CREATE TABLE

M.POOJA
513423104036

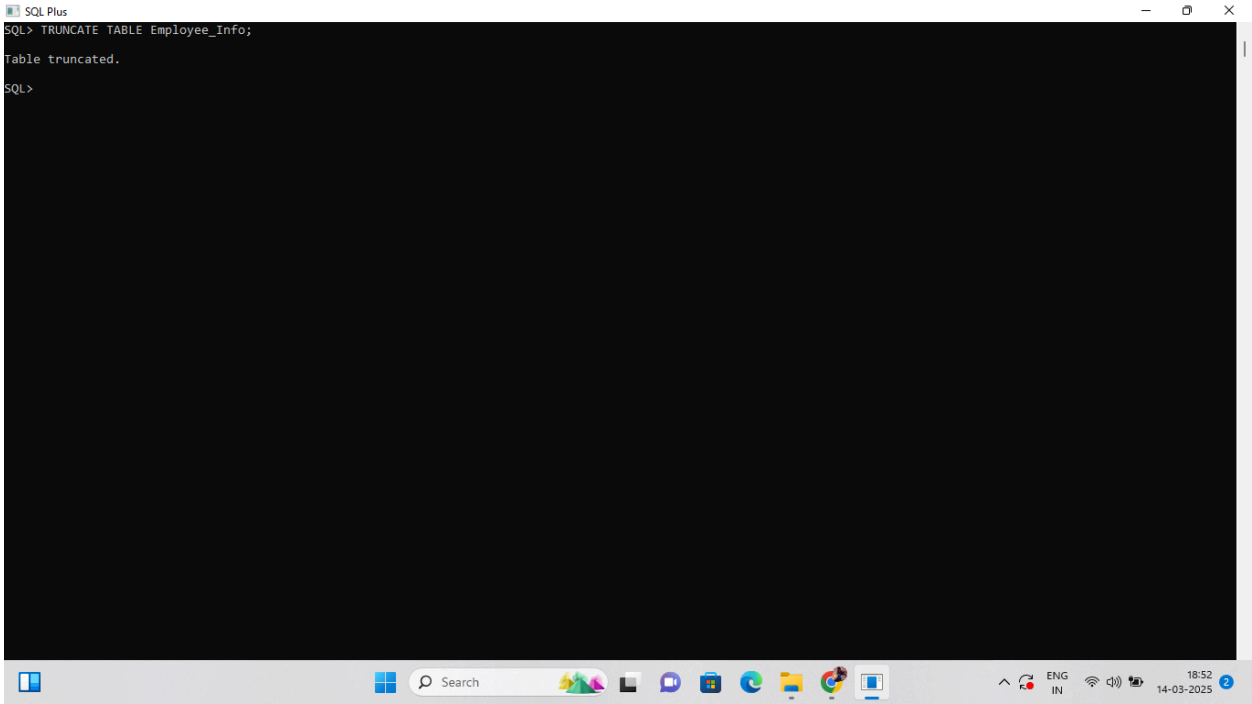
SQL Plus

```
Enter user-name: system
Enter password:
Last Successful login time: Fri Mar 14 2025 18:39:32 +05:30

Connected to:
Oracle Database 21c Express Edition Release 21.0.0.0.0 - Production
Version 21.3.0.0.0

SQL> drop table Employee_Info;

Table dropped.

SQL> SET LINESIZE 200;
SQL> SET PAGESIZE 50;
SQL> CREATE TABLE Employee_Info(EmployeeID INT, EmployeeName VARCHAR(15), EmergencyContactName VARCHAR(20), PhoneNumber VARCHAR(10), Address VARCHAR(20), City VARCHAR(1
5), Country VARCHAR(15));

Table created.

SQL> INSERT INTO Employee_Info VALUES (1, 'John Doe', 'Jane Doe', '1234567890', '123 Main St', 'New York', 'USA');

1 row created.

SQL> SELECT * FROM Employee_Info;

EMPLOYEEID EMPLOYEENAME    EMERGENCYCONTACTNAME PHONENUMBE ADDRESS              CITY            COUNTRY
---------- --------------- -------------------- ---------- -------------------- --------------- ---------------
         1 John Doe        Jane Doe             1234567890 123 Main St          New York        USA

SQL>
```

A **table** in SQL is a structured collection of data, similar to an Excel spreadsheet, where **columns** represent different types of information and **rows** represent individual records.

TRUNCATING TABLE

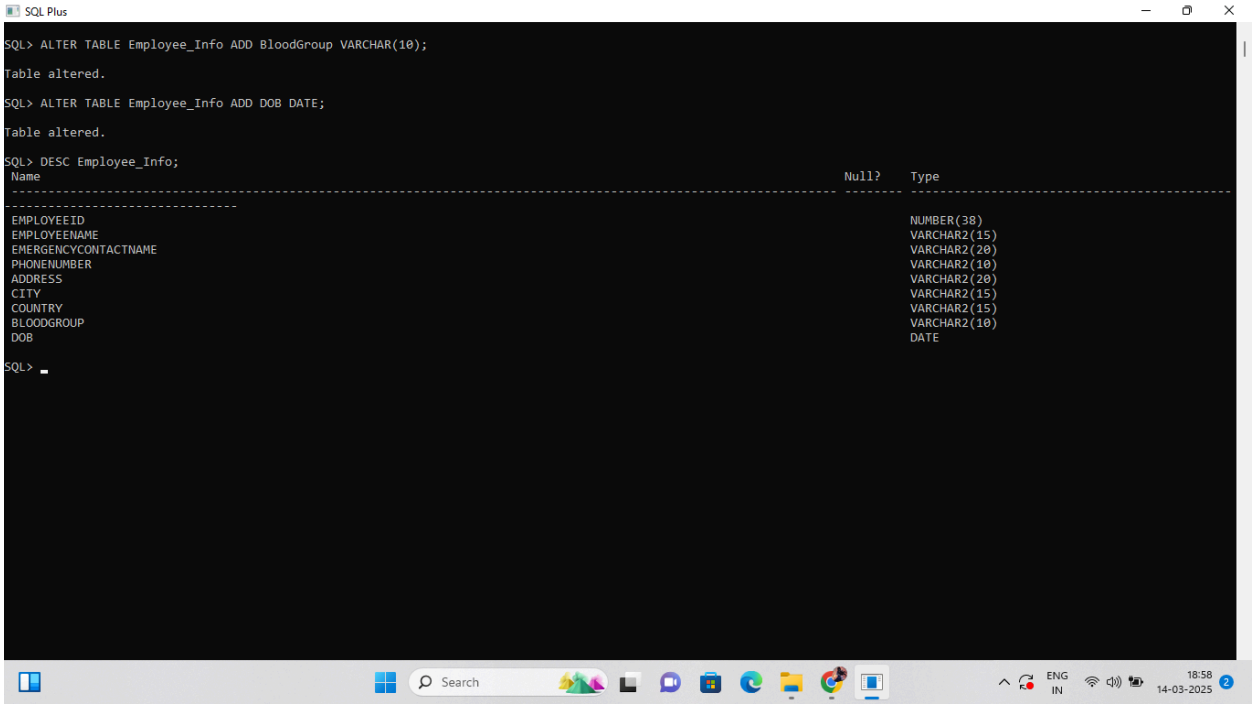
M.POOJA
513423104036



Truncating a table means deleting all the rows (records) from a table, but the table structure remains intact. It is a fast and efficient way to remove all data from a table without logging individual row deletions.\

ALTER TABLE

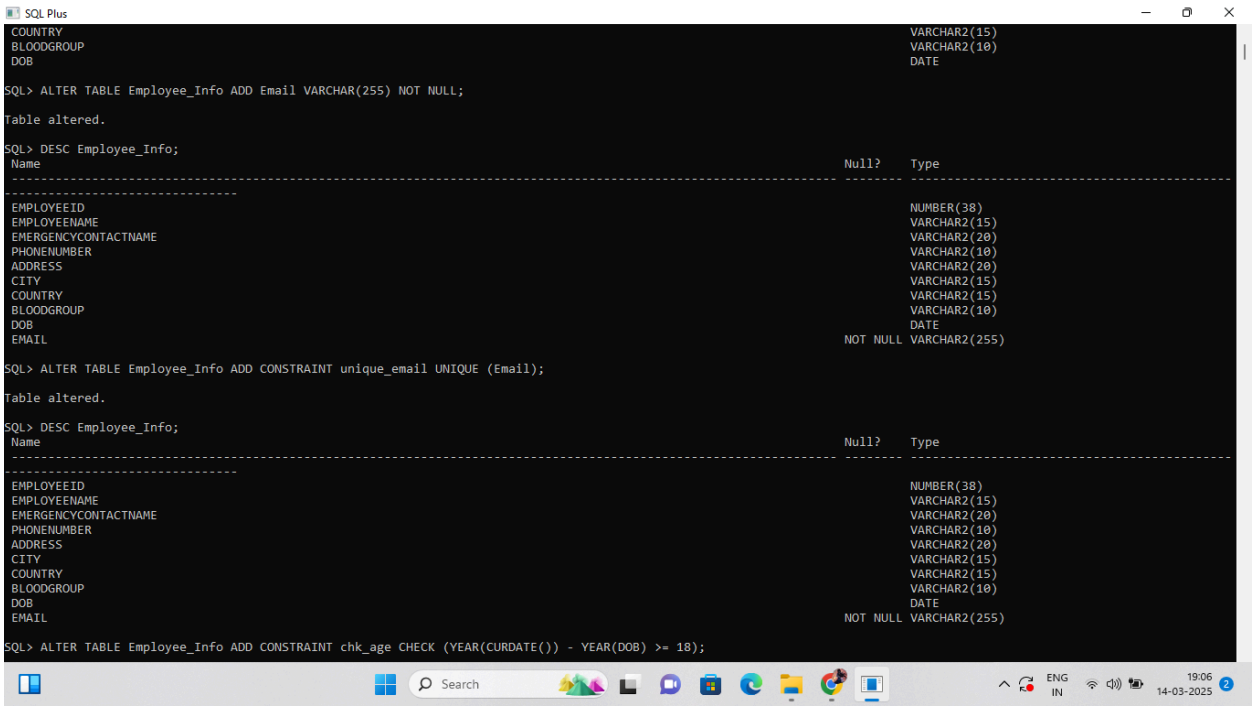
M.POOJA
513423104036



The **ALTER TABLE** statement in SQL is used to modify an existing table's structure without deleting or recreating it. This allows you to make changes to columns, constraints, and other properties without losing data.

