WEEK -2 HANDS ON

1. --Ranking and window functions.

Query:

DROP TABLE IF EXISTS Sales;

CREATE TABLE Sales (

ID INT,

Employee VARCHAR(50),

Department VARCHAR(10),

Sales INT

);

INSERT INTO Sales (ID, Employee, Department, Sales) VALUES

(1, 'Pritika', 'A', 5000),

(2, 'Riti', 'A', 7000),

(3, 'Rinki', 'B', 4000),

(4, 'Priti', 'B', 4000),

(5, 'Pakhi', 'A', 7000);

SELECT

Employee,

Department,

Sales,

ROW\_NUMBER() OVER (PARTITION BY Department ORDER BY Sales DESC) AS RowNum,

RANK() OVER (PARTITION BY Department ORDER BY Sales DESC) AS RankNum,

DENSE\_RANK() OVER (PARTITION BY Department ORDER BY Sales DESC) AS DenseRankNum,

NTILE(2) OVER (PARTITION BY Department ORDER BY Sales DESC) AS Quartile,

SUM(Sales) OVER (PARTITION BY Department ORDER BY Sales) AS RunningTotal,

LAG(Sales) OVER (PARTITION BY Department ORDER BY Sales) AS PrevSales,

LEAD(Sales) OVER (PARTITION BY Department ORDER BY Sales) AS NextSales,

FIRST\_VALUE(Sales) OVER (PARTITION BY Department ORDER BY Sales) AS FirstSale,

LAST\_VALUE(Sales) OVER (

PARTITION BY Department

ORDER BY Sales

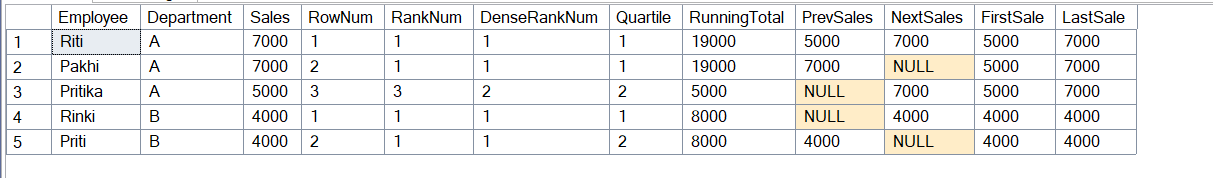
ROWS BETWEEN UNBOUNDED PRECEDING AND UNBOUNDED FOLLOWING

) AS LastSale

FROM Sales

ORDER BY Department, Sales DESC;

Output:



1. --Create a stored procedure.

Query:

IF OBJECT\_ID('dbo.Employees', 'U') IS NOT NULL

DROP TABLE dbo.Employees;

CREATE TABLE Employees (

ID INT,

Name VARCHAR(100),

Department VARCHAR(50),

Salary INT

);

INSERT INTO Employees (ID, Name, Department, Salary) VALUES

(1, 'Pritika', 'HR', 75000),

(2, 'Riti', 'IT', 70000),

(3, 'Rinki', 'Finance', 60000),

(4, 'Priti', 'IT', 75000),

(5, 'Pakhi', 'HR', 52000);

IF OBJECT\_ID('dbo.GetHighSalaryEmployees', 'P') IS NOT NULL

DROP PROCEDURE dbo.GetHighSalaryEmployees;

GO

CREATE PROCEDURE dbo.GetHighSalaryEmployees

@MinSalary INT

AS

BEGIN

SELECT ID, Name, Department, Salary

FROM Employees

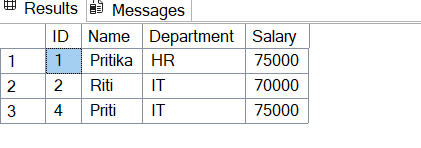
WHERE Salary > @MinSalary;

END;

GO

EXEC dbo.GetHighSalaryEmployees @MinSalary = 60000;

Output:



1. --Return data from a stored procedure.

Query:

IF OBJECT\_ID('dbo.PersonalDetails', 'U') IS NOT NULL

DROP TABLE dbo.PersonalDetails;

CREATE TABLE dbo.PersonalDetails (

ID INT,

Name VARCHAR(100),

Age INT,

City VARCHAR(100)

);

INSERT INTO dbo.PersonalDetails (ID, Name, Age, City) VALUES

(1, 'Pritika', 24, 'New York'),

(2, 'Riti', 25, 'Los Angeles'),

(3, 'Rinki', 35, 'Chicago'),

(4, 'Priti', 40, 'Houston'),

(5, 'Pakhi', 28, 'Seattle');

IF OBJECT\_ID('dbo.GetOlderPeople', 'P') IS NOT NULL

DROP PROCEDURE dbo.GetOlderPeople;

GO

CREATE PROCEDURE dbo.GetOlderPeople

@MinAge INT

AS

BEGIN

SELECT ID, Name, Age, City

FROM dbo.PersonalDetails

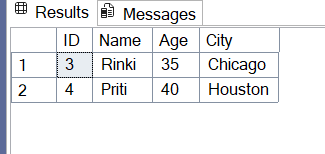
WHERE Age > @MinAge;

END;

GO

EXEC dbo.GetOlderPeople @MinAge = 30;

Output:



1. --Create an index.

Query:

IF OBJECT\_ID('dbo.PersonalDetails', 'U') IS NOT NULL

DROP TABLE dbo.PersonalDetails;

CREATE TABLE dbo.PersonalDetails (

ID INT,

Name VARCHAR(100),

Age INT,

City VARCHAR(100)

);

INSERT INTO dbo.PersonalDetails (ID, Name, Age, City) VALUES

(1, 'Pritika', 24, 'New York'),

(2, 'Riti', 25, 'Los Angeles'),

(3, 'Rinki', 35, 'Chicago'),

(4, 'Priti', 40, 'Houston'),

(5, 'Pakhi', 28, 'Seattle');

IF OBJECT\_ID('dbo.GetOlderPeople', 'P') IS NOT NULL

DROP PROCEDURE dbo.GetOlderPeople;

GO

CREATE PROCEDURE dbo.GetOlderPeople

@MinAge INT

AS

BEGIN

SELECT ID, Name, Age, City

FROM dbo.PersonalDetails

WHERE Age > @MinAge;

END;

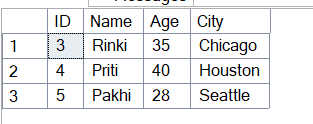
GO

EXEC dbo.GetOlderPeople @MinAge = 25;

CREATE INDEX idx\_age ON PersonalDetails(Age);

CREATE UNIQUE INDEX idx\_unique\_name ON PersonalDetails(Name);

Output:



1. --Functions-table valued function.

Query:

IF OBJECT\_ID('dbo.PersonalDetails', 'U') IS NOT NULL

DROP TABLE dbo.PersonalDetails;

CREATE TABLE dbo.PersonalDetails (

ID INT,

Name VARCHAR(100),

Age INT,

City VARCHAR(100)

);

INSERT INTO dbo.PersonalDetails (ID, Name, Age, City) VALUES

(1, 'Pritika', 24, 'New York'),

(2, 'Rinki', 25, 'Los Angeles'),

(3, 'Riti', 35, 'Chicago'),

(4, 'Priti', 40, 'Houston'),

(5, 'Pakhi', 28, 'Seattle');

IF OBJECT\_ID('dbo.GetOlderPeople', 'P') IS NOT NULL

DROP PROCEDURE dbo.GetOlderPeople;

GO

CREATE PROCEDURE dbo.GetOlderPeople

@MinAge INT

AS

BEGIN

SELECT ID, Name, Age, City

FROM dbo.PersonalDetails

WHERE Age > @MinAge;

END;

GO

--

IF OBJECT\_ID('dbo.GetPeopleAboveAge', 'IF') IS NOT NULL

DROP FUNCTION dbo.GetPeopleAboveAge;

GO

CREATE FUNCTION dbo.GetPeopleAboveAge(@MinAge INT)

RETURNS TABLE

AS

RETURN

(

SELECT ID, Name, Age, City

FROM dbo.PersonalDetails

WHERE Age > @MinAge

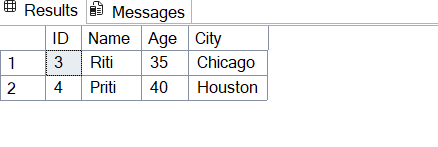
);

GO

-- Use the function like a table

SELECT \* FROM dbo.GetPeopleAboveAge(30);

Output:



1. --Retuen data from a scalar function.

Query:

IF OBJECT\_ID('dbo.GetFullName', 'FN') IS NOT NULL

DROP FUNCTION dbo.GetFullName;

GO

-- Create scalar function

CREATE FUNCTION dbo.GetFullName

(

@FirstName VARCHAR(50),

@LastName VARCHAR(50)

)

RETURNS VARCHAR(101)

AS

BEGIN

DECLARE @FullName VARCHAR(101);

SET @FullName = @FirstName + ' ' + @LastName;

RETURN @FullName;

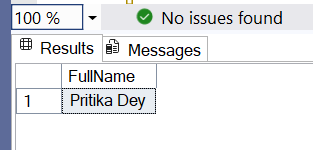
END;

GO

-- Use the function in a SELECT

SELECT dbo.GetFullName('Pritika', 'Dey') AS FullName;

Output:



1. --Create a stored procedure and execute.

Query:

IF OBJECT\_ID('dbo.PersonalDetails', 'U') IS NOT NULL

DROP TABLE dbo.PersonalDetails;

CREATE TABLE dbo.PersonalDetails (

ID INT,

Name VARCHAR(100),

Age INT,

City VARCHAR(100)

);

INSERT INTO dbo.PersonalDetails (ID, Name, Age, City) VALUES

(1, 'Pritika', 24, 'New York'),

(2, 'Riti', 25, 'Los Angeles'),

(3, 'Rinki', 35, 'Chicago'),

(4, 'Priti', 40, 'Houston'),

(5, 'Pakhi', 28, 'Seattle');

IF OBJECT\_ID('dbo.GetOlderPeople', 'P') IS NOT NULL

DROP PROCEDURE dbo.GetOlderPeople;

GO

CREATE PROCEDURE dbo.GetOlderPeople

@MinAge INT

AS

BEGIN

SELECT ID, Name, Age, City

FROM dbo.PersonalDetails

WHERE Age > @MinAge;

END;

GO

EXEC dbo.GetOlderPeople @MinAge = 30;

-- Drop the procedure if it already exists

IF OBJECT\_ID('dbo.GetPeopleByCity', 'P') IS NOT NULL

DROP PROCEDURE dbo.GetPeopleByCity;

GO

-- Create a stored procedure with one input parameter

CREATE PROCEDURE dbo.GetPeopleByCity

@City VARCHAR(100)

AS

BEGIN

SELECT ID, Name, Age, City

FROM dbo.PersonalDetails

WHERE City = @City;

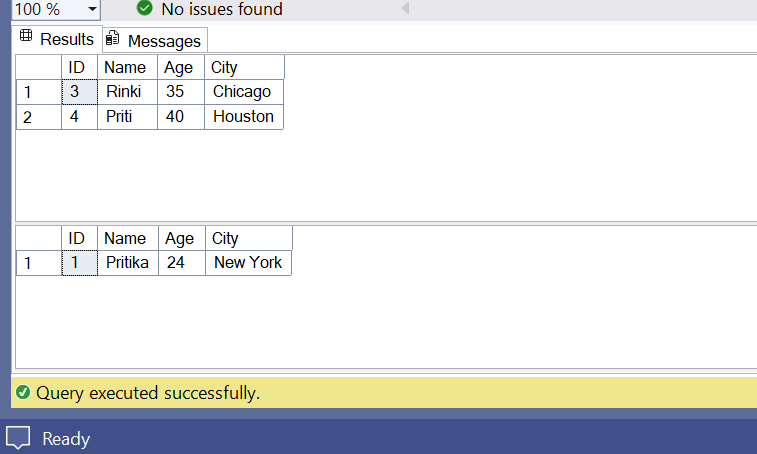
END;

GO

-- Execute the stored procedure with a parameter

EXEC dbo.GetPeopleByCity @City = 'New York';

Output:



1. Write a testable code with moq.

Code:

using System;

using Moq;

using NUnit.Framework;

public interface IPaymentGateway

{

bool ProcessPayment(string orderId, decimal amount);

}

public class Order

{

public string OrderId { get; set; }

public decimal Amount { get; set; }

}

public class OrderService

{

private readonly IPaymentGateway \_paymentGateway;

public OrderService(IPaymentGateway paymentGateway)

{

\_paymentGateway = paymentGateway;

}

public bool PlaceOrder(Order order)

{

return \_paymentGateway.ProcessPayment(order.OrderId, order.Amount);

}

}

[TestFixture]

public class OrderServiceTests

{

[Test]

public void PlaceOrder\_ShouldReturnTrue\_WhenPaymentIsSuccessful()

{

var mockPaymentGateway = new Mock<IPaymentGateway>();

mockPaymentGateway

.Setup(pg => pg.ProcessPayment(It.IsAny<string>(), It.IsAny<decimal>()))

.Returns(true);

var orderService = new OrderService(mockPaymentGateway.Object);

var testOrder = new Order { OrderId = "ORD123", Amount = 100.00m };

var result = orderService.PlaceOrder(testOrder);

Assert.IsTrue(result);

mockPaymentGateway.Verify(pg => pg.ProcessPayment("ORD123", 100.00m), Times.Once);

}

}

Output:

