Module-5 [DBMS]

1. Create Table Name : Student and Exam

ANSWER →

```
use taskdb;

create table Student(Rollno integer primary key auto_increment,Name varchar(10),Branch varchar(30));

insert into Student (Name,Branch)values

('Jay','Computer Science'),

('Suhani','Electronic and Com '),

('Kriti','Electronic and Com ');


create table Exam(Rollno integer ,foreign key (Rollno) references Student (Rollno),S_code varchar(10),Marks integer,P_code varchar(10));

insert into Exam(Rollno,S_code,Marks,P_code)values

(1,'CS11',50,'CS'),

(1,'CS12',60,'CS'),

(2,'EC101',66,'EC'),

(2,'EC102',70,'EC'),

(3,'EC101',45,'EC'),

(3,'EC102',50,'EC');


select * from Student;

select * from exam;
```

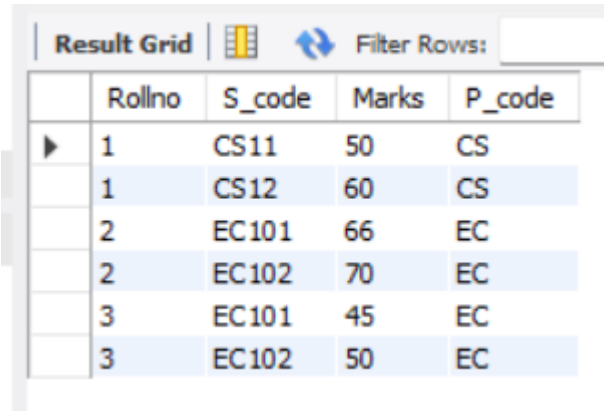
Output :

Student Table :

Result Grid | Filter Rows:

Rollno	Name	Branch
1	Jay	Computer Science
2	Suhani	Electronic and Com
3	Kriti	Electronic and Com
NULL	NULL	NULL

Exam Table :



Result Grid | Filter Rows:

Rollno	S_code	Marks	P_code
1	CS11	50	CS
1	CS12	60	CS
2	EC101	66	EC
2	EC102	70	EC
3	EC101	45	EC
3	EC102	50	EC

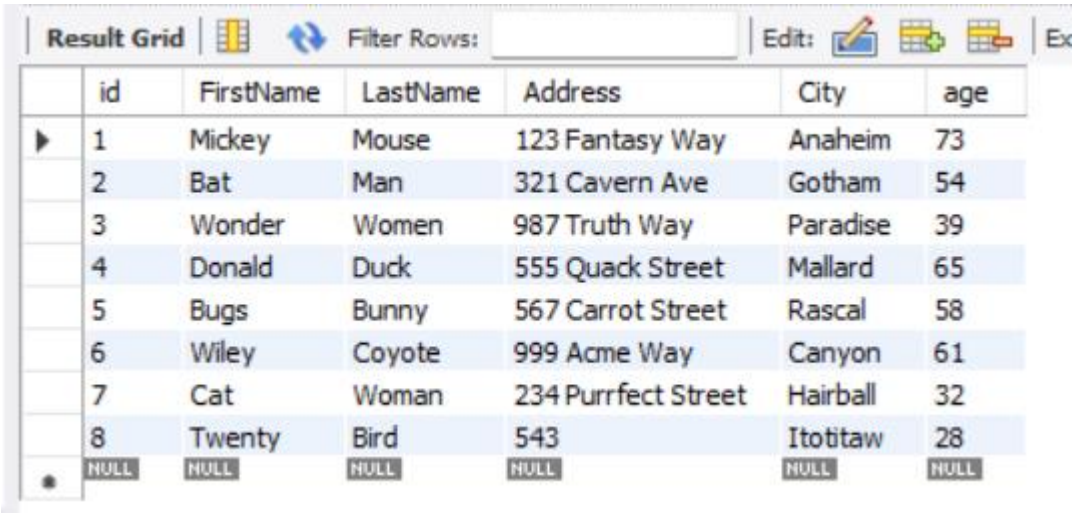
2. Create table given below

ANSWER →

```
create database data_table;

use data_table;


create table persondata(id integer primary key auto_increment, FirstName varchar(10),LastName varchar(10),Address varchar(30),City varchar(10),age integer);


insert into persondata(FirstName,LastName,Address,City,Age)values

('Mickey','Mouse','123 Fantasy Way ','Anaheim',73),

('Bat','Man','321 Cavern Ave','Gotham',54),

('Wonder','Women','987 Truth Way','Paradise',39),

('Donald','Duck','555 Quack Street','Mallard',65),

('Bugs','Bunny','567 Carrot Street','Rascal',58),

('Wiley','Coyote','999 Acme Way','Canyon',61),

('Cat','Woman','234 Purrfect Street','Hairball',32),

('Twenty','Bird','543','Itotitaw',28);


select * from persondata;
```

