**• 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).

**• What is Normalization?**

Normalization is the process of organizing data in a database. It includes creating tables and establishing relationships between those tables according to rules designed both to protect the data and to make the database more flexible by eliminating redundancy and inconsistent dependency.

**• What is Difference between DBMS and RDBMS?**

DBMS stands for Database Management System, and RDBMS is the acronym for the Relational Database Management system. In DBMS, the data is stored as a file, whereas in RDBMS, data is stored in the form of tables.

**• What is MF Cod Rule of RDBMS Systems?**

**Rule 0:** The *foundation rule*:

For any system that is advertised as, or claimed to be, a relational data base management system, that system must be able to manage data bases entirely through its relational capabilities.

**Rule 1:** The *information rule*:

All information in a relational data base is represented explicitly at the logical level and in exactly one way – by values in tables.

**Rule 2:** The *guaranteed access rule*:

Each and every datum (atomic value) in a relational data base is guaranteed to be logically accessible by resorting to a combination of table name, primary key value and column name.

**Rule 3:** *Systematic treatment of null values*:

Null values (distinct from the empty character string or a string of blank characters and distinct from zero or any other number) are supported in fully relational DBMS for representing missing information and inapplicable information in a systematic way, independent of data type.

**Rule 4:** *Dynamic*[*online*](https://en.wikipedia.org/wiki/Online)*[catalog](https://en.wikipedia.org/wiki/Database_catalog" \o "Database catalog) based on the relational model*:

The data base description is represented at the logical level in the same way as ordinary data, so that authorized users can apply the same relational language to its interrogation as they apply to the regular data.

**Rule 5:** The *comprehensive data sublanguage rule*:

A relational system may support several languages and various modes of terminal use (for example, the fill-in-the-blanks mode). However, there must be at least one language whose statements are expressible, per some well-defined syntax, as character strings and that is comprehensive in supporting all of the following items:

1. Data definition.
2. View definition.
3. Data manipulation (interactive and by program).
4. Integrity constraints.
5. Authorization.
6. Transaction boundaries (begin, commit and rollback).

**Rule 6:** The [*view*](https://en.wikipedia.org/wiki/View_(SQL))*updating rule*:

All views that are theoretically updatable are also updatable by the system.

**Rule 7:** Relational Operations Rule / *Possible for high-level insert, update, and delete*:

The capability of handling a base relation or a derived relation as a single operand applies not only to the retrieval of data but also to the insertion, update and deletion of data.

**Rule 8:** *Physical data independence*:

Application programs and terminal activities remain logically unimpaired whenever any changes are made in either storage representations or access methods.

**Rule 9:** *Logical data independence*:

Application programs and terminal activities remain logically unimpaired when information-preserving changes of any kind that theoretically permit unimpairment are made to the base tables.

**Rule 10:** *Integrity independence*:

Integrity constraints specific to a particular relational data base must be definable in the relational data sublanguage and storable in the catalog, not in the application programs.

**Rule 11:** *Distribution independence*:

The end-user must not be able to see that the data is distributed over various locations. Users should always get the impression that the data is located at one site only.

**Rule 12:** The *nonsubversion rule*:

If a relational system has a low-level (single-record-at-a-time) language, that low level cannot be used to subvert or bypass the integrity rules and constraints expressed in the higher level relational language (multiple-records-at-a-time)

**• What do you understand By Data Redundancy?**

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.

**• 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.

**• What is DML Compiler in SQL?**

**A DML (data manipulation language) refers to a computer programming language that allows you to add (insert), delete (delete), and alter (update) data in a database. A DML is typically a sublanguage of a larger database language like SQL, with the DML containing some of the language's operators**.

**• 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.

**• What is save Point? How to create a save Point write a Query?**

A savepoint is a special mark inside a transaction that allows all commands that are executed after it was established to be rolled back, restoring the transaction state to what it was at the time of the savepoint.

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 Savepoint command: SAVEPOINT SAVEPOINT\_NAME; This command is used only in the creation of SAVEPOINT among all the transactions.

**• What is trigger and how to create a Trigger in SQL?**

 An SQL trigger is a database object that is associated with a table and automatically executes a set of SQL statements when a specific event occurs on that table. Triggers are used to enforce business rules, maintain data integrity, and automate certain actions within a database.

