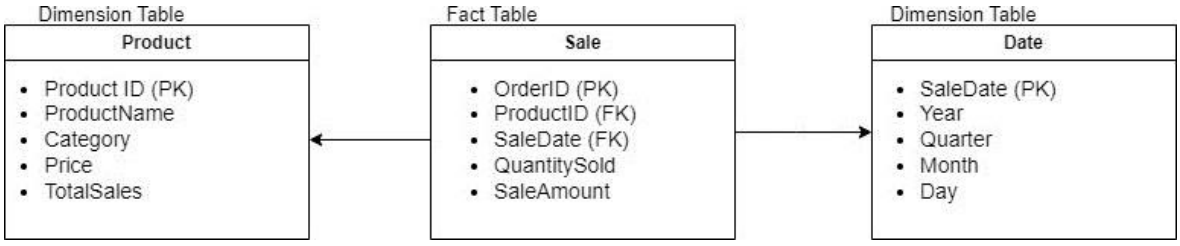
Scenario: You are working for a fictional retail company called "XYZMart," and they are looking to set up a data warehouse to analyze their sales data. Your task is to design a star schema for the data warehouse and create a Python ETL script to populate it.

Star Schema Design:



The above diagram is a star schema for the XYZMart data warehouse using the column names from the sales_data.csv and Products.csv files.

Fact Table:

Sale Table

Column Name	Data Type	Primary Key	Foreign Keys	Description
OrderID	INT	Yes	None	Unique identifier for each order
ProductID	INT	None	ProductID (Product Dimension Table)	Identifier for the product purchased
SaleDate	DATETIME	None	SaleDate (Date Dimension Table)	Date of the sale
QuantitySold	INT	None	None	Number of units of the product sold
SaleAmount	FLOAT	None	None	Total amount of the sale

Dimension Tables:

Product Dimension Table:

Column Name	Data Type	Primary Key	Foreign Keys	Description
ProductID	INT	Yes	None	Unique identifier for each product
ProductName	STRING	None	None	Name of the product
Category	STRING	None	None	Category of the product
Price	Float	None	None	Price of the product
TotalSales	Float	None	None	Total Sales of the product

Date Dimension Table:

Column Name	Data Type	Primary Key	Foreign Keys	Description
SaleDate	DATE	Yes	None	Unique identifier for each date
Year	INT	None	None	Year of the sale
Quarter	INT	None	None	Quarter of the sale (1-4)
Month	INT	None	None	Month of the sale (1-12)
Day	INT	None	None	Day of the sale (1-31)

ETL Script in Python:

I have done this python script in jupyter notebook

pandas and sqlite3 library are used. If these libraries are not present then install
#the libraries by following commands

Pip install pandas

Pip install sqlite3

```
# import pandas and sqlite3 libraries
import pandas as pd
import sqlite3
# Display the maximum number of rows and columns
pd.set_option('display.max_rows',None)
pd.set_option('display.max_columns',None)

# Create a connection to the SQLite3 database
conn = sqlite3.connect('xyzmart.db')

# Extract the data from the CSV files
sales_data_df = pd.read_csv('sales_data.csv')
products_data_df = pd.read_csv('products.csv')

# To check the shape of the csv files
sales_data_df.shape
products_data_df.shape

# To check the duplicate rows are present or not
duplicate_rows = sales_data_df[sales_data_df.duplicated()]
print("Duplicate Rows except first occurrence:")
print(duplicate_rows)

duplicate_rows = products_data_df[products_data_df.duplicated()]
print("Duplicate Rows except first occurrence:")
print(duplicate_rows)
```

```
# Clean the data by removing any rows with missing values.
sales_data_df = sales_data_df.dropna()
products_data_df = products_data_df.dropna()

# To check the rows having all null values
rows_with_null = sales_data_df[sales_data_df.isnull().any(axis=1)]
print(rows_with_null)

rows_with_null = products_data_df[products_data_df.isnull().any(axis=1)]
print(rows_with_null)

# To check the data types of the columns
products_data_df.dtypes
sales_data_df.dtypes

# To convert the data type of SaleDate from object to datetime
sales_data_df['SaleDate'] = pd.to_datetime(sales_data_df['SaleDate'])

# To check the datatypes of sales_data file
sales_data_df.dtypes

# Data Transformation
# Calculate total sales
total_sales = sales_data_df['SaleAmount'].sum()
# print the total_sales
total_sales
```

```
# Aggregate the total Sale Amount by date
sales_by_date_df = sales_data_df.groupby('SaleDate').agg({'SaleAmount': 'sum'})
# print the sales_by_date_df
sales_by_date_df

 # Merge the data from both files
merged_data_df = pd.merge(sales_data_df, products_data_df, on='ProductID')
# print the merged_data_df
merged_data_df
#Aggregate the total sale amount by ProductID
total_sales_by_product = sales_data_df.groupby('ProductID')['SaleAmount'].sum()
#Merge total_sales_by_product with products_data_df
products_data_df = pd.merge(products_data_df, total_sales_by_product,
on='ProductID')
# print the products_data_df
products_data_df

#Creating fact and dimension tables and load the data into tables
# Create the sales fact table if it doesn't exist
cur = conn.cursor()
cur.execute('''
CREATE TABLE IF NOT EXISTS sales_fact (
  OrderID INT PRIMARY KEY,
  ProductID INT,
  QuantitySold INT,
  SaleAmount DECIMAL,
  SaleDate DATE
)''')
```

#**loading the data into sales fact table**

```
sales_data_df.to_sql('sales_fact', conn, if_exists='replace', index=False)

# Create the product dimension table if it doesn't exist
cur.execute('''
CREATE TABLE IF NOT EXISTS product_dimension (
  ProductID INT PRIMARY KEY,
  ProductName VARCHAR(255),
  Category VARCHAR(255),
  Price DECIMAL,
  TotalSales DECIMAL
)''')

# Insert the transformed data into the product dimension table
products_data_df.to_sql('product_dimension', conn, if_exists='replace', index=False)

# Create the date dimension table if it doesn't exist
cur.execute('''
CREATE TABLE IF NOT EXISTS date_dimension (
  SaleDate DATE PRIMARY KEY,
  year INT,
  quarter INT,
  month INT,
  day INT
)''')
```

```
# Populate the date dimension table with the sale dates from the sales_data_df to
#find year, quarter, month, day

date_dimension_df = sales_data_df.copy()

date_dimension_df['year'] = date_dimension_df['SaleDate'].dt.year

date_dimension_df['quarter'] = date_dimension_df['SaleDate'].dt.quarter

date_dimension_df['month'] = date_dimension_df['SaleDate'].dt.month

date_dimension_df['day'] = date_dimension_df['SaleDate'].dt.day


#drop the OrderID, ProductID, QuantitySold, SaleAmount columns not needed in
dimension table

date_dimension_df.drop(['OrderID','ProductID','QuantitySold','SaleAmount'],axis=1,i
nplace=True)


# Insert the data into the date dimension table

date_dimension_df.to_sql('date_dimension', conn, if_exists='replace', index=False)


# Commit changes and close connection

conn.commit()

conn.close()

print("ETL process completed successfully.")
```

Note:

XYZMart database is created in the local device and can be check on https://inloop.github.io/sqlite-viewer/ website.