EXPERIMENT 1

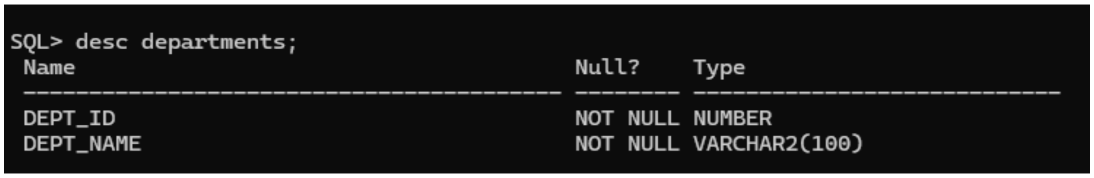
Create a table with constraints

1. **NOT NULL Constraint:** NOT NULL ensures that no NULL values are allowed in the columns.
2. **UNIQUE Constraint:** UNIQUE ensures all values in columns are different (no duplicates allowed).
3. **PRIMARY KEY Constraint:** PRIMARY KEY uniquely identifies each row and implies NOT NULL + UNIQUE.
4. **FOREIGN KEY Constraint:** A **FOREIGN KEY** is a column (or set of columns) in one table that references the primary key in another table to enforce referential integrity.
5. **CHECK Constraint:** CHECK enforces logical rules (e.g., quantity must be between 1 and 999).
6. **DEFAULT Constraint:** DEFAULT automatically sets a value if none is given (e.g., today's date).
7. **CREATE INDEX Constraint:** INDEX improves query performance on large tables, especially for searching/sorting.

Source code:

```
CREATE TABLE departments (
   dept_id INT PRIMARY KEY,
   dept_name VARCHAR(100) UNIQUE NOT NULL);
```

Output:

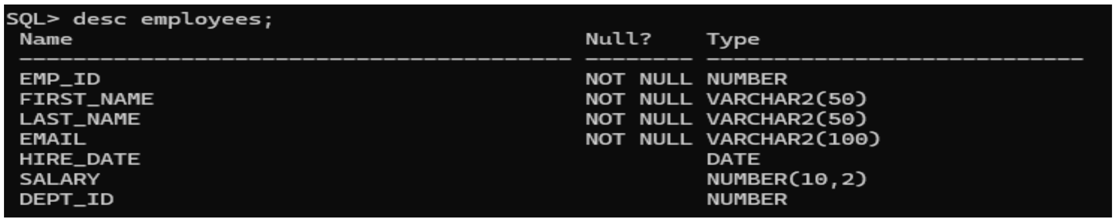


```
SQL> desc departments;
 Name                                      Null?    Type
 ----------------------------------------- -------- ----------------------------
 DEPT_ID                                   NOT NULL NUMBER
 DEPT_NAME                                 NOT NULL VARCHAR2(100)
```

Source code:

```
CREATE TABLE employees (
   emp_id INT PRIMARY KEY,
   first_name VARCHAR(50) NOT NULL,
   last_name VARCHAR(50) NOT NULL,
   email VARCHAR(100) UNIQUE NOT NULL,
   hire_date DATE DEFAULT CURRENT_DATE,
   salary DECIMAL(10,2) CHECK (salary >= 30000),
   dept_id INT, CONSTRAINT fk_department FOREIGN KEY (dept_id)
      REFERENCES departments(dept_id));
```

Output:

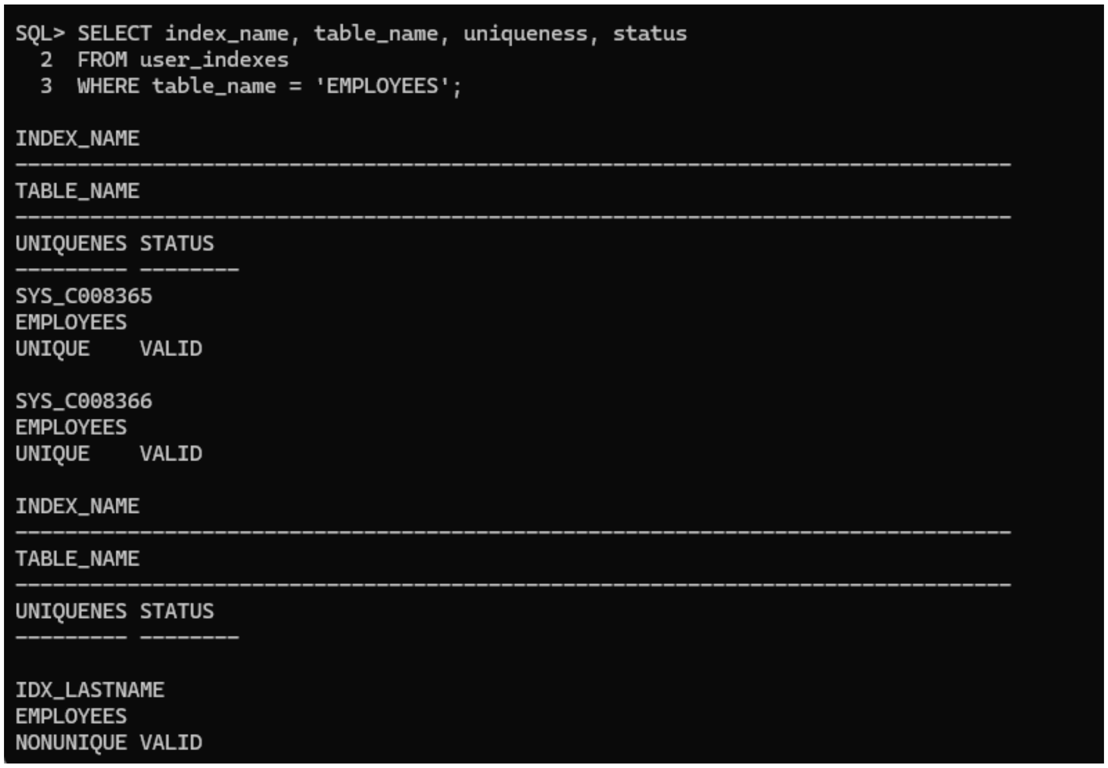


```
SQL> desc employees;
 Name                                      Null?    Type
 ----------------------------------------- -------- ----------------------------
 EMP_ID                                    NOT NULL NUMBER
 FIRST_NAME                                NOT NULL VARCHAR2(50)
 LAST_NAME                                 NOT NULL VARCHAR2(50)
 EMAIL                                     NOT NULL VARCHAR2(100)
 HIRE_DATE                                          DATE
 SALARY                                             NUMBER(10,2)
 DEPT_ID                                            NUMBER
```

Source code:

CREATE INDEX idx_lastname ON employees(last_name);

Output:



```
SQL> SELECT index_name, table_name, uniqueness, status
  2  FROM user_indexes
  3  WHERE table_name = 'EMPLOYEES';

INDEX_NAME
--------------------------------------------------------------------------------
TABLE_NAME
--------------------------------------------------------------------------------
UNIQUENES STATUS
--------- --------
SYS_C008365
EMPLOYEES
UNIQUE    VALID

SYS_C008366
EMPLOYEES
UNIQUE    VALID

INDEX_NAME
--------------------------------------------------------------------------------
TABLE_NAME
--------------------------------------------------------------------------------
UNIQUENES STATUS
--------- --------

IDX_LASTNAME
EMPLOYEES
NONUNIQUE VALID
```

EXPERIMENT 2

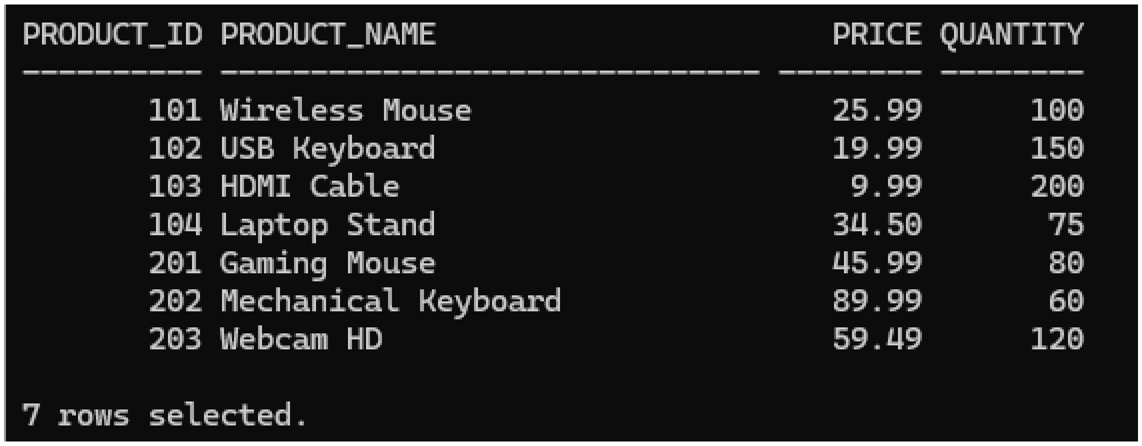
Implementation of SQL Commands

1. **Insert values with a single entry**: Adds one row of data into a table using the INSERT INTO statement.
2. **Insert values with multiple entries**: Adds multiple rows at once using a single INSERT INTO statement with multiple value sets.
3. **ALTER Table Structure**: Modifies the structure of an existing table, such as adding or removing columns.
4. **VIEW Table structure**: Displays the schema of a table using commands like DESCRIBE or SHOW COLUMNS.
5. **UPDATE table**: Changes existing data in one or more rows using the UPDATE statement with a WHERE clause.
6. **DELETE Rows in table**: Removes specific rows from a table using the DELETE FROM statement with a WHERE clause.
7. **DROP table**: Permanently deletes an entire table and all its data from the database.

Source code:

```
CREATE TABLE products (
   product_id INT PRIMARY KEY,
   product_name VARCHAR(100) NOT NULL,
   price DECIMAL(10, 2) NOT NULL,
   quantity INT NOT NULL);
INSERT INTO products (product_id, product_name, price, quantity)
VALUES (101, 'Wireless Mouse', 25.99, 100);
INSERT INTO products (product_id, product_name, price, quantity) VALUES
(102, 'USB Keyboard', 19.99, 150);
INSERT INTO products (product_id, product_name, price, quantity) VALUES
(103, 'HDMI Cable', 9.99, 200);
INSERT INTO products (product_id, product_name, price, quantity) VALUES
(104, 'Laptop Stand', 34.50, 75);
INSERT ALL
   INTO products (product_id, product_name, price, quantity) VALUES (201,
'Gaming Mouse', 45.99, 80)
   INTO products (product_id, product_name, price, quantity) VALUES (202,
'Mechanical Keyboard', 89.99, 60)
   INTO products (product_id, product_name, price, quantity) VALUES (203,
'Webcam HD', 59.49, 120)
SELECT * FROM dual;
```

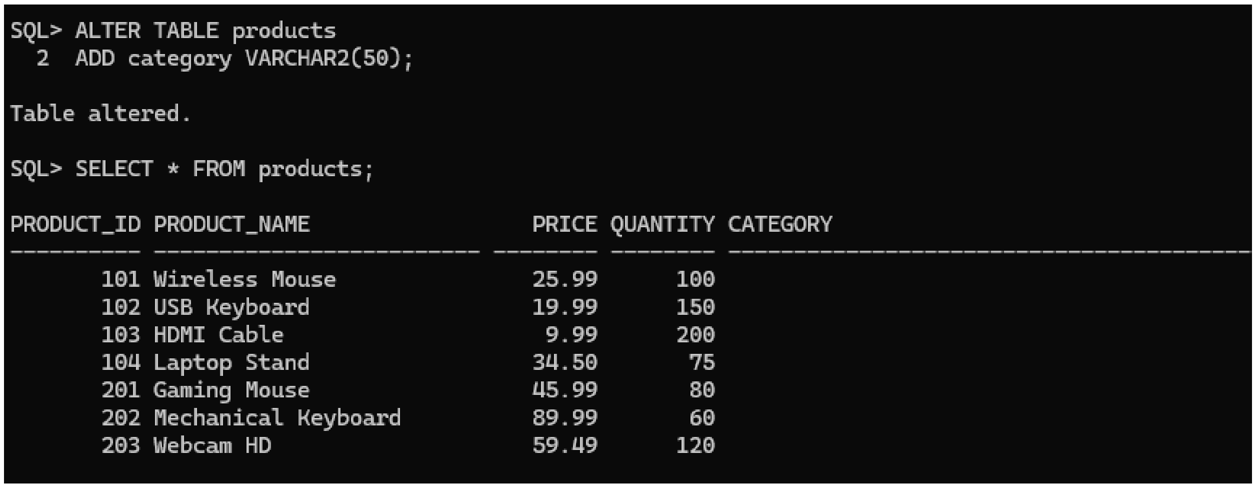
Output:



```
PRODUCT_ID PRODUCT_NAME                         PRICE QUANTITY
---------- ------------------------------ -------- --------
       101 Wireless Mouse                     25.99      100
       102 USB Keyboard                       19.99      150
       103 HDMI Cable                          9.99      200
       104 Laptop Stand                       34.50       75
       201 Gaming Mouse                       45.99       80
       202 Mechanical Keyboard                89.99       60
       203 Webcam HD                          59.49      120

7 rows selected.
```

Source code:
ALTER TABLE products
ADD category VARCHAR2(50);

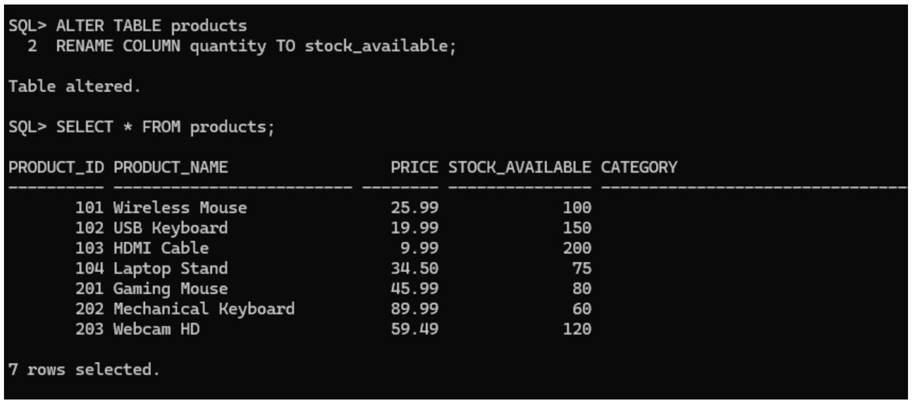
Output:



```
SQL> ALTER TABLE products
  2  ADD category VARCHAR2(50);

Table altered.

SQL> SELECT * FROM products;

PRODUCT_ID PRODUCT_NAME                   PRICE QUANTITY CATEGORY
---------- ------------------------- -------- -------- ----------------------------------------
       101 Wireless Mouse               25.99      100
       102 USB Keyboard                 19.99      150
       103 HDMI Cable                    9.99      200
       104 Laptop Stand                 34.50       75
       201 Gaming Mouse                 45.99       80
       202 Mechanical Keyboard          89.99       60
       203 Webcam HD                    59.49      120
```

Source code:
ALTER TABLE products
REBNAME COLUMN quantity TO stock_available;

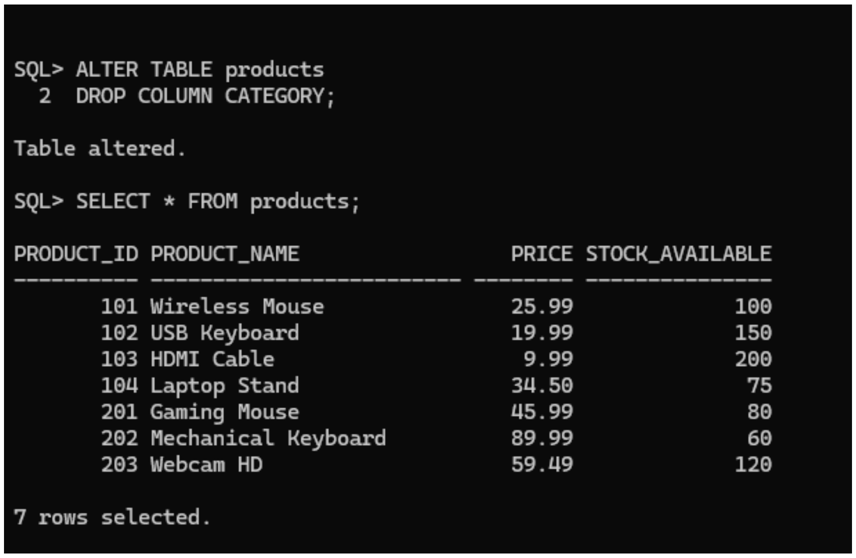
Output:



```
SQL> ALTER TABLE products
  2  RENAME COLUMN quantity TO stock_available;

Table altered.

SQL> SELECT * FROM products;

PRODUCT_ID PRODUCT_NAME                   PRICE STOCK_AVAILABLE CATEGORY
---------- ------------------------- -------- --------------- --------------------------------
       101 Wireless Mouse               25.99             100
       102 USB Keyboard                 19.99             150
       103 HDMI Cable                    9.99             200
       104 Laptop Stand                 34.50              75
       201 Gaming Mouse                 45.99              80
       202 Mechanical Keyboard          89.99              60
       203 Webcam HD                    59.49             120

7 rows selected.
```

Source code:
ALTER TABLE products
DROP COLUMN CATEGORY;

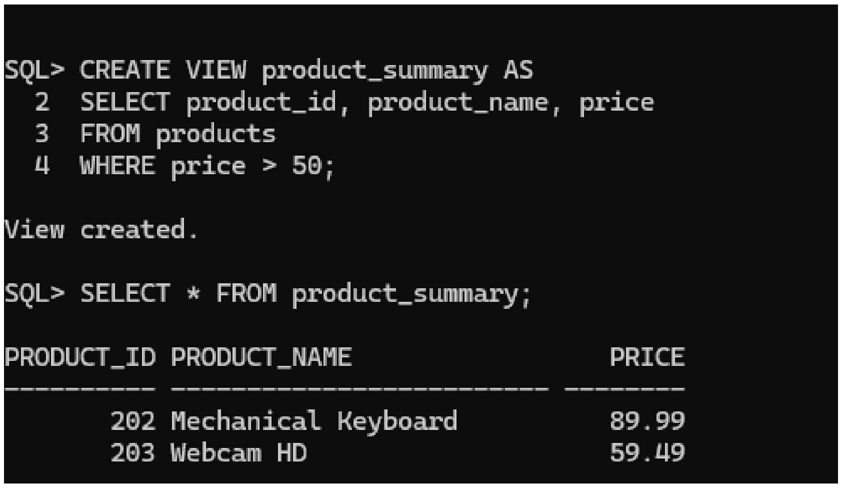
Output:



```
SQL> ALTER TABLE products
  2  DROP COLUMN CATEGORY;

Table altered.

SQL> SELECT * FROM products;

PRODUCT_ID PRODUCT_NAME                 PRICE STOCK_AVAILABLE
---------- ------------------------- -------- ---------------
       101 Wireless Mouse               25.99             100
       102 USB Keyboard                 19.99             150
       103 HDMI Cable                    9.99             200
       104 Laptop Stand                 34.50              75
       201 Gaming Mouse                 45.99              80
       202 Mechanical Keyboard          89.99              60
       203 Webcam HD                    59.49             120

7 rows selected.
```

Source code:
CREATE VIEW product_summary AS
SELECT products_id,product_name,price
FROM products
WHERE price > 50;

