UNIVERSITI MALAYSIA PAHANG
AL-SULTAN ABDULLAH

BSD2343 DATA WAREHOUSING

2023/2024 SEMESTER II

GROUP NAME: KIMBALL

TITLE:

Full Cycle of Coffee Shop Management: Inventory, Orders, and Staffing (SGD 8: DECENT WORK AND ECONOMIC GROWTH)

PREPARE FOR:

DR NOR AZUANA BINTI RAMLI

PREPARED BY:

MATRIC ID	NAME	SECTION
SD22056	ELIANE HO WAN WEN	01G
SD22061	ANG MEI YING	01G
SD22016	SRI SHAMNEE SAI A/P RAJAH	02G
SD22032	NUR A'RIFAH AKMAL BINTI HUSSIN	01G
SD22051	MOHAMMAD BASHARIL AIMAN BIN MOHAMMAD BAKHTIAR	01G

TABLE OF CONTENT

1.0 BACKGROUND

1.1 Project Background

In the bustling world of coffee shops, understanding and optimizing operations are critical for success. With the increasing reliance on data-driven decision-making, a comprehensive dataset like "Coffee Shop Sales/Inventory/Staff" presents an invaluable resource for extracting actionable insights.

The "Coffee Shop Sales/Inventory/Staff" dataset is a meticulously curated collection encompassing various facets of coffee shop operations. It comprises several interrelated tables, each offering granular insights into different aspects of the business, including customer orders, inventory management, staff details, and shift scheduling. With its structured format and comprehensive coverage, the dataset serves as a foundation for in-depth analysis and informed decision-making.

Our primary goal is to integrate disparate data sources into a unified, coherent data model within the data warehouse. This involves reconciling data from the Orders, Items, Recipes, Ingredients, Inventory, Staff, Shift, and Rota tables to establish meaningful relationships and ensure data consistency and accuracy.

With the data warehouse in place, we aim to equip stakeholders with robust analytical capabilities to gain actionable insights into coffee shop operations. From assessing business performance metrics such as total sales and average order value to delving into inventory management and staff efficiency, our objective is to empower stakeholders with the tools to make data-driven decisions.

Efficient querying and data retrieval are crucial for a responsive and scalable data warehouse. Thus, we'll focus on optimizing data storage, indexing, and query execution to ensure optimal performance, even with large volumes of data.

Drawing inspiration from previous analyses conducted using the dataset, we aim to explore a myriad of questions and uncover valuable insights, including but not limited to:

- Total Orders and Sales Trends
- Menu Performance and Customer Preferences
- Inventory Management and Replenishment Strategies
- Staff Efficiency and Labor Cost Optimization

A well-defined schema will be developed, outlining the structure and relationships between various tables within the data warehouse. This schema will serve as the blueprint for organizing and storing data effectively. Extract, Transform, Load (ETL) processes will be established to populate the data warehouse with data from the source dataset. These processes will ensure that data is extracted efficiently, transformed into the appropriate format, and loaded into the warehouse while maintaining data quality and integrity. Predefined queries and reports will be developed to enable stakeholders to explore key metrics and glean actionable insights from the data stored in the warehouse. These queries and reports will provide valuable insights into coffee shop operations, allowing stakeholders to make informed decisions. The performance of the data warehouse will be evaluated through benchmarking exercises. This will involve measuring various performance metrics, including query response times and resource utilization, to identify areas for optimization and ensure that the warehouse operates efficiently. Comprehensive documentation will be provided, elucidating the data warehouse architecture, ETL processes, and analytical capabilities. Additionally, user training sessions will be conducted to ensure that stakeholders are proficient in utilizing the warehouse to its full potential, facilitating adoption and utilization.

These deliverables are essential components of the project, ensuring that the data warehouse is effectively implemented, optimized for performance, and utilized to drive informed decision-making in coffee shop operations.

By harnessing the power of data warehousing and analytics, we aspire to unlock the full potential of the "Coffee Shop Sales/Inventory/Staff" dataset, empowering coffee shop owners and managers to make informed decisions, enhance operational efficiency, and ultimately, delight customers with exceptional experiences.

1.2 Description of Data

The dataset that we used for this study is “Coffee Shop Data” . The data is divided into many tables, including orders, items, recipes, ingredients, inventory, staff, shift and rota. Each of them focuses on a different aspect of the business.

Orders Table

Variable	Data Type	Description
row_id	Integer	A unique identifier is assigned to each row.
order_id	String	A unique identifier is assigned to each order.
created_at	String	The timestamp when the order was received.
item_id	String	A unique identifier is assigned to each item.
quantity	Integer	The quantity of order.
cust_name	String	The customer’s name.
in_or_out	String	Key for whether the order was dine-in or takeout.

Items Table

Variable	Data Type	Description
item_id	String	A unique identifier is assigned to each item.
sku	String	The stock keeping unit.
item_name	String	The name of the item.
item_cat	String	The category of item.
item_size	String	The size for item.
item_price	Float	The price for item.

Recipes Table

Variable	Data Type	Description
row_id	Integer	A unique identifier is assigned to each row.
recipe_id	String	A unique identifier is assigned to each recipe.
ing_id	String	A unique identifier is assigned to each ingredient.
quantity	Integer	The quantity for the ingredient.

Ingredients

Variable	Data Type	Description
ing_id	String	A unique identifier is assigned to each ingredient.
ing_name	String	The name of the ingredient.
ing_weight	Integer	The weight of the ingredients.
ing_meas	String	The measurement name for the ingredient.
ing_price	Float	The price for ingredients.

Inventory

Variable	Data Type	Description
inv_id	String	A unique identifier is assigned to each inventory.
ing_id	String	A unique identifier is assigned to each ingredient.
quantity	Integer	The quantity for each ingredient.

Staff Table

Variable	Data Type	Description
staff_id	String	A unique identifier is assigned to each staff member.
first_name	String	The first name of the staff.
last_name	String	The last name of the staff.
position	String	The position of staff.
sal_per_hour	Integer	The salary for staff per hour.

Shift Table

Variable	Data Type	Description
shift_id	String	A unique identifier is assigned to each shift.
day_of_week	String	The day that staff worked.
start_time	Integer	The staff started working.
end_time	Integer	The time staff ended working.

Rota Table

Variable	Data Type	Description
row_id	Integer	A unique identifier is assigned to each row.
rota_id	String	A unique identifier is assigned to each staff working schedule.
date	Integer	Staff working date.
shift_id	String	A unique identifier is assigned to each shift.
staff_id	String	A unique identifier is assigned to each staff member.

1.3 Problem to be Solved

A coffee shop should offer a variety of drinks and food. The company also needs to consider factors about pricing. It should be competitive yet profitable, taking into account ingredient costs, overhead expenses, and market trends. Besides, managing inventory efficiently, arrange schedules for employees are also crucial elements that businesses must consider so that business operations run smoothly.

However, there are a few issues that will exist. One of them is inventory management. How can the coffee shop make sure it doesn't run out of ingredients while also keeping costs low? To determine the best rates for each menu item, this requires looking at the costs of ingredients, the prices of competitors, customer preferences, and the flexibility of demand. The objective is to determine the best price strategy that increases profits and revenue while not reducing customer purchases. Furthermore, menu pricing strategy is also the issue that businesses have to face. How should the restaurant price its menu items to maximise the profit while competing in the market? It needs to analyse ingredient costs, competitor pricing, customer preferences, and customers demand to set optimal prices for each menu item. The goal is to find the pricing strategy that maximises profit without discouraging customer purchases. The last of them is staff scheduling. How does the management team consider scheduling its staff shifts to reduce labour costs and provide sufficient coverage during peak hours? This involves factors such as staff availability, peak demand times, and labour laws to create efficient schedules. The objective is to provide excellent service without overstaffing or understaffing.

To overcome the problem faced, the businesses can use software to put a data driven approach into inventory management that tracks ingredient usage, current inventory levels, and supplier information. This system is inspired by big companies like Lotus’s and Marrybrown. The system would analyse historical order data to predict demand and automatically reorder quantities for each ingredient. While applying this system, it can minimise stockout while supplying enough inventory to meet the customer demands. For staff scheduling, businesses can use scheduling software that construct effective shift plans. By improving staffing levels based on peak hours and minimising labour costs through effective shifts, it can maximise productivity and employee satisfaction. Lastly, for menu pricing, businesses need to analyse costs of ingredients, the prices of competitors, customer preferences, and the flexibility of demand. By using analytics tools, the businesses can identify pricing strategies like offering good deals and promotions, and updating menu prices based on market trends that maximise revenue and profit without discouraging customers.

1.4 Objectives

The objective of this project is:

1. To examine customer behaviours and order patterns

2. To determine the cost control measures by monitoring ingredient usage and inventory levels

3. To observe staff workloads to optimise scheduling and improve operational effectiveness

4. To determine the sales and revenue trend of Coffee Shop

1.5 Data Schema

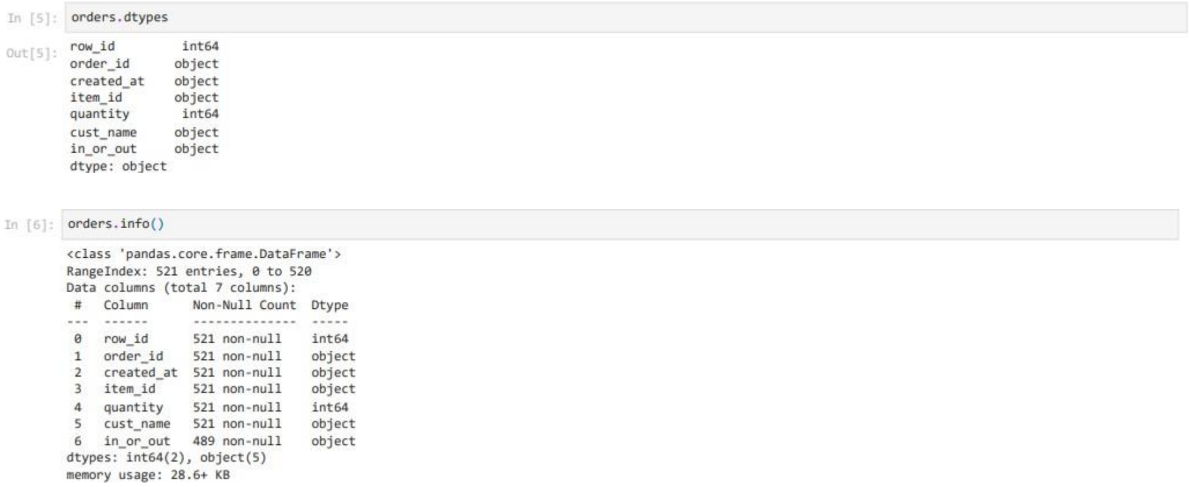
A data schema is a collection of database objects, including tables, views, indexes and synonyms. There are a variety of ways of arranging schema objects in the schema models designed for the data warehousing. The coffee shop dataset consists of eight tables which are orders, items, recipe, rota, ingredients, inventory, staff and shift.

We used 2 libraries in jupyter to display data schema which are pandas and numpy.

import pandas as pd

import numpy as np

- For orders

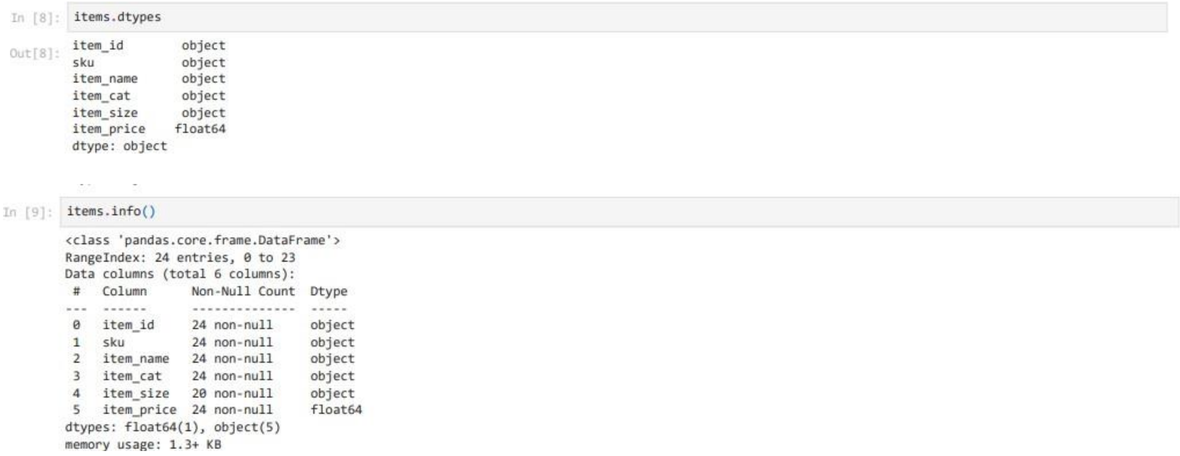


```
In [5]: orders.dtypes

Out[5]: row_id          int64
        order_id       object
        created_at     object
        item_id        object
        quantity        int64
        cust_name      object
        in_or_out      object
        dtype: object


In [6]: orders.info()

        <class 'pandas.core.frame.DataFrame'>
        RangeIndex: 521 entries, 0 to 520
        Data columns (total 7 columns):
         #   Column      Non-Null Count  Dtype 
        ---  ------      --------------  ----- 
         0   row_id      521 non-null    int64 
         1   order_id    521 non-null    object
         2   created_at  521 non-null    object
         3   item_id     521 non-null    object
         4   quantity    521 non-null    int64 
         5   cust_name   521 non-null    object
         6   in_or_out   489 non-null    object
        dtypes: int64(2), object(5)
        memory usage: 28.6+ KB
```

Figure 1.5.1: Orders tables

Based on Figure 1.5.1 above, the data schema for the orders consists of seven columns. This data frame consists of only two data types which are integer and string. There are two columns that are integer data types while the rest are string data types.

- For items



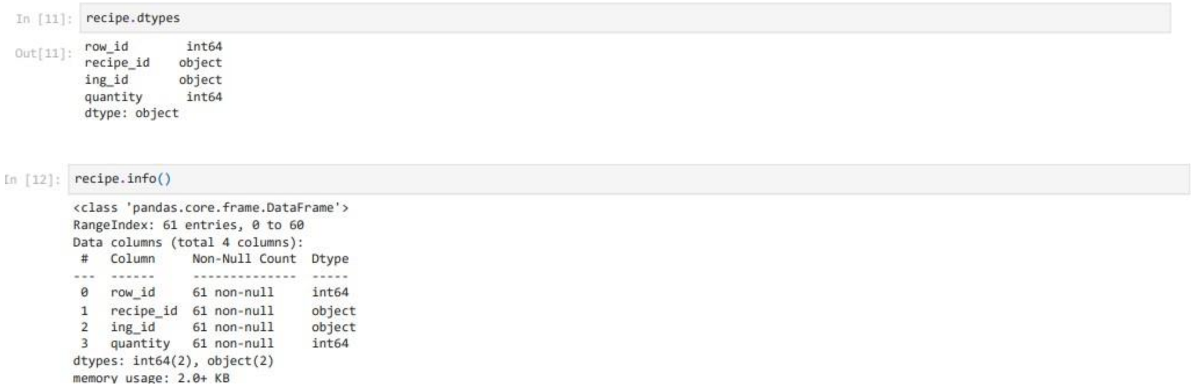
```
In [8]:  items.dtypes

Out[8]:  item_id       object
         sku           object
         item_name     object
         item_cat      object
         item_size     object
         item_price   float64
         dtype: object
```

```
In [9]:  items.info()

         <class 'pandas.core.frame.DataFrame'>
         RangeIndex: 24 entries, 0 to 23
         Data columns (total 6 columns):
          #   Column      Non-Null Count  Dtype
         ---  ------      --------------  -----
          0   item_id     24 non-null     object
          1   sku         24 non-null     object
          2   item_name   24 non-null     object
          3   item_cat    24 non-null     object
          4   item_size   20 non-null     object
          5   item_price  24 non-null     float64
         dtypes: float64(1), object(5)
         memory usage: 1.3+ KB
```

Figure 1.5.2: Items tables

Based on Figure 1.5.2 above, the data schema for the items consists of six columns. This data frame consists of float and string data types. Only one column has float data types while the rest are string data types.

- For recipe



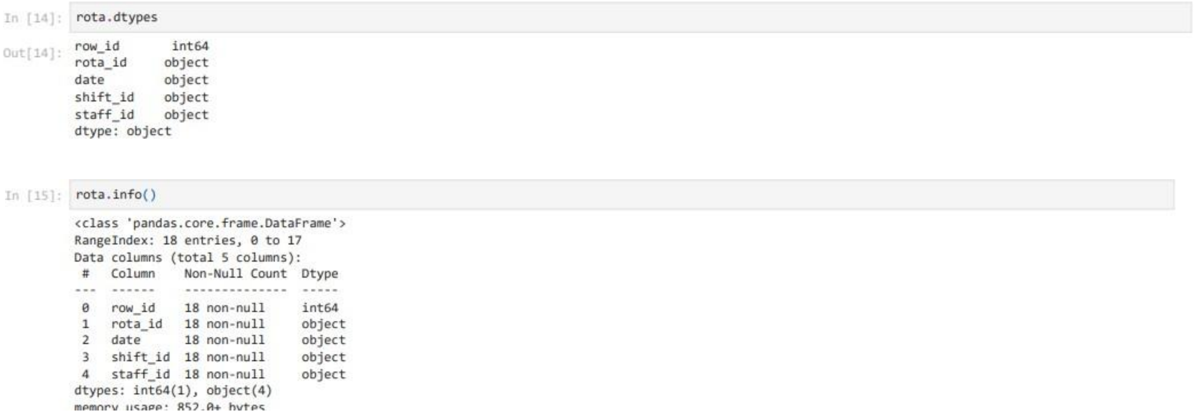
```
In [11]:  recipe.dtypes

Out[11]:  row_id        int64
          recipe_id    object
          ing_id       object
          quantity      int64
          dtype: object
```

```
In [12]:  recipe.info()

          <class 'pandas.core.frame.DataFrame'>
          RangeIndex: 61 entries, 0 to 60
          Data columns (total 4 columns):
           #   Column     Non-Null Count  Dtype
          ---  ------     --------------  -----
           0   row_id     61 non-null     int64
           1   recipe_id  61 non-null     object
           2   ing_id     61 non-null     object
           3   quantity   61 non-null     int64
          dtypes: int64(2), object(2)
          memory usage: 2.0+ KB
```

Figure 1.5.3: Recipe tables

Based on Figure 1.5.3 above, the data schema for the recipe consists of four columns. This data frame consists of integer and string data types. There are two columns that are integer data types while the other two columns are string data types.

- For rota



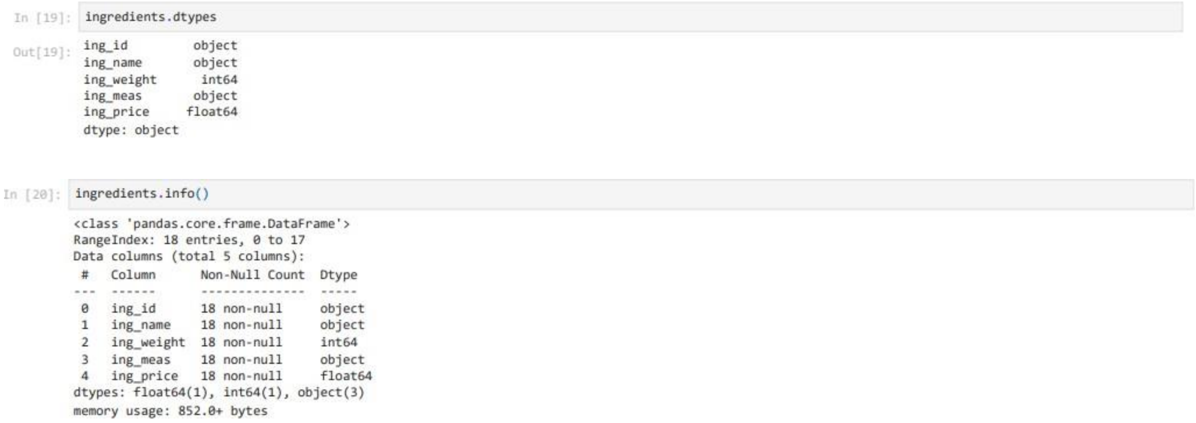
```
In [14]: rota.dtypes

Out[14]: row_id        int64
         rota_id      object
         date         object
         shift_id     object
         staff_id     object
         dtype: object
```

```
In [15]: rota.info()

         <class 'pandas.core.frame.DataFrame'>
         RangeIndex: 18 entries, 0 to 17
         Data columns (total 5 columns):
          #   Column    Non-Null Count  Dtype
         ---  ------    --------------  -----
          0   row_id    18 non-null     int64
          1   rota_id   18 non-null     object
          2   date      18 non-null     object
          3   shift_id  18 non-null     object
          4   staff_id  18 non-null     object
         dtypes: int64(1), object(4)
         memory usage: 852.0+ bytes
```

Figure 1.5.4: Rota tables

Based on Figure 1.5.4 above, the data schema for the items consists of five columns. This data frame consists of integer and string data types. Only one column has integer data types while the rest are string data types.

- For ingredients



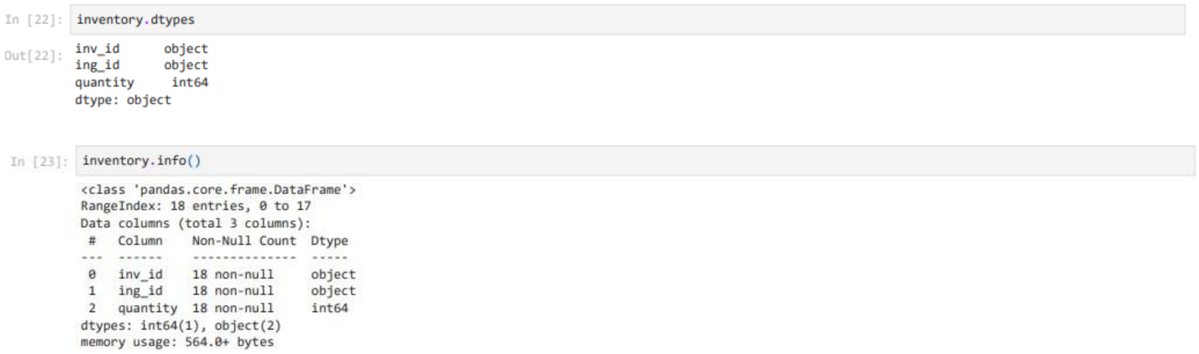
```
In [19]: ingredients.dtypes

Out[19]: ing_id         object
         ing_name       object
         ing_weight      int64
         ing_meas       object
         ing_price     float64
         dtype: object
```

```
In [20]: ingredients.info()

         <class 'pandas.core.frame.DataFrame'>
         RangeIndex: 18 entries, 0 to 17
         Data columns (total 5 columns):
          #   Column      Non-Null Count  Dtype
         ---  ------      --------------  -----
          0   ing_id      18 non-null     object
          1   ing_name    18 non-null     object
          2   ing_weight  18 non-null     int64
          3   ing_meas    18 non-null     object
          4   ing_price   18 non-null     float64
         dtypes: float64(1), int64(1), object(3)
         memory usage: 852.0+ bytes
```

Figure 1.5.5: Ingredients tables

Based on Figure 1.5.5 above, the data schema for the ingredients consists of five columns. This data frame consists of string, integer and float data types. One column has the data type of integer which is “ing_weight”, float for one column which is “ing_price” and three columns have the string data types.

- For inventory



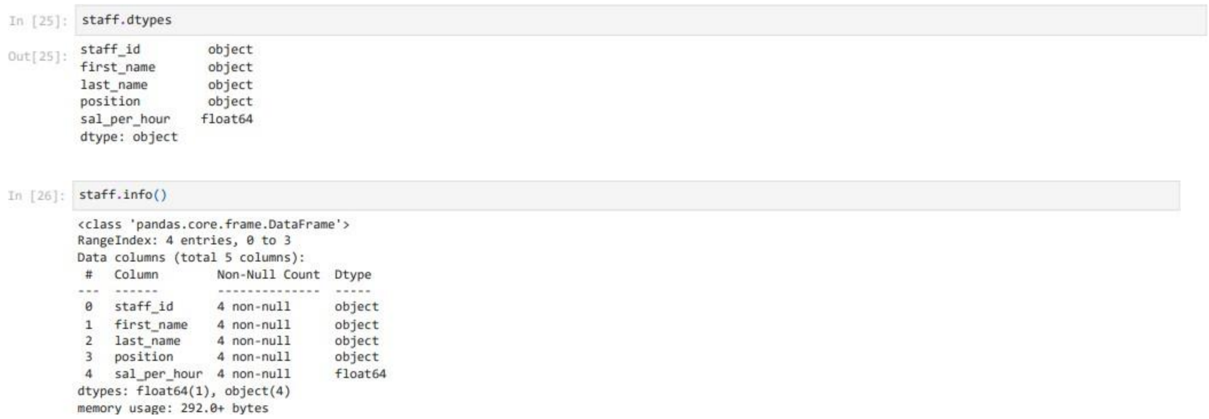
```
In [22]: inventory.dtypes

Out[22]: inv_id      object
         ing_id      object
         quantity     int64
         dtype: object
```

```
In [23]: inventory.info()

<class 'pandas.core.frame.DataFrame'>
RangeIndex: 18 entries, 0 to 17
Data columns (total 3 columns):
 #   Column    Non-Null Count  Dtype 
---  ------    --------------  ----- 
 0   inv_id    18 non-null     object
 1   ing_id    18 non-null     object
 2   quantity  18 non-null     int64 
dtypes: int64(1), object(2)
memory usage: 564.0+ bytes
```

Figure 1.5.6: Inventory tables

Based on Figure 1.5.6 above, the data schema for the inventory consists of three columns. This data frame consists of string and integer data types. Two columns have the data type of string and integer for one column.

- For staff



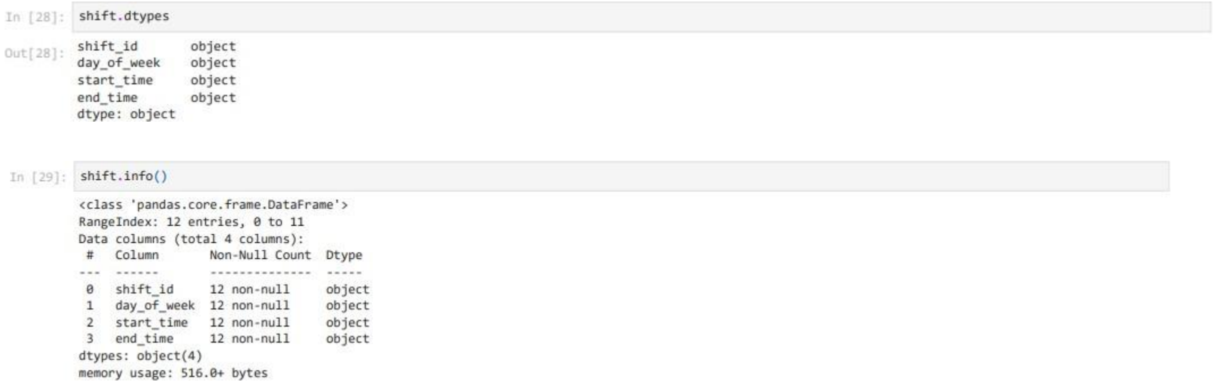
```
In [25]: staff.dtypes

Out[25]: staff_id        object
         first_name      object
         last_name       object
         position        object
         sal_per_hour   float64
         dtype: object
```

```
In [26]: staff.info()

<class 'pandas.core.frame.DataFrame'>
RangeIndex: 4 entries, 0 to 3
Data columns (total 5 columns):
 #   Column        Non-Null Count  Dtype  
---  ------        --------------  -----  
 0   staff_id      4 non-null      object 
 1   first_name    4 non-null      object 
 2   last_name     4 non-null      object 
 3   position      4 non-null      object 
 4   sal_per_hour  4 non-null      float64
dtypes: float64(1), object(4)
memory usage: 292.0+ bytes
```

Figure 1.5.7: Staff tables

Based on Figure 1.5.7 above, the data schema for the staff consists of five columns. This data frame consists of string and float data types. One column has the data type of float meanwhile the rest have the string data types.

- For shift



```
In [28]: shift.dtypes

Out[28]: shift_id       object
         day_of_week    object
         start_time     object
         end_time       object
         dtype: object


In [29]: shift.info()

         <class 'pandas.core.frame.DataFrame'>
         RangeIndex: 12 entries, 0 to 11
         Data columns (total 4 columns):
          #   Column       Non-Null Count  Dtype 
         ---  ------       --------------  ----- 
          0   shift_id     12 non-null     object
          1   day_of_week  12 non-null     object
          2   start_time   12 non-null     object
          3   end_time     12 non-null     object
         dtypes: object(4)
         memory usage: 516.0+ bytes
```

Figure 1.5.8: Shift tables

Based on Figure 1.5.8 above, the data schema for the shift consists of four columns. The data type for this data frame is a string. All the columns have the same data type.

2.0 ARCHITECTURE AND ETL PIPELINE

2.1 Pipeline Structure

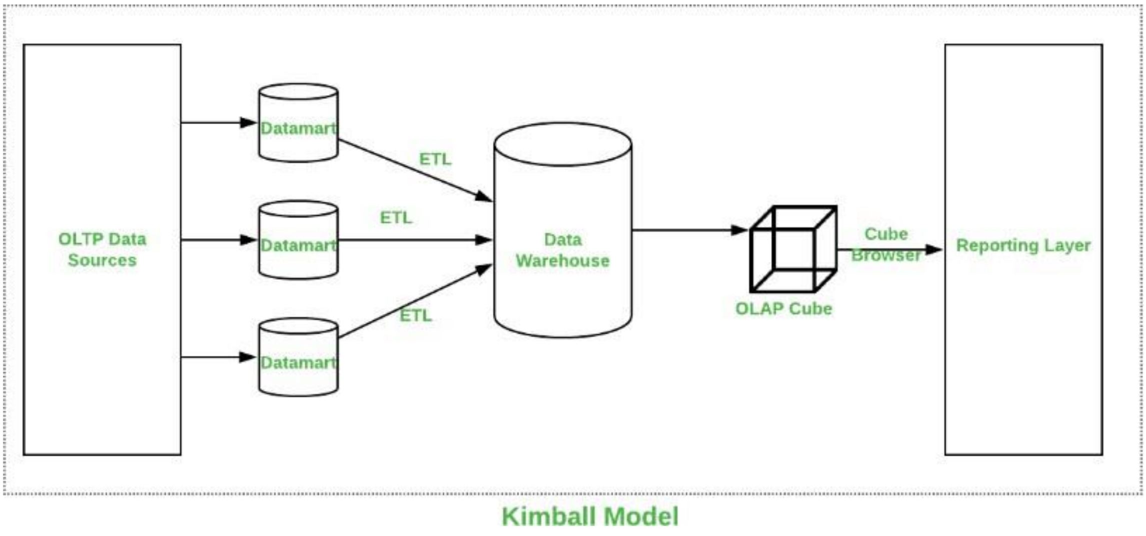


Figure 2.1.1

In this project, we are using Kimball methodology which follows a bottom-up approach. We begin by organising data for specific parts of the coffee shop business and then gradually bring it all together into a data warehouse. We have created tables for different aspects of our coffee shop business such as ingredients, inventory, items, orders, recipes, rota, shifts, and staff details. Each table holds specific information, like ingredients' names and prices, inventory quantities, order details, and staff schedules. The Kimball approach helps us focus on what's most important for our business needs and makes it easier to manage and understand our data. We are taking a step-by-step approach, starting with small parts and growing from there, which allows us to be more flexible and responsive to changes in our business.

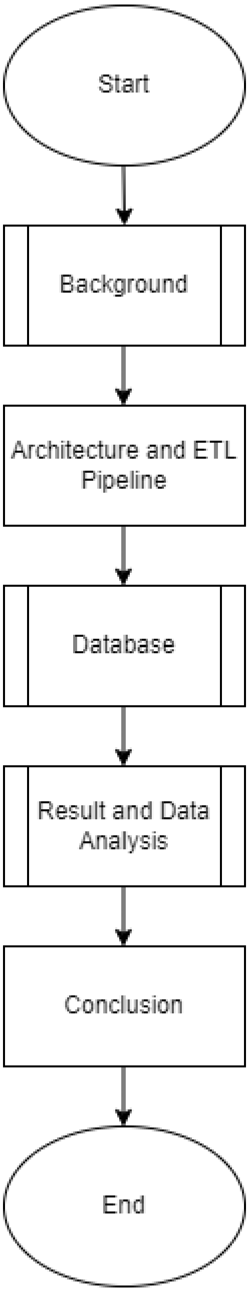


Figure 2.1.2 Flow of the project

Figure 2.1.2 shows the overall procedure of the project, which our project will complete in six stages.

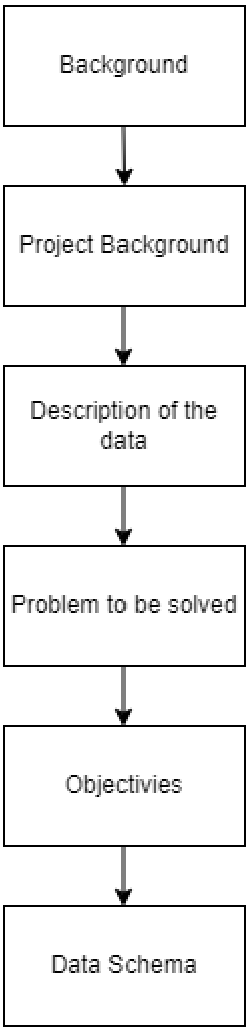


Figure 2.1.3 Process of the background

Figure 2.1.3 shows the background process, which covers the project's background, the data's description, the issue that must be resolved, the project's goals, and the data schema. The information utilised in this study came from Kaggle. Following that, the architecture was developed to ensure that the project would go smoothly and in accordance with the plan.

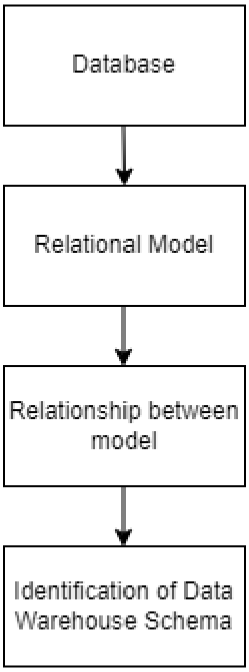

Database
Relational Model
Relationship between model
Identification of Data Warehouse Schema

Figure 2.1.4 Process of Database

Figure 2.1.4 shows the process of the database including relational model, relationship between model and identification of the data warehouse schema. In this project, our group uses Microsoft Power BI to create the relational model. After that, we proceed with the Extract, Transform and Load (ETL) pipeline by using Jupyter Notebook and pgAdmin.

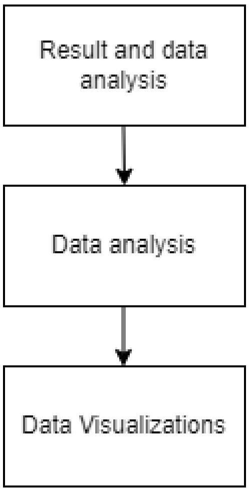


Figure 2.1.5 Process for the result and data analysis

Figure 2.1.5 shows the findings and data analysis, which also involves data visualization.

2.2 ETL Pipeline

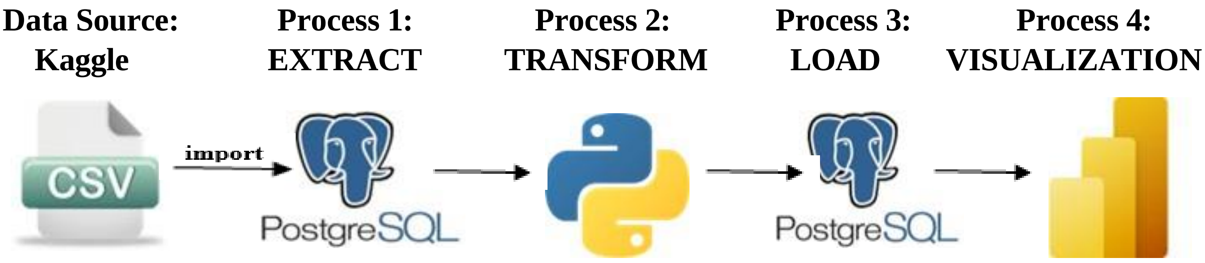


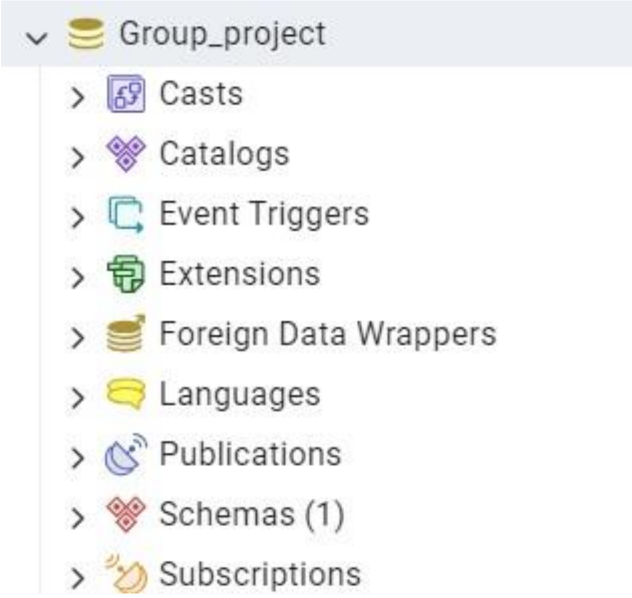
Figure 2.2

In order to build a data warehouse, it is required to run ETL tools which has three tasks: extracting data from different data sources, transforming data, loading transformed data to data warehouse. In this project, we used PostgreSQL to extract raw data from various CSV files,and then transform the data using Python in the Jupyter Notebook by connecting the server of PostgreSQL with the Jupyter Notebook. In the transforming process, all data is being cleaned and confirmed so that the data gained is correct, complete, consistent and unambiguous. The process includes data cleaning, transforming and integration. Next, we load the cleaned data back to PostgreSQL in multidimensional structure and finally visualise the data pattern and relationship using a data visualisation tool like Microsoft PowerBI.

2.3 ETL Process

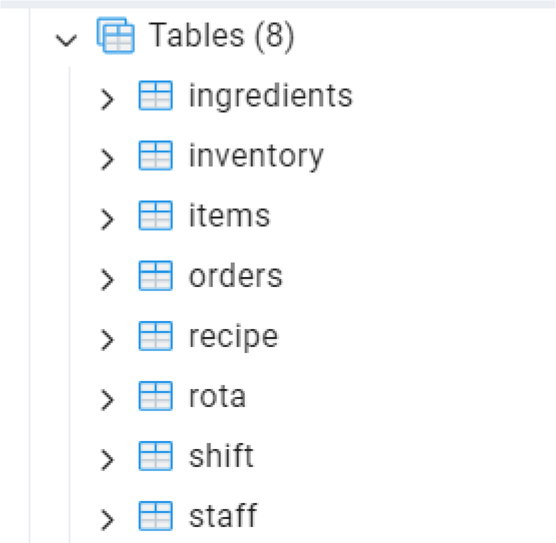
2.3.1 Extract

Firstly, we start our project by finding datasets that contain at least 4 tables. We had found the datasets about coffee shop management using Kaggle. The founded dataset consists of 8 tables which are ingredients, inventory, items, orders, recipe, rota, staff and shift. All of these data are raw data. Before we start the ETL process, we need to store the datasets in a database by using PostgreSQL. We need to create a new database, tables, and import the dataset into each corresponding table.



Group_project
Casts
Catalogs
Event Triggers
Extensions
Foreign Data Wrappers
Languages
Publications
Schemas (1)
Subscriptions

Figure 2.3.1 Database in PostgreSQL

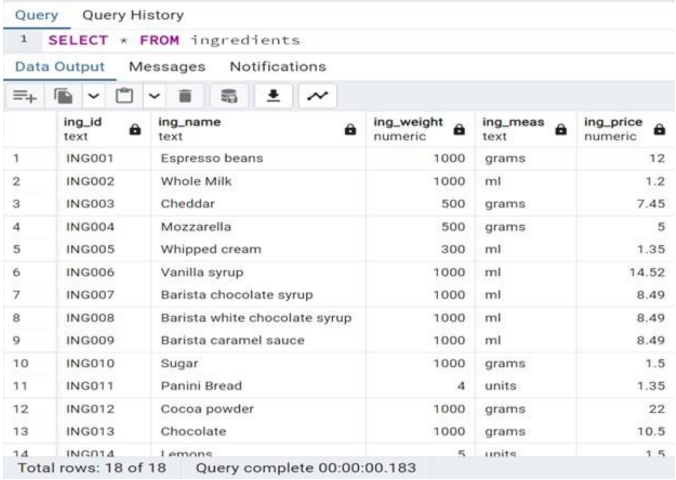


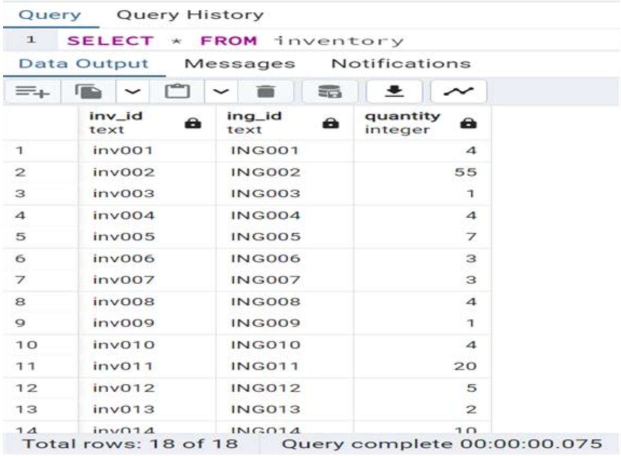
Tables (8)
ingredients
inventory
items
orders
recipe
rota
shift
staff

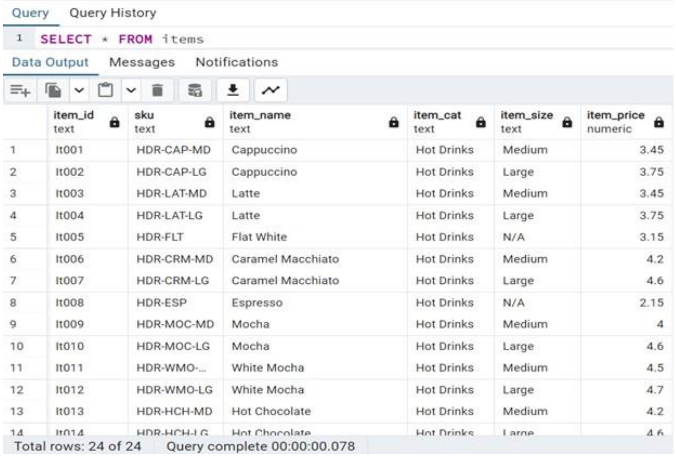
Figure 2.3.2 Tables Created

Figure 2.3.1 and 2.3.2. show that we had created a new database named Group_project in PostgreSQL, which contains 8 tables. Next, we import the dataset manually to each corresponding table. After the raw data has been extracted into PostgreSQL, we need to connect our PostgreSQL server with the Jupyter Notebook to proceed to the next step which is transformation of data.

Run a query (Select * from {table name}) to view the data in the table

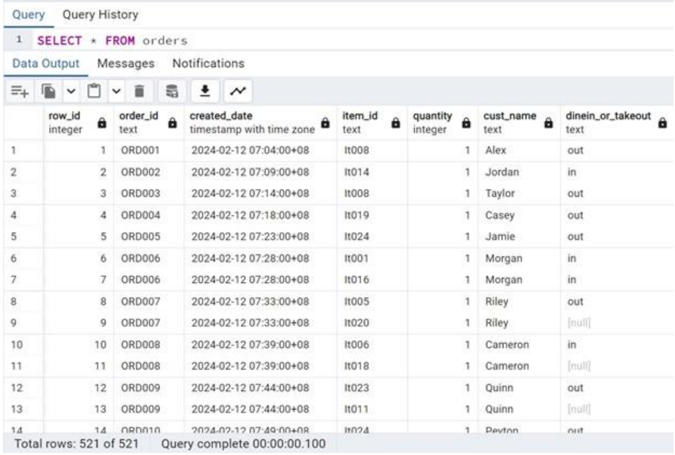
<table>
<tr><th>QUERY</th><th>OUTPUT</th></tr>
<tr><td>SELECT * FROM ingredients</td><td>


1 SELECT * FROM ingredients
<table>
<tr><th></th><th>ing_id
text</th><th>ing_name
text</th><th>ing_weight
numeric</th><th>ing_meas
text</th><th>ing_price
numeric</th></tr>
<tr><td>1</td><td>ING001</td><td>Espresso beans</td><td>1000</td><td>grams</td><td>12</td></tr>
<tr><td>2</td><td>ING002</td><td>Whole Milk</td><td>1000</td><td>ml</td><td>1.2</td></tr>
<tr><td>3</td><td>ING003</td><td>Cheddar</td><td>500</td><td>grams</td><td>7.45</td></tr>
<tr><td>4</td><td>ING004</td><td>Mozzarella</td><td>500</td><td>grams</td><td>5</td></tr>
<tr><td>5</td><td>ING005</td><td>Whipped cream</td><td>300</td><td>ml</td><td>1.35</td></tr>
<tr><td>6</td><td>ING006</td><td>Vanilla syrup</td><td>1000</td><td>ml</td><td>14.52</td></tr>
<tr><td>7</td><td>ING007</td><td>Barista chocolate syrup</td><td>1000</td><td>ml</td><td>8.49</td></tr>
<tr><td>8</td><td>ING008</td><td>Barista white chocolate syrup</td><td>1000</td><td>ml</td><td>8.49</td></tr>
<tr><td>9</td><td>ING009</td><td>Barista caramel sauce</td><td>1000</td><td>ml</td><td>8.49</td></tr>
<tr><td>10</td><td>ING010</td><td>Sugar</td><td>1000</td><td>grams</td><td>1.5</td></tr>
<tr><td>11</td><td>ING011</td><td>Panini Bread</td><td>4</td><td>units</td><td>1.35</td></tr>
<tr><td>12</td><td>ING012</td><td>Cocoa powder</td><td>1000</td><td>grams</td><td>22</td></tr>
<tr><td>13</td><td>ING013</td><td>Chocolate</td><td>1000</td><td>grams</td><td>10.5</td></tr>
<tr><td>14</td><td>ING014</td><td>Lemons</td><td>5</td><td>units</td><td>1.5</td></tr>
</table>
Total rows: 18 of 18 Query complete 00:00:00.183
</td></tr>
<tr><td>SELECT * FROM inventory</td><td>


1 SELECT * FROM inventory
<table>
<tr><th></th><th>inv_id
text</th><th>ing_id
text</th><th>quantity
integer</th></tr>
<tr><td>1</td><td>inv001</td><td>ING001</td><td>4</td></tr>
<tr><td>2</td><td>inv002</td><td>ING002</td><td>55</td></tr>
<tr><td>3</td><td>inv003</td><td>ING003</td><td>1</td></tr>
<tr><td>4</td><td>inv004</td><td>ING004</td><td>4</td></tr>
<tr><td>5</td><td>inv005</td><td>ING005</td><td>7</td></tr>
<tr><td>6</td><td>inv006</td><td>ING006</td><td>3</td></tr>
<tr><td>7</td><td>inv007</td><td>ING007</td><td>3</td></tr>
<tr><td>8</td><td>inv008</td><td>ING008</td><td>4</td></tr>
<tr><td>9</td><td>inv009</td><td>ING009</td><td>1</td></tr>
<tr><td>10</td><td>inv010</td><td>ING010</td><td>4</td></tr>
<tr><td>11</td><td>inv011</td><td>ING011</td><td>20</td></tr>
<tr><td>12</td><td>inv012</td><td>ING012</td><td>5</td></tr>
<tr><td>13</td><td>inv013</td><td>ING013</td><td>2</td></tr>
<tr><td>14</td><td>inv014</td><td>ING014</td><td>10</td></tr>
</table>
Total rows: 18 of 18 Query complete 00:00:00.075
</td></tr>
<tr><td>SELECT * FROM items</td><td>


1 SELECT * FROM items
<table>
<tr><th></th><th>item_id
text</th><th>sku
text</th><th>item_name
text</th><th>item_cat
text</th><th>item_size
text</th><th>item_price
numeric</th></tr>
<tr><td>1</td><td>It001</td><td>HDR-CAP-MD</td><td>Cappuccino</td><td>Hot Drinks</td><td>Medium</td><td>3.45</td></tr>
<tr><td>2</td><td>It002</td><td>HDR-CAP-LG</td><td>Cappuccino</td><td>Hot Drinks</td><td>Large</td><td>3.75</td></tr>
<tr><td>3</td><td>It003</td><td>HDR-LAT-MD</td><td>Latte</td><td>Hot Drinks</td><td>Medium</td><td>3.45</td></tr>
<tr><td>4</td><td>It004</td><td>HDR-LAT-LG</td><td>Latte</td><td>Hot Drinks</td><td>Large</td><td>3.75</td></tr>
<tr><td>5</td><td>It005</td><td>HDR-FLT</td><td>Flat White</td><td>Hot Drinks</td><td>N/A</td><td>3.15</td></tr>
<tr><td>6</td><td>It006</td><td>HDR-CRM-MD</td><td>Caramel Macchiato</td><td>Hot Drinks</td><td>Medium</td><td>4.2</td></tr>
<tr><td>7</td><td>It007</td><td>HDR-CRM-LG</td><td>Caramel Macchiato</td><td>Hot Drinks</td><td>Large</td><td>4.6</td></tr>
<tr><td>8</td><td>It008</td><td>HDR-ESP</td><td>Espresso</td><td>Hot Drinks</td><td>N/A</td><td>2.15</td></tr>
<tr><td>9</td><td>It009</td><td>HDR-MOC-MD</td><td>Mocha</td><td>Hot Drinks</td><td>Medium</td><td>4</td></tr>
<tr><td>10</td><td>It010</td><td>HDR-MOC-LG</td><td>Mocha</td><td>Hot Drinks</td><td>Large</td><td>4.6</td></tr>
<tr><td>11</td><td>It011</td><td>HDR-WMO-...</td><td>White Mocha</td><td>Hot Drinks</td><td>Medium</td><td>4.5</td></tr>
<tr><td>12</td><td>It012</td><td>HDR-WMO-LG</td><td>White Mocha</td><td>Hot Drinks</td><td>Large</td><td>4.7</td></tr>
<tr><td>13</td><td>It013</td><td>HDR-HCH-MD</td><td>Hot Chocolate</td><td>Hot Drinks</td><td>Medium</td><td>4.2</td></tr>
<tr><td>14</td><td>It014</td><td>HDR-HCH-LG</td><td>Hot Chocolate</td><td>Hot Drinks</td><td>Large</td><td>4.6</td></tr>
</table>
Total rows: 24 of 24 Query complete 00:00:00.078
</td></tr>
</table>

SELECT * FROM orders

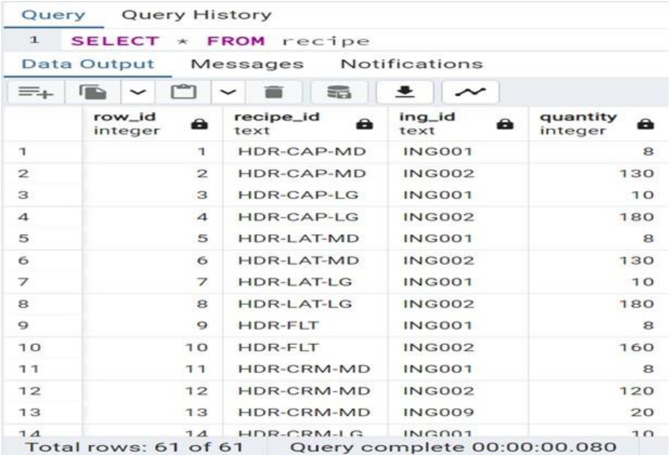


```
SELECT * FROM orders
```

	row_id integer	order_id text	created_date timestamp with time zone	item_id text	quantity integer	cust_name text	dinein_or_takeout text
1	1	ORD001	2024-02-12 07:04:00+08	It008	1	Alex	out
2	2	ORD002	2024-02-12 07:09:00+08	It014	1	Jordan	in
3	3	ORD003	2024-02-12 07:14:00+08	It008	1	Taylor	out
4	4	ORD004	2024-02-12 07:18:00+08	It019	1	Casey	out
5	5	ORD005	2024-02-12 07:23:00+08	It024	1	Jamie	out
6	6	ORD006	2024-02-12 07:28:00+08	It001	1	Morgan	in
7	7	ORD006	2024-02-12 07:28:00+08	It016	1	Morgan	in
8	8	ORD007	2024-02-12 07:33:00+08	It005	1	Riley	out
9	9	ORD007	2024-02-12 07:33:00+08	It020	1	Riley	[null]
10	10	ORD008	2024-02-12 07:39:00+08	It006	1	Cameron	in
11	11	ORD008	2024-02-12 07:39:00+08	It018	1	Cameron	[null]
12	12	ORD009	2024-02-12 07:44:00+08	It023	1	Quinn	out
13	13	ORD009	2024-02-12 07:44:00+08	It011	1	Quinn	[null]
14	14	ORD010	2024-02-12 07:49:00+08	It024	1	Peyton	out

Total rows: 521 of 521   Query complete 00:00:00.100

SELECT * FROM recipe

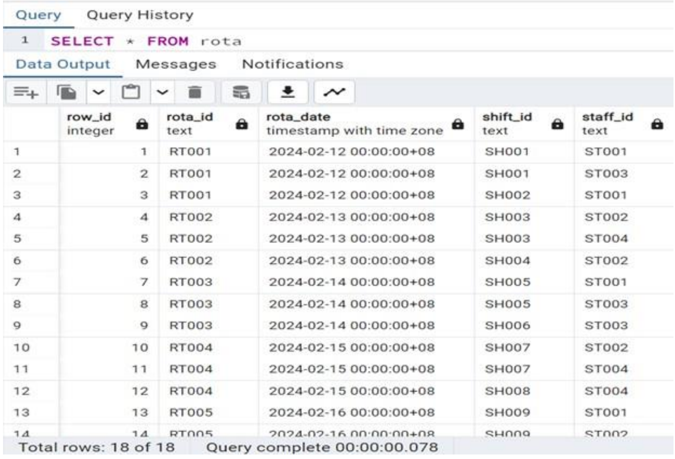


```
SELECT * FROM recipe
```

	row_id integer	recipe_id text	ing_id text	quantity integer
1	1	HDR-CAP-MD	ING001	8
2	2	HDR-CAP-MD	ING002	130
3	3	HDR-CAP-LG	ING001	10
4	4	HDR-CAP-LG	ING002	180
5	5	HDR-LAT-MD	ING001	8
6	6	HDR-LAT-MD	ING002	130
7	7	HDR-LAT-LG	ING001	10
8	8	HDR-LAT-LG	ING002	180
9	9	HDR-FLT	ING001	8
10	10	HDR-FLT	ING002	160
11	11	HDR-CRM-MD	ING001	8
12	12	HDR-CRM-MD	ING002	120
13	13	HDR-CRM-MD	ING009	20
14	14	HDR-CRM-LG	ING001	10

Total rows: 61 of 61   Query complete 00:00:00.080

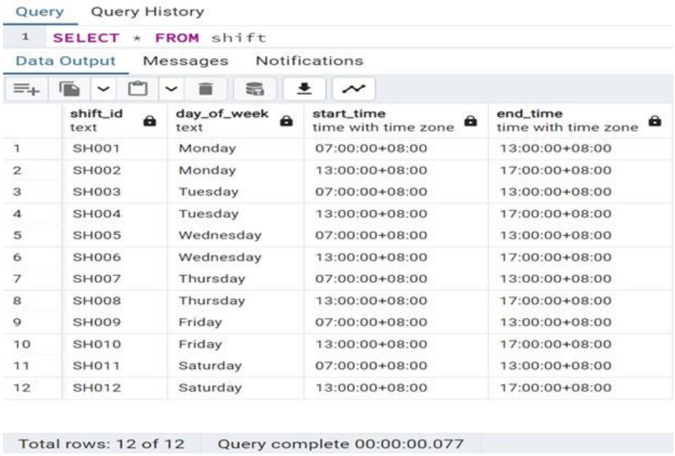
SELECT * FROM rota



```
SELECT * FROM rota
```

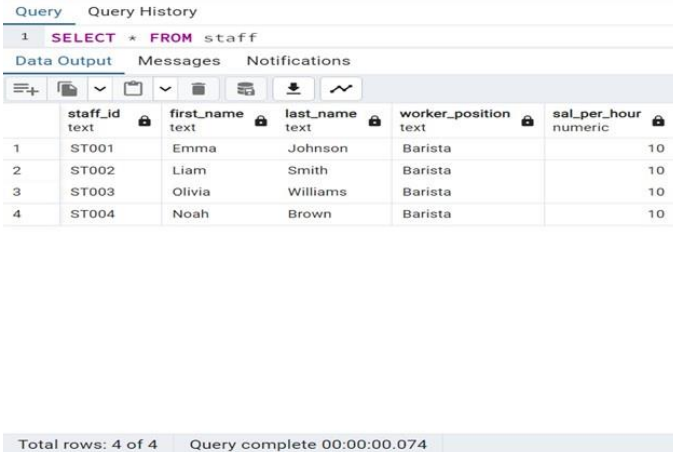
	row_id integer	rota_id text	rota_date timestamp with time zone	shift_id text	staff_id text
1	1	RT001	2024-02-12 00:00:00+08	SH001	ST001
2	2	RT001	2024-02-12 00:00:00+08	SH001	ST003
3	3	RT001	2024-02-12 00:00:00+08	SH002	ST001
4	4	RT002	2024-02-13 00:00:00+08	SH003	ST002
5	5	RT002	2024-02-13 00:00:00+08	SH003	ST004
6	6	RT002	2024-02-13 00:00:00+08	SH004	ST002
7	7	RT003	2024-02-14 00:00:00+08	SH005	ST001
8	8	RT003	2024-02-14 00:00:00+08	SH005	ST003
9	9	RT003	2024-02-14 00:00:00+08	SH006	ST003
10	10	RT004	2024-02-15 00:00:00+08	SH007	ST002
11	11	RT004	2024-02-15 00:00:00+08	SH007	ST004
12	12	RT004	2024-02-15 00:00:00+08	SH008	ST004
13	13	RT005	2024-02-16 00:00:00+08	SH009	ST001
14	14	RT005	2024-02-16 00:00:00+08	SH009	ST002

Total rows: 18 of 18   Query complete 00:00:00.078

<table>
<tr>
<td>SELECT * FROM shift</td>
<td>


1 SELECT * FROM shift

<table>
<tr><th></th><th>shift_id
text</th><th>day_of_week
text</th><th>start_time
time with time zone</th><th>end_time
time with time zone</th></tr>
<tr><td>1</td><td>SH001</td><td>Monday</td><td>07:00:00+08:00</td><td>13:00:00+08:00</td></tr>
<tr><td>2</td><td>SH002</td><td>Monday</td><td>13:00:00+08:00</td><td>17:00:00+08:00</td></tr>
<tr><td>3</td><td>SH003</td><td>Tuesday</td><td>07:00:00+08:00</td><td>13:00:00+08:00</td></tr>
<tr><td>4</td><td>SH004</td><td>Tuesday</td><td>13:00:00+08:00</td><td>17:00:00+08:00</td></tr>
<tr><td>5</td><td>SH005</td><td>Wednesday</td><td>07:00:00+08:00</td><td>13:00:00+08:00</td></tr>
<tr><td>6</td><td>SH006</td><td>Wednesday</td><td>13:00:00+08:00</td><td>17:00:00+08:00</td></tr>
<tr><td>7</td><td>SH007</td><td>Thursday</td><td>07:00:00+08:00</td><td>13:00:00+08:00</td></tr>
<tr><td>8</td><td>SH008</td><td>Thursday</td><td>13:00:00+08:00</td><td>17:00:00+08:00</td></tr>
<tr><td>9</td><td>SH009</td><td>Friday</td><td>07:00:00+08:00</td><td>13:00:00+08:00</td></tr>
<tr><td>10</td><td>SH010</td><td>Friday</td><td>13:00:00+08:00</td><td>17:00:00+08:00</td></tr>
<tr><td>11</td><td>SH011</td><td>Saturday</td><td>07:00:00+08:00</td><td>13:00:00+08:00</td></tr>
<tr><td>12</td><td>SH012</td><td>Saturday</td><td>13:00:00+08:00</td><td>17:00:00+08:00</td></tr>
</table>

Total rows: 12 of 12 Query complete 00:00:00.077
</td>
</tr>
<tr>
<td>SELECT * FROM staff</td>
<td>


1 SELECT * FROM staff

<table>
<tr><th></th><th>staff_id
text</th><th>first_name
text</th><th>last_name
text</th><th>worker_position
text</th><th>sal_per_hour
numeric</th></tr>
<tr><td>1</td><td>ST001</td><td>Emma</td><td>Johnson</td><td>Barista</td><td>10</td></tr>
<tr><td>2</td><td>ST002</td><td>Liam</td><td>Smith</td><td>Barista</td><td>10</td></tr>
<tr><td>3</td><td>ST003</td><td>Olivia</td><td>Williams</td><td>Barista</td><td>10</td></tr>
<tr><td>4</td><td>ST004</td><td>Noah</td><td>Brown</td><td>Barista</td><td>10</td></tr>
</table>

Total rows: 4 of 4 Query complete 00:00:00.074
</td>
</tr>
</table>

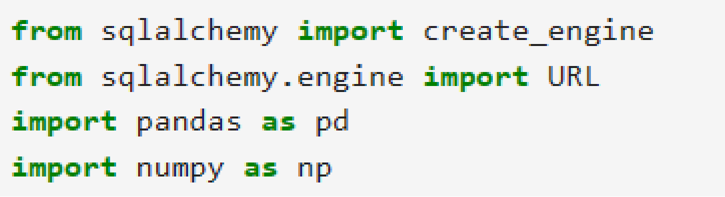
Before transforming data to Jupyter Notebook, we need to create a new database named “AdventureWorks” and User with password “demopass”. It is used to connect PgAdmin to Jupyter Notebook.

2.3.2 Transforms

After the raw data has been extracted into pgAdmin, we need to connect our pgAdmin with the Jupyter Notebook to proceed to the next step which transforms the data.
Before starting the process, we are required to install some packages in Jupyter NoteBook:

- !pip install psycopg2
- !pip install SQLalchemy

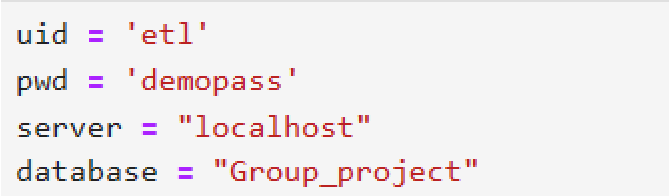
After installing the packages, we need to call the create_engine function and URLfor connecting to the database. Besides that, we also import some necessary libraries that are going to be used for the data cleaning process.



```
from sqlalchemy import create_engine
from sqlalchemy.engine import URL
import pandas as pd
import numpy as np
```

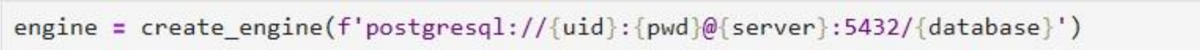
Figure 2.3.3 Call function and import libraries

Next, we create a connection to postgreSQL database using following command:



```
uid = 'etl'
pwd = 'demopass'
server = "localhost"
database = "Group_project"
```

Figure 2.3.4 Show the information of postgreSQL server and database

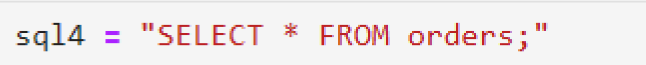


```
engine = create_engine(f'postgresql://{uid}:{pwd}@{server}:5432/{database}')
```

Figure 2.3.5 Show the connection to the PostgreSQL database

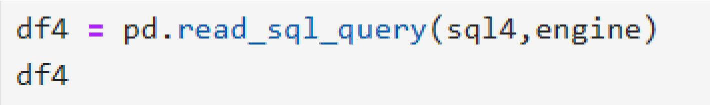
After establishing a connection to PostgreSQL, the next step is to clean the data to ensure it is ready for analysis. Data cleaning typically involves removing inconsistencies, unused columns, duplicates value, handling missing values, and transforming data into a more usable format.

Firstly, we execute a SQL query to retrieve the data and then read the data using Pandas. This data will be used for further cleaning processes.



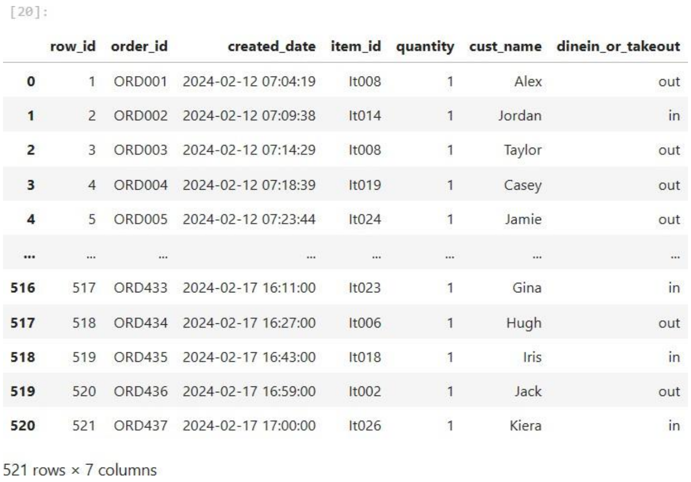
```
sql4 = "SELECT * FROM orders;"
```

Figure 2.3.6 Select all data from table "orders"



```
df4 = pd.read_sql_query(sql4,engine)
df4
```

Figure 2.3.7 Read data using Pandas



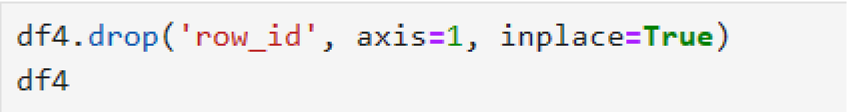
[20]:

	row_id	order_id	created_date	item_id	quantity	cust_name	dinein_or_takeout
0	1	ORD001	2024-02-12 07:04:19	It008	1	Alex	out
1	2	ORD002	2024-02-12 07:09:38	It014	1	Jordan	in
2	3	ORD003	2024-02-12 07:14:29	It008	1	Taylor	out
3	4	ORD004	2024-02-12 07:18:39	It019	1	Casey	out
4	5	ORD005	2024-02-12 07:23:44	It024	1	Jamie	out
...	...	...	...	...	...	...	...
516	517	ORD433	2024-02-17 16:11:00	It023	1	Gina	in
517	518	ORD434	2024-02-17 16:27:00	It006	1	Hugh	out
518	519	ORD435	2024-02-17 16:43:00	It018	1	Iris	in
519	520	ORD436	2024-02-17 16:59:00	It002	1	Jack	out
520	521	ORD437	2024-02-17 17:00:00	It026	1	Kiera	in

521 rows × 7 columns

Figure 2.3.8 Output from table "orders"

After the data successfully stored into data frame by using pandas library, then we can start the cleaning process by drop unused column:

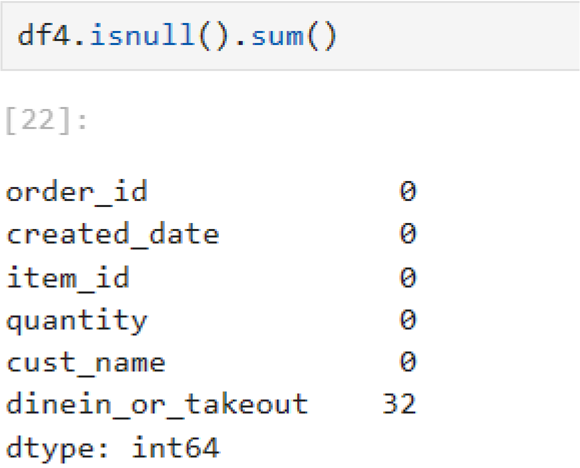


```
df4.drop('row_id', axis=1, inplace=True)
df4
```

Figure 2.3.9 Drop column "row_id"

In this dataset, we had removed the unused column which is 'row_id'. Row_id is the row no and it does not have any meaningful insight. Therefore, it is important to remove the unused column because it can lead to faster queries and improve the performance.

After dropping the unused column, we continue our cleaning process by checking null values.

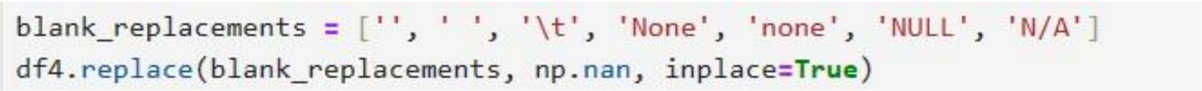


```
df4.isnull().sum()
```

```
[22]:

order_id              0
created_date          0
item_id               0
quantity              0
cust_name             0
dinein_or_takeout    32
dtype: int64
```

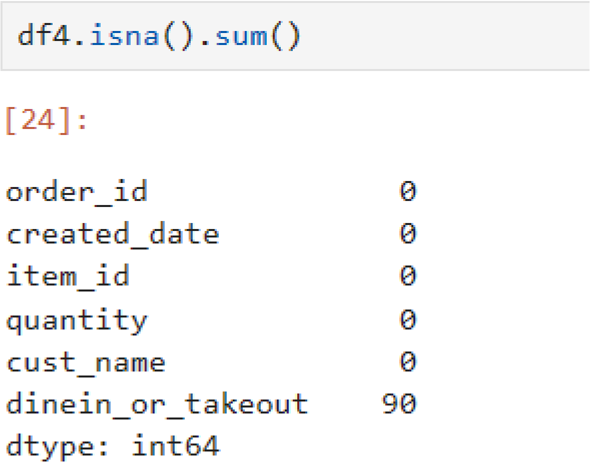
Figure 2.3.10 Checking null values using isnull().sum()

We discovered that this dataset actually contains 90 null values but only 32 of them were detected. Therefore, we replace all the blank data with ‘NaN’ using numpy library to make it easy to be detect.



```
blank_replacements = ['', ' ', '\t', 'None', 'none', 'NULL', 'N/A']
df4.replace(blank_replacements, np.nan, inplace=True)
```

Figure 2.3.11 Replace blank data with ‘NaN’

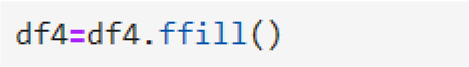


```
df4.isna().sum()
```

```
[24]:

order_id               0
created_date           0
item_id                0
quantity               0
cust_name              0
dinein_or_takeout     90
dtype: int64
```

Figure 2.3.12 Checking null values using isna().sum()

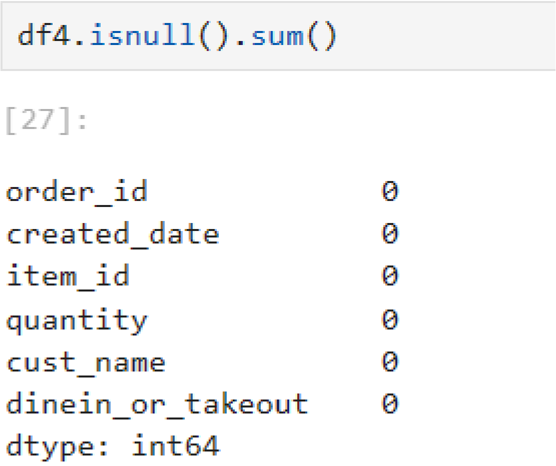
After replacing the blank data with ‘NaN’, we check again for the total number of null values in the dataset. We got the correct number of null values which is 90. Since the dataset contains the null values, the next step of the cleaning process is to handle the missing values. The following command is forwardfill method which also known as last observation carried forward by carrying forward the last observed value to fill in the gaps:



```
df4=df4.ffill()
```

Figure 2.3.13 Handling null values with forwardfill method

In this dataset, we handle the missing data using an imputation method which is forwardfill. Since our dataset is not so large, deleting the rows that contain null values will cause a significant loss of data, potentially affecting the accuracy of analysis. In addition, the data type for missing values is text, therefore we are not able to replace the null values with mean and median. After filling the missing values, we recheck again the total number of missing values in the dataset using following command:

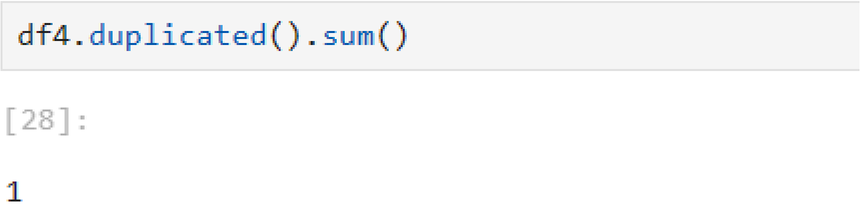


```
df4.isnull().sum()
```

```
[27]:

order_id            0
created_date        0
item_id             0
quantity            0
cust_name           0
dinein_or_takeout   0
dtype: int64
```

Figure 2.3.14 Check total number of NULL values

After checking for null values, we continue our cleaning process with a check for duplicate values. It is important to check duplicate data to avoid data redundancy.We check the number of duplicate value using following command:

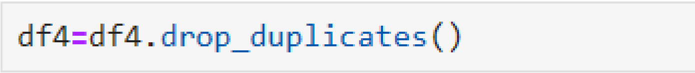


```
df4.duplicated().sum()
```

```
[28]:

1
```

Figure 2.3.15 Check number of duplicate values

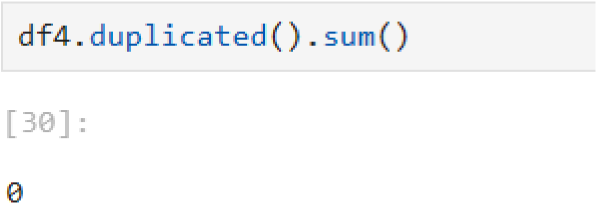
Removing the duplicate values using following command if any duplicates values found in the dataset:



```
df4=df4.drop_duplicates()
```

Figure 2.3.16 Drop the duplicates data

Recheck for duplicate data in dataset using following command:

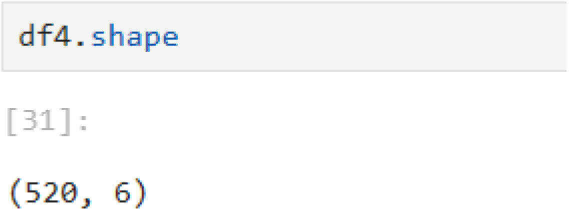


```
df4.duplicated().sum()
```

```
[30]:

0
```

Figure 2.3.17 Check for duplicate data

After all cleaning process is done, we check the shape of dataset to know about the size and structure of the data.



```
df4.shape
```

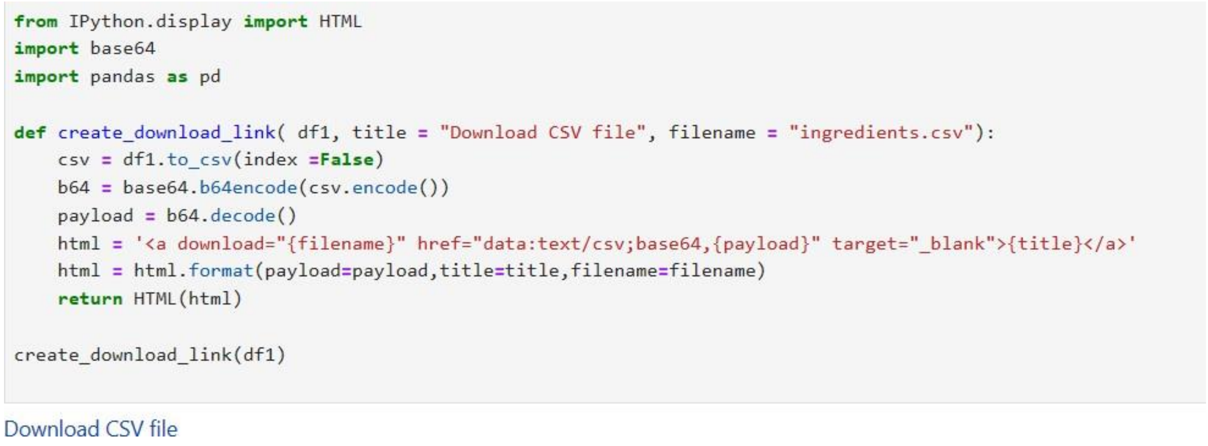
```
[31]:

(520, 6)
```

Figure 2.3.18 Determine shape of dataset

Repeat all cleaning processes for all the data frames.

After the cleaning process is done for all dataframe, we need to download all datasets in csv files to be loaded into PostgreSQL for further analysis.

Create a link to download the csv file using following command:



```
from IPython.display import HTML
import base64
import pandas as pd

def create_download_link( df1, title = "Download CSV file", filename = "ingredients.csv"):
    csv = df1.to_csv(index =False)
    b64 = base64.b64encode(csv.encode())
    payload = b64.decode()
    html = '<a download="{filename}" href="data:text/csv;base64,{payload}" target="_blank">{title}</a>'
    html = html.format(payload=payload,title=title,filename=filename)
    return HTML(html)

create_download_link(df1)
```

Download CSV file

Figure 2.3.19 Download CSV file

Repeat the command for other dataframes to generate new csv files with the cleaned data.

2.3.3 Load

After we cleaned our data, it is time to load our data into PostgreSQL.

Firstly, we create a new database and tables in PostgreSQL.

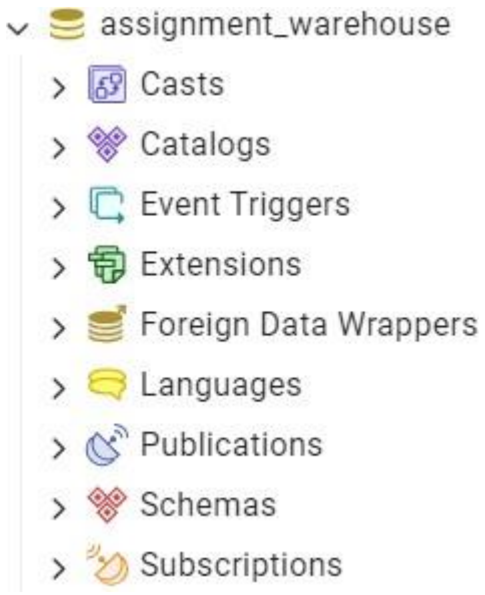


Figure 2.3.20 New database named ‘assignment_warehouse’

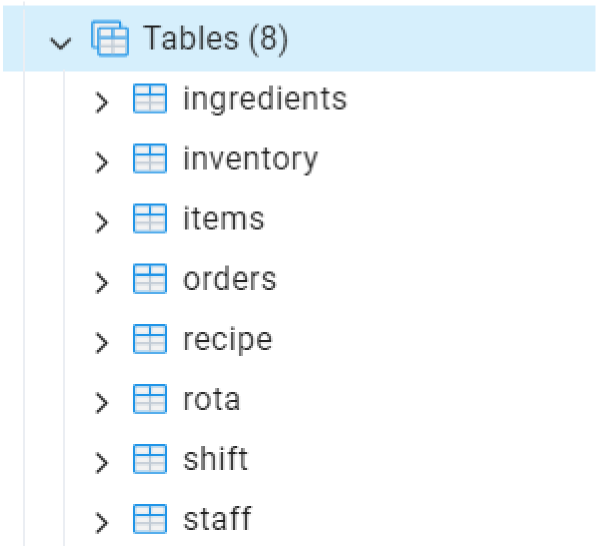
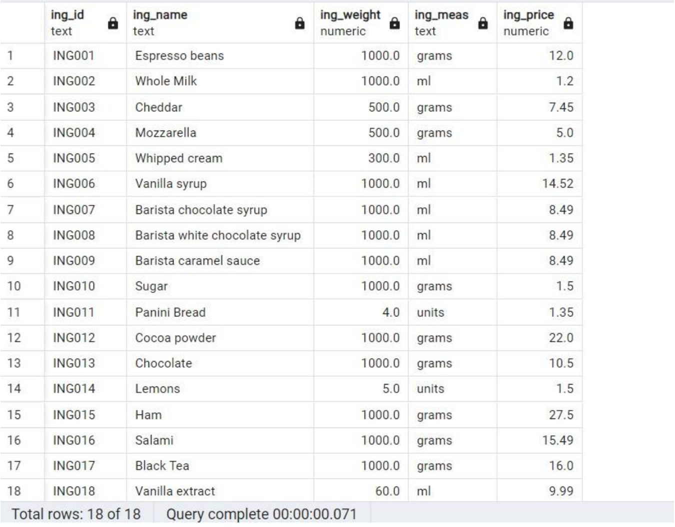
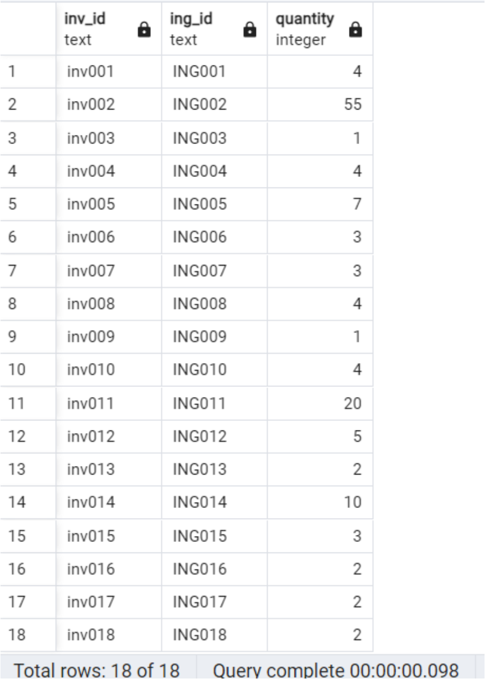


Figure 2.3.21 Tables created under ‘assignment_warehouse’ database

After creating a new database and tables, we manually import the cleaned dataset from the CSV files into the corresponding tables.

Run a query (Select * from {table name}) to view the data in the table

QUERY	OUTPUT
SELECT * FROM ingredients;	
SELECT * FROM inventory;	

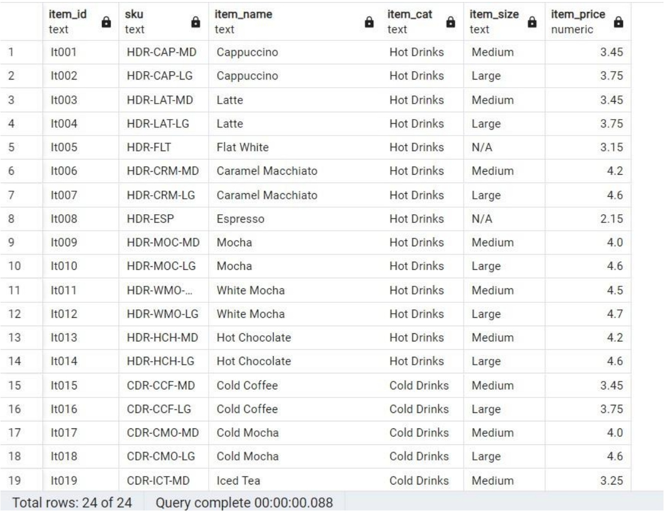
	ing_id text	ing_name text	ing_weight numeric	ing_meas text	ing_price numeric
1	ING001	Espresso beans	1000.0	grams	12.0
2	ING002	Whole Milk	1000.0	ml	1.2
3	ING003	Cheddar	500.0	grams	7.45
4	ING004	Mozzarella	500.0	grams	5.0
5	ING005	Whipped cream	300.0	ml	1.35
6	ING006	Vanilla syrup	1000.0	ml	14.52
7	ING007	Barista chocolate syrup	1000.0	ml	8.49
8	ING008	Barista white chocolate syrup	1000.0	ml	8.49
9	ING009	Barista caramel sauce	1000.0	ml	8.49
10	ING010	Sugar	1000.0	grams	1.5
11	ING011	Panini Bread	4.0	units	1.35
12	ING012	Cocoa powder	1000.0	grams	22.0
13	ING013	Chocolate	1000.0	grams	10.5
14	ING014	Lemons	5.0	units	1.5
15	ING015	Ham	1000.0	grams	27.5
16	ING016	Salami	1000.0	grams	15.49
17	ING017	Black Tea	1000.0	grams	16.0
18	ING018	Vanilla extract	60.0	ml	9.99

Total rows: 18 of 18   Query complete 00:00:00.071

	inv_id text	ing_id text	quantity integer
1	inv001	ING001	4
2	inv002	ING002	55
3	inv003	ING003	1
4	inv004	ING004	4
5	inv005	ING005	7
6	inv006	ING006	3
7	inv007	ING007	3
8	inv008	ING008	4
9	inv009	ING009	1
10	inv010	ING010	4
11	inv011	ING011	20
12	inv012	ING012	5
13	inv013	ING013	2
14	inv014	ING014	10
15	inv015	ING015	3
16	inv016	ING016	2
17	inv017	ING017	2
18	inv018	ING018	2

Total rows: 18 of 18   Query complete 00:00:00.098

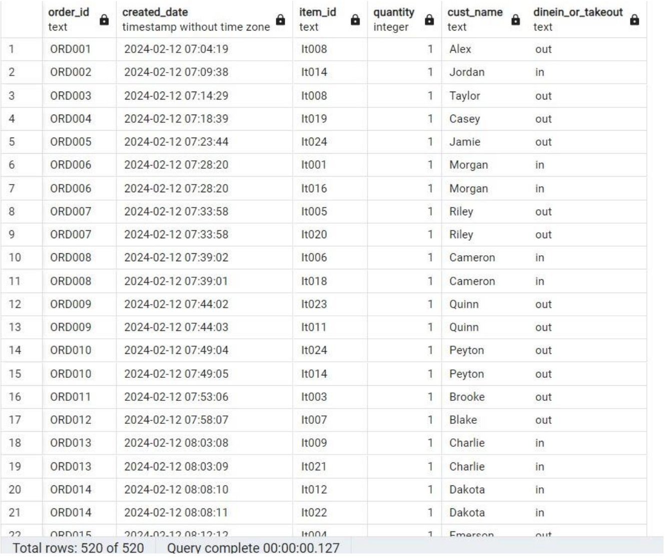
SELECT * FROM items;



	item_id text	sku text	item_name text	item_cat text	item_size text	item_price numeric
1	It001	HDR-CAP-MD	Cappuccino	Hot Drinks	Medium	3.45
2	It002	HDR-CAP-LG	Cappuccino	Hot Drinks	Large	3.75
3	It003	HDR-LAT-MD	Latte	Hot Drinks	Medium	3.45
4	It004	HDR-LAT-LG	Latte	Hot Drinks	Large	3.75
5	It005	HDR-FLT	Flat White	Hot Drinks	N/A	3.15
6	It006	HDR-CRM-MD	Caramel Macchiato	Hot Drinks	Medium	4.2
7	It007	HDR-CRM-LG	Caramel Macchiato	Hot Drinks	Large	4.6
8	It008	HDR-ESP	Espresso	Hot Drinks	N/A	2.15
9	It009	HDR-MOC-MD	Mocha	Hot Drinks	Medium	4.0
10	It010	HDR-MOC-LG	Mocha	Hot Drinks	Large	4.6
11	It011	HDR-WMO-...	White Mocha	Hot Drinks	Medium	4.5
12	It012	HDR-WMO-LG	White Mocha	Hot Drinks	Large	4.7
13	It013	HDR-HCH-MD	Hot Chocolate	Hot Drinks	Medium	4.2
14	It014	HDR-HCH-LG	Hot Chocolate	Hot Drinks	Large	4.6
15	It015	CDR-CCF-MD	Cold Coffee	Cold Drinks	Medium	3.45
16	It016	CDR-CCF-LG	Cold Coffee	Cold Drinks	Large	3.75
17	It017	CDR-CMO-MD	Cold Mocha	Cold Drinks	Medium	4.0
18	It018	CDR-CMO-LG	Cold Mocha	Cold Drinks	Large	4.6
19	It019	CDR-ICT-MD	Iced Tea	Cold Drinks	Medium	3.25

Total rows: 24 of 24 Query complete 00:00:00.088

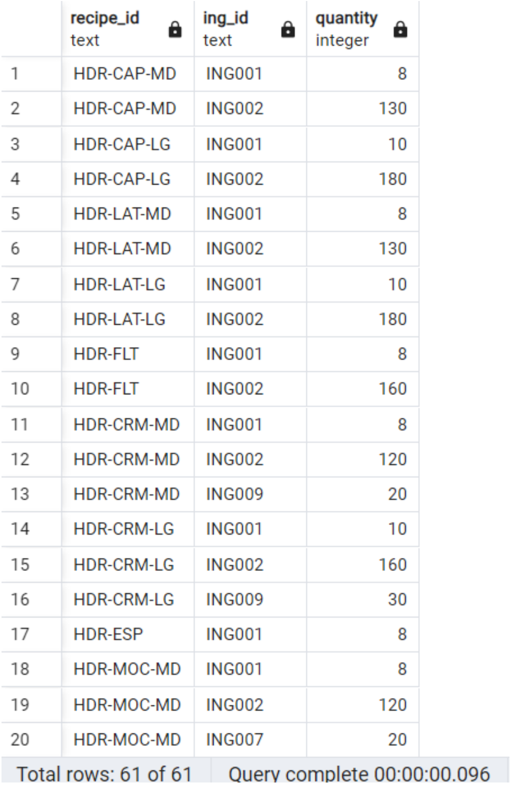
SELECT * FROM orders;



	order_id text	created_date timestamp without time zone	item_id text	quantity integer	cust_name text	dinein_or_takeout text
1	ORD001	2024-02-12 07:04:19	It008	1	Alex	out
2	ORD002	2024-02-12 07:09:38	It014	1	Jordan	in
3	ORD003	2024-02-12 07:14:29	It008	1	Taylor	out
4	ORD004	2024-02-12 07:18:39	It019	1	Casey	out
5	ORD005	2024-02-12 07:23:44	It024	1	Jamie	out
6	ORD006	2024-02-12 07:28:20	It001	1	Morgan	in
7	ORD006	2024-02-12 07:28:20	It016	1	Morgan	in
8	ORD007	2024-02-12 07:33:58	It005	1	Riley	out
9	ORD007	2024-02-12 07:33:58	It020	1	Riley	out
10	ORD008	2024-02-12 07:39:02	It006	1	Cameron	in
11	ORD008	2024-02-12 07:39:01	It018	1	Cameron	in
12	ORD009	2024-02-12 07:44:02	It023	1	Quinn	out
13	ORD009	2024-02-12 07:44:03	It011	1	Quinn	out
14	ORD010	2024-02-12 07:49:04	It024	1	Peyton	out
15	ORD010	2024-02-12 07:49:05	It014	1	Peyton	out
16	ORD011	2024-02-12 07:53:06	It003	1	Brooke	out
17	ORD012	2024-02-12 07:58:07	It007	1	Blake	out
18	ORD013	2024-02-12 08:03:08	It009	1	Charlie	in
19	ORD013	2024-02-12 08:03:09	It021	1	Charlie	in
20	ORD014	2024-02-12 08:08:10	It012	1	Dakota	in
21	ORD014	2024-02-12 08:08:11	It022	1	Dakota	in
22	ORD015	2024-02-12 08:12:12	It004	1	Emerson	out

Total rows: 520 of 520 Query complete 00:00:00.127

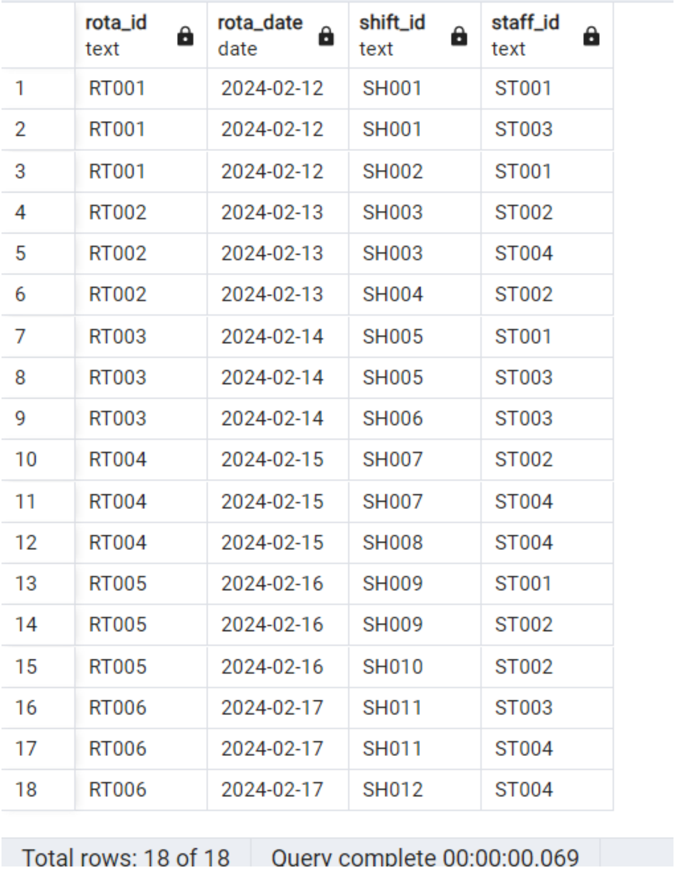
SELECT * FROM recipe;



	recipe_id text	ing_id text	quantity integer
1	HDR-CAP-MD	ING001	8
2	HDR-CAP-MD	ING002	130
3	HDR-CAP-LG	ING001	10
4	HDR-CAP-LG	ING002	180
5	HDR-LAT-MD	ING001	8
6	HDR-LAT-MD	ING002	130
7	HDR-LAT-LG	ING001	10
8	HDR-LAT-LG	ING002	180
9	HDR-FLT	ING001	8
10	HDR-FLT	ING002	160
11	HDR-CRM-MD	ING001	8
12	HDR-CRM-MD	ING002	120
13	HDR-CRM-MD	ING009	20
14	HDR-CRM-LG	ING001	10
15	HDR-CRM-LG	ING002	160
16	HDR-CRM-LG	ING009	30
17	HDR-ESP	ING001	8
18	HDR-MOC-MD	ING001	8
19	HDR-MOC-MD	ING002	120
20	HDR-MOC-MD	ING007	20

Total rows: 61 of 61 Query complete 00:00:00.096

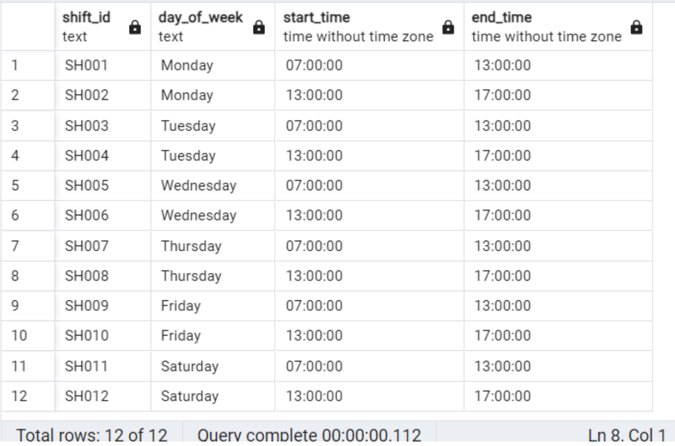
SELECT * FROM rota;



	rota_id text	rota_date date	shift_id text	staff_id text
1	RT001	2024-02-12	SH001	ST001
2	RT001	2024-02-12	SH001	ST003
3	RT001	2024-02-12	SH002	ST001
4	RT002	2024-02-13	SH003	ST002
5	RT002	2024-02-13	SH003	ST004
6	RT002	2024-02-13	SH004	ST002
7	RT003	2024-02-14	SH005	ST001
8	RT003	2024-02-14	SH005	ST003
9	RT003	2024-02-14	SH006	ST003
10	RT004	2024-02-15	SH007	ST002
11	RT004	2024-02-15	SH007	ST004
12	RT004	2024-02-15	SH008	ST004
13	RT005	2024-02-16	SH009	ST001
14	RT005	2024-02-16	SH009	ST002
15	RT005	2024-02-16	SH010	ST002
16	RT006	2024-02-17	SH011	ST003
17	RT006	2024-02-17	SH011	ST004
18	RT006	2024-02-17	SH012	ST004

Total rows: 18 of 18 Query complete 00:00:00.069

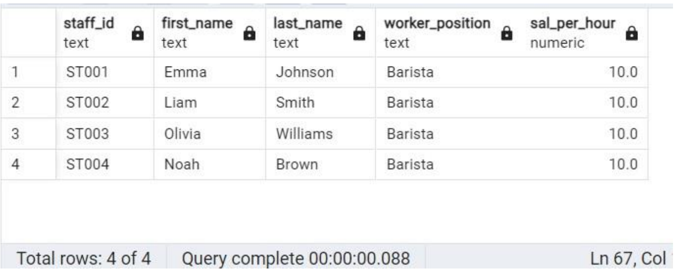
```
SELECT * FROM shift;
```



	shift_id text	day_of_week text	start_time time without time zone	end_time time without time zone
1	SH001	Monday	07:00:00	13:00:00
2	SH002	Monday	13:00:00	17:00:00
3	SH003	Tuesday	07:00:00	13:00:00
4	SH004	Tuesday	13:00:00	17:00:00
5	SH005	Wednesday	07:00:00	13:00:00
6	SH006	Wednesday	13:00:00	17:00:00
7	SH007	Thursday	07:00:00	13:00:00
8	SH008	Thursday	13:00:00	17:00:00
9	SH009	Friday	07:00:00	13:00:00
10	SH010	Friday	13:00:00	17:00:00
11	SH011	Saturday	07:00:00	13:00:00
12	SH012	Saturday	13:00:00	17:00:00

```
Total rows: 12 of 12    Query complete 00:00:00.112    Ln 8, Col 1
```

```
SELECT * FROM staff;
```



	staff_id text	first_name text	last_name text	worker_position text	sal_per_hour numeric
1	ST001	Emma	Johnson	Barista	10.0
2	ST002	Liam	Smith	Barista	10.0
3	ST003	Olivia	Williams	Barista	10.0
4	ST004	Noah	Brown	Barista	10.0

```
Total rows: 4 of 4    Query complete 00:00:00.088    Ln 67, Col
```

3.0 DATABASE

3.1 Relational Model and Relationship between Data

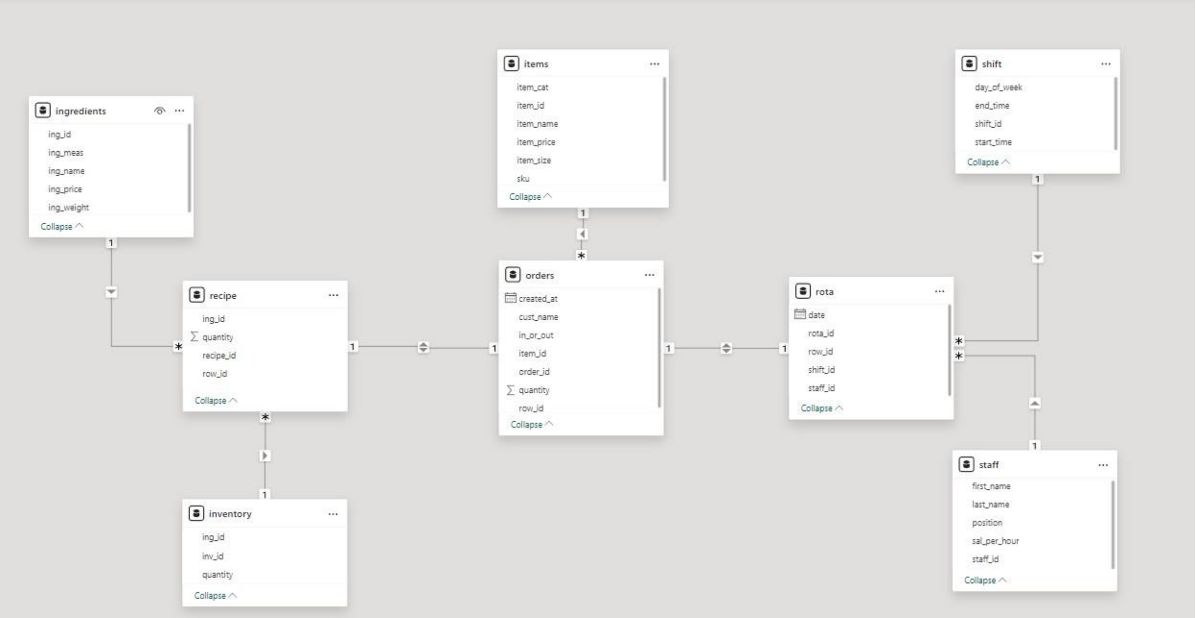


Figure 3.1 Relational Model using Power BI

3.2 Relationship between Data

Data	Relationship
orders -> items	many to one
recipe -> ingredients	many to one
recipe -> inventory	many to one
recipe -> orders	one to one
rota -> orders	one to one
rota -> shift	many to one
rota -> staff	many to one

3.3 Identification of Data Warehouse Schema

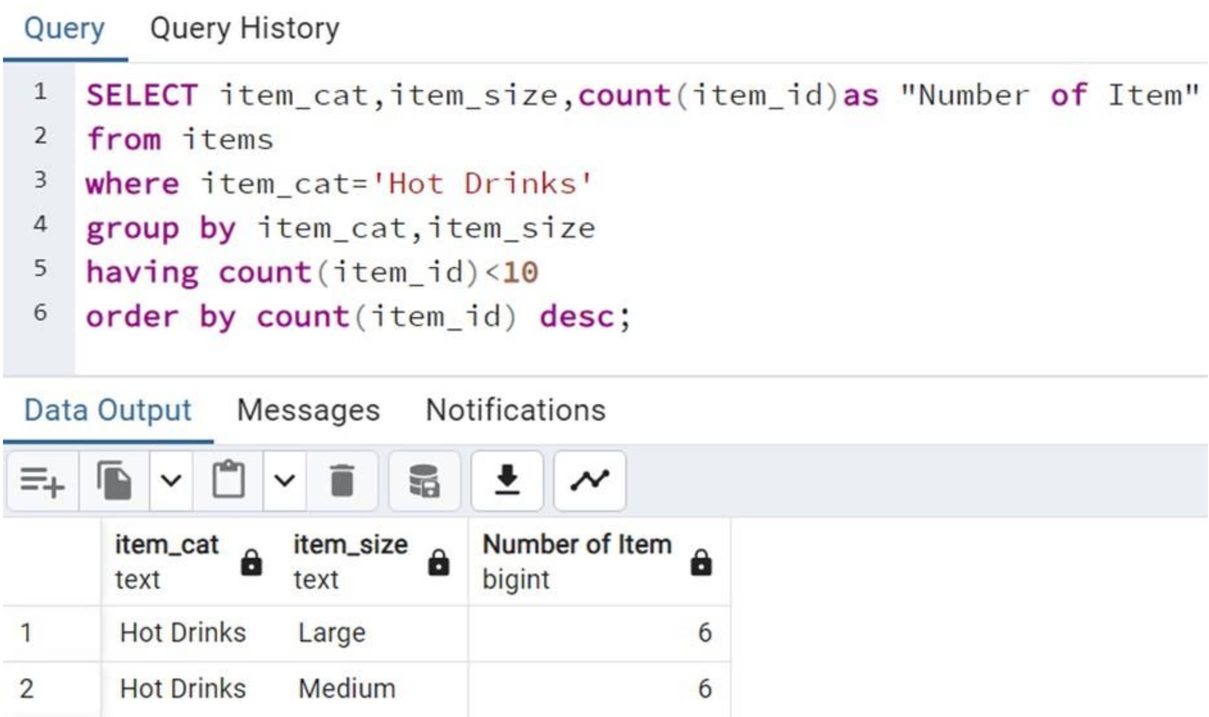
Based on Figure 3.1 above, the data warehouse schema of these datasets is Snowflake Schema because it has one fact table and is connected to three dimensional tables. The fact table is orders, and the dimensional table is items, recipe and rota. The recipe and rota dimension is connected to two child tables. The child tables for recipe dimension is inventory and ingredient while child tables for rota dimension is shift and staff.

4.0 RESULT AND DATA ANALYSIS

4.1 OLAP

We did OLAP processing, such as dicing, cube, rollup, and slicing using PostgreSQL.

4.1.1 Dicing

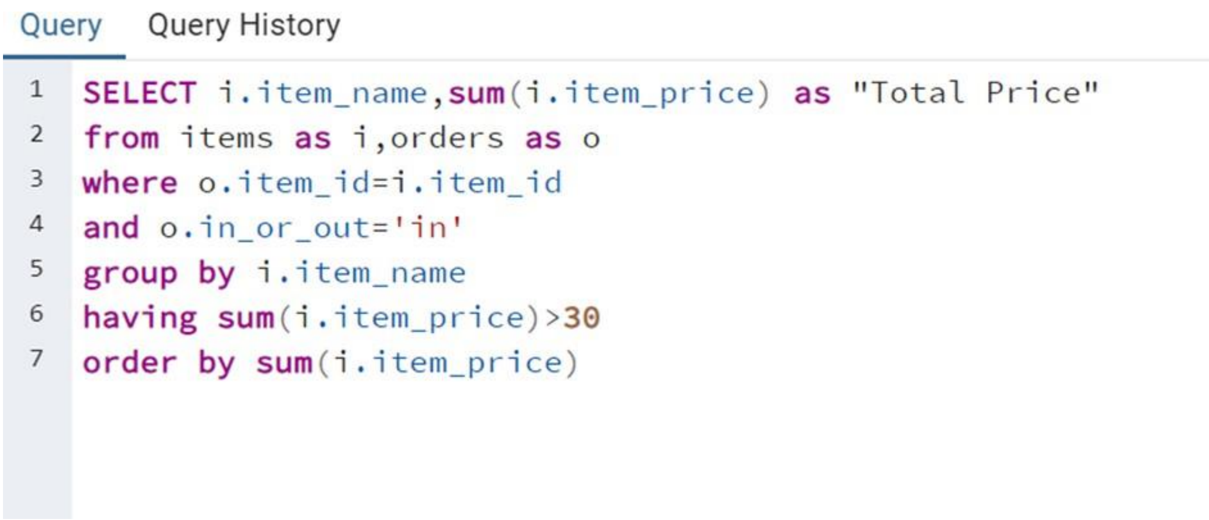


```sql
SELECT item_cat,item_size,count(item_id)as "Number of Item"
from items
where item_cat='Hot Drinks'
group by item_cat,item_size
having count(item_id)<10
order by count(item_id) desc;
```

	item_cat text	item_size text	Number of Item bigint
1	Hot Drinks	Large	6
2	Hot Drinks	Medium	6

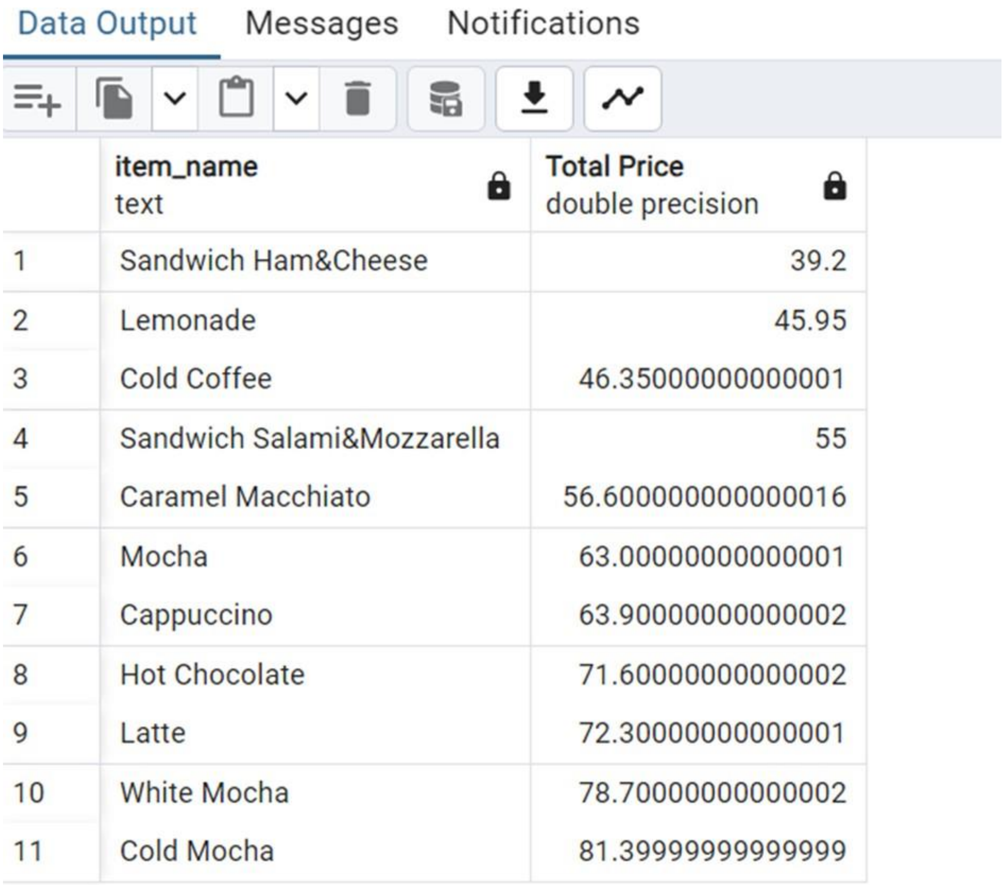
Figure 4.1.1

The OLAP operation shows the number of items and their corresponding category, size and item IDs. This operation filters by “Hot Drinks” category with count of item ID less than 10. According to the output, there are six items in the “Hot Drinks” category in “Large” size and six items in “Medium” size. This also shows that the business needs to focus on adding small size items to the shop. This is because there might be people who want to buy items only for themselves but the big portion in “Medium” and “Large” size will stop them from buying it. If we add “Small” size for the item, it is obvious that the number of customers in the shop will be increased.



```sql
SELECT i.item_name,sum(i.item_price) as "Total Price"
from items as i,orders as o
where o.item_id=i.item_id
and o.in_or_out='in'
group by i.item_name
having sum(i.item_price)>30
order by sum(i.item_price)
```

Figure 4.1.2

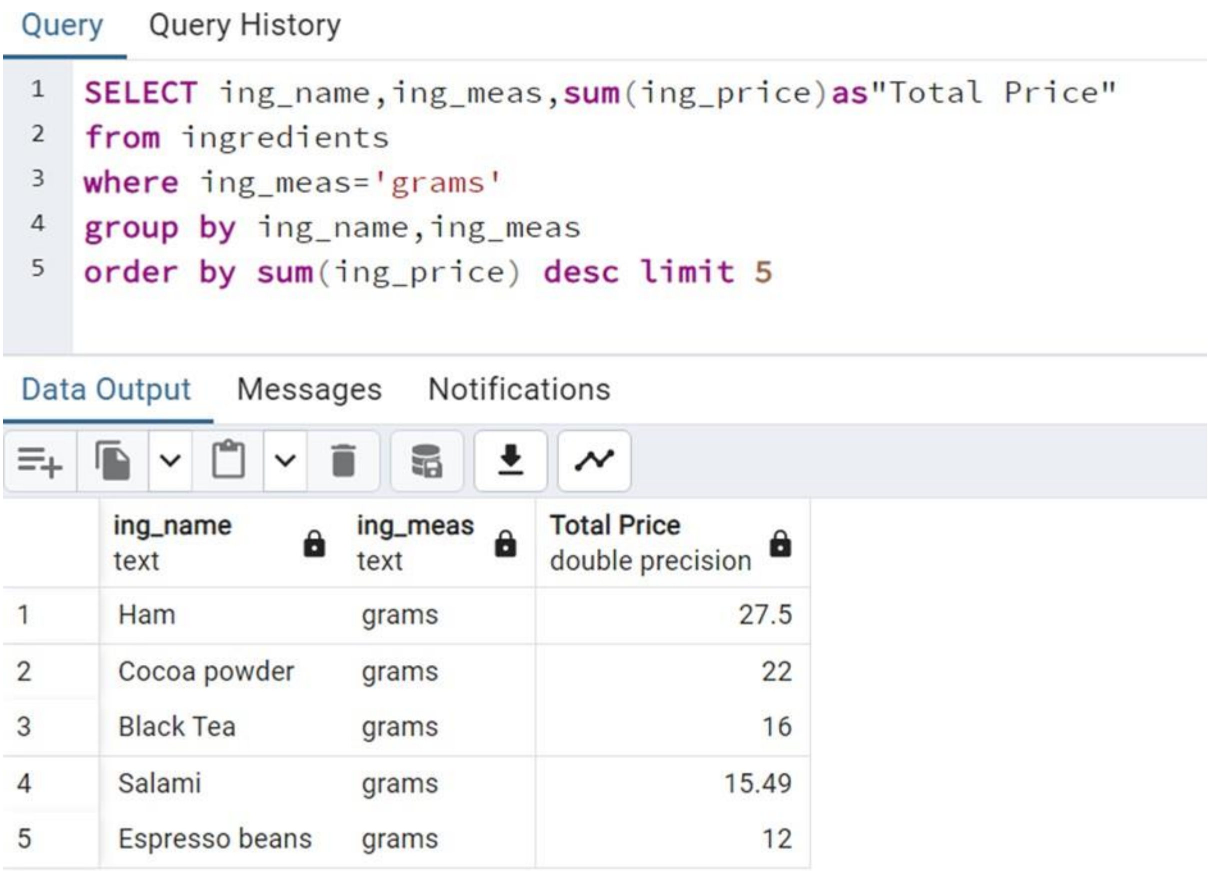


	item_name text	Total Price double precision
1	Sandwich Ham&Cheese	39.2
2	Lemonade	45.95
3	Cold Coffee	46.35000000000001
4	Sandwich Salami&Mozzarella	55
5	Caramel Macchiato	56.600000000000016
6	Mocha	63.00000000000001
7	Cappuccino	63.90000000000002
8	Hot Chocolate	71.60000000000002
9	Latte	72.30000000000001
10	White Mocha	78.70000000000002
11	Cold Mocha	81.39999999999999

Figure 4.1.3

This dicing operation shows the item name and total price for the item. The output is filtered to a total price of more than 30. The final output is ordered based on total price in ascending order. This operation clearly shows the highest total price is 81.39 which is “Cold Mocha”. We can say that the maximum revenue of the shop is from this item. So the business needs to make sure that this item should be always available for the customer to buy. The stock needed for this particular item always has to be extra.

4.1.2 Slicing



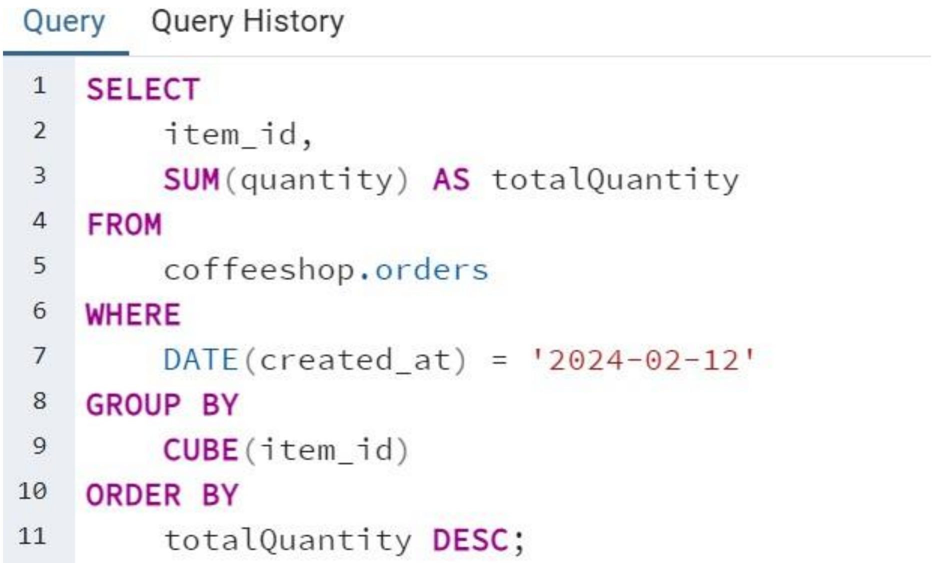
```
SELECT ing_name,ing_meas,sum(ing_price)as"Total Price"
from ingredients
where ing_meas='grams'
group by ing_name,ing_meas
order by sum(ing_price) desc limit 5
```

	ing_name text	ing_meas text	Total Price double precision
1	Ham	grams	27.5
2	Cocoa powder	grams	22
3	Black Tea	grams	16
4	Salami	grams	15.49
5	Espresso beans	grams	12

Figure 4.1.4

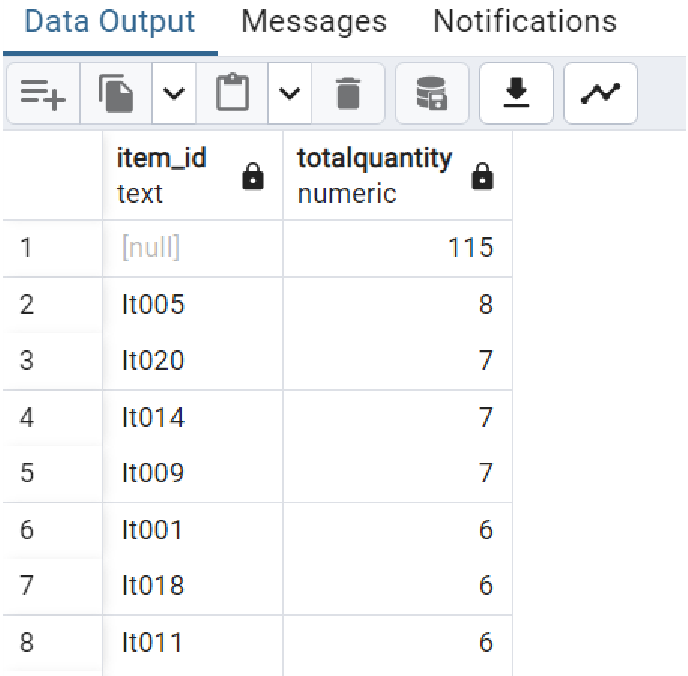
The output above shows ingredient names, measurements and the total price. In this slicing method is used to specify the ingredients that only can be measured as grams are included. So that we can find out which ingredients cost high, which is “Ham”. “Espresso Beans” cost least in the list. This also limits the outputs to top 5. This operation helps the business by making them assess the current inventory levels of ingredients in grams to make informed of the ingredients they in shop.

4.1.3 Cube



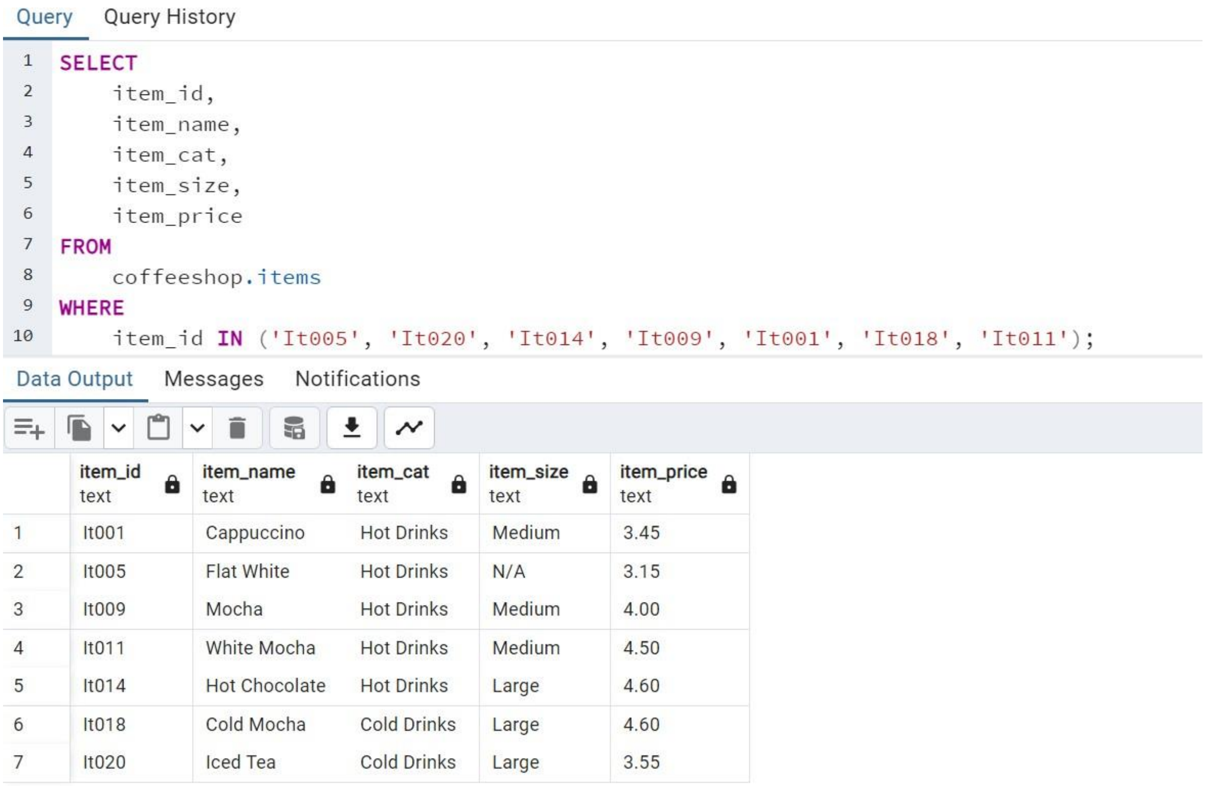
```
SELECT
    item_id,
    SUM(quantity) AS totalQuantity
FROM
    coffeeshop.orders
WHERE
    DATE(created_at) = '2024-02-12'
GROUP BY
    CUBE(item_id)
ORDER BY
    totalQuantity DESC;
```

Figure 4.1.5



	item_id text	totalquantity numeric
1	[null]	115
2	It005	8
3	It020	7
4	It014	7
5	It009	7
6	It001	6
7	It018	6
8	It011	6

Figure 4.1.6



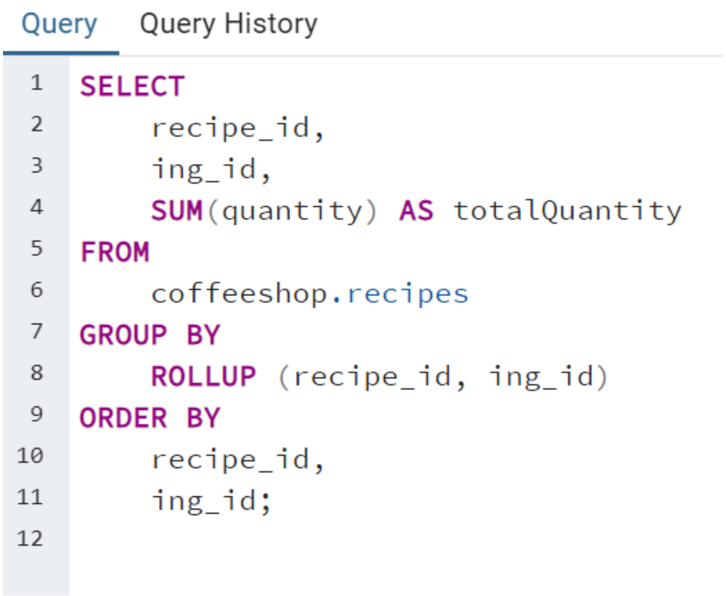
```
SELECT
    item_id,
    item_name,
    item_cat,
    item_size,
    item_price
FROM
    coffeeshop.items
WHERE
    item_id IN ('It005', 'It020', 'It014', 'It009', 'It001', 'It018', 'It011');
```

	item_id text	item_name text	item_cat text	item_size text	item_price text
1	It001	Cappuccino	Hot Drinks	Medium	3.45
2	It005	Flat White	Hot Drinks	N/A	3.15
3	It009	Mocha	Hot Drinks	Medium	4.00
4	It011	White Mocha	Hot Drinks	Medium	4.50
5	It014	Hot Chocolate	Hot Drinks	Large	4.60
6	It018	Cold Mocha	Cold Drinks	Large	4.60
7	It020	Iced Tea	Cold Drinks	Large	3.55

Figure 4.1.7

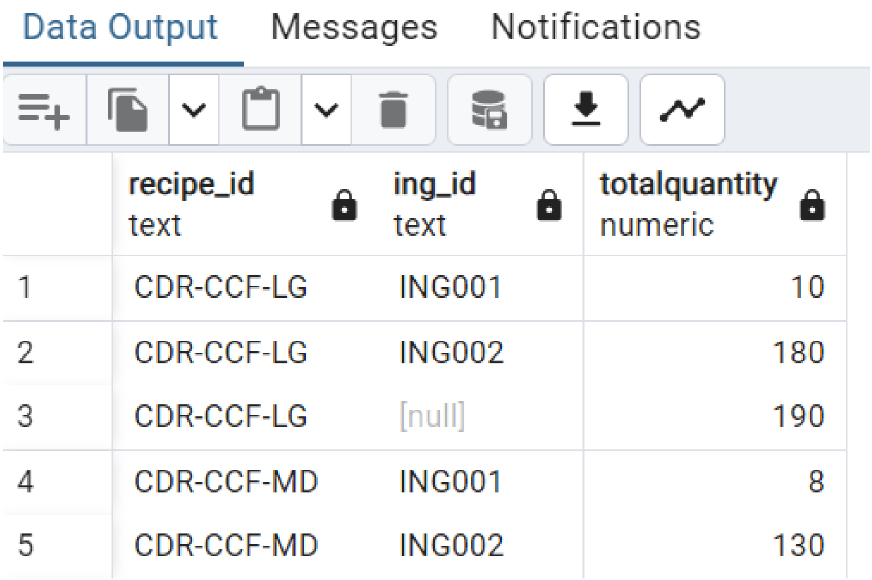
Based on the cube operation above (Figure 4.1.5 & Figure 4.1.6), we can identify the item that has the most quantity of orders on a specific date which is on 12-02-2024. From the above output on Figure 4.1.6, it shows a total of 115 orders on 12th February. We can discover that the item with item_id ‘It005’ has the most orders on that day. Hence, I did a quick check on the items table to see what is the items with the id ‘It005', 'It020', 'It014', 'It009', 'It001', 'It018', 'It011' (7 top items ordered on 12th February). From Figure 4.1.7, ‘It005’ with a total of 8 orders is a Flat White hot drink. This drink has the highest total sale on that day probably because it is on Monday which people most likely to have ‘Monday Blues’ and need a cup of Flat White to re-energize theirselves past weekends instead of its affordable price.

4.1.4 Roll-Up



```
SELECT
    recipe_id,
    ing_id,
    SUM(quantity) AS totalQuantity
FROM
    coffeeshop.recipes
GROUP BY
    ROLLUP (recipe_id, ing_id)
ORDER BY
    recipe_id,
    ing_id;
```

Figure 4.1.8



	recipe_id text	ing_id text	totalquantity numeric
1	CDR-CCF-LG	ING001	10
2	CDR-CCF-LG	ING002	180
3	CDR-CCF-LG	[null]	190
4	CDR-CCF-MD	ING001	8
5	CDR-CCF-MD	ING002	130

Figure 4.1.9

The output shows the total quantity of each ingredient used in each recipe. The sql rollup operation creates subtotals at each level of detail, from ingredient to recipe. For example, in the recipe ‘CDR-CCF-LG’, ingredient ‘ING001’ has a total quantity of 10, ingredient ‘ING002’ has 180, and the total quantity of all ingredients for this recipe is 190, indicated by NULL in the ingredient column.

4.2 Visualization

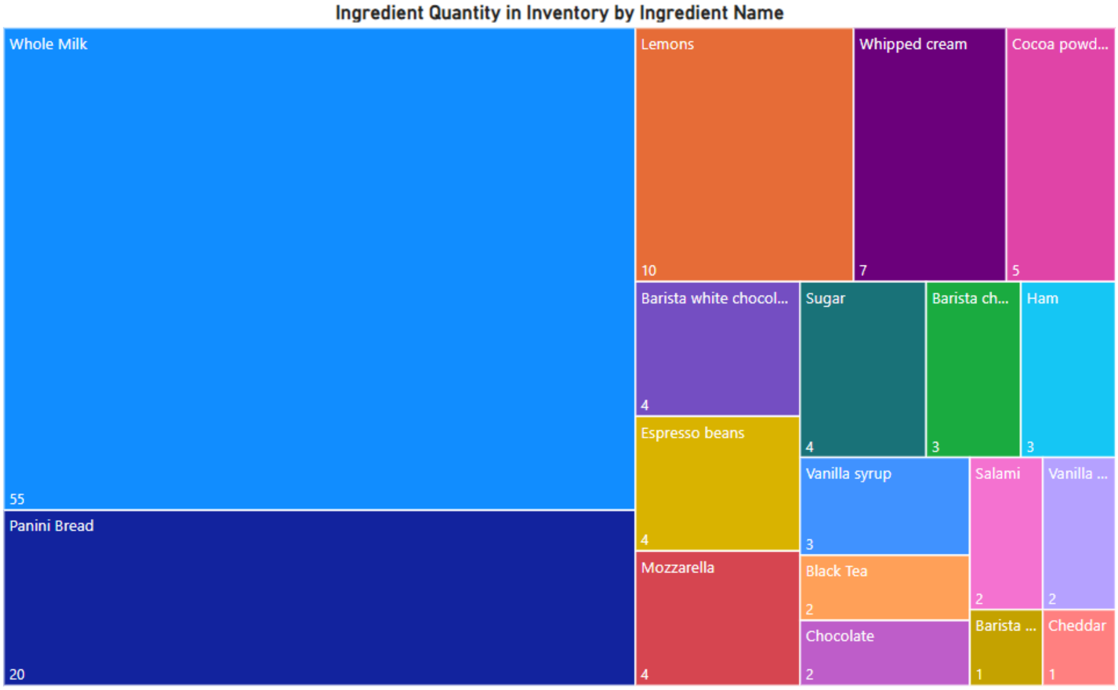


Figure 4.2.1 Tree map

The data shows the ingredient quantity in inventory by ingredient name. The highest ingredient quantity in inventory is whole milk which is 55 while the lowest ingredient quantity in inventory are barista caramel sauce and cheddar which is only 1. The whole milk is the highest quantity due to the staff making a wrong prediction about the sales of the item that contained whole milk. The staff predicted the sales of the item that contained whole milk will be higher than the reality. When the product that contained whole milk was not selling fast, the remaining stock of whole milk will reach the expire date and it will be wasted and the cost spent on the whole milk will not return back. The barista caramel sauce and cheddar is the lowest quantity because according to the sales report, the least food that customers ordered is a sandwich with salami & mozzarella and ham and cheese. So, the ingredient cheddar is the lowest quantity in the chart. The next lowest ingredient in the chart is barista caramel sauce as the sales of the caramel macchiato is out of the expectation and it really has a good result in the sales report. So, the quantity of the barista caramel sauce in the ingredient chart is the lowest than others. Therefore, they are the reasons why the whole milk is the highest quantity while the barista caramel sauce and cheddar are the lowest quantity in inventory.

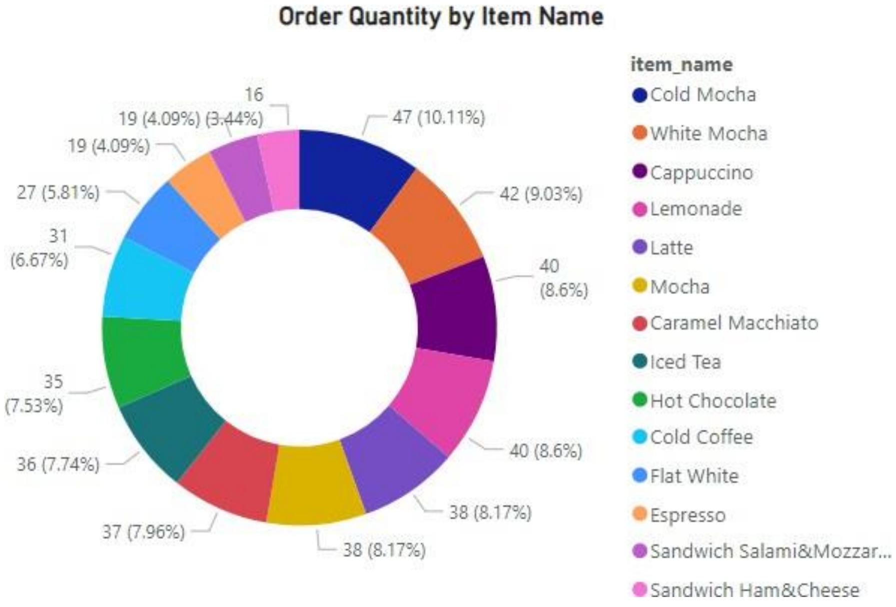


Figure 4.2.2 Donut Chart

From the donut chart view, the highest order quantity by customers of the Coffee Shop is cold mocha which is 47 cups, 10.11% while the lowest order quantity by customers is sandwich ham & cheese which is 16, 3.44%. This highest order quantity is cold mocha due to the Coffee Shop that has 2 types of customers which are dine-in and dine-out. Most of the dine-out customers are the office workers or students that are in the path and do not have much time to sit in the shop and have a proper breakfast meal. They prefer to grab the drinks and reach the office or college. The cold mocha is also the favourite drink among the customers of the Coffee Shop. The reason of the sandwich ham & cheese is the lowest order quantity by customers is because the price of the item maybe will be higher compared to the price other than Coffee Shop. So, customers are not willing to buy sandwiches in Coffee Shop. Furthermore, the number of customers that dine-out is more than dine-in as they prefer to buy drinks compared to hot items due to inconvenience. Therefore, cold mocha is the highest order quantity by customers while sandwich ham & cheese is the lowest order quantity by customers of the Coffee Shop.

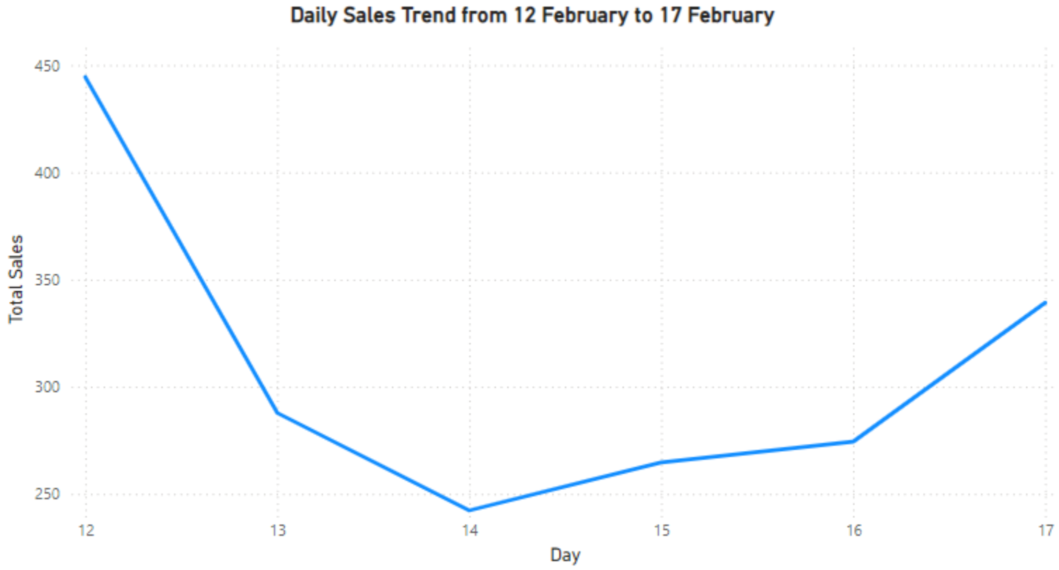


Figure 4.2.3 Line Chart

We have created a new measure which is ‘total sales’ to analyze sales of coffee shop by using following function:

```
Total Sales = SUMX('public orders','public orders'[quantity]*RELATED('public
items'[item_price]))
```

The above line chart shows the daily sales trends of coffee shop. As we can see here, from 12 February 2024 to 14 February 2024, the sales of coffee shop decreased from RM444.45 to RM242.40. This might have happened because fewer people visited the shop during weekdays. However, the sales increased from 14 February to 17 February 2024, reaching RM339.25. This increase could be due to more customers coming in towards the weekend or due to promotions during this period.

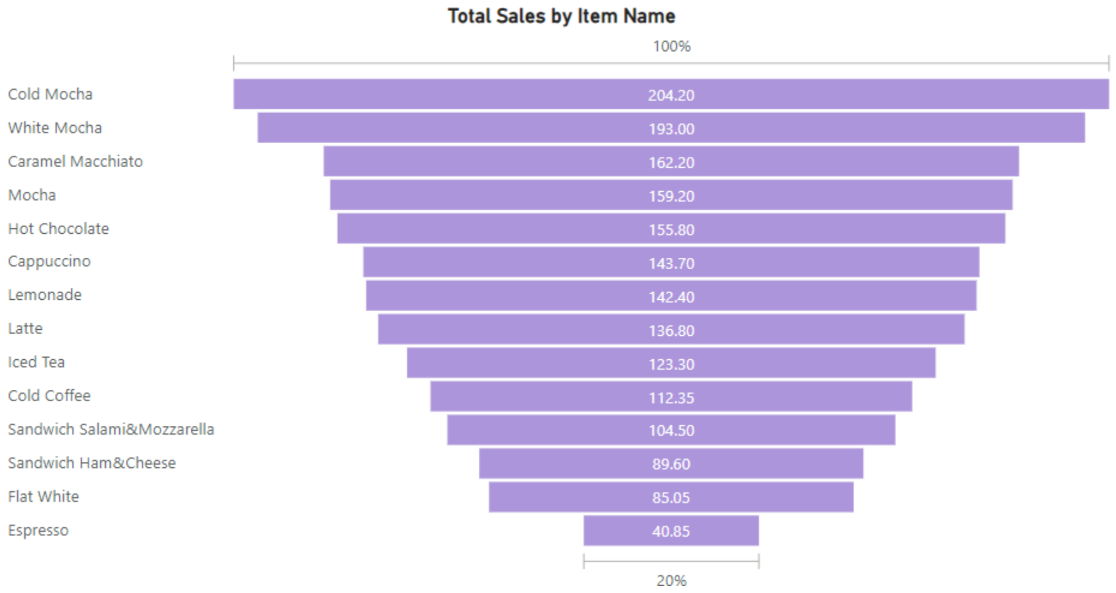


Figure 4.2.4 Funnel Chart

Based on the funnel chart view, the highest total sales of Coffee Shop is cold mocha which is 204.20 while the lowest total sales is espresso which is 40.85. The cold mocha is the highest total sales due to it being the highest order quantity by customers of Coffee Shop. The flavour of the cold mocha is recommended and favoured by most of the customers as they are willing to purchase the drink before they go to work or college. The packaging of the drink is convenient as the customers can enjoy the drink during their walk or in the car. On the other hand, espresso is the lowest sales of Coffee Shop although the order quantity of espresso is higher than the both of the types of sandwich. This is because the price of the espresso is lower than the price of the sandwich, so the total sales of espresso will become lower than the total sales of the sandwich. Therefore, the cold mocha is the highest total sales of Coffee Shop while espresso is the lowest total sales of Coffee Shop.

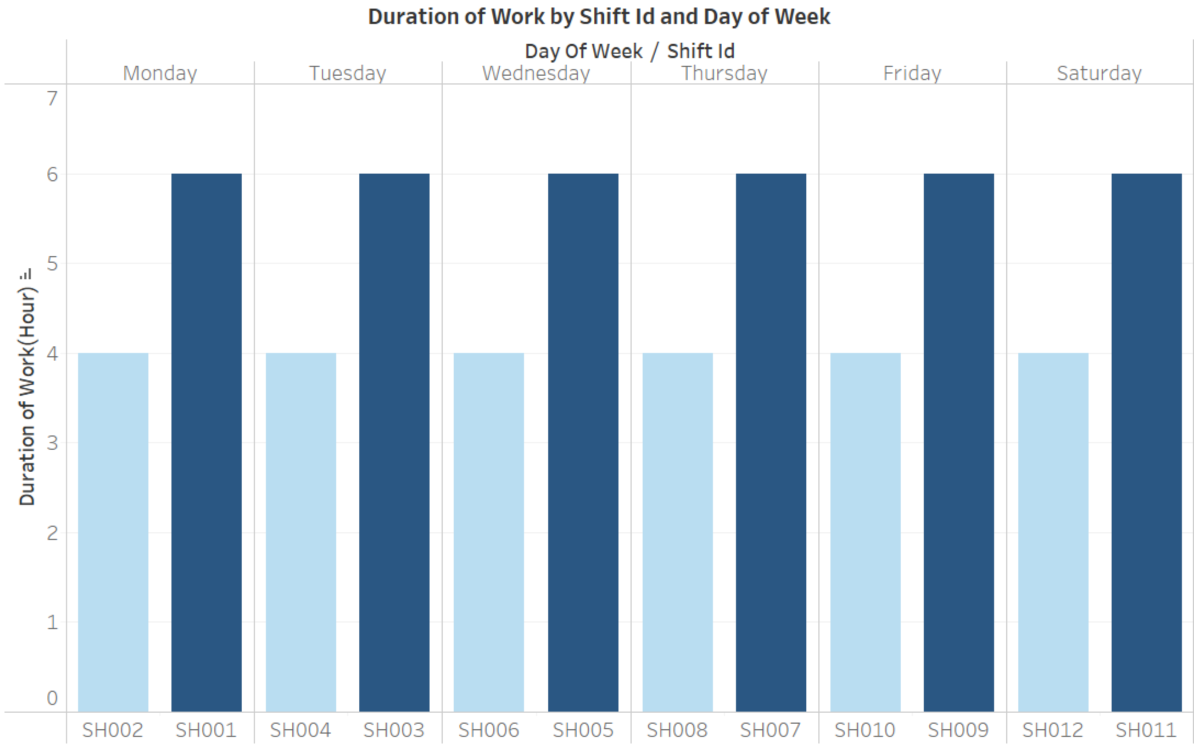


Figure 4.2.5 Side-by-side Bars

Based on the side-by-side bars view, we can see that the coffee shop operates with two distinct shifts each day, one's working shift is 4 hours and another is 6 hours. Having two shifts allows the coffee shop to optimise staffing levels based on different times of the day, potentially ensuring that there are enough staff during peak hours while minimising costs during slower periods. This also shows that the coffee shop provides flexibility in work scheduling to avoid overloading staff.

5.0 CONCLUSION

The data schema consists of eight tables which are orders, items, recipes, ingredients, inventory, staff, shift, and rota. Each table focuses on a different aspect of the business, such as orders, inventory management, staff details, and shift scheduling. The Kimball methodology is used to organise data for specific parts of the coffee shop business and gradually bring it all together into a data warehouse.

By applying the Kimball methodology and utilising the ETL process, it guarantees optimal performance and data consistency for the project, rendering it a valuable asset for the coffee shop. By providing a unified view of the business, the data warehouse can help the coffee shop to identify trends, patterns, and correlations that may not have been supposed before, leading to better business strategies and improved operational efficiency.

From the analysis result, it helps the coffee shop to identify areas of improvement and opportunities for growth. For instance, by analysing sales data, the coffee shop can determine which menu items are favoured by customers and which ones are less popular, allowing them to adjust their offerings accordingly. Similarly, by monitoring inventory levels, the coffee shop can optimise their ordering process and reduce waste. Furthermore, the data warehouse project can help the coffee shop to improve customer satisfaction by providing insights into customer preferences and behaviours. By analysing customer data, the coffee shop can tailor their offerings to meet the needs and preferences of their customers, leading to increased customer loyalty and repeat business. It also observes staff workloads to optimise scheduling and improve operational effectiveness, and determining the sales and revenue trend of the coffee shop.

In conclusion, the data warehouse project is a valuable asset for the coffee shop, providing stakeholders with robust analytical capabilities to gain actionable insights into coffee shop operations and make data-driven decisions. The project's success will enable the coffee shop to improve operational efficiency, identify areas of improvement and opportunities for growth, and improve customer satisfaction, ultimately leading to increased revenue and profitability.

6.0 REFERENCES

Buuck, B. (2022, May 31). Schemas Used in Data Warehouses: Star, Galaxy, and Snowflake. StreamSets. https://streamsets.com/blog/schemas-data-warehouses-star-galaxy-snowflake/

GeeksforGeeks. (2023, July 7). *What is Data Structure?* GeeksforGeeks. https://www.geeksforgeeks.org/data-structure-meaning/

Lutkevich, B., & Biscobing, J. (2021, June 24). *relational database*. Data Management. https://www.techtarget.com/searchdatamanagement/definition/relational-database

Naeem, T. (2024, March 21). Data Warehouse Concepts: Kimball vs. Inmon Approach | Astera. *Astera*. https://www.astera.com/type/blog/data-warehouse-concepts/

Raman, R. (2019, January 25). ETL Process in Data Warehouse - GeeksforGeeks. GeeksforGeeks. https://www.geeksforgeeks.org/etl-process-in-data-warehouse/

ThoughtSpot, T. (2023, December 7). *The fundamentals of data warehouse architecture*. ThoughtSpot.https://www.thoughtspot.com/data-trends/data-modeling/data-warehouse-architecture

7.0 APPENDIX

PostgreSQL(extract)

Query to create a new database:

```
CREATE DATABASE Group_project;
```

Query to create a new database:

```
CREATE TABLE ingredients(
ing_id text,
ing_name text,
ing_weight numeric,
ing_meas text,
ing_price numeric
);

CREATE TABLE inventory(
inv_id    text,
ing_id text,
quantity int
);

CREATE TABLE items(
item_id text,
sku text,
item_name text,
item_cat text,
item_size text,
item_price numeric
);

CREATE TABLE orders(
row_id int,
order_id text,
created_date timestamp,
item_id text,
quantity int,
cust_name text,
dinein_or_takeout text
);

CREATE TABLE recipe(
row_id int,
recipe_id text,
ing_id text,
quantity int
);

CREATE TABLE rota(
row_id int,
rota_id    text,
rota_date timestamp,
shift_id text,
staff_id text
);

CREATE TABLE shift(
shift_id text,
day_of_week text,
start_time timestamp,
end_time timestamp
);

CREATE TABLE staff(
staff_id text,
first_name text,
last_name text,
worker_position text,
sal_per_hour numeric
);
```

Query to connect the PostgreSQL with Jupyter Notebook

```
CREATE DATABASE "AdventureWorks"
    WITH
    OWNER = postgres
    ENCODING = 'UTF8'
    LC_COLLATE = 'English_Malaysia.1252'
    LC_CTYPE = 'English_Malaysia.1252'
    TABLESPACE = pg_default
    CONNECTION LIMIT = -1;

CREATE USER etl with PASSWORD 'demopass';

GRANT CONNECT ON DATABASE "AdventureWorks" TO etl;

GRANT SELECT, INSERT, UPDATE, DELETE ON ALL TABLES IN SCHEMA
public TO etl;
```

Jupyter NoteBook (transformaton and cleaning process)

```
!pip install psycopg2
!pip install SQLalchemy
from sqlalchemy import create_engine
from sqlalchemy.engine import URL
import pandas as pd
import numpy as np
uid = 'etl'
pwd = 'demopass'
server = "localhost"
database = "Group_project"
engine = create_engine(f'postgresql://{uid}:{pwd}@{server}:5432/{database}')
```

Cleaning data

Table 1

```
sql1= "SELECT * FROM ingredients;"
df1 = pd.read_sql_query(sql1,engine)
df1
df1.isnull().sum()
df1.duplicated().sum()
df1.shape
```

Table 2

```
sql2 = "SELECT * FROM inventory;"
df2 = pd.read_sql_query(sql2,engine)
df2
df2.isnull().sum()
df2.duplicated().sum()
df2.shape
```

Table 3

```
sql3 = "SELECT * FROM items;"
df3 = pd.read_sql_query(sql3,engine)
df3
df3.isnull().sum()
df3.duplicated().sum()
df3.shape
```

Table 4

```
sql4 = "SELECT * FROM orders;"
df4 = pd.read_sql_query(sql4,engine)
df4
df4.drop('row_id', axis=1, inplace=True)
df4
blank_replacements = ['', ' ', '\t', 'None', 'none', 'NULL', 'N/A']
df4.replace(blank_replacements, np.nan, inplace=True)
```

```
df4.isna().sum()
df4=df4.ffill()
df4
df4.isnull().sum()
df4.duplicated().sum()
df4=df4.drop_duplicates()
df4.duplicated().sum()
df4.shape
```

Table 5

```
sql5 = "SELECT * FROM recipe;"
df5 = pd.read_sql_query(sql5,engine)
df5
df5.drop('row_id', axis=1, inplace=True)
df5
df5.isnull().sum()
df5.duplicated().sum()
Df5.shape
```

Table 6

```
sql6 = "SELECT * FROM rota;"
df6 = pd.read_sql_query(sql6,engine)
df6
df6.drop('row_id', axis=1, inplace=True)
df6
df6.isnull().sum()
df6.duplicated().sum()
df6.shape
```

Table 7

```
sql7 = "SELECT * FROM shift;"
df7 = pd.read_sql_query(sql7,engine)
df7
df7.isnull().sum()
```

```
df7.duplicated().sum()
df7.shape
```

Table 8

```
sql8= "SELECT * FROM staff;"
df8 = pd.read_sql_query(sql8,engine)
df8
df8.isnull().sum()
df8.duplicated().sum()
df8.shape
```

Download CSV

Table 1

```
from IPython.display import HTML
import base64
import pandas as pd

def create_download_link( df1, title = "Download CSV file", filename = "ingredients.csv"):
        csv = df1.to_csv(index =False)
        b64 = base64.b64encode(csv.encode())
        payload = b64.decode()
        html = '<a download="{filename}" href="data:text/csv;base64,{payload}" target="_blank">{title}</a>'
        html = html.format(payload=payload,title=title,filename=filename)
        return HTML(html)

create_download_link(df1)
```

Table 2

```
from IPython.display import HTML
import base64
import pandas as pd
```

```
def create_download_link( df2, title = "Download CSV file", filename =
"inventory.csv"):
        csv = df2.to_csv(index =False)
        b64 = base64.b64encode(csv.encode())
        payload = b64.decode()
        html = '<a download="{filename}" href="data:text/csv;base64,{payload}"
target="_blank">{title}</a>'
        html = html.format(payload=payload,title=title,filename=filename)
        return HTML(html)
create_download_link(df2)
```

Table 3

```
from IPython.display import HTML
import base64
import pandas as pd

def create_download_link( df3, title = "Download CSV file", filename = "items.csv"):
        csv = df3.to_csv(index =False)
        b64 = base64.b64encode(csv.encode())
        payload = b64.decode()
        html = '<a download="{filename}" href="data:text/csv;base64,{payload}"
target="_blank">{title}</a>'
        html = html.format(payload=payload,title=title,filename=filename)
        return HTML(html)

create_download_link(df3)
```

Table 4

```
from IPython.display import HTML
import base64
import pandas as pd

def create_download_link( df4, title = "Download CSV file", filename = "orders.csv"):
        csv = df4.to_csv(index =False)
        b64 = base64.b64encode(csv.encode())
        payload = b64.decode()
        html = '<a download="{filename}" href="data:text/csv;base64,{payload}"
target="_blank">{title}</a>'
        html = html.format(payload=payload,title=title,filename=filename)
        return HTML(html)

create_download_link(df4)
```

Table 5

```
from IPython.display import HTML
import base64
import pandas as pd

def create_download_link( df5, title = "Download CSV file", filename = "recipe.csv"):
        csv = df5.to_csv(index =False)
        b64 = base64.b64encode(csv.encode())
        payload = b64.decode()
        html = '<a download="{filename}" href="data:text/csv;base64,{payload}"
target="_blank">{title}</a>'
        html = html.format(payload=payload,title=title,filename=filename)
        return HTML(html)

create_download_link(df5)
```

Table 6

```
from IPython.display import HTML
import base64
import pandas as pd

def create_download_link( df6, title = "Download CSV file", filename = "rota.csv"):
        csv = df6.to_csv(index =False)
        b64 = base64.b64encode(csv.encode())
        payload = b64.decode()
```

```
    html = '<a download="{filename}" href="data:text/csv;base64,{payload}"
target="_blank">{title}</a>'
    html = html.format(payload=payload,title=title,filename=filename)
    return HTML(html)

create_download_link(df6)
```

Table 7

```
from IPython.display import HTML
import base64
import pandas as pd

def create_download_link( df7, title = "Download CSV file", filename = "shift.csv"):
    csv = df7.to_csv(index =False)
    b64 = base64.b64encode(csv.encode())
    payload = b64.decode()
    html = '<a download="{filename}" href="data:text/csv;base64,{payload}"
target="_blank">{title}</a>'
    html = html.format(payload=payload,title=title,filename=filename)
    return HTML(html)

create_download_link(df7)
```

Table 8

```
from IPython.display import HTML
import base64
import pandas as pd

def create_download_link( df8, title = "Download CSV file", filename = "staff.csv"):
    csv = df8.to_csv(index =False)
    b64 = base64.b64encode(csv.encode())
    payload = b64.decode()
    html = '<a download="{filename}" href="data:text/csv;base64,{payload}"
target="_blank">{title}</a>'
```

```
    html = html.format(payload=payload,title=title,filename=filename)
    return HTML(html)

create_download_link(df8)
```

PostgreSQL(load the new dataset into new database and tables)

Create new database

```
CREATE DATABASE Group_project;
```

Create new tables to load the cleaned data

```
CREATE TABLE ingredients(
ing_id text,
ing_name text,
ing_weight numeric,
ing_meas text,
ing_price numeric
);

CREATE TABLE inventory(
inv_id text,
ing_id text,
quantity int
);

CREATE TABLE items(
item_id text,
sku text,
item_name text,
item_cat text,
item_size text,
item_price numeric
);
```

```
CREATE TABLE orders(
order_id text,
created_date timestamp,
item_id text,
quantity int,
cust_name text,
dinein_or_takeout text
);

CREATE TABLE recipe(
recipe_id text,
ing_id text,
quantity int
);

CREATE TABLE rota(
rota_id text,
rota_date date,
shift_id text,
staff_id text
);

CREATE TABLE shift(
shift_id text,
day_of_week text,
start_time time,
end_time time
);

CREATE TABLE staff(
staff_id text,
first_name text,
last_name text,
worker_position text,
sal_per_hour numeric
);
```