**Module:4 Database**

• 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)](https://www.oracle.com/in/database/what-is-database/#WhatIsDBMS). Together, the data and the DBMS, along with the applications that are associated with them, are referred to as a database system, often shortened to just database.

-Data within the most common types of databases in operation today is typically modeled in rows and columns in a series of tables to make processing and data querying efficient. The data can then be easily accessed, managed, modified, updated, controlled, and organized. Most databases use structured query language (SQL) for writing and querying data.

-----

• What is Normalization?

- Normalization is a process in database design that involves organizing data in a relational database to minimize redundancy and dependency issues. It aims to structure the database schema in a way that eliminates data anomalies and improves data integrity and efficiency.

The normalization process typically involves breaking down a database table into multiple tables, based on specific rules and principles. These rules are defined by normal forms, which provide guidelines for organizing data effectively.

The most commonly used normal forms are:

1. First Normal Form (1NF): This form requires that each column in a table contains atomic (indivisible) values. It eliminates repeating groups and ensures that each attribute holds only a single value.

2. Second Normal Form (2NF): In addition to meeting the requirements of 1NF, this form addresses partial dependencies. It means that no non-key attribute should be functionally dependent on only a portion of the primary key. If such dependencies exist, the table should be split into multiple tables.

3. Third Normal Form (3NF): In addition to the requirements of 2NF, this form deals with transitive dependencies. It states that no non-key attribute should be functionally dependent on another non-key attribute. If a transitive dependency exists, the dependent attributes should be moved to a separate table.

- There are higher normal forms beyond 3NF, such as Boyce-Codd Normal Form (BCNF), Fourth Normal Form (4NF), and Fifth Normal Form (5NF), each addressing more complex dependency scenarios. These higher normal forms are generally used in advanced cases where specific data anomalies need to be resolved.

- By following the normalization process and applying the appropriate normal forms, the database designer can ensure that data is organized efficiently, minimizing redundancy, and improving data integrity. This leads to more robust and maintainable database structures, reducing the risk of data inconsistencies and anomalies.

-----

• What is Difference between DBMS and RDBMS?

- In short, the main differences between DBMS (Database Management System) and RDBMS (Relational Database Management System) can be summarized as follows:

- Data Model: DBMS supports various data models (hierarchical, network, relational), while RDBMS specifically deals with the relational data model.

- Data Structure: DBMS can use different data structures, while RDBMS organizes data into tables with rows and columns.

- Data Integrity and Constraints: DBMS may or may not enforce strict integrity constraints, whereas RDBMS enforces strong integrity constraints (e.g., primary key uniqueness, referential integrity).

- Query Language: DBMS provides a data retrieval language specific to the data model, while RDBMS uses the standardized Structured Query Language (SQL).

- Scalability and Performance: RDBMS is designed for handling large-scale data and offers optimized performance for complex queries, while DBMS may or may not be optimized for these aspects.

- Overall, RDBMS is a specialized type of DBMS that focuses on relational data and enforces strict integrity constraints, using SQL as the query language.

-----

• What is MF Cod Rule of RDBMS Systems?

### - Rule zero

This rule states that for a system to qualify as an **RDBMS**, it must be able to manage database entirely through the relational capabilities.

### - Rule 1: Information rule

All information(including metadata) is to be represented as stored data in cells of tables. The rows and columns have to be strictly unordered.

### - Rule 2: Guaranted Access

Each unique piece of data(atomic value) should be accesible by : **Table Name + Primary Key(Row) + Attribute(column)**.

**NOTE:** Ability to directly access via POINTER is a violation of this rule.

### - Rule 3: Systematic treatment of NULL

Null has several meanings, it can mean missing data, not applicable or no value. It should be handled consistently. Also, Primary key must not be null, ever. Expression on NULL must give null.

### - Rule 4: Active Online Catalog

Database dictionary(catalog) is the structure description of the complete **Database** and it must be stored online. The Catalog must be governed by same rules as rest of the database. The same query language should be used on catalog as used to query database.

### -Rule 5: Powerful and Well-Structured Language

One well structured language must be there to provide all manners of access to the data stored in the database. Example: **SQL**, etc. If the database allows access to the data without the use of this language, then that is a violation.

### -Rule 6: View Updation Rule

All the view that are theoretically updatable should be updatable by the system as well.

### -Rule 7: Relational Level Operation

There must be Insert, Delete, Update operations at each level of relations. Set operation like Union, Intersection and minus should also be supported.

### -Rule 8: Physical Data Independence

The physical storage of data should not matter to the system. If say, some file supporting table is renamed or moved from one disk to another, it should not effect the application.

### -Rule 9: Logical Data Independence

If there is change in the logical structure(table structures) of the database the user view of data should not change. Say, if a table is split into two tables, a new view should give result as the join of the two tables. This rule is most difficult to satisfy.

-Rule 10: Integrity Independence

The database should be able to enforce its own integrity rather than using other programs. Key and Check constraints, trigger etc, should be stored in Data Dictionary. This also make **RDBMS** independent of front-end.

### -Rule 11: Distribution Independence

A database should work properly regardless of its distribution across a network. Even if a database is geographically distributed, with data stored in pieces, the end user should get an impression that it is stored at the same place. This lays the foundation of **distributed database**.

### -Rule 12: Nonsubversion Rule

If low level access is allowed to a system it should not be able to subvert or bypass integrity rules to change the data. This can be achieved by some sort of looking or encryption.

-----

• What do you understand By Data Redundancy?

- Data redundancy refers to the unnecessary repetition or duplication of data within a database. It occurs when the same piece of data is stored multiple times in different parts of a database or across multiple databases. Redundancy can occur at the attribute (column) level or the record (row) level.

- Data redundancy can have several negative consequences:

1. Wasted Storage: Storing duplicate data consumes additional storage space, leading to inefficiency and increased storage costs.

2. Data Inconsistency: Redundant data creates the risk of inconsistencies. If the same data is updated in one place but not in others, inconsistencies can arise, leading to confusion and incorrect results.

3. Update Anomalies: Redundant data can cause update anomalies, meaning that changes to data must be made in multiple places, making it more likely for inconsistencies or errors to occur during updates.

4. Decreased Performance: Redundancy can impact database performance. It requires more time and resources to update or retrieve redundant data, slowing down operations.

5. Increased Complexity: Redundant data complicates the database structure and makes it more challenging to maintain and manage. It can also make it harder to understand and analyze the data effectively.

- To mitigate data redundancy, normalization techniques are applied in database design. Normalization involves organizing data into well-structured tables, eliminating redundant data, and establishing appropriate relationships between tables. By minimizing redundancy, data integrity, storage efficiency, and overall database performance can be improved.

-----

• What is DDL Interpreter?

- A DDL (Data Definition Language) interpreter is a component or module within a database management system (DBMS) that is responsible for processing and executing DDL statements. DDL is a subset of SQL (Structured Query Language) used for defining and managing the structure of a database.

-The DDL interpreter receives DDL statements issued by users or database administrators and performs the following tasks:

1. Syntax Validation: The interpreter checks the syntax of the DDL statements to ensure they adhere to the rules and grammar of the database's DDL syntax.

2. Semantic Analysis: The interpreter performs semantic analysis to verify the correctness and validity of the DDL statements. This includes checking for the existence of tables, columns, constraints, and other database objects referenced in the statements.

3. Metadata Management: The interpreter updates the metadata of the database system based on the executed DDL statements. Metadata includes information about the database schema, tables, columns, constraints, indexes, and other database objects.

4. Schema Modification: The interpreter applies the necessary modifications to the database schema as specified by the DDL statements. This may involve creating or altering tables, adding or modifying columns, defining constraints, creating indexes, and so on.

5. Dependency Handling: The DDL interpreter handles dependencies between database objects. For example, when creating or altering a table, it ensures that referenced tables or columns exist.

