1. What do you understand By Database?

- A database is an organized collection of structured information, or data, typically stored electronically in a computer system. A database is usually controlled by a database management system (DBMS).

2. What is Normalization?

- As an SQL Developer, you often work with enormous amounts of data stored in different tables that are present inside multiple databases. It often becomes strenuous to extract the data if it is not organized correctly. Using Normalization, you can solve the problem of data redundancy and organize the data using different forms. This tutorial will help you get to know the concept of Normalization in SQL.

3. What is Difference between DBMS and RDBMS?

RDBMS	DBMS
Data stored is in table format	Data stored is in the file format
Multiple data elements are accessible together	Individual access of data elements
Data in the form of a table are linked together	No connection between data
Normalisation is not achievable	There is normalisation

Support distributed database	No support for distributed database
Data is stored in a large amount	Data stored is a small quantity
Here, redundancy of data is reduced with the help of key and indexes in RDBMS	Data redundancy is common
RDBMS supports multiple users	DBMS supports a single user
It features multiple layers of security while handling data	There is only low security while handling data
The software and hardware requirements are higher	The software and hardware requirements are low
Oracle, SQL Server.	XML, Microsoft Access.

4. What is MF Cod Rule of RDBMS Systems?

- Codd’s rule in DBMS also known as Codd's 12 rules/commandments is a set of thirteen rules (numbered 0 to 12) that define a database to be a correct Relational Database Management System (RDBMS). If a database follows Codd's 12 rules, it is called a True relational database management system. These rules were originally set out in 1970 by Edgar F. Codd and were developed further by him in 1985.

5. What do you understand By Data Redundancy?

- **Possible data inconsistency: -**
  - Data redundancy occurs when the same piece of data exists in multiple places, whereas data inconsistency is when the same data exists in different formats in multiple tables. Unfortunately, data redundancy can cause data inconsistency, which can provide a company with unreliable and/or meaningless information.
- **Increase in data corruption: -**
  - Data corruption is when data becomes damaged as a result of errors in writing, reading, storage, or processing. When the same data fields are repeated in a database or file storage system, data corruption arises. If a file gets corrupted, for example, and an employee tries to open it, they may get an error message and not be able to complete their task.

6. What is DDL Interpreter?

- DDL Interpreter DDL expands to Data Definition Language. DDL Interpreter as the name suggests interprets the DDL statements such as schema definition statements like create, delete, etc. The result of this interpretation is a set of a table that contains the meta-data which is stored in the data dictionary.

7. What is DML Compiler in SQL?

- DML Compiler: It processes the DML statements into low level instruction (machine language), so that they can be executed. DDL Interpreter: It processes the DDL statements into a set of tables containing meta data (data about data).

- INSERT: command to add new or new value to the database.
- UPDATE: command to change or update current/existing data to a more recent value within the database.
- DELETE: command to delete or delete the values or data information of the current table in the database.

8. What is SQL Key Constraints writing an Example of SQL Key Constraints?

- SQL constraints are used to specify rules for the data in a table.
- Constraints are used to limit the type of data that can go into a table. This ensures the accuracy and reliability of the data in the table. If there is any violation between the constraint and the data action, the action is aborted.
- Constraints can be column level or table level. Column level constraints apply to a column, and table level constraints apply to the whole table.

9. What is save Point? How to create a save Point write a Query?

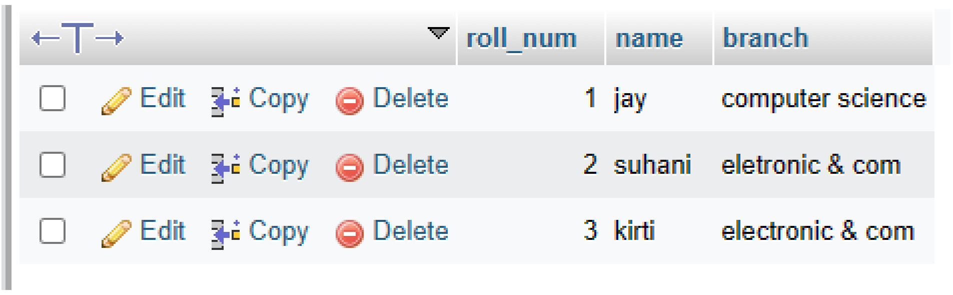
- A SAVEPOINT is a point in a transaction in which you can roll the transaction back to a certain point without rolling back the entire transaction. Syntax for Save point command: SAVEPOINT SAVEPOINT_NAME; This command is used only in the creation of SAVEPOINT among all the transactions.

