VIETNAM NATIONAL UNIVERSITY, HANOI

INTERNATIONAL SCHOOL

FINAL REPORT

DATA WAREHOUSE AND BUSINESS ANALYTICS

TOPIC:

FORECASTING SALE IN WALMART WEEKLY

Course Code: INS3073.02

Lecturer: Assoc. Prof. Dr. Trần Thị Ngân

GROUP 5

Student Name	**Student ID**
Nguyễn Thị Ngọc Lan	20070943
Hoàng Thị Lan	20070942
Đặng Thị Hiền	20070927

Hanoi, 5/2024

Table of Contents

I. Introduction

1. Background

The constant use of expanding technologies has led to an explosion of data generation in the 21st century. Big-box retailers like Walmart view this data as their most valuable resource since it allows them to forecast future sales and customer trends and create strategies for making money and competing with other businesses. Walmart is a multinational retail company based in the United States with about 11,000 locations across more than 27 countries and more than 2.2 million employees (Wikipedia, n.d.).

Walmart promises "everyday low prices" to attract customers. With a wide range of products, the company generates almost $500 billion in revenue annually. As a result, it is critical that the business use sophisticated methodologies to predict future sales and, consequently, profits. Walmart, the largest company in the world by revenue, sells a wide range of goods, including groceries, electronics, apparel, home furnishings, and body care products. As a result, it collects a lot of consumer data, which it uses to forecast customer purchasing trends, future sales, and promotional strategies. In addition, Walmart also develops cutting-edge in-store technologies. Using contemporary technology strategies is essential for the company to thrive in the cutting edge global market of today and provide goods and services that set it apart from its rivals.

2. Problem overview

In this report, we use historical sales data of 45 Walmart stores located in different regions. The main focus of this research is to forecast Walmart's sales based on the available historical data and identify whether factors like temperature, unemployment, fuel prices, etc affect the weekly sales of particular stores under study. The employment of modern technological approaches is crucial for the organization to survive in today's cutting-edge global market and create products and services that distinguish them from its competitors.

In addition, Walmart holds a number of promotional markdown sales on the days that immediately follow the major US holidays. As a result, the company needs to know how these sales affect weekly sales in order to allocate resources to these important strategic projects. In order to increase client retention, boost demand, and boost profitability, Walmart must also comprehend consumer requirements and purchasing patterns. The firm can use the study's

findings to better understand market circumstances throughout the year and allocate resources based on profitability and demand in the specific location.

Furthermore, the use of big data analytics will help in the effective analysis of historical data to produce insights and observations, assist in identifying stores that may be at risk, forecast and boost future sales and profits, and assess if the company is on the right track.

In the dataset we use, there are 4 sets: feature, test, train, and store. Below is the specific data of each dataset:

Dataset Name	Variables	Describe
Feature	Store	The store number
	Date	The week
	Temperature	Average temperature in the region
	Fuel_Price	Cost of fuel in the region
	MarkDown1 - 5	Anonymized data related to promotional markdowns that Walmart is running
	CPI	Prevailing consumer price index
	Unemployment	Prevailing unemployment rate
	IsHoliday	Whether the week is a special holiday week
Train	Store	The store number
	Dept	The department number
	Date	The week

	IsHoliday	Whether the week is a special holiday week
	Weekly_sales	Sales for the given department in the given store
Test	This file is identical to train.csv, except we have withheld the weekly sales. You must predict the sales for each triplet of store, department, and date in this file.	
Store	Store	Store number from 1 to 45
	Type	Store type has been provided, there are 3 types — A, B and C
	Size	Stores size has provided

For convenience, the four holidays fall within the following weeks in the dataset (not all holidays are in the data):

Super Bowl: 12-Feb-10, 11-Feb-11, 10-Feb-12, 8-Feb-13

Labor Day: 10-Sep-10, 9-Sep-11, 7-Sep-12, 6-Sep-13

Thanksgiving: 26-Nov-10, 25-Nov-11, 23-Nov-12, 29-Nov-13

Christmas: 31-Dec-10, 30-Dec-11, 28-Dec-12, 27-Dec-13

3. Problem objectives

The goal of the project is to develop a complex forecasting model capable of predicting weekly sales for each Walmart store. In this report, we use Power BI tool to visualize data of 45 stores and use machine learning models to forecast sales to optimize inventory management and minimize out-of-stock situations inventory, excess inventory and related costs. By achieving these goals, Walmart can improve operational efficiency, optimize inventory management, and ultimately improve customer satisfaction and profitability across its retail network.

4. Business questions

Question 1: What are the total sales, number of departments, and number of stores in the dataset?

Question 2: What are the total sales on holiday and non-holiday?

Question 3: What is the difference in average sales between non-holiday and holiday?

Question 4: What is the total sales for the week for each type of store?

Question 5: Which time during the year has the highest and lowest total weekly sales?

Question 6: Which department has the highest sales?

Question 7: Which quarter has the highest sales totals for each year?

Question 8: What is the monthly sales for each year?

Question 9: How does the unemployment rate impact sales?

Question 10: How does sales volume vary across different temperature ranges?

Question 11: What happens to sales based on Consumer Price Index (CPI) or the consumer's economic condition?

Question 12: How have fuel prices fluctuated over time?

Question 13: Correlation between the different factors that affect sales.

Question 14: Which model has the best accuracy metrics for forecasting Walmart's weekly sales?

II. Database prepare

1. Data design

The data design starts with understanding the relationships and structures in the OLTP database, represented by the Entity-Relationship Diagram (ERD). Below is an analysis of the provided ERD:

ERD Analysis

The ERD for the Walmart sales database includes the following entities:

Feature:

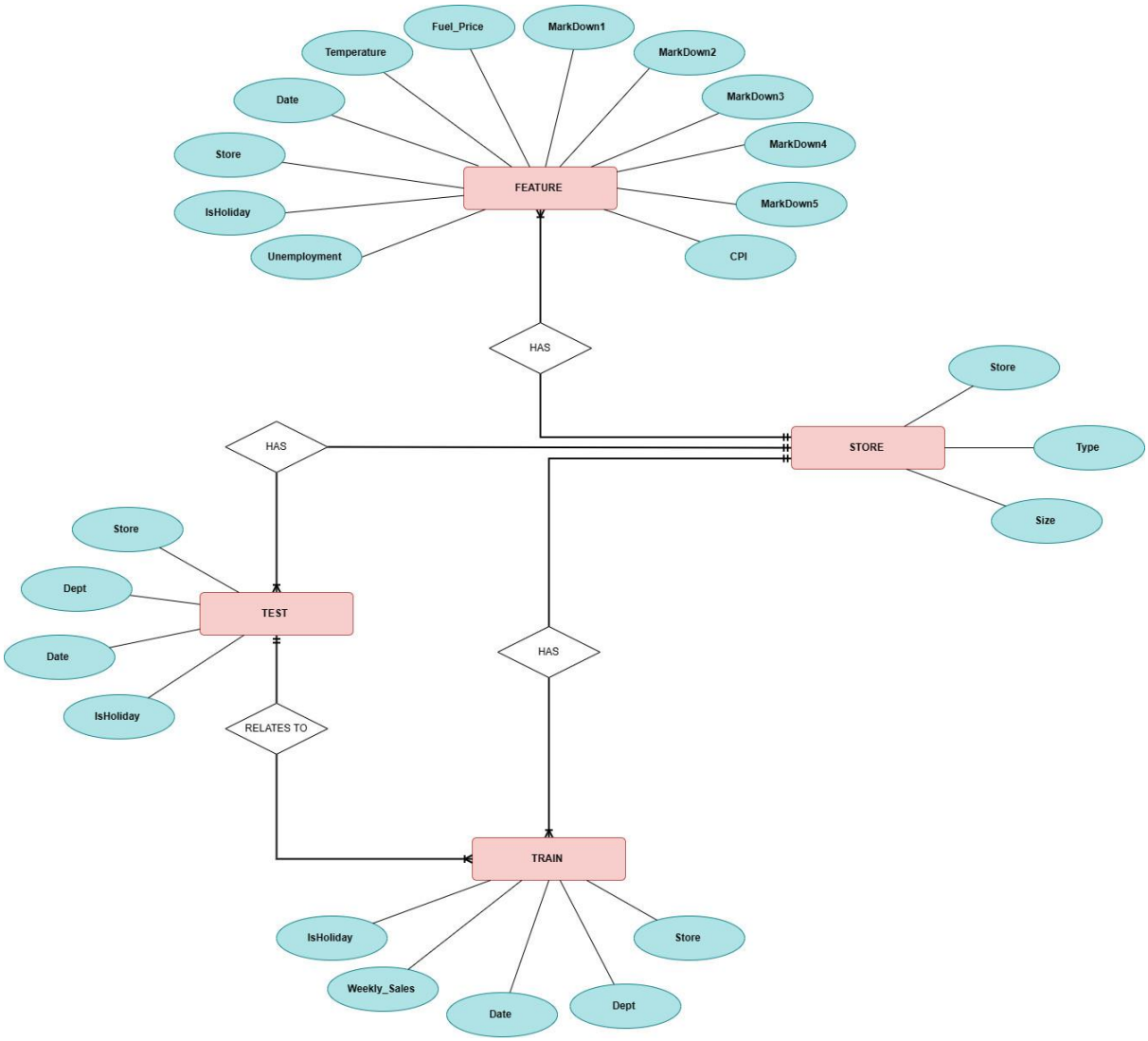
- Attributes: Store, Date, Temperature, Fuel_Price, MarkDown1, MarkDown2, MarkDown3, MarkDown4, MarkDown5, CPI, Unemployment, IsHoliday.
- Primary Keys: Store, Date.

Store:

- Attributes: Store, Type, Size.
- Primary Key: Store.

Test:

- Attributes: Store, Dept, Date, IsHoliday.
- Primary Keys: Store, Dept, Date.
- Foreign Keys: Store, Date.



The ERD represents a transactional system (OLTP) optimized for insert, update, and delete operations. However, for analytical purposes (OLAP), the data needs to be transformed into a star schema with fact and dimension tables.

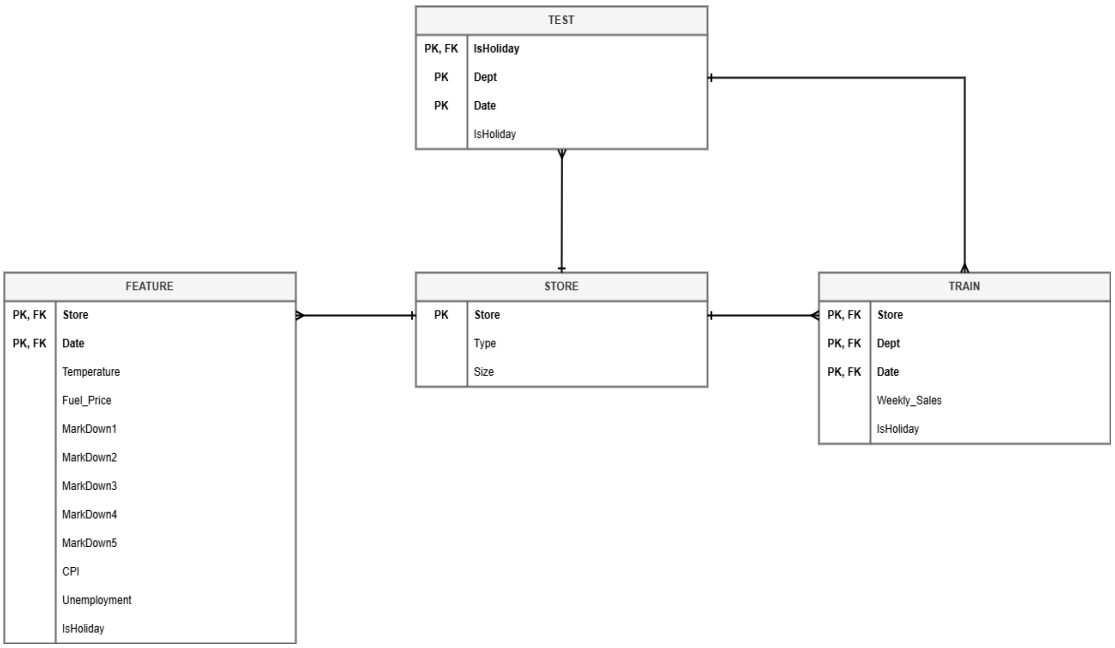
2. OLTP database design

The SalesWalmart dataset features a detailed schema that exemplifies an online transaction processing (OLTP) system for a fictitious company. The provided ERD shows the interrelations

among key business entities, elucidating the dynamic nature of sales, customer management, and order processing. Below is a summary of each table in the ERD and their interconnections:

- **STORE**: This table holds data about each store, such as its type and size. It functions as a reference point for various sales transactions, ensuring each sale is associated with a specific store. This table is referenced by the FEATURE, TEST, and TRAIN tables.
- **FEATURE**: This table logs various metrics and features for each store on specific dates, including temperature, fuel price, markdowns, CPI, and unemployment rates. It plays a crucial role in tracking external factors that may impact sales. It is linked to the STORE table through the Store ID and to the TRAIN table via the Date.
- **TEST**: This table records information about sales transactions used for testing purposes. Each record contains details such as the store, department, date, and whether it was a holiday. This table aids in analyzing sales trends and patterns during test phases. It is related to the STORE table via the Store ID.
- **TRAIN**: This table, similar to the TEST table, captures data on actual sales transactions used for model training. It includes information like the store, department, date, weekly sales, and holiday status. This comprehensive dataset supports the training of sales prediction models. It is connected to the STORE table through the Store ID and to the FEATURE table through the Date.

The schema is dominated by common one-to-many relationships within the repository, account, and sale item master files. At the heart of the transaction system is the TRAIN table, which connects stores, departments, and transaction details. The use of foreign keys and indexes ensures referential integrity and optimal query performance, essential for OLTP systems that support business operations.



3. OLTP Database Code implementation

To pull the OLTP dataset from Google Bigquery, the following SQL commands can be used:

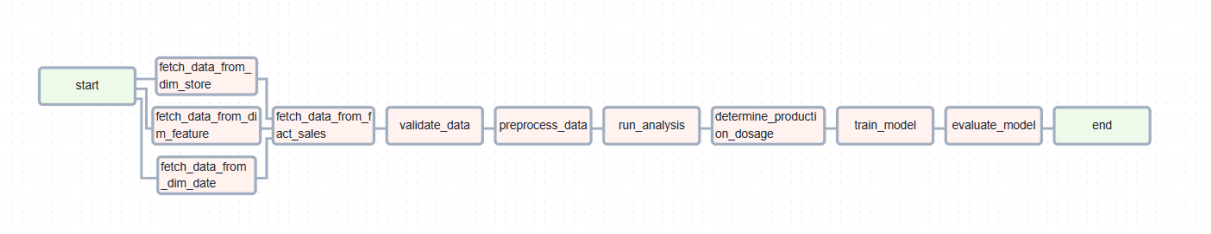
- SELECT * FROM SalesWalmart.sw.FEATURE;
- SELECT * FROM SalesWalmart.sw.STORE;
- SELECT * FROM SalesWalmart.sw.TEST;
- SELECT * FROM SalesWalmart.sw.TRAIN;

The SQL commands above are used to pull the entire dataset from the OLTP system. Each command selects all records from a specific table within the SalesWalmart database:

- The first command pulls data from the STORE table, which contains information about each store.
- The second command pulls data from the FEATURE table, recording various metrics and features for each store.
- The third command pulls data from the TEST table, capturing information about sales transactions used for testing purposes.
- The fourth command pulls data from the TRAIN table, recording actual sales transactions used for training models.

These commands facilitate data retrieval from the SQL Server, allowing for further analysis and processing of the SalesWalmart dataset.

4. ETL design & development



ETL design with Airflow

- Identify data sources: First, identify the data sources to extract including the dim_store, dim_date, dim_feature, and fact_sales tables.
- Create ETL functions: Create Python functions to perform Extract, Transform and Load steps for each data table.
- Define DAG (Directed Acyclic Graph): Create a DAG to define tasks and the sequence of tasks in the ETL process.
- Set up PythonOperator tasks: Use PythonOperator to execute ETL functions and subsequent data processing steps such as validation, preprocess, analysis, production dosage, training, and evaluation.
- Set up dependencies between tasks: Set up the execution order of tasks in the DAG to ensure the process runs in the correct order.

Operation of the ETL Process

DummyOperator (start and end): Marks the start and end of the DAG.

Extract

The data extraction steps from the CSV files are performed through the tasks:

- fetch_data_from_dim_date: Get data from dim_date tables
- fetch_data_from_dim_store: Get data from dim_store tables
- fetch_data_from_dim_feature: Get data from dim_feature tables
- fetch_data_from_fact_sales: Get data from fact_sale tables

These tasks read data from the corresponding CSV files and convert the data into JSON strings for easy processing in the subsequent steps.

Transform

After being extracted, the data will be transformed through the tasks:

- validate_data: Validate the data.
- preprocess_data: Preprocess the data, such as cleaning and standardizing the data.
- run_analysis: Perform data analysis to prepare for model training.

Load

After the data has been transformed, the steps to load the data and train the model will be performed:

- determine_production_dosage: Determine the production dosage based on the analysis.
- train_model: Use the processed data to train the machine learning model.
- evaluate_model: Evaluate the trained model to check its performance and accuracy.

Summary

The train_model_etl DAG manages the ETL process and trains the machine learning model using data from the CSV files. The steps in the process include:

    - Extract: Read data from the corresponding CSV files for the dim_date, dim_store, dim_feature, and fact_sales tables.
    - Transform: Preprocess and analyze the data to prepare for model training.
    - Load: Train and evaluate the machine learning model.

Airflow DAG

The provided code defines an Airflow DAG (Directed Acyclic Graph) that orchestrates the process of training a machine learning model using data from various sources. The DAG is designed to run daily and is responsible for fetching data, validating it, preprocessing it, performing analysis, determining production dosage, training the model, and evaluating the model. Below is a detailed explanation of each part of the code:

Import Statements

The code begins by importing necessary modules:

```
import os
import json
import pandas as pd
from airflow import DAG
from airflow.decorators import task, dag
from airflow.operators.dummy import DummyOperator
from pendulum import datetime
```

- os: Provides functions to interact with the operating system.
- json: For handling JSON data.
- pandas: A data manipulation and analysis library.
- airflow: The main Airflow library.
- dag, task: Airflow decorators for defining DAGs and tasks.
- DummyOperator: A no-op operator often used for logical grouping.
- datetime: Provides date and time manipulation.

Default Arguments

A dictionary of default arguments for the DAG is defined:

```
default_args = {
    'owner': 'airflow',
    'depends_on_past': False,
    'email_on_failure': False,
    'email_on_retry': False,
    'retries': 1,
}
```

Base Path

The base path where CSV files are stored:

```
base_path = '/opt/airflow/dags/data'
```

Function to Read CSV Files

A function to read CSV files and convert the data to a JSON string:

```
def read_csv_file(file_name):
    file_path = os.path.join(base_path, file_name)
    df = pd.read_csv(file_path)
    print(df.head())
    # Chuyển đổi dataframe thành JSON string
    return df.to_json(orient='split')
```

Task Definitions

Tasks are defined using the @task decorator. Each task fetches data from a specific CSV file or performs an operation on the data.

```python
@task
def fetch_data_from_dim_date():
    return read_csv_file('dim_date.csv')

@task
def fetch_data_from_dim_store():
    return read_csv_file('dim_store.csv')

@task
def fetch_data_from_dim_feature():
    return read_csv_file('dim_feature.csv')

@task
def fetch_data_from_fact_sales():
    return read_csv_file('fact_sales.csv')

@task
def validate_data(data_json):
    df = pd.read_json(data_json, orient='split')
    print("Validating data")
    return df.to_json(orient='split')

@task
def preprocess_data(data_json):
    df = pd.read_json(data_json, orient='split')
    print("Preprocessing data")
    return df.to_json(orient='split')
```

```python
@task
def run_analysis(data_json):
    df = pd.read_json(data_json, orient='split')
    print("Running analysis")
    return df.to_json(orient='split')

@task
def determine_production_dosage(data_json):
    df = pd.read_json(data_json, orient='split')
    print("Determining production dosage")
    return df.to_json(orient='split')

@task
def train_model(data_json):
    df = pd.read_json(data_json, orient='split')
    print("Training model")
    return df.to_json(orient='split')

@task
def evaluate_model(data_json):
    df = pd.read_json(data_json, orient='split')
    print("Evaluating model")
    return df.to_json(orient='split')
```

DAG Definition

The DAG is defined using the @dag decorator:

```
@dag(
    start_date=datetime(2024, 1, 1),
    schedule="@daily",
    catchup=False,
    default_args=default_args,
    tags=["bigquery", "model_training"],
)
def train_model_etl():
    start = DummyOperator(task_id='start')
    end = DummyOperator(task_id='end')

    fetch_dim_date_task = fetch_data_from_dim_date()
    fetch_dim_store_task = fetch_data_from_dim_store()
    fetch_dim_feature_task = fetch_data_from_dim_feature()
    fetch_fact_sales_task = fetch_data_from_fact_sales()
```

```
validate_data_task = validate_data(fetch_fact_sales_task)
preprocess_data_task = preprocess_data(validate_data_task)
run_analysis_task = run_analysis(preprocess_data_task)
determine_production_dosage_task = determine_production_dosage(run_analysis_task)
train_model_task = train_model(determine_production_dosage_task)
evaluate_model_task = evaluate_model(train_model_task)

start >> [fetch_dim_date_task, fetch_dim_store_task, fetch_dim_feature_task] >> fetch_fact_sales_task
fetch_fact_sales_task >> validate_data_task >> preprocess_data_task >> run_analysis_task
run_analysis_task >> determine_production_dosage_task >> train_model_task >> evaluate_model_task
evaluate_model_task >> end

train_model_etl()
```

Explanation

- The DAG is scheduled to run daily starting from January 1, 2024.
- The DAG starts with a DummyOperator named start and ends with a DummyOperator named end.
- Data fetching tasks are defined to read data from CSV files for dimension tables (dim_date, dim_store, dim_feature) and fact table (fact_sales).
- Subsequent tasks validate, preprocess, run analysis, determine production dosage, train the model, and evaluate the model.
- The tasks are linked to define the order of execution, ensuring that the data flows through the necessary stages from fetching to evaluation.

5. Configuration

Required Libraries and Tools

- Astro CLI
- Google Cloud BigQuery
- Pandas
- Pendulum
- Airflow

Dockerfile

```
FROM quay.io/astronomer/astro-runtime:11.3.0

# Install any OS-level dependencies you need here
# RUN apt-get update && apt-get install -y ...

# Install Python packages needed for your project
COPY requirements.txt .
RUN pip install --no-cache-dir -r requirements.txt

# Copy the service account key
COPY include/saleswalmart-424512-7c1d4c742a6f.json /opt/airflow/keys/saleswalmart-424512-7c1d4c742a6f.json

# Set environment variable
ENV GOOGLE_APPLICATION_CREDENTIALS=/opt/airflow/keys/saleswalmart-424512-7c1d4c742a6f.json

# Load environment variables
ENV AIRFLOW__CORE__ENABLE_XCOM_PICKLING=True

# Copy CSV files to the container
COPY dags/data/dim_store.csv /opt/airflow/dags/data/dim_store.csv
COPY dags/data/dim_date.csv /opt/airflow/dags/data/dim_date.csv
COPY dags/data/dim_feature.csv /opt/airflow/dags/data/dim_feature.csv
COPY dags/data/fact_sales.csv /opt/airflow/dags/data/fact_sales.csv
COPY OLTP-csv/feature.csv /opt/airflow/dags/data/feature.csv
COPY OLTP-csv/store.csv /opt/airflow/dags/data/store.csv
COPY OLTP-csv/test.csv /opt/airflow/dags/data/test.csv
COPY OLTP-csv/train.csv /opt/airflow/dags/data/train.csv
```

requirements.txt

```
apache-airflow-providers-google
google-auth
google-cloud-bigquery
```

III. Building data warehouse

1. Step 1: Choose business processes

In the context of Walmart, the chosen business process centers around sales, specifically examining the performance and trends within the Sales department. The decision to focus on sales is rooted in the project's overarching objectives, aiming to derive actionable insights to enhance sales efficiency, understand market dynamics, and optimize strategic decision-making.

Walmart OLTP Bus Matrix for Sale			
	Feature	Store	Date
Sale Forecast	X	X	X
Sale Actual	X	X	X
Sale Return		X	X

In the bus matrix:

- DimFeature, DimStore and DimDate are dimensions.
- Sale Forecast, Sale Actual and Sale Return are potential business processes.

For each business process:

- Sale Forecast involves feature, store and date.
- Sale Actual shares the same dimensions as the Sale Forecast involves feature, store and date.
- Sale Return includes store and date.

2. **Step 2: Declare the Grain**

Transaction Level (Atomic level):

For our project, the chosen grain is the transaction level, epitomized by the FactSales fact table. At this finest granularity, each row within FactSales meticulously captures the details of an individual sales transaction.

Daily/Periodic Level:

To facilitate analysis over specific time frames, our design incorporates the DimDate dimension. This enables aggregation at daily or periodic levels and on special occasions, which products will be chosen by consumers helps you predict how to import goods in a timely manner. The DateID attribute in DimDate acts as a linchpin for time-based analyses, supporting the extraction of meaningful patterns and trends.

Summary:

Elevating our perspective for broader business insights, especially in monthly or yearly contexts, involves leveraging additional dimensions such as DimFeature, DimStore. Aggregating data based on these dimensions provides a higher-level view of sales performance

across key business aspects. This summary-level granularity proves invaluable for strategic decision-making and long-term planning. By selecting this granularity, we opt for the lowest level of detail, capturing individual sales line items. This strategic choice not only aligns with best practices but also positions us for flexibility in subsequent analyses and reporting. In essence, our data warehouse is designed to seamlessly transition from the intricacies of individual transactions to the broader perspectives essential for strategic business insights.

3. Step 3: Identify the Dimensions

Dimension Table: DimFeature

- Feature_id: A unique identifier for each factor
- Store_id: A unique identifier for each store
- Date_id: A unique identifier for each date
- Temperature: Average temperature in the region
- Fuel_price: Cost of fuel in the region
- Markdown 1-5: Anonymized data related to promotional markdown that Walmart running
- CPI: Prevailing consumer price index
- Unemployment: Prevailing unemployment rate
- Purpose: This table capture information about factors affecting weekly revenue

Code implement

```
INSERT INTO `saleswalmart-424512.OLAPSW.dim_feature` (temperature, fuel_price, markdown1, markdown2, markdown3,
markdown4, markdown5, cpi, unemployment)
SELECT DISTINCT temperature, fuel_price, markdown1, markdown2, markdown3, markdown4, markdown5, cpi, unemployment
FROM `SalesWalmartOLTP.feature`;
```

Dimension Table: DimDate

- Date_id: A surrogate primary key for the date dimension
- Is_holiday: Whether the week is a special holiday week
- Purpose: The DimDate table serves as a time dimension, enabling time-based analysis and reporting.

Code implement

```
INSERT INTO `saleswalmart-424512.OLAPSW.dim_date` (date_id, is_holiday)
SELECT DISTINCT date_id, is_holiday FROM (
    SELECT date_id, is_holiday FROM `SalesWalmartOLTP.test`
    UNION ALL
    SELECT date_id, is_holiday FROM `SalesWalmartOLTP.train`
);
```

Dimension Table: DimStore

- Store_id: A unique identifier for each store
- Type: The store type has been provided, there are 3 types — A, B and C
- Size: The stores size has provided
- Purpose: This dimension table provides the type and size of the store.

Code implement

```
INSERT INTO `saleswalmart-424512.OLAPSW.dim_store` (store_id, type, size)
SELECT DISTINCT store_id, type, size FROM `SalesWalmartOLTP.store`;
```

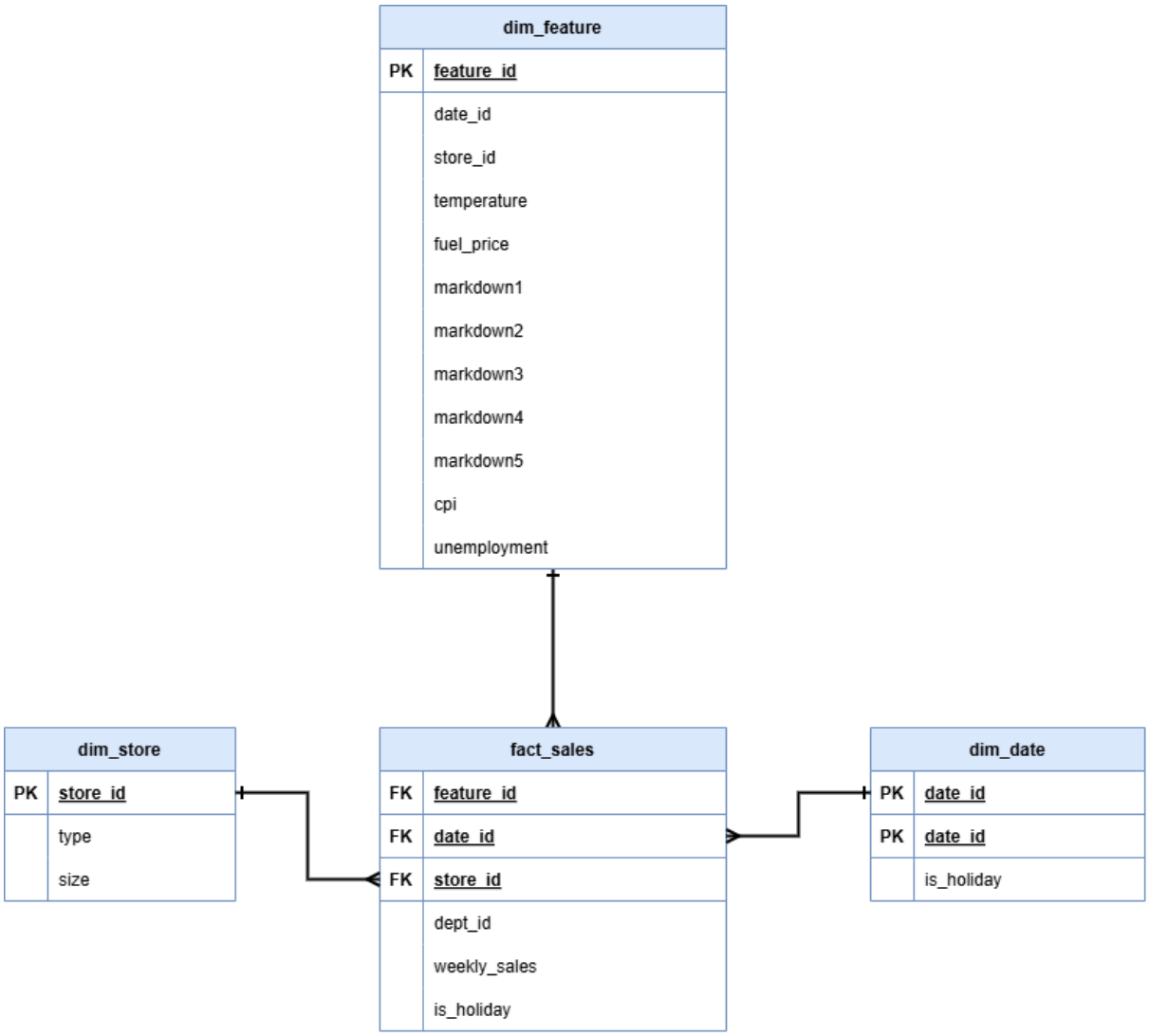
4. **Step 4: Identify the Facts**

- Store_id: A unique identifier for each store
- Dept_id: A unique identifier for each dept
- Date_id: A surrogate primary key for the date dimension
- Feature_id: A unique identifier for each factor
- Weekly_sale: Sales for the given department in the given store
- Is_holiday: Whether the week is a special holiday week

Code implement

```
INSERT INTO `saleswalmart-424512.OLAPSW.fact_sales` (store_id, dept_id, date_id, feature_id, weekly_sales,
is_holiday)
SELECT
    train.store_id,
    train.dept_id,
    train.date_id,
    feature.feature_id,
    train.weekly_sales,
    train.is_holiday
FROM `SalesWalmartOLTP.train` AS train
JOIN `saleswalmart-424512.OLAPSW.dim_feature` AS feature
ON train.temperature = feature.temperature
AND train.fuel_price = feature.fuel_price
AND train.markdown1 = feature.markdown1
AND train.markdown2 = feature.markdown2
AND train.markdown3 = feature.markdown3
AND train.markdown4 = feature.markdown4
AND train.markdown5 = feature.markdown5
AND train.cpi = feature.cpi
AND train.unemployment = feature.unemployment;
```

Dimensional modeling - Star schema generation



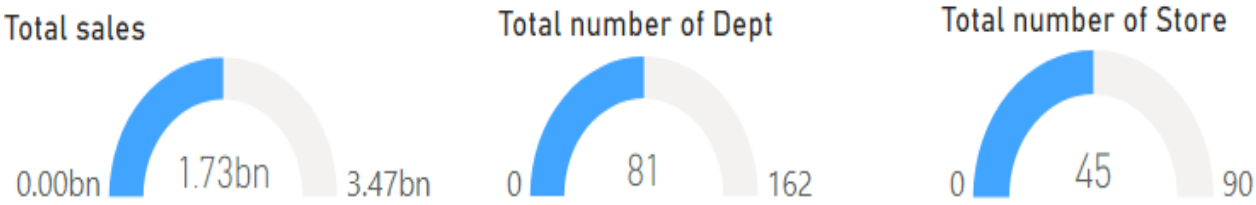
IV. Results & Implication

1. Explore the data

This exploratory data analysis aims to uncover insights from our OLAP data warehouse, focusing on weekly sales and factors influencing Walmart sales. Using visualizations created in Power BI, we identified patterns and trends that would support our strategic decision-making.

Answering the following question:

1.1 What are the total sales, number of departments, and number of stores in the data set?



Insight:

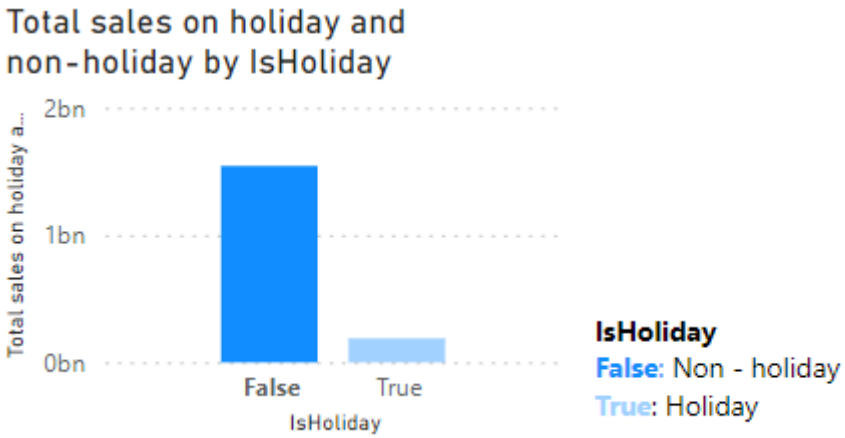
The image shows a set of three semi-circular progress bars, each representing different metrics:

- Total weekly sales: Marked at 1.73Bn, which may indicate that the total weekly sales in the data set is 1.73Bn (Billion)
- Total number of Dept: There are 81 dept
- Total number of Store: The number of stores are 45

Interpretation:

- Based on Walmart's total weekly sales, we can see that Walmart is generating a very large amount of revenue each week, reflecting the large scale and scope of operations of the business.
- Total number of dept (81): This indicates that Walmart offers a diverse range of products and services, classified into 81 different departments. This diversity may be an important factor contributing to the company's high sales.
- Total number of stores: This shows that the dataset includes information from 45 different Walmart stores. With such a large number of stores, data can provide insight into sales performance across the store system, helping Walmart optimize business strategy and operations.

1.2 What are the total sales on holiday and non-holiday?



IsHoliday	Total sales on holiday and non-holiday
False	1,544,222,888.61
True	188,925,044.35

Insight:

- Total sales on non-holiday are more 1.5B
- Total sales on holidays is nearly 200 million, specifically approximately 189 million

=> Which is normal because the number of holidays is very small compared with the number of other days.

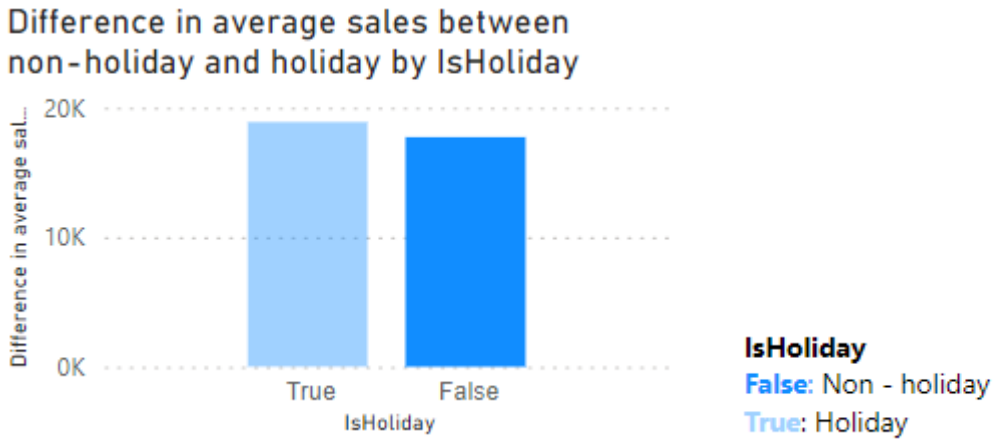
Interpretation:

During holidays, although shopping demand may increase due to promotions and holiday shopping activities, the total number of these days is not enough to generate as much revenue as on weekdays.

This helps understand the revenue distribution between weekdays and holidays, helping Walmart plan its business strategy more effectively.

- During the holidays, Walmart can focus on special marketing campaigns, increased advertising and promotions to maximize revenue from increased shopping activities.
- During weekdays, the company can focus on maintaining supply stability and service quality to ensure continuously high sales.

1.3 What is the difference in average sales between non-holiday and holiday?



IsHoliday	Difference in average sales between non-holiday and holiday
True	18,907.63
False	17,736.64

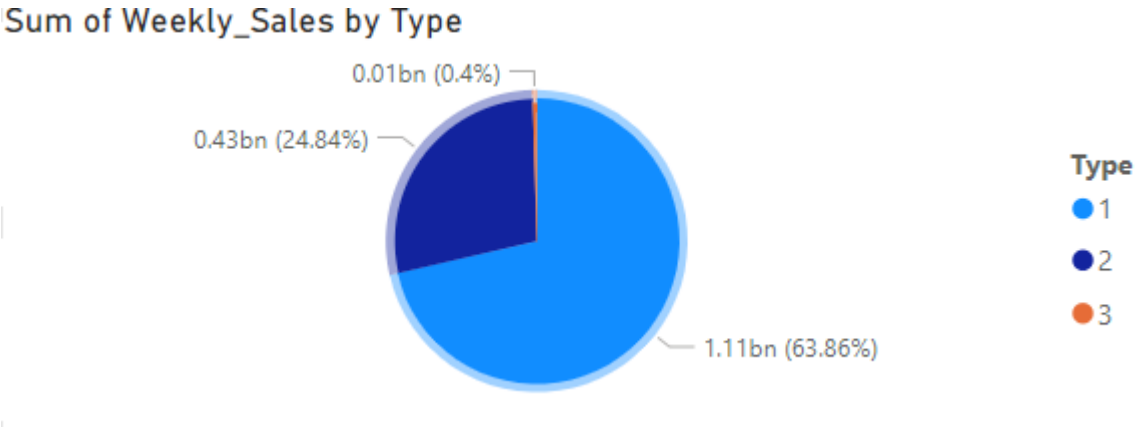
Insight:

Looking at the chart, we can see that the average sales on holidays and non-holidays are approximately equal. However, the rate of sales on holidays is higher than on other days.

Interpretation:

- Although the number of holidays is small, the percentage of sales during these days is higher, indicating that customers spend more during the holidays. This may be due to increased demand for gift shopping, event planning, and preparation for special occasions.
- Walmart can take advantage of the holidays to increase sales by implementing promotions, discounts, and special advertising campaigns. At the same time, inventory management needs to be done carefully to ensure that it meets the increased shopping demand during the holidays.

1.4 What is the total sales for the week for each type of store?



Type	Sum of Weekly_Sales
1	1,106,747,029.41
2	430,589,910.89
3	6,885,948.31

Insight:

- The total sales of store 1 is more than 1 billion.
- Total sales of store 2 is more than 400 million.
- Meanwhile, the total sales of store 3 is only over 6 million.

=> Thus, we can see a huge difference in sales between different types of stores.

Interpretation:

There are large differences in sales between stores. This may reflect differences in size, geographic location, customer traffic, and operating efficiency of each store.

- Walmart may need to reconsider its strategy for growing and supporting stores with lower sales. For stores like store 3, it may be necessary to improve marketing activities, adjust products and services, or even reconsider location to increase customer appeal and increase sales.
- With high-volume stores like stores 1 and 2, Walmart can continue to invest and expand successful business strategies at these stores to maintain and grow revenue.

1.5 Which time during the year has the highest and lowest total weekly sales?

Insight:

- The highest total weekly sales was on December 23, 2011 with a milestone of more than 70 billion falling near Christmas.
- The lowest weekly sales total was on June 29, 2012.

Interpretation:

- On December 23, 2011, total weekly sales peaked, surpassing US$70 billion. This falls right around Christmas, one of the biggest shopping occasions of the year. This increase in sales can be explained by the demand for gifts, decorations, and other holiday-related products.
- On June 29, 2012, lowest weekly sales total. This is the middle of summer, when shopping activity usually declines because customers tend to travel or participate in outdoor activities, focusing less on shopping.

=> This information shows that seasonal factors have a major impact on Walmart's weekly sales.

- To make the most of holiday sales, Walmart should focus on inventory management, offer seasonal products, and deploy strong marketing campaigns to attract customers.
- During low sales periods such as summer, Walmart can create special promotions, offer products suitable for outdoor activities and travel, and hold events to stimulate purchases. shopping.

1.6 Which department has the highest sales?

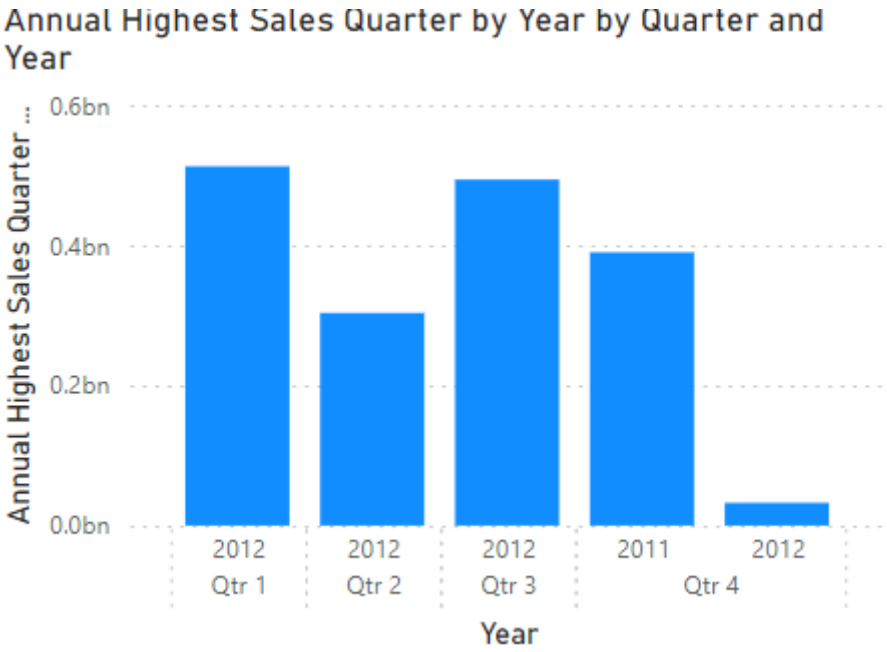
Insight:

- The department with the highest sales is 92.

Interpretation:

- Department numbers do not have to be consecutive from 1 to 81. Departments with codes outside this range may exist due to the reorganization or addition of new departments in Walmart's system.
- With the highest sales, department number 92 may be one of Walmart's key business areas. This could be a specialized department or a very popular product category that accounts for a large portion of the company's total revenue.

1.7 Which quarter has the highest sales totals for each year?



Insight:

- The first quarter of 2012 had the highest weekly sales at over half a billion.

Interpretation:

- By understanding and exploiting strong sales trends in the first quarter of 2012, Walmart can develop more effective business strategies to maintain and grow sales during other key periods of the year. . This not only helps the company achieve better business results but also ensures stability and sustainable development.

1.8 What is the monthly sales for each year?

Year, Month	The monthly sales for each year
2011, November	145,128,753.16
2011, December	244,998,377.22
2012, January	139,875,459.33
2012, February	170,870,917.44
2012, March	202,120,185.13
2012, April	122,021,336.52
2012, May	18,456,021.01
2012, June	163,113,659.25
2012, July	168,311,703.23
2012, August	198,704,214.19
2012, September	127,013,719.42
2012, October	32,533,587.06

Insight:

You can see the results in the table above:

- In November 2011 sales were 145 million.
- In December 2011 sales were nearly 245 million, this was the month with the highest sales in 2 years (2011 vs 2012).

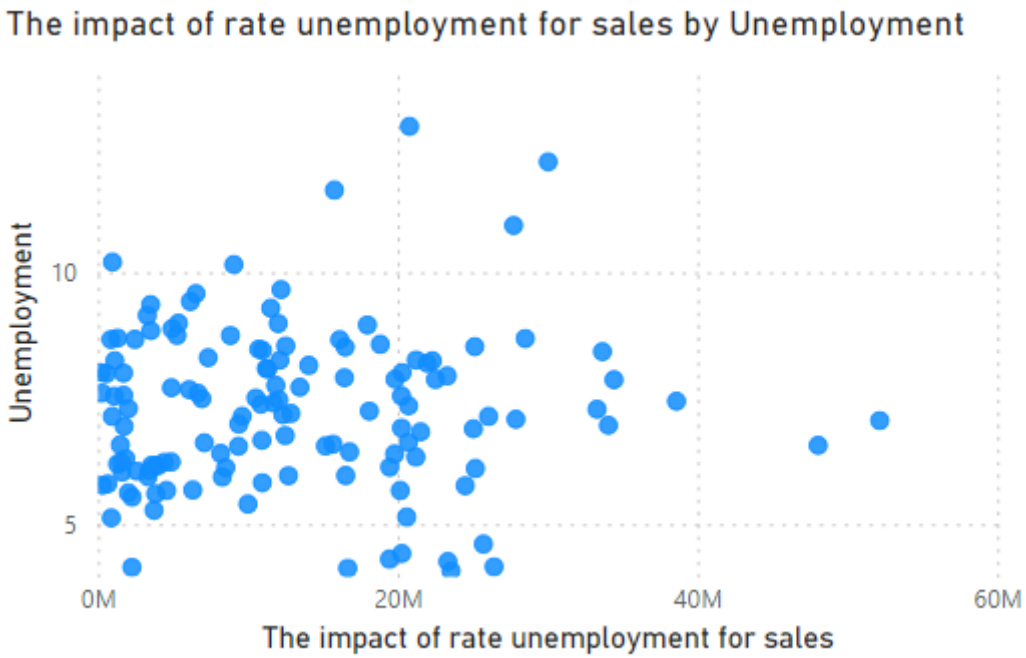
Similarly, you can see the sales of other months over the years in the results table above.

Interpretation:

- In November 2011, sales reached 145 million. This is a fairly high sales level, reflecting increased shopping demand at the beginning of the holiday season.

- In December 2011, sales reached nearly 245 million. This is the month with the highest sales, reflecting the shopping boom during the Christmas and year-end seasons. The strong sales increase in December shows the importance of the holidays to Walmart's business.

1.9 How does the unemployment rate impact sales?



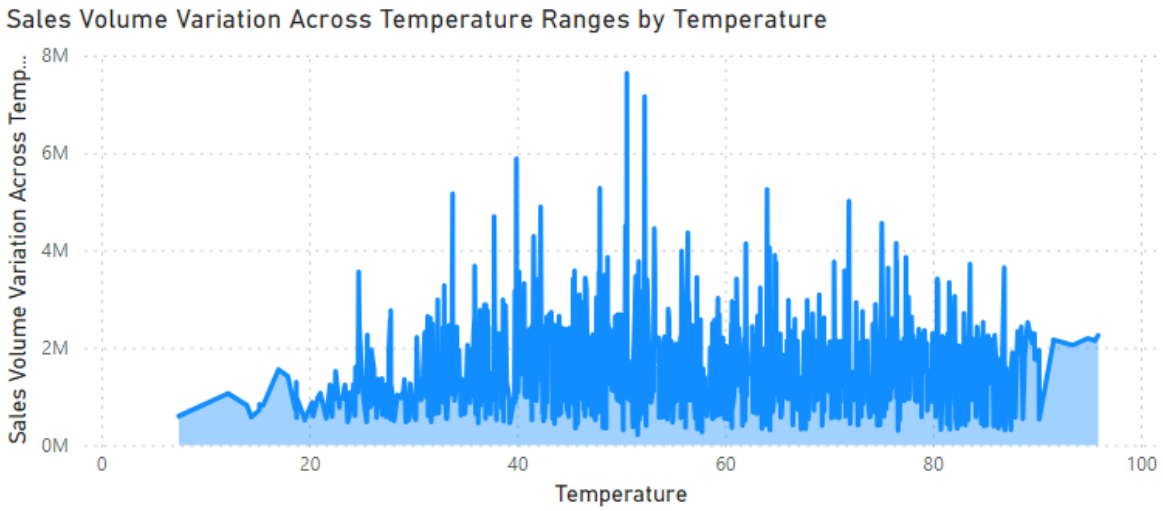
Insight:

- From the scatter chart, it shows that the higher the unemployment rate, the lower the sales. However, there are also exceptions that show that the unemployment rate does not affect sales.

Interpretation:

- High unemployment rates often lead to reduced disposable income, which in turn reduces customers' purchasing power and negatively affects sales. This highlights the importance of the general economic situation for Walmart's business.
- The presence of outliers shows that unemployment does not always affect sales. Factors such as business strategy, promotional programs, product quality, and customer service also play an important role in maintaining and growing sales.

1.10 How does sales volume vary across different temperature ranges?



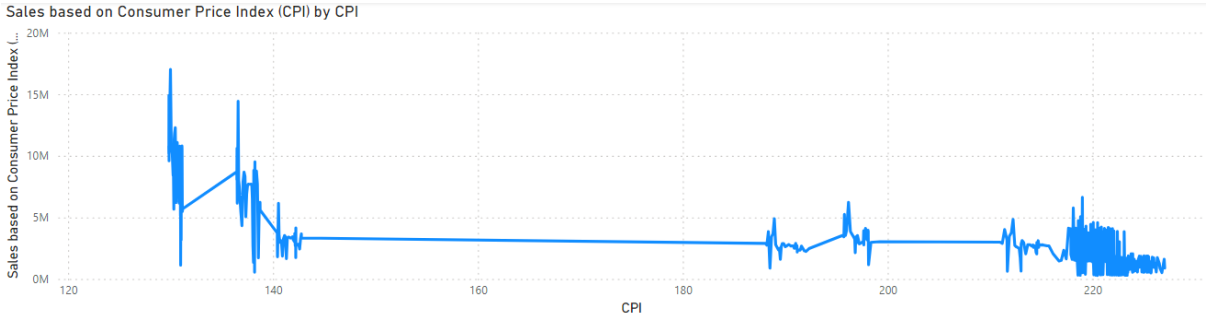
Insight:

- Sales are not affected by changes in temperature.

Interpretation:

- Sales were not affected by temperature suggesting that Walmart could focus on other factors such as the holiday season, local economy, or merchandising strategies to optimize business performance.
- Walmart can use this information to adjust sales and inventory management strategies effectively, without necessarily relying too much on temperature factors.

1.11 What happens to sales based on Consumer Price Index (CPI) or the consumer's economic condition?



Insight:

There is significant variation between sales and CPI:

- There are notable peaks in sales at certain CPI values, particularly around the 130 - 150 range and again around the 220 range. The chart shows periods of high sales followed by sharp drops.
- Low Activity Period: Between CPI values of approximately 150 and 190, and again around 200 - 215, sales appear to be relatively stable and lower compared to other periods.

=> In summary, the chart indicates that sales have a volatile relationship with CPI, with significant spikes and drops at various CPI values, rather than showing a smooth trend.

Interpretation:

- Large fluctuations between sales and CPI may reflect the complexity of the market and the impact of many different factors, not just CPI but also overall economic conditions, monetary policy, and specific business factors.
- Identifying peaks and dips in sales at specific CPI values can help Walmart better understand the factors that influence customer shopping behavior and develop business strategies suitable business.
- Walmart can use this information to flexibly adjust business strategies and resource management, respond to market fluctuations and optimize business performance.

1.12 How have fuel prices fluctuated over time?

Insight:

The line chart provides information about the fluctuation of fuel prices over time, from 2011 to 2012.

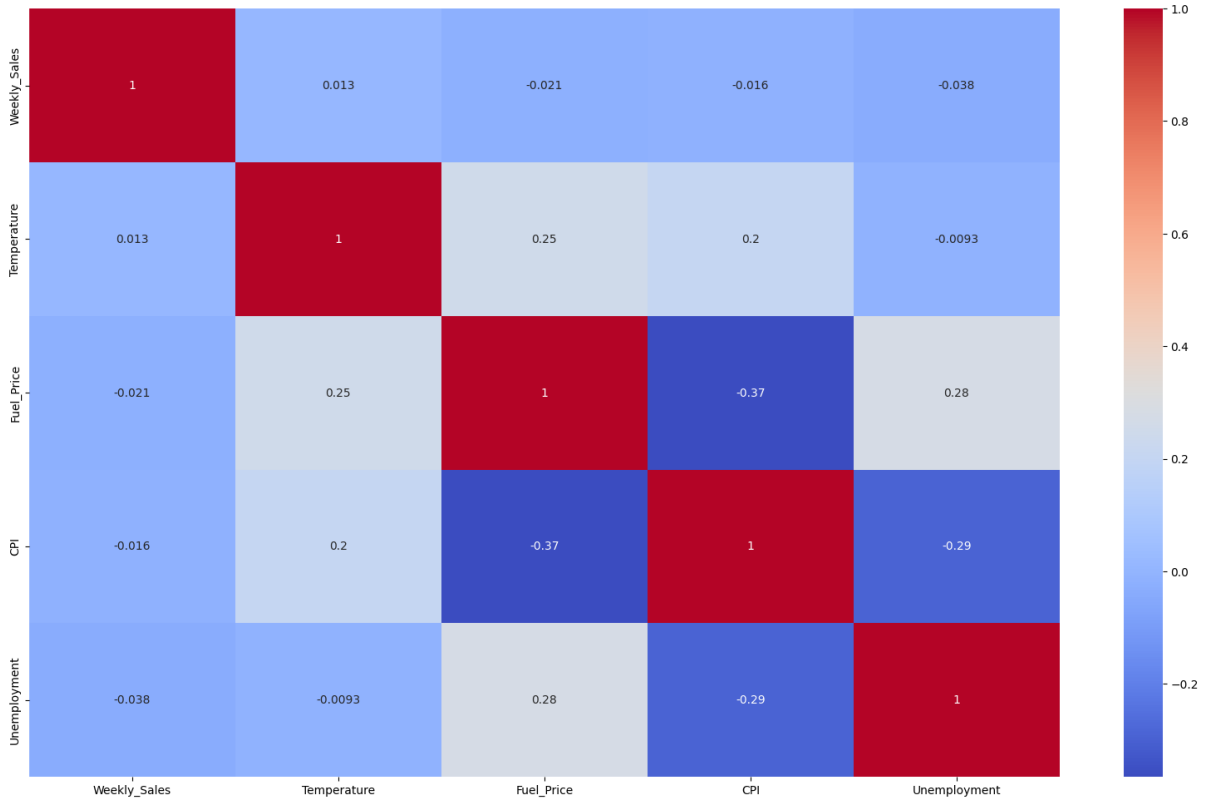
- In the first period (October 2011 to January 2012), fuel prices tended to decrease slightly initially, then fluctuate with several peaks and troughs.
- During the middle period (February to May 2012), fuel prices fluctuate strongly with clear peaks and troughs, especially around March and May.
- The final period (June to September 2012) continues to show sharp increases and decreases in fuel prices, with sharp drops in July and August followed by a slight decrease in September.

=> Overall, the chart shows that fuel prices have experienced many fluctuations throughout 2012, with many consecutive peaks and troughs without a clear upward or downward trend.

Interpretation:

- The sharp and volatile fluctuations in fuel prices throughout 2012 may reflect uncertainty in energy markets and the impact of volatile factors such as oil market volatility and economic policy global.
- Information about fuel price fluctuations can help businesses like Walmart manage risk and plan their business more effectively, by adjusting pricing strategies, inventory management, and cost forecasting.

1.13 Correlation between the different factors that affect sales.



- "Weekly_Sales" has a very low correlation with all other variables (correlation values range from -0.038 to 0.013), suggesting that there is no strong association between weekly sales and the other variables in the set.
- "Fuel_Price" has a moderate negative correlation with "CPI" (Consumer Price Index) (-0.37), meaning that when fuel prices increase, CPI tends to decrease and vice versa.
- "Fuel_Price" also has a moderate positive correlation with "Unemployment" (0.28), showing that when fuel prices increase, the unemployment rate also tends to increase.
- "Temperature" has a moderate positive correlation with "Fuel_Price" (0.25) and "CPI" (0.2).

2. Forecast Model Building

2.1 Data preprocessing

Classification of variables:

```
num_features = data.select_dtypes(include=['int64', 'float64']).columns.to_list()
num_features.remove('Weekly_Sales')
cat_features = data.select_dtypes(include=['object']).columns.to_list()
```

- num_features: Selects all columns in the data data that have data type int64 or float64 and converts them to a list. Then remove the column 'Weekly_Sales' from this list.
- cat_features: Selects all columns in the data data whose data type is object (categorical variable) and converts them to a list.

Data scaling:

```
X = data.drop('Weekly_Sales', axis=1)
y = data['Weekly_Sales']

X_train, X_test, y_train, y_test = train_test_split(X, y, test_size=0.3, random_state=42)
```

- X: Create a DataFrame containing all columns except the 'Weekly_Sales' column.
- y: Create a Series containing the value of column 'Weekly_Sales'.
- train_test_split: Split data into training set and testing set, with 30% of data for testing set. random_state=42 to ensure reproducibility of data splitting

Define data preprocessing steps:

```
preprocessor = ColumnTransformer(
    transformers=[
        ('num', MinMaxScaler(), num_features),
        ('cat', BinaryEncoder(), cat_features)
    ])
```

- ‘num': Apply MinMaxScaler() to numeric columns (num_features). MinMaxScaler normalizes the values of variables to be between 0 and 1.
- 'cat': Apply BinaryEncoder() to categorical columns (cat_features). BinaryEncoder encodes categorical variables into binary form.

```
X_train = preprocessor.fit_transform(X_train)
X_test = preprocessor.transform(X_test)
```

2.2 Training model and Evaluation

2.2.1 Linear Regression

```
from sklearn.linear_model import LinearRegression
from sklearn.metrics import mean_absolute_error, mean_squared_error, r2_score
lr_model = LinearRegression()
lr_model.fit(X_train, y_train)
```

```
▾ LinearRegression
LinearRegression()
```

```
y_train_pred_lr = lr_model.predict(X_train)
y_test_pred_lr = lr_model.predict(X_test)
```

Train a linear regression model on the training data set and use the trained model to predict the value of the target variable on the training set.

Results:

```
Linear Regression:
MAE (Train): 16073.308913856128
MSE (Train): 557806986.5629182
R^2 (Train): 0.07647822948931982
MAE (Test): 16069.931187043374
MSE (Test): 579464530.4118108
R^2 (Test): 0.07958568383908526
```

Evaluation:

- The Linear Regression model has poor performance, with a very low R^2 value (about 0.08), which indicates that the model only explains 8% of the variation in the training and testing data.
- The MAE and MSE are high for both the training and test sets, indicating large prediction errors.

2.2.2 Random Forest

```
rf_model = RandomForestRegressor(max_depth=10, n_estimators=100, random_state=42)
rf_model.fit(X_train, y_train)
```

```
RandomForestRegressor
RandomForestRegressor(max_depth=10, random_state=42)
```

```
y_train_pred_rf = rf_model.predict(X_train)
y_test_pred_rf = rf_model.predict(X_test)
```

Initialize an object of the RandomForestRegressor class with the following parameters:

- max_depth=10: The maximum depth of each decision tree in the forest is 10. This helps control overfitting.
- n_estimators=100: The number of decision trees in the forest is 100.
- random_state=42: Set the seed for the random number generator to ensure reproducibility of results.

Results:

```
Random Forest Regressor:
MAE (Train): 4214.924437676221
MSE (Train): 52678364.152005404
R^2 (Train): 0.9127841398526856
MAE (Test): 4504.315384239945
MSE (Test): 69710860.55815813
R^2 (Test): 0.8892721285217802
```

Evaluation:

- The Random Forest Regressor model has very good performance, with R^2 on the training and test sets being 0.9128 and 0.8893, respectively, indicating that the model explains about 89% of the variation in the test data.
- MAE and MSE are significantly lower than Linear Regression, indicating that this model is more accurate in its predictions.

2.2.3 Gradient Boosting Regressor

```
gb_model = GradientBoostingRegressor(n_estimators=100, random_state=42)
gb_model.fit(X_train, y_train)
```

```
GradientBoostingRegressor
GradientBoostingRegressor(random_state=42)
```

```
y_train_pred_gb = gb_model.predict(X_train)
y_test_pred_gb = gb_model.predict(X_test)
```

Initialize an object of the GradientBoostingRegressor class with the following parameters:

- n_estimators=100: The number of decision trees in the model is 100. A larger number of trees usually helps the model learn better but also increases training time and can lead to overfitting if not controlled Good.
- random_state=42: Set the seed for the random number generator to ensure reproducibility of results. This means that if you run this code multiple times, you will get the same result.

Then, train the Gradient Boosting Regressor model on the training data set.

Results:

```
Gradient Boosting Regressor:
MAE (Train): 7278.968425805837
MSE (Train): 140291461.2220332
R^2 (Train): 0.7677293010371994
MAE (Test): 7383.64541850957
MSE (Test): 158626681.3631622
R^2 (Test): 0.7480393349563775
```

Evaluation:

- The Gradient Boosting Regressor model also performed quite well, with R^2 on the training and test sets being 0.7677 and 0.7480, respectively.
- The MAE and MSE of this model are lower than Linear Regression but higher than Random Forest Regressor.

2.2.4 XGBoost Regressor

```
xgb_model = XGBRegressor(n_estimators=100, random_state=42)
xgb_model.fit(X_train, y_train)
```

```
XGBRegressor(base_score=None, booster=None, callbacks=None,
             colsample_bylevel=None, colsample_bynode=None,
             colsample_bytree=None, device=None, early_stopping_rounds=None,
             enable_categorical=False, eval_metric=None, feature_types=None,
             gamma=None, grow_policy=None, importance_type=None,
             interaction_constraints=None, learning_rate=None, max_bin=None,
             max_cat_threshold=None, max_cat_to_onehot=None,
             max_delta_step=None, max_depth=None, max_leaves=None,
             min_child_weight=None, missing=nan, monotone_constraints=None,
             multi_strategy=None, n_estimators=100, n_jobs=None,
             num_parallel_tree=None, random_state=42, ...)
```

```
y_train_pred_xgb = xgb_model.predict(X_train)
y_test_pred_xgb = xgb_model.predict(X_test)
```

Similarly, initialize and train the XGBoost Regressor model.

Results:

```
XGBoost Regressor:
MAE (Train): 2984.6385654969095
MSE (Train): 22661500.561522525
R^2 (Train): 0.9624809483833071
MAE (Test): 3388.1545879240475
MSE (Test): 38827892.66969141
R^2 (Test): 0.9383262539742588
```

Evaluation:

- The XGBoost Regressor model has the best performance of the tested models, with R^2 on the training and test sets of 0.9625 and 0.9383, respectively, indicating that the model explains about 94% of the variation in test data.
- MAE and MSE are the lowest among the models, indicating that this model has the smallest prediction error.

3. Overall evaluation

- XGBoost Regressor is the best performing model with the highest R^2 and lowest MAE and MSE, indicating that it is the most accurate model for this problem.
- Random Forest Regressor is also a good choice, with performance close to XGBoost.
- Gradient Boosting Regressor has decent performance but not as good as the above two models.
- Linear Regression has the worst performance and is not suitable for this problem.

=> Based on accuracy metrics, XGBoost Regressor is the best model for forecasting Walmart's weekly sales, with Random Forest Regressor also a good choice. Other models such as Gradient Boosting Regressor and Linear Regression have poorer performance and are not as suitable as the above two models.

Therefore, it is recommended to use XGBoost Regressor or Random Forest Regressor to forecast Walmart's weekly sales.

V. Conclusion

The Walmart Sales Forecasting initiative has significantly leveraged advanced data analytics and machine learning to revolutionize sales prediction and strategy formulation for Walmart. As we conclude this project, it is crucial to evaluate our accomplishments, the insights gained, and the future trajectory for Walmart's data-driven strategies.

Key Insights and Achievements

- **Data Structuring and Utilization**: By implementing a detailed star schema within our Data Warehouse, we have enabled robust, multi-dimensional data analysis. This

structure has been pivotal in addressing intricate business queries and enhancing strategic decision-making.

- **Integration of Advanced Analytical Tools**: The deployment of tools such as Power BI, Google Bigquery, and various machine learning models has streamlined our data processing and analytical capabilities. These integrations have automated data workflows and improved the accuracy and clarity of our analyses.
- **Predictive Modeling and Business Intelligence**: Utilizing Python's machine learning algorithms, we have transitioned from descriptive analytics to predictive modeling. This advancement has empowered us to forecast sales trends and identify market opportunities proactively.
- **Enhanced Decision-Making**: Our project has significantly bolstered the decision-making processes within Walmart's sales and marketing departments. Data-driven insights have led to more informed strategic decisions, aligning closely with our initial goals.
- **Customized Sales and Marketing Strategies**: In the final phase, we applied the derived insights to develop practical and tailored strategies aimed at improving sales performance, optimizing market positioning, and capitalizing on emerging trends.

Future Directions

The foundation established by this project will continue to foster Walmart's growth and operational efficiency. As data analytics and machine learning evolve, we anticipate further refining our predictive models and expanding our analytical capabilities to incorporate new data sources and market dynamics. The project has also fostered a culture of data-driven decision-making within Walmart, emphasizing the value of informed strategies and adaptive market responses. As the market landscape evolves, our approach will adapt to maintain a competitive edge in sales excellence through advanced data analytics.

In conclusion, the Walmart Sales Forecasting project exemplifies the transformative potential of data analytics in enhancing business strategy and performance. It has not only achieved its initial objectives but also paved the way for a more insightful, proactive, and data-centric approach to sales and marketing strategy in the future.

VI. References

1. R. Kimball, The Data Warehouse Toolkit. New York: Wiley Computer Publishing, 2013
2. Chung, S. (n.d). *Building a Data Mining Model using Data Warehouse and OLAP*. Available at: https://eecs.csuohio.edu/~sschung/cis611/DWDatamingMDXTutorial_611.pdf [Accessed 20 May 2024]
3. Kaula, R. (2009). *Business Rules for Data Warehouse.* Available at: https://citeseerx.ist.psu.edu/viewdoc/download?doi=10.1.1.866.4304&rep=rep1&ty%20pe=pdf [Accessed 20 May 2024]
4. Wikipedia. (n.d). *Walmart.* Available at: https://en.wikipedia.org/wiki/Walmart [Accessed 15 May 2024]

VII. Contribution

No.	Student Name Student ID	Tasks	Contribution
1	Nguyễn Thị Ngọc Lan 20070943	Building data warehouse	33.33%
2	Hoàng Thị Lan 20070942	Introduction Analysis and Answer the questions	33.33%
3	Đặng Thị Hiền 20070927	Database prepare Conclusion	33.33%