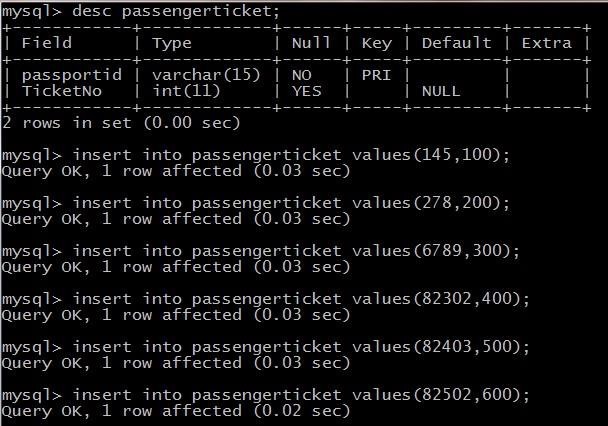
mysql> Select p.name from passenger2 p

where p.passportid IN (select p2.passportid from passenger2 p2 where p2.sex='M');

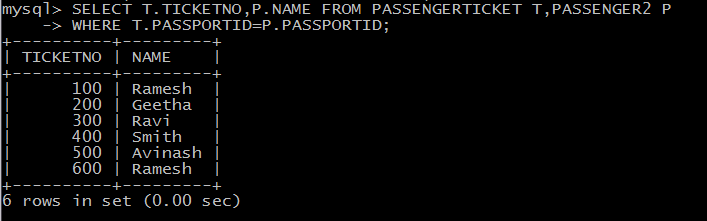




1. Display the ticket numbers and names of all the passengers.



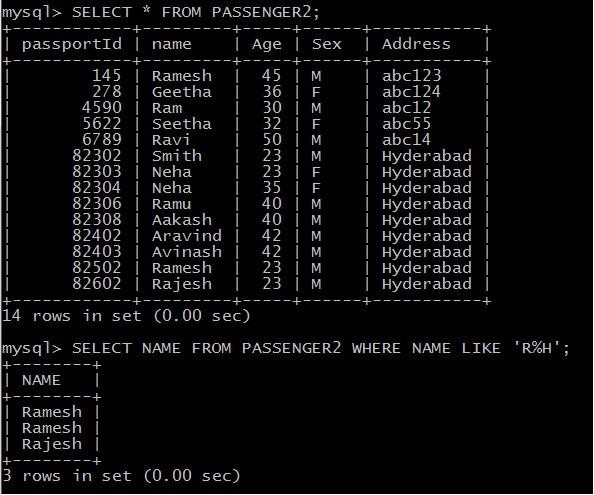
mysql> select t.ticketno,p.name from passengerticket t,passenger2 p where t.passportid = p.passportid;



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

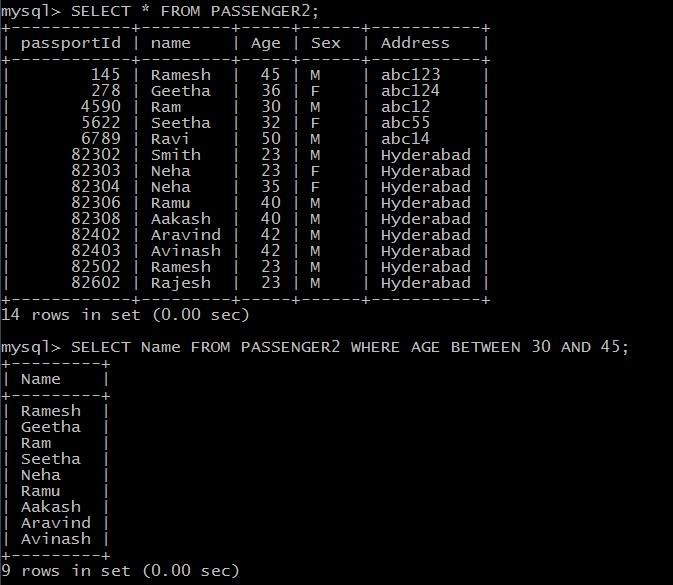
MySQL> SELECT Name FROM Passenger WHERE name LIKE ‘R%H’

|  |
| --- |
| **Name** |
| Rajesh |
| Ramesh |
| Ramesh |



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

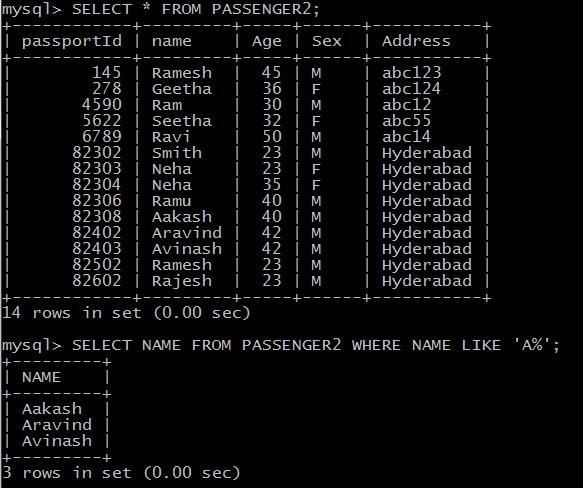
MySQL> SELECT Name FROM PASSENGER WHERE AGE BETWEEN 30 AND 45