Output :

- Persondata Table :



Result Grid | Filter Rows: | Edit:

id	FirstName	LastName	Address	City	age
1	Mickey	Mouse	123 Fantasy Way	Anaheim	73
2	Bat	Man	321 Cavern Ave	Gotham	54
3	Wonder	Women	987 Truth Way	Paradise	39
4	Donald	Duck	555 Quack Street	Mallard	65
5	Bugs	Bunny	567 Carrot Street	Rascal	58
6	Wiley	Coyote	999 Acme Way	Canyon	61
7	Cat	Woman	234 Purrfect Street	Hairball	32
8	Twenty	Bird	543	Itotitaw	28
NULL	NULL	NULL	NULL	NULL	NULL

3. What is SQL Key Constraints? Write an Example of SQL Key Constraints?

ANSWER → SQL Key Constraints are rules applied to columns in a database table to enforce data integrity and ensure that the data adheres to specific rules. Key constraints help maintain the accuracy and consistency of the data stored in relational databases. Here are some common types of SQL key constraints:

1. Primary Key Constraint : Ensures that each row in a table has a unique, non-null identifier. A primary key constraint is applied to one or more columns to uniquely identify each record in the table.

2. Foreign Key Constraint : Ensures that the value in a column (or a set of columns) matches values in another table, thus maintaining referential integrity between tables.

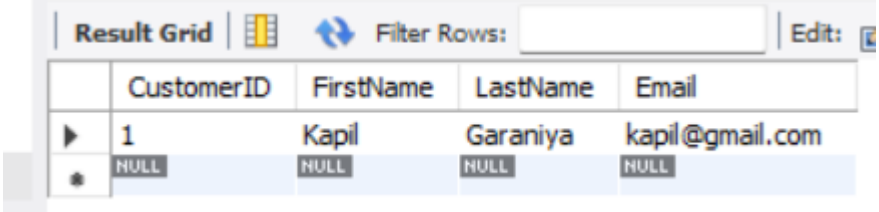
3. Unique Key Constraint : Ensures that all values in a column (or a set of columns) are unique across the table, meaning no duplicate values are allowed.

4. Not Null Constraint: Ensures that a column cannot have a NULL value, meaning every record must have a value for that column.

5. Check Constraint : Ensures that all values in a column satisfy a specific condition.

Examples :

```
use data_table;


create table info(CustomerID int primary key auto_increment, FirstName varchar(20), LastName varchar(20),
Email varchar(30) unique not null);


insert into info(FirstName, LastName, Email) values

("Kapil","Garaniya","kapil@gmail.com");


select * from info;
```



Result Grid | Filter Rows: | Edit:

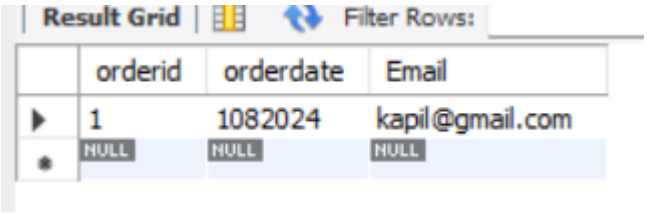
CustomerID	FirstName	LastName	Email
1	Kapil	Garaniya	kapil@gmail.com
NULL	NULL	NULL	NULL

create table product(orderid int primary key auto_increment, orderdate varchar(20) not null, Email varchar(30) unique not null, foreign key product(Email) references info(Email));

insert into product(orderdate, Email) values

(01082024,"kapil@gmail.com");

select * from product;



4. What is SQL View Create a View of Student Table?

ANSWER → An SQL view is a virtual table that provides a way to represent data from one or more tables in a simplified or customized manner. Views do not store data themselves but rather display data stored in other tables based on a query. They can be used to simplify complex queries, enforce data security by restricting access to specific columns or rows, and present data in a more readable format.

