Data Pipeline Design for a Store Sales System

By

Group Members: Zheng Zheng, Yu Swe Zin Aung (Kelly)

IE 6700 Data Management

PROJECT REPORT

submitted to

Rushdi Alsaleh

Northeastern University-Vancouver

Presented April 22, 2023

Content

Introduction

Business background

Favorita is the largest retail chain in Ecuador, operating a network of supermarkets and hypermarkets nationwide. The stores offer a wide range of products across thousands of product families, including groceries, fresh produce, household items, electronics, clothing, and more. In addition, the company sources products from local and international suppliers, ensuring a wide variety of choices for customers. The company's business strategy centres around providing high-quality products at affordable prices to customers across Ecuador.

Business problems

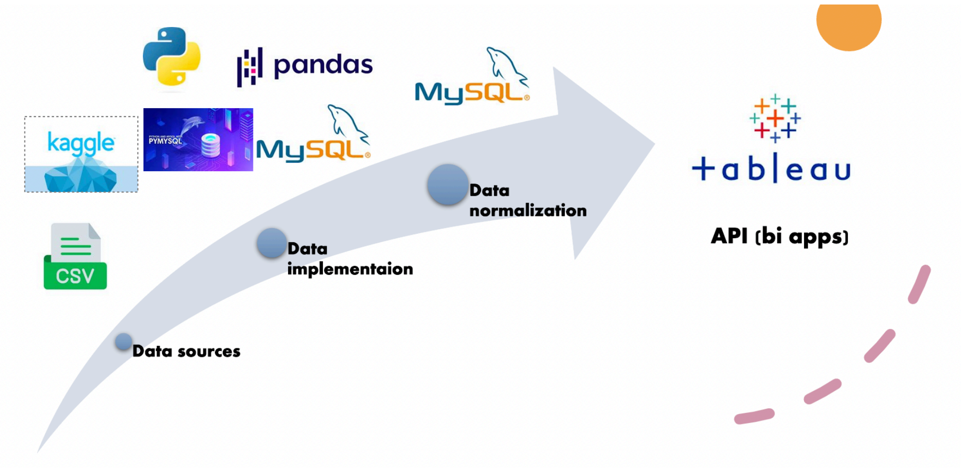
One of the primary business problems is that retailers need help managing their inventory levels. Favorita stores must optimize inventory levels to reduce waste and avoid stockouts or overstocking. Favorita identifies which products are most popular and which marketing campaigns drive the most sales to maximize marketing strategies and allocate resources more effectively. In addition, some products may experience higher demand during specific seasons or holidays. Favorita must determine these trends and adjust its inventory and marketing campaigns accordingly.

By understanding sale trends accurately, Favorita can optimize its operations, reduce waste, and provide a better shopping experience for its customers.

Requirements

Accurate sales analysis is crucial for the retail stores of Favorita, as it helps them optimize their inventory levels and reduce waste. We will use descriptive analytic to understand sales for the thousands of product families sold at Favorita stores located in Ecuador. The report involves analyzing historical sales data to identify patterns and trends and using this information to understand sales trends on time series.

Data Pipeline

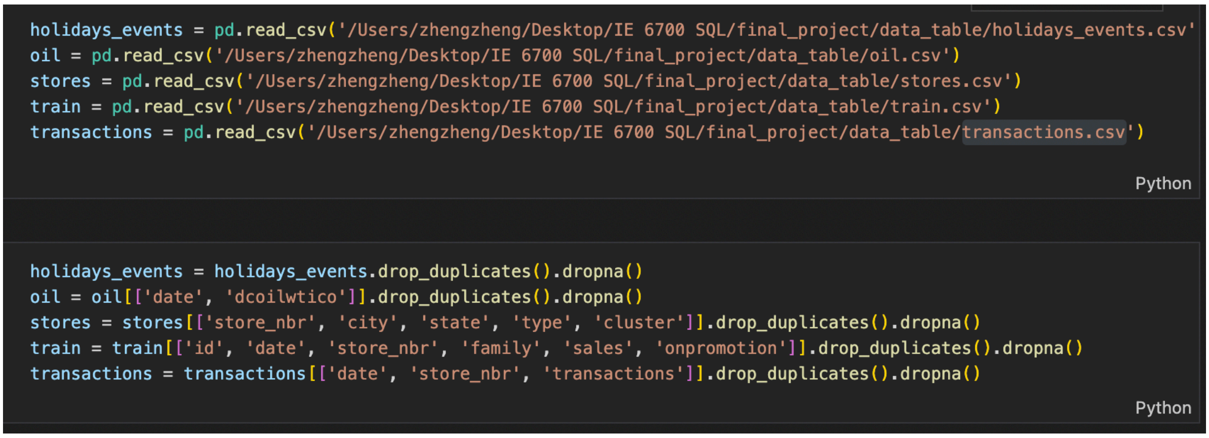


Data pipeline outline

The store sales data, which includes five csv files (holidays_events.csv, oil.csv, stores.csv, train.csv, transactions.csv), was sourced from the Kaggle website. The data was imported into Python using the "Pandas" library, and underwent preprocessing procedures, such as removal of 'NA' and duplicate entries. We established a connection to the MySQL server using the "PYMYSQL" library, created a database, and inserted the processed data into corresponding tables. The tables were then normalized using MySQL and inserted into a new database. Finally, Tableau was connected to the database and relevant dashboards were created to meet the identified business requirements.

Data implementation

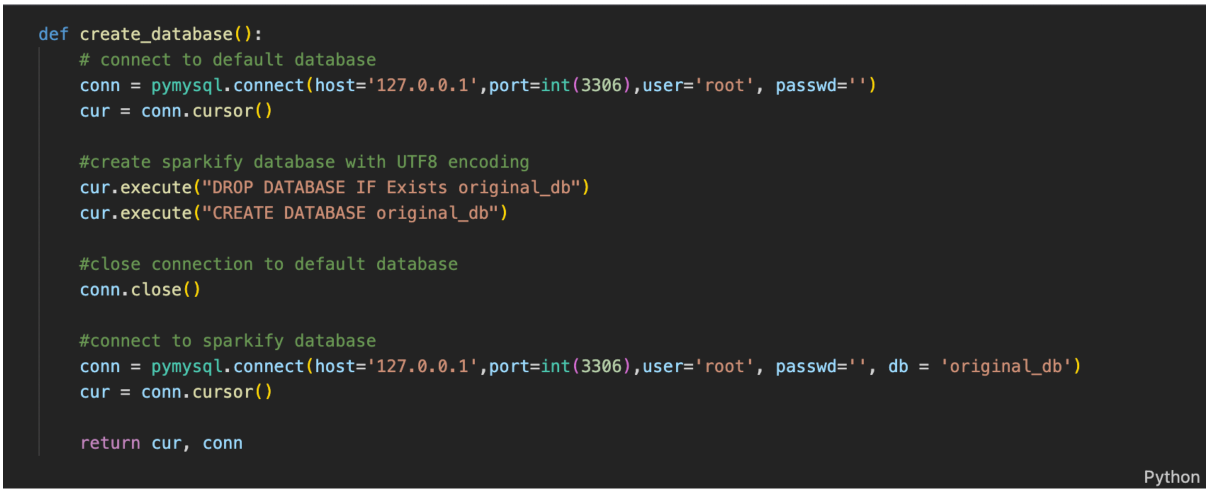
Step1: Import csv files with "Pandas" library in Python and preprocessing



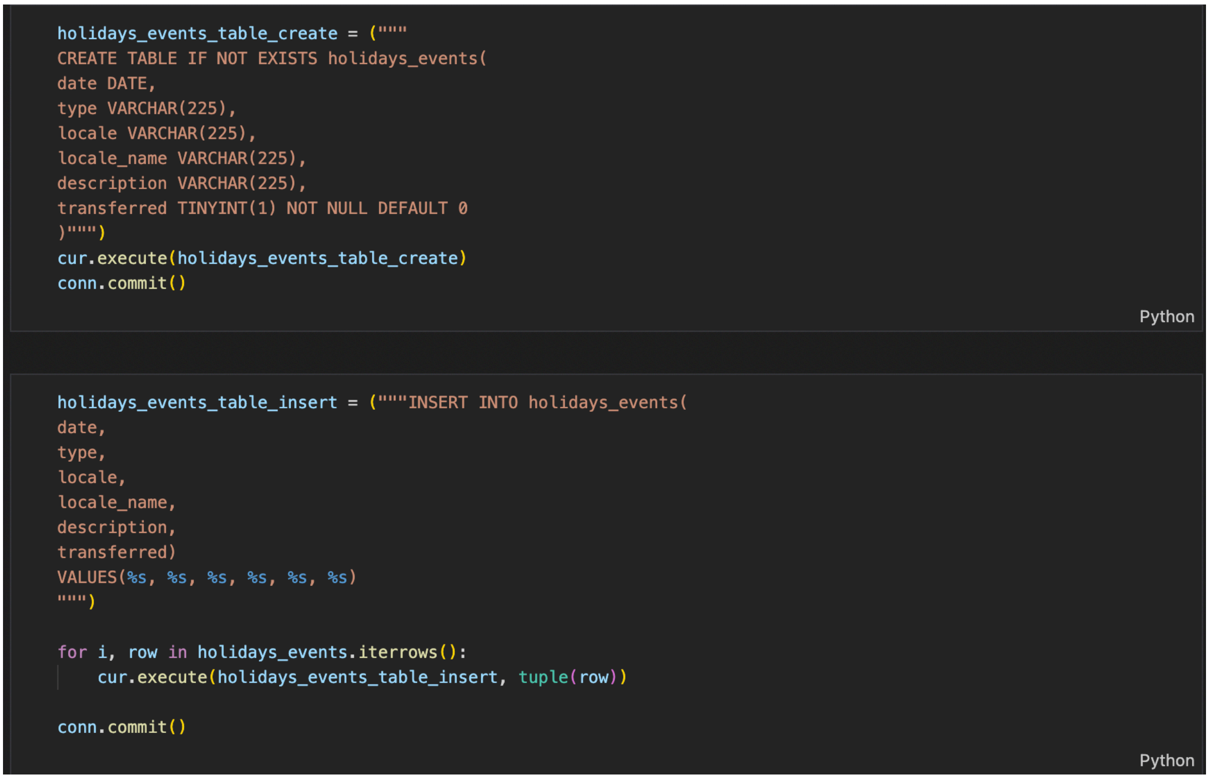
```python
holidays_events = pd.read_csv('/Users/zhengzheng/Desktop/IE 6700 SQL/final_project/data_table/holidays_events.csv'
oil = pd.read_csv('/Users/zhengzheng/Desktop/IE 6700 SQL/final_project/data_table/oil.csv')
stores = pd.read_csv('/Users/zhengzheng/Desktop/IE 6700 SQL/final_project/data_table/stores.csv')
train = pd.read_csv('/Users/zhengzheng/Desktop/IE 6700 SQL/final_project/data_table/train.csv')
transactions = pd.read_csv('/Users/zhengzheng/Desktop/IE 6700 SQL/final_project/data_table/transactions.csv')
```

```python
holidays_events = holidays_events.drop_duplicates().dropna()
oil = oil[['date', 'dcoilwtico']].drop_duplicates().dropna()
stores = stores[['store_nbr', 'city', 'state', 'type', 'cluster']].drop_duplicates().dropna()
train = train[['id', 'date', 'store_nbr', 'family', 'sales', 'onpromotion']].drop_duplicates().dropna()
transactions = transactions[['date', 'store_nbr', 'transactions']].drop_duplicates().dropna()
```

Step 2: Connect to MySQL server, create database, create tables, and insert data



```python
def create_database():
    # connect to default database
    conn = pymysql.connect(host='127.0.0.1',port=int(3306),user='root', passwd='')
    cur = conn.cursor()

    #create sparkify database with UTF8 encoding
    cur.execute("DROP DATABASE IF Exists original_db")
    cur.execute("CREATE DATABASE original_db")

    #close connection to default database
    conn.close()

    #connect to sparkify database
    conn = pymysql.connect(host='127.0.0.1',port=int(3306),user='root', passwd='', db = 'original_db')
    cur = conn.cursor()

    return cur, conn
```

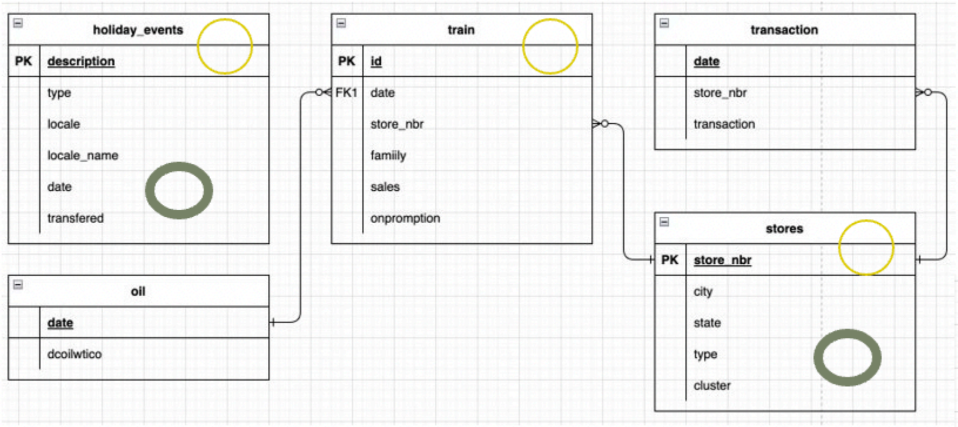


```python
holidays_events_table_create = ("""
CREATE TABLE IF NOT EXISTS holidays_events(
date DATE,
type VARCHAR(225),
locale VARCHAR(225),
locale_name VARCHAR(225),
description VARCHAR(225),
transferred TINYINT(1) NOT NULL DEFAULT 0
)""")
cur.execute(holidays_events_table_create)
conn.commit()
```

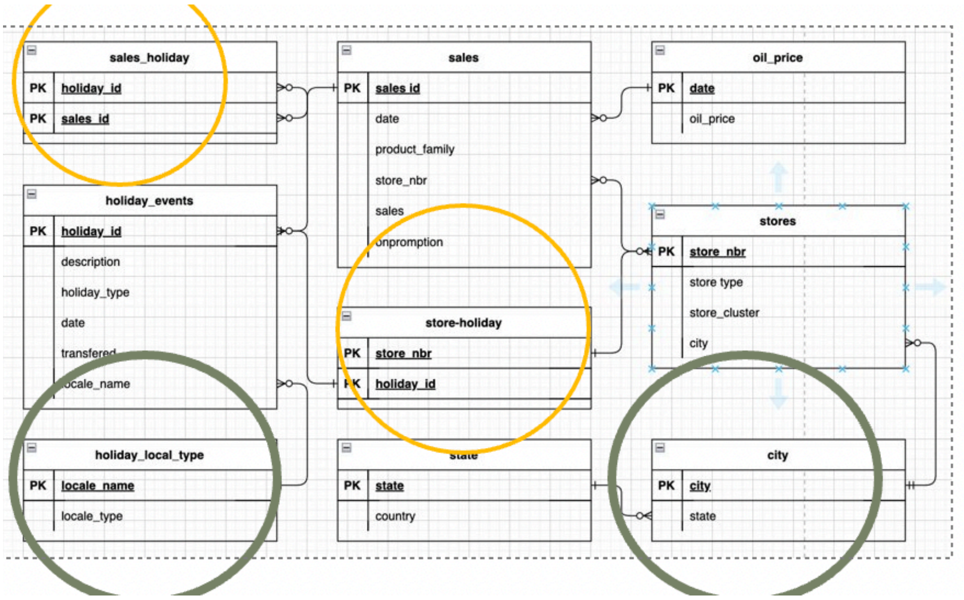
```python
holidays_events_table_insert = ("""INSERT INTO holidays_events(
date,
type,
locale,
locale_name,
description,
transferred)
VALUES(%s, %s, %s, %s, %s, %s)
""")

for i, row in holidays_events.iterrows():
    cur.execute(holidays_events_table_insert, tuple(row))

conn.commit()
```

Here we only listed the codes for one table as an example.

Data normalization



ERD of original database



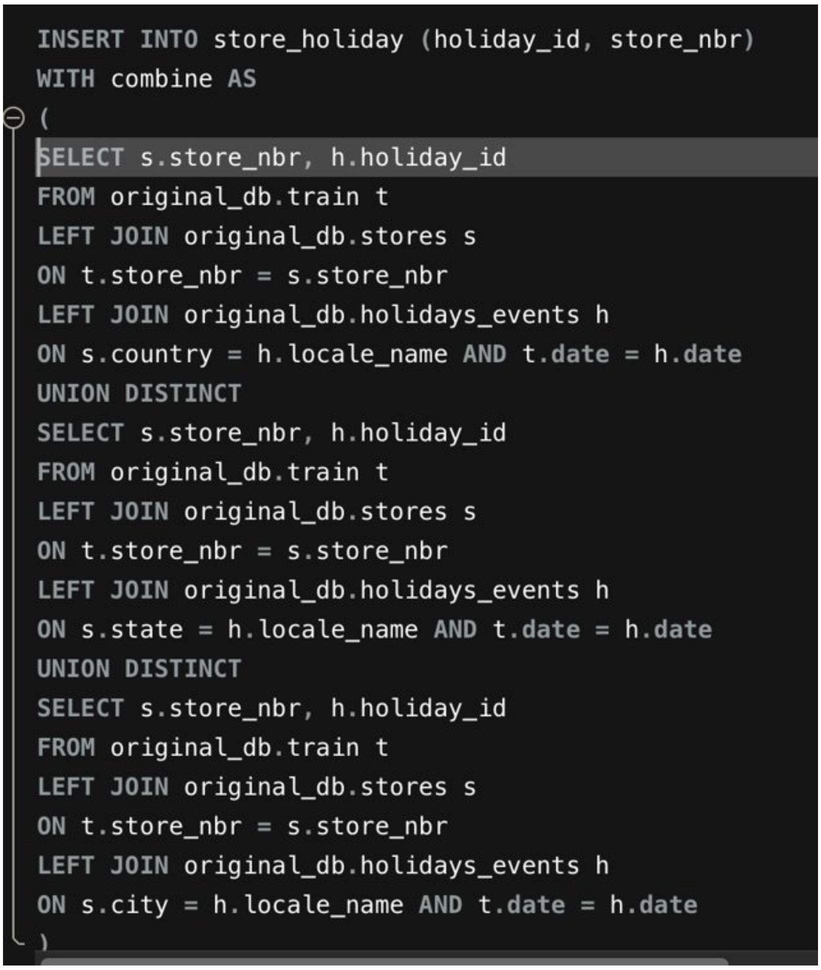
ERD of new database

The main issue with the database was the complex many-to-many relationship between the holiday_events, train, and stores tables, which are marked with orange circles on the ERD. To solve this problem, we involved two transaction tables, sales_holiday table, and store_holiday table.

The second issue with the original database was the transitive dependency, which are marked with red circles. Let's use the 'stores' table as an example. According to the definition of NF3 table, all non-primary keys should depend on the primary keys. But in this table, the 'country' key doesn't directly depend on the primary key store_nbr. Instead, it is transitively dependent on the primary key store_nbr. Thus, it was necessary to pick out those keys to eliminate the transitive dependency relationship.

Step 1 solving the many-to-many issue

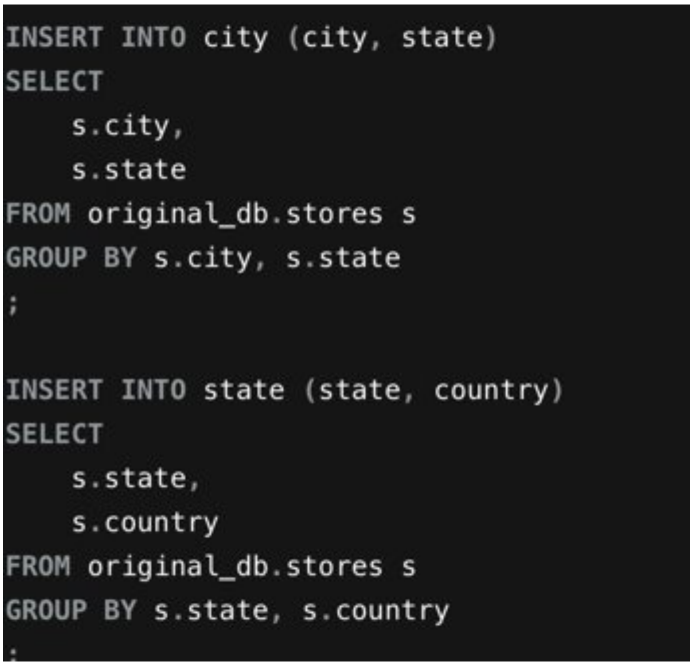
Firstly, we used a left join to join the holiday_events, train, and stores tables. As some holidays are national level, some are state level, and some are city level, we used a left join to build three sub-tables. We then used the distinct union function to combine all three sub-tables into one.



```
INSERT INTO store_holiday (holiday_id, store_nbr)
WITH combine AS
(
SELECT s.store_nbr, h.holiday_id
FROM original_db.train t
LEFT JOIN original_db.stores s
ON t.store_nbr = s.store_nbr
LEFT JOIN original_db.holidays_events h
ON s.country = h.locale_name AND t.date = h.date
UNION DISTINCT
SELECT s.store_nbr, h.holiday_id
FROM original_db.train t
LEFT JOIN original_db.stores s
ON t.store_nbr = s.store_nbr
LEFT JOIN original_db.holidays_events h
ON s.state = h.locale_name AND t.date = h.date
UNION DISTINCT
SELECT s.store_nbr, h.holiday_id
FROM original_db.train t
LEFT JOIN original_db.stores s
ON t.store_nbr = s.store_nbr
LEFT JOIN original_db.holidays_events h
ON s.city = h.locale_name AND t.date = h.date
)
```

The example code to solve the transitive dependency issue

Step 2 solving the transitive dependency issue



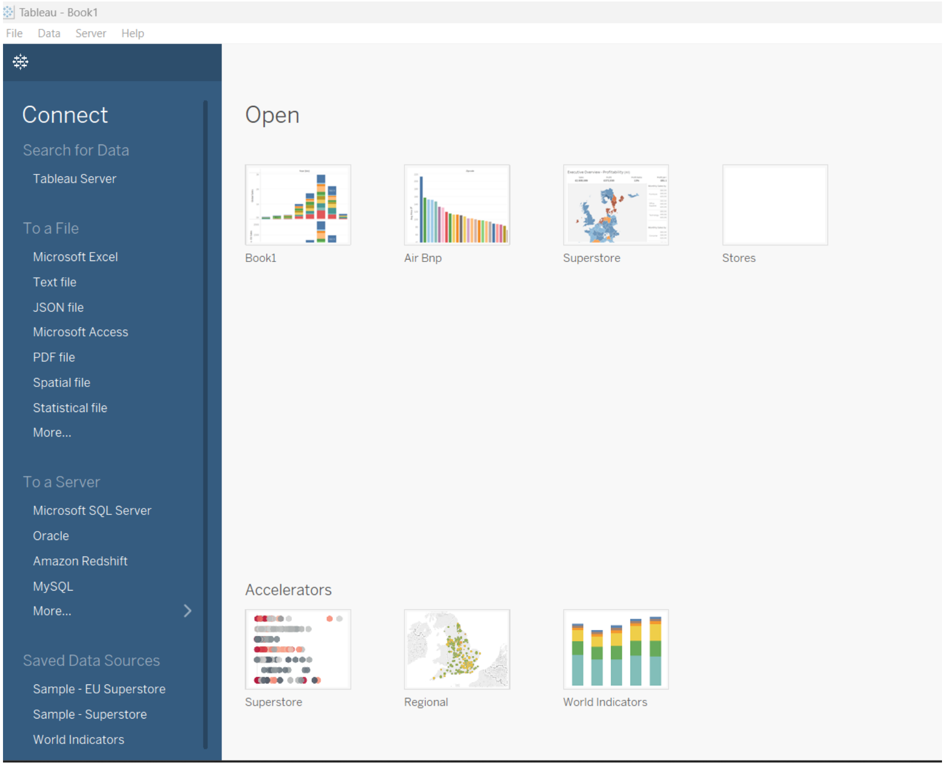
```
INSERT INTO city (city, state)
SELECT
    s.city,
    s.state
FROM original_db.stores s
GROUP BY s.city, s.state
;

INSERT INTO state (state, country)
SELECT
    s.state,
    s.country
FROM original_db.stores s
GROUP BY s.state, s.country
;
```

The example code to solve the transitive dependency issue

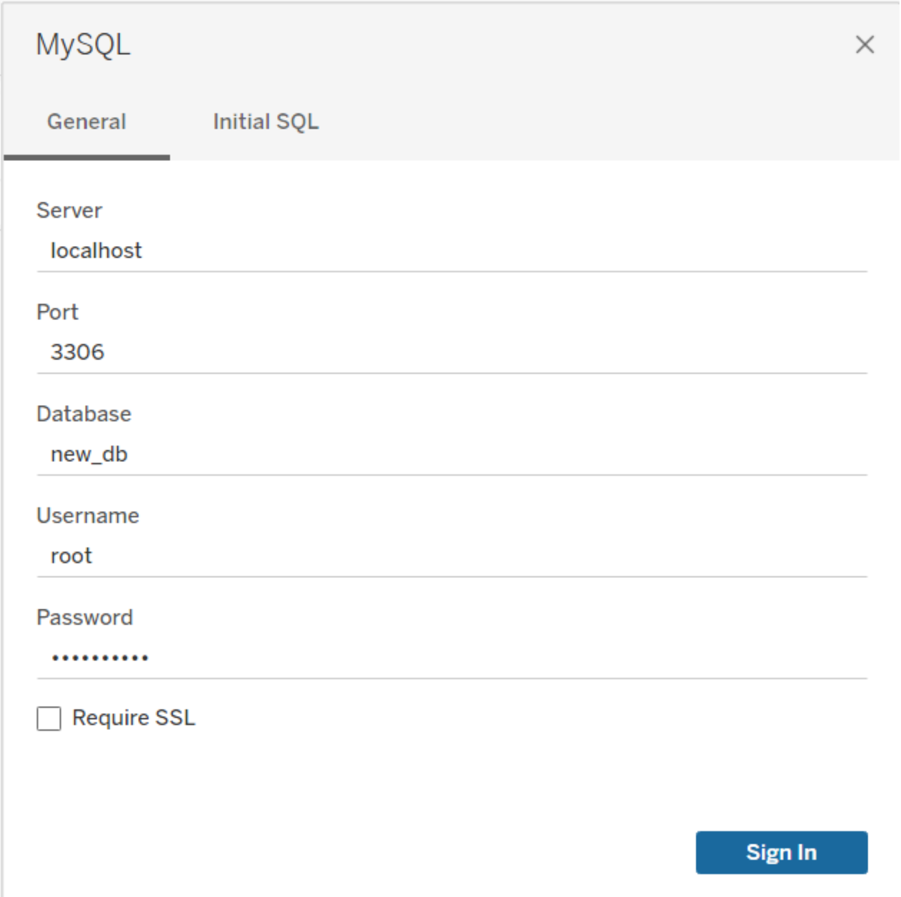
Data Visualization on Tableau

Connect Tableau with MySQL

Step 1: Click on MySQL



Step 2: To establish a connection to the local database, we need to set up the server as [localhost]. In this particular project, we have created a database named [store-sales] and defined its name in the connection settings. Once filling in the necessary information in the form, it can proceed to click on the [Sign in] button.



MySQL

General | Initial SQL

Server: localhost

Port: 3306

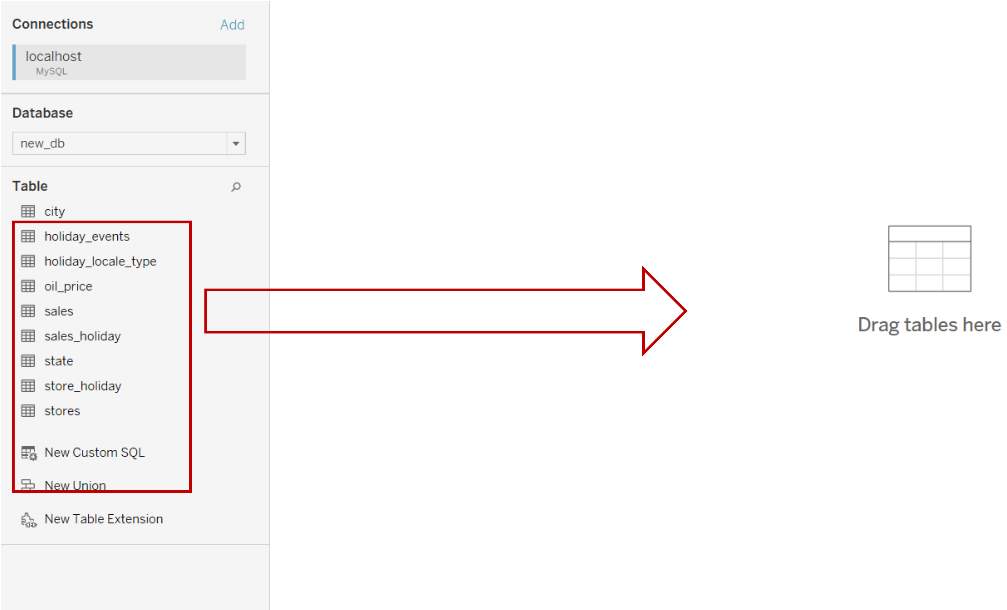
Database: new_db

Username: root

Password: ••••••••••

□ Require SSL

Step 3: we need to drag and drop all six tables into the designated area labeled "Drag tables." This is because we will be utilizing all the available data from these tables for the purpose of conducting descriptive analysis.

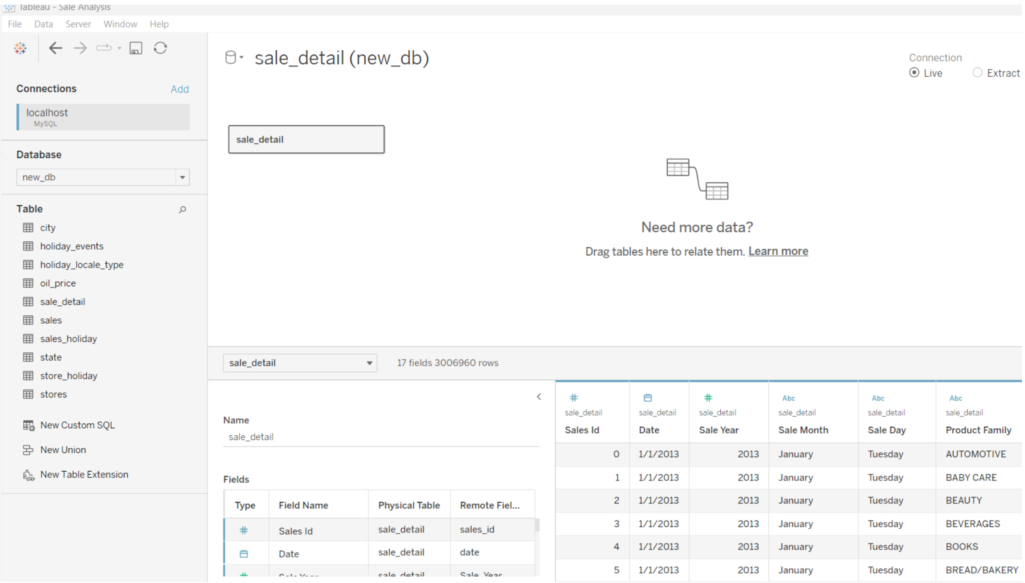


Connections: localhost (MySQL)

Database: new_db

Table:
- city
- holiday_events
- holiday_locale_type
- oil_price
- sales
- sales_holiday
- state
- store_holiday
- stores
- New Custom SQL
- New Union
- New Table Extension

Due to the large size of the data, generating an analytic report in Tableau was too slow. To address this issue, a view table was created by joining all the tables required for the analysis. The query for this view table can be found in Appendix A: View Table Query. The view table is called [sale_detail], and since it contains all the necessary attributes, it will be the only table used in the Tableau data box.



Name
sale_detail

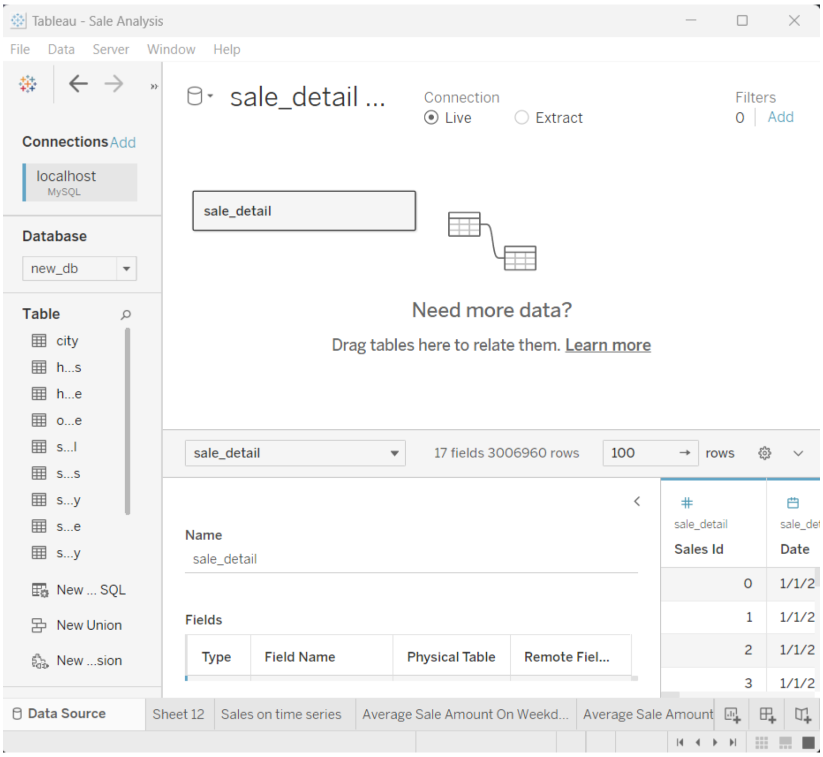
Fields

Type	Field Name	Physical Table	Remote Fiel...
#	Sales Id	sale_detail	sales_id
📅	Date	sale_detail	date

sale_detail Sales Id	sale_detail Date	sale_detail Sale Year	sale_detail Sale Month	sale_detail Sale Day	sale_detail Product Family
0	1/1/2013	2013	January	Tuesday	AUTOMOTIVE
1	1/1/2013	2013	January	Tuesday	BABY CARE
2	1/1/2013	2013	January	Tuesday	BEAUTY
3	1/1/2013	2013	January	Tuesday	BEVERAGES
4	1/1/2013	2013	January	Tuesday	BOOKS
5	1/1/2013	2013	January	Tuesday	BREAD/BAKERY

Creating charts and dashboard on Tableau

In order to conduct a descriptive analysis, it is necessary to produce graphs. Each graph needs to be created on a separate sheet in Tableau. To make a new sheet, it should click on the "New Worksheet" icon located at the bottom of the page, as shown in the following figure



Name
sale_detail

Fields

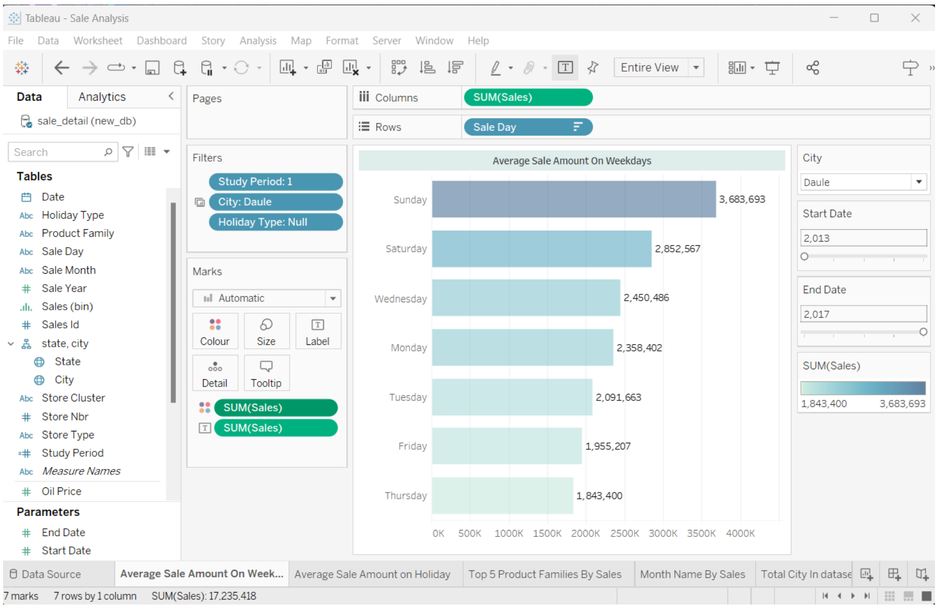
Type	Field Name	Physical Table	Remote Fiel...

sale_detail Sales Id	sale_det Date
0	1/1/2
1	1/1/2
2	1/1/2
3	1/1/2

After creating the new sheet, it will see the list of available data attributes on the left side of the new sheet as per the following figure.

Those data columns will be dragged drop on the [Columns, Rows, Filters, Marks, Page] section as per analytic chart requirements—one example of a tableau chart as per the following figure.

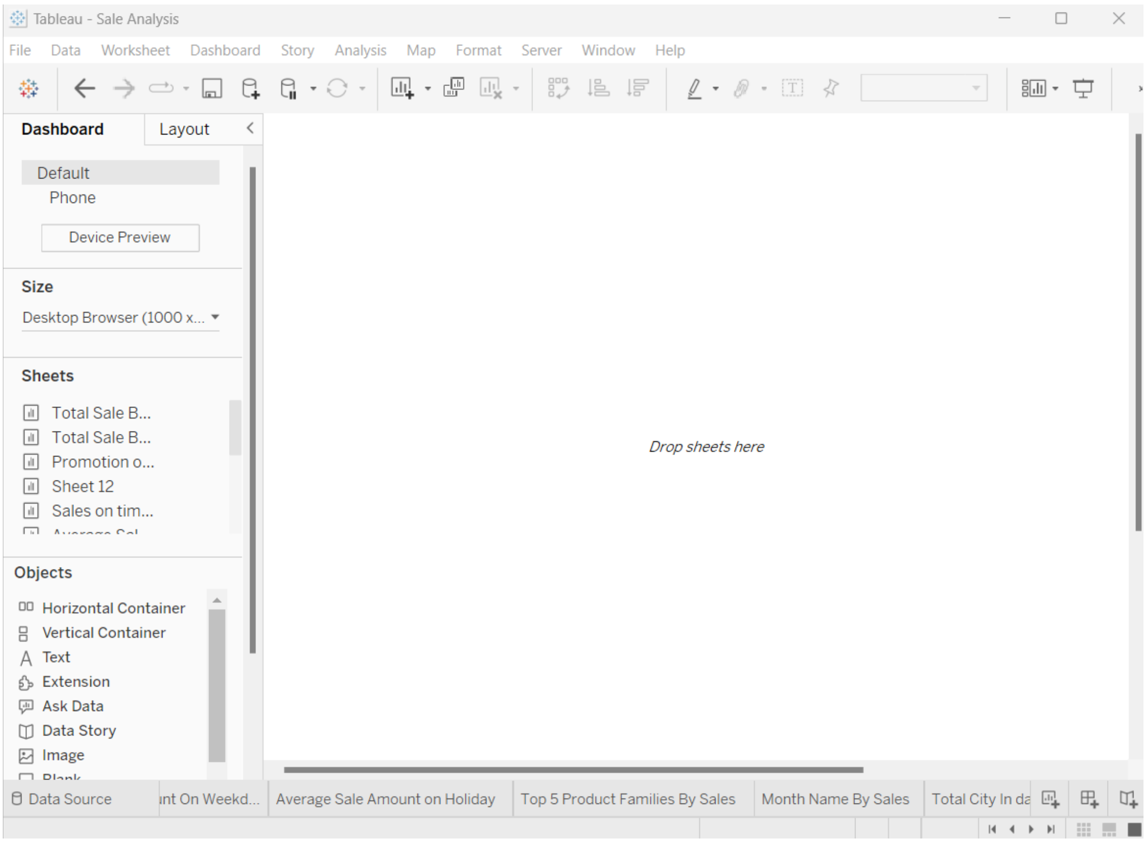
After all the required descriptive analytic charts are completed, all those charts will be allocated in place. It is called a dashboard on the tableau. To create a new dashboard, the dashboard logo is clicked at the bottom of the page as per the following figure.



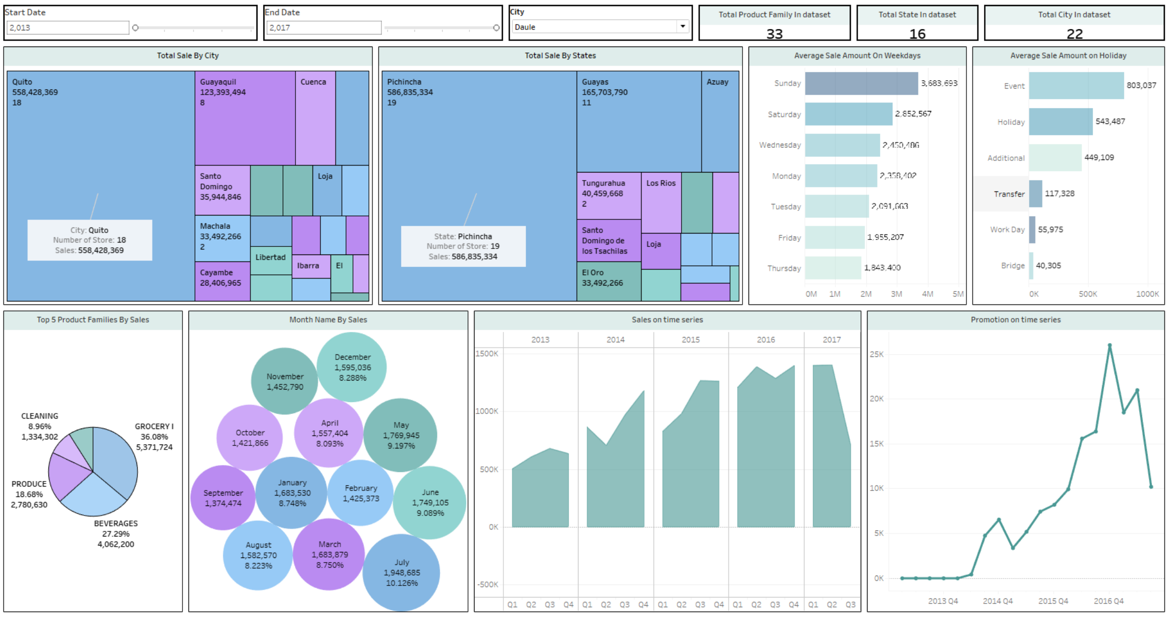
Columns: SUM(Sales)
Rows: Sale Day
Filters: Study Period: 1, City: Daule, Holiday Type: Null
Marks: Automatic; SUM(Sales), SUM(Sales)
City: Daule
Start Date: 2,013
End Date: 2,017
SUM(Sales): 1,843,400 – 3,683,693

7 marks 7 rows by 1 column SUM(Sales): 17,235,418

After creating a new dashboard, the empty dashboard can be seen in the following figure. All the designed sheets can be accessed in the [Sheets] section. Each sheet can be dragged and dropped into the [Drop sheets here] area on the dashboard.



After arranging all the sheets on the dashboard, the complete dashboard can be viewed in the following figure.



Descriptive analysis

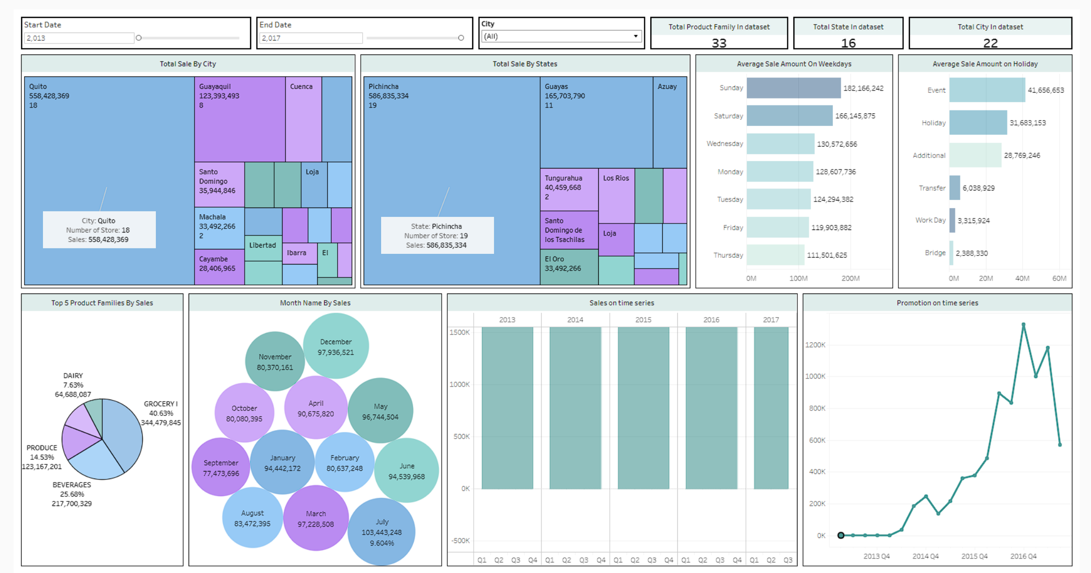
Since the dashboard is dynamic to analyze the sale data on year range as per data picker of Start Date and End Date, 2013 is selected as the start date, and 2017 is selected as the end date.

City Quito has the highest sales with 18 stores. It is seen that 50 per cent of company sales come from the city.

By default, it chooses all cities to study the sale performance. However, it can select one specific city from the city dropdown list based on user requirements.

When comparing the average sale amount on weekdays and holidays, there is a similar average sale on holidays from weekdays. It has the highest average deal on Sunday, followed by Saturday.

Grocery and Beverage family products have a high percentage of sale amount comparing other family products. The sale trend is consistent from 2013 to 2017. However, they started promotion in 2014, and the promotion sale dramatically increased till 2016. After that, the promotional sales dropped significantly.



Start Date 2,013

End Date 2,017

City (All)

Total Product Family In dataset 33

Total State In dataset 16

Total City In dataset 22