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

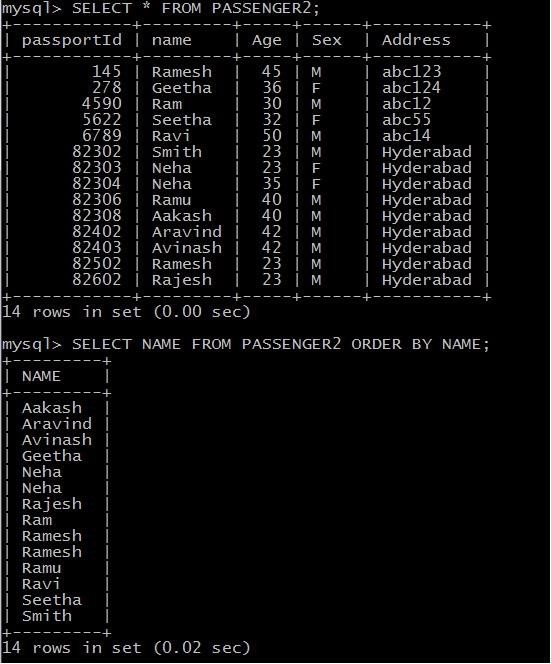
MySQL> SELECT \* FROM PASSENGER WHERE NAME LIKE ‘A%’;

|  |
| --- |
| **Name** |
| Akash |
| Arivind |
| Avinash |



1. Display the sorted list of Passengers names

MySQL> SELECT NAME FROM PASSENGER ORDER BY NAME;



### 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

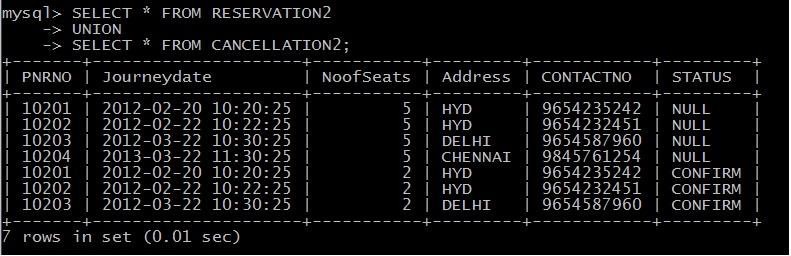
MYSQL> CREATE TABLE CANCELLATION2(PNRNO INT PRIMARY KEY,JOURNEYDATE DATETIME, NOOFSEATS INT,ADDRESS VARCHAR(20),CONTACTNO INT,STATUS VARCHAR(10),FOREIGN KEY(PNRNO) REFERENCES RESERVATION2(PNRNO));

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');

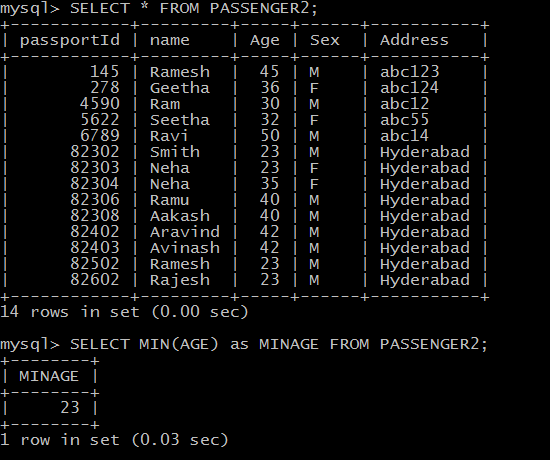
mysql> INSERT INTO CANCELLATION2 VALUES(10203,'2012-03-22 10:30:25',2,'DELHI',9654587960,'CONFIRM');

MySQL> SELECT \* FROM RESERVATION UNION SELECT \* FROM CANCELLATION;



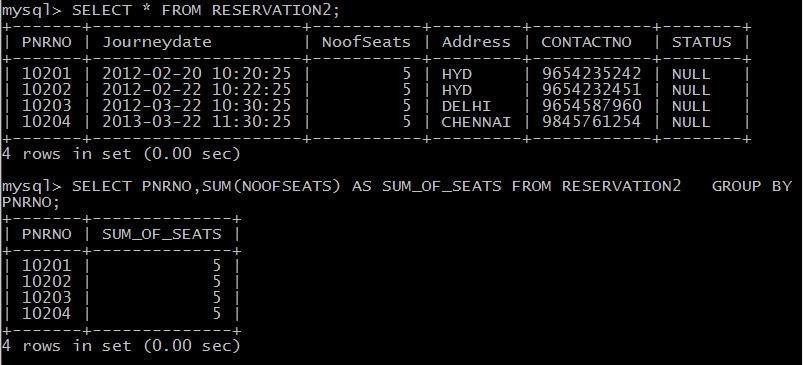
1. Display the Minimum age of the Passenger