10. What is trigger and how to create a Trigger in SQL?

- A trigger is a special type of stored procedure that automatically runs when an event occurs in the database server. DML triggers run when a user tries to modify data through a data manipulation language (DML) event. DML events are INSERT, UPDATE, or DELETE statements on a table or view.
- DDL triggers run in response to a variety of data definition language (DDL) events. These events primarily correspond to Transact-SQL CREATE, ALTER, and DROP statements, and certain system stored procedures that perform DDL-like operations.

11. create a table student.

```
1.  CREATE TABLE student (roll_num int AUTO_INCREMENT,
                  name varchar (30),
                  branch varchar (30),
                 PRIMARY KEY (roll_num)
               );
2. INSERT INTO student VALUES ('',"jay”, “computer science");
   INSERT INTO student VALUES ('',"suhani”, “eletronic & com");
```



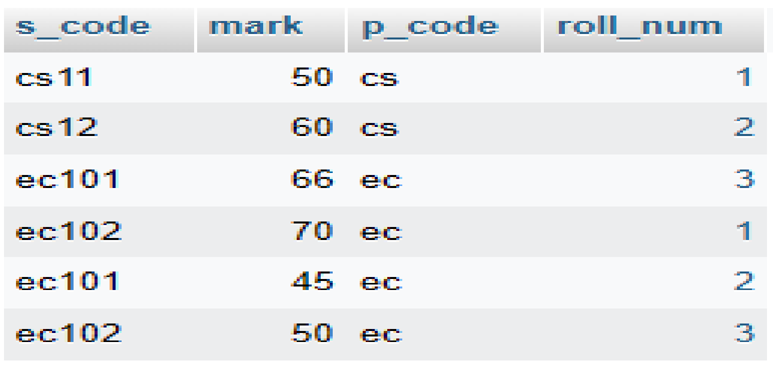
				roll_num	name	branch
☐	Edit	Copy	Delete	1	jay	computer science
☐	Edit	Copy	Delete	2	suhani	eletronic & com
☐	Edit	Copy	Delete	3	kirti	electronic & com

12. Create Table Exam

```
1. CREATE TABLE exam (s_code int AUTO_INCREMENT,
                  mark int,
                  p_code varchar (30),
                  roll_num int,
                  FOREIGN KEY (roll_num) REFERENCES student(roll_num)
                  );
```

- INSERT INTO `exam` (`s_code`, `mark`, `p_code`, `roll_num`) VALUES ('cs11', '50', 'cs', '1');
- INSERT INTO `exam` (`s_code`, `mark`, `p_code`, `roll_num`) VALUES ('cs12', '60', 'cs', '2');
- INSERT INTO `exam` (`s_code`, `mark`, `p_code`, `roll_num`) VALUES ('ec101', '66', 'ec', '3');

- INSERT INTO `exam` (`s_code`, `mark`, `p_code`, `roll_num`) VALUES ('ec102', '70', 'ec', '1');
- INSERT INTO `exam` (`s_code`, `mark`, `p_code`, `roll_num`) VALUES ('ec101', '45', 'ec', '2');
- INSERT INTO `exam` (`s_code`, `mark`, `p_code`, `roll_num`) VALUES ('ec102', '50', 'ec', '3');



s_code	mark	p_code	roll_num
cs11	50	cs	1
cs12	60	cs	2
ec101	66	ec	3
ec102	70	ec	1
ec101	45	ec	2
ec102	50	ec	3

13. create table people.

