WEEK-2) - SQL

QUESTION-01)

Exercise 1: Ranking and Window Functions

Goal: Use ROW_NUMBER(), RANK(), DENSE_RANK(), OVER(), and PARTITION BY.

Scenario:

Find the top 3 most expensive products in each category using different ranking functions.

Steps:

1. Use ROW_NUMBER() to assign a unique rank within each category.

2. Use RANK() and DENSE_RANK() to compare how ties are handled.

3. Use PARTITION BY Category and ORDER BY Price DESC?

Solution:-

```
create database sqll;
use sqll;
-- Database Schema
CREATE TABLE Customers (
    CustomerID INT PRIMARY KEY,
    Name VARCHAR(100),
    Region VARCHAR(50)
);

CREATE TABLE Products (
    ProductID INT PRIMARY KEY,
    ProductName VARCHAR(100),
```

```
    Category VARCHAR(50),
    Price DECIMAL(10, 2)
);

CREATE TABLE Orders (
    OrderID INT PRIMARY KEY,
    CustomerID INT,
    OrderDate DATE,
    FOREIGN KEY (CustomerID) REFERENCES Customers(CustomerID)
);

CREATE TABLE OrderDetails (
    OrderDetailID INT PRIMARY KEY,
    OrderID INT,
    ProductID INT,
    Quantity INT,
    FOREIGN KEY (OrderID) REFERENCES Orders(OrderID),
    FOREIGN KEY (ProductID) REFERENCES Products(ProductID)
);

-- Sample Data
INSERT INTO Customers (CustomerID, Name, Region) VALUES
(1, 'Alice', 'North'),
(2, 'Bob', 'South'),
(3, 'Charlie', 'East'),
```

(4, 'David', 'West');

INSERT INTO Products (ProductID, ProductName, Category, Price) VALUES

(1, 'Laptop', 'Electronics', 1200.00),

(2, 'Smartphone', 'Electronics', 800.00),

(3, 'Tablet', 'Electronics', 600.00),

(4, 'Headphones', 'Accessories', 150.00);

INSERT INTO Orders (OrderID, CustomerID, OrderDate) VALUES

(1, 1, '2023-01-15'),

(2, 2, '2023-02-20'),

(3, 3, '2023-03-25'),

(4, 4, '2023-04-30');

INSERT INTO OrderDetails (OrderDetailID, OrderID, ProductID, Quantity) VALUES

(1, 1, 1, 1),

(2, 2, 2, 2),

(3, 3, 3, 1),

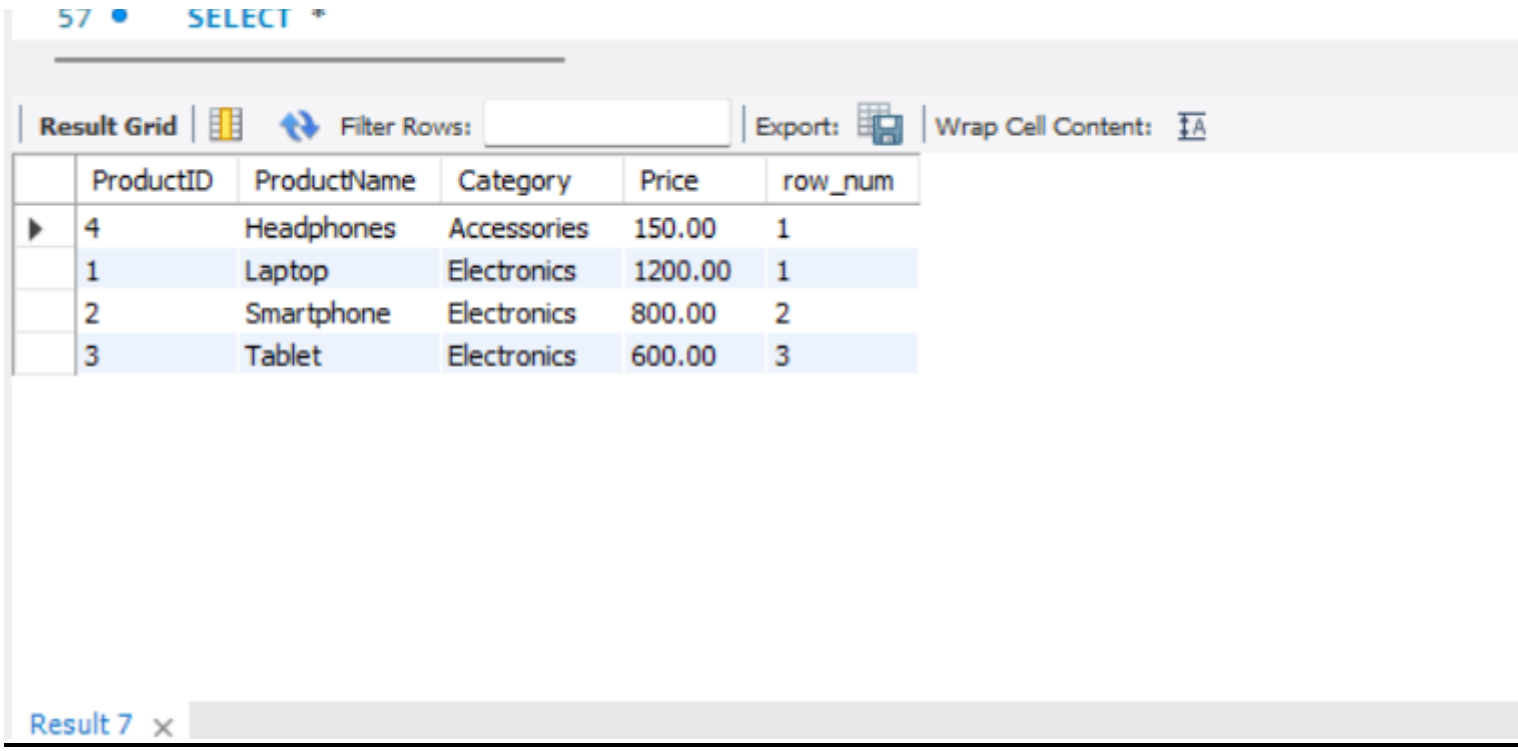
(4, 4, 4, 3);