CONSTRAINTS

M.POOJA
513423104036



```
SQL Plus
 COUNTRY                                                                                                   VARCHAR2(15)
 BLOODGROUP                                                                                                VARCHAR2(10)
 DOB                                                                                                       DATE

SQL> ALTER TABLE Employee_Info ADD Email VARCHAR(255) NOT NULL;

Table altered.

SQL> DESC Employee_Info;
 Name                                                                                          Null?    Type
 ----------------------------------------------------------------------------------------------------------------------------------
 EMPLOYEEID                                                                                             NUMBER(38)
 EMPLOYEENAME                                                                                           VARCHAR2(15)
 EMERGENCYCONTACTNAME                                                                                   VARCHAR2(20)
 PHONENUMBER                                                                                            VARCHAR2(10)
 ADDRESS                                                                                                VARCHAR2(20)
 CITY                                                                                                   VARCHAR2(15)
 COUNTRY                                                                                                VARCHAR2(15)
 BLOODGROUP                                                                                             VARCHAR2(10)
 DOB                                                                                                    DATE
 EMAIL                                                                                         NOT NULL VARCHAR2(255)

SQL> ALTER TABLE Employee_Info ADD CONSTRAINT unique_email UNIQUE (Email);

Table altered.

SQL> DESC Employee_Info;
 Name                                                                                          Null?    Type
 ----------------------------------------------------------------------------------------------------------------------------------
 EMPLOYEEID                                                                                             NUMBER(38)
 EMPLOYEENAME                                                                                           VARCHAR2(15)
 EMERGENCYCONTACTNAME                                                                                   VARCHAR2(20)
 PHONENUMBER                                                                                            VARCHAR2(10)
 ADDRESS                                                                                                VARCHAR2(20)
 CITY                                                                                                   VARCHAR2(15)
 COUNTRY                                                                                                VARCHAR2(15)
 BLOODGROUP                                                                                             VARCHAR2(10)
 DOB                                                                                                    DATE
 EMAIL                                                                                         NOT NULL VARCHAR2(255)

SQL> ALTER TABLE Employee_Info ADD CONSTRAINT chk_age CHECK (YEAR(CURDATE()) - YEAR(DOB) >= 18);
```

NOT NULL:

Ensures a column cannot store NULL (empty) values.

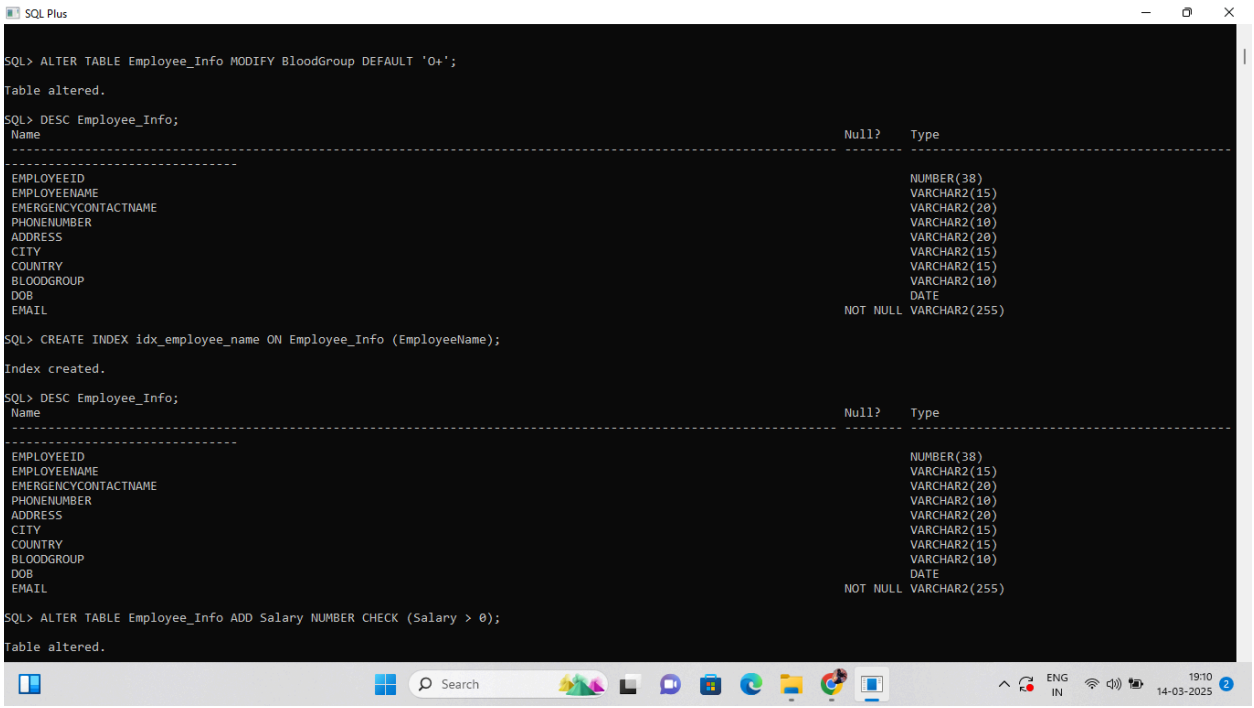
Used when a column must always have a value (e.g., Employee Name).

UNIQUE:

Ensures all values in a column are distinct.

Useful for fields like email or phone numbers.

M.POOJA
513423104036



```
SQL> ALTER TABLE Employee_Info MODIFY BloodGroup DEFAULT 'O+';

Table altered.

SQL> DESC Employee_Info;
 Name                                                                                                                                Null?    Type
 ----------------------------------------------------------------------------------------------------------------------------------- -------- ----------------------------------------------
-------------------------------
 EMPLOYEEID                                                                                                                                   NUMBER(38)
 EMPLOYEENAME                                                                                                                                 VARCHAR2(15)
 EMERGENCYCONTACTNAME                                                                                                                         VARCHAR2(20)
 PHONENUMBER                                                                                                                                  VARCHAR2(10)
 ADDRESS                                                                                                                                      VARCHAR2(20)
 CITY                                                                                                                                         VARCHAR2(15)
 COUNTRY                                                                                                                                      VARCHAR2(15)
 BLOODGROUP                                                                                                                                   VARCHAR2(10)
 DOB                                                                                                                                          DATE
 EMAIL                                                                                                                               NOT NULL VARCHAR2(255)

SQL> CREATE INDEX idx_employee_name ON Employee_Info (EmployeeName);

Index created.

SQL> DESC Employee_Info;
 Name                                                                                                                                Null?    Type
 ----------------------------------------------------------------------------------------------------------------------------------- -------- ----------------------------------------------
-------------------------------
 EMPLOYEEID                                                                                                                                   NUMBER(38)
 EMPLOYEENAME                                                                                                                                 VARCHAR2(15)
 EMERGENCYCONTACTNAME                                                                                                                         VARCHAR2(20)
 PHONENUMBER                                                                                                                                  VARCHAR2(10)
 ADDRESS                                                                                                                                      VARCHAR2(20)
 CITY                                                                                                                                         VARCHAR2(15)
 COUNTRY                                                                                                                                      VARCHAR2(15)
 BLOODGROUP                                                                                                                                   VARCHAR2(10)
 DOB                                                                                                                                          DATE
 EMAIL                                                                                                                               NOT NULL VARCHAR2(255)

SQL> ALTER TABLE Employee_Info ADD Salary NUMBER CHECK (Salary > 0);

Table altered.
```

DEFAULT:

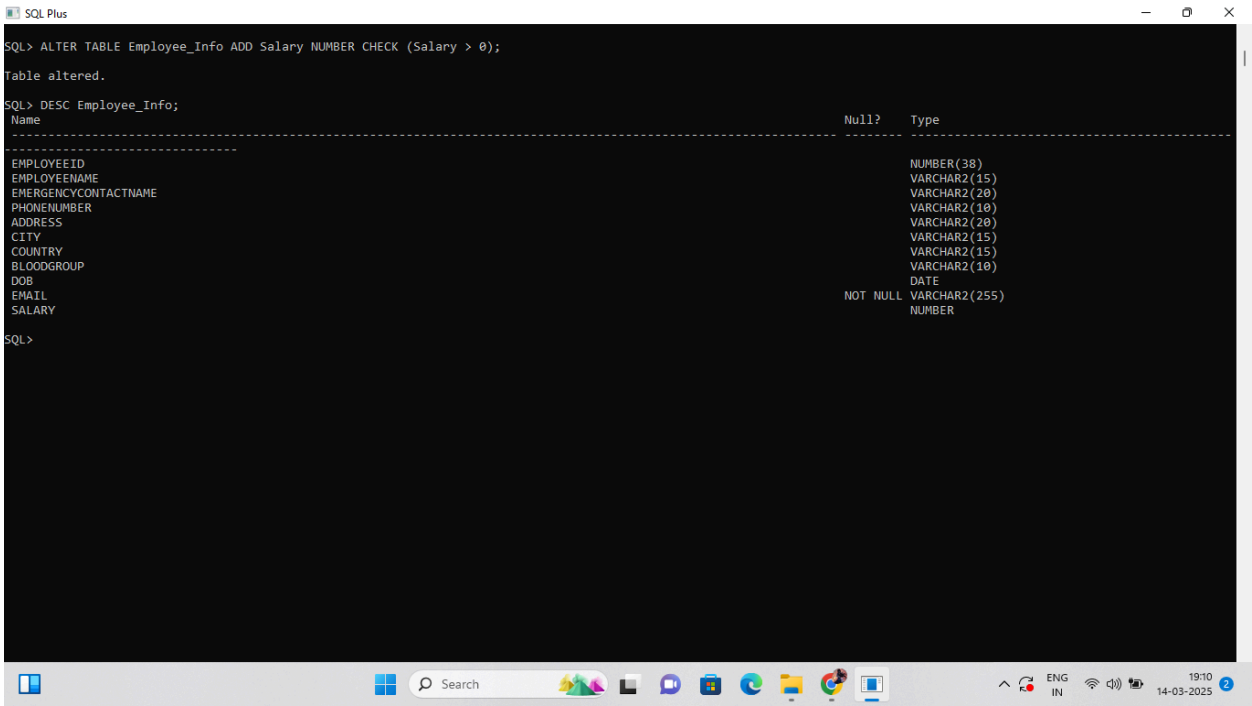
Assigns a default value to a column when no value is provided.
Example: Default country as 'India' if no country is specified.