The syntax for creating a trigger is −

CREATE [OR REPLACE ] TRIGGER trigger\_name

{BEFORE | AFTER | INSTEAD OF }

{INSERT [OR] | UPDATE [OR] | DELETE}

[OF col\_name]

ON table\_name

[REFERENCING OLD AS o NEW AS n]

[FOR EACH ROW]

WHEN (condition)

DECLARE

Declaration-statements

BEGIN

Executable-statements

EXCEPTION

Exception-handling-statements

END;

1. **Create Table Name : Student and Exam.**

**Student table:>**

**create table student**

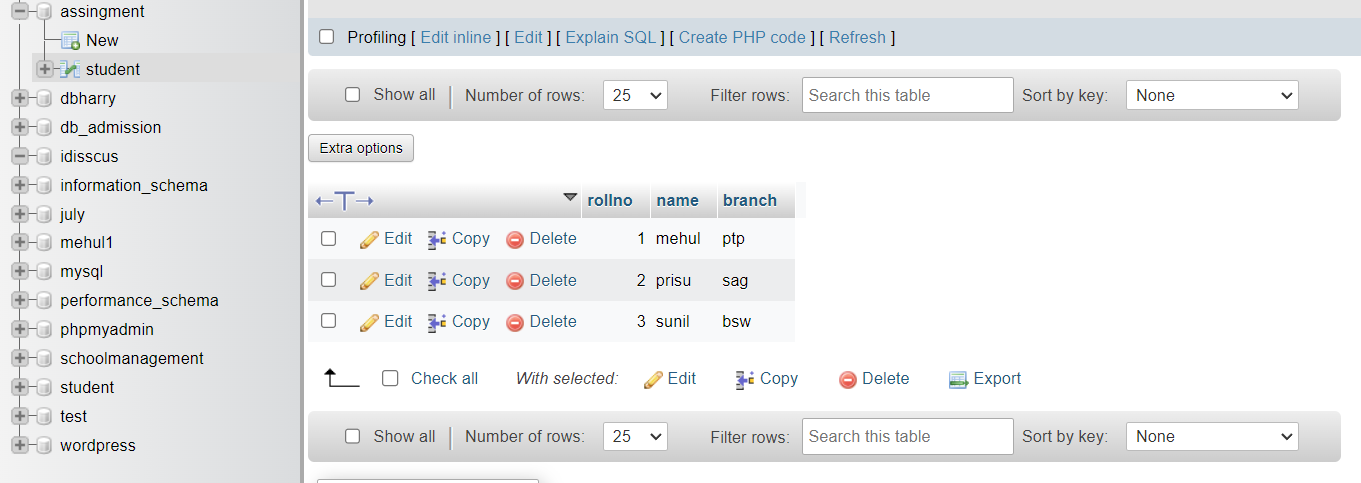
**(**

**rollno int PRIMARY key AUTO\_INCREMENT,**

**name varchar(50),**

**branch varchar(50),**

**);**

****

**Exam table:>**

**Create table exam(**

**Rollno int,**

**s\_code varchar(50),**

**Marks int(50),**

**p\_code varchar(50),**

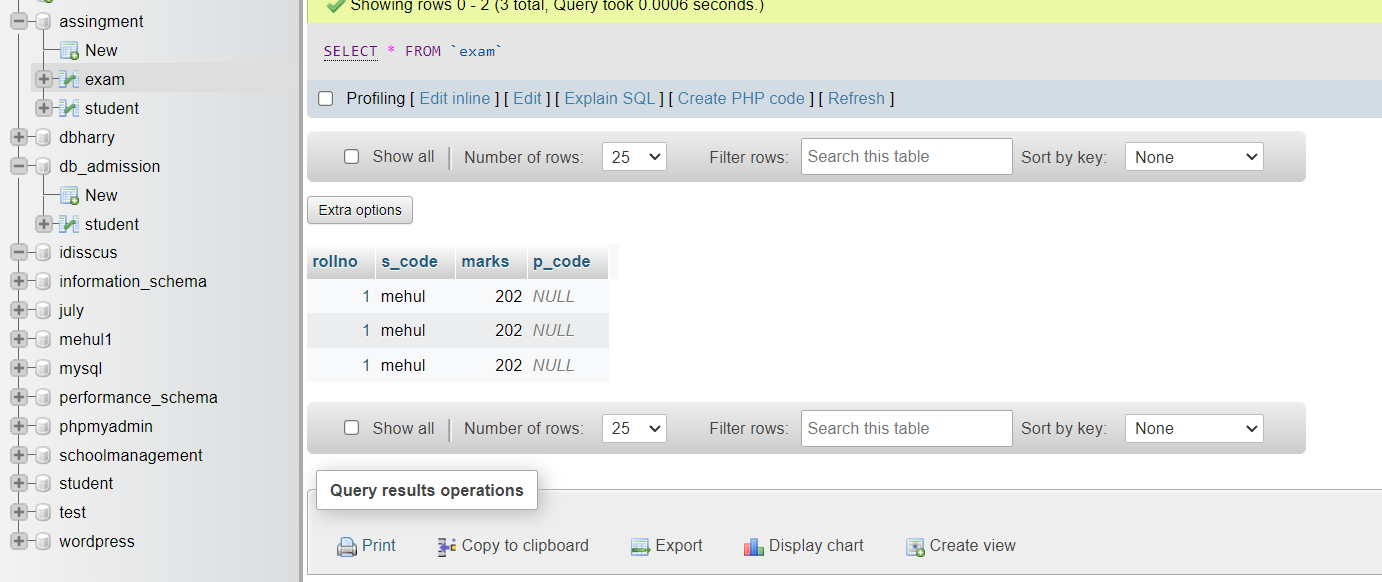
**FOREIGN key (rollno) REFRANCES student(rollno)**

**)**

**INSERT INTO exam(rollno,s\_code,marks) VALUES ('1','mehul','202');**

**INSERT INTO exam(rollno,s\_code,marks) VALUES ('1','mehul','202');**

**INSERT INTO exam(rollno,s\_code,marks) VALUES ('1','mehul','202');**

****

**Create employee table:>**

**CREATE TABLE employee(**

**employee\_id int PRIMARY KEY AUTO\_INCREMENT,**

**first\_name varchar(50),**

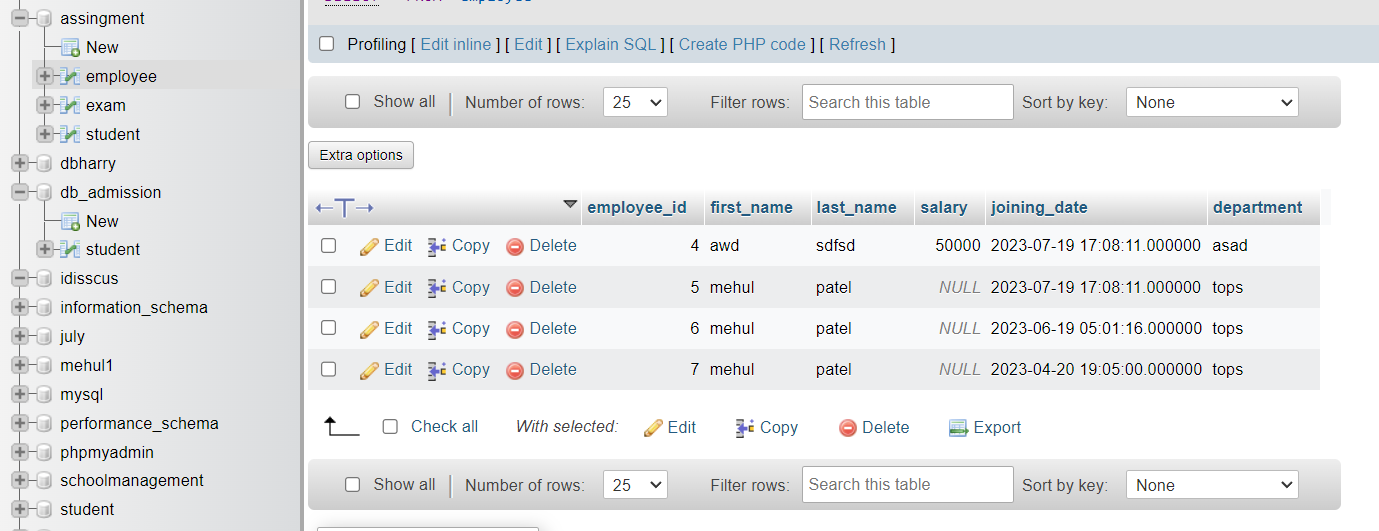
