Database

**Que-1: What do you understand By Database?**

**Ans :** A database is a systematic or organized collection of related information that is stored in such a way that it can be easily accessed, retrieved, managed, and updated. It is where all data is stored, very much like a library that houses a wide range of books from different genres. Think of data as books.

In a database, you can organize the data in rows and columns in the form of a table. Indexing the data makes it easy to find and retrieve it again as and when required. Many websites on the World Wide Web are managed with the help of databases. To create a database so that the data is accessible to users through only one set of software programs, database handlers are used.

MySQL, SQL Server, [MongoDB](https://intellipaat.com/blog/what-is-mongodb/), Oracle Database,  [PostgreSQL](https://intellipaat.com/blog/what-is-postgresql/), Informix, Sybase, and others are all different types of databases commonly used today. These modern databases are managed by a Database Management System (DBMS). To interact with and manipulate the data stored in these databases, a widely used language called [Structured Query Language](https://intellipaat.com/blog/tutorial/sql-tutorial/introduction-to-sql/) (SQL) is utilized.

**Que-2: What is Normalization?**

**Ans : Normalization**is a database design technique that reduces data redundancy and eliminates undesirable characteristics like Insertion, Update and Deletion Anomalies. Normalization rules divides larger tables into smaller tables and links them using relationships. The purpose of Normalisation in SQL is to eliminate redundant (repetitive) data and ensure data is stored logically.

The inventor of the [relational model](https://www.guru99.com/relational-data-model-dbms.html) Edgar Codd proposed the theory of normalization of data with the introduction of the First Normal Form, and he continued to extend theory with Second and Third Normal Form. Later he joined Raymond F. Boyce to develop the theory of Boyce-Codd Normal Form.

Here is a list of Normal Forms in SQL:

* 1NF (First Normal Form)
* 2NF (Second Normal Form)
* 3NF (Third Normal Form)
* BCNF (Boyce-Codd Normal Form)
* 4NF (Fourth Normal Form)
* 5NF (Fifth Normal Form)
* 6NF (Sixth Normal Form)

**Que-3: What is Difference between DBMS and RDBMS?**

**Ans : DBMS**-DBMS stands for Database Management System. It is software that enables the creation, organization, retrieval, manipulation, and management of databases. A DBMS provides an interface for users and applications to interact with the database, allowing them to store, retrieve, update, and delete data.

**RDBMS**-RDBMS stands for Relational Database Management System. It is a type of DBMS (Database Management System) software that specifically uses the relational data model to organize and manage data. In an RDBMS, data is structured in the form of tables with rows and columns, and relationships between tables are established using keys.

| **DBMS** | **RDBMS** |
| --- | --- |
| [DBMS](https://www.geeksforgeeks.org/introduction-of-dbms-database-management-system-set-1/) stores data as file. | [RDBMS](https://www.geeksforgeeks.org/rdbms-architecture/) stores data in tabular form. |
| Data elements need to access individually. | Multiple data elements can be accessed at the same time. |
| No relationship between data. | Data is stored in the form of tables which are related to each other. |
| Normalization is not present. | Normalization is present. |
| DBMS does not support distributed database. | RDBMS supports distributed database. |
| It stores data in either a navigational or hierarchical form. | It uses a tabular structure where the headers are the column names, and the rows contain corresponding values. |
| It deals with small quantity of data. | It deals with large amount of data. |
| Data redundancy is common in this model. | Keys and indexes do not allow Data redundancy. |
| It is used for small organization and deal with small data. | It is used to handle large amount of data. |
| Not all Codd rules are satisfied. | All 12 Codd rules are satisfied. |
| Security is less | More security measures provided. |
| It supports single user. | It supports multiple users. |
| Data fetching is slower for the large amount of data. | Data fetching is fast because of relational approach. |
| The data in a DBMS is subject to low security levels with regards to data manipulation. | There exists multiple levels of data security in a RDBMS. |
| Low software and hardware necessities. | Higher software and hardware necessities. |
| Examples:[XML](https://www.geeksforgeeks.org/xml-basics/), Window Registry, Forxpro, dbaseIIIplus etc. | Examples: [MySQL](https://www.geeksforgeeks.org/architecture-of-mysql/), [PostgreSQL](https://www.geeksforgeeks.org/what-is-postgresql-introduction/), SQL Server, Oracle, Microsoft Access etc. |

**Que-4: What is MF Cod Rule of RDBMS Systems?**

**Ans :** Relational database theory was first introduced by Edgar Frank Codd in 1970. Edgar Frank Codd was a British computer scientist who, while working for IBM, invented the relational model for database management the theoretical basis for relational databases. Codd defined 13 rules, often termed “Cod’s 12 rules” because he numbered them from zero through 12, on satisfying a relational model, These rules serve as the framework for what a truly relational database should be.

**These rules are:**

### **Rule 0: The Foundation Rule**

For a system to be qualified as a relational database management system, it must be able to manage databases entirely through its relational capabilities.

### **Rule 1: The Information Rule**

The information in a relational database must be stored in columns or rows of a table, i.e., a cell.

### **Rule 2: The Guaranteed Access Rule**

Each and every datum in a relational database must be logically accessible using the combination of the table name, primary key value, and column name.

A datum is an atomic value, i.e., a piece of information that cannot be broken down further.

### **Rule 3: Systematic Treatment Of NULL Values**

NULL values are fully supported in a relational database and represent missing information or inapplicable information in a systematic way, independent of the data type. NULL values are different from empty strings, blank spaces, and 00.

### **Rule 4: Active/Dynamic Online Catalog Based On The Relational Model**

Database Description (Catalog) of a complete database must be stored online. The rules of the rest of the database must also apply to the Catalog. The query language used to query the database should be used for the catalog also.

### **Rule 5: The Comprehensive Data Sublanguage Rule**

Relational systems can support multiple languages and different modes of using terminals, such as fill-in-the-blanks mode. However, there must be at least one language whose statements are expressible according to a well-defined syntax.

### **Rule 6: The View Updating Rule**

Theoretically, updatable views are also practically updatable by the database system.