- ⇨ **Creating a View :**

use data_table;

create table student (rollno int primary key auto_increment , firstname varchar(20), lastname varchar(20), age int);

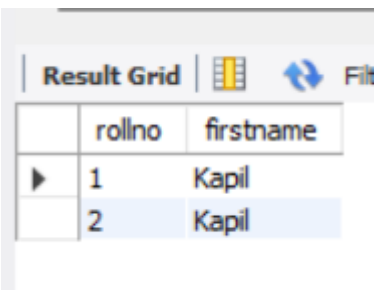
insert into student(firstname, lastname, age) values

("Kapil","Garaniya",20);

create VIEW studentoverview as select rollno, firstname FROM student;

select * from studentoverview;

Output :



5. How to Create a Table user write a SQL query?

ANSWER →

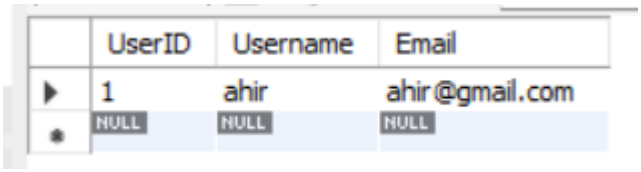
use data_table;

CREATE TABLE Users(UserID INTEGER PRIMARY KEY AUTO_INCREMENT, Username varchar(20) UNIQUE NOT NULL, Email varchar(20) UNIQUE NOT NULL);

insert into Users(Username, Email) values("ahir","ahir@gmail.com");

select * from Users;

Output :



	UserID	Username	Email
▶	1	ahir	ahir@gmail.com
*	NULL	NULL	NULL

6. What is SQL and How to Create a table with Foreign Key?

ANSWER → SQL (Structured Query Language) is a standard programming language used to manage and manipulate relational databases. It provides commands for querying, inserting, updating, and deleting data, as well as for creating and modifying database structures.

- ⇨ **table with Foreign Key :**

create database stdata;

use stdata;

create table Student(Rollno integer primary key auto_increment,Name varchar(10),Branch varchar(30));

insert into Student (Name,Branch)values

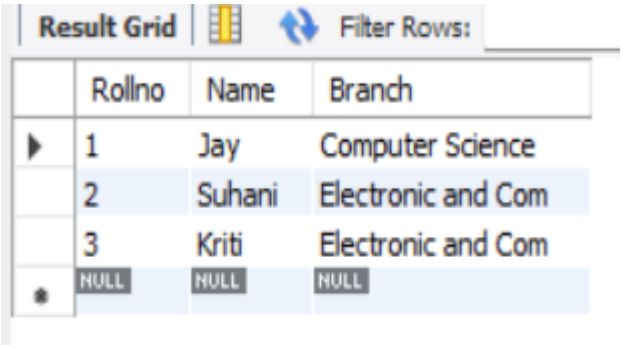
('Jay','Computer Science'),

('Suhani','Electronic and Com '),

('Kriti','Electronic and Com ');

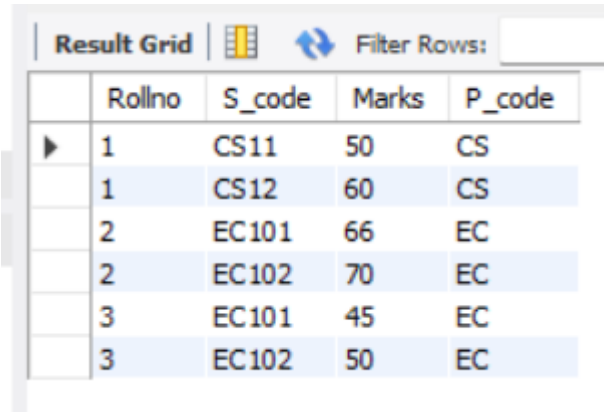
```
create table Exam(Rollno integer ,foreign key (Rollno) references Student (Rollno),S_code varchar(10),Marks integer,P_code varchar(10));


insert into Exam(Rollno,S_code,Marks,P_code)values

(1,'CS11',50,'CS'),

(1,'CS12',60,'CS'),

(2,'EC101',66,'EC'),

(2,'EC102',70,'EC'),

(3,'EC101',45,'EC'),

(3,'EC102',50,'EC');


select * from Student;

select * from exam;
```