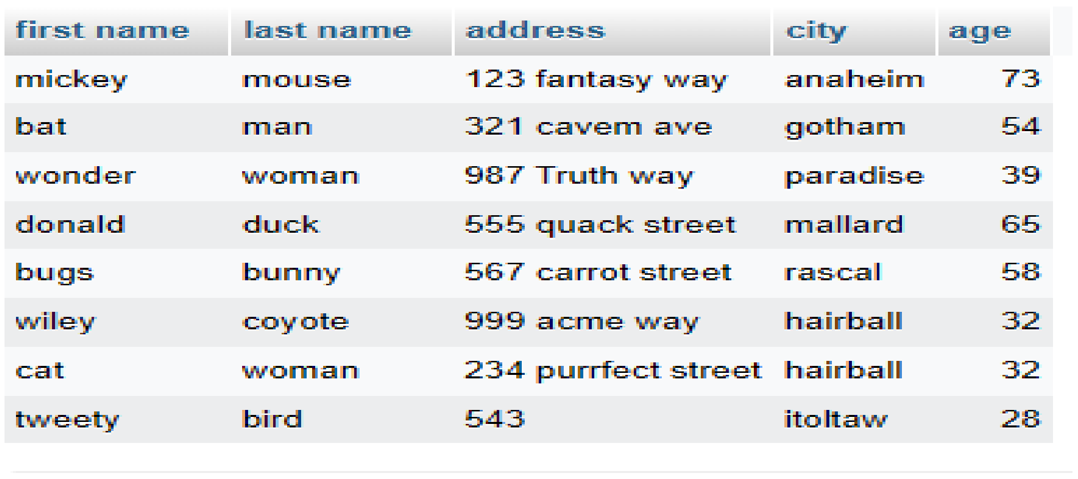
- CREATE TABLE people (first name varchar (20),

  Last name varchar (20),

  Address varchar (50),

  city varchar (30),

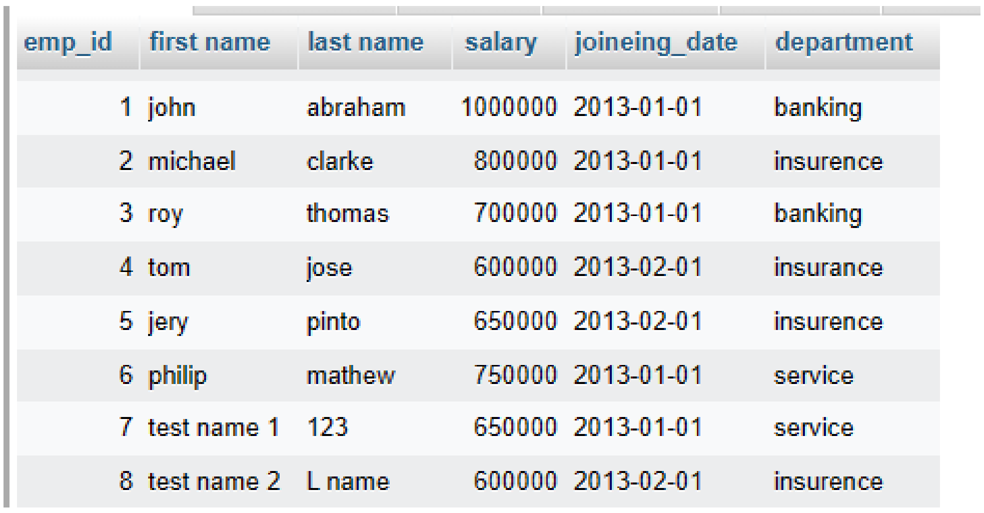
  age int);



first name	last name	address	city	age
mickey	mouse	123 fantasy way	anaheim	73
bat	man	321 cavem ave	gotham	54
wonder	woman	987 Truth way	paradise	39
donald	duck	555 quack street	mallard	65
bugs	bunny	567 carrot street	rascal	58
wiley	coyote	999 acme way	hairball	32
cat	woman	234 purrfect street	hairball	32
tweety	bird	543	itoltaw	28

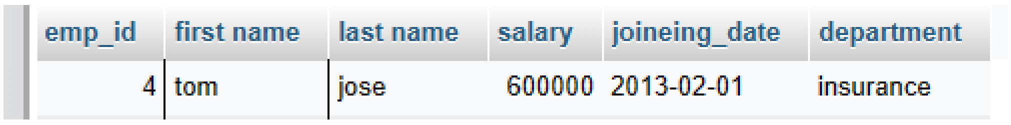
14. create table employee

- CREATE TABLE `employe` (`emp_id` INT NOT NULL , `first name` VARCHAR(30) NOT NULL , `last name` VARCHAR(30) NOT NULL , `salary` INT NOT NULL , `joineing_date` DATE NOT NULL , `department` VARCHAR(30));
- INSERT INTO `employe` (`emp_id`, `first name`, `last name`, `salary`, `joineing_date`, `department`) VALUES ('1', 'john', 'abraham', '1000000', '2013-01-01', 'banking');
- INSERT INTO `employe` (`emp_id`, `first name`, `last name`, `salary`, `joineing_date`, `department`) VALUES ('2', 'michael', 'clarke', '800000', '2013-01-01', 'insurence');
- INSERT INTO `employe` (`emp_id`, `first name`, `last name`, `salary`, `joineing_date`, `department`) VALUES ('3', 'roy', 'thomas', '700000', '2013-01-01', 'banking');
- INSERT INTO `employe` (`emp_id`, `first name`, `last name`, `salary`, `joineing_date`, `department`) VALUES ('4', 'tom', 'jose', '600000', '2013-02-01', 'insurance');
- INSERT INTO `employe` (`emp_id`, `first name`, `last name`, `salary`, `joineing_date`, `department`) VALUES ('5', 'jery', 'pinto', '650000', '2013-02-01', 'insurence');
- INSERT INTO `employe` (`emp_id`, `first name`, `last name`, `salary`, `joineing_date`, `department`) VALUES ('6', 'philip', 'mathew', '750000', '2013-01-01', 'service');
- INSERT INTO `employe` (`emp_id`, `first name`, `last name`, `salary`, `joineing_date`, `department`) VALUES ('7', 'test name 1', '123’, '650000', '2013-01-01', service);
- INSERT INTO `employe` (`emp_id`, `first name`, `last name`, `salary`, `joineing_date`, `department`) VALUES ('8', 'test name 2', 'last name', '600000', '2013-02-01', 'insurence');



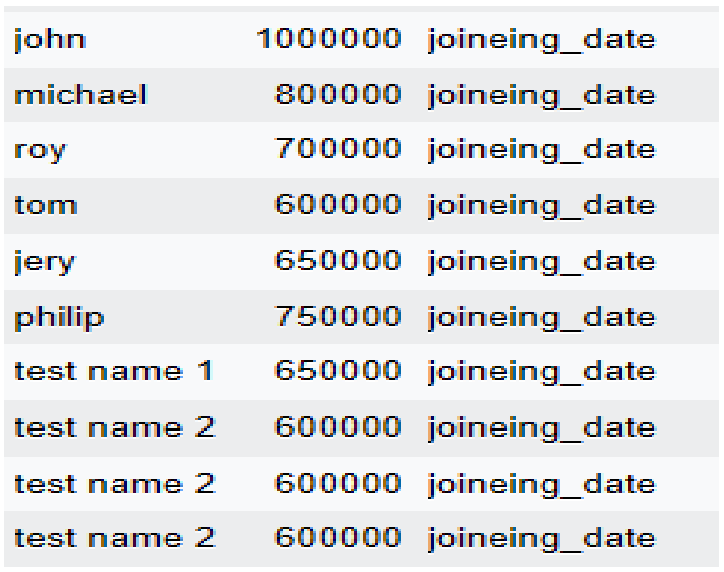
emp_id	first name	last name	salary	joineing_date	department
1	john	abraham	1000000	2013-01-01	banking
2	michael	clarke	800000	2013-01-01	insurence
3	roy	thomas	700000	2013-01-01	banking
4	tom	jose	600000	2013-02-01	insurance
5	jery	pinto	650000	2013-02-01	insurence
6	philip	mathew	750000	2013-01-01	service
7	test name 1	123	650000	2013-01-01	service
8	test name 2	L name	600000	2013-02-01	insurence

1.SELECT * FROM `employe` WHERE 'first name' ='tom’;