### **Rule 7: High-Level Insert, Update & Delete Rule**

The database system must follow high-level relational operations such as insertion, updation, and deletion at each level or row by row. It also supports the union, intersection, and subtraction operations in database systems.

### **Rule 8: Physical Data Independence**

The working of a database system should be independent of the physical storage of its data. If a file is modified (renamed or moved to another location), it should not interfere with the working of the system.

### **Rule 9: Logical Data Independence**

If there is a change in the logical structure (table structure) of the database, the user view of the data must not change.  
Say a table is partitioned into two tables, the new view should give the result as the join of the two tables.

### **Rule 10: Integrity Independence**

[Integrity constraints](https://www.scaler.com/topics/dbms/integrity-constraints-in-dbms/) specific to a particular relational database must be defined in the relational data sub-language and stored in the catalog and not in the application.

### **Rule 11: Distribution Independence**

A database should work properly regardless of its distribution across a network. The end-user should not be able to see that the data is distributed over many locations, they should always get the impression that the data is located at a single site only.

### **Rule 12: The Non-subversion Rule**

If a relational system allows low-level access, that low cannot be used to subvert or bypass the integrity rules to modify the data. This can be achieved with some sort of looking or encryption.

**Que-5: What do you understand By Data Redundancy?**

**Ans :** Data redundancy means the occurrence of duplicate copies of similar data. It is done intentionally to keep the same piece of data at different places, or it occurs accidentally.

In DBMS, when the same data is stored in different tables, it causes data redundancy.

Sometimes, it is done on purpose for recovery or backup of data, faster access of data, or updating data easily. Redundant data costs extra money, demands higher storage capacity, and requires extra effort to keep all the files up to date.

Sometimes, unintentional duplicity of data causes a problem for the database to work properly, or it may become harder for the end user to access data. Redundant data unnecessarily occupy space in the database to save identical copies, which leads to space constraints, which is one of the major problems.

**Que-6: What is DDL Interpreter?**

**Ans :** A DDL (Data Definition Language) interpreter is a component or module within a database management system (DBMS) that processes and executes DDL statements. DDL statements are used to define and manage the structure or schema of a database, including creating, altering, and dropping database objects such as tables, views, indexes, and constraints.

The DDL interpreter is responsible for parsing and interpreting DDL statements issued by users or applications. It validates the syntax and semantics of the statements, checks for permissions and access rights, and performs the necessary operations to implement the requested changes in the database schema.

Here are a few common tasks performed by a DDL interpreter:

**1. Creating database objects:** The DDL interpreter processes the CREATE statements to create tables, views, indexes, constraints, and other database objects. It ensures that the specified objects are created with the correct structure and attributes.

**2. Modifying database objects:** The ALTER statements are used to modify existing database objects. The DDL interpreter interprets these statements to add or remove columns, change data types, rename objects, or make other alterations to the schema.

**3. Dropping database objects:** The DROP statements are used to remove database objects from the schema. The DDL interpreter handles these statements to delete tables, views, indexes, or other objects from the database.

**4. Enforcing data integrity:** DDL statements can define constraints such as primary keys, foreign keys, unique constraints, and check constraints. The DDL interpreter ensures that these constraints are properly defined and enforced to maintain data integrity.

Overall, the DDL interpreter plays a crucial role in managing the structure of a database by interpreting and executing DDL statements, enabling users and applications to define, modify, and remove database objects as needed.

**Que-7: What is DML Compiler in SQL?**

**Ans :** In SQL (Structured Query Language), a DML (Data Manipulation Language) compiler is a component or module within a database management system (DBMS) that processes and executes DML statements. DML statements are used to retrieve, manipulate, and modify data stored in a database.

The DML compiler is responsible for parsing and interpreting DML statements issued by users or applications. It validates the syntax and semantics of the statements, checks for permissions and access rights, and performs the necessary operations to retrieve or modify data in the database.

Here are a few common tasks performed by a DML compiler:

**1. Data retrieval:** The SELECT statement is used to retrieve data from one or more tables in the database. The DML compiler interprets the SELECT statement, processes any specified conditions or filters, and returns the requested data.

**2. Data insertion:** The INSERT statement is used to insert new records into a table. The DML compiler validates the syntax of the INSERT statement, checks if the specified columns and values are valid, and performs the insertion operation.

**3. Data modification:** The UPDATE statement is used to modify existing records in a table. The DML compiler interprets the UPDATE statement, validates the syntax, checks for the specified conditions, and performs the necessary modifications to update the data.

**4. Data deletion:** The DELETE statement is used to delete records from a table. The DML compiler processes the DELETE statement, validates the syntax, checks for conditions, and removes the specified records from the table.

The DML compiler works in conjunction with other components of the DBMS, such as the query optimizer and transaction manager, to execute DML statements efficiently and maintain data integrity. It ensures that the specified DML operations are performed correctly and consistently, allowing users and applications to manipulate and manage data stored in the database.

**Que-8: What is SQL Key Constraints .**

**Ans :** Constraints are the rules that we can apply on the type of data in a table. That is, we can specify the limit on the type of data that can be stored in a particular column in a table using constraints.

The available constraints in SQL are with syntax for student table: 

* **NOT NULL**: This constraint tells that we cannot store a null value in a column. That is, if a column is specified as NOT NULL then we will not be able to store null in this particular column any more.
* **UNIQUE**: This constraint when specified with a column, tells that all the values in the column must be unique. That is, the values in any row of a column must not be repeated.
* **PRIMARY KEY**: A primary key is a field which can uniquely identify each row in a table. And this constraint is used to specify a field in a table as primary key.
* **FOREIGN KEY**: A Foreign key is a field which can uniquely identify each row in a another table. And this constraint is used to specify a field as Foreign key.
* **CHECK**: This constraint helps to validate the values of a column to meet a particular condition. That is, it helps to ensure that the value stored in a column meets a specific condition.
* **DEFAULT**: This constraint specifies a default value for the column when no value is specified by the user.

**Que-9: What is save Point? How to create a save Point write a Query?**

**Ans :** In the context of databases, a savepoint is a named marker within a transaction that allows you to roll back part of a transaction to a specific point in time. It helps you create intermediate checkpoints within a transaction, enabling you to undo changes made after that point while keeping the changes made earlier.