Output :



Result Grid | Filter Rows:

Rollno	Name	Branch
1	Jay	Computer Science
2	Suhani	Electronic and Com
3	Kriti	Electronic and Com
NULL	NULL	NULL



Result Grid | Filter Rows:

Rollno	S_code	Marks	P_code
1	CS11	50	CS
1	CS12	60	CS
2	EC101	66	EC
2	EC102	70	EC
3	EC101	45	EC
3	EC102	50	EC

7. What is trigger and how to Create a Trigger in SQL?

ANSWER →

A trigger is a stored procedure in a database that automatically invokes whenever a special event in the database occurs. For example, a trigger can be invoked when a row is inserted into a specified table or when specific table columns are updated. In simple words, a trigger is a collection of SQL statements with particular names that are

stored in system memory. It belongs to a specific class of stored procedures that are automatically invoked in response to database server events. Every trigger has a table attached to it.

Because a trigger cannot be called directly, unlike a stored procedure, it is referred to as a special procedure. A trigger is automatically called whenever a data modification event against a table takes place, which is the main distinction between a trigger and a procedure. On the other hand, a stored procedure must be called directly.

use data_table;

Trigger>> delimiter //

create procedure getall()

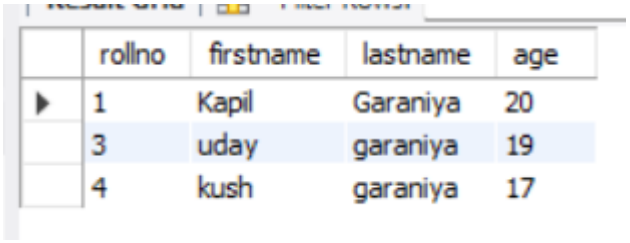
begin

select * from stud;

end

// delimiter;

call getall();



rollno	firstname	lastname	age
1	Kapil	Garaniya	20
3	uday	garaniya	19
4	kush	garaniya	17

8. What is Difference Between DBMS and RDBMS?

ANSWER →

DBMS is a system for managing databases, storing data in files without enforcing relationships between data entities.

RDBMS is a type of DBMS that stores data in tables, enforces relationships through foreign keys, and uses SQL for querying, making it suitable for handling large and complex databases.

9. What is Normalization?

ANSWER → Normalization is the process of organizing data in a database to reduce redundancy and improve data integrity. It involves dividing a database into tables and defining relationships between them according to specific rules, known as normal forms. The goal is to ensure that each piece of data is stored only once, which minimizes duplication and helps maintain consistency across the database.

Table Name: Employee

Employee_id	First_name	Last_name	Salary	Joining_date	Department
1	John	Abraham	1000000	01-JAN-13 12.00.00 AM	Banking
2	Michael	Clarke	800000	01-JAN-13 12.00.00 AM	Insurance
3	Roy	Thomas	700000	01-FEB-13 12.00.00 AM	Banking
4	Tom	Jose	600000	01-FEB-13 12.00.00 AM	Insurance
5	Jerry	Pinto	650000	01-FEB-13 12.00.00 AM	Insurance
6	Philip	Mathew	750000	01-JAN-13 12.00.00 AM	Services
7	TestName1	123	650000	01-JAN-13 12.00.00 AM	Services
8	TestName2	Lname%	600000	01-FEB-13 12.00.00 AM	Insurance

Table Name: Incentive

Employee_ref_id	Incentive_date	Incentive_amount
1	01-FEB-13	5000
2	01-FEB-13	3000
3	01-FEB-13	4000
1	01-JAN-13	4500
2	01-JAN-13	3500

create database job;

use job;

CREATE TABLE Employee (Employee_id INT PRIMARY KEY AUTO_INCREMENT, First_name VARCHAR(50), Last_name VARCHAR(50), Salary INT, Joining_date DATETIME,Department VARCHAR(50));

INSERT INTO Employee (First_name, Last_name, Salary, Joining_date, Department) VALUES