**last\_name varchar(50),**

**salary int(50),**

**joining\_date DATETIME(6),**

**department varchar(50)**

**)**

****

**Create incentive table:>**

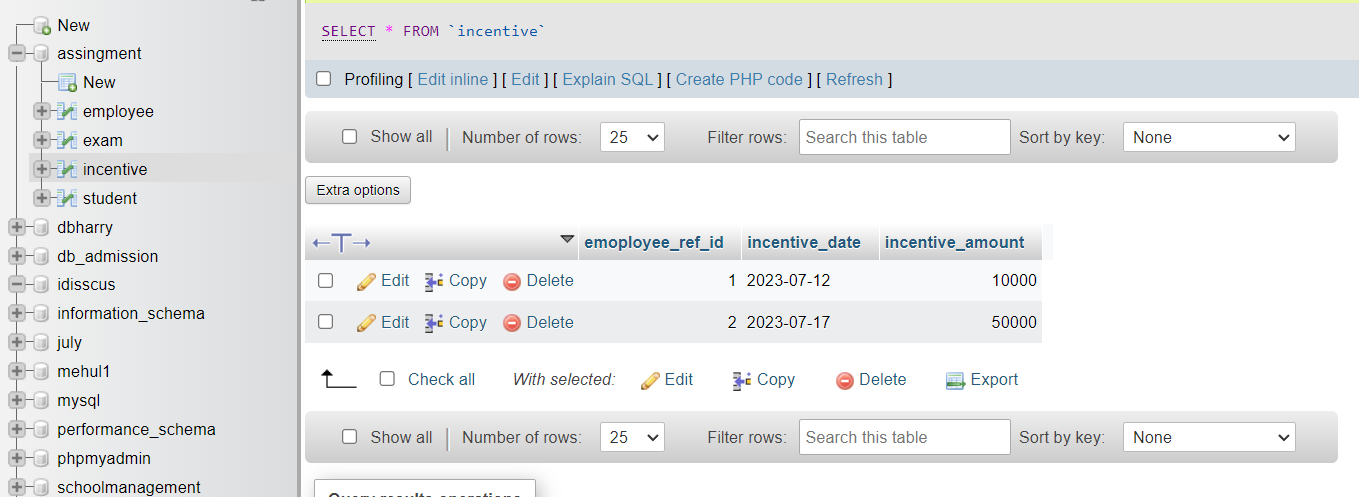
**CREATE TABLE incentive(**

**emoployee\_ref\_id int PRIMARY KEY AUTO\_INCREMENT,**

**incentive\_date DATE,**

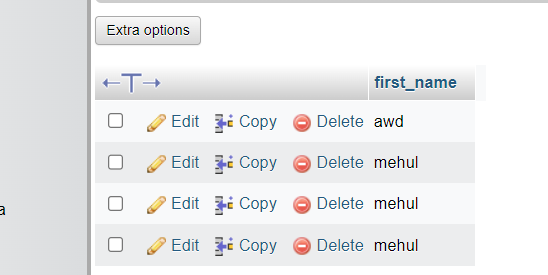
**incentive\_amount int(50)**

**)**

****

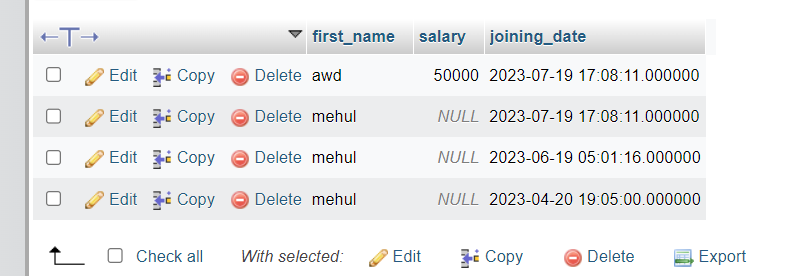
1. **Get First\_Name from employee table using Tom name “Employee Name”?**