Savepoints are useful when you want to perform a partial rollback in a transaction instead of rolling back the entire transaction.

To create a savepoint and write a query to do so, you need to use a database management system that supports savepoints (e.g., MySQL, Oracle, PostgreSQL). The specific syntax may vary depending on the database system you are using. Here's an example of how to create a savepoint using SQL:

**Syntax:** SAVEPOINT your\_savepoint\_name;

Once you create a savepoint, you can continue executing your transactional queries. If you reach a point where you want to roll back to the savepoint, you can issue a rollback statement.

**Here's an example of rolling back to a savepoint:**

ROLLBACK TO SAVEPOINT your\_savepoint\_name;

This rollback statement will undo all changes made after the specified savepoint. It effectively restores the data to the state it was at the time the savepoint was created.

**Que-10: What is trigger and how to create a Trigger in SQL?**

**Ans :** In SQL, a trigger is a database object that is associated with a table and automatically executes a set of actions or queries in response to specific events or conditions occurring on the table. These events can include insertions, deletions, or updates of data in the table.Triggers are often used to enforce business rules, perform data validation, maintain data integrity, or automate certain tasks in the database.

To create a trigger in SQL, you need to use the `CREATE TRIGGER` statement. The specific syntax may vary slightly depending on the database management system you are using. Here's a general example:**Syntax:**

CREATE TRIGGER trigger\_name

{BEFORE | AFTER} {INSERT | UPDATE | DELETE} ON table\_name

[FOR EACH ROW]

[WHEN condition]

BEGIN

-- SQL statements or actions to be executed

END;

Let's break down the components of this syntax:

- **CREATE TRIGGER**: This is the statement used to create a trigger.

- **trigger\_name**: Provide a unique name for the trigger.

- **BEFORE | AFTER**: Specify whether the trigger should be executed before or after the event (insert, update, delete) occurs.

- **INSERT | UPDATE | DELETE**: Choose the event that will trigger the execution of the trigger.

- **table\_name**: Specify the name of the table associated with the trigger.

- **FOR EACH ROW:** This optional clause indicates that the trigger should be fired for each affected row.

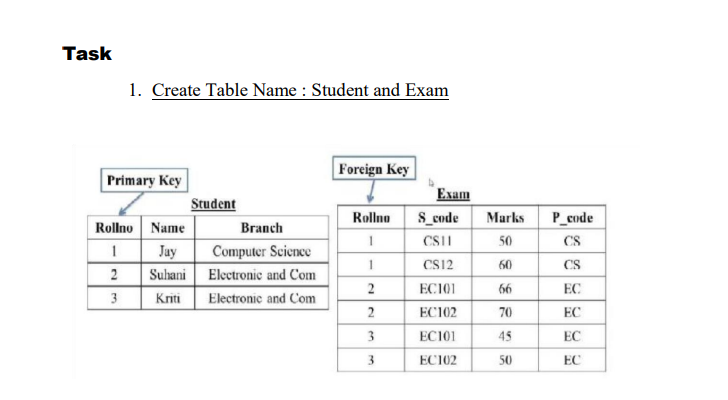
- **WHEN** condition: This optional clause allows you to define additional conditions that must be satisfied for the trigger to execute.

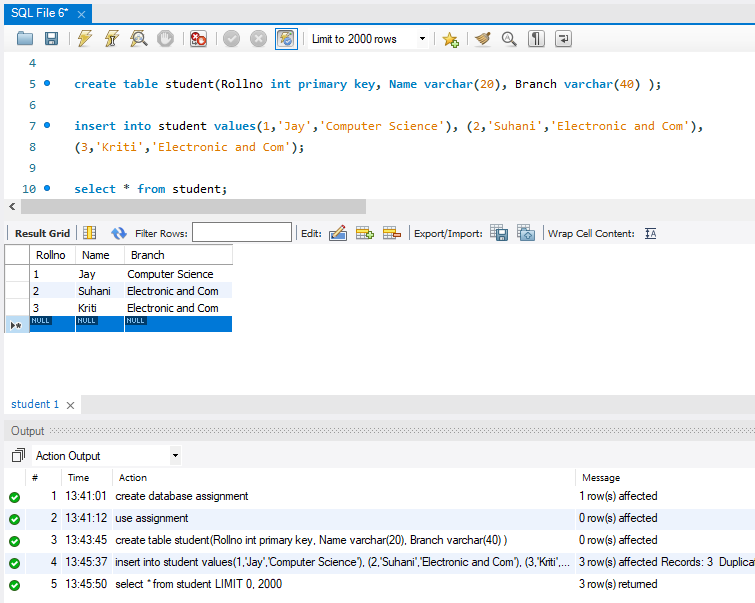
- **BEGIN and END**: These keywords enclose the set of SQL statements or actions that the trigger will execute.

Within the BEGIN and END block, you can include any SQL statements or actions that you want the trigger to perform when the specified event occurs. For example, you can insert, update, or delete data in other tables, raise an error, or modify the data being affected.

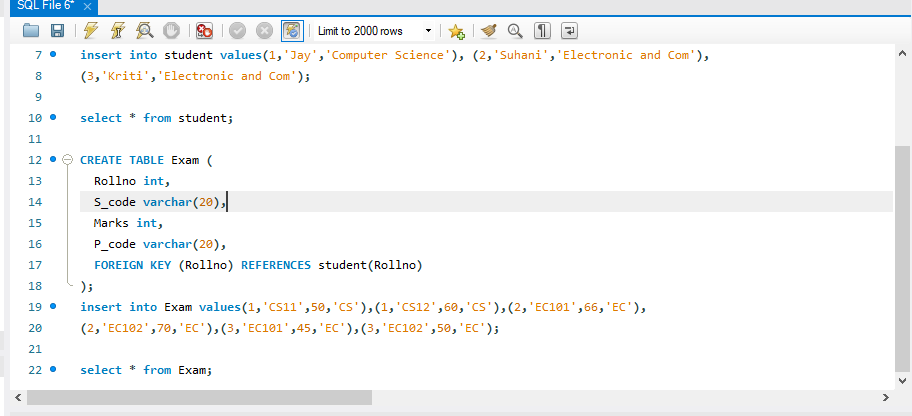
Remember to consult the documentation of your specific database management system for the precise syntax and additional options available for creating triggers, as they may vary between systems.