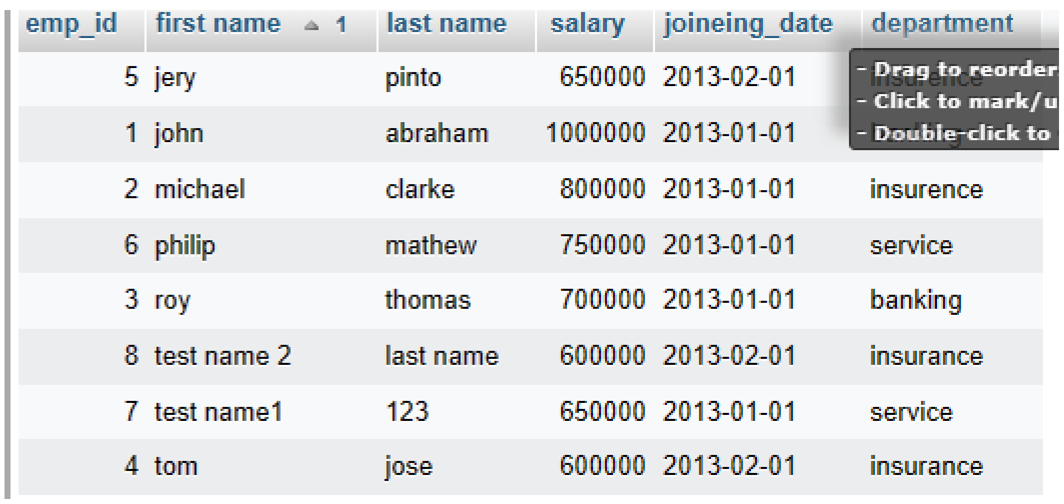
emp_id	first name	last name	salary	joineing_date	department
4	tom	jose	600000	2013-02-01	insurance

2. SELECT `first name`, `salary`, 'joineing_date' FROM `employe`;



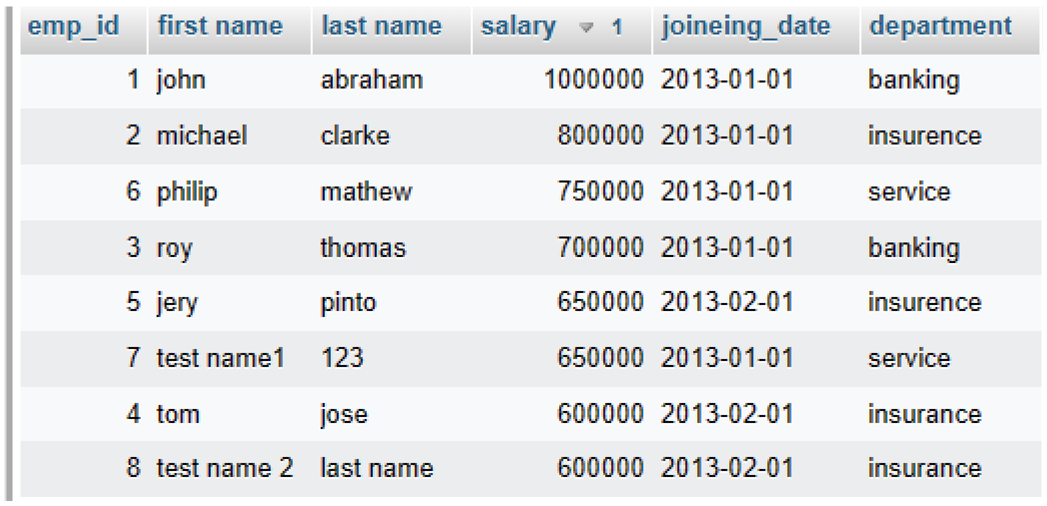
john	1000000	joineing_date
michael	800000	joineing_date
roy	700000	joineing_date
tom	600000	joineing_date
jery	650000	joineing_date
philip	750000	joineing_date
test name 1	650000	joineing_date
test name 2	600000	joineing_date
test name 2	600000	joineing_date
test name 2	600000	joineing_date

3. SELECT * FROM `employe` ORDER BY `first name` asc;



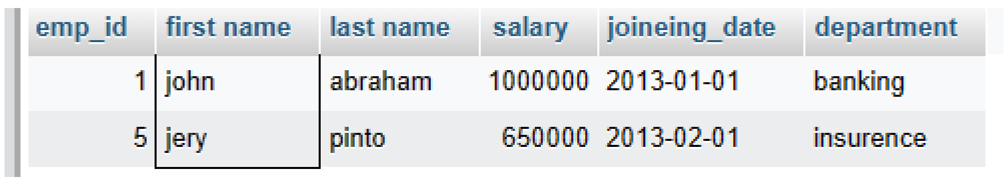
emp_id	first name ▲ 1	last name	salary	joineing_date	department
5	jery	pinto	650000	2013-02-01	insurence
1	john	abraham	1000000	2013-01-01	[illegible]
2	michael	clarke	800000	2013-01-01	insurence
6	philip	mathew	750000	2013-01-01	service
3	roy	thomas	700000	2013-01-01	banking
8	test name 2	last name	600000	2013-02-01	insurance
7	test name1	123	650000	2013-01-01	service
4	tom	jose	600000	2013-02-01	insurance

4. SELECT * FROM `employe` ORDER BY salary DESC;



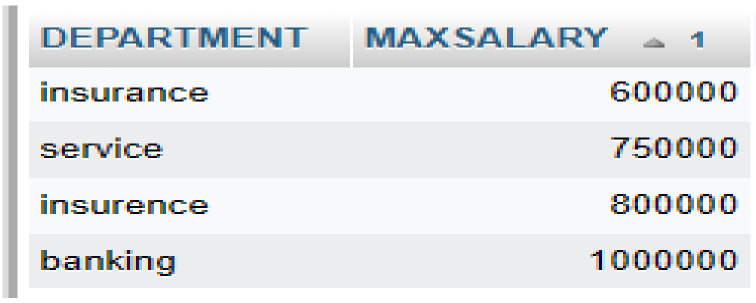
emp_id	first name	last name	salary ▼ 1	joineing_date	department
1	john	abraham	1000000	2013-01-01	banking
2	michael	clarke	800000	2013-01-01	insurence
6	philip	mathew	750000	2013-01-01	service
3	roy	thomas	700000	2013-01-01	banking
5	jery	pinto	650000	2013-02-01	insurence
7	test name1	123	650000	2013-01-01	service
4	tom	jose	600000	2013-02-01	insurance
8	test name 2	last name	600000	2013-02-01	insurance

5. SELECT * FROM `employe` WHERE `first name` IN ('john','jery');



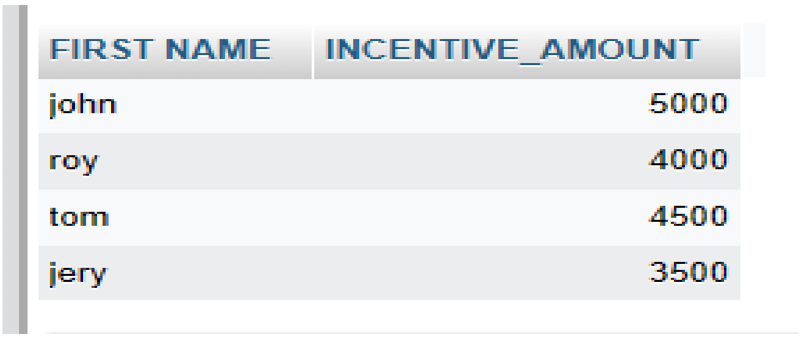
emp_id	first name	last name	salary	joineing_date	department
1	john	abraham	1000000	2013-01-01	banking
5	jery	pinto	650000	2013-02-01	insurence

6. SELECT DEPARTMENT, MAX(SALARY) MAXSALARY FROM EMPLOYE GROUP BY DEPARTMENT ORDER BY MAXSALARY ASC;



DEPARTMENT	MAXSALARY ▲ 1
insurance	600000
service	750000
insurence	800000
banking	1000000

7. SELECT `FIRST NAME`,INCENTIVE_AMOUNT FROM EMPLOYE A INNER JOIN INCENTIVE B ON A.EMP_ID=B.EMP_REF_ID AND INCENTIVE_AMOUNT >3000;

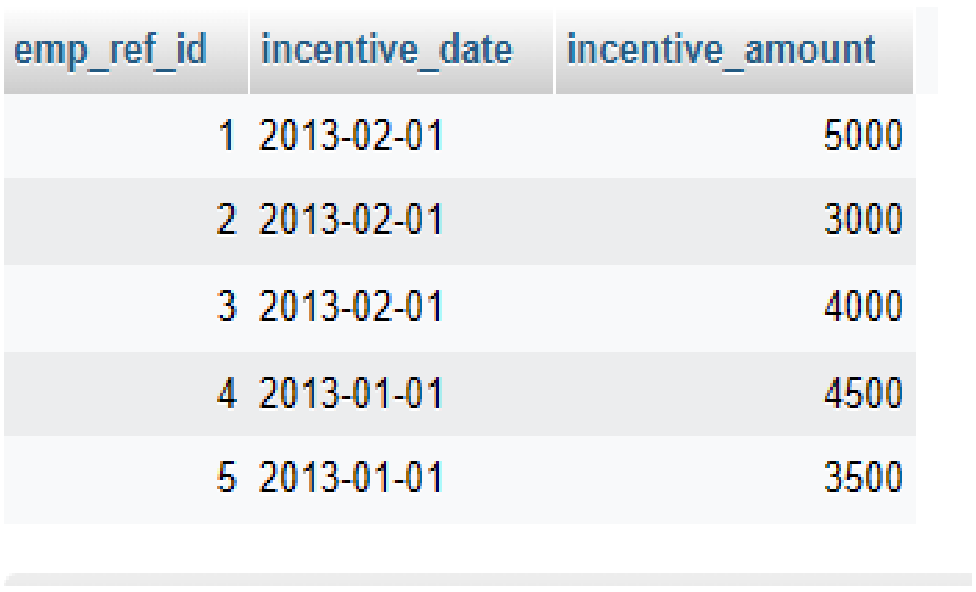


FIRST NAME	INCENTIVE_AMOUNT
john	5000
roy	4000
tom	4500
jery	3500

15. create table incentive

- CREATE TABLE incentive` (`emp_ref_id` INT NOT NULL, `incentive_date` DATE NOT NULL, `incentive_amount` INT NOT NULL) ENGINE = InnoDB;

- INSERT INTO `incentive` (`emp_ref_id`, `incentive_date`, `incentive_amount`) VALUES ('1', '2013-02-01', '5000'), ('2', '2013-02-01', '3000'), ('3', '2013-02-01', '4000'), ('4', '2013-01-01', '4500'), ('5', '2013-01-01', '3500');

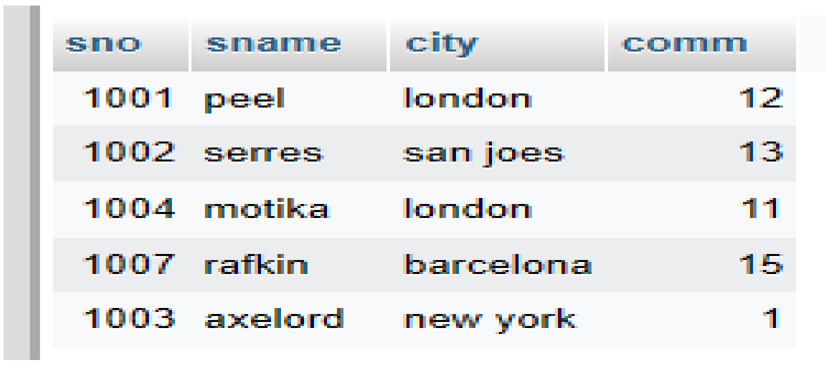


emp_ref_id	incentive_date	incentive_amount
1	2013-02-01	5000
2	2013-02-01	3000
3	2013-02-01	4000
4	2013-01-01	4500
5	2013-01-01	3500

16. create table salesman

- CREATE TABLE`salesperson` (`sno` INT NOT NULL, `sname` VARCHAR (50) NOT NULL, `city` VARCHAR (30) NOT NULL, `comm` INT NOT NULL) ENGINE = InnoDB;
- Expand Requery Edit Bookmark Database: assinment Queried time: 17:12:14

- INSERT INTO `salesperson` (`sno`, `sname`, `city`, `comm`) VALUES ('1001', 'peel', 'london', '12'), ('1002', 'serres', 'san joes', '13'), ('1004', 'motika', 'london', '11'), ('1007', 'rafkin', 'barcelona', '15'), ('1003', 'axelord', 'new york', '1');

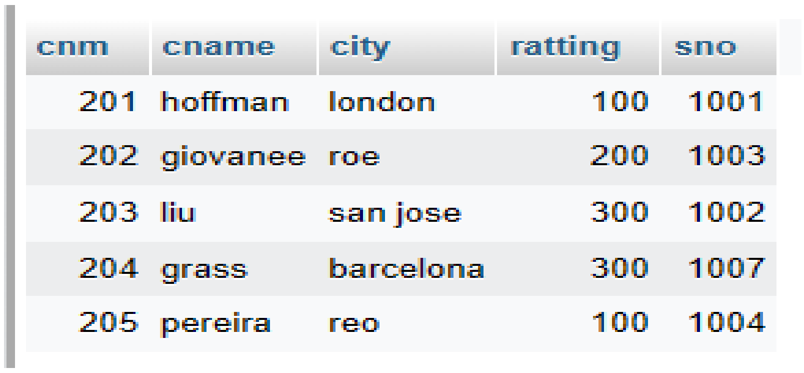


sno	sname	city	comm
1001	peel	london	12
1002	serres	san joes	13
1004	motika	london	11
1007	rafkin	barcelona	15
1003	axelord	new york	1

