**DATA ENGINEERING**

**Assignment-2**

**\*\*Introduction to SQL:**

**1. Write a query to retrieve the first 10 records from the Person.Person table.**

Query: Select Top 10 \* from Person.Person

Description: Using Top Clasuse to retrieve data.

**2. Write a query to retrieve all columns from the Production.Product table where the Color is**

**'Red'**

Query: Select \* from Production.Product where Color = 'red'

Description: Using Where Keyword to filter the data based on color.

**\*\*Advanced SQL Queries:**

**1. Create a query to list the ProductID and the average ListPrice from the Production.Product table, grouped by ProductID.**

Query: Select p.ProductID,Avg(ListPrice) As Average from Production.Product p group by P.ProductID

Description: Using Avg Aggregate function and calculating Average List Price and grouping by ProductID

**2. Create a query to list the DepartmentID and the number of employees in each department from the HumanResources.EmployeeDepartmentHistory table, grouped by DepartmentID.**

Query:

Select e.DepartmentID,count(e.DepartmentID) As countOfDept from HumanResources.EmployeeDepartmentHistory e group by e.DepartmentID

Description:Using Count Aggregate function to count number of employees and grouping by DepartmentID

**\*\*SQL for Data Engineering:**

**1. Write a query to join the Sales.SalesOrderHeader and Sales.SalesOrderDetail tables to get a list of sales order numbers and their respective product IDs.**

Query:

Select a.SalesOrderID,b.ProductID from Sales.SalesOrderHeader a Inner Join Sales.SalesOrderDetail b On a.SalesOrderID = b.SalesOrderID

Description: Joining two tables and retrieving the needed information

**2. Write a query to join the Sales.Customer and Person.Person tables to get a list of customer names and their email addresses.**

Query:

Select b.FirstName,b.LastName,c.EmailAddress from Sales.Customer a Inner Join Person.Person b On a.PersonID = b.BusinessEntityID Inner Join Person.EmailAddress c On b.BusinessEntityID = c.BusinessEntityID

Description: Joining three Tables Sales.Customer,Person.Person and Person.EmailAddress to get required information.

**\*\*Writing Queries for Data Manipulation and Analysis:**

**1. Create a query to update the PhoneNumber in the Person.PersonPhone table where BusinessEntityID is 5.**

Query: Update Person.PersonPhone Set PhoneNumber = '111-222-3333' where BusinessEntityID = 5

Description: Updating phone number using BusinessEntityID.

**2. Write a query to delete records from the Sales.ShoppingCartItem table where the Quantity is 1.**

**Query:** Delete from Sales.ShoppingCartItem where Quantity = 3;

Description: Deleting Records whose Quantity is equal to 3.

**\*\*SQL Optimization Techniques:**

**1. Optimize the following query to improve performance:**

**Query: SELECT \* FROM Sales.SalesOrderHeader WHERE OrderDate = '2022-01-01';**

Query:Select DueDate,ShipDate from Sales.SalesOrderHeader where OrderDate = '2022-01-01'

**2. Analyze and optimize the following query:**

**Query: SELECT \* FROM Production.Product WHERE ListPrice > 1000;**

Query : Select Name,ListPrice From Production.Product where ListPrice>1000

**\*\*Working with Indexes and Views:**

**1. Create an index on the SalesOrderNumber column in the Sales.SalesOrderHeader table.**

Query:

Create Index Ind\_Sales On Sales.SalesOrderHeader(SalesOrderNumber)

Select SalesOrderID,SalesOrderNumber,OrderDate From Sales.SalesOrderHeader With(Index(Ind\_Sales)) where SalesOrderNumber = '5043665'

Description:Creating an Index on the SalesOrderNumber column in the Sales.SalesOrderHeader table

**2. Drop an existing index named IX\_Product\_Name on the Production.Product table.**

Query:Drop Index Ind\_Sales On Sales.SalesOrderHeader

Description: Droping the existing Index.

**\*\*Creating and Using Indexes and Views:**

**1. Create a view that shows the ProductID, Name, and ProductNumber from the Production.Product table.**

Query:

Create View vi\_Product As Select ProductID,Name,ProductNumber From Production.Product

Select \* from vi\_Product

Description: Creating views based on Production.Product Table and Retrieving the information.

**2. Create a view that joins the Sales.SalesOrderHeader and Sales.SalesOrderDetail tables and includes columns for SalesOrderID, OrderDate, and TotalDue.**

Query:

Create View vi\_Sales As Select a.SalesOrderID,a.OrderDate,a.TotalDue from Sales.SalesOrderHeader a Join Sales.SalesOrderDetail b On a.SalesOrderID = b.SalesOrderID

Select \* from vi\_Sales

Description:Creating Views on joining two tables and retrieving the required information.

**\*\*Transactions and Concurrency Control:**

**1. Write a transaction to transfer $500 from account A to account B ensuring ACID properties.**

Query:

Begin Transaction;

Declare @cash money =500;

Update Person.Account Set amount = amount+@cash Where Person\_name='A'

Update Person.Account Set amount = amount-@cash Where Person\_name='B'

Commit Transaction;

Description:Writing a transaction to transfer 500 from account A to account B ensuring ACID Properties

**2. Write a transaction to insert a new record into the Sales.SpecialOffer table and update the ModifiedDate in the Production.Product table, ensuring both operations are atomic.**

Query: Begin Transaction;

Insert Into Sales.SpecialOffer (Description, DiscountPct,Type,Category,

StartDate, EndDate,MinQty,MaxQty,rowguid,ModifiedDate)

Values ( 'New Offer', 10, 'Volume Discount','Customer',GetDate(), DateAdd(DAY, 30, GetDate()),15,24,'1290C4F5-191F-4337-AB6B-0A2DDE03CBF9', '2011-04-01 00:00:00.000');

Update Production.Product

Set ModifiedDate = GetDate()

Where ProductID = '007';

Commit Transaction;

Description:Transaction to insert a new record into the Table and update Modified Date in the table

**3. Explain what happens when two transactions try to update the same record simultaneously and how SQL Server handles it.**

**-**The second transaction will attempts to acquire an exclusive lock for its update. Because both transactions are converting to exclusive locks, and they are each waiting for the other transaction to release its shared lock, so a deadlock occurs.

**\*\*Stored Procedures**

**1. Create a stored procedure to insert a new product into the Production.Product table.**

Query:

Create Procedure InsProduct

@Name NVARCHAR(255),

@ProductNumber NVARCHAR(25),

@ListPrice MONEY,

@Color NVARCHAR(15),

@Size NVARCHAR(10),

@Weight DECIMAL(8, 2),

@ProductCategoryID INT,

@ProductModelID INT

AS

BEGIN

Insert into Production.Product (Name, ProductNumber, ListPrice, Color, Size, Weight, ProductSubcategoryID, ProductModelID)

values (@Name, @ProductNumber, @ListPrice, @Color, @Size, @Weight, @ProductCategoryID, @ProductModelID);

end

**2. Create a stored procedure to update the StandardCost of a product in the Production.Product table based on the ProductID and a new cost parameter.**

Query:

Create Procedure UpdProduct

@PID INT,

@NewCost Money

As

Begin

Update Production.Product Set StandardCost = @NewCost where ProductID = @PID

end

exec UpdProduct @PID = 1,@NewCost = 100

**\*\*Writing Triggers**

**1. Write a trigger that updates the ModifiedDate column in the Production.Product table whenever a record is updated.**

Query:

Create Trigger trg\_Product On Production.Product

After Update

As

Begin

Update Production.Product Set ModifiedDate = GetDate();

End

**2. Write a trigger to log changes to the ListPrice in the Production.Product table into a new table called ProductPriceHistory.**

Query:create table #tempV(

productID int,

listPrice money

)

Create Trigger Up\_no

On Production.Product

After update

As

Begin

Insert into #tempV(productID,listPrice)

select d.ProductID,d.ListPrice from deleted d

End

**\*\*Slowly Changing Dimensions (SCD) and Change Data Capture (CDC)**

**1. Describe a scenario in the Adventure Works database where a Slowly Changing Dimension (SCD) Type 2 would be appropriate.**

In HumanResources.EmployeeDepartmentHistory table SCD-2 can be used to track DepartmentID,Start and end date.

**\*\*ETL (Extract, Transform, Load):**

**1. Outline the steps to extract data from the Sales.SalesOrderHeader table, transform it to aggregate sales by year, and load it into a new table called YearlySales.**

**Extracting Date:**

Select \* from Sales.SalesOrderHeader

**Creating New Table:**

Create Table newSales(

Year Int,

TotalSales Decimal(12,2)

)

**Transforming and Loading:**

Insert into newSales(Year,TotalSales)

Select Year(OrderDate) As year,Sum(TotalDue) As TotalSales from Sales.SalesOrderHeader group by Year(OrderDate)

Select \* from newSales

**2. Describe the ETL process and its importance in data warehousing, using examples from the Adventure Works database.**

**ETL(Extraction,Transformation,Loading)**:ETL is the process of combining data from multiple sources into a large, central repository called a data warehouse. ETL uses a set of business rules to clean and organize raw data and prepare it for storage, data analytics.

Extracting data involves retrieving it from various sources such as databases, files, APIs, or web scraping. For Adventure Works, this includes:

Transforming data prepares extracted data for storage in the data warehouse:

- Data Cleaning and Standardization: Removes duplicates, handles missing values, and ensures consistent formats.

- Integration and Aggregation: Combines and aggregates data to a unified granularity:

- Aggregating sales data to compute monthly or customer-level totals.

- Enriching customer profiles by integrating sales data with customer details.

Loading data involves inserting transformed data into the target database or data warehouse:

Importance in Data Warehousing

ETL is pivotal for effective data warehousing:

- Data Integration: Integrates diverse data sources, ensuring coherence and consistency.

- Data Quality: Enhances data accuracy and reliability through rigorous transformation processes.

- Business Intelligence: Supports strategic insights:

Example: Adventure Works

In Adventure Works, ETL optimizes data utilization:

Extract: Data pulled from transactional databases and external sources.

Transform:

- Aggregates sales data for strategic insights.

- Standardizes product categories and calculates performance metrics.

Load: Transforms data loaded into dimensional models for enhanced analysis and reporting.

Implementing robust ETL processes empowers Adventure Works to leverage data effectively, driving informed decisions and operational efficiencies through integrated, high-quality data insights.

**3. Write a SQL script to extract data from the Sales.Customer table, transform it to calculate the TotalSpend for each customer, and load it into a new table called CustomerSpend.**

**Extraction:** Select \* from Sales.Customer

**Transformation**:

create table CustomerSpend1 (

CustomerID int primary key,

TotalSpend money

);

**Loading:**

insert into CustomerSpend1 (CustomerID, TotalSpend)

select

c.CustomerID,

sum(soh.TotalDue) as TotalSpend

from

Sales.Customer c

inner join

Sales.SalesOrderHeader soh on c.CustomerID = soh.CustomerID

group by

c.CustomerID

select \* from CustomerSpend

**4. Write a query to extract data from the Production.Product table, transform it by adding a 10% markup to the ListPrice, and insert it into a new table called ProductWithMarkup.**

**Extraction:**

Select \* from Production.Product;

**Transformation:**

create table Product12 (

ProductID int primary key,

ListPrice money

);

Update Product12 Set ListPrice = 1.1\*ListPrice

**Loading:**

Insert into Product12 (ProductID,ListPrice)

Select ProductID,ListPrice from Production.Product

**\*\*Reporting and Data Visualization**

**1. Write a query to generate a report showing the total sales amount by TerritoryID from the Sales.SalesOrderHeader table.**

**Query:** Select TerritoryID,Sum(TotalDue) As Total from Sales.SalesOrderHeader group by TerritoryID



**2. Create a simple report that lists the top 5 products by sales amount using data from the Sales.SalesOrderDetail table.**



**DashBoard:**



**3. Describe how to connect Power BI to the Adventure Works database and load data from the Sales.SalesOrderHeader and Sales.SalesOrderDetail tables.**

Open Power BI Desktop application.

Get Data:

Click on "Get Data" from the Home tab.

Choose "SQL Server" from the list of data sources.

Connect to Database:

Enter the server name where your Adventure Works database is hosted.

Select the Adventure Works database from the list.

Choose authentication method (Windows or database credentials).

Load Data:

Select the tables SalesOrderHeader and SalesOrderDetail from the Navigator window.

Click "Load" to load data into Power BI.