('John', 'Abraham', 1000000, '2013-01-01 00:00:00', 'Banking'),

('Michael', 'Clarke', 800000, '2013-01-01 00:00:00', 'Insurance'),

('Roy', 'Thomas', 700000, '2013-02-01 00:00:00', 'Banking'),

('Tom', 'Jose', 600000, '2013-02-01 00:00:00', 'Insurance'),

('Jerry', 'Pinto', 650000, '2013-02-01 00:00:00', 'Insurance'),

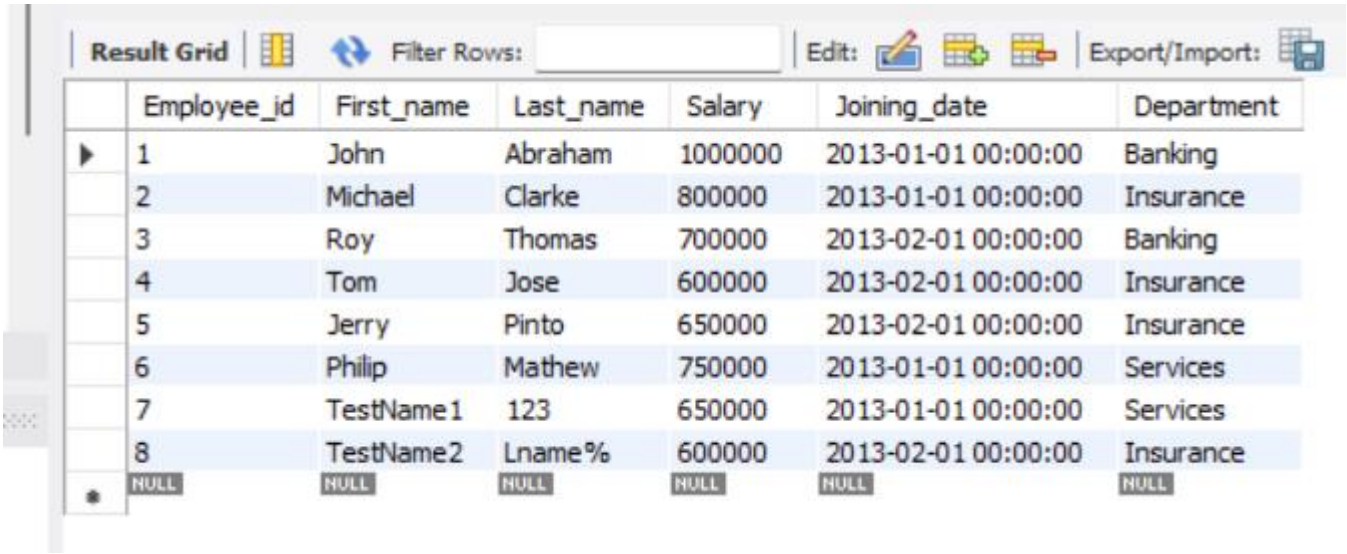
('Philip', 'Mathew', 750000, '2013-01-01 00:00:00', 'Services'),

('TestName1', '123', 650000, '2013-01-01 00:00:00', 'Services'),

('TestName2', 'Lname%', 600000, '2013-02-01 00:00:00', 'Insurance');

SELECT * FROM employee;

Output :



Result Grid | Filter Rows: | Edit: | Export/Import:

Employee_id	First_name	Last_name	Salary	Joining_date	Department
1	John	Abraham	1000000	2013-01-01 00:00:00	Banking
2	Michael	Clarke	800000	2013-01-01 00:00:00	Insurance
3	Roy	Thomas	700000	2013-02-01 00:00:00	Banking
4	Tom	Jose	600000	2013-02-01 00:00:00	Insurance
5	Jerry	Pinto	650000	2013-02-01 00:00:00	Insurance
6	Philip	Mathew	750000	2013-01-01 00:00:00	Services
7	TestName1	123	650000	2013-01-01 00:00:00	Services
8	TestName2	Lname%	600000	2013-02-01 00:00:00	Insurance
NULL	NULL	NULL	NULL	NULL	NULL

CREATE TABLE Incentive (Employee_ref_id INT, Incentive_date DATE, Incentive_amount INT, FOREIGN KEY (Employee_ref_id) REFERENCES Employee(Employee_id));

INSERT INTO Incentive (Employee_ref_id, Incentive_date, Incentive_amount) VALUES

(1, '2013-02-01', 5000),

(2, '2013-02-01', 3000),

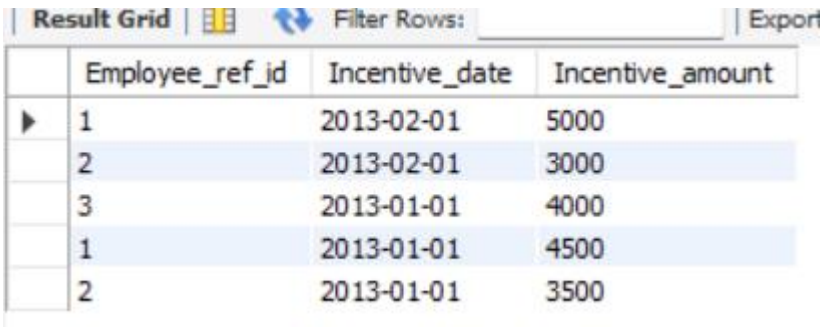
(3, '2013-01-01', 4000),

(1, '2013-01-01', 4500),

(2, '2013-01-01', 3500);

SELECT * FROM Incentive;

Output :



Result Grid | Filter Rows: | Export

Employee_ref_id	Incentive_date	Incentive_amount
1	2013-02-01	5000
2	2013-02-01	3000
3	2013-01-01	4000
1	2013-01-01	4500
2	2013-01-01	3500

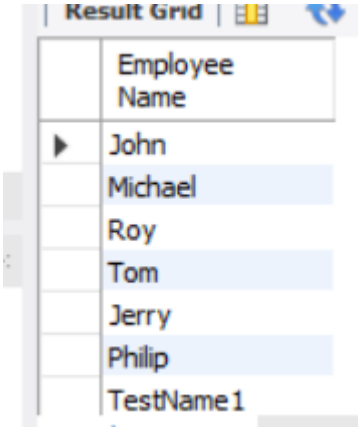
1. Get First_Name from employee table using alias name “Employee Name”.

ANSWER →

use job;

select First_name from Employee;

Output :



Result Grid

Employee Name
John
Michael
Roy
Tom
Jerry
Philip
TestName1

2. Get FIRST_NAME, Joining year, Joining Month and Joining Date from employee table.

ANSWER →

use job;

select First_name, year(Joining_date)Joining_Year, monthname(Joining_date)Joining_Month, date(Joining_date) Joining_Date from Employee;

Output :

First_name	Joining_Year	Joining_Month	Joining_Day
John	2013	January	1
Michael	2013	January	1
Roy	2013	February	1
Tom	2013	February	1
Jerry	2013	February	1
Philip	2013	January	1
TestName1	2013	January	1

3. Get all employee details from the employee table order by First_Name Ascending and Salary descending.

ANSWER →

use job;

select * from Employee order by First_name , Salary desc;

Output :

Employee_id	First_name	Last_name	Salary	Joining_date	Department
5	Jerry	Pinto	650000	2013-02-01 00:00:00	Insurance
1	John	Abraham	1000000	2013-01-01 00:00:00	Banking
2	Michael	Clarke	800000	2013-01-01 00:00:00	Insurance
6	Philip	Mathew	750000	2013-01-01 00:00:00	Services
3	Roy	Thomas	700000	2013-02-01 00:00:00	Banking
7	TestName1	123	650000	2013-01-01 00:00:00	Services
8	TestName2	Lname%	600000	2013-02-01 00:00:00	Insurance
4	Tom	Jose	600000	2013-02-01 00:00:00	Insurance
NULL	NULL	NULL	NULL	NULL	NULL

4. Get employee details from employee table whose first name contains 'o'. Get employee details from employee table whose joining month is "january".

ANSWER →

use job;

select * from Employee where First_name like '%o%';

Output :

Employee_id	First_name	Last_name	Salary	Joining_date	Department
1	John	Abraham	1000000	2013-01-01 00:00:00	Banking
3	Roy	Thomas	700000	2013-02-01 00:00:00	Banking
4	Tom	Jose	600000	2013-02-01 00:00:00	Insurance
NULL	NULL	NULL	NULL	NULL	NULL

select * from Employee where MONTHNAME(Joining_date) = 'January';

Output :

Employee_id	First_name	Last_name	Salary	Joining_date	Department
1	John	Abraham	1000000	2013-01-01 00:00:00	Banking
2	Michael	Clarke	800000	2013-01-01 00:00:00	Insurance
6	Philip	Mathew	750000	2013-01-01 00:00:00	Services
7	TestName1	123	650000	2013-01-01 00:00:00	Services
NULL	NULL	NULL	NULL	NULL	NULL

