**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:**

* I created the database as “mydb”

use mydb;

CREATE TABLE Products (

ProductID INT PRIMARY KEY,

ProductName VARCHAR(100),

Category VARCHAR(50),

Price DECIMAL(10, 2)

);

* After then i used the “SQL Exercise - Index” file data for creating the table inside the database and inserting the items inside the table from.

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),

(5, 'Smartwatch', 'Accessories', 150.00),

(6, 'Charger', 'Accessories', 80.00),

(7, 'Monitor', 'Electronics', 800.00);

Now i setup the database with data to do exercise 1: ranking and window functions.

**1. Use ROW\_NUMBER() to assign a unique rank within each category.**

use mydb;

SELECT

ProductID,

ProductName,

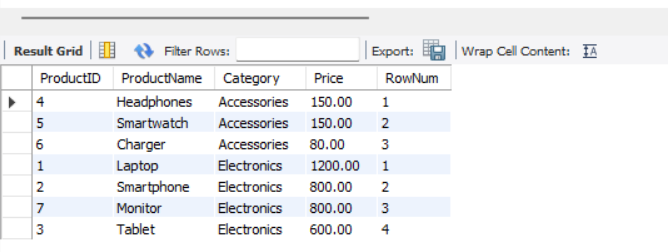
Category,

Price,

ROW\_NUMBER() OVER (PARTITION BY Category ORDER BY Price DESC) AS RowNum

FROM Products;

**Output:**



**2. Use RANK() and DENSE\_RANK() to compare how ties are handled.**

Here is the code for the rank() function

use mydb;

SELECT

ProductID,

ProductName,

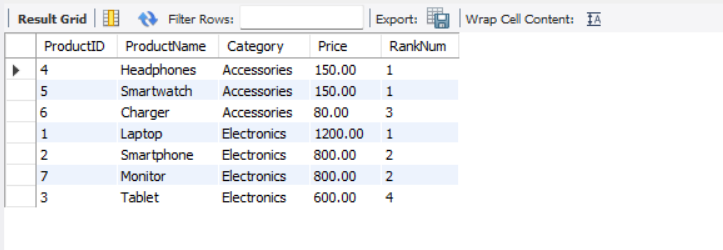
Category,

Price,

RANK() OVER (PARTITION BY Category ORDER BY Price DESC) AS RankNum

FROM Products;

**Output:**



Here is the code for the dense() function:

SELECT

ProductID,

ProductName,

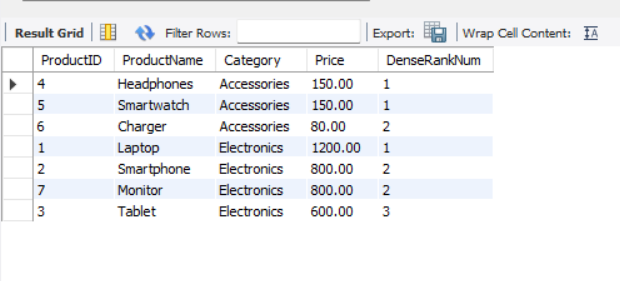
Category,

Price,

DENSE\_RANK() OVER (PARTITION BY Category ORDER BY Price DESC) AS DenseRankNum

FROM Products;

**Output:**



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

use mydb;

SELECT \*

FROM (

SELECT

ProductID,

ProductName,

Category,

Price,

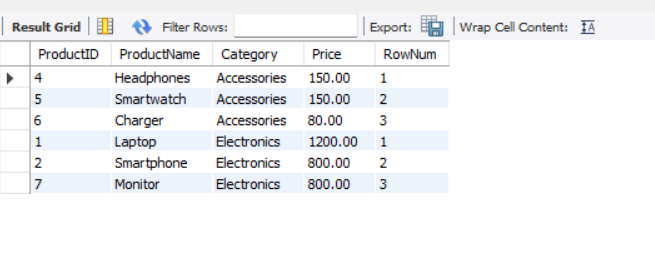
ROW\_NUMBER() OVER (PARTITION BY Category ORDER BY Price DESC) AS RowNum

FROM Products

) AS Ranked

WHERE RowNum <= 3;

**Output:**



**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.**

I created the new table for the Database Schema structure for an Employee Management System from hands-on learning.

**Departments Table**

CREATE TABLE Departments (

DepartmentID INT PRIMARY KEY,

DepartmentName VARCHAR(100)

);

Employees Table

CREATE TABLE Employees (

EmployeeID INT PRIMARY KEY,

FirstName VARCHAR(50),

LastName VARCHAR(50),

DepartmentID INT FOREIGN KEY REFERENCES Departments(DepartmentID),

Salary DECIMAL(10,2),

JoinDate DATE

);

**Sample Data**

**The following sample data can be used for testing:**

**Departments Sample Data**

INSERT INTO Departments (DepartmentID, DepartmentName) VALUES

(1, 'HR'),

(2, 'Finance'),

(3, 'IT'),

(4, 'Marketing');

Employees Sample Data

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');

**Solution:**

* I imported the Delimiter from the mysql. And i created the procedure of GetEmployeeCountByDepartment

USE mydb;

DELIMITER //

CREATE PROCEDURE GetEmployeeCountByDepartment(IN DeptID INT)

BEGIN

SELECT COUNT(\*) AS TotalEmployees

FROM Employees

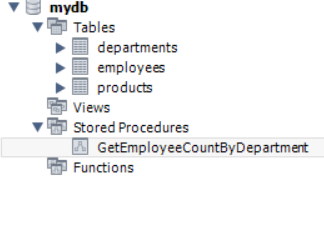
WHERE DepartmentID = DeptID;

END;

//

DELIMITER ;

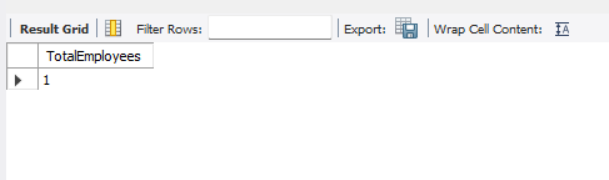
After created and store the procedure in the database “as mydb”



After storing i can call the procedure by its name getemployeecountbydeperatment.

**CALL GetEmployeeCountByDepartment(1);**

**Output:**

****

**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:**

**I created the procedure with the name GetEmployeesByDepartment**

USE mydb;

DELIMITER //

CREATE PROCEDURE GetEmployeesByDepartment(IN DeptID INT)

BEGIN

SELECT \*

FROM Employees

WHERE DepartmentID = DeptID;

END;

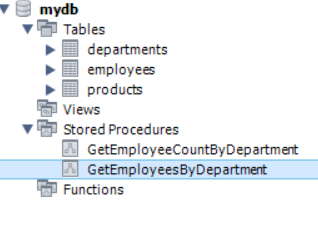
//

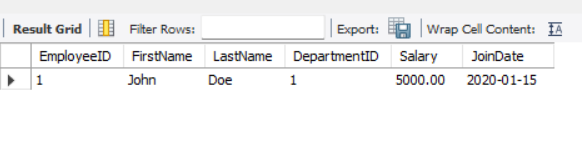
DELIMITER ;

//I call the procedure by its name

CALL GetEmployeesByDepartment(1);

**Output:**

****

****

**Exercise 4: Execute a Stored Procedure( same procedure in the above sql querry)**

**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.**

**Stored Procedure: InsertEmployee (Insert New Employee)**

USE mydb;

DELIMITER //

CREATE PROCEDURE sp\_InsertEmployee (

IN FirstName VARCHAR(50),

IN LastName VARCHAR(50),

IN DepartmentID INT,

IN Salary DECIMAL(10,2),

IN JoinDate DATE

)

BEGIN

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

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

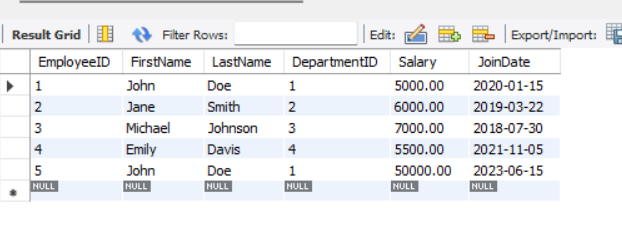
END;

//

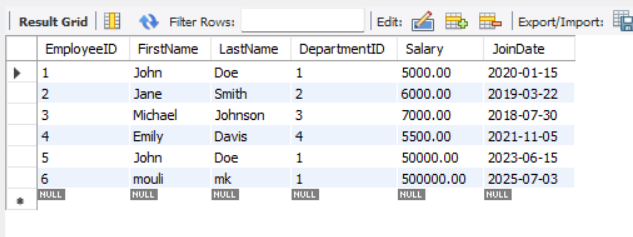
DELIMITER ;

CALL sp\_InsertEmployee('John', 'Doe', 1, 50000.00, '2023-06-15');

**Before insert the employee**



**After inserting employee**

****

**Exercise 7: Return Data from a Scalar Function**

**Goal: Return the annual salary for a specific employee using `fn\_CalculateAnnualSalary`.**

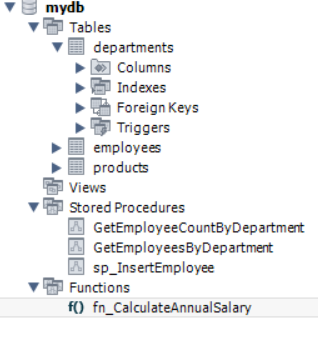
**Steps:**

**1. Execute the `fn\_CalculateAnnualSalary` function for an employee with `EmployeeID = 1`.**

**2. Verify the result.**

**Solution:**

I created the function for calculate the annual salary of the particular employee by using their employeeID



**use mydb;**

**DELIMITER //**

**CREATE FUNCTION fn\_CalculateAnnualSalary(EmpID INT)**

**RETURNS DECIMAL(10,2)**

**DETERMINISTIC**

**BEGIN**

**DECLARE AnnualSalary DECIMAL(10,2);**

**SELECT Salary \* 12 INTO AnnualSalary**

**FROM Employees**

**WHERE EmployeeID = EmpID;**

**RETURN AnnualSalary;**

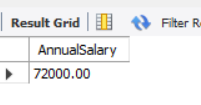
**END;**

**//**

**DELIMITER ;**

**SELECT fn\_CalculateAnnualSalary(2) AS AnnualSalary;**

**Output:**

****

**Request Note :** sorry for late submission because of my semester examination. I was delayed to submit my week 2 progress.