Project Title:

End-to-End Sales Data Warehouse and Visualization Solution

Project Overview:

The project demonstrates the complete process of designing and implementing a data warehouse using SQL Server Management Studio (SSMS), performing data extraction, transformation, and loading (ETL) using Python, and creating interactive business intelligence reports using Power BI. The project focuses on centralizing sales data from the AdventureWorks database and providing insightful visualizations for decision-making.

Technologies Used:

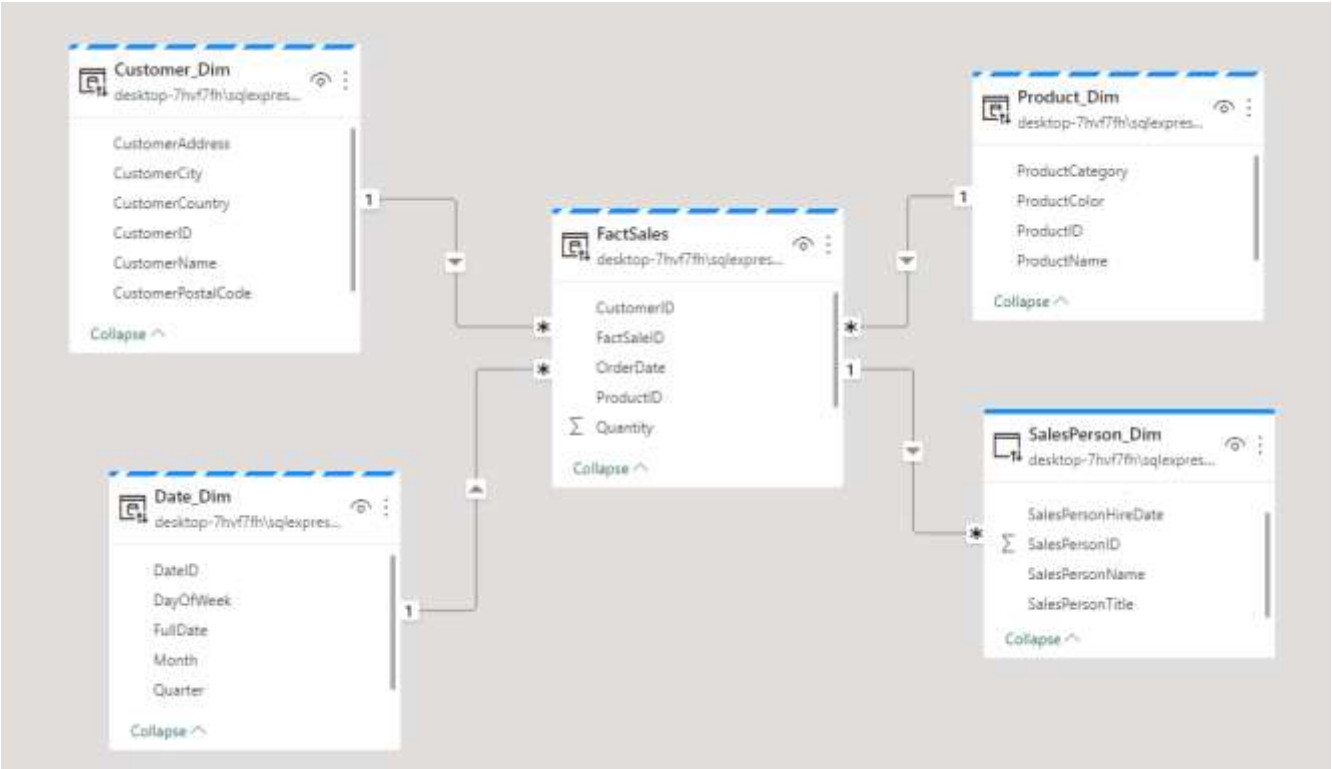
- **Database & Data Warehousing**: SQL Server Management Studio (SSMS)
- **ETL Process**: Python
- **Data Visualization**: Power BI

Project Phases:

1. Data Warehouse Design

- **Source Database**: AdventureWorks
- **Objective**: Design a data warehouse optimized for reporting and analytics.
- **Schema**: Star schema was chosen to streamline querying.
- **Fact Table**:
  - **Table Name**: FactSales
  - **Description**: Contains transactional sales data such as order amount, order date, and customer ID.
- **Dimension Tables**:
  - **Table Name 1**: Product_Dim
  - **Table Name 2**: Customer_Dim

- **Table Name 3**: SalesPerson_Dim
- **Table Name 4**: Date_Dim
- **Description**: Includes supporting tables such as Customer, Product, Date, and Sales Person.