INDEX:

Improves search performance but does not enforce data integrity.

Used to speed up queries on frequently searched columns.

M.POOJA
513423104036



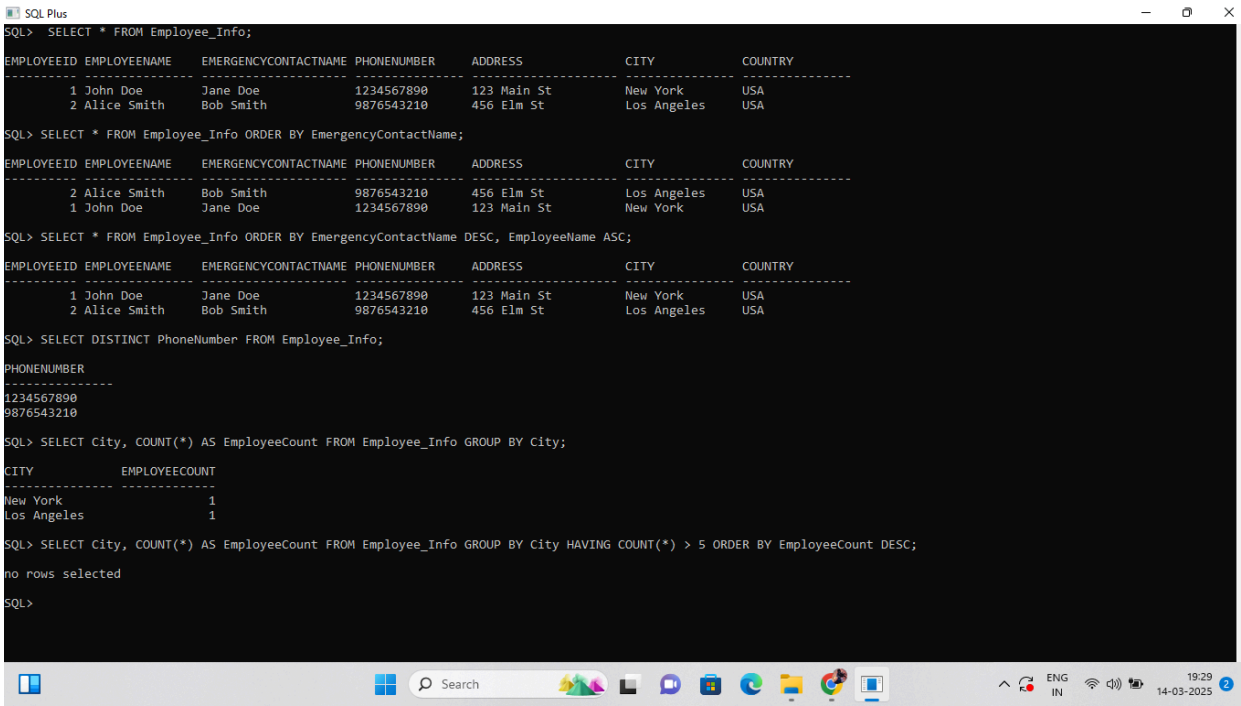
```
SQL> ALTER TABLE Employee_Info ADD Salary NUMBER CHECK (Salary > 0);

Table altered.

SQL> DESC Employee_Info;
 Name                                                                                                                                  Null?    Type
 ------------------------------------------------------------------------------------------------------------------------------------ -------- ----------------------------------------------
--------------------------------
 EMPLOYEEID                                                                                                                                      NUMBER(38)
 EMPLOYEENAME                                                                                                                                    VARCHAR2(15)
 EMERGENCYCONTACTNAME                                                                                                                            VARCHAR2(20)
 PHONENUMBER                                                                                                                                     VARCHAR2(10)
 ADDRESS                                                                                                                                         VARCHAR2(20)
 CITY                                                                                                                                            VARCHAR2(15)
 COUNTRY                                                                                                                                         VARCHAR2(15)
 BLOODGROUP                                                                                                                                      VARCHAR2(10)
 DOB                                                                                                                                             DATE
 EMAIL                                                                                                                                   NOT NULL VARCHAR2(255)
 SALARY                                                                                                                                          NUMBER

SQL>
```

CHECK:

Enforces specific conditions on column values.
Example: Ensuring the salary is greater than a certain amount.

SELECT STATEMENT

M.POOJA
513423104036

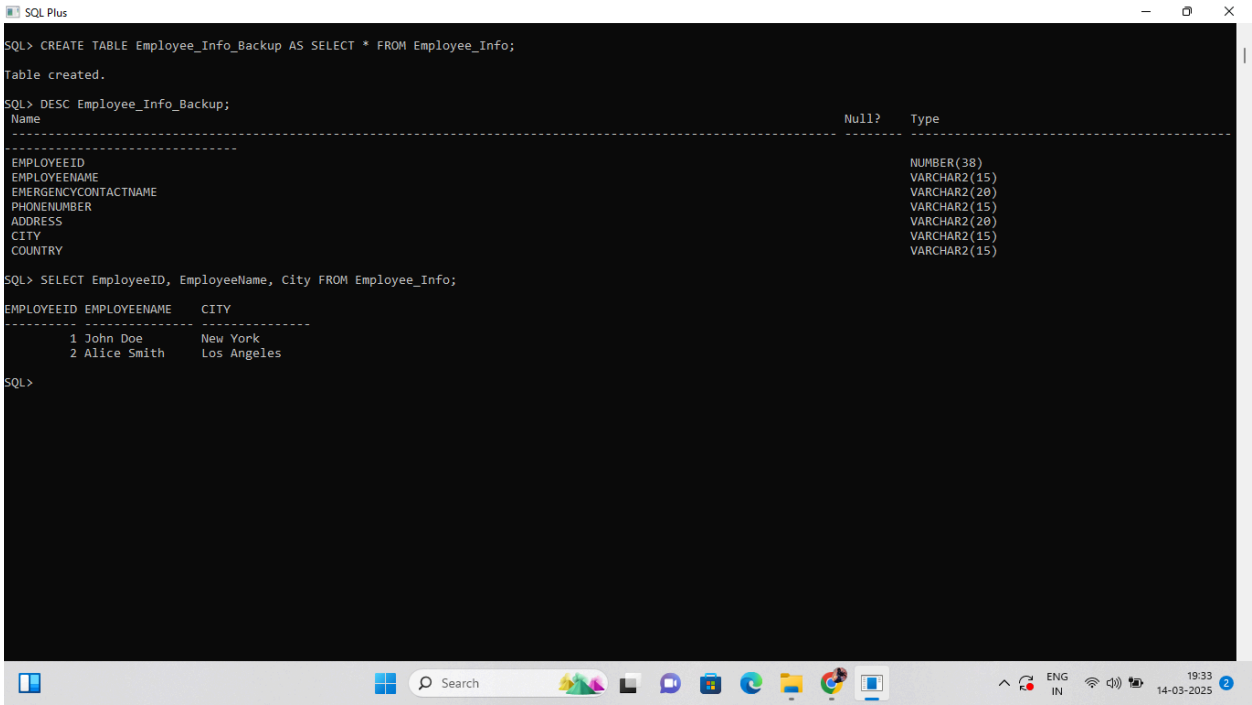


```
SQL Plus
SQL>  SELECT * FROM Employee_Info;

EMPLOYEEID EMPLOYEENAME    EMERGENCYCONTACTNAME PHONENUMBER     ADDRESS              CITY           COUNTRY
---------- --------------- -------------------- --------------- -------------------- -------------- ---------------
         1 John Doe        Jane Doe             1234567890      123 Main St          New York       USA
         2 Alice Smith     Bob Smith            9876543210      456 Elm St           Los Angeles    USA

SQL> SELECT * FROM Employee_Info ORDER BY EmergencyContactName;

EMPLOYEEID EMPLOYEENAME    EMERGENCYCONTACTNAME PHONENUMBER     ADDRESS              CITY           COUNTRY
---------- --------------- -------------------- --------------- -------------------- -------------- ---------------
         2 Alice Smith     Bob Smith            9876543210      456 Elm St           Los Angeles    USA
         1 John Doe        Jane Doe             1234567890      123 Main St          New York       USA

SQL> SELECT * FROM Employee_Info ORDER BY EmergencyContactName DESC, EmployeeName ASC;

EMPLOYEEID EMPLOYEENAME    EMERGENCYCONTACTNAME PHONENUMBER     ADDRESS              CITY           COUNTRY
---------- --------------- -------------------- --------------- -------------------- -------------- ---------------
         1 John Doe        Jane Doe             1234567890      123 Main St          New York       USA
         2 Alice Smith     Bob Smith            9876543210      456 Elm St           Los Angeles    USA

SQL> SELECT DISTINCT PhoneNumber FROM Employee_Info;

PHONENUMBER
---------------
1234567890
9876543210

SQL> SELECT City, COUNT(*) AS EmployeeCount FROM Employee_Info GROUP BY City;

CITY            EMPLOYEECOUNT
--------------- -------------
New York                    1
Los Angeles                 1

SQL> SELECT City, COUNT(*) AS EmployeeCount FROM Employee_Info GROUP BY City HAVING COUNT(*) > 5 ORDER BY EmployeeCount DESC;

no rows selected

SQL>
```

The **SELECT** statement is the most commonly used SQL command. It is used to **retrieve data** from one or more tables in a database.