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



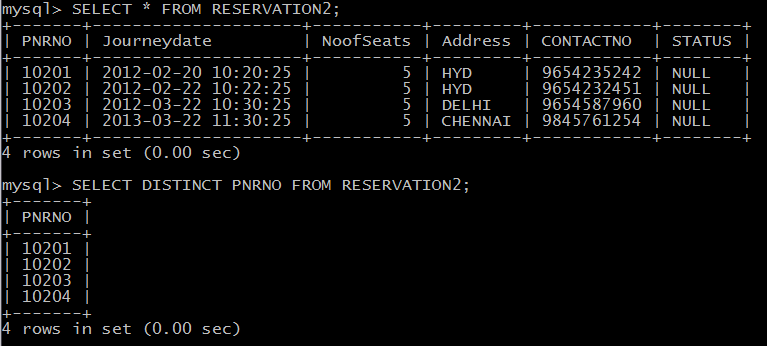
1. Find number of tickets booked for each PNR\_No using GROUP BY CLAUSE

MySQL> SELECT PNRNO,SUM(No\_of\_SEATS) AS SUM\_OF\_SEATS FROM RESERVATION2 GROUP BY PNRNO;

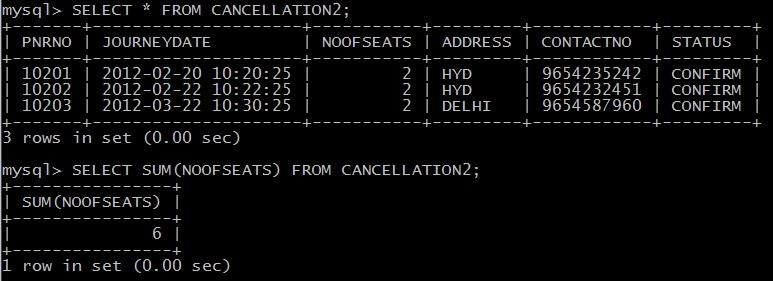


* 1. Find the distinct PNR Numbers that are present.

MySQL> SELECT DISTINCT PNR\_NO FROM RESERVATION2;

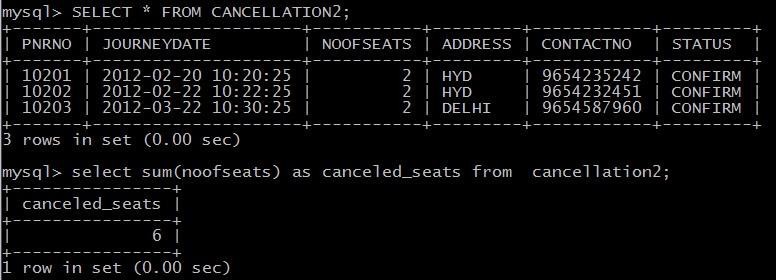


* 1. Mysql> select sum(Noofseats) from Cancellation2;



* 1. Find the total number of cancelled seats.

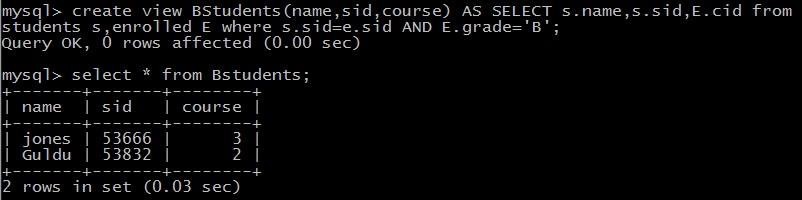
MySQL> select sum(noofseats) as canceled\_seats from cancellation2;



**Creation and Droping of Views**

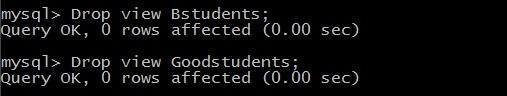
**mysql**> create table students(sid int primary key,name varchar(15),login varchar(15), age int,gpa real); mysql> create table Enrolled(sid int,cid int,grade varchar(5),primary key(sid,cid), foreign key(sid) references students(sid));

**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';



**Syntax: Drop view viewname;**

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

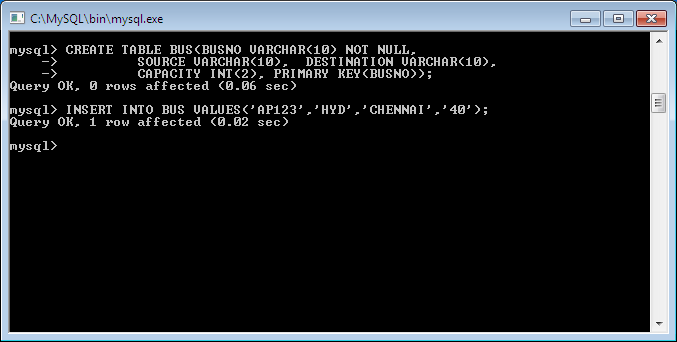


**EXPERIMENT – 8 TRIGGERS**

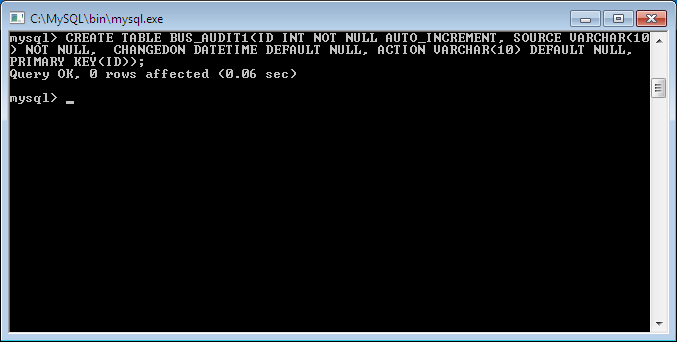
**Aim:** Creation of insert trigger, delete trigger and update trigger.