5. Get department, total salary with respect to a department from employee table order by total salary descending.

ANSWER →

use job;

select Department, sum(Salary) Total_Salary FROM Employee group by Department order by Total_Salary desc;

Output :

	Department	Total_Salary
▶	Insurance	2650000
	Banking	1700000
	Services	1400000

6. Get department wise maximum salary from employee table order by salary ascending.

ANSWER →

use job;

select Department, max(Salary) Max_Salary from Employee group by Department order by Max_Salary;

Output :

	Department	Max_Salary
▶	Services	750000
	Insurance	800000
	Banking	1000000

7. Select first_name, incentive amount from employee and incentives table for those employees who have incentives and incentive amount greater than 3000.

ANSWER →

use job;

select Employee.First_name ,Incentive.Incentive_amount from Employee join Incentive

on Employee.Employee_id = Incentive.Employee_ref_id where Incentive_amount>3000;

Output :

	First_name	Incentive_amount
▶	John	5000
	Roy	4000
	John	4500
	Michael	3500

8. Select 2nd Highest salary from employee table.

ANSWER →

use job;

select first_name, Salary from Employee where Salary = (select MAX(Salary) from Employee where Salary < (select MAX(Salary) from Employee));

Output :

	first_name	Salary
▶	John	1000000

9. Select first_name, incentive amount from employee and incentives table for all employees who got incentives using left join.

ANSWER →

use job;

select employee.employee_id, employee.First_name, incentive.Incentive_amount

from Employee LEFT JOIN incentive ON Employee.employee_id=incentive.employee_ref_id;

Output :

	employee_id	First_name	Incentive_amount
▶	1	John	4500
	1	John	5000
	2	Michael	3500
	2	Michael	3000
	3	Roy	4000
	4	Tom	NULL
	5	Jerry	NULL
	6	Philip	NULL
	7	TestName1	NULL
	8	TestName2	NULL
	9	Kapil	NULL

10. Create View OF Employee table in which store first name ,last name and salary only.

ANSWER →

use job;

create view Employee_View as select First_name, Last_name, Salary from Employee;

select * from Employee_View;

Output :

	First_name	Last_name	Salary
▶	John	Abraham	1000000
	Michael	Clarke	800000
	Roy	Thomas	700000
	Tom	Jose	600000
	Jerry	Pinto	650000
	Philip	Mathew	750000
	TestName1	123	650000
	TestName2	Lname%	600000

11. Create Procedure to find out department wise highest salary.

ANSWER →

```
use job;

DELIMITER //
create PROCEDURE HSalary()
BEGIN
   select Department, max(Salary) Highest_Salary from Employee group by Department;
END //
DELIMITER ;
call HSalary();
```

Output :

	Department	Highest_Salary
▶	Banking	1000000
	Insurance	800000
	Services	750000

12. Create After Insert trigger on Employee table which insert records in viewtable.

ANSWER →

```
use job;

create table Employee_backup(id int primary key auto_increment,
First_name varchar(10),
Last_name varchar(10),
```

```
Salary int,
joining_date datetime,
Department varchar(20));
delimiter //
create trigger Emplog
after insert
on Employee
for each row
begin
insert into Employee_backup(First_name,Last_name,Salary,joining_date,Department)values
(NEW.First_name,NEW.Last_name,NEW.Salary,NEW.joining_date,NEW.Department);
end //
delimiter ;
```

```
insert into Employee(First_name,Last_name,Salary,joining_date,Department)values
('Kapil','Garaniya',1000999,'2012-01-01 01:00:00','Banking');
```

```
select * from Employee;
```

Output :

Employee_id	First_name	Last_name	Salary	Joining_date	Department
1	John	Abraham	1000000	2013-01-01 00:00:00	Banking
2	Michael	Clarke	800000	2013-01-01 00:00:00	Insurance
3	Roy	Thomas	700000	2013-02-01 00:00:00	Banking
4	Tom	Jose	600000	2013-02-01 00:00:00	Insurance
5	Jerry	Pinto	650000	2013-02-01 00:00:00	Insurance
6	Philip	Mathew	750000	2013-01-01 00:00:00	Services
7	TestName1	123	650000	2013-01-01 00:00:00	Services
8	TestName2	Lname%	600000	2013-02-01 00:00:00	Insurance
9	Kapil	Garaniya	1000999	2012-01-01 01:00:00	Banking
NULL	NULL	NULL	NULL	NULL	NULL

```
select * from Employee_backup;
```