2. ETL Process

- **Objective**: Extract data from the AdventureWorks database and load the data into the data warehouse.
- **ETL Tool**: Python
- **ETL Workflow**:
  - **Extract**: Data extracted from the AdventureWorks database.
  - **Load**: Load transformed data into the data warehouse.
- **ETL Script**:

Fact Table ETL:

jupyter ETL_Script Last Checkpoint: 24 minutes ago (unsaved changes)

File Edit View Insert Cell Kernel Widgets Help Trusted Python 3 (ipykernel)

In [ ]:
```
import numpy as np
import pandas as pd
import pyodbc
import sqlalchemy
from sqlalchemy import create_engine

s_user = 'localhost'
s_pwd = '*********'
s_db = 'AdventureWorks2019'
s_server = '************'
s_driver = '{***************}'

d_user = 'localhost'
d_pwd = '**********'
d_db = 'AdventureWorksDW-2024'
d_server = '************'
d_driver = '******************'

s_conn_str= f'DRIVER={s_driver};SERVER={s_server};DATABASE={s_db};UID={s_user};PWD={s_pwd}'
query = '''SELECT SalesOrderHeader.SalesOrderID,
       ProductID,
       CustomerID,
       SalesPersonID,
       CAST(CONVERT(varchar, OrderDate, 112) AS INT) AS OrderDate,
       TotalDue AS SalesAmount,
       OrderQty AS Quantity
FROM Sales.SalesOrderDetail
JOIN Sales.SalesOrderHeader ON SalesOrderDetail.SalesOrderID = SalesOrderHeader.SalesOrderID;

'''

s_conn = pyodbc.connect(s_conn_str)

df = pd.read_sql(query, s_conn)

df =df.drop_duplicates(subset='CustomerID')

d_conn_str = f'mssql+pyodbc://{d_user}:{d_pwd}@{d_server}/{d_db}?driver={d_driver}'

engine = create_engine(d_conn_str)

try:
    d_conn_str = f'mssql+pyodbc://{d_user}:{d_pwd}@{d_server}/{d_db}?driver={d_driver}'
    engine = create_engine(d_conn_str)
    print('Destination Connection build Successfully')
except Exception as e:
    print(f'There is an Error while Connecting to Destination: {e}')

df.to_sql('FactSales', con=engine, if_exists='append', index=False)
```

Customer Table ETL:

jupyter ETL_Script Last Checkpoint: 25 minutes ago (autosaved)

File Edit View Insert Cell Kernel Widgets Help Notebook saved Trusted Python 3 (ipykernel)

In [ ]:

```python
import numpy as np
import pandas as pd
import pyodbc
import sqlalchemy
from sqlalchemy import create_engine

s_user = 'localhost'
s_pwd = '*********'
s_db = 'AdventureWorks2019'
s_server = '************'
s_driver = '{***************}'

d_user = 'localhost'
d_pwd = '**********'
d_db = 'AdventureWorksDW-2024'
d_server = '************'
d_driver = '******************'

s_conn_str= f'DRIVER={s_driver};SERVER={s_server};DATABASE={s_db};UID={s_user};PWD={s_pwd}'
query = '''SELECT CustomerID, FirstName + ' ' + LastName CustomerName, AddressLine1 CustomerAddress,
City CustomerCity, StateProvince.Name CustomerState, CountryRegion.Name CustomerCountry, PostalCode CustomerPostalCode
FROM Sales.Customer
JOIN Person.Person ON Customer.PersonID = Person.BusinessEntityID
JOIN Person.BusinessEntityAddress ON Person.BusinessEntityID = BusinessEntityAddress.BusinessEntityID
JOIN Person.Address ON BusinessEntityAddress.AddressID = Address.AddressID
JOIN Person.StateProvince ON Address.StateProvinceID = StateProvince.StateProvinceID
JOIN Person.CountryRegion ON StateProvince.CountryRegionCode = CountryRegion.CountryRegionCode;'''

s_conn = pyodbc.connect(s_conn_str)

df = pd.read_sql(query, s_conn)

df =df.drop_duplicates(subset='CustomerID')

d_conn_str = f'mssql+pyodbc://{d_user}:{d_pwd}@{d_server}/{d_db}?driver={d_driver}'

engine = create_engine(d_conn_str)

try:
    d_conn_str = f'mssql+pyodbc://{d_user}:{d_pwd}@{d_server}/{d_db}?driver={d_driver}'
    engine = create_engine(d_conn_str)
    print('Destination Connection build Successfully')
except Exception as e:
    print(f'There is an Error while Connecting to Destination: {e}')

df.to_sql('FactSales', con=engine, if_exists='append', index=False)
```

Product Table ETL:

jupyter ETL_Script Last Checkpoint: 17 minutes ago (unsaved changes)

File Edit View Insert Cell Kernel Widgets Help

In [ ]:

```python
import numpy as np
import pandas as pd
import pyodbc
import sqlalchemy
from sqlalchemy import create_engine

s_user = 'localhost'
s_pwd = '*********'
s_db = 'AdventureWorks2019'
s_server = '************'
s_driver = '{***************}'

d_user = 'localhost'
d_pwd = '***********'
d_db = 'AdventureWorksDW-2024'
d_server = '************'
d_driver = '******************'

s_conn_str= f'DRIVER={s_driver};SERVER={s_server};DATABASE={s_db};UID={s_user};PWD={s_pwd}'
query = ''' SELECT
        p.ProductID,
        p.Name AS ProductName,
        pc.Name AS ProductCategory,
        psc.Name AS ProductSubcategory,
        p.Color AS ProductColor
    FROM
        Production.Product p
        JOIN Production.ProductSubcategory psc ON p.ProductSubcategoryID = psc.ProductSubcategoryID
        JOIN Production.ProductCategory pc ON psc.ProductCategoryID = pc.ProductCategoryID;'''

s_conn = pyodbc.connect(s_conn_str)

df = pd.read_sql(query, s_conn)

df =df.drop_duplicates(subset='CustomerID')

d_conn_str = f'mssql+pyodbc://{d_user}:{d_pwd}@{d_server}/{d_db}?driver={d_driver}'

engine = create_engine(d_conn_str)

try:
    d_conn_str = f'mssql+pyodbc://{d_user}:{d_pwd}@{d_server}/{d_db}?driver={d_driver}'
    engine = create_engine(d_conn_str)
    print('Destination Connection build Successfully')
except Exception as e:
    print(f'There is an Error while Connecting to Destination: {e}')

df.to_sql('Product_Dim', con=engine, if_exists='append', index=False)
```

SalesPerson Table ETL:

jupyter ETL_Script Last Checkpoint: 20 minutes ago (unsaved changes)

File Edit View Insert Cell Kernel Widgets Help

Trusted | Python 3 (ipykernel)

Run Code

In [ ]:
```
import numpy as np
import pandas as pd
import pyodbc
import sqlalchemy
from sqlalchemy import create_engine

s_user = 'localhost'
s_pwd = '*********'
s_db = 'AdventureWorks2019'
s_server = '************'
s_driver = '{***************}'

d_user = 'localhost'
d_pwd = '**********'
d_db = 'AdventureWorksDW-2024'
d_server = '************'
d_driver = '******************'

s_conn_str= f'DRIVER={s_driver};SERVER={s_server};DATABASE={s_db};UID={s_user};PWD={s_pwd}'
query = '''SELECT Employee.BusinessEntityID SalesPersonID, FirstName + ' ' + LastName SalesPersonName,
JobTitle SalesPersonTitle, HireDate SalesPersonHireDate, Department.Name SalesPersonDepartment
FROM HumanResources.Employee
JOIN Person.Person ON Employee.BusinessEntityID = Person.BusinessEntityID
JOIN HumanResources.EmployeeDepartmentHistory ON Employee.BusinessEntityID = EmployeeDepartmentHistory.BusinessEntityID
JOIN HumanResources.Department ON EmployeeDepartmentHistory.DepartmentID = Department.DepartmentID;'''

s_conn = pyodbc.connect(s_conn_str)

df = pd.read_sql(query, s_conn)

df =df.drop_duplicates(subset='CustomerID')

d_conn_str = f'mssql+pyodbc://{d_user}:{d_pwd}@{d_server}/{d_db}?driver={d_driver}'

engine = create_engine(d_conn_str)

try:
    d_conn_str = f'mssql+pyodbc://{d_user}:{d_pwd}@{d_server}/{d_db}?driver={d_driver}'
    engine = create_engine(d_conn_str)
    print('Destination Connection build Successfully')
except Exception as e:
    print(f'There is an Error while Connecting to Destination: {e}')

df.to_sql('SalesPerson_Dim', con=engine, if_exists='append', index=False)
```

Date Table ETL:

jupyter ETL_Script Last Checkpoint: 22 minutes ago (unsaved changes)

File Edit View Insert Cell Kernel Widgets Help

Trusted | Python 3 (ipykernel)

In [ ]:

```
import numpy as np
import pandas as pd
import pyodbc
import sqlalchemy
from sqlalchemy import create_engine

s_user = 'localhost'
s_pwd = '*********'
s_db = 'AdventureWorks2019'
s_server = '************'
s_driver = '{***************}'

d_user = 'localhost'
d_pwd = '**********'
d_db = 'AdventureWorksDW-2024'
d_server = '************'
d_driver = '******************'

s_conn_str= f'DRIVER={s_driver};SERVER={s_server};DATABASE={s_db};UID={s_user};PWD={s_pwd}'
query = '''SELECT CAST(CONVERT(varchar, OrderDate, 112) AS INT) AS DateID,
       OrderDate AS FullDate,
       YEAR(OrderDate) AS Year,
       DATEPART(QUARTER, OrderDate) AS Quarter,
       MONTH(OrderDate) AS Month,
       DATEPART(WEEKDAY, OrderDate) AS DayOfWeek
FROM (SELECT DISTINCT OrderDate FROM Sales.SalesOrderHeader) AS Dates;
'''

s_conn = pyodbc.connect(s_conn_str)

df = pd.read_sql(query, s_conn)

df =df.drop_duplicates(subset='CustomerID')

d_conn_str = f'mssql+pyodbc://{d_user}:{d_pwd}@{d_server}/{d_db}?driver={d_driver}'

engine = create_engine(d_conn_str)

try:
    d_conn_str = f'mssql+pyodbc://{d_user}:{d_pwd}@{d_server}/{d_db}?driver={d_driver}'
    engine = create_engine(d_conn_str)
    print('Destination Connection build Successfully')
except Exception as e:
    print(f'There is an Error while Connecting to Destination: {e}')

df.to_sql('Date_Dim', con=engine, if_exists='append', index=False)
```

3. Data Warehouse Implementation

- **Implementation**: SQL Server was used to create tables and relationships.
- **Data Flow**: The data flows from the source database (AdventureWorks) into the data warehouse after transformation.
- **SQL Queries**:

Table Name: FactSales

```sql
CREATE TABLE FactSales (
    SalesOrderID INT,
    ProductID INT,
    CustomerID INT,
    OrderDate INT,
    SalesAmount DECIMAL(18,2),
    Quantity INT,
    FOREIGN KEY (ProductID) REFERENCES Product_Dim(ProductID),
    FOREIGN KEY (CustomerID) REFERENCES Customer_Dim(CustomerID),
    FOREIGN KEY (OrderDate) REFERENCES Date_Dim(DateID)
);
```

Table Name : Product_Dim

```sql
CREATE TABLE Product_Dim (
    ProductID INT PRIMARY KEY,
    ProductName NVARCHAR(100),
    ProductCategory NVARCHAR(50),
    ProductSubcategory NVARCHAR(50),
    ProductColor NVARCHAR(20)
```

Table Name : Customer_Dim

```sql
CREATE TABLE Customer_Dim (
    CustomerID INT PRIMARY KEY,
    CustomerName NVARCHAR(100),
    CustomerAddress NVARCHAR(100),
    CustomerCity NVARCHAR(50),
    CustomerState NVARCHAR(50),
    CustomerCountry NVARCHAR(50),
    CustomerPostalCode NVARCHAR(20)
);
```

Table Name : SalesPerson_Dim

```
CREATE TABLE SalesPerson_Dim (
    SalesPersonID INT PRIMARY KEY,
    SalesPersonName NVARCHAR(100),
    SalesPersonTitle NVARCHAR(50),
    SalesPersonHireDate DATE,
    SalesPersonDepartment NVARCHAR(50)
);
```

Table Name : Date_Dim

```
CREATE TABLE Date_Dim (
    DateID INT PRIMARY KEY,
    FullDate DATE,
    Year INT,
    Quarter INT,
    Month INT,
    DayOfWeek INT
);
```

4. Data Visualization with Power BI

- **Objective**: Create interactive reports and dashboards for sales performance and customer insights.
- **Key Reports**:
  - **Sales Performance Report**: Visualization of total sales over time, top-selling products, and sales distribution by region.

Key Outcomes:

- **Efficient Data Movement**: The ETL process automated data extraction, transformation, and loading, ensuring up-to-date data availability in the data warehouse.
- **Improved Business Insights**: Power BI reports provided stakeholders with crucial insights into sales performance, customer behavior, and product trends, enabling better decision-making.
- **Scalable Solution**: The project's architecture allows easy scaling by adding more tables, enhancing ETL logic, or expanding Power BI reports.

Conclusion:

This end-to-end solution showcases expertise in database management, data warehousing concepts, ETL design, and data visualization, ensuring seamless data integration and valuable business insights through Power BI reports.