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

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



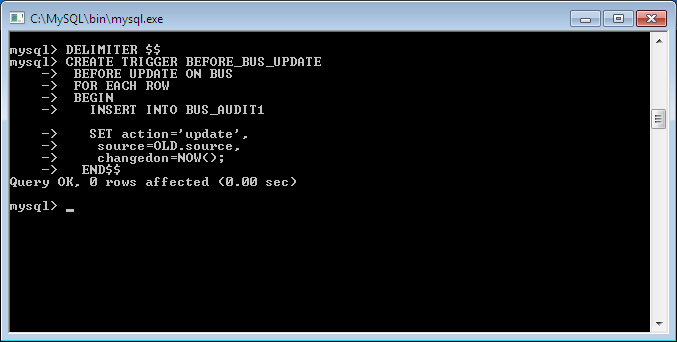
CREATE TABLE BUS\_AUDIT1(ID INT NOT NULL AUTO\_INCREMENT, SOURCE VARCHAR(10) NOT NULL, CHANGEDON DATETIME DEFAULT NULL, ACTION VARCHAR(10) DEFAULT NULL, PRIMARY KEY(ID));



CREATE TRIGGER BEFORE\_BUS\_UPDATE BEFORE UPDATE ON BUS FOR EACH ROW BEGIN

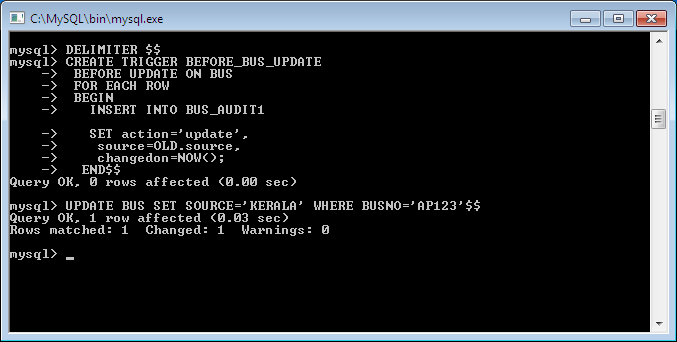
INSERT INTO BUS\_AUDIT1

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



UPDATE :

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



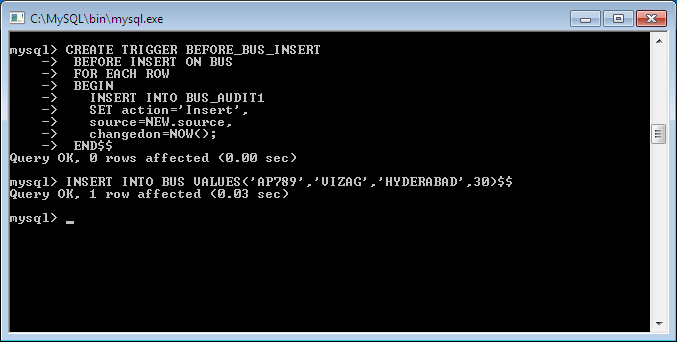
|  |  |  |  |
| --- | --- | --- | --- |
| **SNo** | **Source** | **Changedon** | **Action** |
| **1** | **Banglore** | **2014:03:23 12:51:00** | **Insert** |
| **2** | **Kerela** | **2014:03:25:12:56:00** | **Update** |
| **3** | **Mumbai** | **2014:04:26:12:59:02** | **Delete** |

INSERT:

CREATE TRIGGER BEFORE\_BUS\_INSERT BEFORE INSERT ON BUS FOR EACH ROW BEGIN

INSERT INTO BUS\_AUDIT1

SET action='Insert', source=NEW.source, changedon=NOW(); END$$ MYSQL>INSERT INTO BUS VALUES('AP789','VIZAG','HYDERABAD',30)$$



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

CREATE TRIGGER BEFORE\_BUS\_DELETE BEFORE DELETE ON BUS FOR EACH ROW BEGIN

DELETE FROM BUS\_AUDIT1

SET action='Insert', source=NEW.source, changedon=NOW(); END$$ DELETE FROM BUS WHERE SOURCE=’HYDERABAD’$$