Output :

id	First_name	Last_name	Salary	joining_date	Department
1	Kapil	Garaniya	1000999	2012-01-01 01:00:00	Banking
NULL	NULL	NULL	NULL	NULL	NULL

TABLE-1

TABLE NAME- SALSEPERSON

(PK)SNo	SNAME	CITY	COMM
1001	Peel	London	.12
1002	Serres	San Jose	.13
1004	Motika	London	.11
1007	Rafkin	Barcelona	.15
1003	Axelrod	New York	.1

TABLE-2

TABLE NAME- CUSTOMER

(PK)CNM.	CNAME	CITY	RATING	(FK)SNo
201	Hoffman	London	100	1001
202	Giovanne	Roe	200	1003
203	Liu	San Jose	300	1002
204	Grass	Barcelona	100	1002
206	Clemens	London	300	1007
207	Pereira	Roe	100	1004

1. Create Table Name : salesperson and customer

ANSWER →

create database shop;

use shop;

create table salesperson (sno int primary key, sname varchar(100), city varchar(100), comm decimal(3, 2));

insert into salesperson (sno, sname, city, comm) values

(1001, 'peel', 'london', 0.12),

(1002, 'serres', 'san jose', 0.13),

(1004, 'motika', 'london', 0.11),

(1007, 'rafkin', 'barcelona', 0.15),

(1003, 'axelrod', 'new york', 0.10);

select * from salesperson;

Output :

sno	sname	city	comm
1001	peel	london	0.12
1002	serres	san jose	0.13
1003	axelrod	new york	0.10
1004	motika	london	0.11
1007	rafkin	barcelona	0.15
NULL	NULL	NULL	NULL

create table customer(cnm int primary key, cname varchar(100), city varchar(100), rating int, sno int,foreign key (sno) references salesperson(sno));

insert into customer (cnm, cname, city, rating, sno) values

(201, 'hoffman', 'london', 100, 1001),

(202, 'giovanne', 'roe', 200, 1003),

(203, 'liu', 'san jose', 300, 1002),

(204, 'grass', 'barcelona', 100, 1007),

(206, 'clemens', 'london', 300, 1007),

(207, 'pereira', 'roe', 100, 1004);

select * from customer;

Output :

cnm	cname	city	rating	sno
201	hoffman	london	100	1001
202	giovanne	roe	200	1003
203	liu	san jose	300	1002
204	grass	barcelona	100	1007
206	clemens	london	300	1007
207	pereira	roe	100	1004
NULL	NULL	NULL	NULL	NULL

2. Names and cities of all salespeople in London with commission above 0.10.

ANSWER →

use shop;

select sname, city, comm from salesperson where city = 'london' and comm > 0.10;

Output :

sname	city	comm
peel	london	0.12
motika	london	0.11

3. All salespeople either in Barcelona or in London.

ANSWER →

use shop;

select * from salesperson where city in ('barcelona', 'london');

Output :

sno	sname	city	comm
1001	peel	london	0.12
1004	motika	london	0.11
1007	rafkin	barcelona	0.15
NULL	NULL	NULL	NULL

4. All salespeople with commission between 0.10 and 0.12. (Boundary values should be excluded).

ANSWER →

use shop;

select * from salesperson where comm > 0.10 and comm < 0.12;

Output :

sno	sname	city	comm
1004	motika	london	0.11
NULL	NULL	NULL	NULL

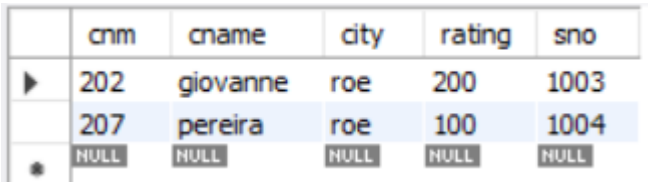
5. All customers excluding those with rating > 100 unless they are located inRome.

ANSWER →

use shop;

select * from customer where rating < 100 or city = 'roe';

Output :



	cnm	cname	city	rating	sno
▶	202	giovanne	roe	200	1003
	207	pereira	roe	100	1004
*	NULL	NULL	NULL	NULL	NULL