17. create table customer

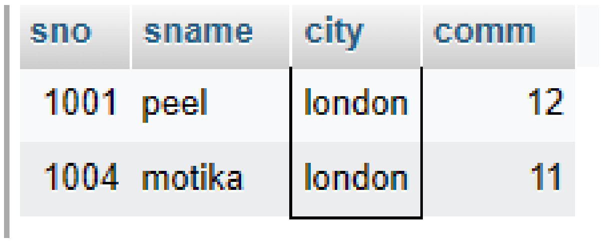
- CREATE TABLE `customer` (`cnm` INT NOT NULL, `cname` VARCHAR (50) NOT NULL, `city` VARCHAR (30) NOT NULL, `ratting` INT NOT NULL, `sno` INT NOT NULL) ENGINE = InnoDB;

- INSERT INTO `customer` (`cnm`, `cname`, `city`, `ratting`, `sno`) VALUES ('201', 'hoffman', 'london', '100', '1001'), ('202', 'giovanee', 'roe', '200', '1003'), ('203', 'liu', 'san jose', '300', '1002'), ('204', 'grass', 'barcelona', '300', '1007'), ('205', 'pereira', 'reo', '100', '1004');

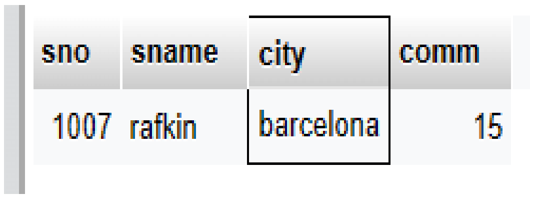


cnm	cname	city	ratting	sno
201	hoffman	london	100	1001
202	giovanee	roe	200	1003
203	liu	san jose	300	1002
204	grass	barcelona	300	1007
205	pereira	reo	100	1004

1. select * from order where amount> 1000;
2. SELECT sname, city from salesperson where city='london' and comm>0.12;
3. All salespeople either in Barcelona or in London.
   a. SELECT * FROM salesperson WHERE city='barcelona';
4. SELECT * FROM salesperson WHERE city=london;

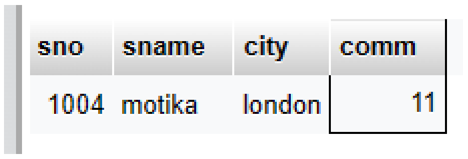


sno	sname	city	comm
1001	peel	london	12
1004	motika	london	11



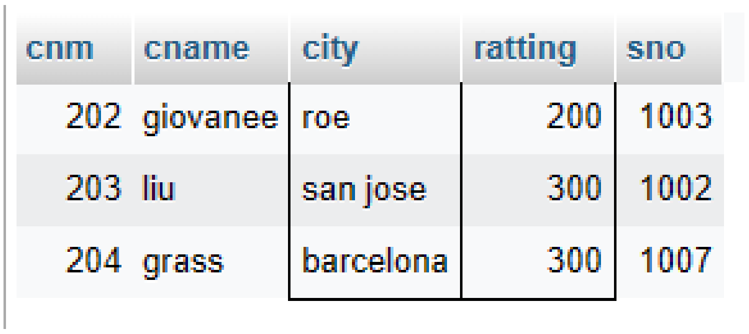
sno	sname	city	comm
1007	rafkin	barcelona	15

4. SELECT *FROM salesperson WHERE (comm > 10 AND comm< 12);



sno	sname	city	comm
1004	motika	london	11

5. SELECT *FROM customer WHERE ratting>100 OR city='rome';



cnm	cname	city	ratting	sno
202	giovanee	roe	200	1003
203	liu	san jose	300	1002
204	grass	barcelona	300	1007