SELECT INTO STATEMENT

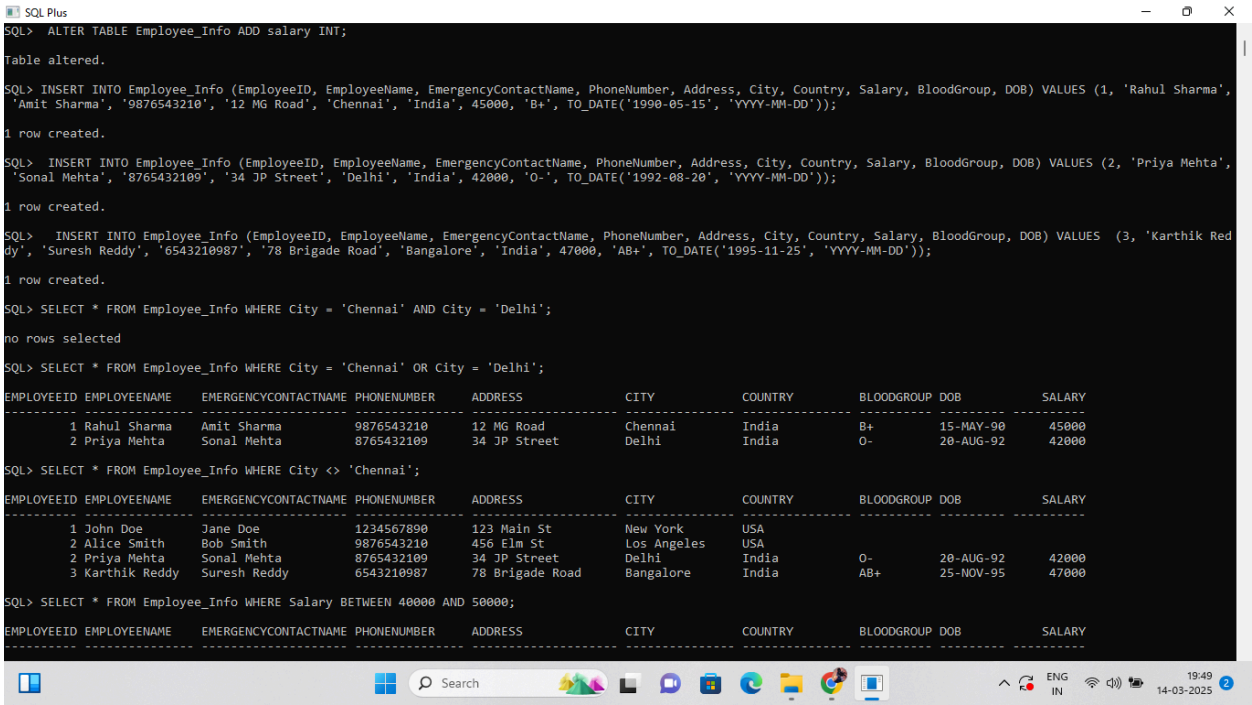
M.POOJA
513423104036



The **SELECT INTO** statement is used to **copy data** from one table into a new table. It creates the new table and inserts the selected data in a single step.

SELECT OPERATOR

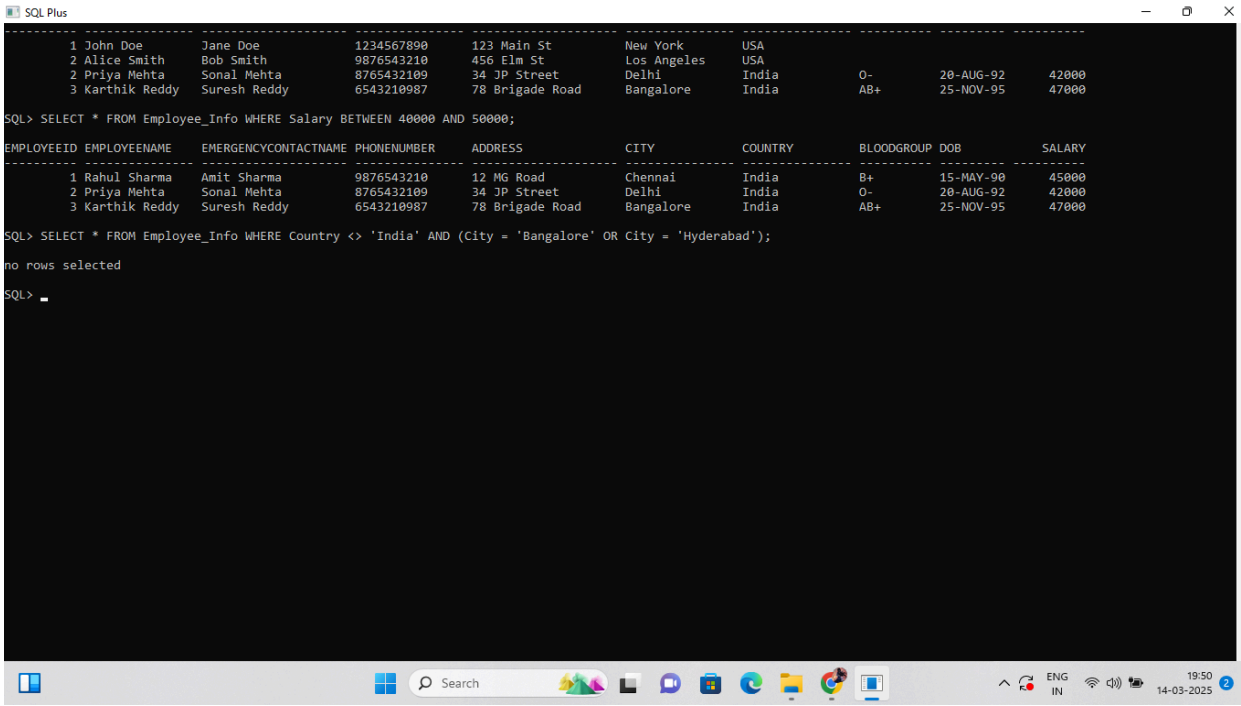
M.POOJA
513423104036



```
SQL Plus
SQL>  ALTER TABLE Employee_Info ADD salary INT;

Table altered.

SQL> INSERT INTO Employee_Info (EmployeeID, EmployeeName, EmergencyContactName, PhoneNumber, Address, City, Country, Salary, BloodGroup, DOB) VALUES (1, 'Rahul Sharma',
 'Amit Sharma', '9876543210', '12 MG Road', 'Chennai', 'India', 45000, 'B+', TO_DATE('1990-05-15', 'YYYY-MM-DD'));

1 row created.

SQL>  INSERT INTO Employee_Info (EmployeeID, EmployeeName, EmergencyContactName, PhoneNumber, Address, City, Country, Salary, BloodGroup, DOB) VALUES (2, 'Priya Mehta',
 'Sonal Mehta', '8765432109', '34 JP Street', 'Delhi', 'India', 42000, 'O-', TO_DATE('1992-08-20', 'YYYY-MM-DD'));

1 row created.

SQL>   INSERT INTO Employee_Info (EmployeeID, EmployeeName, EmergencyContactName, PhoneNumber, Address, City, Country, Salary, BloodGroup, DOB) VALUES  (3, 'Karthik Red
dy', 'Suresh Reddy', '6543210987', '78 Brigade Road', 'Bangalore', 'India', 47000, 'AB+', TO_DATE('1995-11-25', 'YYYY-MM-DD'));

1 row created.

SQL> SELECT * FROM Employee_Info WHERE City = 'Chennai' AND City = 'Delhi';

no rows selected

SQL> SELECT * FROM Employee_Info WHERE City = 'Chennai' OR City = 'Delhi';

EMPLOYEEID EMPLOYEENAME    EMERGENCYCONTACTNAME PHONENUMBER     ADDRESS              CITY           COUNTRY        BLOODGROUP DOB           SALARY
---------- --------------- -------------------- --------------- -------------------- -------------- -------------- ---------- --------- ----------
         1 Rahul Sharma    Amit Sharma          9876543210      12 MG Road           Chennai        India          B+         15-MAY-90      45000
         2 Priya Mehta     Sonal Mehta          8765432109      34 JP Street         Delhi          India          O-         20-AUG-92      42000

SQL> SELECT * FROM Employee_Info WHERE City <> 'Chennai';

EMPLOYEEID EMPLOYEENAME    EMERGENCYCONTACTNAME PHONENUMBER     ADDRESS              CITY           COUNTRY        BLOODGROUP DOB           SALARY
---------- --------------- -------------------- --------------- -------------------- -------------- -------------- ---------- --------- ----------
         1 John Doe        Jane Doe             1234567890      123 Main St          New York       USA
         2 Alice Smith     Bob Smith            9876543210      456 Elm St           Los Angeles    USA
         2 Priya Mehta     Sonal Mehta          8765432109      34 JP Street         Delhi          India          O-         20-AUG-92      42000
         3 Karthik Reddy   Suresh Reddy         6543210987      78 Brigade Road      Bangalore      India          AB+        25-NOV-95      47000

SQL> SELECT * FROM Employee_Info WHERE Salary BETWEEN 40000 AND 50000;

EMPLOYEEID EMPLOYEENAME    EMERGENCYCONTACTNAME PHONENUMBER     ADDRESS              CITY           COUNTRY        BLOODGROUP DOB           SALARY
---------- --------------- -------------------- --------------- -------------------- -------------- -------------- ---------- --------- ----------
```

The **SELECT** statement in SQL is often used with operators to filter, compare, and manipulate data efficiently. These operators help refine query results based on specific conditions.

M.POOJA
513423104036



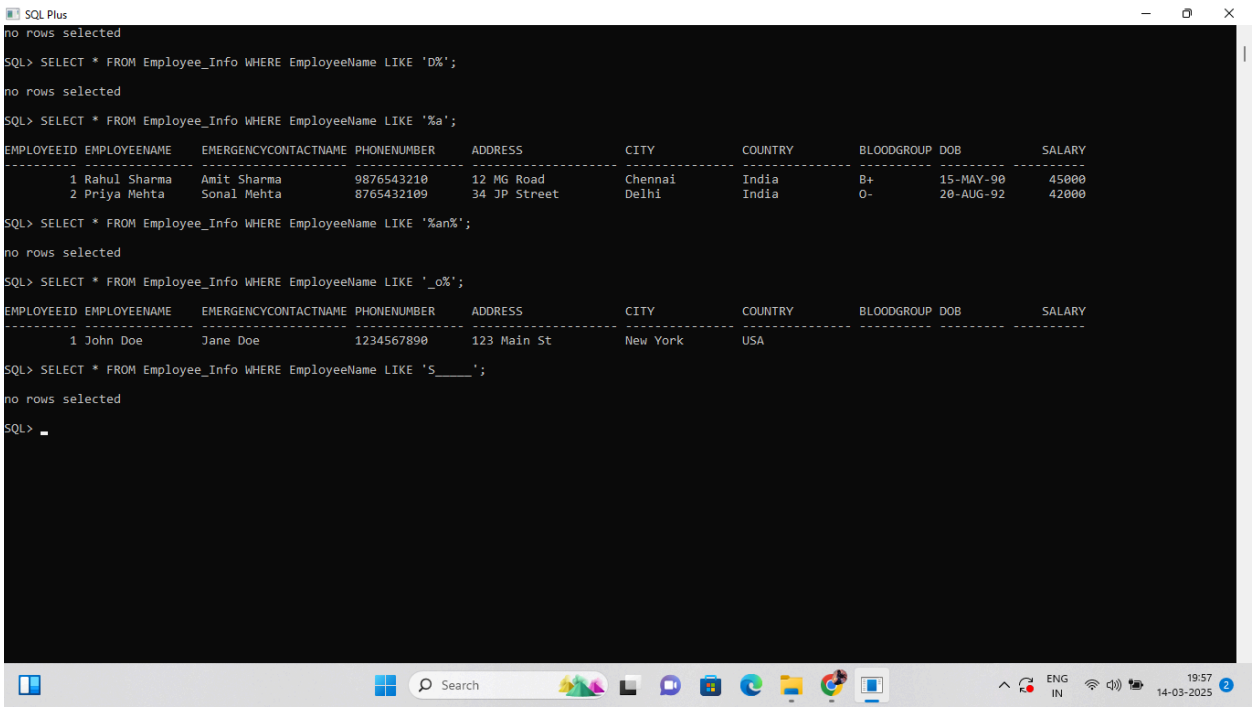
```
SQL Plus
---------- --------------- -------------------- --------------- -------------------- --------------- --------------- ---------- --------- ----------
         1 John Doe        Jane Doe             1234567890      123 Main St          New York        USA
         2 Alice Smith     Bob Smith            9876543210      456 Elm St           Los Angeles     USA
         2 Priya Mehta     Sonal Mehta          8765432109      34 JP Street         Delhi           India           O-         20-AUG-92      42000
         3 Karthik Reddy   Suresh Reddy         6543210987      78 Brigade Road      Bangalore       India           AB+        25-NOV-95      47000

SQL> SELECT * FROM Employee_Info WHERE Salary BETWEEN 40000 AND 50000;

EMPLOYEEID EMPLOYEENAME    EMERGENCYCONTACTNAME PHONENUMBER     ADDRESS              CITY            COUNTRY         BLOODGROUP DOB           SALARY
---------- --------------- -------------------- --------------- -------------------- --------------- --------------- ---------- --------- ----------
         1 Rahul Sharma    Amit Sharma          9876543210      12 MG Road           Chennai         India           B+         15-MAY-90      45000
         2 Priya Mehta     Sonal Mehta          8765432109      34 JP Street         Delhi           India           O-         20-AUG-92      42000
         3 Karthik Reddy   Suresh Reddy         6543210987      78 Brigade Road      Bangalore       India           AB+        25-NOV-95      47000

SQL> SELECT * FROM Employee_Info WHERE Country <> 'India' AND (City = 'Bangalore' OR City = 'Hyderabad');

no rows selected

SQL>
```

.

Arithmetic operators perform mathematical operations, while **comparison operators** such as `=`, `>`, `<`, and `!=` help compare values. **Logical operators** like `AND`, `OR`, and `NOT` allow combining multiple conditions in a query.

LIKE OPERATOR

M.POOJA
513423104036

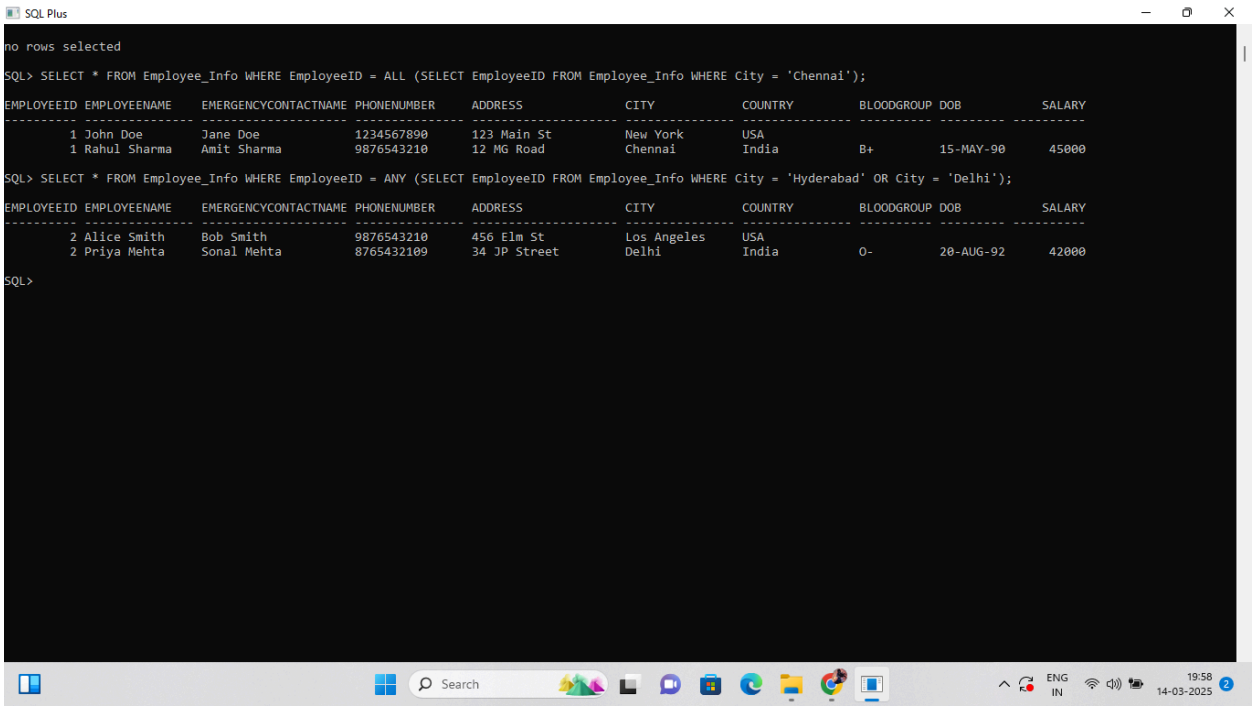


```
SQL Plus
no rows selected

SQL> SELECT * FROM Employee_Info WHERE EmployeeName LIKE 'D%';

no rows selected

SQL> SELECT * FROM Employee_Info WHERE EmployeeName LIKE '%a';

EMPLOYEEID EMPLOYEENAME    EMERGENCYCONTACTNAME PHONENUMBER     ADDRESS              CITY           COUNTRY        BLOODGROUP DOB          SALARY
---------- --------------- -------------------- --------------- -------------------- -------------- -------------- ---------- --------- ----------
         1 Rahul Sharma    Amit Sharma          9876543210      12 MG Road           Chennai        India          B+         15-MAY-90      45000
         2 Priya Mehta     Sonal Mehta          8765432109      34 JP Street         Delhi          India          O-         20-AUG-92      42000

SQL> SELECT * FROM Employee_Info WHERE EmployeeName LIKE '%an%';

no rows selected

SQL> SELECT * FROM Employee_Info WHERE EmployeeName LIKE '_o%';

EMPLOYEEID EMPLOYEENAME    EMERGENCYCONTACTNAME PHONENUMBER     ADDRESS              CITY           COUNTRY        BLOODGROUP DOB          SALARY
---------- --------------- -------------------- --------------- -------------------- -------------- -------------- ---------- --------- ----------
         1 John Doe        Jane Doe             1234567890      123 Main St          New York       USA

SQL> SELECT * FROM Employee_Info WHERE EmployeeName LIKE 'S_____';

no rows selected

SQL>
```

The **LIKE** operator in SQL is used in the `WHERE` clause to search for a specified pattern in a column. It is especially useful for filtering text-based data when you do not know the exact value but need to match a specific pattern.

ALL AND ANY OPERATOR

M.POOJA
513423104036

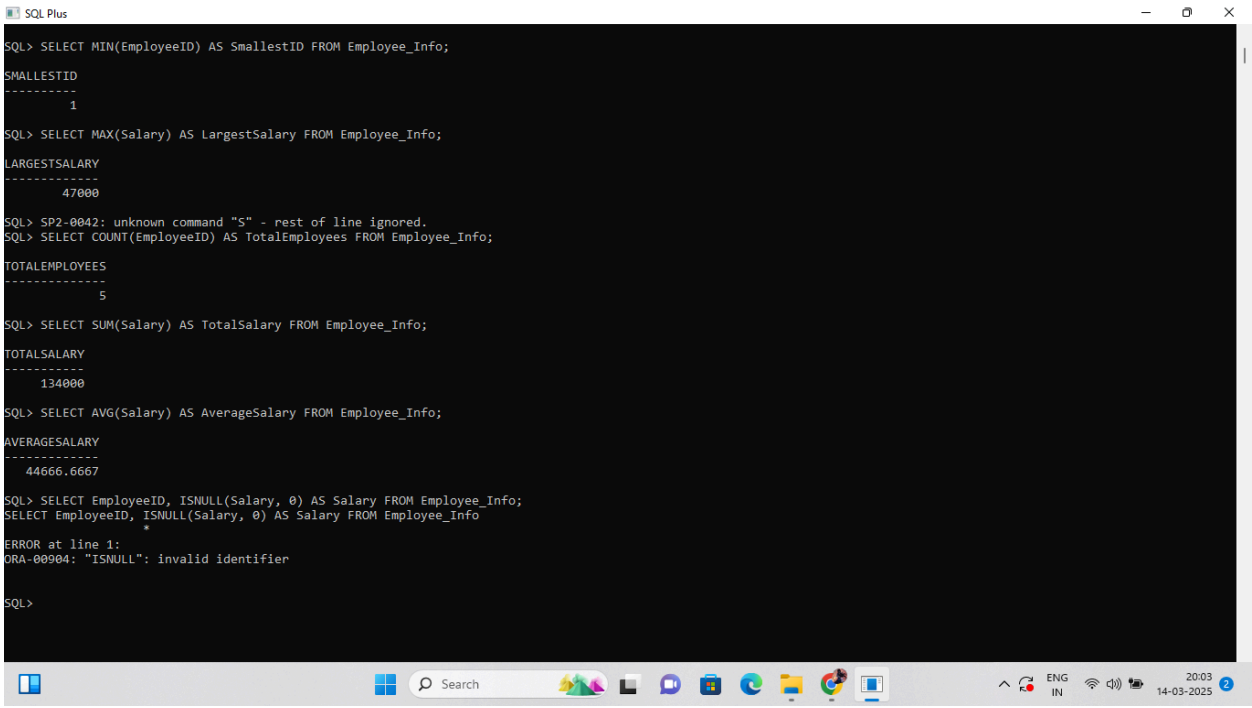


```
SQL Plus

no rows selected

SQL> SELECT * FROM Employee_Info WHERE EmployeeID = ALL (SELECT EmployeeID FROM Employee_Info WHERE City = 'Chennai');

EMPLOYEEID EMPLOYEENAME    EMERGENCYCONTACTNAME PHONENUMBER     ADDRESS              CITY            COUNTRY         BLOODGROUP DOB           SALARY
---------- --------------- -------------------- --------------- -------------------- --------------- --------------- ---------- --------- ----------
         1 John Doe        Jane Doe             1234567890      123 Main St          New York        USA
         1 Rahul Sharma    Amit Sharma          9876543210      12 MG Road           Chennai         India           B+         15-MAY-90      45000

SQL> SELECT * FROM Employee_Info WHERE EmployeeID = ANY (SELECT EmployeeID FROM Employee_Info WHERE City = 'Hyderabad' OR City = 'Delhi');

EMPLOYEEID EMPLOYEENAME    EMERGENCYCONTACTNAME PHONENUMBER     ADDRESS              CITY            COUNTRY         BLOODGROUP DOB           SALARY
---------- --------------- -------------------- --------------- -------------------- --------------- --------------- ---------- --------- ----------
         2 Alice Smith     Bob Smith            9876543210      456 Elm St           Los Angeles     USA
         2 Priya Mehta     Sonal Mehta          8765432109      34 JP Street         Delhi           India           O-         20-AUG-92      42000

SQL>
```

SQL provides various operators to refine and filter data. The **AND** and **ANY** operators are commonly used to apply conditions in queries, especially within the `WHERE` clause.

AGGREGATE FUNCTIONS

M.POOJA
513423104036

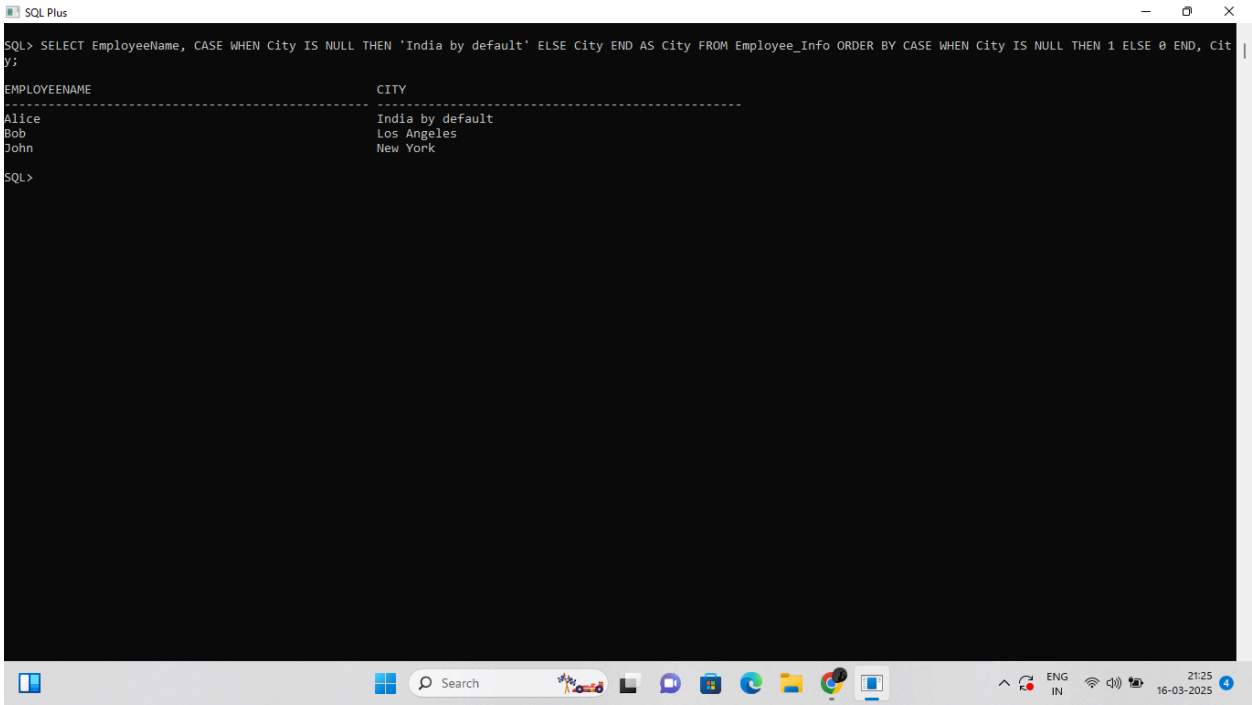


```
SQL> SELECT MIN(EmployeeID) AS SmallestID FROM Employee_Info;

SMALLESTID
----------
         1

SQL> SELECT MAX(Salary) AS LargestSalary FROM Employee_Info;

LARGESTSALARY
-------------
        47000

SQL> SP2-0042: unknown command "S" - rest of line ignored.
SQL> SELECT COUNT(EmployeeID) AS TotalEmployees FROM Employee_Info;

TOTALEMPLOYEES
--------------
             5

SQL> SELECT SUM(Salary) AS TotalSalary FROM Employee_Info;

TOTALSALARY
-----------
     134000

SQL> SELECT AVG(Salary) AS AverageSalary FROM Employee_Info;

AVERAGESALARY
-------------
   44666.6667

SQL> SELECT EmployeeID, ISNULL(Salary, 0) AS Salary FROM Employee_Info;
SELECT EmployeeID, ISNULL(Salary, 0) AS Salary FROM Employee_Info
                   *
ERROR at line 1:
ORA-00904: "ISNULL": invalid identifier


SQL>
```

Aggregate functions in SQL perform calculations on multiple rows of data and return a **single summary value**. These functions are commonly used with the `GROUP BY` clause to analyze and summarize data.

CASE STATEMENTS

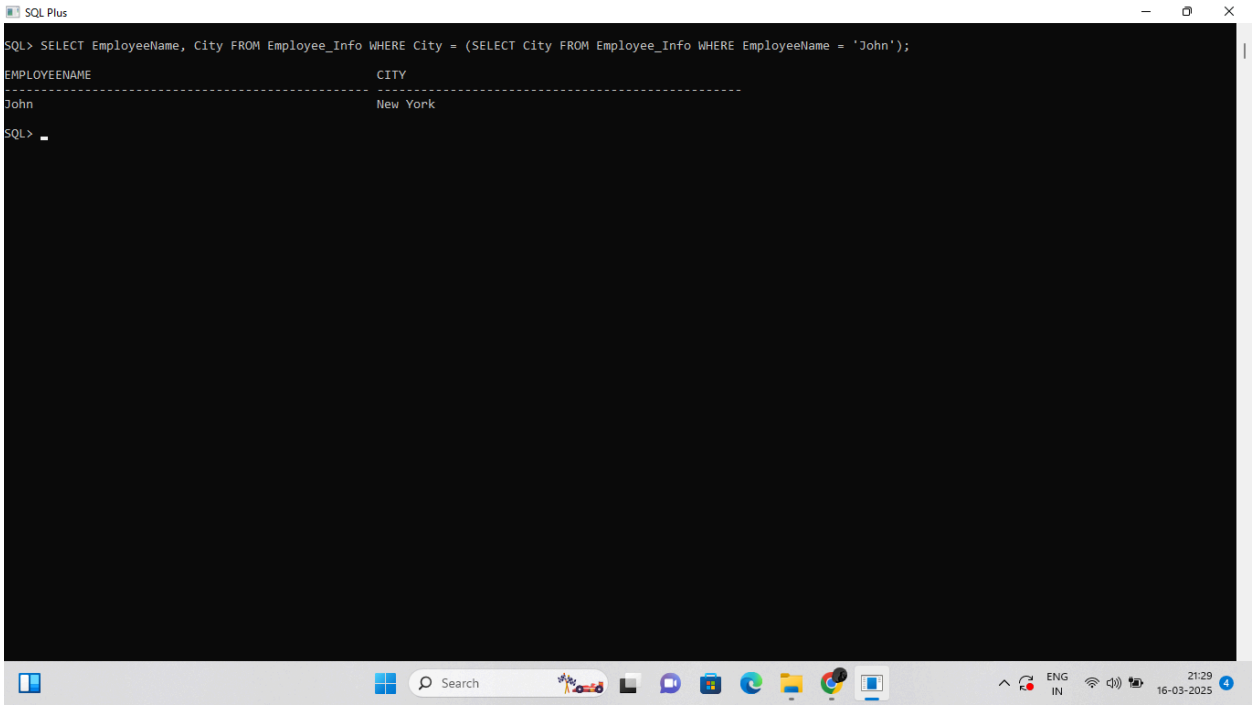
M.POOJA
513423104036



The **CASE statement** in SQL is used to perform **conditional logic** within a query. It acts like an **IF-ELSE** statement in programming, allowing you to return different values based on specific conditions.

NESTED QUERY

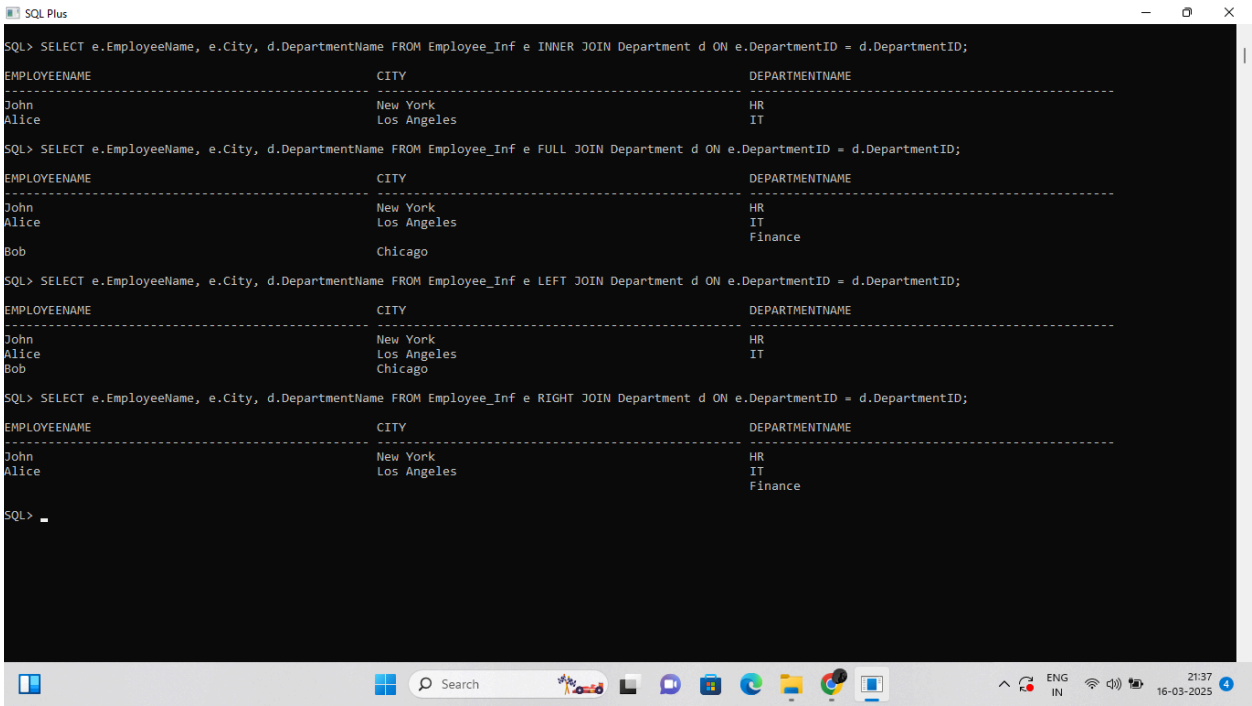
M.POOJA
513423104036



A **nested query**, also known as a **subquery**, is a query that is placed inside another query. It allows you to use the result of one query as an input for another query.

JOINS

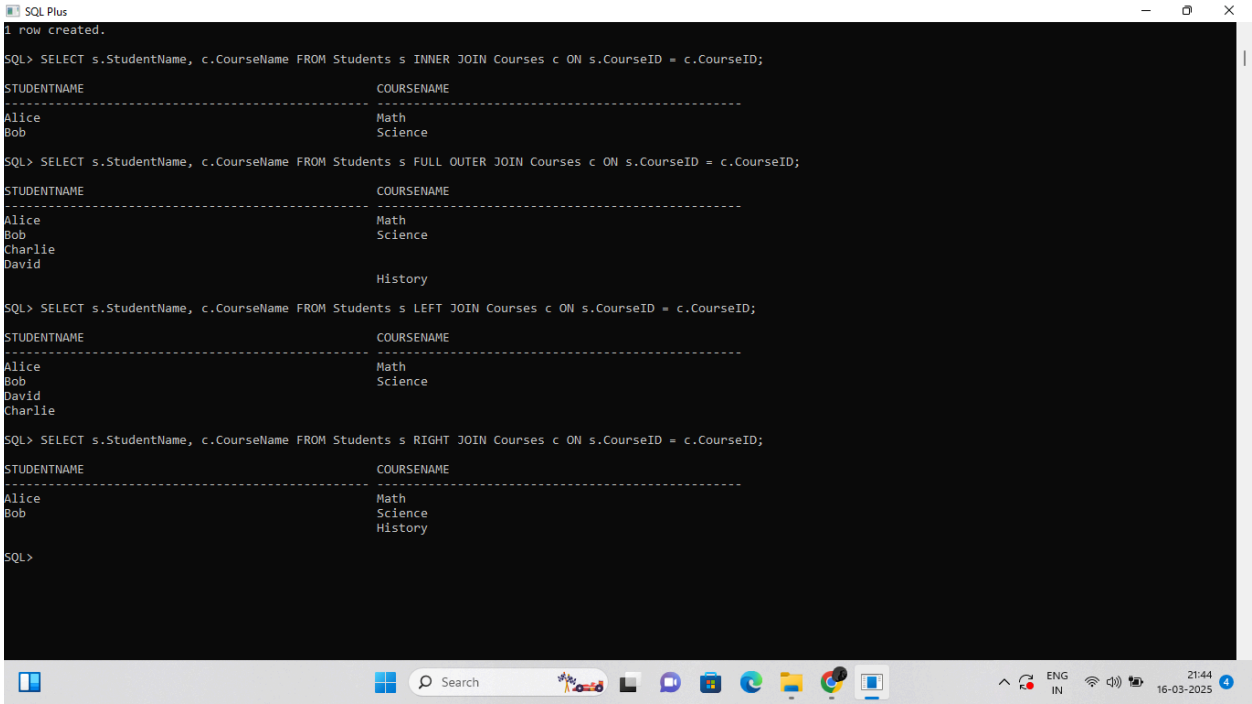
M.POOJA
513423104036



```
SQL Plus
SQL> SELECT e.EmployeeName, e.City, d.DepartmentName FROM Employee_Inf e INNER JOIN Department d ON e.DepartmentID = d.DepartmentID;

EMPLOYEENAME                                       CITY                                               DEPARTMENTNAME
-------------------------------------------------- -------------------------------------------------- --------------------------------------------------
John                                               New York                                           HR
Alice                                              Los Angeles                                        IT

SQL> SELECT e.EmployeeName, e.City, d.DepartmentName FROM Employee_Inf e FULL JOIN Department d ON e.DepartmentID = d.DepartmentID;

EMPLOYEENAME                                       CITY                                               DEPARTMENTNAME
-------------------------------------------------- -------------------------------------------------- --------------------------------------------------
John                                               New York                                           HR
Alice                                              Los Angeles                                        IT
                                                                                                      Finance
Bob                                                Chicago

SQL> SELECT e.EmployeeName, e.City, d.DepartmentName FROM Employee_Inf e LEFT JOIN Department d ON e.DepartmentID = d.DepartmentID;

EMPLOYEENAME                                       CITY                                               DEPARTMENTNAME
-------------------------------------------------- -------------------------------------------------- --------------------------------------------------
John                                               New York                                           HR
Alice                                              Los Angeles                                        IT
Bob                                                Chicago

SQL> SELECT e.EmployeeName, e.City, d.DepartmentName FROM Employee_Inf e RIGHT JOIN Department d ON e.DepartmentID = d.DepartmentID;

EMPLOYEENAME                                       CITY                                               DEPARTMENTNAME
-------------------------------------------------- -------------------------------------------------- --------------------------------------------------
John                                               New York                                           HR
Alice                                              Los Angeles                                        IT
                                                                                                      Finance

SQL>
```

A **JOIN** in SQL is used to combine rows from two or more tables based on a related column between them. It helps in retrieving meaningful information by linking data stored in separate tables.

M.POOJA
513423104036



```
SQL Plus
1 row created.

SQL> SELECT s.StudentName, c.CourseName FROM Students s INNER JOIN Courses c ON s.CourseID = c.CourseID;

STUDENTNAME                                        COURSENAME
-------------------------------------------------- --------------------------------------------------
Alice                                              Math
Bob                                                Science

SQL> SELECT s.StudentName, c.CourseName FROM Students s FULL OUTER JOIN Courses c ON s.CourseID = c.CourseID;

STUDENTNAME                                        COURSENAME
-------------------------------------------------- --------------------------------------------------
Alice                                              Math
Bob                                                Science
Charlie
David
                                                   History

SQL> SELECT s.StudentName, c.CourseName FROM Students s LEFT JOIN Courses c ON s.CourseID = c.CourseID;

STUDENTNAME                                        COURSENAME
-------------------------------------------------- --------------------------------------------------
Alice                                              Math
Bob                                                Science
David
Charlie

SQL> SELECT s.StudentName, c.CourseName FROM Students s RIGHT JOIN Courses c ON s.CourseID = c.CourseID;

STUDENTNAME                                        COURSENAME
-------------------------------------------------- --------------------------------------------------
Alice                                              Math
Bob                                                Science
                                                   History

SQL>
```

INNER JOIN

Returns **only matching records** from both tables based on the condition.

LEFT JOIN (LEFT OUTER JOIN)

Returns **all records from the left table**, and only matching records from the right table.

RIGHT JOIN (RIGHT OUTER JOIN)

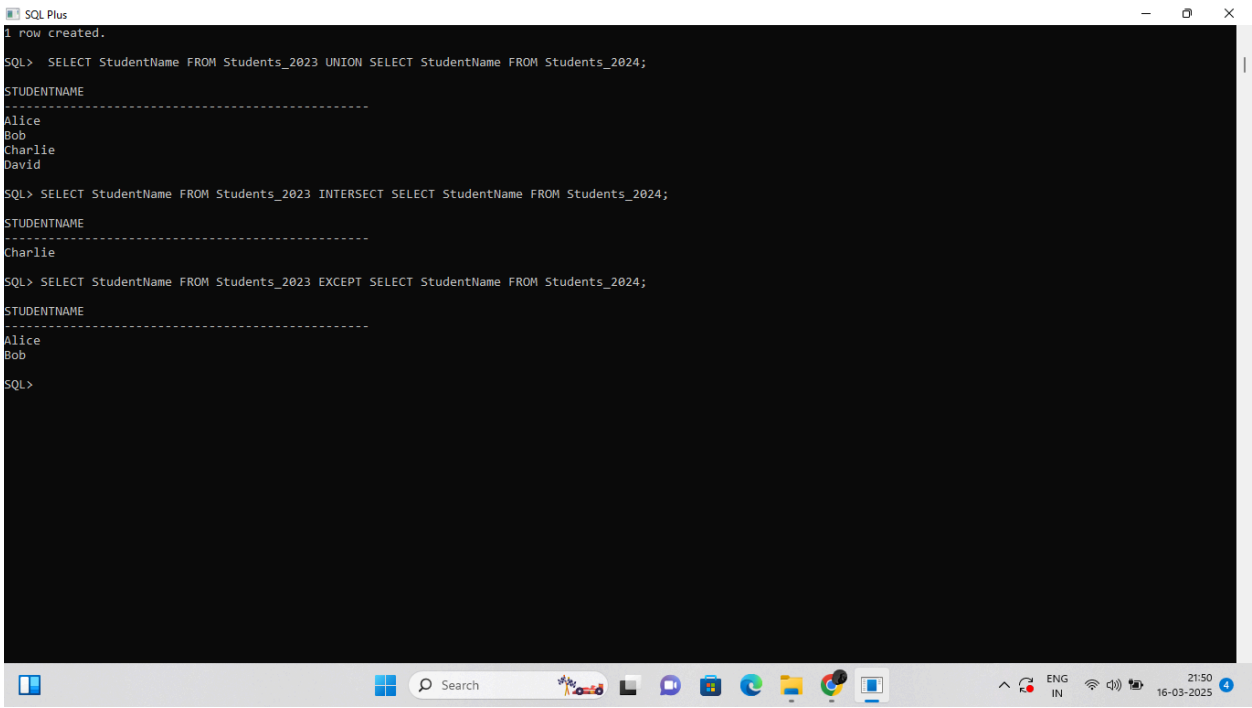
Returns **all records from the right table**, and only matching records from the left table.

FULL JOIN (FULL OUTER JOIN)

Returns **all records from both tables**, showing NULL where there is no match.

SET OPERATIONS

M.POOJA
513423104036



```
SQL Plus
1 row created.

SQL>  SELECT StudentName FROM Students_2023 UNION SELECT StudentName FROM Students_2024;

STUDENTNAME
--------------------------------------------------
Alice
Bob
Charlie
David

SQL> SELECT StudentName FROM Students_2023 INTERSECT SELECT StudentName FROM Students_2024;

STUDENTNAME
--------------------------------------------------
Charlie

SQL> SELECT StudentName FROM Students_2023 EXCEPT SELECT StudentName FROM Students_2024;

STUDENTNAME
--------------------------------------------------
Alice
Bob

SQL>
```

Set operations in SQL are used to **combine results from two or more queries** into a single result set. These operations work similarly to mathematical set operations, allowing **merging, intersecting, or subtracting data** between query results.

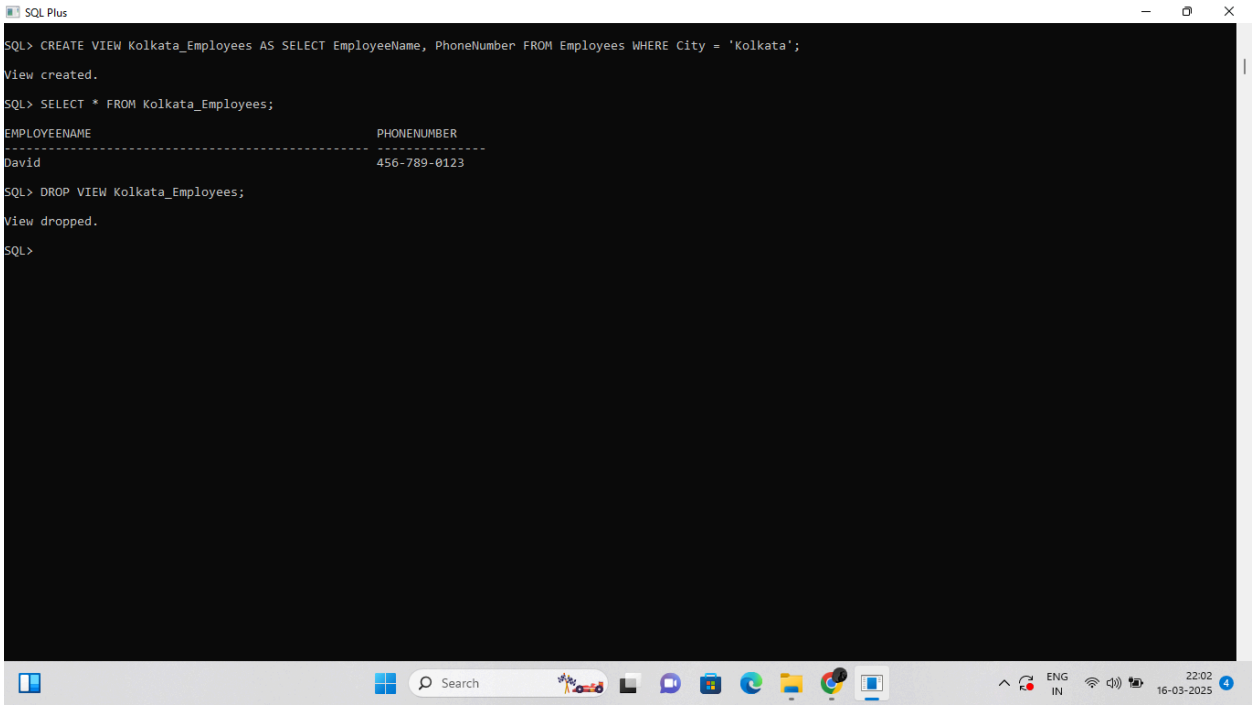
UNION : Combines results from two queries and removes duplicates.

INTERSECT : Returns **only the common records** found in both queries

EXCEPT : Returns records from the first query **that are not in the second query**.

VIEWS

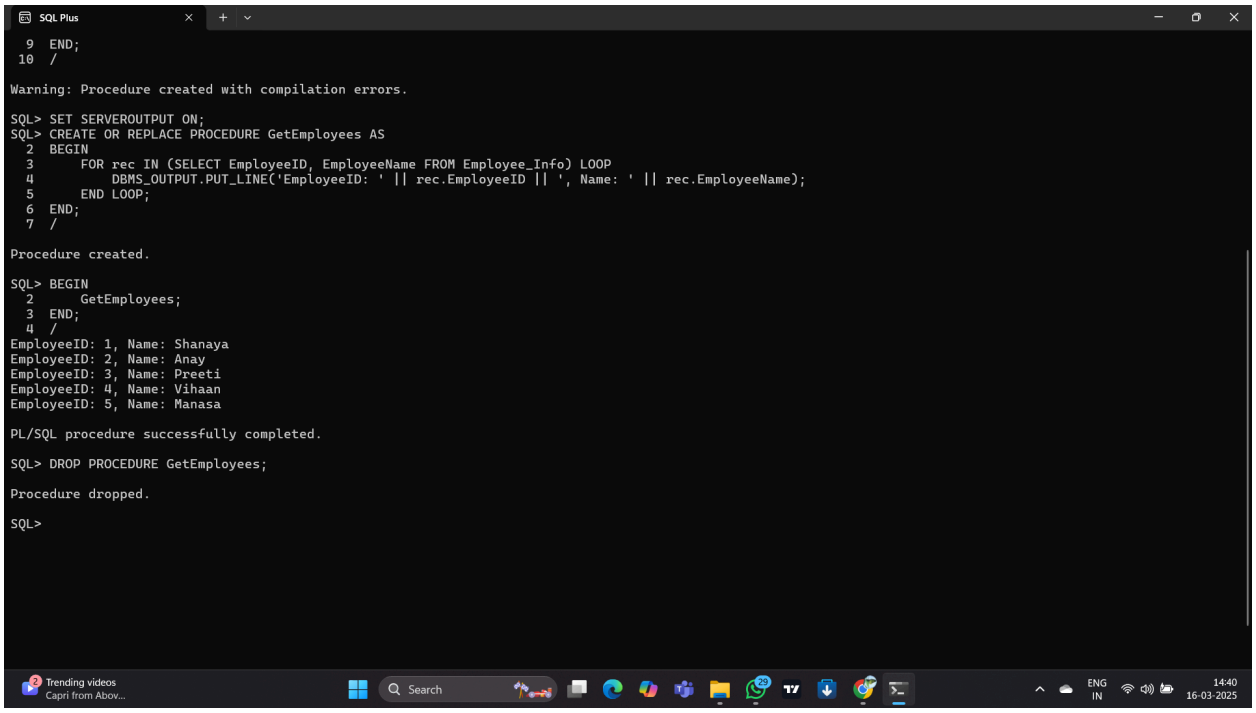
M.POOJA
513423104036



A **view** in SQL is a **virtual table** that displays data from one or more tables. It does not store data itself but **fetches data dynamically** from the underlying tables whenever queried. Views are used to **simplify complex queries, enhance security, and improve data management**.

STORED PROCEDURES

M.POOJA
513423104036



```
  9  END;
 10  /

Warning: Procedure created with compilation errors.

SQL> SET SERVEROUTPUT ON;
SQL> CREATE OR REPLACE PROCEDURE GetEmployees AS
  2  BEGIN
  3      FOR rec IN (SELECT EmployeeID, EmployeeName FROM Employee_Info) LOOP
  4          DBMS_OUTPUT.PUT_LINE('EmployeeID: ' || rec.EmployeeID || ', Name: ' || rec.EmployeeName);
  5      END LOOP;
  6  END;
  7  /

Procedure created.

SQL> BEGIN
  2      GetEmployees;
  3  END;
  4  /
EmployeeID: 1, Name: Shanaya
EmployeeID: 2, Name: Anay
EmployeeID: 3, Name: Preeti
EmployeeID: 4, Name: Vihaan
EmployeeID: 5, Name: Manasa

PL/SQL procedure successfully completed.

SQL> DROP PROCEDURE GetEmployees;

Procedure dropped.

SQL>
```

A **Stored Procedure** in SQL is a **precompiled block of SQL statements** that can be executed multiple times. It is stored in the database and can be called whenever needed, reducing the need to rewrite SQL queries repeatedly.