**SELECT first\_name FROM employee**

****

1. **Get FIRST\_NAME, Joining Date, and Salary from employee table.**

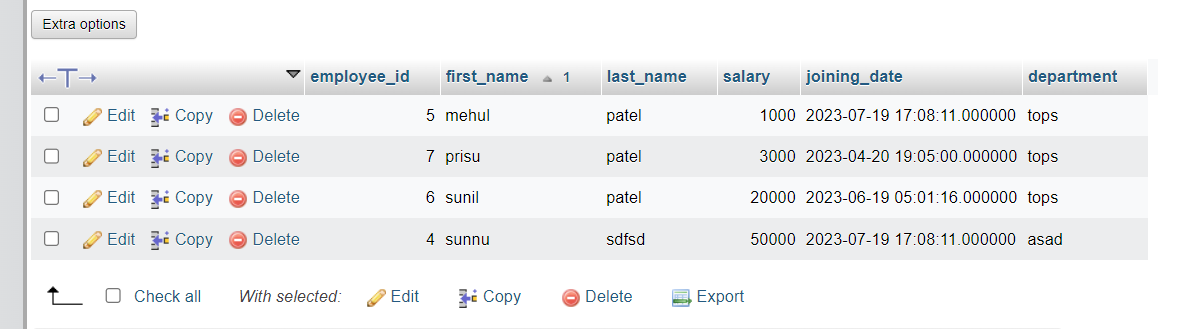
**SELECT first\_name,salary,joining\_date FROM `employee`**

****

1. **Get all employee details from the employee table order by First\_Name Ascending and Salary descending?**

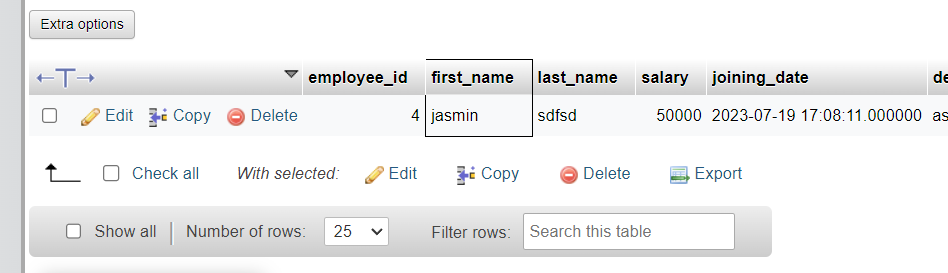
**SELECT \* FROM `employee` ORDER BY first\_name;**

**SELECT \* FROM `employee` ORDER BY salary DESC;**

****

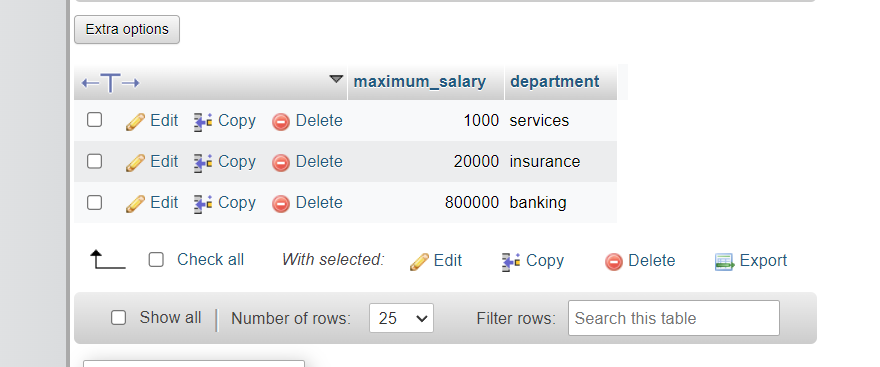
1. **Get employee details from employee table whose first name contains ‘J’.**

**SELECT \* FROM `employee` WHERE first\_name LIKE "j%";**

****

1. **Get department wise maximum salary from employee table order by salaryascending?**

