Lab Task 13

**Task 1: Creating Functions and Procedures**

* Create a function called CalculateDiscount that takes the OrderTotal as input and returns the discount amount (10% of the order total).
* Create a function called **CalculateTax** that calculates the tax amount (5% of the total) for a given order total.
* Create a procedure called **InsertOrder** that takes input parameters for **CustomerID**, **ProductID**, and **Quantity**, and inserts a new order into the **Orders** table and updates the **OrderDetails** table accordingly.
* Write a script to execute the **UpdateInventory** procedure for **ProductID = 1** and **Quantity = 10** and verify that the inventory is updated correctly.

**Task 2: Indexes**

* Create a clustered index on the OrderDate column of the Orders table.
* Create a non-clustered index on the ProductID column of the OrderDetails table.
* Create a unique clustered index on the **CustomerID** column of the **Customers** table.
* Create a non-clustered index on the **LastName** column of the **Customers** table.
* Drop the indexes created in this task and compare the query performance before and after dropping the indexes.

**Task 3: Backup and Restore**

* Backup the current database.
* Restore the database from the backup file.
* Verify that the database has been successfully restored.
* Perform a full database backup and specify a backup location.
* Perform a differential backup and verify that only the changes since the last full backup are included.
* Restore the database to a point in time before the last backup and verify the data consistency.

**Task 4: Views**

* Create a view called ProductSales that displays the total sales amount for each product.
* Create a view called CustomerOrders that displays the total number of orders placed by each customer.
* Test both views to ensure they return the expected results.
* Create a view called **HighValueOrders** that displays orders with a total value greater than $500.
* Create a view called **ProductInventory** that displays the current inventory level for each product.
* Write a query to retrieve the total number of orders placed by each customer using the **CustomerOrders** view.

CREATE TABLE Customers (

CustomerID INT PRIMARY KEY,

FirstName NVARCHAR(50),

LastName NVARCHAR(50),

Email NVARCHAR(100)

);

CREATE TABLE Products (

ProductID INT PRIMARY KEY,

ProductName NVARCHAR(50),

Price DECIMAL(10, 2),

Inventory INT

);

CREATE TABLE Orders (

OrderID INT PRIMARY KEY,

CustomerID INT,

OrderDate DATE,

TotalAmount DECIMAL(10, 2),

CONSTRAINT FK\_CustomerID FOREIGN KEY (CustomerID) REFERENCES Customers(CustomerID)

);

CREATE TABLE OrderDetails (

OrderDetailID INT PRIMARY KEY,

OrderID INT,

ProductID INT,

Quantity INT,

CONSTRAINT FK\_OrderID FOREIGN KEY (OrderID) REFERENCES Orders(OrderID),

CONSTRAINT FK\_ProductID FOREIGN KEY (ProductID) REFERENCES Products(ProductID)

);

INSERT INTO Customers (CustomerID, FirstName, LastName, Email)

VALUES (1, 'John', 'Doe', 'john.doe@example.com');

INSERT INTO Products (ProductID, ProductName, Price, Inventory)

VALUES (1, 'Product A', 10.00, 100),

(2, 'Product B', 15.00, 50),

(3, 'Product C', 20.00, 75);

INSERT INTO Orders (OrderID, CustomerID, OrderDate, TotalAmount)

VALUES (1, 1, '2022-01-01', 25.00),

(2, 1, '2022-01-05', 30.00);

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

VALUES (1, 1, 1, 2),

(2, 1, 2, 1),

(3, 2, 3, 3);