1. Use ROW_NUMBER() to assign a unique rank within each category.

QUERY:-1)

```
SELECT *
FROM (
   SELECT
      ProductID,
      ProductName,
      Category,
      Price,
      ROW_NUMBER() OVER (PARTITION BY Category ORDER BY Price DESC) AS row_num
   FROM Products
) AS ranked
WHERE row_num <= 3;
```

OUTPUT:-

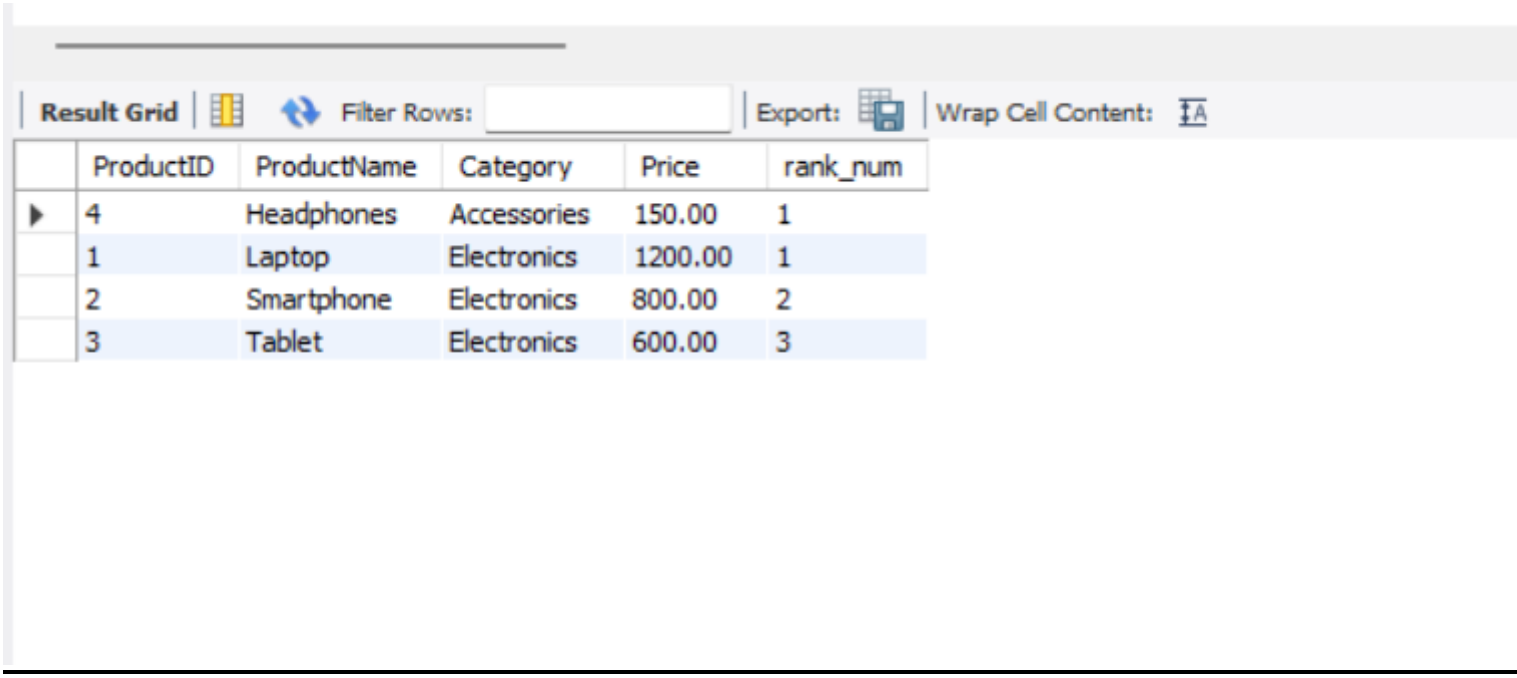


2. Use RANK() and DENSE_RANK() to compare how ties are handled.

QUERY 2:-

```
SELECT *
FROM (
   SELECT
      ProductID,
      ProductName,
      Category,
      Price,
      RANK() OVER (PARTITION BY Category ORDER BY Price DESC) AS rank_num
   FROM Products
) AS ranked
WHERE rank_num <= 3;
```

OUTPUT:-



Result Grid | Filter Rows: | Export: | Wrap Cell Content:

ProductID	ProductName	Category	Price	rank_num
4	Headphones	Accessories	150.00	1
1	Laptop	Electronics	1200.00	1
2	Smartphone	Electronics	800.00	2
3	Tablet	Electronics	600.00	3

```
SELECT *
FROM (
```

```
SELECT
    ProductID,
    ProductName,
    Category,
    Price,
    DENSE_RANK() OVER (PARTITION BY Category ORDER BY Price DESC) AS dense_rank_num
FROM Products
) AS ranked
WHERE dense_rank_num <= 3;
```

OUTPUT:-

92

Result Grid | Filter Rows: | Export: | Wrap Cell Content:

ProductID	ProductName	Category	Price	dense_rank_num
4	Headphones	Accessories	150.00	1
1	Laptop	Electronics	1200.00	1
2	Smartphone	Electronics	800.00	2
3	Tablet	Electronics	600.00	3

QUESTION -02)

Exercise 1: Create a Stored Procedure

Goal: Create a stored procedure to retrieve employee details by department.

Steps:

1. Define the stored procedure with a parameter for DepartmentID.

2. Write the SQL query to select employee details based on the DepartmentID.

3. Create a stored procedure named `sp_InsertEmployee` with the following code:

CREATE PROCEDURE sp_InsertEmployee

@FirstName VARCHAR(50),

@LastName VARCHAR(50),

@DepartmentID INT,

@Salary DECIMAL(10,2),

@JoinDate DATE

AS

BEGIN

INSERT INTO Employees (FirstName, LastName, DepartmentID, Salary, JoinDate)

VALUES (@FirstName, @LastName, @DepartmentID, @Salary, @JoinDate);

END;

SOLUTION)

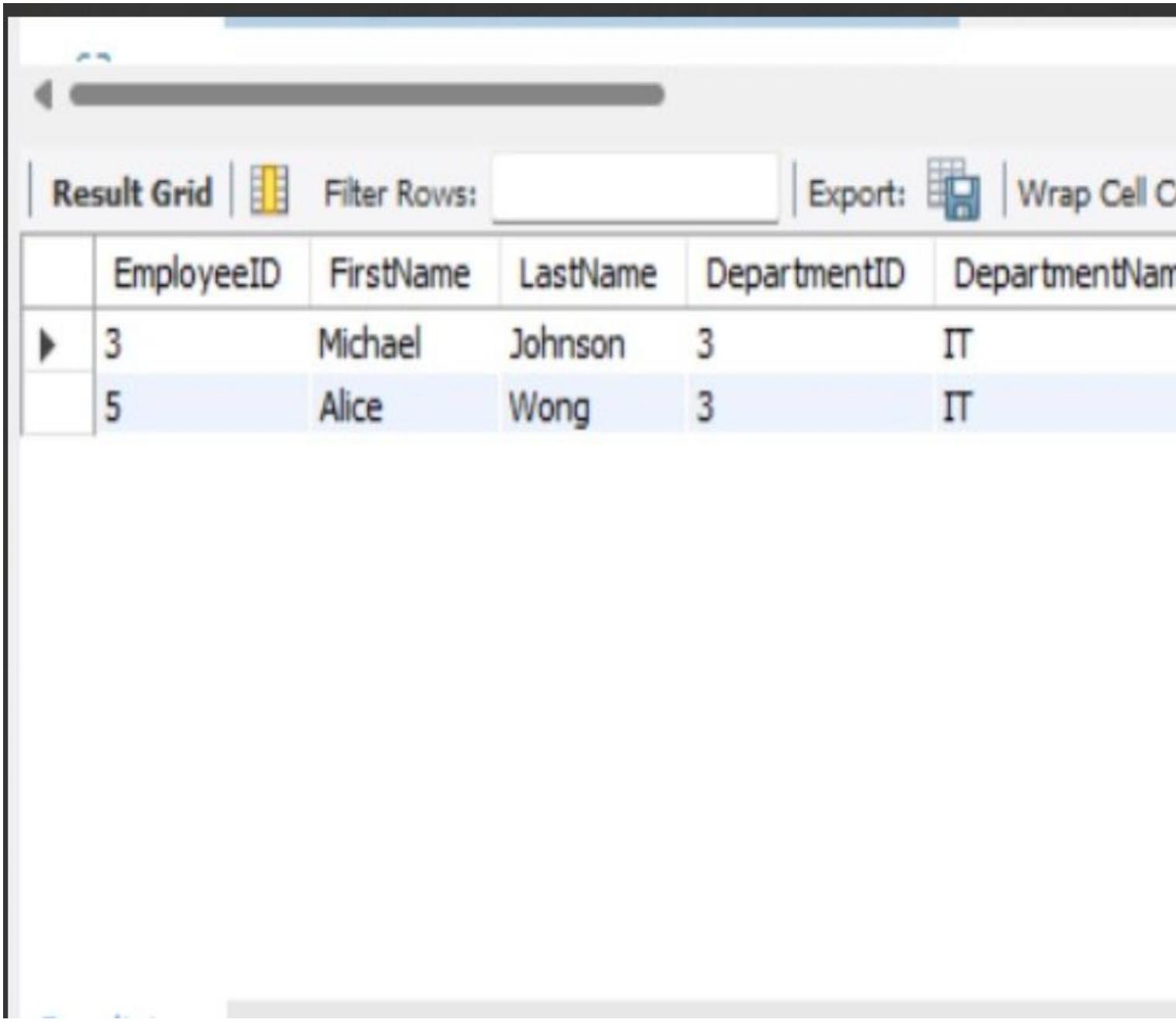
```
use Employee_Management_System;


cREATE TABLE Departments (
 DepartmentID INT PRIMARY KEY,
 DepartmentName VARCHAR(100)
);
```

```
CREATE TABLE Employees (
    EmployeeID INT PRIMARY KEY,
    FirstName VARCHAR(50) NOT NULL,
    LastName VARCHAR(50) NOT NULL,
    DepartmentID INT NOT NULL,
    Salary DECIMAL(10,2) NOT NULL,
    JoinDate DATE NOT NULL,
    FOREIGN KEY (DepartmentID) REFERENCES
Departments(DepartmentID)
);


INSERT INTO Departments (DepartmentID, DepartmentName) VALUES
(1, 'HR'),
(2, 'Finance'),
(3, 'IT'),
(4, 'Marketing');


INSERT INTO Employees (EmployeeID, FirstName, LastName,
DepartmentID, Salary,
JoinDate) VALUES
(1, 'John', 'Doe', 1, 5000.00, '2020-01-15'),
(2, 'Jane', 'Smith', 2, 6000.00, '2019-03-22'),
```

```
(3, 'Michael', 'Johnson', 3, 7000.00, '2018-07-30'),
(4, 'Emily', 'Davis', 4, 5500.00, '2021-11-05');



DELIMITER $$
CREATE PROCEDURE sp_GetEmployeesByDepartment (
  IN p_DepartmentID INT
)
BEGIN
  SELECT
    e.EmployeeID,
    e.FirstName,
    e.LastName,
    e.DepartmentID,
    d.DepartmentName,
    e.Salary,
    e.JoinDate
  FROM Employees AS e
  JOIN Departments AS d
    ON e.DepartmentID = d.DepartmentID
  WHERE e.DepartmentID = p_DepartmentID;
END $$
DELIMITER ;
```

```
DELIMITER $$
CREATE PROCEDURE sp_InsertEmployee (
  IN p_FirstName    VARCHAR(50),
  IN p_LastName     VARCHAR(50),
  IN p_DepartmentID INT,
  IN p_Salary       DECIMAL(10,2),
  IN p_JoinDate     DATE
)
BEGIN
  INSERT INTO Employees
    (FirstName, LastName, DepartmentID, Salary, JoinDate)
  VALUES
    (p_FirstName, p_LastName, p_DepartmentID, p_Salary, p_JoinDate);
END $$
DELIMITER ;



CALL sp_InsertEmployee('Alice', 'Wong', 3, 7200.00, '2025-06-26');


CALL sp_GetEmployeesByDepartment(3);
```

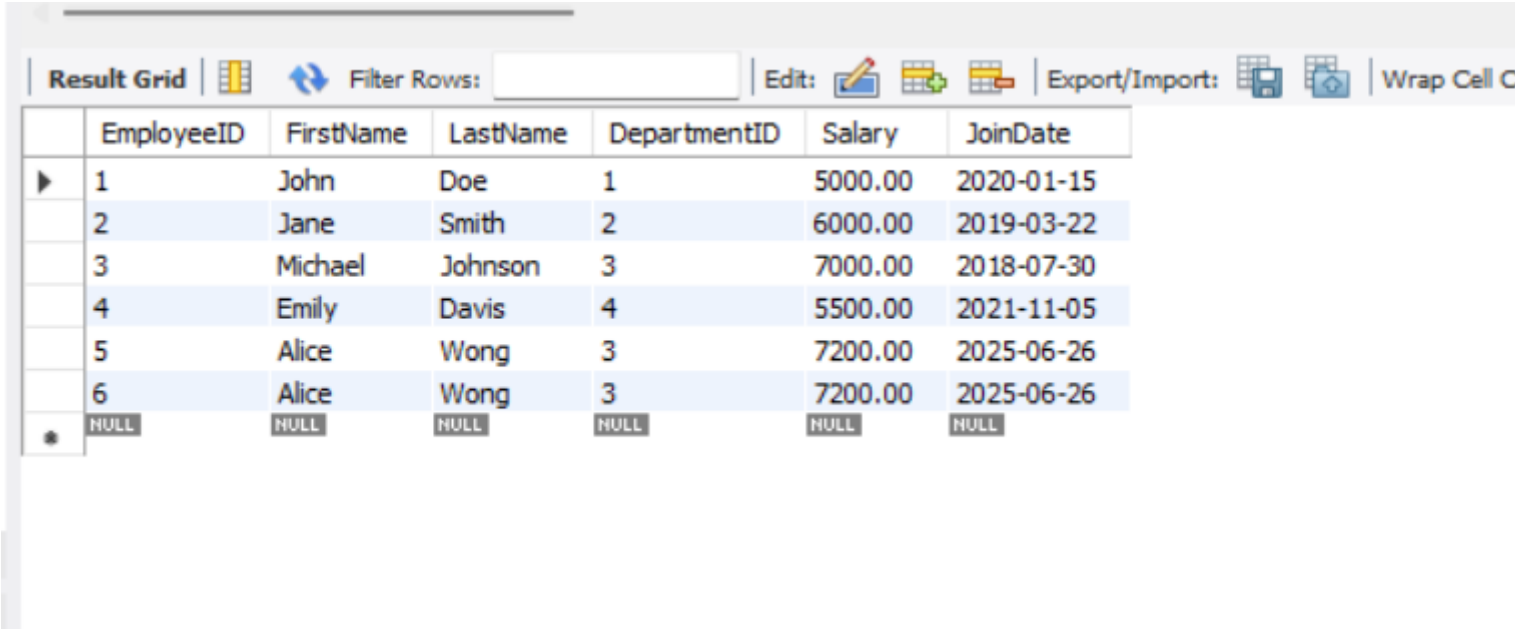
OUTPUT:-



Querry:

SELECT * FROM Employees ORDER BY EmployeeID;

Output:-



Result Grid | Filter Rows: | Edit: | Export/Import: | Wrap Cell C

EmployeeID	FirstName	LastName	DepartmentID	Salary	JoinDate
1	John	Doe	1	5000.00	2020-01-15
2	Jane	Smith	2	6000.00	2019-03-22
3	Michael	Johnson	3	7000.00	2018-07-30
4	Emily	Davis	4	5500.00	2021-11-05
5	Alice	Wong	3	7200.00	2025-06-26
6	Alice	Wong	3	7200.00	2025-06-26
NULL	NULL	NULL	NULL	NULL	NULL

QUESTION2) Exercise 7: Return Data from a Scalar Function

Goal: Return the annual salary for a specific employee using `fn_CalculateAnnualSalary`.

Steps:

1. Execute the `fn_CalculateAnnualSalary` function for an employee with `EmployeeID = 1`.

2. Verify the result

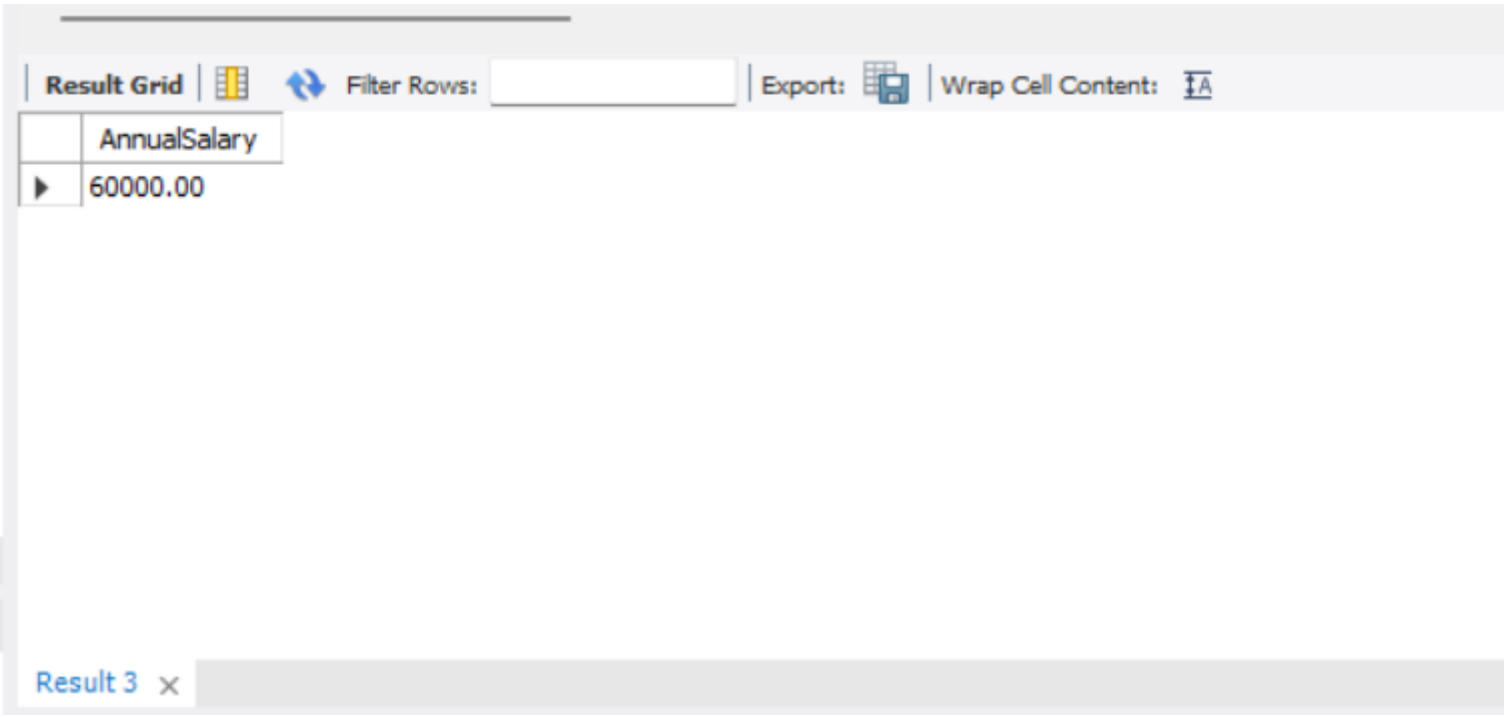
SOLUTION:-

```
USE EmployeeDB;


DROP FUNCTION IF EXISTS fn_CalculateAnnualSalary;

DELIMITER $$

CREATE FUNCTION fn_CalculateAnnualSalary (

  p_EmployeeID INT
```

```
) RETURNS DECIMAL(12,2)
DETERMINISTIC
BEGIN
  DECLARE v_MonthlySalary DECIMAL(10,2);
  DECLARE v_AnnualSalary DECIMAL(12,2);

  -- Fetch the employee's monthly salary
  SELECT Salary
    INTO v_MonthlySalary
    FROM Employees
   WHERE EmployeeID = p_EmployeeID;

  -- Compute annual salary
  SET v_AnnualSalary = v_MonthlySalary * 12;

  RETURN v_AnnualSalary;
END $$
DELIMITER ;
```

###SELECT fn_CalculateAnnualSalary(1) AS AnnualSalary;

OUTPUT:-



-- 2.b Side-by-side with Employee record

###SELECT

e.EmployeeID,

e.FirstName,

e.LastName,

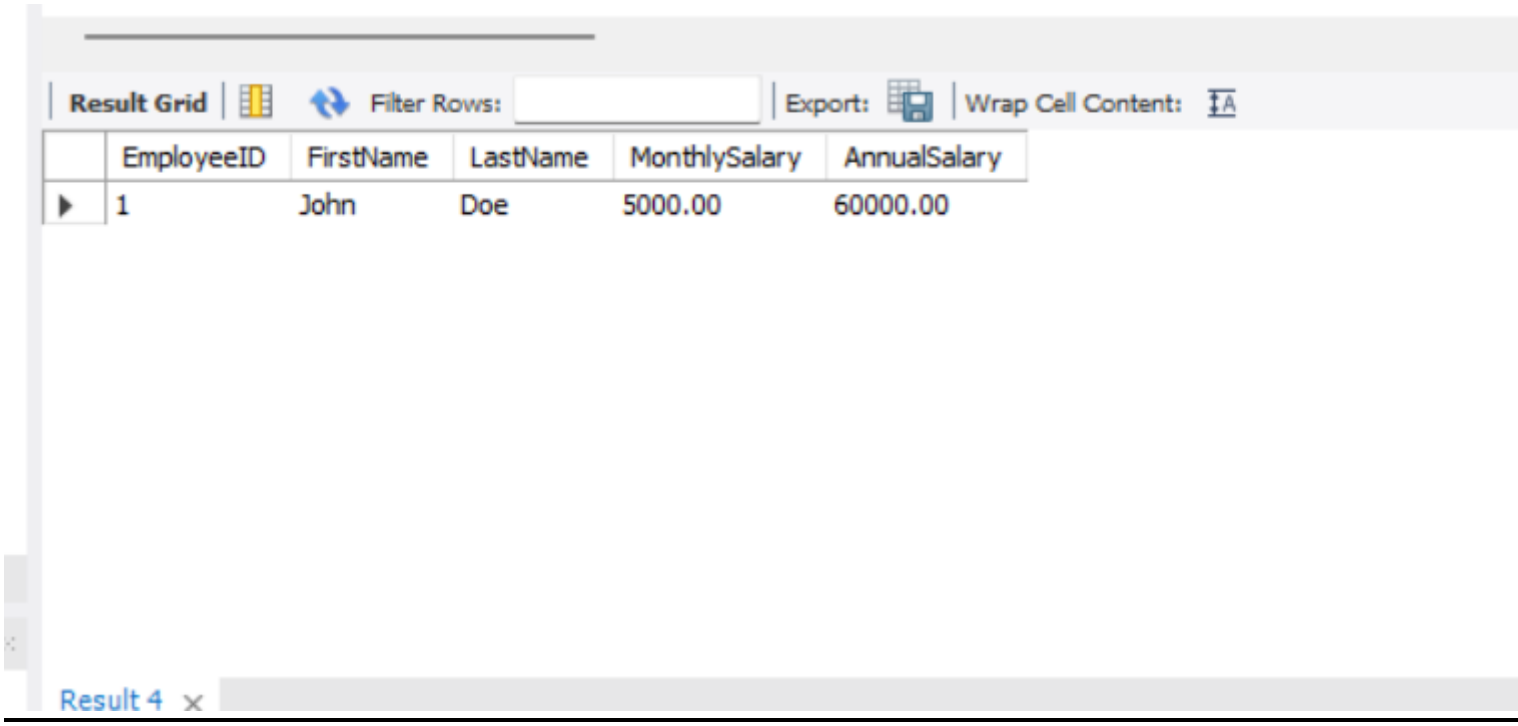
e.Salary AS MonthlySalary,

fn_CalculateAnnualSalary(e.EmployeeID) AS AnnualSalary

FROM Employees e

WHERE e.EmployeeID = 1;

OUTPUT:-



QUESTION-03)

Exercise 5: Return Data from a Stored Procedure

Goal: Create a stored procedure that returns the total number of employees in a

department.

Steps:

1. Define the stored procedure with a parameter for DepartmentID.

2. Write the SQL query to count the number of employees in the specified department.

3. Save the stored procedure by executing the Stored procedure content ?

Solution:-

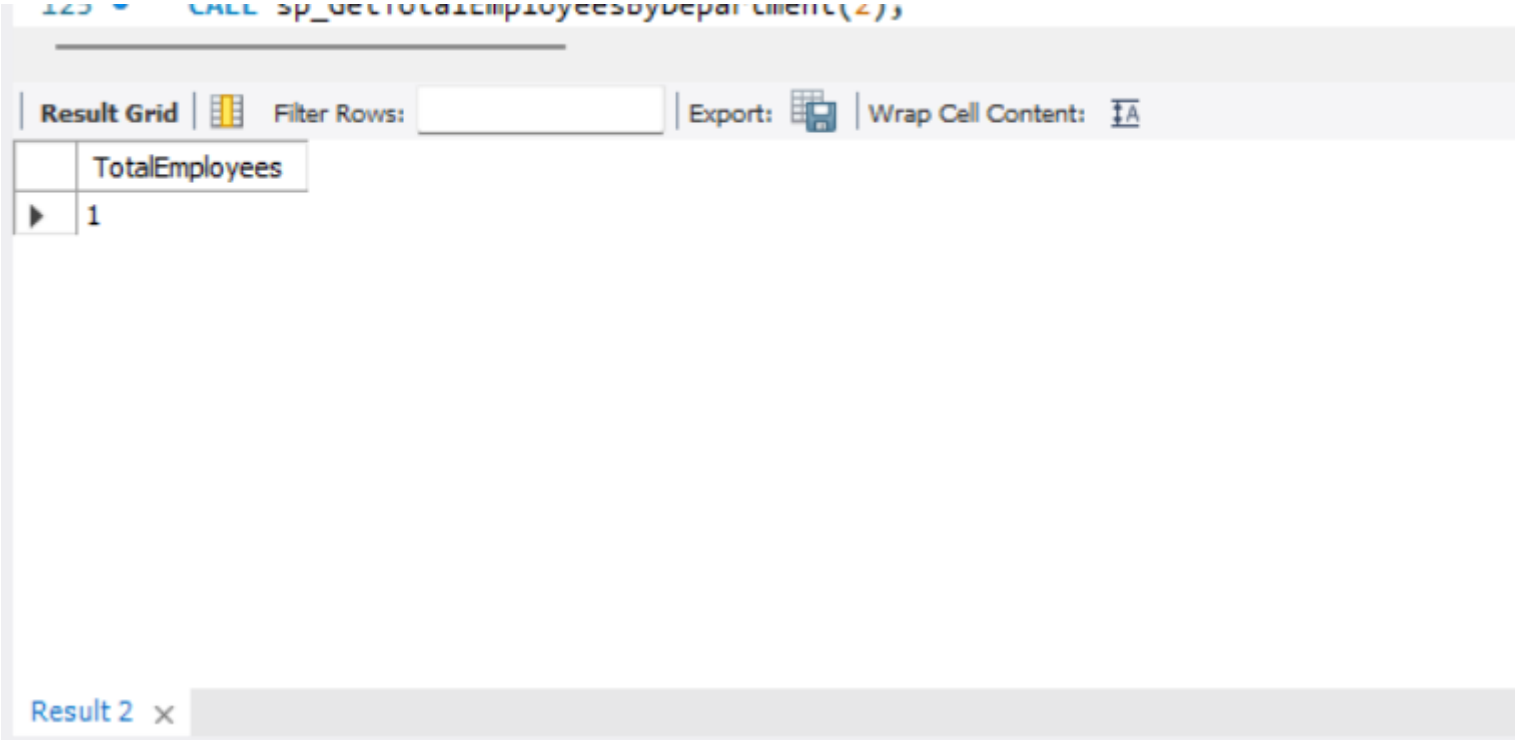
```
DELIMITER $$


CREATE PROCEDURE sp_GetTotalEmployeesByDepartment (
    IN dept_id INT
)
```

```
BEGIN
    SELECT COUNT(*) AS TotalEmployees
    FROM Employees
    WHERE DepartmentID = dept_id;
END $$

DELIMITER ;
```

QUERY :-

```
CALL sp_GetTotalEmployeesByDepartment(2);
```

OUTPUT:-



Exercise 4: Execute a Stored Procedure

Goal: Execute the stored procedure to retrieve employee details for a specific department.

Steps:

1. Write the SQL command to execute the stored procedure with a DepartmentID

parameter.

2. Execute the command and review the results ?

SOLUTION :-

```
CALL sp_GetEmployeesByDepartment(2);
```

OUTPUT :-