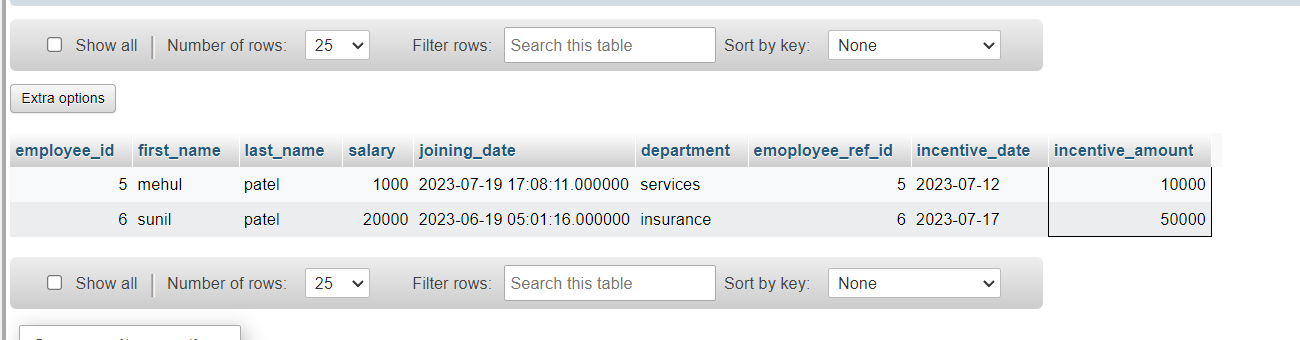
**SELECT MAX(salary) as maximum\_salary,department FROM `employee` GROUP BY department ORDER BY salary ;**

****

1. **Select first\_name, incentive amount from employee and incentives table forthose employees who have incentives and incentive amount greater than 3000.**

**SELECT \* FROM employee INNER join incentive on employee.employee\_id=incentive.emoployee\_ref\_id**

**WHERE incentive\_amount >3000**

****

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

**Create table given below: Salesperson and Customer:**

**Salesperson>**

**:** CREATE TABLE salesperson(

sno int PRIMARY KEY AUTO\_INCREMENT,

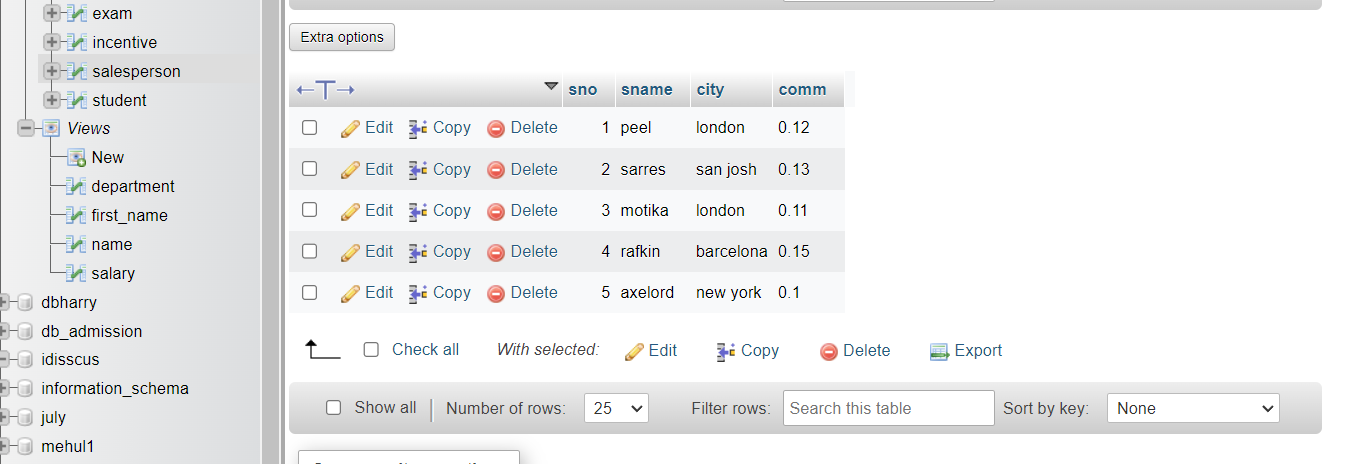
sname varchar(50),

city varchar(50),

comm varchar(50)

)

INSERT INTO salesperson(sname,city,comm)VALUES('peel','london',.12),('sarres','san josh',.13),('motika','london',.11),('rafkin','barcelona',.15),('axelord','new york',.1)



