1. Data Extraction and Basic Queries:

1a. Retrieve a list of all products with their names, product numbers, and list

prices.

Query: SELECT Name,ProductNumber,ListPrice from Production.product

Description:Retrieve product names, numbers, and list prices from the Production.Product table.

1b. List the top 10 most expensive products.

Query: SELECT TOP 10 Name,ProductNumber,ListPrice from Production.product ORDER BY ListPrice DESC

Description:Retrieve the top 10 products by name, product number, and list price in descending order from the Production.Product table.

1c. Find the total number of orders for each customer.

Query: SELECT t1.CustomerID,t2.FirstName AS CustomerName,COUNT(t3.SalesOrderID) AS TotalOrders

FROM Sales.Customer AS t1 JOIN Person.Person AS t2 ON t1.PersonID = t2.BusinessEntityID

JOIN Sales.SalesOrderHeader AS t3 ON t1.CustomerID = t3.CustomerID

GROUP BY t1.CustomerID, t2.FirstName ORDER BY TotalOrders DESC

Description:Retrieve the customer ID, first name, and the total number of orders for each customer, ordered by the total number of orders in descending order.

2. Advanced SQL Queries:

2a. Calculate the total sales for each product category.

Query: SELECT cat.Name AS ProductCategory, SUM(ord.TotalDue) AS TotalSales

FROM Sales.SalesOrderHeader ord JOIN

Sales.SalesOrderDetail det ON ord.SalesOrderID = det.SalesOrderID

JOIN

Production.Product prod ON det.ProductID = prod.ProductID

JOIN

Production.ProductSubcategory subcat ON prod.ProductSubcategoryID = subcat.ProductSubcategoryID

JOIN

Production.ProductCategory cat ON subcat.ProductCategoryID = cat.ProductCategoryID

GROUP BY

cat.Name

ORDER BY

TotalSales DESC;

Description:Retrieve the total sales amount for each product category from the AdventureWorks database.

2b. Identify the top 5 customers by total purchase amount.

Query: SELECT TOP 5 cust.CustomerID, SUM(ord.TotalDue) AS TotalPurchaseAmount

FROM Sales.Customer AS cust

JOIN Sales.SalesOrderHeader AS ord ON cust.CustomerID = ord.CustomerID

GROUP BY cust.CustomerID

ORDER BY TotalPurchaseAmount DESC;

Description:Retrieve the top 5 customers by their ID along with the total purchase amount they have spent, ordered from highest to lowest total purchase amount.

2c. Find the average order quantity for each product.

Query: SELECT t1.ProductID,AVG(t2.OrderQty) AS AverageOfProduct FROM Production.Product AS t1 JOIN Sales.SalesOrderDetail AS t2 ON t1.ProductID=t2.ProductID GROUP BY(t1.ProductID) ORDER BY AverageOfProduct DESC

Description:Retrieve the product ID and the average order quantity for each product, ordered by the average order quantity in descending order.

3. Data Manipulation and Analysis:

3a. Update the list price of all products in the 'Bikes' category by increasing it by

10%.

Query: UPDATE prod SET prod.ListPrice = prod.ListPrice \* 1.1

FROM Production.Product prod

JOIN Production.ProductSubcategory subcat ON prod.ProductSubcategoryID = subcat.ProductSubcategoryID

JOIN Production.ProductCategory cat ON subcat.ProductCategoryID = cat.ProductCategoryID

WHERE cat.Name = 'Bikes';

Description:Update the list prices of products belonging to the 'Bikes' category by increasing them by 10%.

3b.Delete all records of products that have never been sold.

Query: DELETE FROM Production.Product WHERE ProductID NOT IN (

SELECT ProductID

FROM Sales.SalesOrderDetail

);

Description:Delete products from the Production.Product table that have not been ordered, based on their absence in the Sales.SalesOrderDetail table.

3c. Insert a new product into the database.

INSERT INTO Production.Product (Name,ProductNumber,StandardCost,ListPrice,SellStartDate,ProductSubcategoryID,ProductModelID,ModifiedDate

)

VALUES ('Atlas cycle', 'NP-0002', 25.00,50.00, GETDATE(),22222,1000, GETDATE()

);

Description:Inserts a new product named 'Atlas cycle' into the Production.Product table with specified details including name, product number, standard cost, list price, sell start date, product subcategory ID, product model ID, and modified date.

4. SQL Optimization Techniques:

4a. Create an index on the SalesOrderDetail table to optimize queries for

retrieving sales order details by product ID.

Query: CREATE NONCLUSTERED INDEX SalesOrderDetail\_ProductID\_Assignment

ON Sales.SalesOrderDetail (ProductID);

Query: SELECT \*from Sales.SalesOrderDetail where ProductID=893

Description:Creates a nonclustered index on the ProductID column in Sales.SalesOrderDetail table for optimized querying and retrieval of records filtered by product ID

4b. Write a query to find all orders placed in the last 30 days and analyze its

execution plan.

Query: SELECT \*FROM Sales.SalesOrderHeader WHERE OrderDate BETWEEN '2014-06-01 00:00:00.000' AND '2014-06-30 00:00:00.000'

Description:Retrives the details of sales for 30 days

5. Working with Indexes and Views:

5a. Create a view that shows the total sales amount for each salesperson.

Query: CREATE VIEW SPTotalSales AS

SELECT sp.BusinessEntityID AS SalesPersonID,

person.FirstName AS SalesPersonName,

SUM(soh.TotalDue) AS TotalSalesAmount

FROM Sales.SalesPerson AS sp

JOIN Sales.SalesOrderHeader AS soh ON sp.BusinessEntityID = soh.SalesPersonID

JOIN Person.Person AS person ON sp.BusinessEntityID = person.BusinessEntityID

GROUP BY sp.BusinessEntityID, person.FirstName, person.LastName;

Query: SELECT \*FROM SPTOtalSales

Description:Create a view from a table for abstraction

5b.Use this view to find the top 3 salespersons by total sales.

Query: SELECT TOP 3 \*FROM SPTotalSales ORDER BY TotalSalesAmount DESC

Description:The Above query fetches top 3 salespersons by their total sales

7. Stored Procedures and Triggers:

7a.Write a stored procedure to generate a sales report for a given date range.

Query: CREATE PROCEDURE SalesReportASSIGNMENT @StartDate DATE,@EndDate DATE

AS BEGIN

SELECT soh.OrderDate, p.Name AS ProductName, sod.OrderQty, sod.LineTotal

FROM Sales.SalesOrderHeader AS soh

JOIN Sales.SalesOrderDetail AS sod ON soh.SalesOrderID = sod.SalesOrderID

JOIN Production.Product AS p ON sod.ProductID = p.ProductID

WHERE soh.OrderDate BETWEEN @StartDate AND @EndDate;

END;

GO

DECLARE @start\_date DATE = '2011-07-01', @end\_date DATE = '2012-07-31';

EXEC GenerateSalesReport @StartDate = @start\_date, @EndDate = @end\_date;

Description:A Procedure is set of predefined sql statement which can promote code reusability

7b. Create a trigger that automatically updates the stock quantity when a new sales order is placed.

Query: CREATE TRIGGER after\_stock

ON Sales.SalesOrderDetail

AFTER INSERT

AS

BEGIN

UPDATE Production.ProductInventory

SET Quantity = Quantity + i.OrderQty

FROM Production.ProductInventory pi

JOIN inserted i ON pi.ProductID = i.ProductID;

END;

Description:To automate the updations trigger is used

8. ETL and Data Processing:

8a. Build a simple ETL pipeline that extracts customer data, transforms it by

standardizing phone number formats, and loads it into a new table.

Query: SELECT PhoneNumberTypeID,PhoneNumber FROM Person.PersonPhone;

SELECT PhoneNumberTypeID,CONCAT('+1 (', SUBSTRING(PhoneNumber, 1, 3), ') ', SUBSTRING(PhoneNumber, 4, 3), '-', SUBSTRING(PhoneNumber, 7, 4)) AS FormattedPhoneNumber FROM Person.PersonPhone;

CREATE TABLE TransformedPersonPhone (

PhoneNumberTypeID INT,

FormattedPhoneNumber VARCHAR(20)

);

INSERT INTO TransformedPersonPhone (PhoneNumberTypeID, FormattedPhoneNumber)

SELECT PhoneNumberTypeID,CONCAT('+1 (', SUBSTRING(PhoneNumber, 1, 3), ') ', SUBSTRING(PhoneNumber, 4, 3), '-', SUBSTRING(PhoneNumber, 7, 4)) FROM Person.PersonPhone;

Description:ETL Simulated using the SQL

8b. Design a data model for a reporting database that consolidates sales, product,and customer data.

Query:CREATE DATABASE SalesDetails;

CREATE TABLE SalesCustomers (

CustomerID INT PRIMARY KEY,

FirstName VARCHAR(50),

LastName VARCHAR(50),

Email VARCHAR(100),

Phone VARCHAR(20),

Address VARCHAR(100),

City VARCHAR(50),

StateProvince VARCHAR(50),

PostalCode VARCHAR(20)

);

CREATE TABLE SalesProducts (

ProductID INT PRIMARY KEY,

ProductName VARCHAR(100),

Description VARCHAR(255),

CategoryName VARCHAR(50),

Price DECIMAL(10, 2),

SupplierName VARCHAR(100),

Weight DECIMAL(8, 2),

ProductDimensions VARCHAR(50)

);

CREATE TABLE SalesOrders (

OrderID INT PRIMARY KEY,

CustomerID INT,

ProductID INT,

OrderDate DATE,

Quantity INT,

TotalPrice DECIMAL(10, 2),

PaymentType VARCHAR(50),

ShippingAddress VARCHAR(100),

FOREIGN KEY (CustomerID) REFERENCES SalesCustomers(CustomerID),

FOREIGN KEY (ProductID) REFERENCES SalesProducts(ProductID)

);

Description:Design model for the requirement

9.Reporting and Data Visualization:

9a.Create a basic report showing monthly sales trends for the current year







Description:Different visualizations of the same data to get insights

9b.Develop a visualization that displays the distribution of product categories in terms of total sales.