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

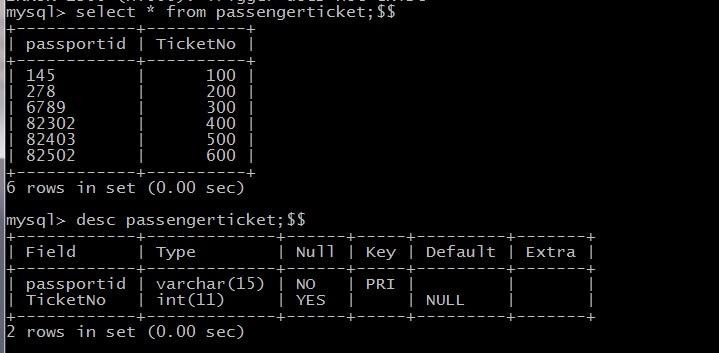
Examples

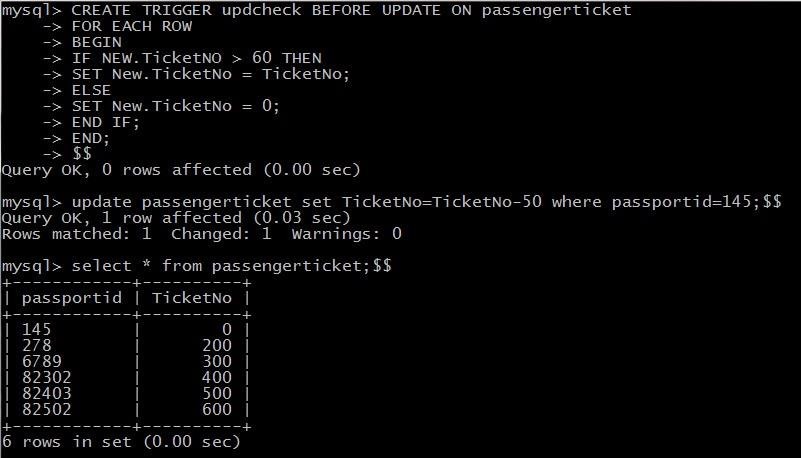
CREATE TRIGGER updcheck1 BEFORE UPDATE ON passengerticket FOR EACH ROW BEGIN

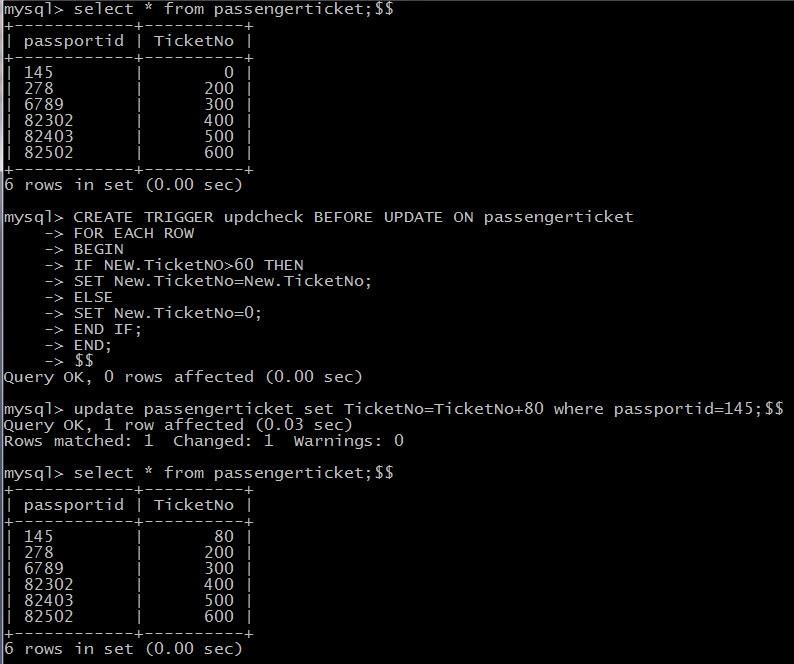
IF NEW.TicketNO > 60 THEN

SET New.TicketNo = New.TicketNo; ELSE SET New.TicketNo = 0; END IF;

END;







**EXPERIMENT – 9 PROCEDURES**

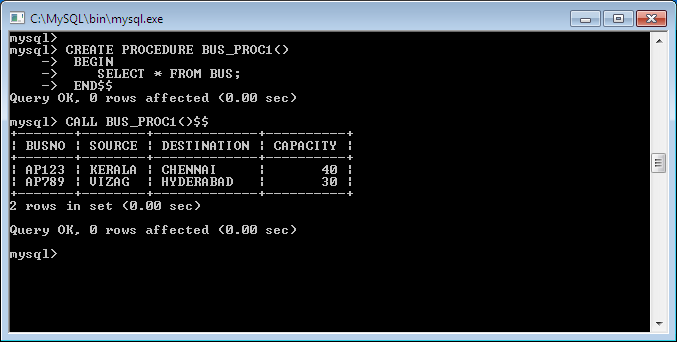
**Aim:** Creation of stored Procedures and Execution of Procedures and Modification of Procedures.

Ex1:

CREATE PROCEDURE BUS\_PROC1() BEGIN SELECT \* FROM BUS;

END$$

CALL BUS\_PROC1()$$



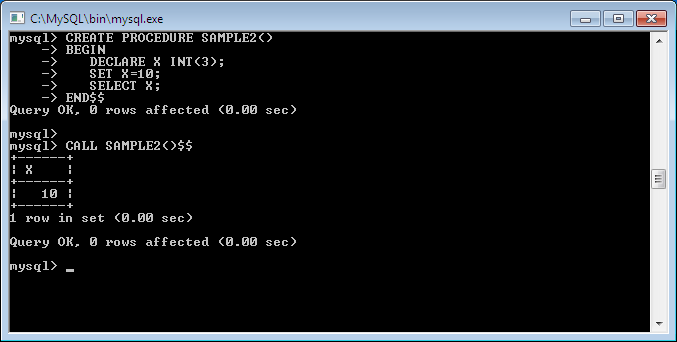
Ex2:

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

SELECT X;

END$$

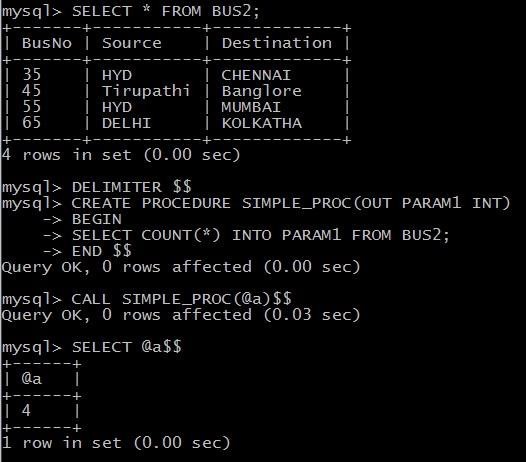
Mysql> CALL SAMPLE2()$$



Ex3: CREATE PROCEDURE SIMPLE\_PROC(OUT PARAM1 INT) BEGIN SELECT COUNT(\*) INTO PARAM1 FROM BUS;

END$$

Mysql> CALL SIMPLE\_PROC(@a)$$ Mysql> select @a;



**EXPERIMENT – 10**

### Cursors

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

### Cursors

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

* + 1. **. 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;

* + 1. **. 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.

* + 1. **. 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;**

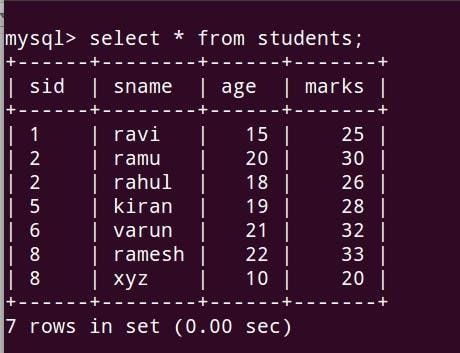
**declare v\_name varchar(20); declare v\_finished integer default 0;**

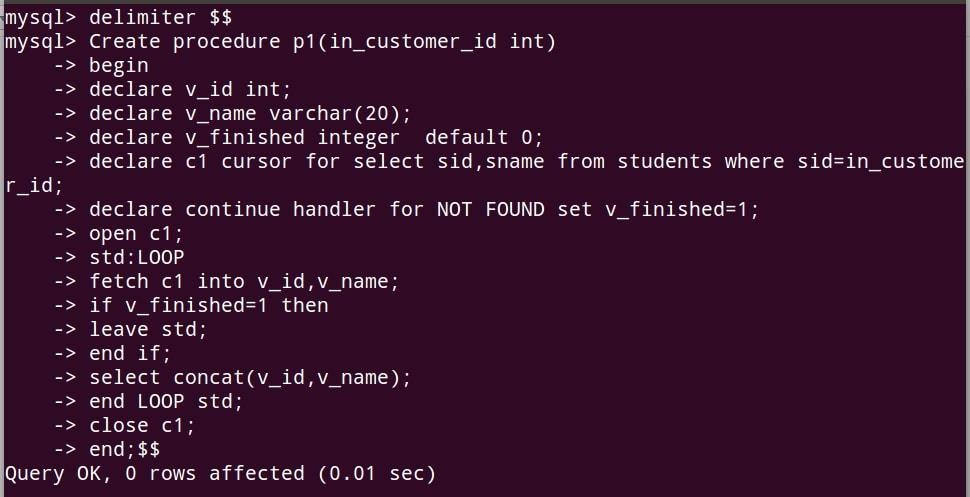
**declare c1 cursor for select sid,sname from students where sid=in\_customer\_id; declare continue handler for NOT FOUND set v\_finished=1;**

**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;**



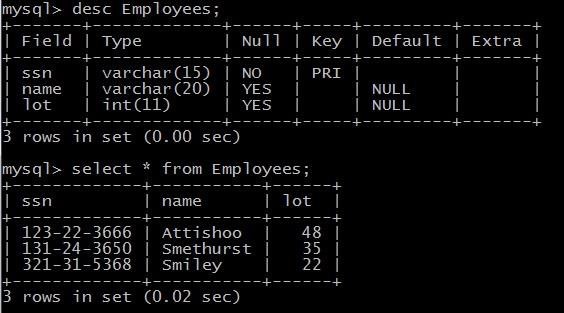


**ADDITIONAL PROGRAMMS**

**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);

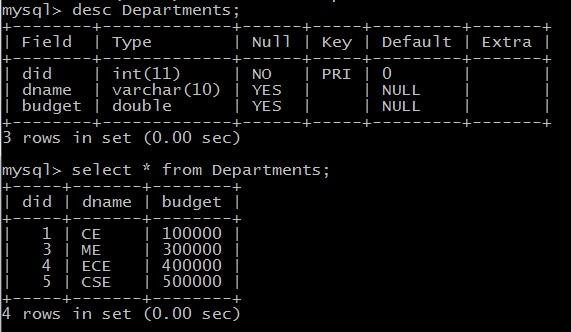
mysql> insert into Employees values('321-31-5368','Smiley',22); mysql> insert into Employees values('131-24-3650','Smethurst',35);



**DEPARTMENT TABLE**

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

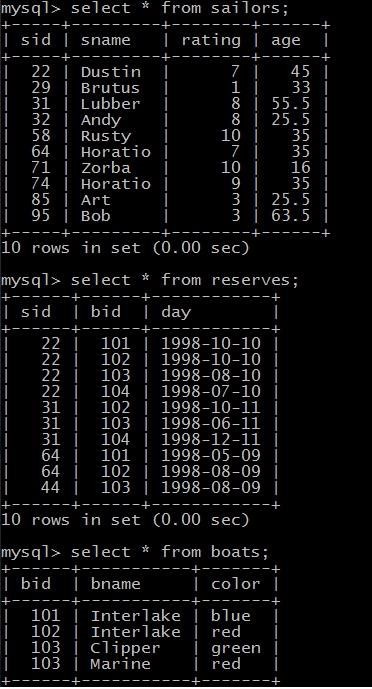
mysql> insert into Departments values(05,'CSE',500000); mysql> insert into Departments values(04,'ECE',400000); mysql> insert into Departments values(03,'ME',300000); mysql> insert into Departments values(01,'CE',100000);



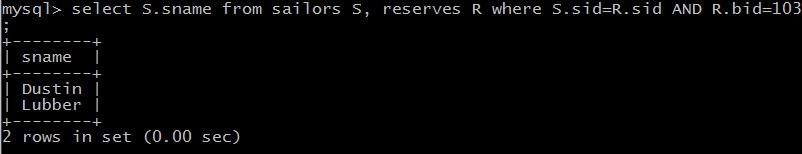
# Sailors , Reserves , Boats Tables

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);

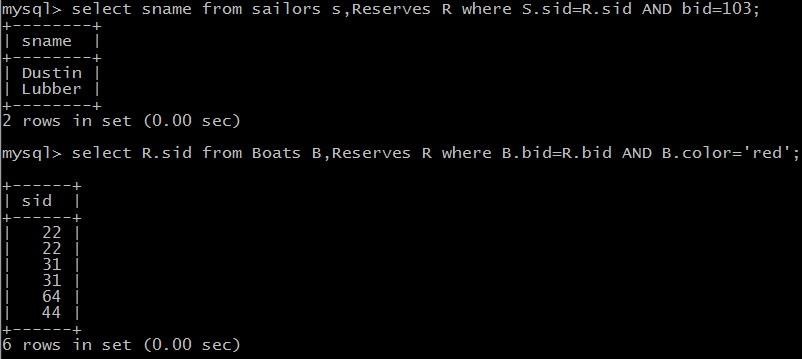
Mysql>Create table Boats(Bid int,Bname varchar(15),Color varchar(15);



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



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';



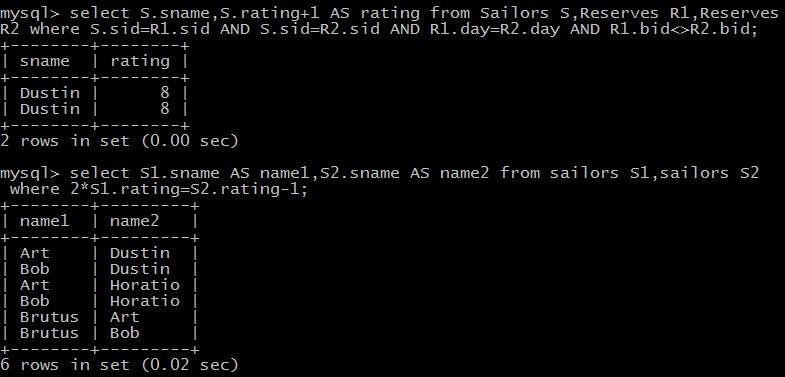
mysql> select S.sname from sailors S,reserves R,Boats B where S.sid=R.sid AND R.bid=B.bid AND B.color='red';

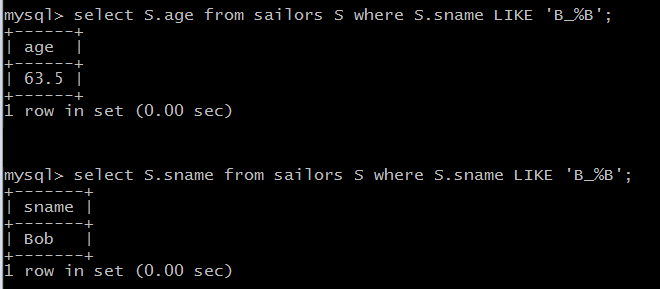
mysql> select B.color from Sailors S,Reserves R,Boats B where S.sid=R.sid AND R.bid=B.bid AND S.sname='Lubber';



mysql> select S.sname,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;

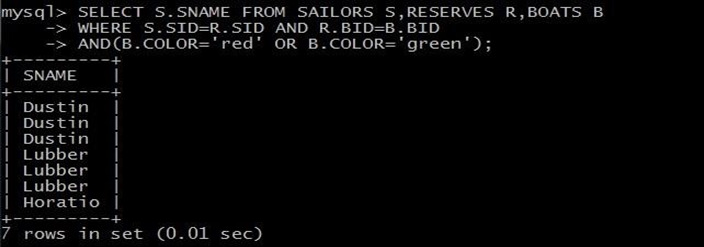
mysql> select S1.sname AS name1,S2.sname AS name2 from sailors S1,sailors S2 where 2\*S1.rating=S2.rating-1;



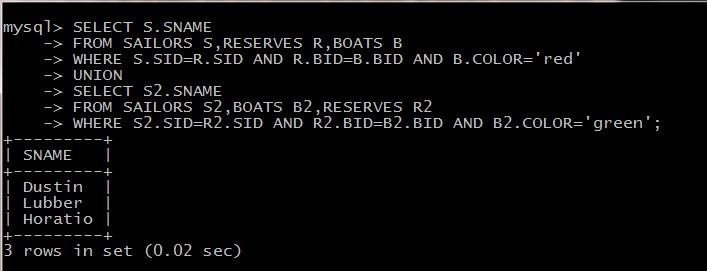


**USING UNION , INTERSECT , AND EXCEPT**

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



**OR**



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

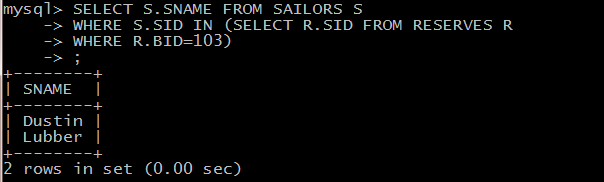
WHERE S.SID=R.SID AND R.BID=B.BID AND B.COLOR='red' INTERSECT SELECT S2.SNAME

FROM SAILORS S2,RESERVES R2,BOATS B2

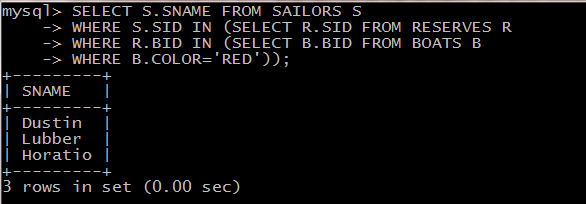
WHERE S2.SID=R2.SID AND R2.BID=B2.BID AND B2.COLOR='green';

**NESTED QUERIES**

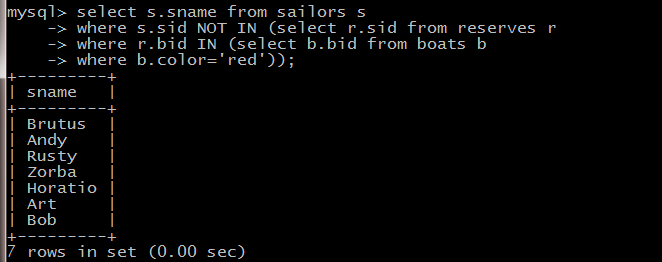
1. Find the Names of sailors who have reserved boat 103



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

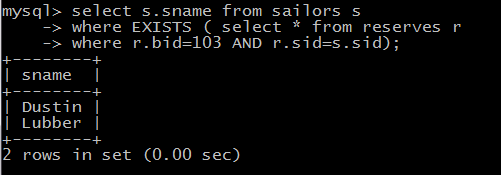


1. Find the names of Sailors who have NOT reserved a red Boat



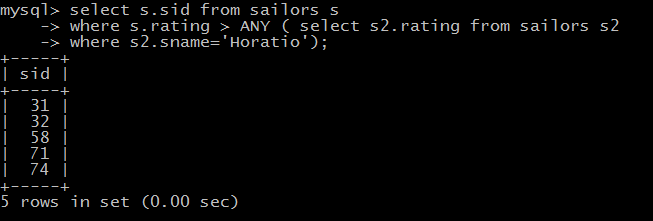
Correlated Nested Queries:

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



**Set Comparison Operators:**

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

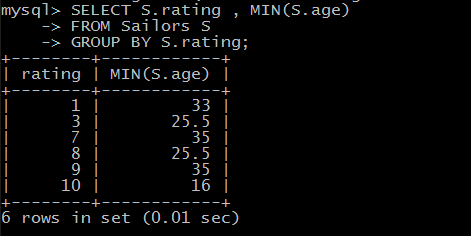


1. Find the sailors with the highest rating.

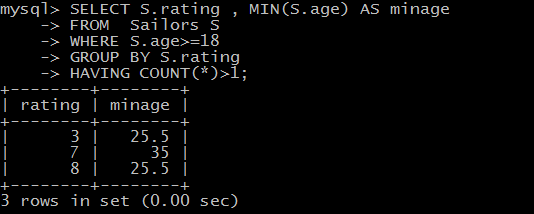
mysql> SELECT S.sid FORM Sailors WHERE S.rating>=ALL(SELECT S2.rating FROM Sailors S2);

### The GROUP BY and HAVING Clauses:

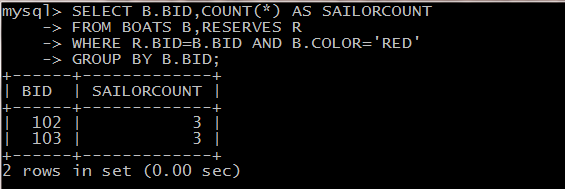
1. Find the age of the youngest sailor for each rating level.



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



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



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