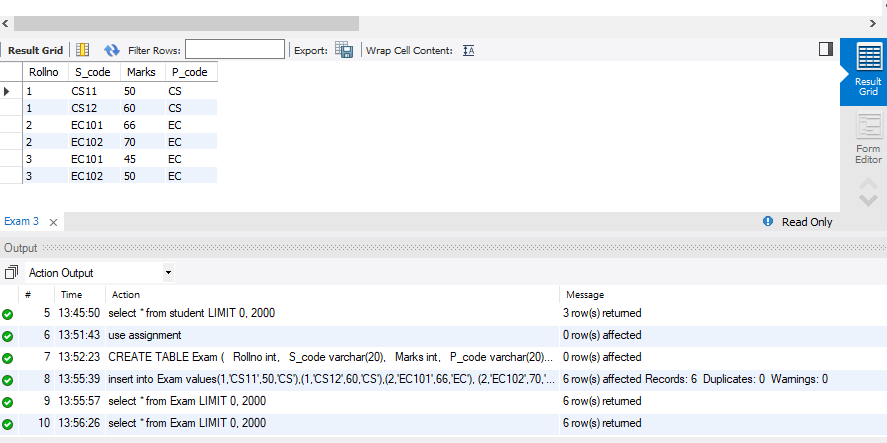
**Task:1**

****

**Ans- Student Table: **

**Exam Table:**

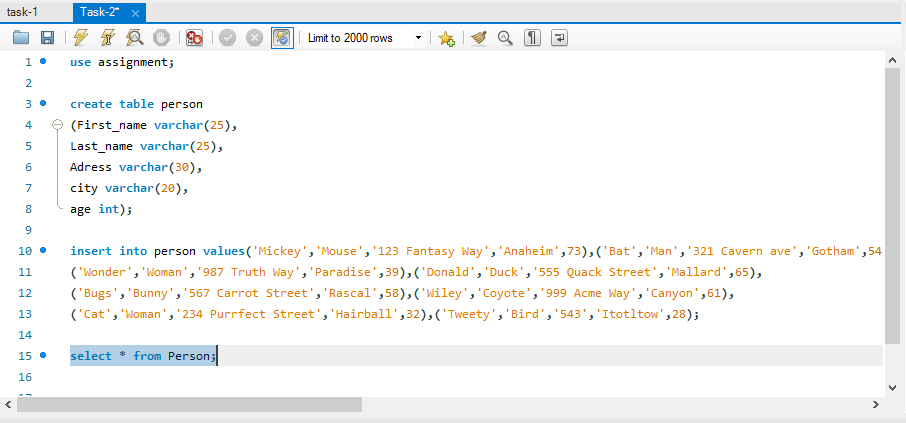
****

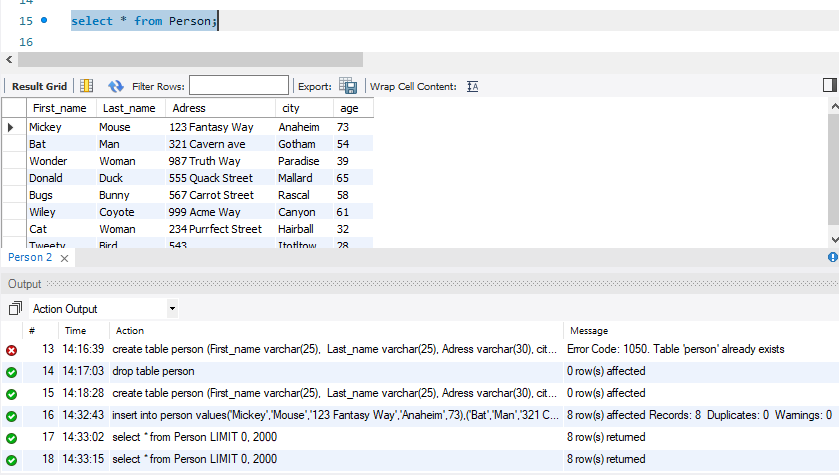
****

**Task:2**

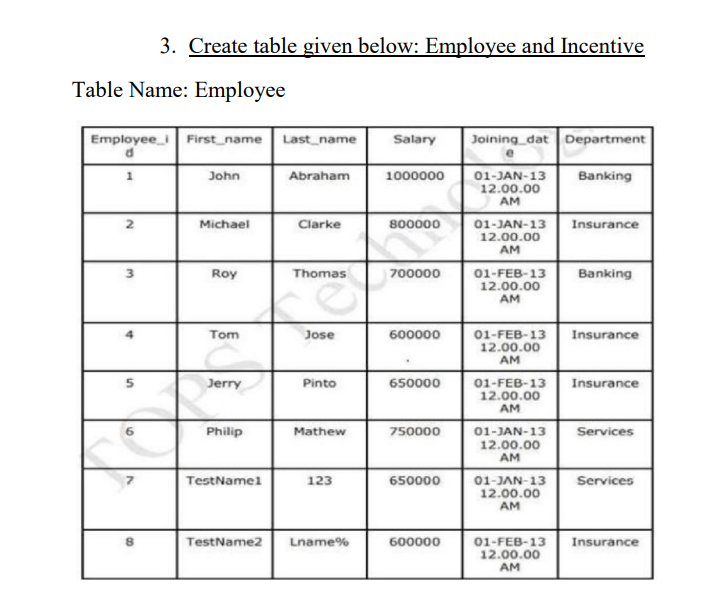
****

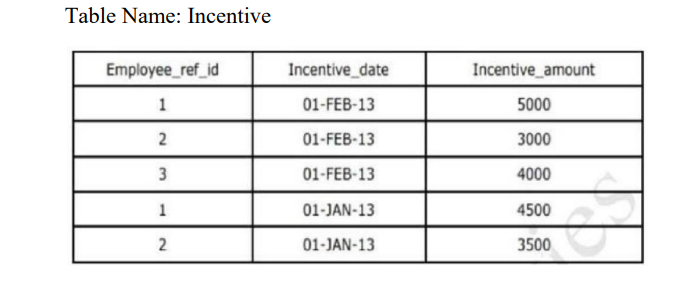
**Ans-**

****

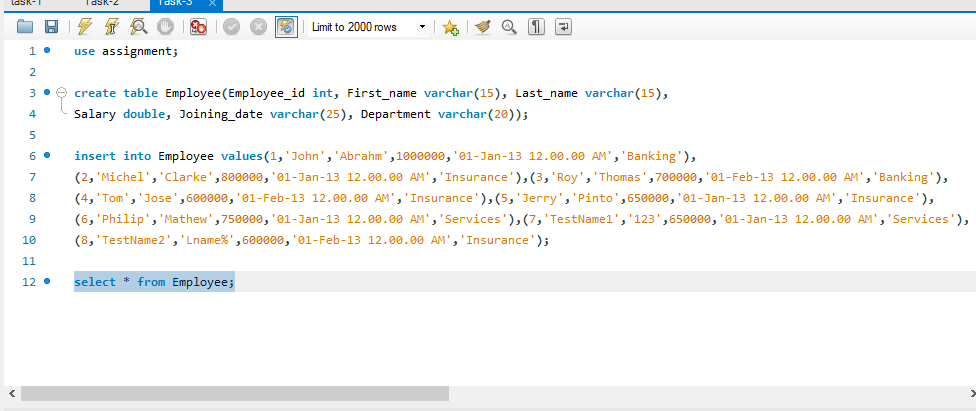