6. Access Control: The DDL interpreter enforces access control mechanisms to ensure that only authorized users can execute DDL statements and modify the database schema.

- Overall, the DDL interpreter plays a crucial role in processing and executing DDL statements, enabling users and administrators to define and modify the structure of a database. It ensures the integrity and consistency of the database schema and manages the underlying metadata associated with the database objects.

-----

• What is DML Compiler in SQL?

- In SQL (Structured Query Language), a DML (Data Manipulation Language) compiler is a component or module within a database management system (DBMS) responsible for processing and executing DML statements. DML is a subset of SQL used for retrieving, inserting, modifying, and deleting data within a database.

The DML compiler performs the following tasks:

1. Syntax Validation: The DML compiler checks the syntax of the DML statements to ensure they conform to the rules and grammar of the database's DML syntax.

2. Semantic Analysis: The compiler performs semantic analysis to validate the correctness and validity of the DML statements. This includes checking the existence of tables, columns, and other database objects referenced in the statements.

3. Query Optimization: The DML compiler optimizes the execution of DML statements to achieve the best possible performance. It analyzes the query and selects an efficient execution plan, considering factors such as indexing, join algorithms, and access paths.

4. Access Control: The DML compiler enforces access control mechanisms to ensure that users have the necessary permissions to execute DML statements and manipulate data.

5. Execution Plan Generation: Based on the analyzed DML statement, the compiler generates an execution plan that outlines how the DBMS will process the statement. The plan includes the steps and operations required to retrieve, modify, or delete data.

6. Data Manipulation: The DML compiler executes the generated execution plan to manipulate data. It performs operations such as reading data from tables, filtering data based on conditions, updating or inserting data, and deleting data as per the DML statement.

Overall, the DML compiler is responsible for processing and executing DML statements in SQL. It ensures the syntactic and semantic correctness of the statements, optimizes their execution, enforces access control, and performs the necessary data manipulation operations specified by the DML statements.

-----

• 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.

The following constraints are commonly used in SQL:

* [NOT NULL](https://www.w3schools.com/sql/sql_notnull.asp) - Ensures that a column cannot have a NULL value
* [UNIQUE](https://www.w3schools.com/sql/sql_unique.asp) - Ensures that all values in a column are different
* [PRIMARY KEY](https://www.w3schools.com/sql/sql_primarykey.asp) - A combination of a NOT NULL and UNIQUE. Uniquely identifies each row in a table
* [FOREIGN KEY](https://www.w3schools.com/sql/sql_foreignkey.asp) - Prevents actions that would destroy links between tables
* [CHECK](https://www.w3schools.com/sql/sql_check.asp) - Ensures that the values in a column satisfies a specific condition
* [DEFAULT](https://www.w3schools.com/sql/sql_default.asp) - Sets a default value for a column if no value is specified
* [CREATE INDEX](https://www.w3schools.com/sql/sql_create_index.asp) - Used to create and retrieve data from the database very quickly

-----

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

- A save point is a logical rollback point within a transaction. When you set a save point, whenever an error occurs past a save point, you can undo the events you have done up to the save point using the rollback.

- MySQL InnoDB provides support for the statements SAVEPOINT, ROLLBACK TO SAVEPOINT, RELEASE SAVEPOINT.

- The SAVEPOINT statement is used to set a save point for the transaction with the specified name. If a save point with the given name already exists the old one will be deleted.

Query:

SAVEPOINT my\_savepoint;

ROLLBACK TO SAVEPOINT my\_savepoint;

-----

• 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.

- create trigger stud\_marks

before INSERT

on

Student

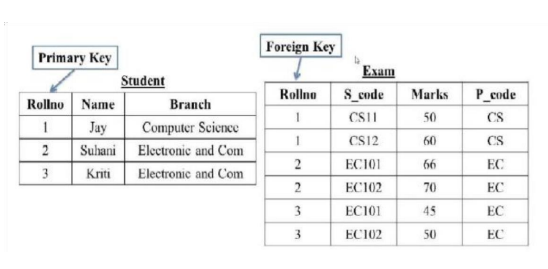
for each row

set Student.total = Student.subj1 + Student.subj2 + Student.subj3, Student.per = Student.total \* 60 / 100;

-----

**Task**

1. Create Table Name : Student and Exam



QUERY:

For student:

create table student(Rollno int(11) PRIMARY key AUTO\_INCREMENT,Name varchar(255),Branch varchar(255));

insert into student(Name,Branch)

Values(‘Jay’,’Computer Science’),

(‘Suhani’,’Electronic and Com’),

(‘Kriti’,’Electronic and Com’);

For Exam:

create table exam(Rollno int(11),S\_code varchar(255),Marks int(5),P\_code varchar(255), foreign key(Rollno) REFERENCES student(Rollno) on DELETE CASCADE on UPDATE CASCADE);

insert into exam(Rollno,S\_code,Marks,P\_code)

Values(1,’CS11’,50,‘CS’),

(1,’CS12’,60,‘CS’),

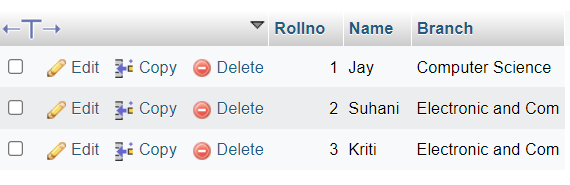
(2,’EC101’,66,‘EC’),

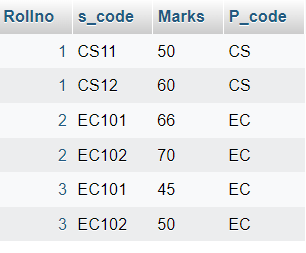
(2,’EC102’,70,‘EC’),

(3,’EC101’,45,‘EC’),

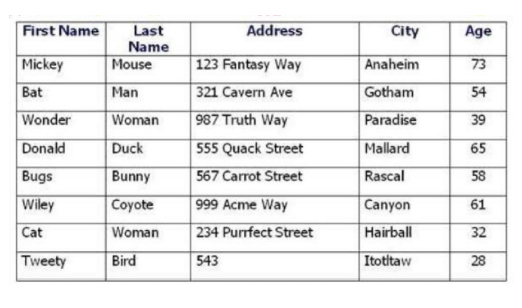
(3,’EC102’,50,‘EC’);

OUTPUT:





2. Create table given below



QUERY:

create table cartoon(First\_Name varchar(255),Last\_Name varchar(255),Address varchar(255),City varchar(255),Age int(11));

insert into cartoon(First\_Name,Last\_Name,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),

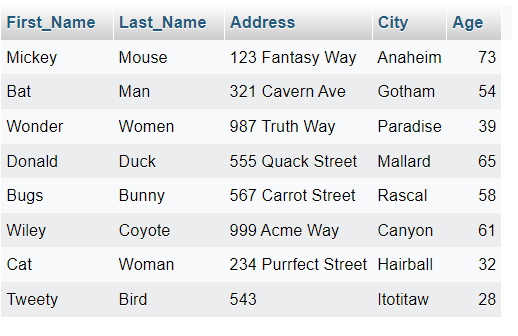
(‘Bugs’,’Bunny’,’567 Carrot Street’,‘Rascal’,58),

(‘Wiley’, ‘Coyote’, ‘999 Acme Way’,‘Canyon’,61)

(‘Cat’, ‘Woman’, ‘234 Purrfect Street’,‘ Hairball’,32)

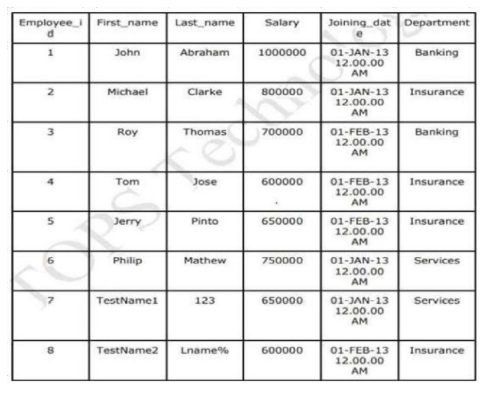
(‘Tweety’,’Bird’,‘543’,‘Itotitaw’,28);