**Customer:>**

CREATE TABLE customer(

cnm int PRIMARY KEY AUTO\_INCREMENT,

cname varchar(50),

city varchar(50),

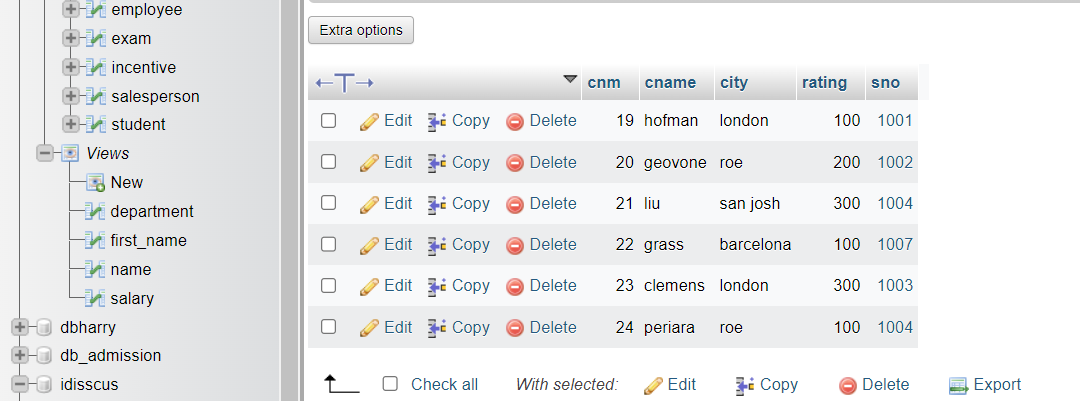
rating int(50),

sno int,

FOREIGN key(sno) REFERENCES salesperson(sno)

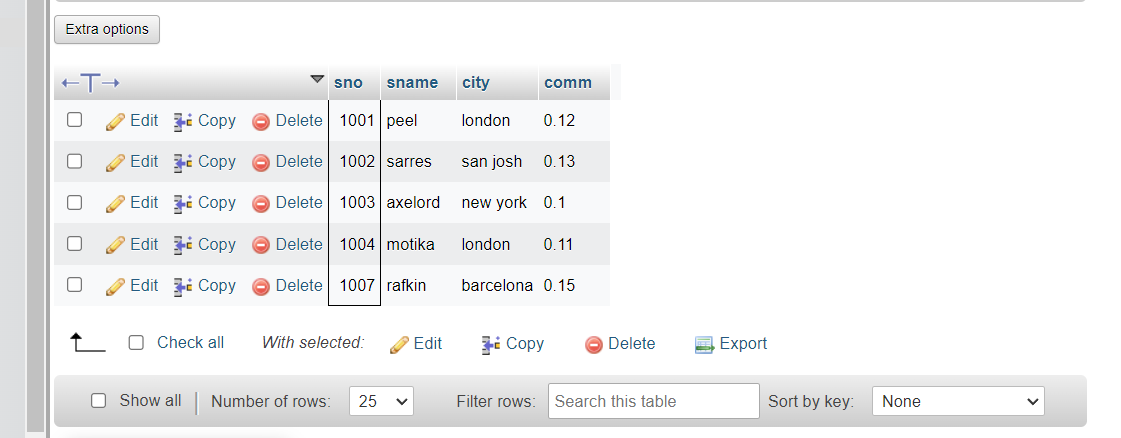
)

INSERT INTO customer(cname,city,rating,sno) VALUES ('hofman','london','100','1001'),('geovone','roe','200','1002'),('liu','san josh','300','1004'),('grass','barcelona','100','1007'),('clemens','london','300','1003'),('periara','roe','100','1004')



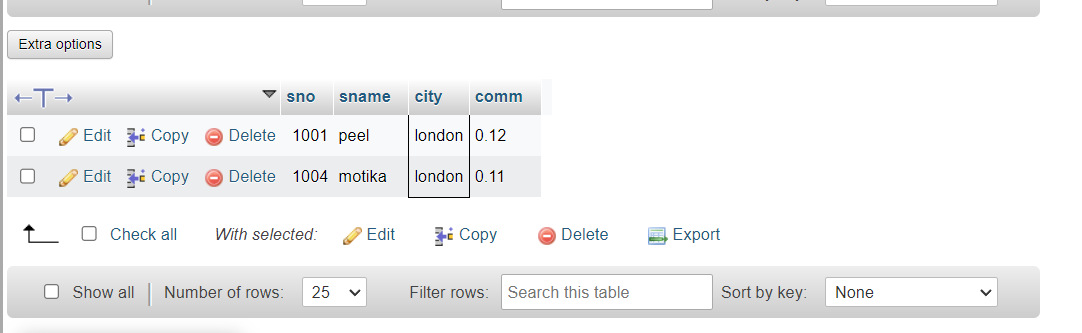
1. All orders for more than $1000.

SELECT \* FROM `salesperson` WHERE sno>1000



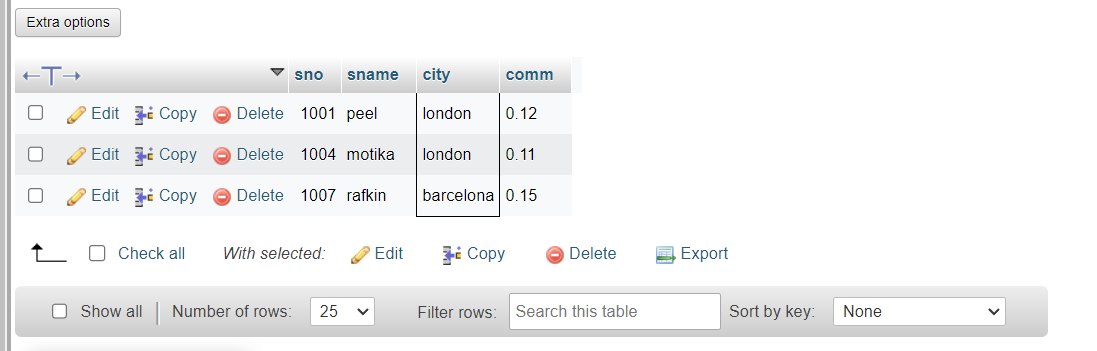
1. Names and cities of all salespeople in London with commission above 0.12.

SELECT \* FROM `salesperson` WHERE city='london'



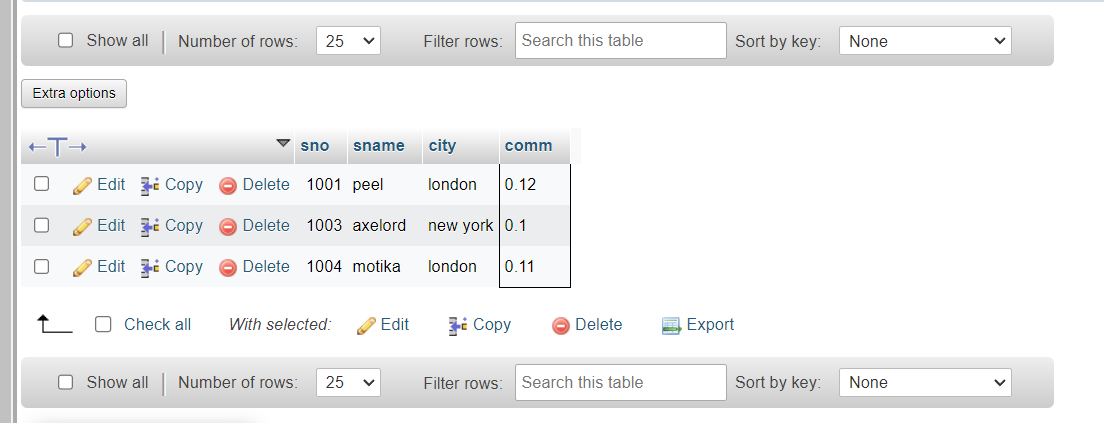
1. All salespeople either in Barcelona or in London.

SELECT \* FROM `salesperson` WHERE city in ('barcelona', 'london')



1. All salespeople with commission between 0.10 and 0.12. (Boundary valuesshould be excluded).

SELECT \* FROM `salesperson` WHERE comm BETWEEN 0.10 and 0.12



b) All customers excluding those with rating <= 100 unless they are located inRome.

SELECT \* FROM `customer` WHERE rating>=100 AND NOT city='roe'