****

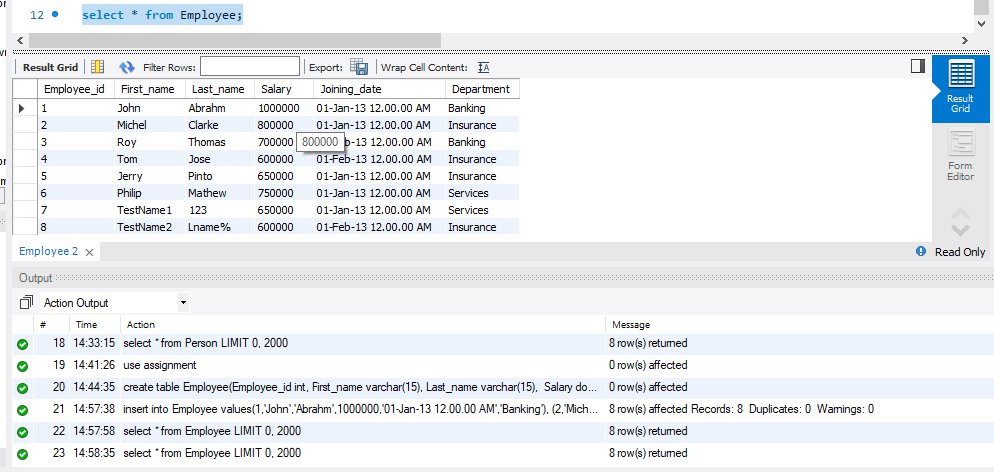
**Task:3**

****

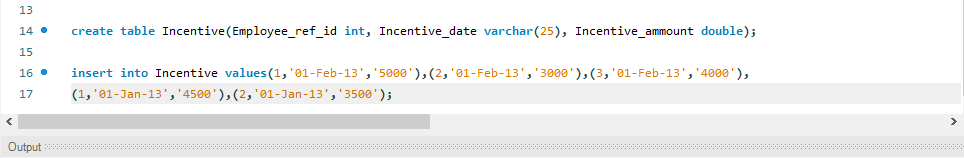
****

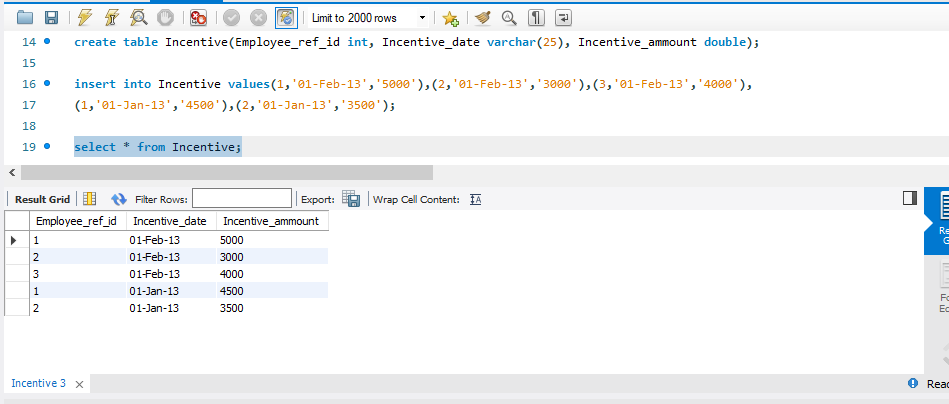
**Ans- Employee Table:**

****

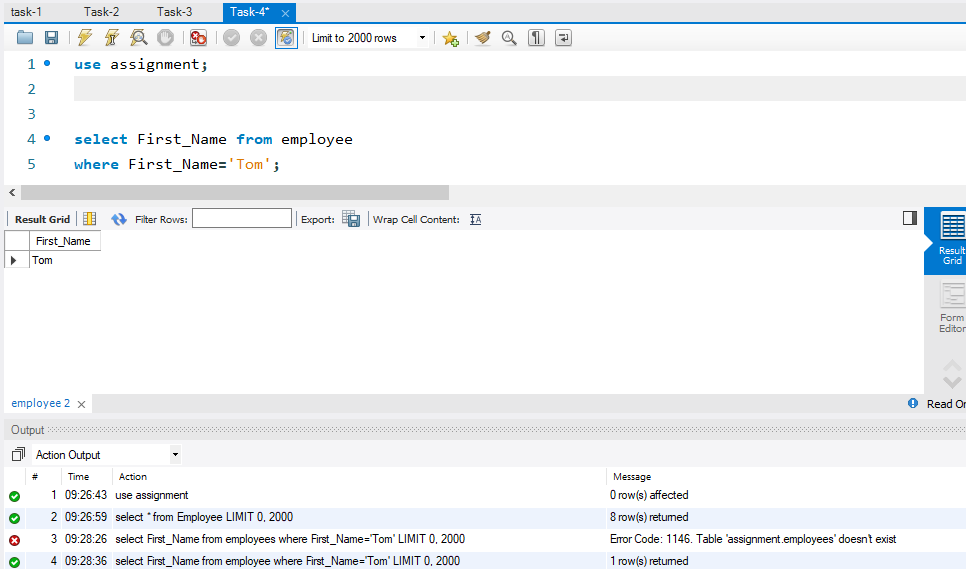
****

**Incentive Table:**

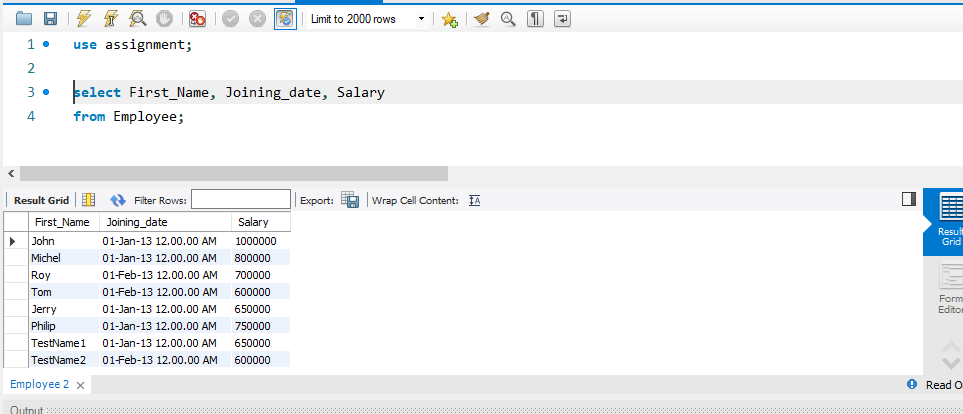
****

****

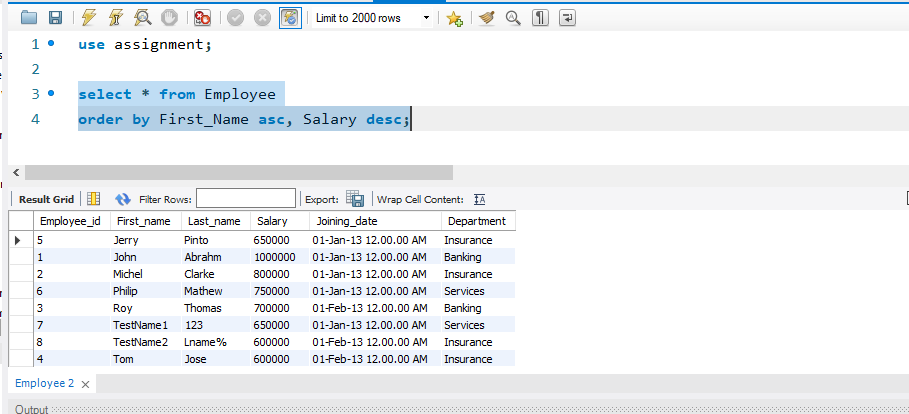
**Task-4:** Get First\_Name from employee table using Tom name “Employee Name”.

****

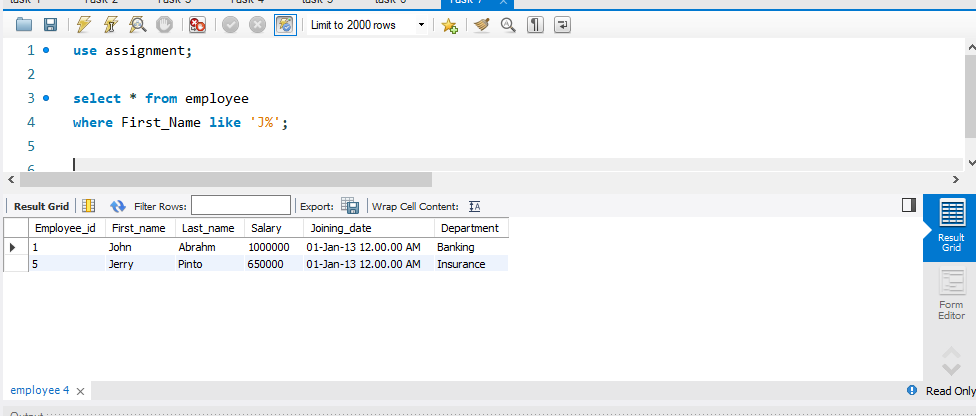
**Task-5:** Get FIRST\_NAME, Joining Date, and Salary from employee table.

****

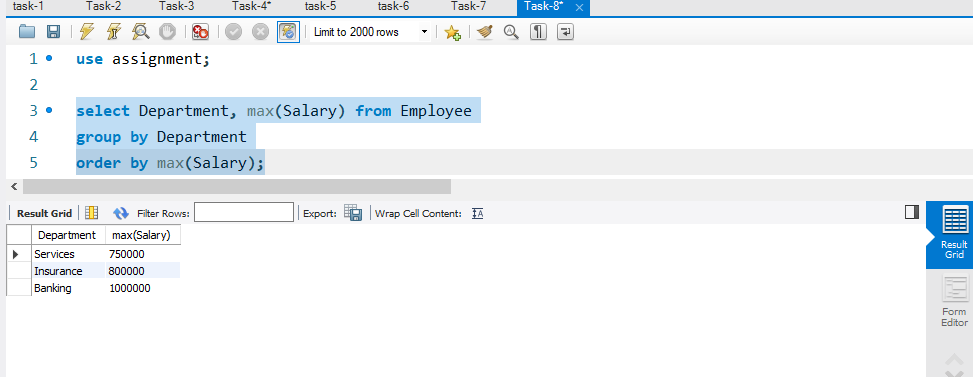
**Task-6:** Get all employee details from the employee table order by First\_Name Ascending and Salary descending?

****

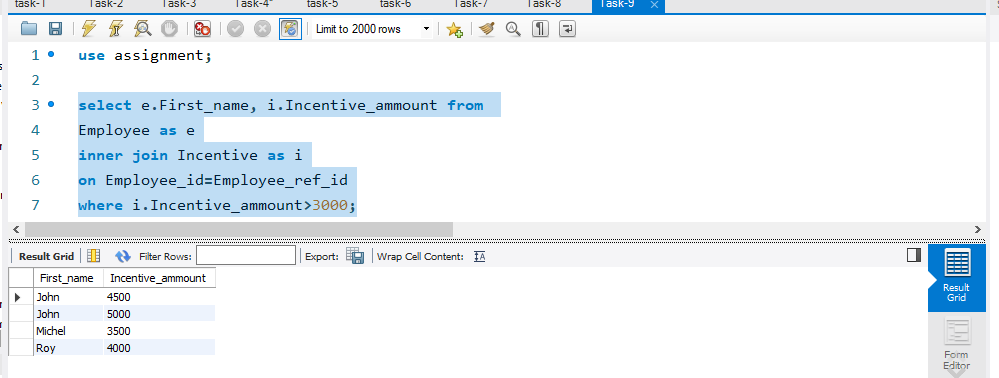
**Task-7:** Get employee details from employee table whose first name contains ‘J’.

****

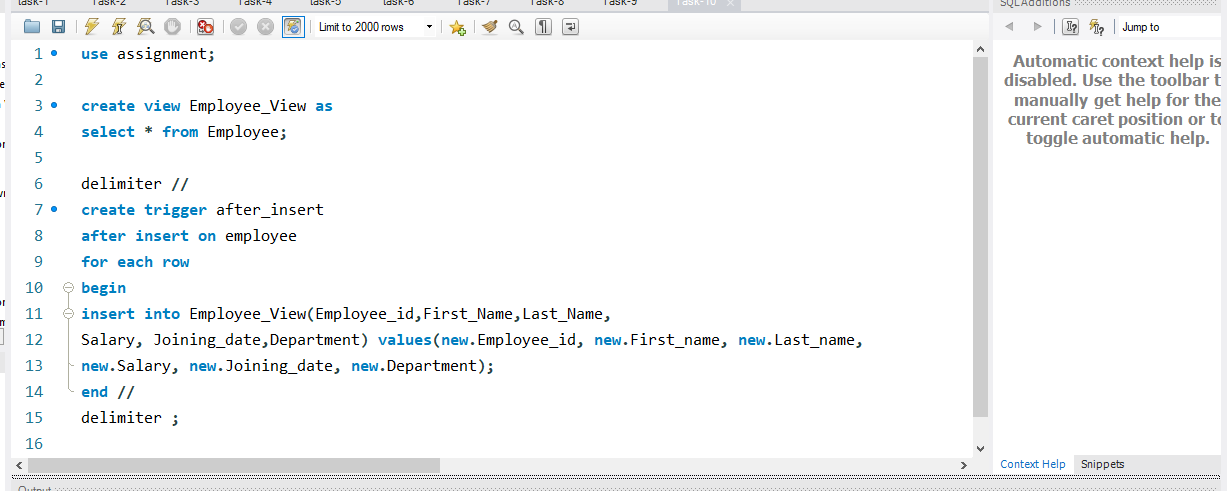
**Task-8:** Get department wise maximum salary from employee table order by salary ascending?



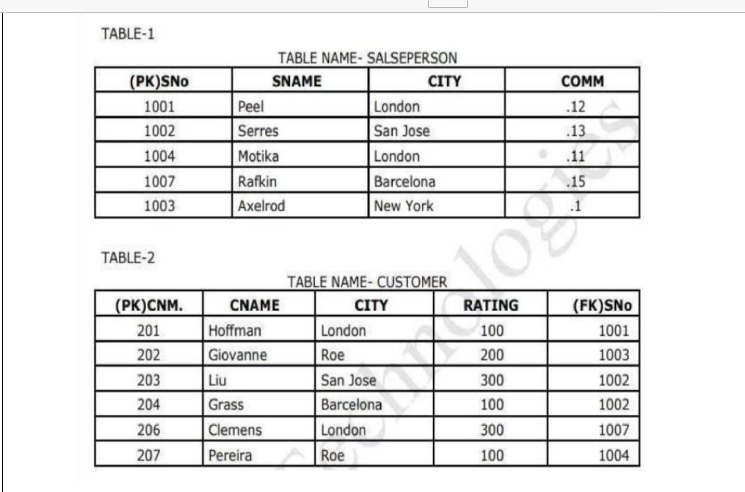
**Task-9:** Select first\_name, incentive amount from employee and incentives table for those employees who have incentives and incentive amount greater than 3000.

****

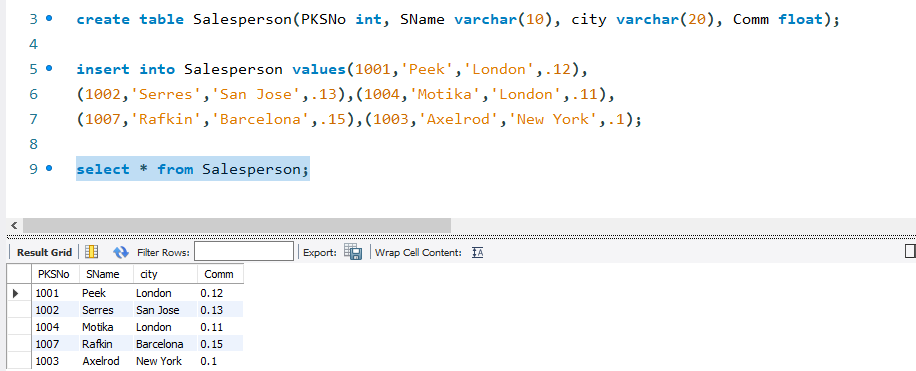
**Task-10:** Create After Insert trigger on Employee table which insert records in view table.



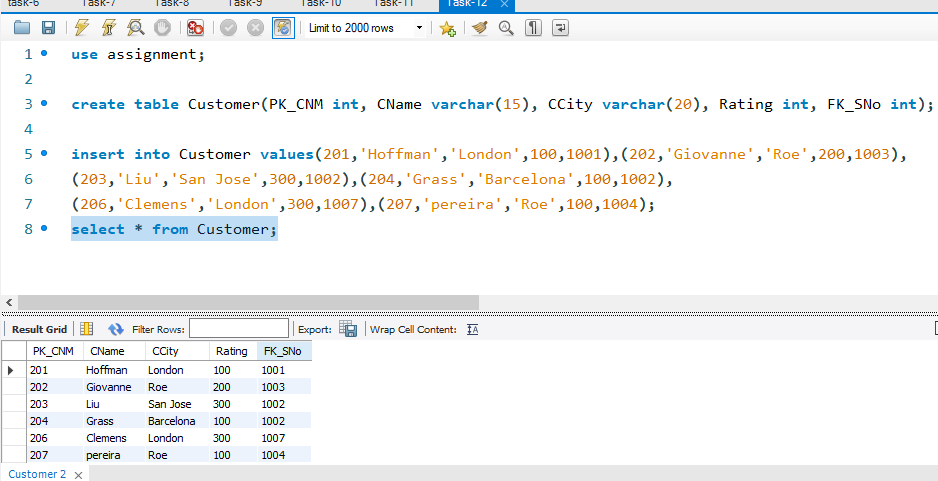
**Task-11:** Create table given below: Salesperson and Customer.

****

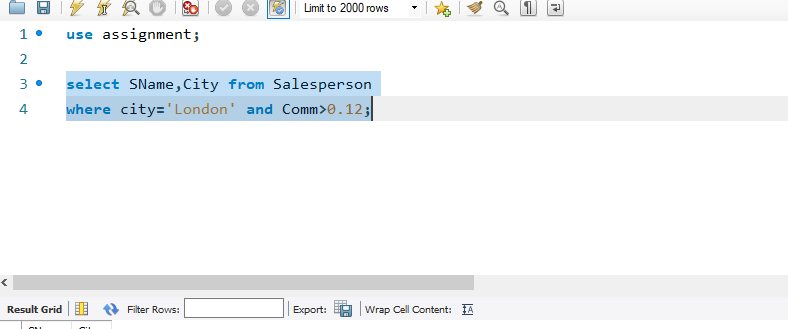
**Ans: Salesperson Table**

****

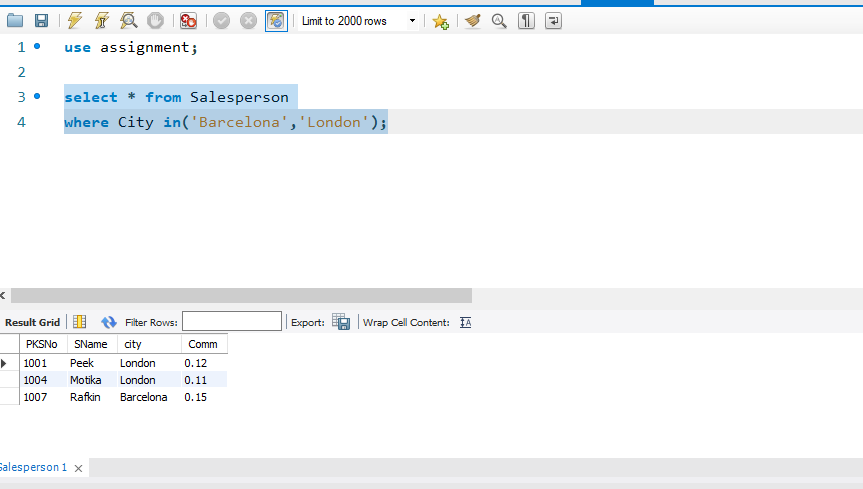
**Customer Table**

****

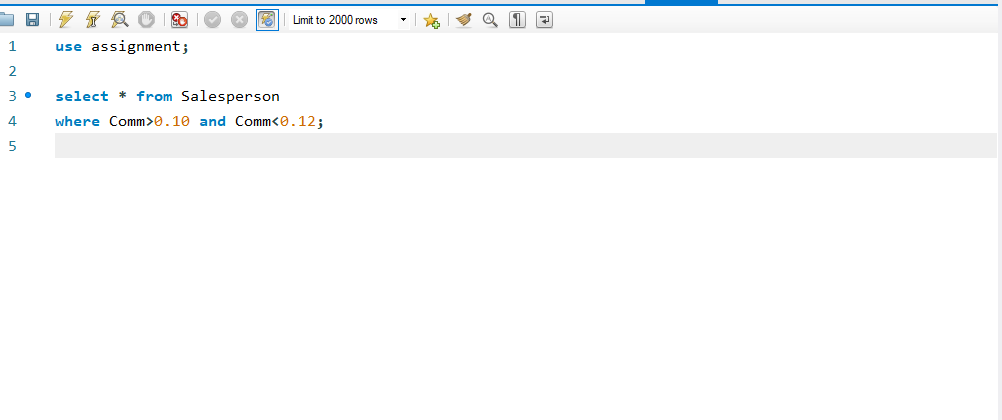
**Task-12:** Names and cities of all salespeople in London with commission above 0.12.

****

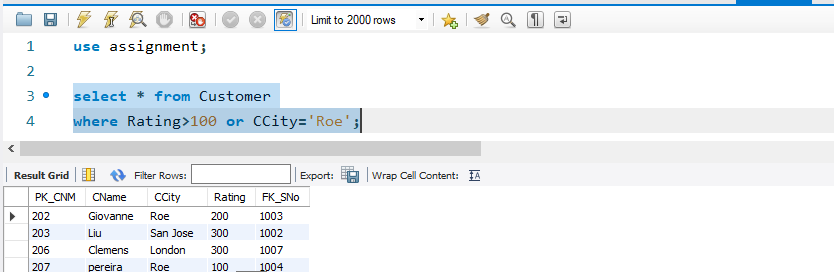
**Task-13:** All salespeople either in Barcelona or in London.

****

**Task-14:** All salespeople with commission between 0.10 and 0.12. (Boundary values should be excluded).

****

**Task-15:** All customers excluding those with rating <= 100 unless they are located in Rome.

****