OUTPUT:



3. Create table given below: Employee and Incentive

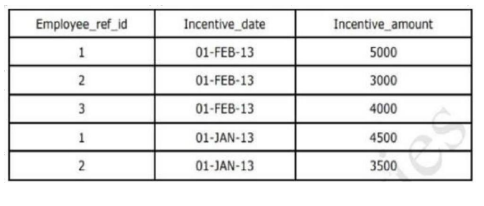
Table Name: Employee



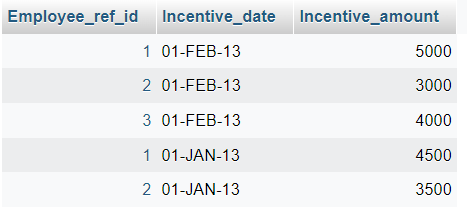
OUTPUT:



Table Name: Incentive



OUTPUT:

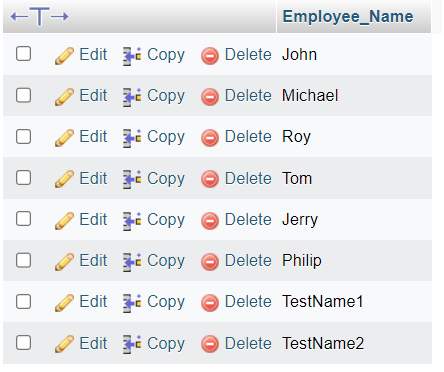


1. Get First\_Name from employee table using Tom name “Employee Name”.

QUERY:

SELECT First\_name as Employee\_Name from employee;

OUTPUT:

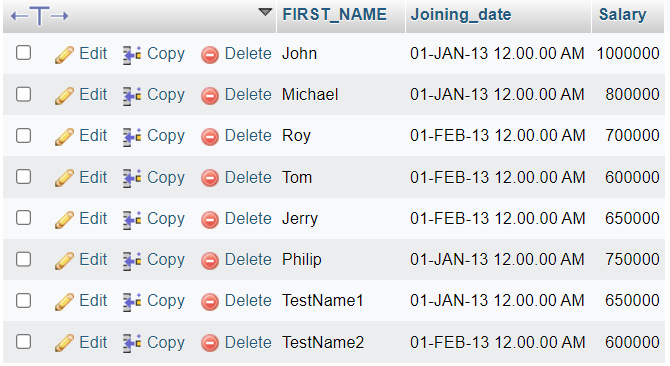


1. Get FIRST\_NAME, Joining Date, and Salary from employee table.

QUERY:

SELECT FIRST\_NAME, Joining Date, Salary from employee;

OUTPUT:

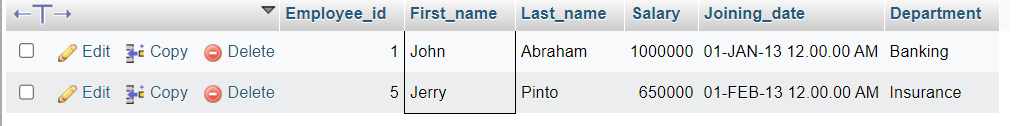


1. Get all employee details from the employee table order by First\_Name Ascending and Salary descending?
2. Get employee details from employee table whose first name contains ‘J’.

QUERY:

[SELECT](http://localhost/phpmyadmin/url.php?url=https://dev.mysql.com/doc/refman/8.0/en/select.html) \* FROM `employee` WHERE First\_name [LIKE](http://localhost/phpmyadmin/url.php?url=https://dev.mysql.com/doc/refman/8.0/en/string-comparison-functions.html%23operator_like) '%J%'

OUTPUT:

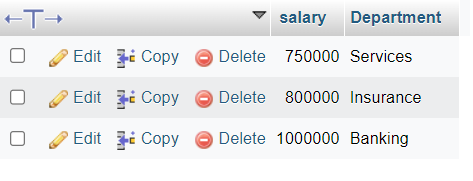


1. Get department wise maximum salary from employee table order by salary ascending?

QUERY:

[select](http://localhost/phpmyadmin/url.php?url=https://dev.mysql.com/doc/refman/8.0/en/select.html) [max](http://localhost/phpmyadmin/url.php?url=https://dev.mysql.com/doc/refman/8.0/en/aggregate-functions.html%23function_max)(Salary) as salary,Department from employee GROUP by Department ORDER BY [MAX](http://localhost/phpmyadmin/url.php?url=https://dev.mysql.com/doc/refman/8.0/en/aggregate-functions.html%23function_max)(Salary);

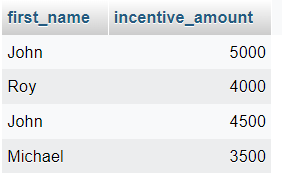
OUTPUT:



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

QUERY: Select employee.first\_name, incentive.incentive\_amount from employee INNER JOIN incentive where employee.Employee\_id=incentive.Employee\_ref\_id and incentive.Incentive\_amount>3000;

OUTPUT:

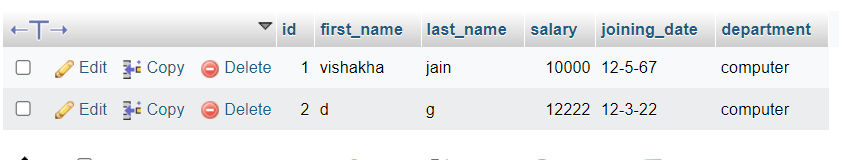


1. Create After Insert trigger on Employee table which insert records in view table

QUERY: [CREATE](http://localhost/phpmyadmin/url.php?url=https://dev.mysql.com/doc/refman/8.0/en/create-trigger.html) [TRIGGER](http://localhost/phpmyadmin/url.php?url=https://dev.mysql.com/doc/refman/8.0/en/create-trigger.html) `afterinsert` AFTER [INSERT](http://localhost/phpmyadmin/url.php?url=https://dev.mysql.com/doc/refman/8.0/en/insert.html) ON `employee` FOR EACH ROW [INSERT](http://localhost/phpmyadmin/url.php?url=https://dev.mysql.com/doc/refman/8.0/en/insert.html) INTO VIEW(id,first\_name,last\_name,salary,joining\_date,department) [values](http://localhost/phpmyadmin/url.php?url=https://dev.mysql.com/doc/refman/8.0/en/miscellaneous-functions.html%23function_values) (new.Employee\_id,new.First\_name,new.Last\_name,new.Salary,new.Joining\_date,new.Department)

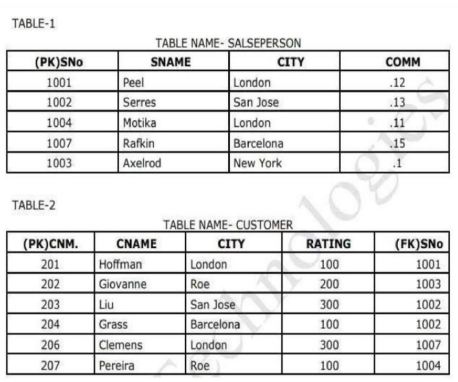
OUTPUT: Employee table:

View table:

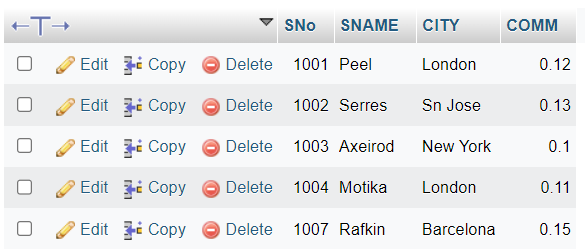


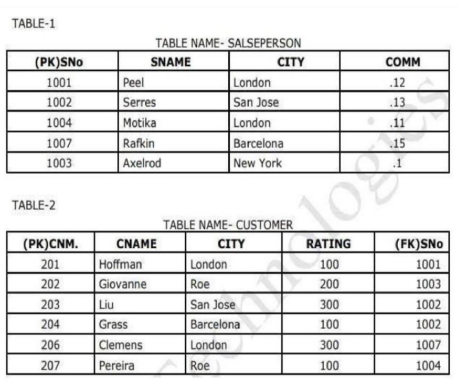
-----

4. Create table given below: Salesperson and Customer

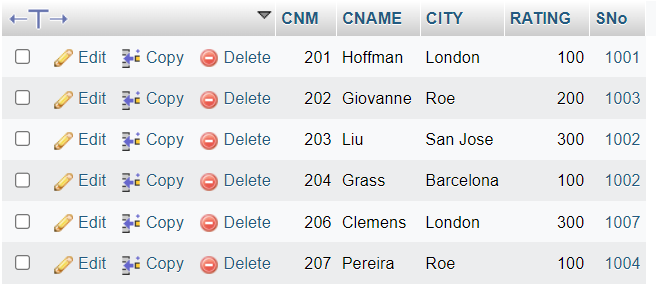


OUTPUT:





OUTPUT:

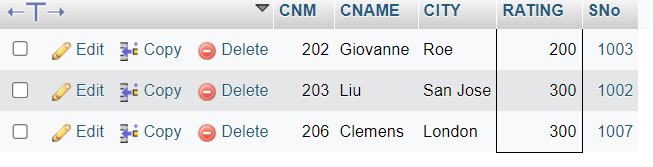


Retrieve the below data from above table

1. All orders for more than $100.

QUERY: SELECT \* FROM customer where RATING>100;

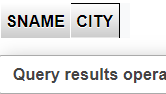
OUTPUT:



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

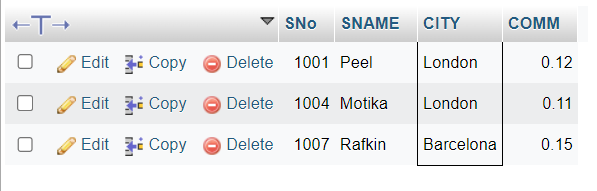
QUERY: SELECT SNAME,CITY from salesperson WHERE CITY='London' AND COMM>0.12

OUTPUT:



1. All salespeople either in Barcelona or in London

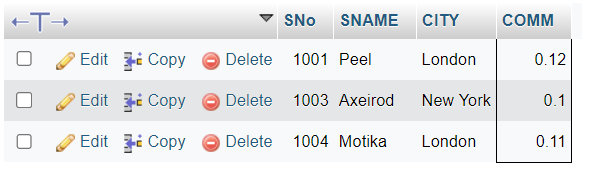
QUERY: SELECT \* from salesperson where CITY='barcelona' OR CITY='london';

OUTPUT: 

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

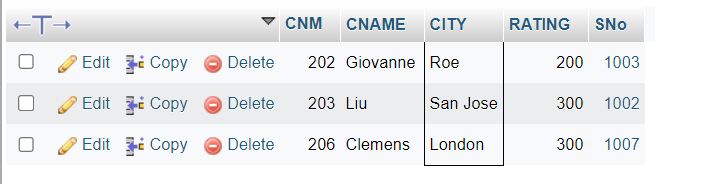
QUERY: SELECT \* from salesperson where COMM>0.10 and COMM<0.12;

OUTPUT:



1. All customers excluding those with rating <= 100 unless they are located in Rom

QUERY:SELECT \* FROM CUSTOMER WHERE RATING>100 AND CITY !=’ROM’;

OUTPUT: 

-----------