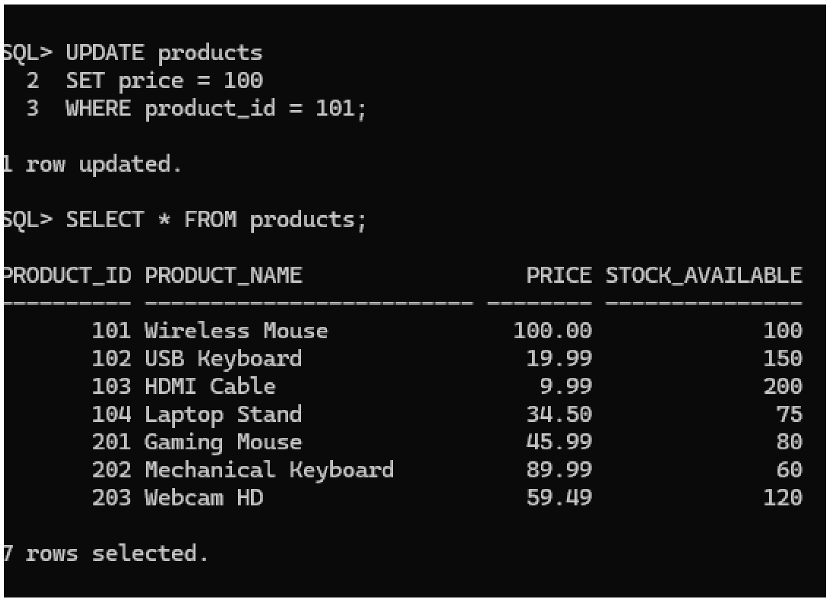
Output:



```
SQL> CREATE VIEW product_summary AS
  2  SELECT product_id, product_name, price
  3  FROM products
  4  WHERE price > 50;

View created.

SQL> SELECT * FROM product_summary;

PRODUCT_ID PRODUCT_NAME                 PRICE
---------- ------------------------- --------
       202 Mechanical Keyboard          89.99
       203 Webcam HD                    59.49
```

Source code:
UPDATE products
SET price=100
WHERE product_id=101;

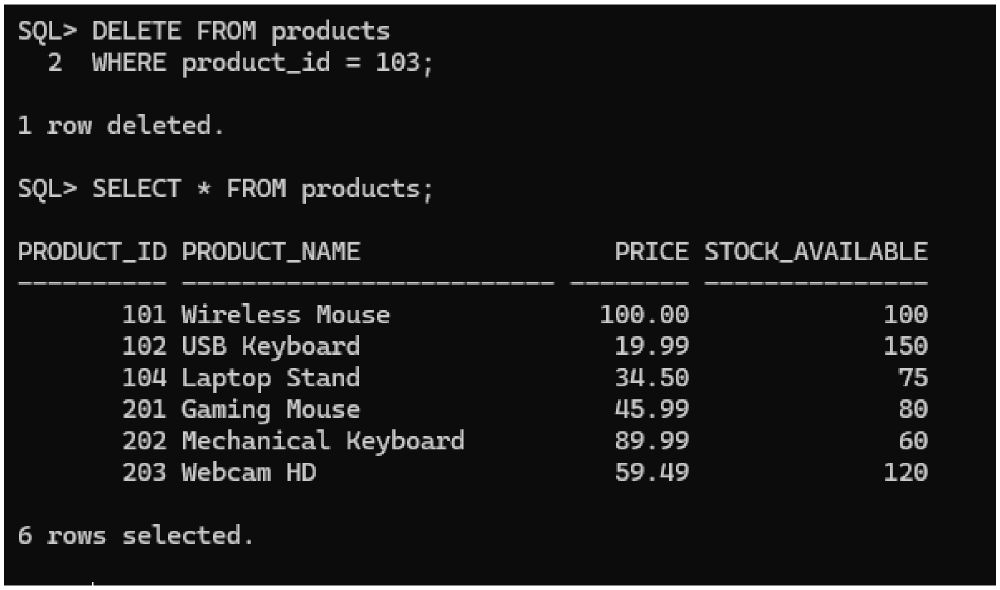
Output:



```
SQL> UPDATE products
  2  SET price = 100
  3  WHERE product_id = 101;

1 row updated.

SQL> SELECT * FROM products;

PRODUCT_ID PRODUCT_NAME                  PRICE STOCK_AVAILABLE
---------- ------------------------- -------- ---------------
       101 Wireless Mouse               100.00             100
       102 USB Keyboard                  19.99             150
       103 HDMI Cable                     9.99             200
       104 Laptop Stand                  34.50              75
       201 Gaming Mouse                  45.99              80
       202 Mechanical Keyboard           89.99              60
       203 Webcam HD                     59.49             120

7 rows selected.
```

Source code:
DELETE FROM products
WHERE products_id=103;

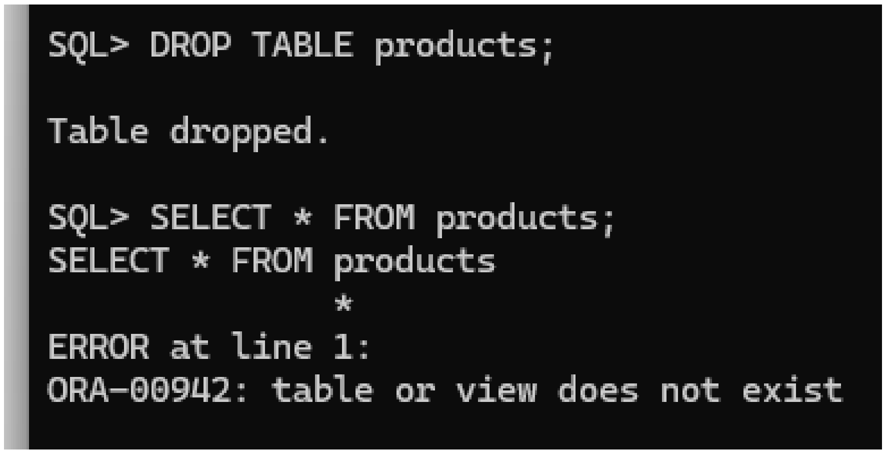
Output:



```
SQL> DELETE FROM products
  2  WHERE product_id = 103;

1 row deleted.

SQL> SELECT * FROM products;

PRODUCT_ID PRODUCT_NAME                  PRICE STOCK_AVAILABLE
---------- ------------------------- -------- ---------------
       101 Wireless Mouse               100.00             100
       102 USB Keyboard                  19.99             150
       104 Laptop Stand                  34.50              75
       201 Gaming Mouse                  45.99              80
       202 Mechanical Keyboard           89.99              60
       203 Webcam HD                     59.49             120

6 rows selected.
```

Source code:
DROP TABLE products;

Output:



```
SQL> DROP TABLE products;

Table dropped.

SQL> SELECT * FROM products;
SELECT * FROM products
              *
ERROR at line 1:
ORA-00942: table or view does not exist
```

EXPERIMENT 3

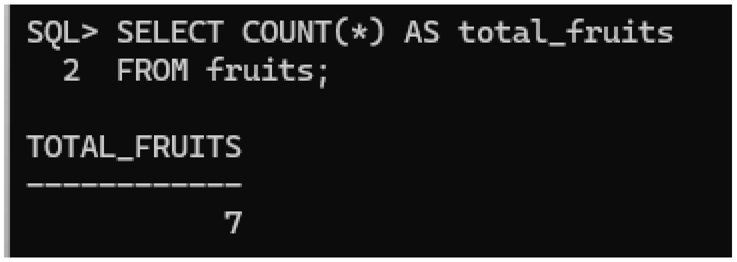
Aggregate Function

1. **MIN()**: Returns the smallest value in a column.
2. **MAX()**: Returns the largest value in a column.
3. **COUNT()**: Returns the number of rows that match a specified condition.
4. **SUM()**: Calculates the total sum of a numeric column.
5. **AVG()**: Computes the average value of a numeric column.

SourceCode:

```
 CREATE TABLE fruits (
       product_id INT PRIMARY KEY,
       product_name VARCHAR2(100) NOT NULL,
       price NUMBER(10,2) NOT NULL,
       quantity INT NOT NULL);
INSERT INTO fruits (product_id, product_name, price, quantity) VALUES (101,
'Apple', 0.99, 100);
INSERT INTO fruits (product_id, product_name, price, quantity) VALUES (102,
'Banana', 0.59, 150);
INSERT INTO fruits (product_id, product_name, price, quantity) VALUES (103,
'Orange', 1.25, 200);
INSERT INTO fruits (product_id, product_name, price, quantity) VALUES (104,
'Mango', 2.50, 75);
INSERT INTO fruits (product_id, product_name, price, quantity) VALUES (201,
'Grapes', 3.00, 80);
INSERT INTO fruits (product_id, product_name, price, quantity) VALUES (202,
'Pineapple', 2.99, 60);
INSERT INTO fruits (product_id, product_name, price, quantity) VALUES (203,
'Watermelon', 5.49, 120);
SELECT COUNT(*) AS total_fruits
FROM fruits;
```

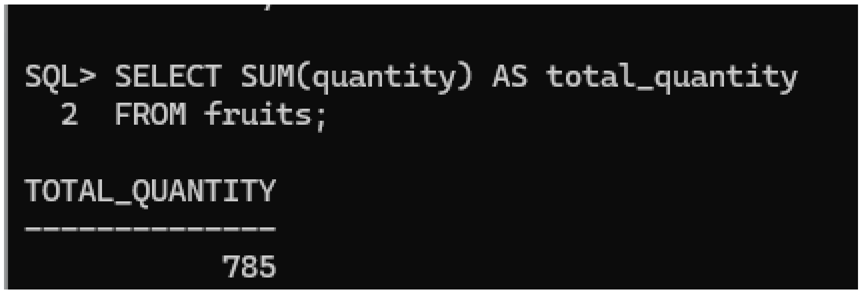
Output:



```
SQL> SELECT COUNT(*) AS total_fruits
  2  FROM fruits;

TOTAL_FRUITS
------------
           7
```

SourceCode:
SELECT SUM(quantity) AS total_quantity
FROM fruits;

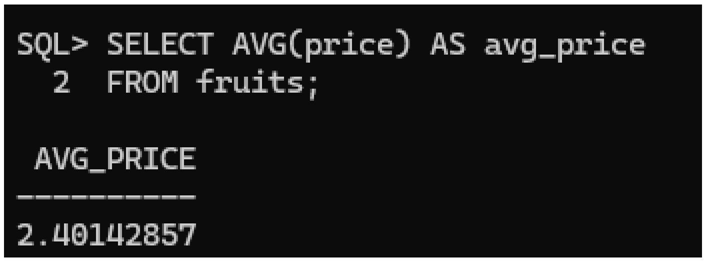
Output:



```
SQL> SELECT SUM(quantity) AS total_quantity
  2  FROM fruits;

TOTAL_QUANTITY
--------------
           785
```

SourceCode:
SELECT AVG(price) AS avg_price
FROM fruits;

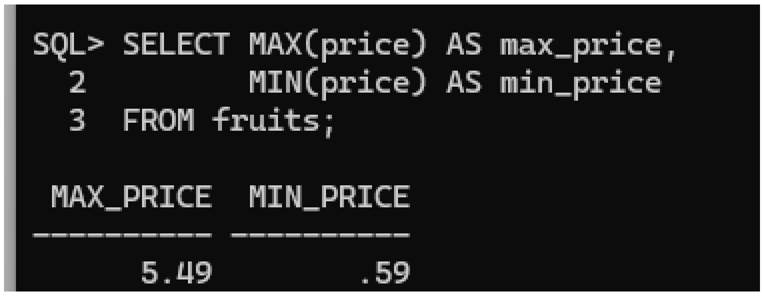
Output:



```
SQL> SELECT AVG(price) AS avg_price
  2  FROM fruits;

 AVG_PRICE
----------
2.40142857
```

SourceCode:
SELECT MAX(price) AS max_price,
MIN(price) AS min_price
FROM fruits;

Output:



```
SQL> SELECT MAX(price) AS max_price,
  2         MIN(price) AS min_price
  3  FROM fruits;

 MAX_PRICE  MIN_PRICE
---------- ----------
      5.49        .59
```

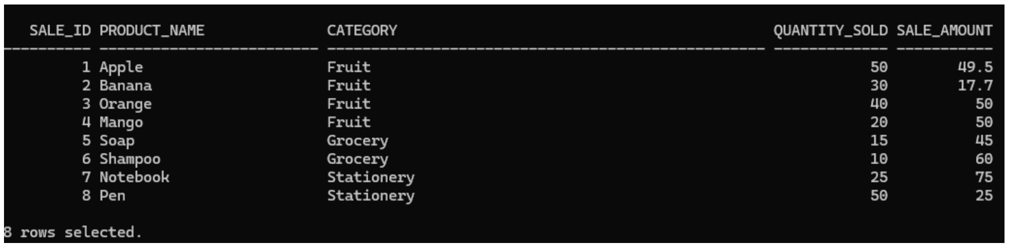
EXPERIMENT 4

1. **GROUP BY**: Organizes rows into groups based on one or more columns, often used with aggregate functions.
2. **ORDER BY**: Sorts the result set of a query by one or more columns in ascending (ASC) or descending (DESC) order.

Source code:

```
CREATE TABLE sales (
   sale_id INT PRIMARY KEY,
   product_name VARCHAR2(100) NOT NULL,
   category VARCHAR2(50),
   quantity_sold INT NOT NULL,
   sale_amount NUMBER(10,2) NOT NULL);
INSERT INTO sales (sale_id, product_name, category, quantity_sold, sale_amount) VALUES (1, 'Apple', 'Fruit', 50, 49.50);
INSERT INTO sales (sale_id, product_name, category, quantity_sold, sale_amount) VALUES (2, 'Banana', 'Fruit', 30, 17.70);
INSERT INTO sales (sale_id, product_name, category, quantity_sold, sale_amount) VALUES (3, 'Orange', 'Fruit', 40, 50.00);
INSERT INTO sales (sale_id, product_name, category, quantity_sold, sale_amount) VALUES (4, 'Mango', 'Fruit', 20, 50.00);
INSERT INTO sales (sale_id, product_name, category, quantity_sold, sale_amount) VALUES (5, 'Soap', 'Grocery', 15, 45.00);
INSERT INTO sales (sale_id, product_name, category, quantity_sold, sale_amount) VALUES (6, 'Shampoo', 'Grocery', 10, 60.00);
INSERT INTO sales (sale_id, product_name, category, quantity_sold, sale_amount) VALUES (7, 'Notebook', 'Stationery', 25, 75.00);
INSERT INTO sales (sale_id, product_name, category, quantity_sold, sale_amount) VALUES (8, 'Pen', 'Stationery', 50, 25.00);
```

Output:

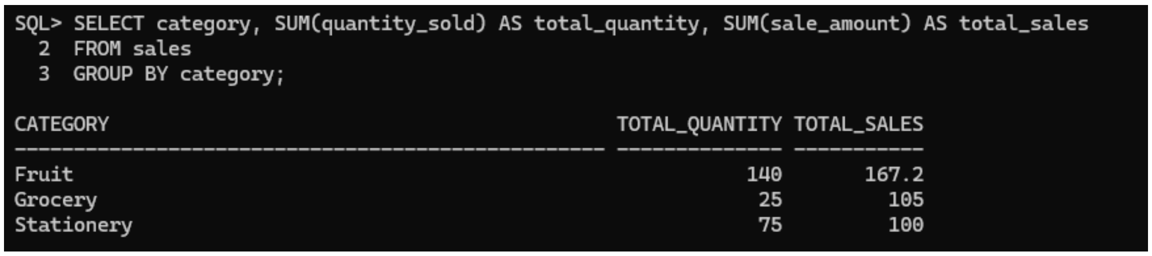


```
   SALE_ID PRODUCT_NAME              CATEGORY                                           QUANTITY_SOLD SALE_AMOUNT
---------- ------------------------- -------------------------------------------------- ------------- -----------
         1 Apple                     Fruit                                                         50        49.5
         2 Banana                    Fruit                                                         30        17.7
         3 Orange                    Fruit                                                         40          50
         4 Mango                     Fruit                                                         20          50
         5 Soap                      Grocery                                                       15          45
         6 Shampoo                   Grocery                                                       10          60
         7 Notebook                  Stationery                                                    25          75
         8 Pen                       Stationery                                                    50          25

8 rows selected.
```

Source code:

```
SELECT category, SUM(quantity_sold) AS total_quantity, SUM(sale_amount) AS total_sales
FROM sales GROUP BY category;
```

Output:

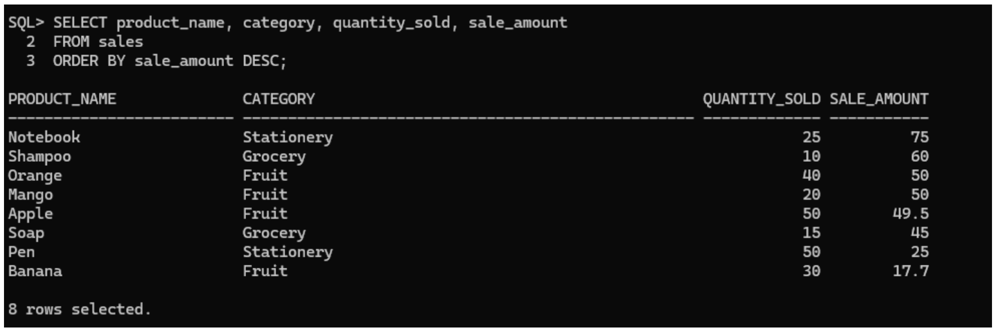


```
SQL> SELECT category, SUM(quantity_sold) AS total_quantity, SUM(sale_amount) AS total_sales
  2  FROM sales
  3  GROUP BY category;

CATEGORY                                           TOTAL_QUANTITY TOTAL_SALES
-------------------------------------------------- -------------- -----------
Fruit                                                         140       167.2
Grocery                                                        25         105
Stationery                                                     75         100
```

Source code:

SELECT product_name, category, quantity_sold, sale_amount
FROM sales
ORDER BY sale_amount DESC;

Output:

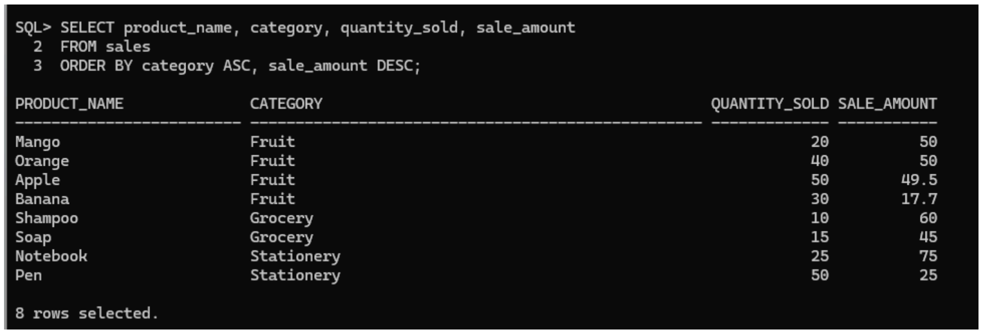


```
SQL> SELECT product_name, category, quantity_sold, sale_amount
  2  FROM sales
  3  ORDER BY sale_amount DESC;

PRODUCT_NAME              CATEGORY                                           QUANTITY_SOLD SALE_AMOUNT
------------------------- -------------------------------------------------- ------------- -----------
Notebook                  Stationery                                                    25          75
Shampoo                   Grocery                                                       10          60
Orange                    Fruit                                                         40          50
Mango                     Fruit                                                         20          50
Apple                     Fruit                                                         50        49.5
Soap                      Grocery                                                       15          45
Pen                       Stationery                                                    50          25
Banana                    Fruit                                                         30        17.7

8 rows selected.
```

Source code:

SELECT product_name, category, quantity_sold, sale_amount
FROM sales
ORDER BY category ASC, sale_amount DESC;

Output:

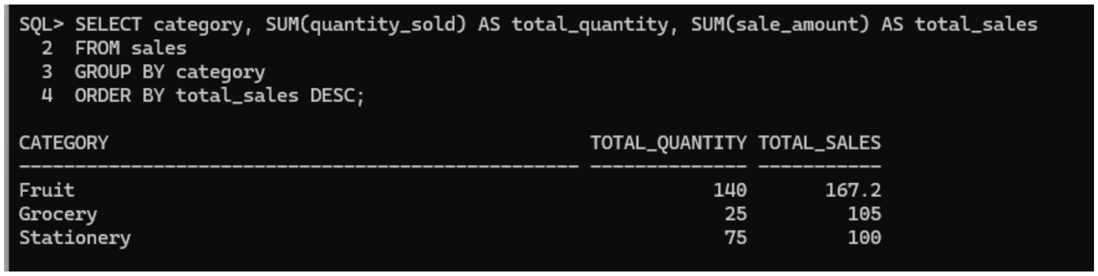


```
SQL> SELECT product_name, category, quantity_sold, sale_amount
  2  FROM sales
  3  ORDER BY category ASC, sale_amount DESC;

PRODUCT_NAME              CATEGORY                                           QUANTITY_SOLD SALE_AMOUNT
------------------------- -------------------------------------------------- ------------- -----------
Mango                     Fruit                                                         20          50
Orange                    Fruit                                                         40          50
Apple                     Fruit                                                         50        49.5
Banana                    Fruit                                                         30        17.7
Shampoo                   Grocery                                                       10          60
Soap                      Grocery                                                       15          45
Notebook                  Stationery                                                    25          75
Pen                       Stationery                                                    50          25

8 rows selected.
```

Source code:

```
SELECT category, SUM(quantity_sold) AS total_quantity, SUM(sale_amount) AS
total_sales
FROM sales
GROUP BY category
ORDER BY total_sales DESC;
```

Output:



```
SQL> SELECT category, SUM(quantity_sold) AS total_quantity, SUM(sale_amount) AS total_sales
  2  FROM sales
  3  GROUP BY category
  4  ORDER BY total_sales DESC;

CATEGORY                                           TOTAL_QUANTITY TOTAL_SALES
-------------------------------------------------- -------------- -----------
Fruit                                                         140       167.2
Grocery                                                        25         105
Stationery                                                     75         100
```

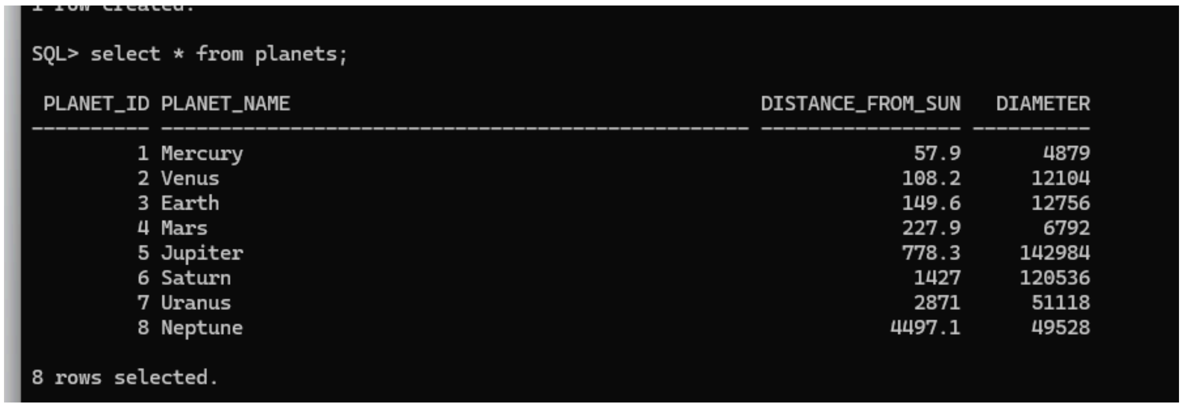
EXPERIMENT 5

1. **Ascending: ASC** sorts query results from lowest to highest.
2. **Descending: DESC** sorts from highest to lowest based on the specified column.

Source code:

```
CREATE TABLE planets (
   planet_id INT PRIMARY KEY,
   planet_name VARCHAR2(50) NOT NULL,
   distance_from_sun NUMBER(10,2),
   diameter NUMBER(10,2));
INSERT INTO planets (planet_id, planet_name, distance_from_sun, diameter)
VALUES (1, 'Mercury', 57.9, 4879);
INSERT INTO planets (planet_id, planet_name, distance_from_sun, diameter)
VALUES (2, 'Venus', 108.2, 12104);
INSERT INTO planets (planet_id, planet_name, distance_from_sun, diameter)
VALUES (3, 'Earth', 149.6, 12756);
INSERT INTO planets (planet_id, planet_name, distance_from_sun, diameter)
VALUES (4, 'Mars', 227.9, 6792);
INSERT INTO planets (planet_id, planet_name, distance_from_sun, diameter)
VALUES (5, 'Jupiter', 778.3, 142984);
INSERT INTO planets (planet_id, planet_name, distance_from_sun, diameter)
VALUES (6, 'Saturn', 1427.0, 120536);
INSERT INTO planets (planet_id, planet_name, distance_from_sun, diameter)
VALUES (7, 'Uranus', 2871.0, 51118);
INSERT INTO planets (planet_id, planet_name, distance_from_sun, diameter)
VALUES (8, 'Neptune', 4497.1, 49528);
```

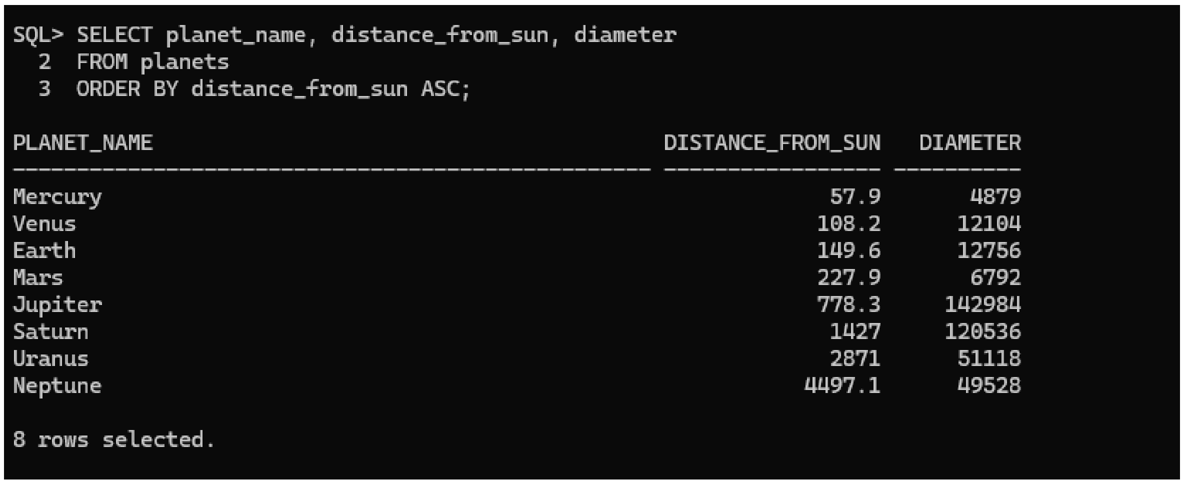
Output:



```
SQL> select * from planets;

 PLANET_ID PLANET_NAME                                        DISTANCE_FROM_SUN   DIAMETER
---------- -------------------------------------------------- ----------------- ----------
         1 Mercury                                                         57.9       4879
         2 Venus                                                          108.2      12104
         3 Earth                                                          149.6      12756
         4 Mars                                                           227.9       6792
         5 Jupiter                                                        778.3     142984
         6 Saturn                                                          1427     120536
         7 Uranus                                                          2871      51118
         8 Neptune                                                       4497.1      49528

8 rows selected.
```

Source code:
Select * from planets;
SELECT planet_name, distance_from_sun, diameter
FROM planets
ORDER BY distance_from_sun ASC;

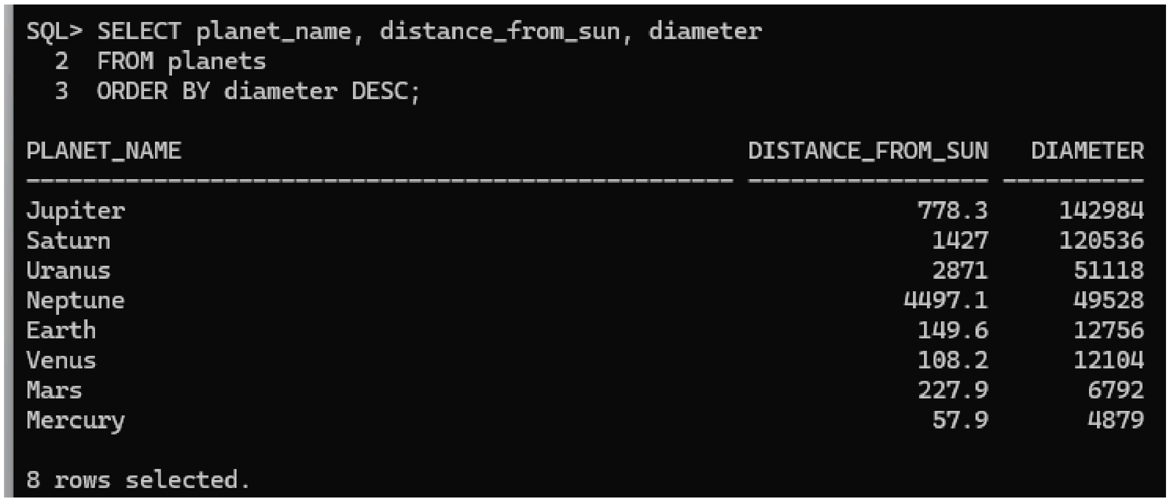
Output:



```
SQL> SELECT planet_name, distance_from_sun, diameter
  2  FROM planets
  3  ORDER BY distance_from_sun ASC;

PLANET_NAME                                        DISTANCE_FROM_SUN   DIAMETER
-------------------------------------------------- ----------------- ----------
Mercury                                                         57.9       4879
Venus                                                          108.2      12104
Earth                                                          149.6      12756
Mars                                                           227.9       6792
Jupiter                                                        778.3     142984
Saturn                                                          1427     120536
Uranus                                                          2871      51118
Neptune                                                       4497.1      49528

8 rows selected.
```

Source code:
SELECT planet_name, distance_from_sun, diameter
FROM planets
ORDER BY diameter DESC;

Output:



```
SQL> SELECT planet_name, distance_from_sun, diameter
  2  FROM planets
  3  ORDER BY diameter DESC;

PLANET_NAME                                        DISTANCE_FROM_SUN   DIAMETER
-------------------------------------------------- ----------------- ----------
Jupiter                                                        778.3     142984
Saturn                                                          1427     120536
Uranus                                                          2871      51118
Neptune                                                       4497.1      49528
Earth                                                          149.6      12756
Venus                                                          108.2      12104
Mars                                                           227.9       6792
Mercury                                                         57.9       4879

8 rows selected.
```

EXPERIMENT 6

SQL Operators

1. **LIKE**: Filters results based on pattern matching using % (any characters) and _ (single character).
2. **BETWEEN**: Checks if a value lies within a specified inclusive range.
3. **OR**: Returns results if **any** of the given conditions are true.

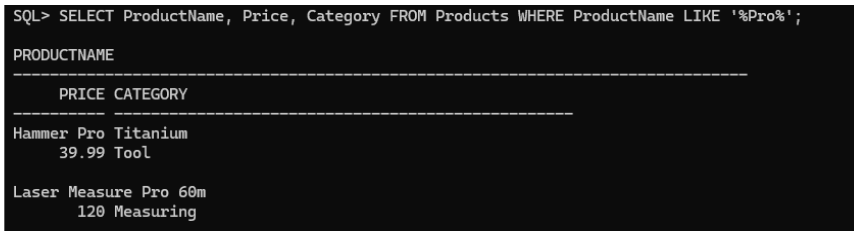
Source code:

```
CREATE TABLE Suppliers(SupplierID INT PRIMARY KEY,SupplierName
VARCHAR(100) NOT NULL,City VARCHAR(50));
CREATE SEQUENCE Suppliers_seq START WITH 1 INCREMENT BY 1;
CREATE OR REPLACE TRIGGER Suppliers_on_insert
BEFORE INSERT ON Suppliers
FOR EACH ROW
BEGIN
  SELECT Suppliers_seq.nextval
  INTO :new.SupplierID
  FROM dual;
END;
CREATE TABLE Products (ProductID INT PRIMARY KEY,ProductName
VARCHAR(100) NOT NULL,Category VARCHAR(50),Price NUMBER(10,
2),StockQuantity INT,SupplierID INT,FOREIGN KEY(SupplierID) REFERENCES
Suppliers(SupplierID));
CREATE SEQUENCE Products_seq
START WITH 1
INCREMENT BY 1;
CREATE OR REPLACE TRIGGER Products_on_insert
BEFORE INSERT ON Products
FOR EACH ROW
BEGIN
  SELECT Products_seq.nextval
  INTO :new.ProductID
  FROM dual;
END;
INSERT INTO Suppliers (SupplierName, City) VALUES ('ToolMaster Pro', 'New
York');
INSERT INTO Suppliers (SupplierName, City) VALUES ('Eastern Lumber Co.',
'Boston');
INSERT INTO Suppliers (SupplierName, City) VALUES ('Quick Fasteners Ltd.',
'Miami');
```

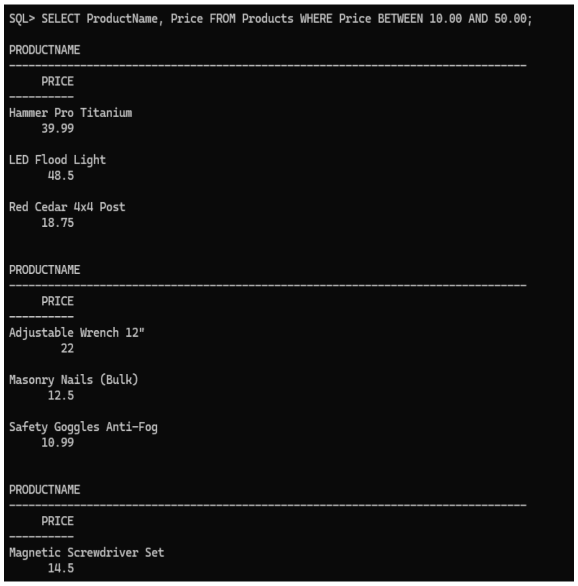
```sql
INSERT INTO Suppliers (SupplierName, City) VALUES ('PowerHouse Electric',
'Chicago');
INSERT INTO Suppliers (SupplierName, City) VALUES ('The Metal Works',
'Seattle');
INSERT INTO Suppliers (SupplierName, City) VALUES ('Apex Safety Gear',
'Dallas');
INSERT INTO Suppliers (SupplierName, City) VALUES ('Prime Plumbing
Supply', 'Phoenix');
INSERT INTO Suppliers (SupplierName, City) VALUES ('Global Adhesives',
'Denver');
INSERT INTO Suppliers (SupplierName, City) VALUES ('Brick & Mortar Co.',
'Atlanta');
INSERT INTO Suppliers (SupplierName, City) VALUES ('Precision Measuring',
'Houston');
INSERT INTO Suppliers (SupplierName, City) VALUES ('Super Wrench Group',
'New York');
INSERT INTO Suppliers (SupplierName, City) VALUES ('Cedar Creek Wood',
'Boston');
INSERT INTO Suppliers (SupplierName, City) VALUES ('Volta Electrical',
'Miami');
INSERT INTO Suppliers (SupplierName, City) VALUES ('Ironclad Hardware',
'Chicago');
INSERT INTO Suppliers (SupplierName, City) VALUES ('AquaFlow Plumbing',
'Seattle');
INSERT INTO Suppliers (SupplierName, City) VALUES ('SureGrip Fasteners',
'Dallas');
INSERT INTO Suppliers (SupplierName, City) VALUES ('Bright Light Solutions',
'Phoenix');
INSERT INTO Suppliers (SupplierName, City) VALUES ('Durable Paint Co.',
'Denver');
INSERT INTO Suppliers (SupplierName, City) VALUES ('Contractor Essentials',
'Atlanta');
INSERT INTO Suppliers (SupplierName, City) VALUES ('Home Fix Depot',
'Houston');
INSERT INTO Products (ProductName, Category, Price, StockQuantity,
SupplierID) VALUES ('Hammer Pro Titanium', 'Tool', 39.99, 150, 1);
INSERT INTO Products (ProductName, Category, Price, StockQuantity,
SupplierID) VALUES ('1/4 inch Hex Bolts', 'Fastener', 5.25, 500, 3);
INSERT INTO Products (ProductName, Category, Price, StockQuantity,
SupplierID) VALUES ('LED Flood Light', 'Electrical', 48.50, 120, 4);
INSERT INTO Products (ProductName, Category, Price, StockQuantity,
SupplierID) VALUES ('Red Cedar 4x4 Post', 'Wood', 18.75, 80, 12);
```

```
INSERT INTO Products (ProductName, Category, Price, StockQuantity,
SupplierID) VALUES ('Adjustable Wrench 12"', 'Tool', 22.00, 120, 11);
INSERT INTO Products (ProductName, Category, Price, StockQuantity,
SupplierID) VALUES ('Hex Head Screws Zinc', 'Fastener', 6.00, 450, 16);
INSERT INTO Products (ProductName, Category, Price, StockQuantity,
SupplierID) VALUES ('Copper Wire Spool 12g', 'Electrical', 75.99, 100, 13);
INSERT INTO Products (ProductName, Category, Price, StockQuantity,
SupplierID) VALUES ('Oak Plywood 3/4"', 'Wood', 65.00, 50, 12);
INSERT INTO Products (ProductName, Category, Price, StockQuantity,
SupplierID) VALUES ('Drill Bit Set (50pcs)', 'Tool', 59.99, 80, 1);
INSERT INTO Products (ProductName, Category, Price, StockQuantity,
SupplierID) VALUES ('Masonry Nails (Bulk)', 'Fastener', 12.50, 600, 3);
INSERT INTO Products (ProductName, Category, Price, StockQuantity,
SupplierID) VALUES ('White Outlet Cover', 'Electrical', 2.99, 750, 13);
INSERT INTO Products (ProductName, Category, Price, StockQuantity,
SupplierID) VALUES ('Pressure Treated Pine', 'Wood', 6.80, 200, 5);
INSERT INTO Products (ProductName, Category, Price, StockQuantity,
SupplierID) VALUES ('Safety Goggles Anti-Fog', 'Safety', 10.99, 250, 6);
INSERT INTO Products (ProductName, Category, Price, StockQuantity,
SupplierID) VALUES ('Aluminum Sheet Metal', 'Metal', 88.00, 40, 5);
INSERT INTO Products (ProductName, Category, Price, StockQuantity,
SupplierID) VALUES ('Magnetic Screwdriver Set', 'Tool', 14.50, 180, 11);
INSERT INTO Products (ProductName, Category, Price, StockQuantity,
SupplierID) VALUES ('Teflon Tape 1/2"', 'Plumbing', 1.99, 900, 7);
INSERT INTO Products (ProductName, Category, Price, StockQuantity,
SupplierID) VALUES ('PVC Pipe Connector 1"', 'Plumbing', 4.50, 320, 15);
INSERT INTO Products (ProductName, Category, Price, StockQuantity,
SupplierID) VALUES ('Heavy Duty Glue Stick', 'Adhesive', 3.75, 400, 8);
INSERT INTO Products (ProductName, Category, Price, StockQuantity,
SupplierID) VALUES ('Laser Measure Pro 60m', 'Measuring', 120.00, 30, 10);
INSERT INTO Products (ProductName, Category, Price, StockQuantity,
SupplierID) VALUES ('Galvanized Steel Pipe', 'Plumbing', 55.00, 60, 15);
SELECT ProductName, Price, Category FROM Products WHERE ProductName
LIKE '%Pro%';
SELECT ProductName, Price FROM Products WHERE Price BETWEEN 10.00
AND 50.00;
SELECT SupplierName, City FROM Suppliers WHERE City = 'New York' OR City
= 'Chicago';
```

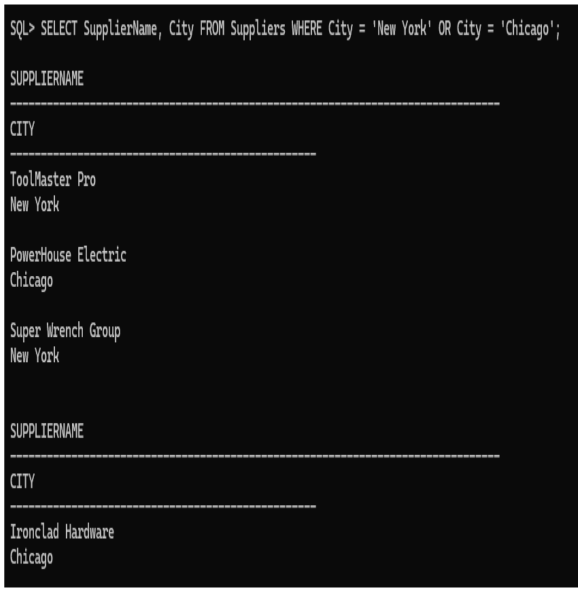
Output:



```
SQL> SELECT ProductName, Price, Category FROM Products WHERE ProductName LIKE '%Pro%';

PRODUCTNAME
--------------------------------------------------------------------------------
     PRICE CATEGORY
---------- --------------------------------------------------
Hammer Pro Titanium
     39.99 Tool

Laser Measure Pro 60m
       120 Measuring
```



```
SQL> SELECT ProductName, Price FROM Products WHERE Price BETWEEN 10.00 AND 50.00;

PRODUCTNAME
--------------------------------------------------------------------------------
     PRICE
----------
Hammer Pro Titanium
     39.99

LED Flood Light
      48.5

Red Cedar 4x4 Post
     18.75


PRODUCTNAME
--------------------------------------------------------------------------------
     PRICE
----------
Adjustable Wrench 12"
        22

Masonry Nails (Bulk)
      12.5

Safety Goggles Anti-Fog
     10.99


PRODUCTNAME
--------------------------------------------------------------------------------
     PRICE
----------
Magnetic Screwdriver Set
      14.5
```



```
SQL> SELECT SupplierName, City FROM Suppliers WHERE City = 'New York' OR City = 'Chicago';

SUPPLIERNAME
--------------------------------------------------------------------------------
CITY
--------------------------------------------------
ToolMaster Pro
New York

PowerHouse Electric
Chicago

Super Wrench Group
New York


SUPPLIERNAME
--------------------------------------------------------------------------------
CITY
--------------------------------------------------
Ironclad Hardware
Chicago
```

EXPERIMENT 7

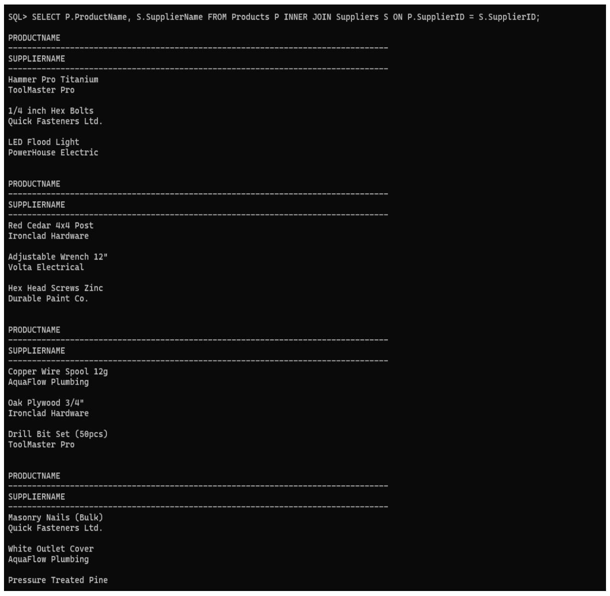
SQL Joins

1. **INNER JOIN**: Returns rows with matching values in both tables.
2. **LEFT JOIN**: Returns all rows from the left table and matched rows from the right.
3. **RIGHT JOIN**: Returns all rows from the right table and matched rows from the left.
4. **OUTER JOIN**: Returns all rows when there is a match in one of the tables.
5. **LEFT JOIN excluding INNER JOIN**: Returns unmatched rows from the left table only.
6. **RIGHT JOIN excluding INNER JOIN**: Returns unmatched rows from the right table only.
7. **OUTER JOIN excluding INNER JOIN**: Returns unmatched rows from both tables.

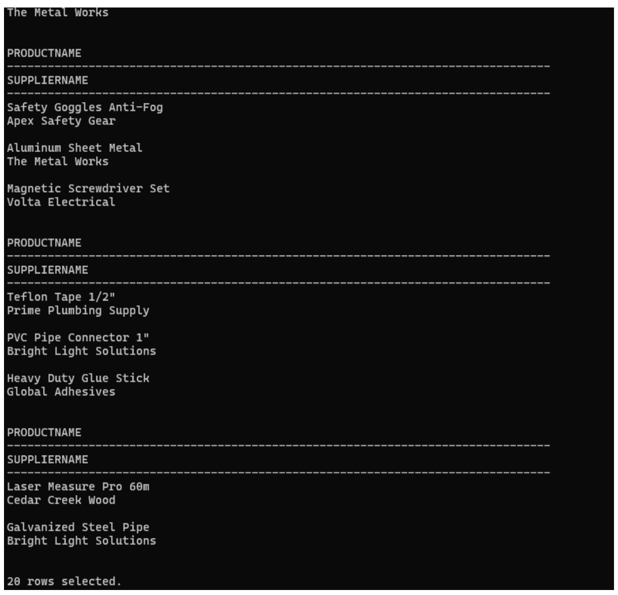
Source code:

SELECT P.ProductName, S.SupplierName FROM Products P INNER JOIN Suppliers S ON P.SupplierID = S.SupplierID;

Output:



```
SQL> SELECT P.ProductName, S.SupplierName FROM Products P INNER JOIN Suppliers S ON P.SupplierID = S.SupplierID;

PRODUCTNAME
--------------------------------------------------------------------------------
SUPPLIERNAME
--------------------------------------------------------------------------------
Hammer Pro Titanium
ToolMaster Pro

1/4 inch Hex Bolts
Quick Fasteners Ltd.

LED Flood Light
PowerHouse Electric


PRODUCTNAME
--------------------------------------------------------------------------------
SUPPLIERNAME
--------------------------------------------------------------------------------
Red Cedar 4x4 Post
Ironclad Hardware

Adjustable Wrench 12"
Volta Electrical

Hex Head Screws Zinc
Durable Paint Co.


PRODUCTNAME
--------------------------------------------------------------------------------
SUPPLIERNAME
--------------------------------------------------------------------------------
Copper Wire Spool 12g
AquaFlow Plumbing

Oak Plywood 3/4"
Ironclad Hardware

Drill Bit Set (50pcs)
ToolMaster Pro


PRODUCTNAME
--------------------------------------------------------------------------------
SUPPLIERNAME
--------------------------------------------------------------------------------
Masonry Nails (Bulk)
Quick Fasteners Ltd.

White Outlet Cover
AquaFlow Plumbing

Pressure Treated Pine
```

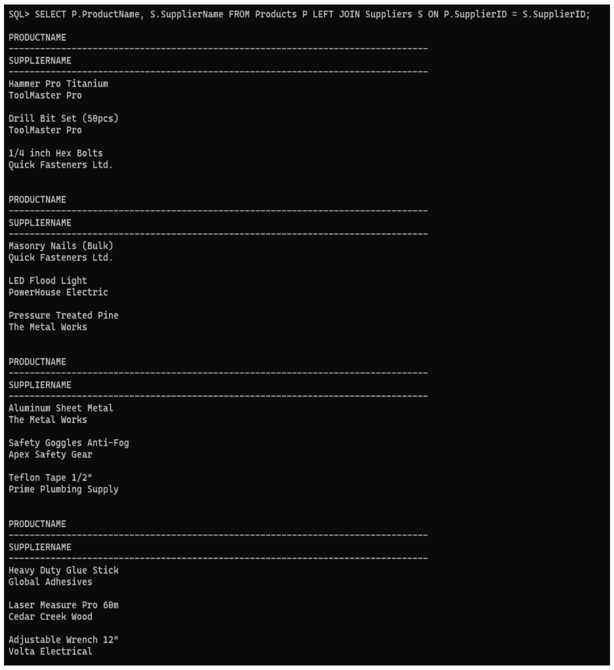


```
The Metal Works


PRODUCTNAME
--------------------------------------------------------------------------------
SUPPLIERNAME
--------------------------------------------------------------------------------
Safety Goggles Anti-Fog
Apex Safety Gear

Aluminum Sheet Metal
The Metal Works

Magnetic Screwdriver Set
Volta Electrical


PRODUCTNAME
--------------------------------------------------------------------------------
SUPPLIERNAME
--------------------------------------------------------------------------------
Teflon Tape 1/2"
Prime Plumbing Supply

PVC Pipe Connector 1"
Bright Light Solutions

Heavy Duty Glue Stick
Global Adhesives


PRODUCTNAME
--------------------------------------------------------------------------------
SUPPLIERNAME
--------------------------------------------------------------------------------
Laser Measure Pro 60m
Cedar Creek Wood

Galvanized Steel Pipe
Bright Light Solutions


20 rows selected.
```

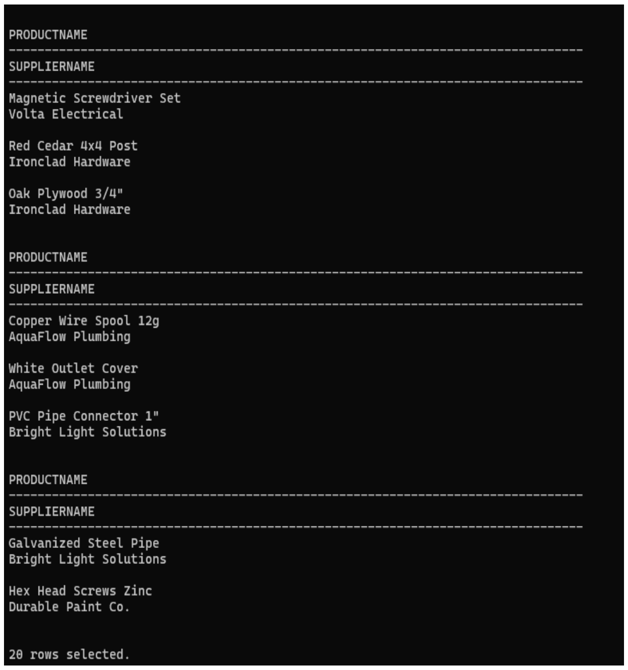
Source code:

SELECT P.ProductName, S.SupplierName FROM Products P LEFT JOIN Suppliers S ON P.SupplierID = S.SupplierID;

Output:



```
SQL> SELECT P.ProductName, S.SupplierName FROM Products P LEFT JOIN Suppliers S ON P.SupplierID = S.SupplierID;

PRODUCTNAME
--------------------------------------------------------------------------------
SUPPLIERNAME
--------------------------------------------------------------------------------
Hammer Pro Titanium
ToolMaster Pro

Drill Bit Set (50pcs)
ToolMaster Pro

1/4 inch Hex Bolts
Quick Fasteners Ltd.


PRODUCTNAME
--------------------------------------------------------------------------------
SUPPLIERNAME
--------------------------------------------------------------------------------
Masonry Nails (Bulk)
Quick Fasteners Ltd.

LED Flood Light
PowerHouse Electric

Pressure Treated Pine
The Metal Works


PRODUCTNAME
--------------------------------------------------------------------------------
SUPPLIERNAME
--------------------------------------------------------------------------------
Aluminum Sheet Metal
The Metal Works

Safety Goggles Anti-Fog
Apex Safety Gear

Teflon Tape 1/2"
Prime Plumbing Supply


PRODUCTNAME
--------------------------------------------------------------------------------
SUPPLIERNAME
--------------------------------------------------------------------------------
Heavy Duty Glue Stick
Global Adhesives

Laser Measure Pro 60m
Cedar Creek Wood

Adjustable Wrench 12"
Volta Electrical
```

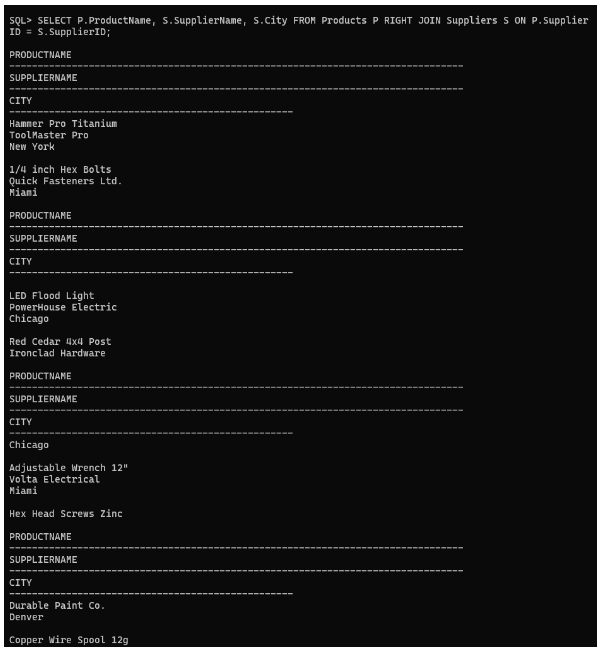


```
PRODUCTNAME
--------------------------------------------------------------------------------
SUPPLIERNAME
--------------------------------------------------------------------------------
Magnetic Screwdriver Set
Volta Electrical

Red Cedar 4x4 Post
Ironclad Hardware

Oak Plywood 3/4"
Ironclad Hardware


PRODUCTNAME
--------------------------------------------------------------------------------
SUPPLIERNAME
--------------------------------------------------------------------------------
Copper Wire Spool 12g
AquaFlow Plumbing

White Outlet Cover
AquaFlow Plumbing

PVC Pipe Connector 1"
Bright Light Solutions


PRODUCTNAME
--------------------------------------------------------------------------------
SUPPLIERNAME
--------------------------------------------------------------------------------
Galvanized Steel Pipe
Bright Light Solutions

Hex Head Screws Zinc
Durable Paint Co.


20 rows selected.
```

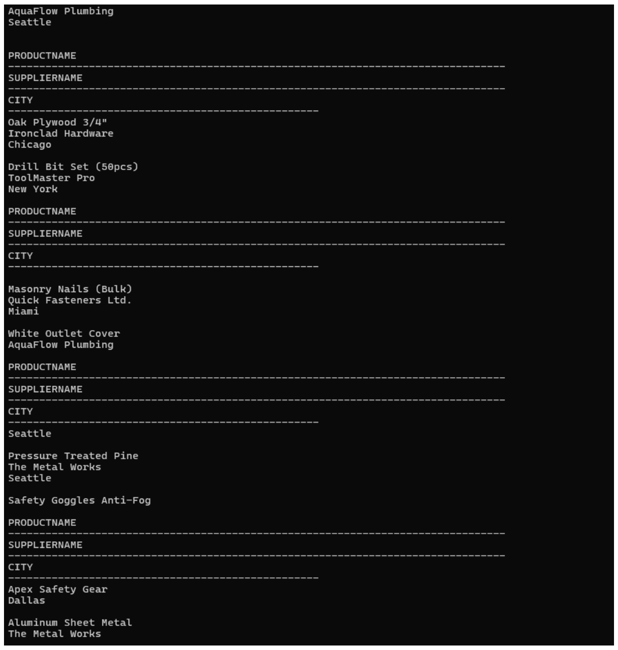
Source code:

SELECT P.ProductName, S.SupplierName, S.City FROM Products P RIGHT JOIN Suppliers S ON P.SupplierID = S.SupplierID;

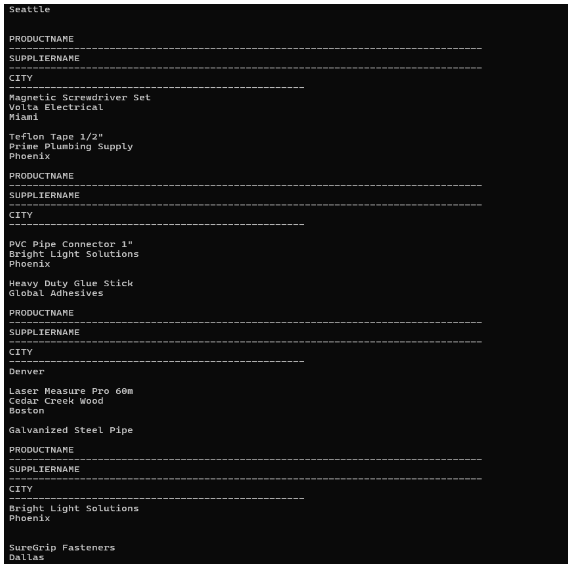
Output:



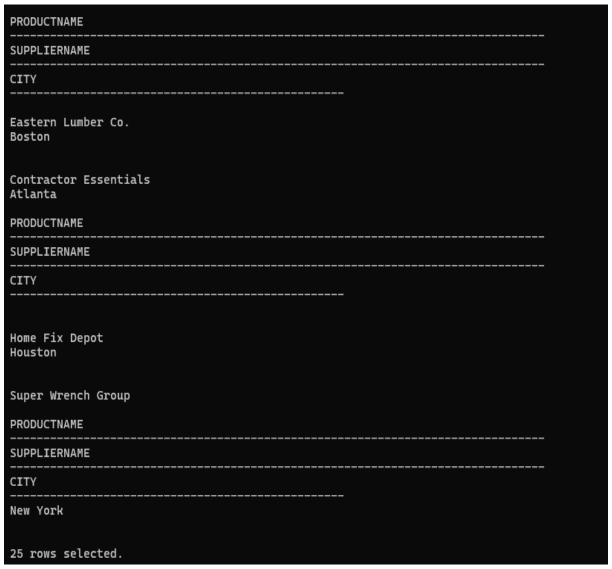
```
SQL> SELECT P.ProductName, S.SupplierName, S.City FROM Products P RIGHT JOIN Suppliers S ON P.Supplier
ID = S.SupplierID;

PRODUCTNAME
--------------------------------------------------------------------------------
SUPPLIERNAME
--------------------------------------------------------------------------------
CITY
--------------------------------------------------
Hammer Pro Titanium
ToolMaster Pro
New York

1/4 inch Hex Bolts
Quick Fasteners Ltd.
Miami

PRODUCTNAME
--------------------------------------------------------------------------------
SUPPLIERNAME
--------------------------------------------------------------------------------
CITY
--------------------------------------------------

LED Flood Light
PowerHouse Electric
Chicago

Red Cedar 4x4 Post
Ironclad Hardware

PRODUCTNAME
--------------------------------------------------------------------------------
SUPPLIERNAME
--------------------------------------------------------------------------------
CITY
--------------------------------------------------
Chicago

Adjustable Wrench 12"
Volta Electrical
Miami

Hex Head Screws Zinc

PRODUCTNAME
--------------------------------------------------------------------------------
SUPPLIERNAME
--------------------------------------------------------------------------------
CITY
--------------------------------------------------
Durable Paint Co.
Denver

Copper Wire Spool 12g
```



```
AquaFlow Plumbing
Seattle


PRODUCTNAME
--------------------------------------------------------------------------------
SUPPLIERNAME
--------------------------------------------------------------------------------
CITY
--------------------------------------------------
Oak Plywood 3/4"
Ironclad Hardware
Chicago

Drill Bit Set (50pcs)
ToolMaster Pro
New York

PRODUCTNAME
--------------------------------------------------------------------------------
SUPPLIERNAME
--------------------------------------------------------------------------------
CITY
--------------------------------------------------

Masonry Nails (Bulk)
Quick Fasteners Ltd.
Miami

White Outlet Cover
AquaFlow Plumbing

PRODUCTNAME
--------------------------------------------------------------------------------
SUPPLIERNAME
--------------------------------------------------------------------------------
CITY
--------------------------------------------------
Seattle

Pressure Treated Pine
The Metal Works
Seattle

Safety Goggles Anti-Fog

PRODUCTNAME
--------------------------------------------------------------------------------
SUPPLIERNAME
--------------------------------------------------------------------------------
CITY
--------------------------------------------------
Apex Safety Gear
Dallas

Aluminum Sheet Metal
The Metal Works
```



```
Seattle


PRODUCTNAME
--------------------------------------------------------------------------------
SUPPLIERNAME
--------------------------------------------------------------------------------
CITY
--------------------------------------------------
Magnetic Screwdriver Set
Volta Electrical
Miami

Teflon Tape 1/2"
Prime Plumbing Supply
Phoenix

PRODUCTNAME
--------------------------------------------------------------------------------
SUPPLIERNAME
--------------------------------------------------------------------------------
CITY
--------------------------------------------------

PVC Pipe Connector 1"
Bright Light Solutions
Phoenix

Heavy Duty Glue Stick
Global Adhesives

PRODUCTNAME
--------------------------------------------------------------------------------
SUPPLIERNAME
--------------------------------------------------------------------------------
CITY
--------------------------------------------------
Denver

Laser Measure Pro 60m
Cedar Creek Wood
Boston

Galvanized Steel Pipe

PRODUCTNAME
--------------------------------------------------------------------------------
SUPPLIERNAME
--------------------------------------------------------------------------------
CITY
--------------------------------------------------
Bright Light Solutions
Phoenix


SureGrip Fasteners
Dallas
```

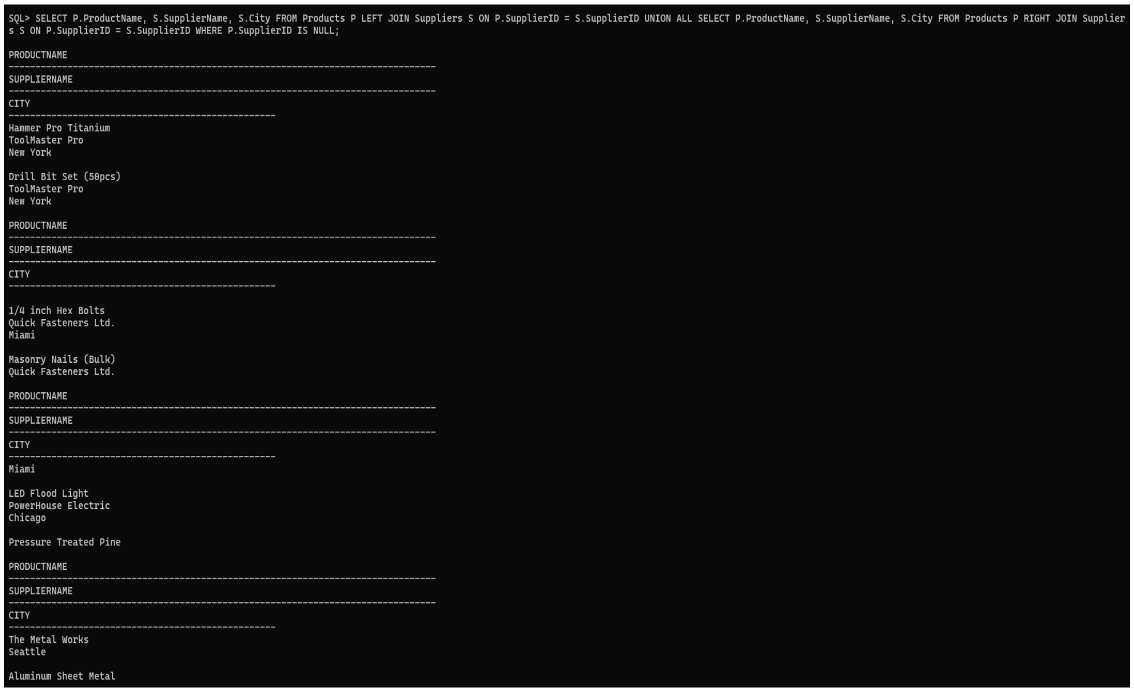


```
PRODUCTNAME
--------------------------------------------------------------------------------
SUPPLIERNAME
--------------------------------------------------------------------------------
CITY
--------------------------------------------------

Eastern Lumber Co.
Boston


Contractor Essentials
Atlanta

PRODUCTNAME
--------------------------------------------------------------------------------
SUPPLIERNAME
--------------------------------------------------------------------------------
CITY
--------------------------------------------------



Home Fix Depot
Houston


Super Wrench Group

PRODUCTNAME
--------------------------------------------------------------------------------
SUPPLIERNAME
--------------------------------------------------------------------------------
CITY
--------------------------------------------------
New York


25 rows selected.
```

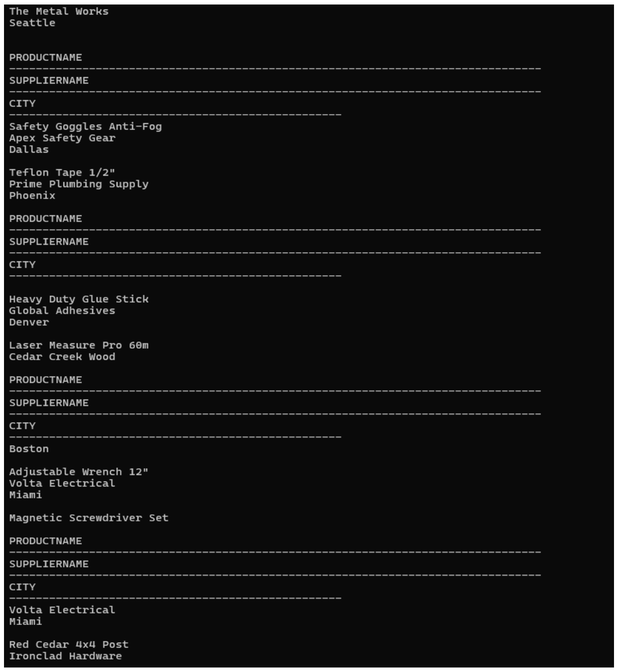
Source code:

SELECT P.ProductName, S.SupplierName, S.City FROM Products P LEFT JOIN Suppliers S ON P.SupplierID = S.SupplierID UNION ALL SELECT P.ProductName, S.SupplierName, S.City FROM Products P RIGHT JOIN Suppliers S ON P.SupplierID = S.SupplierID WHERE P.SupplierID IS NULL;

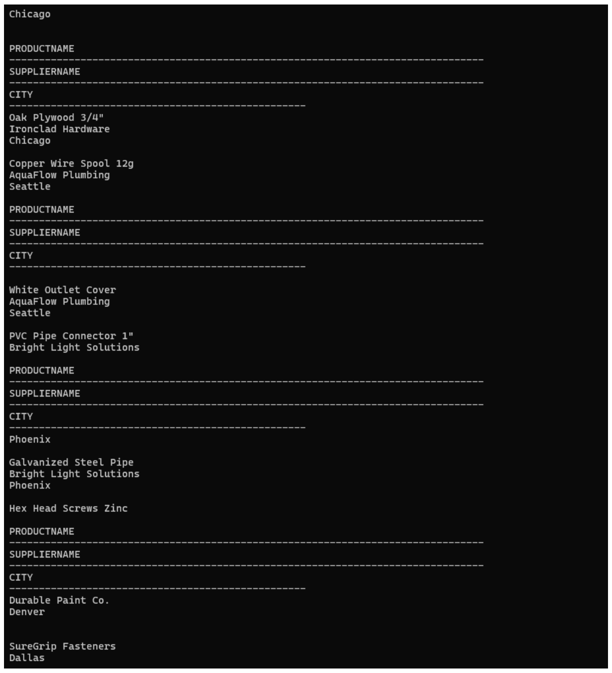
Output:



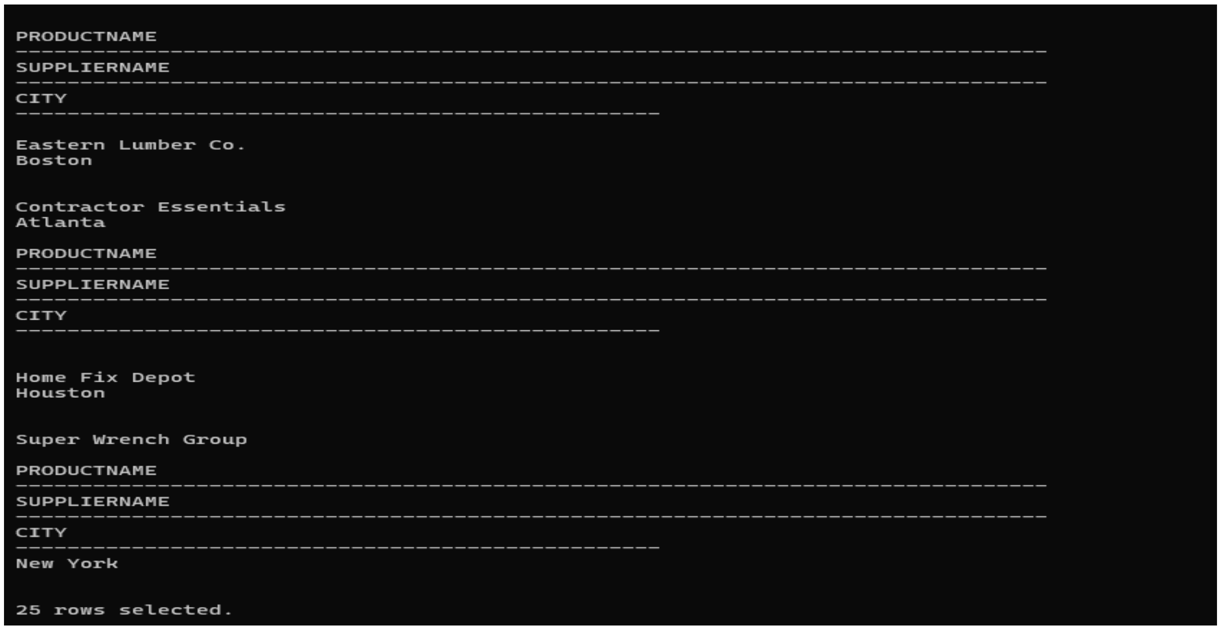
```
SQL> SELECT P.ProductName, S.SupplierName, S.City FROM Products P LEFT JOIN Suppliers S ON P.SupplierID = S.SupplierID UNION ALL SELECT P.ProductName, S.SupplierName, S.City FROM Products P RIGHT JOIN Supplier
s S ON P.SupplierID = S.SupplierID WHERE P.SupplierID IS NULL;

PRODUCTNAME
--------------------------------------------------------------------------------
SUPPLIERNAME
--------------------------------------------------------------------------------
CITY
--------------------------------------------------
Hammer Pro Titanium
ToolMaster Pro
New York

Drill Bit Set (50pcs)
ToolMaster Pro
New York

PRODUCTNAME
--------------------------------------------------------------------------------
SUPPLIERNAME
--------------------------------------------------------------------------------
CITY
--------------------------------------------------

1/4 inch Hex Bolts
Quick Fasteners Ltd.
Miami

Masonry Nails (Bulk)
Quick Fasteners Ltd.

PRODUCTNAME
--------------------------------------------------------------------------------
SUPPLIERNAME
--------------------------------------------------------------------------------
CITY
--------------------------------------------------
Miami

LED Flood Light
PowerHouse Electric
Chicago

Pressure Treated Pine

PRODUCTNAME
--------------------------------------------------------------------------------
SUPPLIERNAME
--------------------------------------------------------------------------------
CITY
--------------------------------------------------
The Metal Works
Seattle

Aluminum Sheet Metal
```



```
The Metal Works
Seattle


PRODUCTNAME
--------------------------------------------------------------------------------
SUPPLIERNAME
--------------------------------------------------------------------------------
CITY
--------------------------------------------------
Safety Goggles Anti-Fog
Apex Safety Gear
Dallas

Teflon Tape 1/2"
Prime Plumbing Supply
Phoenix

PRODUCTNAME
--------------------------------------------------------------------------------
SUPPLIERNAME
--------------------------------------------------------------------------------
CITY
--------------------------------------------------


Heavy Duty Glue Stick
Global Adhesives
Denver

Laser Measure Pro 60m
Cedar Creek Wood

PRODUCTNAME
--------------------------------------------------------------------------------
SUPPLIERNAME
--------------------------------------------------------------------------------
CITY
--------------------------------------------------
Boston

Adjustable Wrench 12"
Volta Electrical
Miami

Magnetic Screwdriver Set

PRODUCTNAME
--------------------------------------------------------------------------------
SUPPLIERNAME
--------------------------------------------------------------------------------
CITY
--------------------------------------------------
Volta Electrical
Miami

Red Cedar 4x4 Post
Ironclad Hardware
```



```
Chicago


PRODUCTNAME
--------------------------------------------------------------------------------
SUPPLIERNAME
--------------------------------------------------------------------------------
CITY
--------------------------------------------------
Oak Plywood 3/4"
Ironclad Hardware
Chicago

Copper Wire Spool 12g
AquaFlow Plumbing
Seattle

PRODUCTNAME
--------------------------------------------------------------------------------
SUPPLIERNAME
--------------------------------------------------------------------------------
CITY
--------------------------------------------------


White Outlet Cover
AquaFlow Plumbing
Seattle

PVC Pipe Connector 1"
Bright Light Solutions

PRODUCTNAME
--------------------------------------------------------------------------------
SUPPLIERNAME
--------------------------------------------------------------------------------
CITY
--------------------------------------------------
Phoenix

Galvanized Steel Pipe
Bright Light Solutions
Phoenix

Hex Head Screws Zinc

PRODUCTNAME
--------------------------------------------------------------------------------
SUPPLIERNAME
--------------------------------------------------------------------------------
CITY
--------------------------------------------------
Durable Paint Co.
Denver


SureGrip Fasteners
Dallas
```

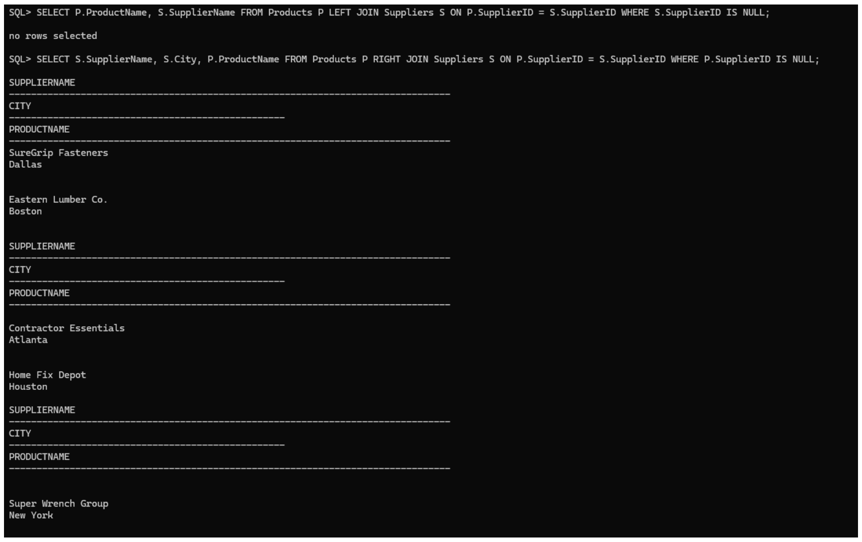


```
PRODUCTNAME
--------------------------------------------------------------------------------
SUPPLIERNAME
--------------------------------------------------------------------------------
CITY
--------------------------------------------------

Eastern Lumber Co.
Boston


Contractor Essentials
Atlanta

PRODUCTNAME
--------------------------------------------------------------------------------
SUPPLIERNAME
--------------------------------------------------------------------------------
CITY
--------------------------------------------------


Home Fix Depot
Houston


Super Wrench Group

PRODUCTNAME
--------------------------------------------------------------------------------
SUPPLIERNAME
--------------------------------------------------------------------------------
CITY
--------------------------------------------------
New York


25 rows selected.
```

Source code:

SELECT P.ProductName, S.SupplierName FROM Products P LEFT JOIN Suppliers S ON P.SupplierID = S.SupplierID WHERE S.SupplierID IS NULL;
SELECT S.SupplierName, S.City, P.ProductName FROM Products P RIGHT JOIN Suppliers S ON P.SupplierID = S.SupplierID WHERE P.SupplierID IS NULL;

Output:

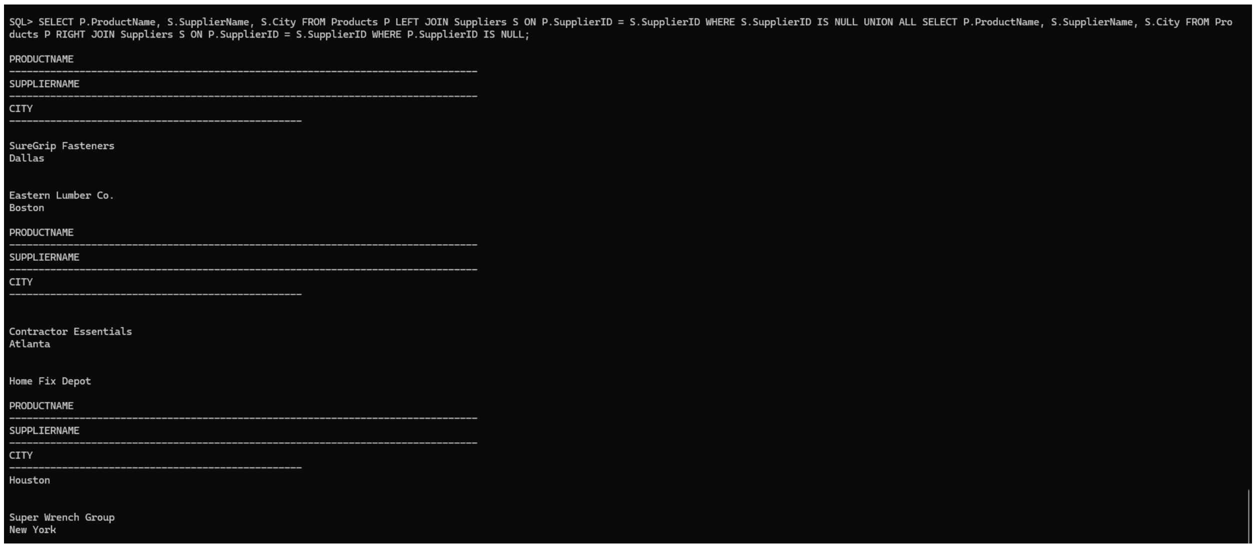


```
SQL> SELECT P.ProductName, S.SupplierName FROM Products P LEFT JOIN Suppliers S ON P.SupplierID = S.SupplierID WHERE S.SupplierID IS NULL;

no rows selected

SQL> SELECT S.SupplierName, S.City, P.ProductName FROM Products P RIGHT JOIN Suppliers S ON P.SupplierID = S.SupplierID WHERE P.SupplierID IS NULL;

SUPPLIERNAME
--------------------------------------------------------------------------------
CITY
--------------------------------------------------
PRODUCTNAME
--------------------------------------------------------------------------------
SureGrip Fasteners
Dallas


Eastern Lumber Co.
Boston


SUPPLIERNAME
--------------------------------------------------------------------------------
CITY
--------------------------------------------------
PRODUCTNAME
--------------------------------------------------------------------------------

Contractor Essentials
Atlanta


Home Fix Depot
Houston

SUPPLIERNAME
--------------------------------------------------------------------------------
CITY
--------------------------------------------------
PRODUCTNAME
--------------------------------------------------------------------------------


Super Wrench Group
New York
```

Source code:

SELECT P.ProductName, S.SupplierName, S.City FROM Products P LEFT JOIN Suppliers S ON P.SupplierID = S.SupplierID WHERE S.SupplierID IS NULL UNION ALL SELECT P.ProductName, S.SupplierName, S.City FROM Products P RIGHT JOIN Suppliers S ON P.SupplierID = S.SupplierID WHERE P.SupplierID IS NULL;

Output:



```
SQL> SELECT P.ProductName, S.SupplierName, S.City FROM Products P LEFT JOIN Suppliers S ON P.SupplierID = S.SupplierID WHERE S.SupplierID IS NULL UNION ALL SELECT P.ProductName, S.SupplierName, S.City FROM Pro
ducts P RIGHT JOIN Suppliers S ON P.SupplierID = S.SupplierID WHERE P.SupplierID IS NULL;

PRODUCTNAME
--------------------------------------------------------------------------------
SUPPLIERNAME
--------------------------------------------------------------------------------
CITY
--------------------------------------------------

SureGrip Fasteners
Dallas


Eastern Lumber Co.
Boston

PRODUCTNAME
--------------------------------------------------------------------------------
SUPPLIERNAME
--------------------------------------------------------------------------------
CITY
--------------------------------------------------


Contractor Essentials
Atlanta


Home Fix Depot

PRODUCTNAME
--------------------------------------------------------------------------------
SUPPLIERNAME
--------------------------------------------------------------------------------
CITY
--------------------------------------------------
Houston


Super Wrench Group
New York
```

EXPERIMENT 8

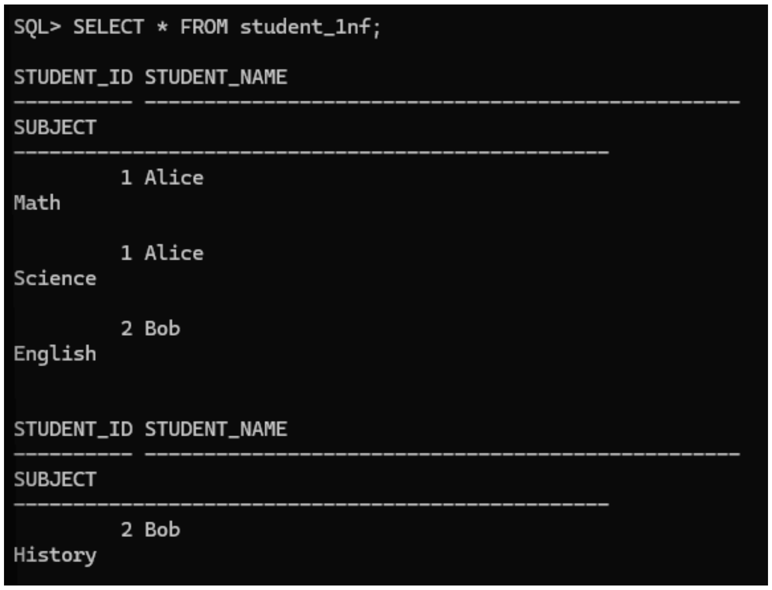
Normal Forms

1. **1NF (First Normal Form)**: Eliminates repeating groups; ensures atomic values in each column.
2. **2NF (Second Normal Form)**: Removes partial dependencies; every non-key attribute fully depends on the primary key.
3. **3NF:** Non-key columns depend only on the whole primary key — not on other non-key columns.

Source code:1NF

```
CREATE TABLE student_1nf (
   student_id INT,
   student_name VARCHAR(50),
   subject VARCHAR(50));
INSERT INTO student_1nf VALUES (1, 'Alice', 'Math');
INSERT INTO student_1nf VALUES (1, 'Alice', 'Science');
INSERT INTO student_1nf VALUES (2, 'Bob', 'English');
INSERT INTO student_1nf VALUES (2, 'Bob', 'History');
SELECT * FROM student_1nf;
```

Output:

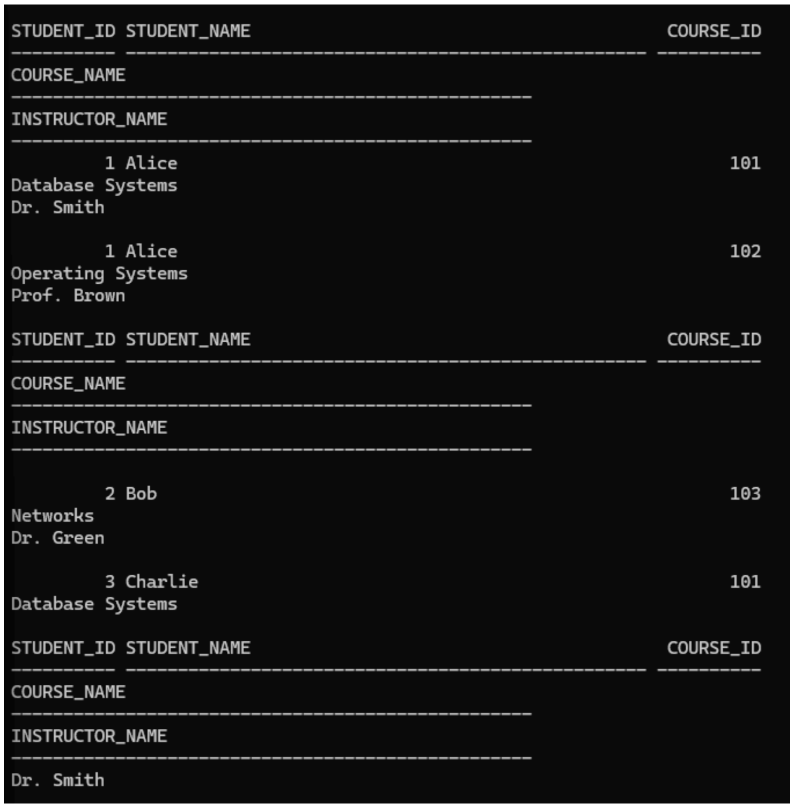


```
SQL> SELECT * FROM student_1nf;

STUDENT_ID STUDENT_NAME
---------- --------------------------------------------------
SUBJECT
--------------------------------------------------
         1 Alice
Math

         1 Alice
Science

         2 Bob
English


STUDENT_ID STUDENT_NAME
---------- --------------------------------------------------
SUBJECT
--------------------------------------------------
         2 Bob
History
```

Source code:2NF:

```
CREATE TABLE Students2 (
   student_id INT PRIMARY KEY,
   student_name VARCHAR2(50));
CREATE TABLE Courses2 (
   course_id INT PRIMARY KEY,
   course_name VARCHAR2(50),
   instructor_name VARCHAR2(50));
CREATE TABLE Enrollments2 (
   student_id INT,
   course_id INT,
   PRIMARY KEY (student_id, course_id),
   FOREIGN KEY (student_id) REFERENCES Students2(student_id),
   FOREIGN KEY (course_id) REFERENCES Courses2(course_id));
INSERT INTO Students2 VALUES (1, 'Alice');
INSERT INTO Students2 VALUES (2, 'Bob');
INSERT INTO Students2 VALUES (3, 'Charlie');
INSERT INTO Courses2 VALUES (101, 'Database Systems', 'Dr. Smith');
INSERT INTO Courses2 VALUES (102, 'Operating Systems', 'Prof. Brown');
INSERT INTO Courses2 VALUES (103, 'Networks', 'Dr. Green');
INSERT INTO Enrollments2 VALUES (1, 101);
INSERT INTO Enrollments2 VALUES (1, 102);
INSERT INTO Enrollments2 VALUES (2, 103);
INSERT INTO Enrollments2 VALUES (3, 101);
COMMIT;
SELECT
   s.student_id,
   s.student_name,
   c.course_id,
   c.course_name,
   c.instructor_name
FROM Enrollments2 e
JOIN Students2 s ON e.student_id = s.student_id
JOIN Courses2 c ON e.course_id = c.course_id
ORDER BY s.student_id, c.course_id;
```

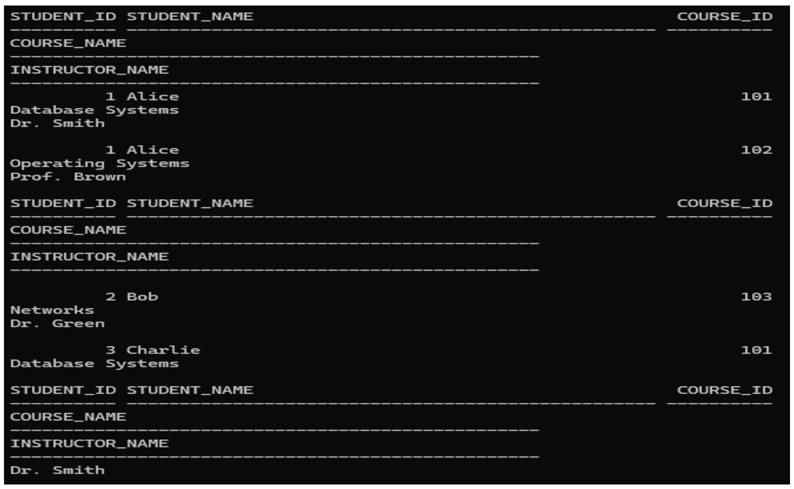
Output:


```
STUDENT_ID STUDENT_NAME                                         COURSE_ID
---------- -------------------------------------------------- ----------
COURSE_NAME
--------------------------------------------------
INSTRUCTOR_NAME
--------------------------------------------------
         1 Alice                                                     101
Database Systems
Dr. Smith

         1 Alice                                                     102
Operating Systems
Prof. Brown

STUDENT_ID STUDENT_NAME                                         COURSE_ID
---------- -------------------------------------------------- ----------
COURSE_NAME
--------------------------------------------------
INSTRUCTOR_NAME
--------------------------------------------------

         2 Bob                                                       103
Networks
Dr. Green

         3 Charlie                                                   101
Database Systems

STUDENT_ID STUDENT_NAME                                         COURSE_ID
---------- -------------------------------------------------- ----------
COURSE_NAME
--------------------------------------------------
INSTRUCTOR_NAME
--------------------------------------------------
Dr. Smith
```

Source code: 3NF

```
CREATE TABLE Students3 (
   student_id INT PRIMARY KEY,
   student_name VARCHAR2(50));
CREATE TABLE Instructors3 (
   instructor_id INT PRIMARY KEY,
   instructor_name VARCHAR2(50));
CREATE TABLE Courses3 (
   course_id INT PRIMARY KEY,
   course_name VARCHAR2(50),
   instructor_id INT,
   FOREIGN KEY (instructor_id) REFERENCES Instructors3(instructor_id));
CREATE TABLE Enrollments3 (
   student_id INT,
   course_id INT,
   PRIMARY KEY (student_id, course_id),
   FOREIGN KEY (student_id) REFERENCES Students3(student_id),
   FOREIGN KEY (course_id) REFERENCES Courses3(course_id));
INSERT INTO Students3 VALUES (1, 'Alice');
INSERT INTO Students3 VALUES (2, 'Bob');
INSERT INTO Students3 VALUES (3, 'Charlie');
INSERT INTO Instructors3 VALUES (201, 'Dr. Smith');
INSERT INTO Instructors3 VALUES (202, 'Prof. Brown');
INSERT INTO Instructors3 VALUES (203, 'Dr. Green');
INSERT INTO Courses3 VALUES (101, 'Database Systems', 201);
INSERT INTO Courses3 VALUES (102, 'Operating Systems', 202);
INSERT INTO Courses3 VALUES (103, 'Networks', 203);
INSERT INTO Enrollments3 VALUES (1, 101);
INSERT INTO Enrollments3 VALUES (1, 102);
INSERT INTO Enrollments3 VALUES (2, 103);
INSERT INTO Enrollments3 VALUES (3, 101);
COMMIT;
SELECT
   s.student_id,
   s.student_name,
   c.course_id,
   c.course_name,
   i.instructor_name
FROM Enrollments3 e
JOIN Students3 s ON e.student_id = s.student_id
JOIN Courses3 c ON e.course_id = c.course_id
JOIN Instructors3 i ON c.instructor_id = i.instructor_id
ORDER BY s.student_id, c.course_id;
```

Output



```
STUDENT_ID STUDENT_NAME                                        COURSE_ID
---------- -------------------------------------------------- ----------
COURSE_NAME
--------------------------------------------------
INSTRUCTOR_NAME
--------------------------------------------------
         1 Alice                                                     101
Database Systems
Dr. Smith

         1 Alice                                                     102
Operating Systems
Prof. Brown

STUDENT_ID STUDENT_NAME                                        COURSE_ID
---------- -------------------------------------------------- ----------
COURSE_NAME
--------------------------------------------------
INSTRUCTOR_NAME
--------------------------------------------------

         2 Bob                                                       103
Networks
Dr. Green

         3 Charlie                                                   101
Database Systems

STUDENT_ID STUDENT_NAME                                        COURSE_ID
---------- -------------------------------------------------- ----------
COURSE_NAME
--------------------------------------------------
INSTRUCTOR_NAME
--------------------------------------------------
Dr. Smith
```

EXPERIMENT 9

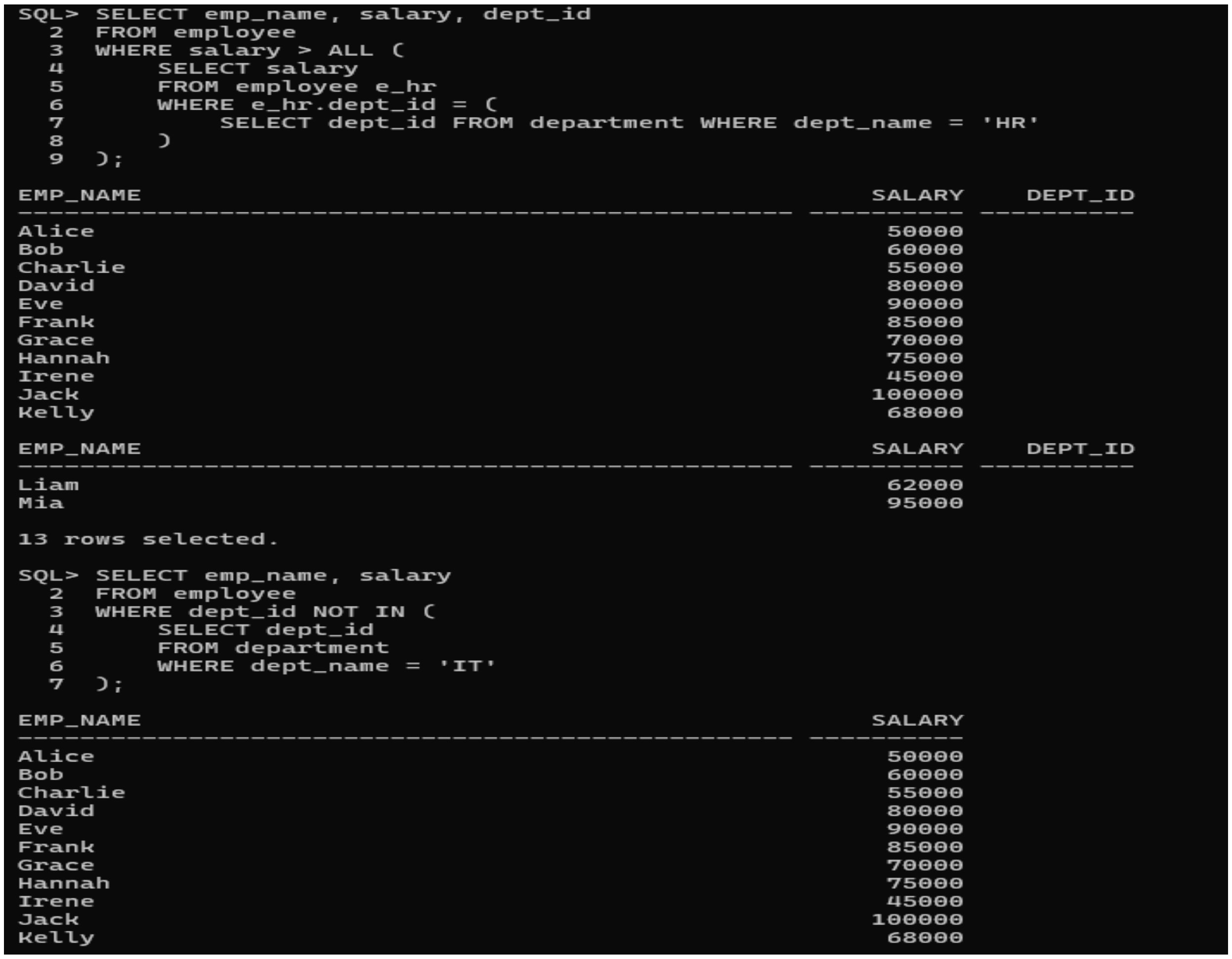
Nested Queries: A query within another SQL query, used to perform intermediate filtering or calculations.

Source code:

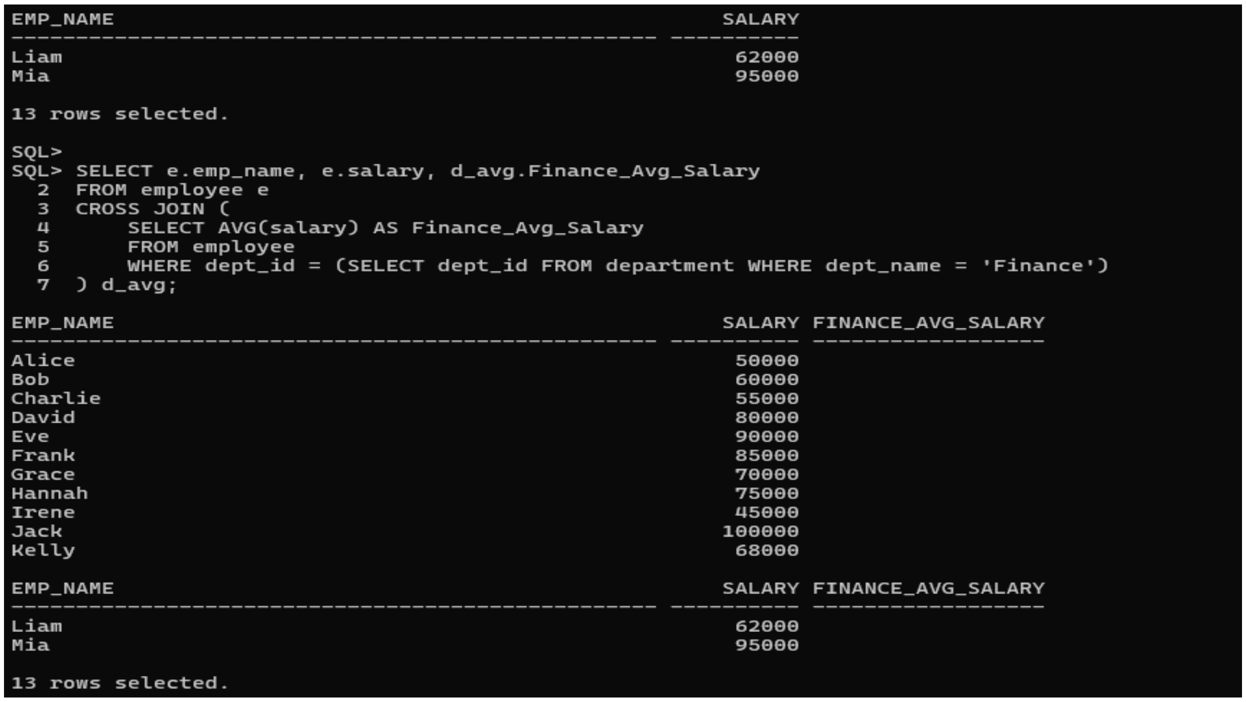
```
CREATE TABLE department ( dept_id INT PRIMARY KEY, dept_name
VARCHAR(50));
CREATE SEQUENCE dept_seq START WITH 1 INCREMENT BY 1;
CREATE OR REPLACE TRIGGER dept_on_insert
BEFORE INSERT ON department
FOR EACH ROW
 BEGIN
SELECT dept_seq.nextval INTO :new.dept_id FROM dual;
 END;
CREATE TABLE employee (emp_id INT PRIMARY KEY,emp_name
VARCHAR(50),salary NUMBER(10,2), dept_id INT, FOREIGN KEY (dept_id)
REFERENCES department(dept_id));
CREATE SEQUENCE emp_seq START WITH 1 INCREMENT BY 1;
CREATE OR REPLACE TRIGGER emp_on_insert
BEFORE INSERT ON employee
FOR EACH ROW
BEGIN
   SELECT emp_seq.nextval INTO :new.emp_id FROM dual;
END;
INSERT INTO employee (emp_name, salary, dept_id) VALUES ('Alice', 50000,
(SELECT dept_id FROM department WHERE dept_name = 'HR'));
INSERT INTO employee (emp_name, salary, dept_id) VALUES ('Bob', 60000,
(SELECT dept_id FROM department WHERE dept_name = 'HR'));
INSERT INTO employee (emp_name, salary, dept_id) VALUES ('Charlie', 55000,
(SELECT dept_id FROM department WHERE dept_name = 'HR'));
INSERT INTO employee (emp_name, salary, dept_id) VALUES ('David', 80000,
(SELECT dept_id FROM department WHERE dept_name = 'IT'));
INSERT INTO employee (emp_name, salary, dept_id) VALUES ('Eve', 90000,
(SELECT dept_id FROM department WHERE dept_name = 'IT'));
INSERT INTO employee (emp_name, salary, dept_id) VALUES ('Frank', 85000,
(SELECT dept_id FROM department WHERE dept_name = 'IT'));
INSERT INTO employee (emp_name, salary, dept_id) VALUES ('Grace', 70000,
(SELECT dept_id FROM department WHERE dept_name = 'Finance'));
INSERT INTO employee (emp_name, salary, dept_id) VALUES ('Hannah',
75000, (SELECT dept_id FROM department WHERE dept_name = 'Finance'));
```

```
INSERT INTO employee (emp_name, salary, dept_id) VALUES ('Irene', 45000,
(SELECT dept_id FROM department WHERE dept_name = 'HR'));
INSERT INTO employee (emp_name, salary, dept_id) VALUES ('Jack', 100000,
(SELECT dept_id FROM department WHERE dept_name = 'IT'));
INSERT INTO employee (emp_name, salary, dept_id) VALUES ('Kelly', 68000,
(SELECT dept_id FROM department WHERE dept_name = 'Finance'));
INSERT INTO employee (emp_name, salary, dept_id) VALUES ('Liam', 62000,
(SELECT dept_id FROM department WHERE dept_name = 'HR'));
INSERT INTO employee (emp_name, salary, dept_id) VALUES ('Mia', 95000,
(SELECT dept_id FROM department WHERE dept_name = 'IT'));
COMMIT;
SELECT emp_name, salary, dept_id
FROM employee
WHERE salary > ALL (
   SELECT salary
   FROM employee e_hr
   WHERE e_hr.dept_id = (
      SELECT dept_id FROM department WHERE dept_name = 'HR'));
SELECT emp_name, salary
FROM employee
WHERE dept_id NOT IN (
   SELECT dept_id
   FROM department
   WHERE dept_name = 'IT');
SELECT e.emp_name, e.salary, d_avg.Finance_Avg_Salary
FROM employee e
CROSS JOIN (
   SELECT AVG(salary) AS Finance_Avg_Salary
   FROM employee
   WHERE dept_id = (SELECT dept_id FROM department WHERE dept_name
= 'Finance')) d_avg;
```

Output:



```
SQL> SELECT emp_name, salary, dept_id
  2  FROM employee
  3  WHERE salary > ALL (
  4      SELECT salary
  5      FROM employee e_hr
  6      WHERE e_hr.dept_id = (
  7          SELECT dept_id FROM department WHERE dept_name = 'HR'
  8      )
  9  );

EMP_NAME                                               SALARY    DEPT_ID
-------------------------------------------------- ---------- ----------
Alice                                                   50000
Bob                                                     60000
Charlie                                                 55000
David                                                   80000
Eve                                                     90000
Frank                                                   85000
Grace                                                   70000
Hannah                                                  75000
Irene                                                   45000
Jack                                                   100000
Kelly                                                   68000

EMP_NAME                                               SALARY    DEPT_ID
-------------------------------------------------- ---------- ----------
Liam                                                    62000
Mia                                                     95000

13 rows selected.

SQL> SELECT emp_name, salary
  2  FROM employee
  3  WHERE dept_id NOT IN (
  4      SELECT dept_id
  5      FROM department
  6      WHERE dept_name = 'IT'
  7  );

EMP_NAME                                               SALARY
-------------------------------------------------- ----------
Alice                                                   50000
Bob                                                     60000
Charlie                                                 55000
David                                                   80000
Eve                                                     90000
Frank                                                   85000
Grace                                                   70000
Hannah                                                  75000
Irene                                                   45000
Jack                                                   100000
Kelly                                                   68000
```



```
EMP_NAME                                               SALARY
-------------------------------------------------- ----------
Liam                                                    62000
Mia                                                     95000

13 rows selected.

SQL>
SQL> SELECT e.emp_name, e.salary, d_avg.Finance_Avg_Salary
  2  FROM employee e
  3  CROSS JOIN (
  4      SELECT AVG(salary) AS Finance_Avg_Salary
  5      FROM employee
  6      WHERE dept_id = (SELECT dept_id FROM department WHERE dept_name = 'Finance')
  7  ) d_avg;

EMP_NAME                                               SALARY FINANCE_AVG_SALARY
-------------------------------------------------- ---------- ------------------
Alice                                                   50000
Bob                                                     60000
Charlie                                                 55000
David                                                   80000
Eve                                                     90000
Frank                                                   85000
Grace                                                   70000
Hannah                                                  75000
Irene                                                   45000
Jack                                                   100000
Kelly                                                   68000

EMP_NAME                                               SALARY FINANCE_AVG_SALARY
-------------------------------------------------- ---------- ------------------
Liam                                                    62000
Mia                                                     95000

13 rows selected.
```

EXPERIMENT 10

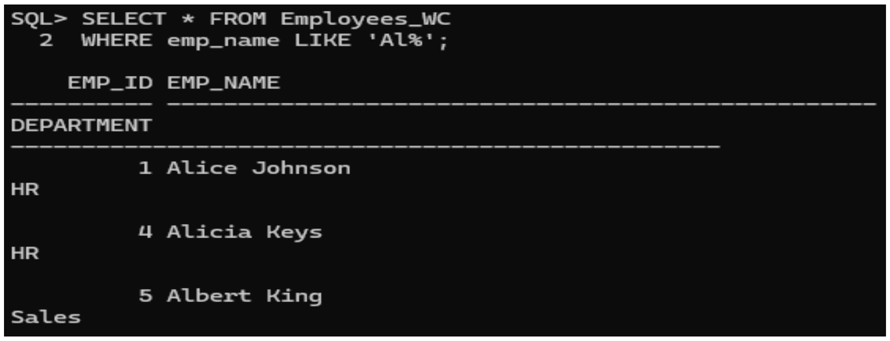
SQL WILD CARD CHARACTERS

1. **%**: Represents **zero or more characters** in a string.
2. **_**: Represents **exactly one character**.

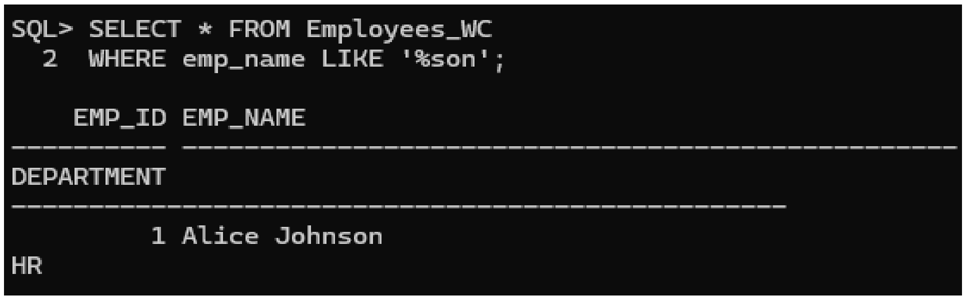
Source code:

```
CREATE TABLE Employees_WC (
   emp_id INT PRIMARY KEY,
   emp_name VARCHAR2(50),
   department VARCHAR2(50));
INSERT INTO Employees_WC VALUES (1, 'Alice Johnson', 'HR');
INSERT INTO Employees_WC VALUES (2, 'Bob Smith', 'Finance');
INSERT INTO Employees_WC VALUES (3, 'Charlie Brown', 'IT');
INSERT INTO Employees_WC VALUES (4, 'Alicia Keys', 'HR');
INSERT INTO Employees_WC VALUES (5, 'Albert King', 'Sales');
INSERT INTO Employees_WC VALUES (6, 'Bobby Ray', 'Finance');
COMMIT;
SELECT * FROM Employees_WC
WHERE emp_name LIKE 'Al%';
SELECT * FROM Employees_WC
WHERE emp_name LIKE '%son';
SELECT * FROM Employees_WC
WHERE emp_name LIKE '%ob%';
SELECT * FROM Employees_WC
WHERE emp_name LIKE '_l%';
SELECT * FROM Employees_WC
WHERE department LIKE 'F%';
```

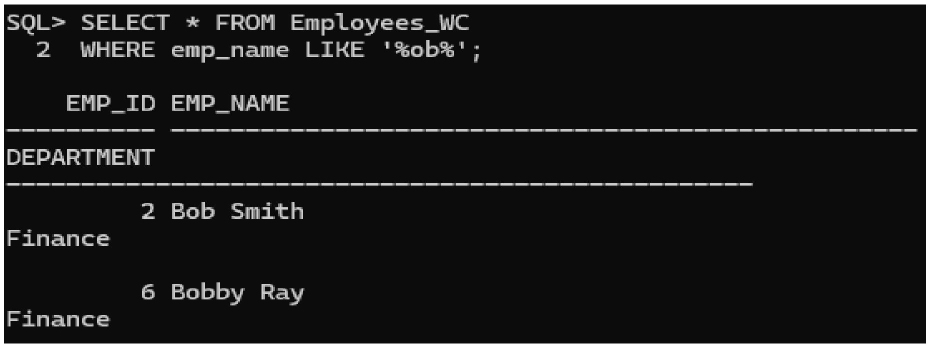
Output:



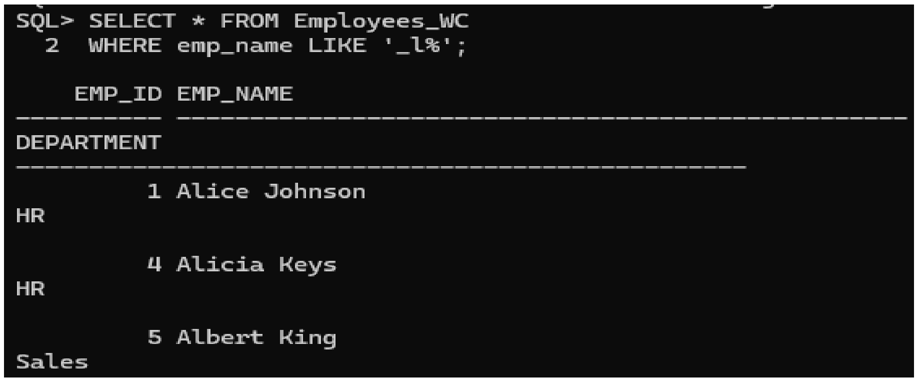
```
SQL> SELECT * FROM Employees_WC
  2  WHERE emp_name LIKE 'Al%';

    EMP_ID EMP_NAME
---------- --------------------------------------------------
DEPARTMENT
--------------------------------------------------
         1 Alice Johnson
HR

         4 Alicia Keys
HR

         5 Albert King
Sales
```



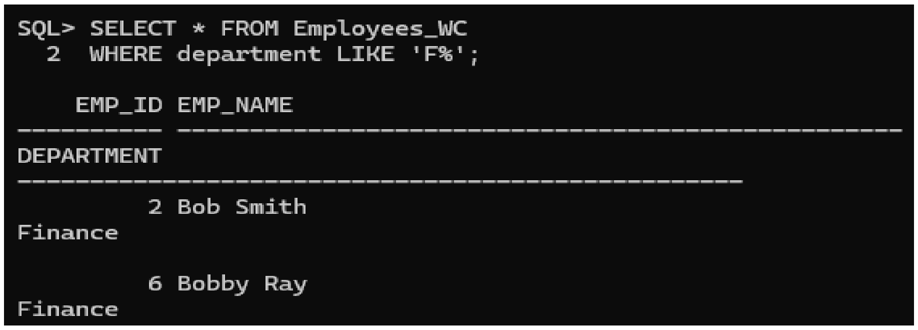
```
SQL> SELECT * FROM Employees_WC
  2  WHERE emp_name LIKE '%son';

    EMP_ID EMP_NAME
---------- --------------------------------------------------
DEPARTMENT
--------------------------------------------------
         1 Alice Johnson
HR
```



```
SQL> SELECT * FROM Employees_WC
  2  WHERE emp_name LIKE '%ob%';

    EMP_ID EMP_NAME
---------- --------------------------------------------------
DEPARTMENT
--------------------------------------------------
         2 Bob Smith
Finance

         6 Bobby Ray
Finance
```



```
SQL> SELECT * FROM Employees_WC
  2  WHERE emp_name LIKE '_l%';

    EMP_ID EMP_NAME
---------- --------------------------------------------------
DEPARTMENT
--------------------------------------------------
         1 Alice Johnson
HR

         4 Alicia Keys
HR

         5 Albert King
Sales
```



```
SQL> SELECT * FROM Employees_WC
  2  WHERE department LIKE 'F%';

    EMP_ID EMP_NAME
---------- --------------------------------------------------
DEPARTMENT
--------------------------------------------------
         2 Bob Smith
Finance

         6 Bobby Ray
Finance
```

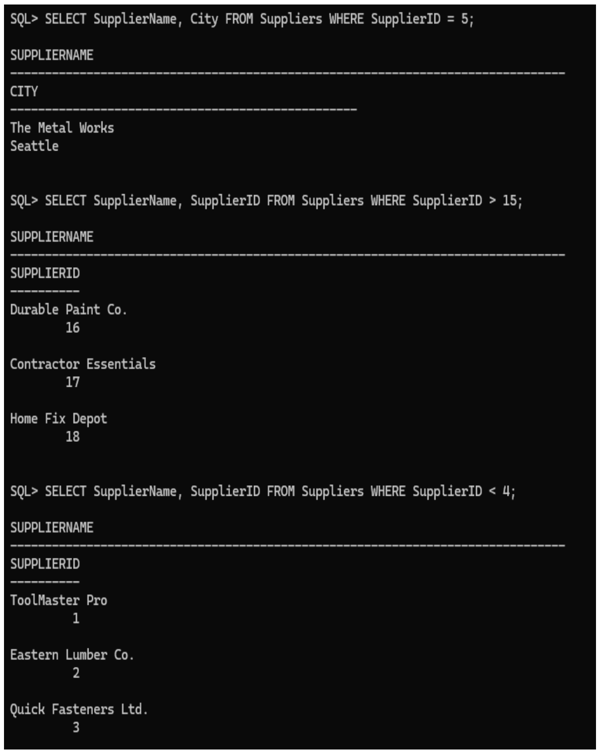
EXPERIMENT 11

SELECT with Comparison Operator: Retrieves rows that meet specific conditions using operators like =, >, <, >=, <=, <>.

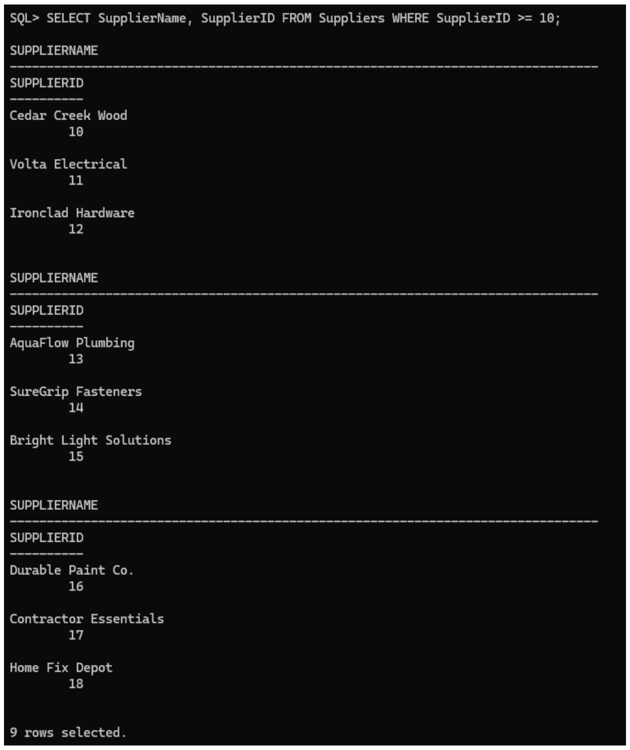
Source code:

```sql
SELECT SupplierName, City FROM Suppliers WHERE SupplierID = 5;

SELECT SupplierName, SupplierID FROM Suppliers WHERE SupplierID > 15;

SELECT SupplierName, SupplierID FROM Suppliers WHERE SupplierID < 4;

SELECT SupplierName, SupplierID FROM Suppliers WHERE SupplierID >= 10;

SELECT SupplierName, SupplierID FROM Suppliers WHERE SupplierID <= 6;

SELECT SupplierName, City FROM Suppliers WHERE City <> 'New York';
```

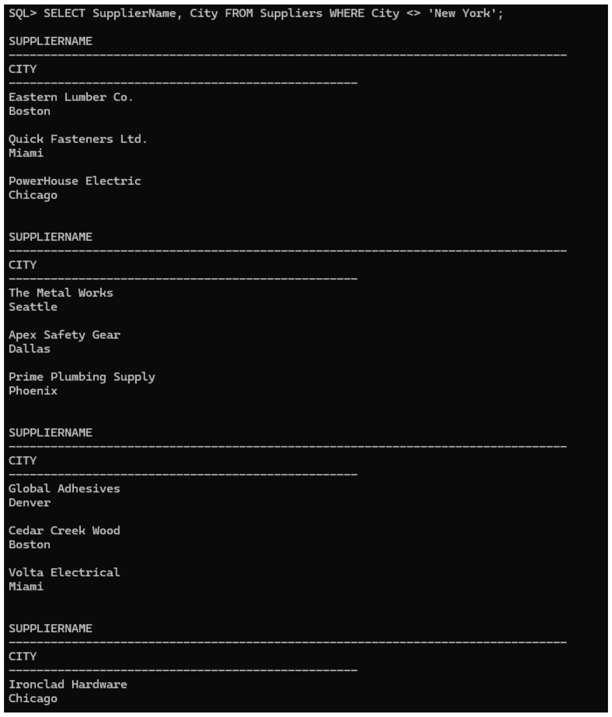
Output:



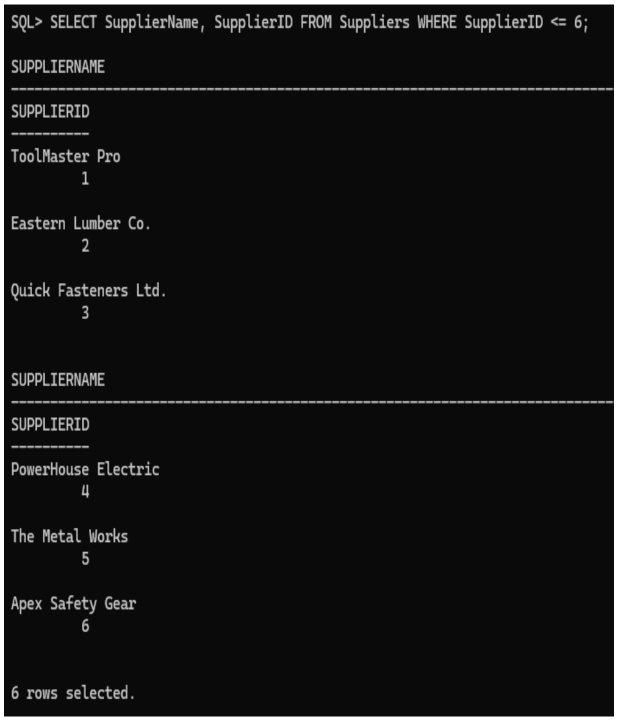
```
SQL> SELECT SupplierName, City FROM Suppliers WHERE SupplierID = 5;

SUPPLIERNAME
--------------------------------------------------------------------------------
CITY
--------------------------------------------------
The Metal Works
Seattle


SQL> SELECT SupplierName, SupplierID FROM Suppliers WHERE SupplierID > 15;

SUPPLIERNAME
--------------------------------------------------------------------------------
SUPPLIERID
----------
Durable Paint Co.
        16

Contractor Essentials
        17

Home Fix Depot
        18


SQL> SELECT SupplierName, SupplierID FROM Suppliers WHERE SupplierID < 4;

SUPPLIERNAME
--------------------------------------------------------------------------------
SUPPLIERID
----------
ToolMaster Pro
         1

Eastern Lumber Co.
         2

Quick Fasteners Ltd.
         3
```



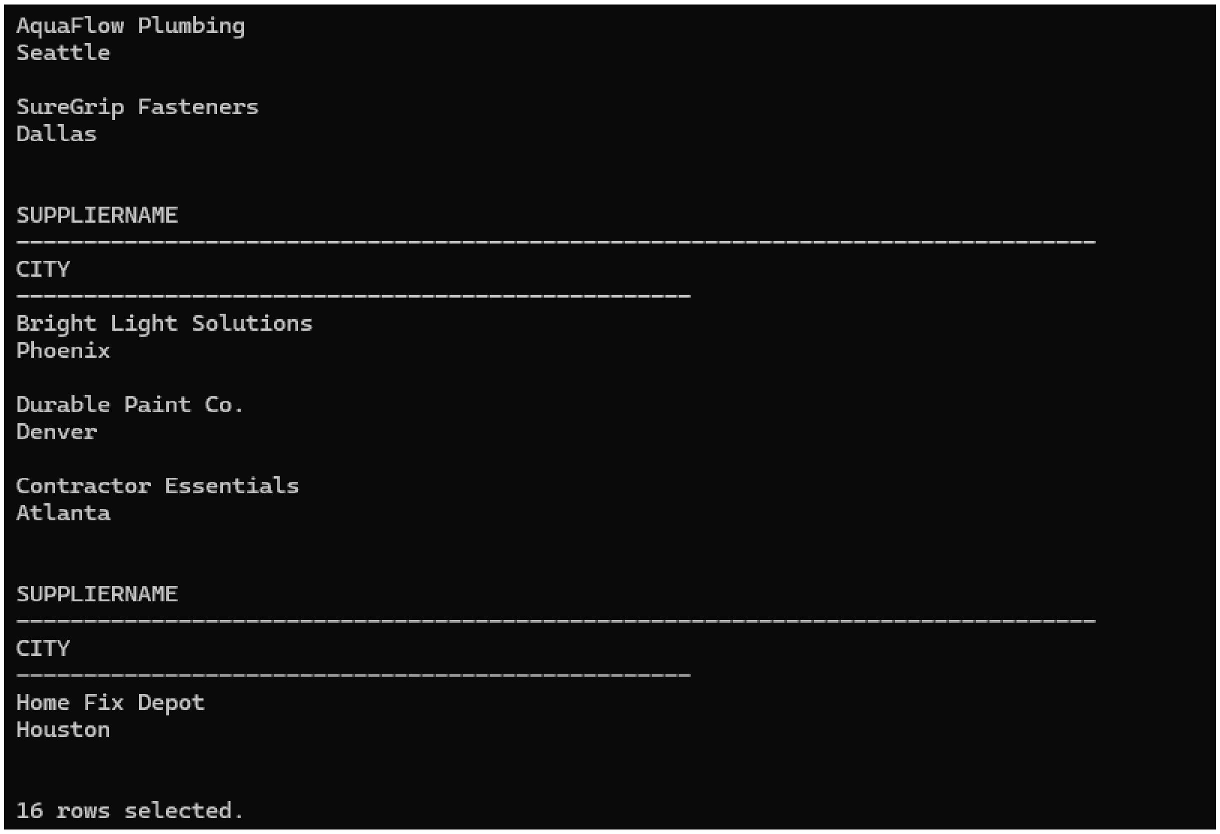
```
SQL> SELECT SupplierName, SupplierID FROM Suppliers WHERE SupplierID >= 10;

SUPPLIERNAME
--------------------------------------------------------------------------------
SUPPLIERID
----------
Cedar Creek Wood
        10

Volta Electrical
        11

Ironclad Hardware
        12


SUPPLIERNAME
--------------------------------------------------------------------------------
SUPPLIERID
----------
AquaFlow Plumbing
        13

SureGrip Fasteners
        14

Bright Light Solutions
        15


SUPPLIERNAME
--------------------------------------------------------------------------------
SUPPLIERID
----------
Durable Paint Co.
        16

Contractor Essentials
        17

Home Fix Depot
        18


9 rows selected.
```



```
SQL> SELECT SupplierName, City FROM Suppliers WHERE City <> 'New York';

SUPPLIERNAME
--------------------------------------------------------------------------------
CITY
--------------------------------------------------
Eastern Lumber Co.
Boston

Quick Fasteners Ltd.
Miami

PowerHouse Electric
Chicago


SUPPLIERNAME
--------------------------------------------------------------------------------
CITY
--------------------------------------------------
The Metal Works
Seattle

Apex Safety Gear
Dallas

Prime Plumbing Supply
Phoenix


SUPPLIERNAME
--------------------------------------------------------------------------------
CITY
--------------------------------------------------
Global Adhesives
Denver

Cedar Creek Wood
Boston

Volta Electrical
Miami


SUPPLIERNAME
--------------------------------------------------------------------------------
CITY
--------------------------------------------------
Ironclad Hardware
Chicago
```



```
SQL> SELECT SupplierName, SupplierID FROM Suppliers WHERE SupplierID <= 6;

SUPPLIERNAME
--------------------------------------------------------------------------------
SUPPLIERID
----------
ToolMaster Pro
         1

Eastern Lumber Co.
         2

Quick Fasteners Ltd.
         3


SUPPLIERNAME
--------------------------------------------------------------------------------
SUPPLIERID
----------
PowerHouse Electric
         4

The Metal Works
         5

Apex Safety Gear
         6


6 rows selected.
```



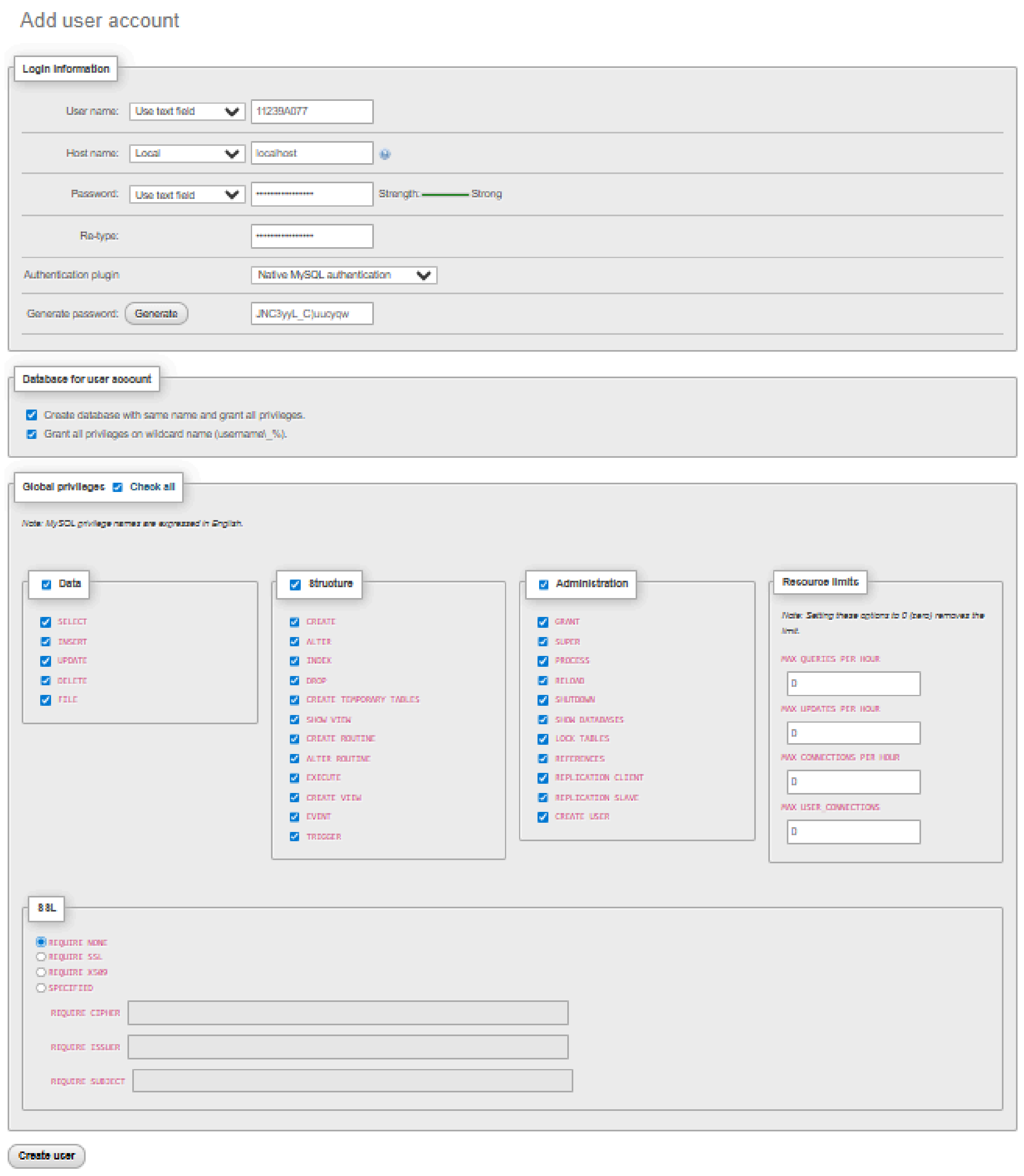
```
AquaFlow Plumbing
Seattle

SureGrip Fasteners
Dallas


SUPPLIERNAME
--------------------------------------------------------------------------------
CITY
--------------------------------------------------
Bright Light Solutions
Phoenix

Durable Paint Co.
Denver

Contractor Essentials
Atlanta


SUPPLIERNAME
--------------------------------------------------------------------------------
CITY
--------------------------------------------------
Home Fix Depot
Houston


16 rows selected.
```

EXPERIMENT 12

Working on Local Host XAMPP Server

1. **Server Variables in XAMPP**: Provide environment and request details via PHP's $_SERVER array.

OUTPUT:



Add user account

Login Information

User name: Use text field | 11239A077

Host name: Local | localhost

Password: Use text field | •••••••••••••••• Strength: Strong

Re-type: ••••••••••••••••

Authentication plugin: Native MySQL authentication

Generate password: Generate | JNC3yyL_C)uucyqw

Database for user account

- [✓] Create database with same name and grant all privileges.
- [✓] Grant all privileges on wildcard name (username_%).

Global privileges [✓] Check all

Note: MySQL privilege names are expressed in English.

[✓] **Data**
- [✓] SELECT
- [✓] INSERT
- [✓] UPDATE
- [✓] DELETE
- [✓] FILE

[✓] **Structure**
- [✓] CREATE
- [✓] ALTER
- [✓] INDEX
- [✓] DROP
- [✓] CREATE TEMPORARY TABLES
- [✓] SHOW VIEW
- [✓] CREATE ROUTINE
- [✓] ALTER ROUTINE
- [✓] EXECUTE
- [✓] CREATE VIEW
- [✓] EVENT
- [✓] TRIGGER

[✓] **Administration**
- [✓] GRANT
- [✓] SUPER
- [✓] PROCESS
- [✓] RELOAD
- [✓] SHUTDOWN
- [✓] SHOW DATABASES
- [✓] LOCK TABLES
- [✓] REFERENCES
- [✓] REPLICATION CLIENT
- [✓] REPLICATION SLAVE
- [✓] CREATE USER

Resource limits

Note: Setting these options to 0 (zero) removes the limit.

MAX QUERIES PER HOUR: 0

MAX UPDATES PER HOUR: 0

MAX CONNECTIONS PER HOUR: 0

MAX USER_CONNECTIONS: 0

SSL

- [✓ REQUIRE NONE]
- REQUIRE SSL
- REQUIRE X509
- SPECIFIED

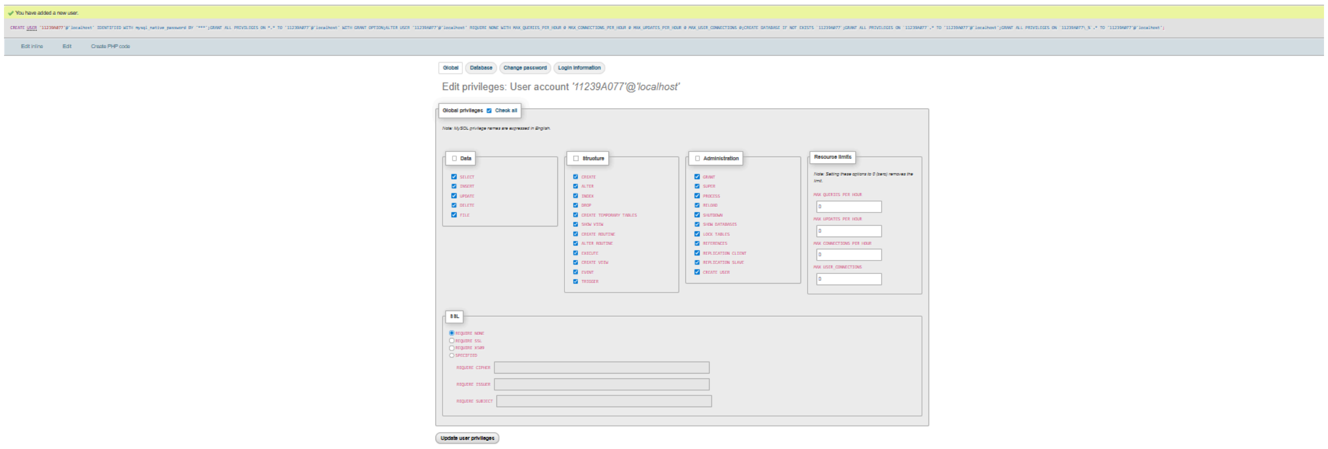
REQUIRE CIPHER:

REQUIRE ISSUER:

REQUIRE SUBJECT:

2. **Hierarchical User Access in XAMPP**: Assign different privileges to MySQL users to control database access levels.

OUTPUT:



You have added a new user.

```
CREATE USER '11239A077'@'localhost' IDENTIFIED WITH mysql_native_password BY '***';GRANT ALL PRIVILEGES ON *.* TO '11239A077'@'localhost' WITH GRANT OPTION;ALTER USER '11239A077'@'localhost' REQUIRE NONE WITH MAX_QUERIES_PER_HOUR 0 MAX_CONNECTIONS_PER_HOUR 0 MAX_UPDATES_PER_HOUR 0 MAX_USER_CONNECTIONS 0;CREATE DATABASE IF NOT EXISTS `11239A077`;GRANT ALL PRIVILEGES ON `11239A077`.* TO '11239A077'@'localhost';GRANT ALL PRIVILEGES ON `11239A077\_%`.* TO '11239A077'@'localhost';
```

Edit privileges: User account *'11239A077'@'localhost'*

Global privileges ✓ Check all

Note: MySQL privilege names are expressed in English.

Data	Structure	Administration
✓ SELECT	✓ CREATE	✓ GRANT
✓ INSERT	✓ ALTER	✓ SUPER
✓ UPDATE	✓ INDEX	✓ PROCESS
✓ DELETE	✓ DROP	✓ RELOAD
✓ FILE	✓ CREATE TEMPORARY TABLES	✓ SHUTDOWN
	✓ SHOW VIEW	✓ SHOW DATABASES
	✓ CREATE ROUTINE	✓ LOCK TABLES
	✓ ALTER ROUTINE	✓ REFERENCES
	✓ EXECUTE	✓ REPLICATION CLIENT
	✓ CREATE VIEW	✓ REPLICATION SLAVE
	✓ EVENT	✓ CREATE USER
	✓ TRIGGER	

Resource limits

Note: Setting these options to 0 (zero) removes the limit.

MAX QUERIES PER HOUR: 0
MAX UPDATES PER HOUR: 0
MAX CONNECTIONS PER HOUR: 0
MAX USER_CONNECTIONS: 0

SSL

● REQUIRE NONE
○ REQUIRE SSL
○ REQUIRE X509
○ SPECIFIED

REQUIRE CIPHER:
REQUIRE ISSUER:
REQUIRE SUBJECT: