UNIVERSITI MALAYSIA PAHANG
AL-SULTAN ABDULLAH

BSD2343
DATA WAREHOUSING
2022/2023 SEMESTER II

GROUP PROJECT
TITLE: Inventory Analysis

PREPARED FOR:
DR AZUANA BINTI RAMLI

NAME	ID NUMBER	SECTION
NUR NABILA BINTI ABD RAHMAN	SD22037	01G
SITI MAISARAH BINTI SUHARDI	SD22006	01G
NURUL ALIS BINTI YUSRI	SD22045	02G
MUHAMMAD SYAHIR BIN MOHD KHALIL	SD22040	02G
SHALINI A/P MAGESWARAN	SD21051	01G

TABLE OF CONTENT

1.0 BACKGROUND

1.1 Project Background

This project focuses on inventory optimization in a manufacturing company that runs a retail wine and spirits shop with several outlets. They offer goods and balancing inventory management, such as stockouts, excess inventory, raw material stock levels, and finished products. The goal of this project is to leverage extensive data analysis to optimize inventory control and extract valuable insights from the company's operation, particularly sales and purchases. To promote SDG12 Responsible Consumption and Production by making sure that this project creates impact in sustainability consumption and production by assessing sustainability by current inventory practice. It is to reduce waste and manage heterogeneity in resource utilization by assessing the environment.

1.2 Description of the selected project

Beginning inventory Table

VARIABLES	DATA TYPE	DESCRIPTION
inventoryId	Integer	Unique identifier for each inventory item.
store	String	Name or code representing the store location.
city	String	Name of the city where the store is located.
brand	String	Brand name of the inventory item.
description	String	Description of the inventory item.
size	String	Size or dimensions of the inventory item.
onHand	Integer	Quantity of the inventory item currently in stock.
price	Float	Price of the inventory item.
startDate	Date	Date when the inventory item was added to stock or made available for sale.

Ending inventory Table

VARIABLES	DATA TYPE	DESCRIPTION
inventoryId	Integer	Unique identifier for each inventory item.
store	String	Name or code representing the store location.
city	String	Name of the city where the store is located.
brand	String	Brand name of the inventory item.
description	String	Description of the inventory item.
size	String	Size or dimensions of the inventory item.
onHand	Integer	Quantity of the inventory item currently in stock.
price	Float	Price of the inventory item.
endDate	Date	Date when the inventory item was sold or removed from stock

Invoice Table

VARIABLES	DATA TYPE	DESCRIPTION
vendorNumber	Integer	Unique identifier for each vendor.
vendorName	String	Name of the vendor supplying the goods
invoiceDate	Date	Date when the invoice was issued by the vendor.
pONumber	String	Purchase order number associated with the transaction.
pODate	Date	Date when the purchase order was issued.
payDate	Date	Date when payment was made to the vendor.
quantity	Integer	Quantity of goods purchased.
dollars	Float	Total amount paid for the goods.
freight	Float	Freight or shipping charges associated with the transaction.
approval	Boolean	Indicates whether the transaction is approved.

Purchase Table

VARIABLES	DATA TYPE	DESCRIPTION
inventoryId	Integer	Unique identifier for the inventory item.
store	String	Name or code representing the store location.
brand	String	Brand name of the inventory item.
description	String	Description of the inventory item.
size	String	Size or dimensions of the inventory item.
vendorNumber	Integer	Unique identifier for each vendor.
vendorName	String	Name of the vendor supplying the goods
pONumber	String	Purchase order number associated with the transaction.
pODate	Date	Date when the purchase order was issued.
receivingDate	Date	Date the product was received.
invoiceDate	Date	Date the invoice was issued.
payDate	Date	Date the payment was made.
purchasePrice	Float	Price at which the product was purchased.
quantity	Integer	Quantity of the product purchased.
dollars	Float	Total cost in dollars (purchasePrice * quantity).
classification	String	Category or classification of the product.

Purchase Price Table

VARIABLES	DATA TYPE	DESCRIPTION
brand	String	Brand name of the inventory item
description	String	Description of the inventory item.
price	Float	Price of the inventory item.
size	String	Size of the inventory item.

volume	String	Volume of the inventory item.
classification	String	Category or classification of the product.
purchasePrice	Float	Purchase price of the inventory item.
vendorNumber	Integer	Unique identifier for each vendor.
vendorName	String	Name of the vendor supplying the goods

Sales Table

VARIABLES	DATA TYPE	DESCRIPTION
inventoryId	Integer	Unique identifier for the inventory item.
store	String	Name or code representing the store location.
brand	String	Brand name of the inventory item.
description	String	Description of the inventory item.
size	String	Size of the inventory item.
salesQuantity	Integer	Quantity of the item sold.
salesDollars	Float	Total sales in dollars.
salesPrice	Float	Sales price per unit.
salesDate	Date	Date of the sale.
volume	String	Volume of the inventory item.
classification	String	Category or classification of the product.
exciseTax	Float	Excise tax applied to the sale.
vendorNumber	Integer	Unique identifier for each vendor.
vendorName	String	Name of the vendor supplying the goods

1.3 Problem to be Solved

The problem to be addressed is the need for comprehensive analysis and understanding of inventory analysis. By determining optimal inventory levels for inventory management, including stockouts, excess inventory, stocks of raw materials and finished products, effectively reducing instances of stockouts and excess inventory. More than a million records of sales, purchases, and inventory employment make depicting data in traditional spreadsheets ineffective. The use of information technology in the company is outdated, especially regarding analysis of sales and purchase data for drawing significant information. Then, there is no current precaution being taken in the stock management to use sustainable methods that would help minimize the depletion of the resources, so their environmental impact analysis is poor. We need to develop strategies for a better linkage of this management with SDG 12 for better consumption and production.

1.4 Objective

Our objective is to analysis data observed in inventories over different time periods and geographical locations into their basic trends' patterns and drivers.

- Analyze the inventory management process and suggest recommendations for improvement.
- Develop a sustainable inventory management strategy for future growth.

1.5 Data schema

A data schema is a collection of database objects, including tables, views, indexes, and synonyms. Schema models designed for data warehousing arrange schema objects in various ways. The inventory dataset consists of six tables: beginning inventory, ending inventory, Invoice Table, Purchase Table, Purchase Price Table, and Sales Table.

We used two Jupyter libraries to display the data schema: Pandas and Numpy.

```
Import pandas as pd
Import numpy as np
```

Beginning Inventory Table

```
[5]  # Load the csv into a dataframe
     beg_inv = spark.read.csv("beg_inv.csv", header=True, inferSchema=True)
```

```
[6]  beg_inv.printSchema()
```

```
root
 |-- inventoryId: string (nullable = true)
 |-- store: integer (nullable = true)
 |-- city: string (nullable = true)
 |-- brand: integer (nullable = true)
 |-- description: string (nullable = true)
 |-- size: string (nullable = true)
 |-- onHand: integer (nullable = true)
 |-- price: double (nullable = true)
 |-- startDate: string (nullable = true)
```

```
datasets = [("Beginning Inventory", beg_inv),
            ("Ending Inventory", end_inv),
            ("Purchase", purchase),
            ("Purchase Invoices", invoice),
            ("Purchase Price", purchase_price),
            ("Sales", sales)]

for index, (dataset_name, dataset) in enumerate(datasets, start=1):
    print(f"{index}. Explore {dataset_name} dataset key details:")
    print(explore_dataset(dataset))
    print("\n")
```

```
1. Explore Beginning Inventory dataset key details:
        Column Data Type  total count  Unique Count  \
0  inventoryid    object       206529        206529
1        store     int64       206529            79
2         city    object       206529            67
3        brand     int64       206529          8094
4  description    object       206529          7291
5         size    object       206529            41
6       onhand     int64       206529           474
7        price   float64       206529           329
8    startdate    object       206529             1
```

Figures 1.5.1 Beginning Inventory Table

The figure 1.5.1 shows the data schema of beginning inventory table. The tables have 9 columns and data type of string, integer and float.

Ending Inventory Table

```
[8]  # Load the csv into a dataframe
     end_inv = spark.read.csv("end_inv.csv", header=True, inferSchema=True)
```

```
[9]  end_inv.printSchema()
```

```
root
 |-- inventoryId: string (nullable = true)
 |-- store: integer (nullable = true)
 |-- city: string (nullable = true)
 |-- brand: integer (nullable = true)
 |-- description: string (nullable = true)
 |-- size: string (nullable = true)
 |-- onHand: integer (nullable = true)
 |-- Price: double (nullable = true)
 |-- endDate: string (nullable = true)
```

```
datasets = [("Beginning Inventory", beg_inv),
            ("Ending Inventory", end_inv),
            ("Purchase", purchase),
            ("Purchase Invoices", invoice),
            ("Purchase Price", purchase_price),
            ("Sales", sales)]

for index, (dataset_name, dataset) in enumerate(datasets, start=1):
    print(f"{index}. Explore {dataset_name} dataset key details:")
    print(explore_dataset(dataset))
    print("\n")
```

```
2. Explore Ending Inventory dataset key details:
         Column Data Type  total count  Unique Count  \
0   inventoryId    object       224489        224489
1         store     int64       224489            80
2          city    object       224489            67
3         brand     int64       224489          9653
4   description    object       224489          8732
5          size    object       224489            47
6        onHand     int64       224489           548
7         Price   float64       224489           354
8       endDate    object       224489             1
```

Figures 1.5.2 Ending Inventory Table

The figure 1.5.2 shows the data schema of the ending inventory table. The tables have 9 columns and data type of string, integer and float.

Invoice Table

```
[10] # Load the csv into a dataframe
     invoice = spark.read.csv("invoice.csv", header=True, inferSchema=True)
```

```
[11] invoice.printSchema()
```

```
root
 |-- vendorNumber: integer (nullable = true)
 |-- vendorName: string (nullable = true)
 |-- invoiceDate: string (nullable = true)
 |-- pONumber: integer (nullable = true)
 |-- pODate: string (nullable = true)
 |-- payDate: string (nullable = true)
 |-- quantity: integer (nullable = true)
 |-- dollars: double (nullable = true)
 |-- freight: double (nullable = true)
 |-- approval: string (nullable = true)
```

```
datasets = [("Beginning Inventory", beg_inv),
            ("Ending Inventory", end_inv),
            ("Purchase", purchase),
            ("Purchase Invoices", invoice),
            ("Purchase Price", purchase_price),
            ("Sales", sales)]

for index, (dataset_name, dataset) in enumerate(datasets, start=1):
    print(f"{index}. Explore {dataset_name} dataset key details:")
    print(explore_dataset(dataset))
    print("\n")
```

```
4. Explore Purchase Invoices dataset key details:
         Column Data Type  total count  Unique Count  \
0  vendorNumber     int64         5543           126
1    vendorName    object         5543           129
2   invoiceDate    object         5543           373
3      pONumber     int64         5543          5543
4        pODate    object         5543           319
5       payDate    object         5543           382
6      quantity     int64         5543          2895
7       dollars   float64         5543          5226
8       freight   float64         5543          4052
9      approval    object         5543             1
```

Figures 1.5.3 Invoice Table

The figure 1.5.3 shows the data schema of invoice table. The tables have 10 columns and data type of string, integer and float.

Purchase Table

```
[12] # Load the csv into a dataframe
     purchase = spark.read.csv("purchase.csv", header=True, inferSchema=True)
```

```
purchase.printSchema()
```

```
root
 |-- inventoryId: string (nullable = true)
 |-- store: integer (nullable = true)
 |-- brand: integer (nullable = true)
 |-- description: string (nullable = true)
 |-- size: string (nullable = true)
 |-- vendorNumber: integer (nullable = true)
 |-- vendorName: string (nullable = true)
 |-- pONumber: integer (nullable = true)
 |-- pODate: string (nullable = true)
 |-- receivingDate: string (nullable = true)
 |-- invoiceDate: string (nullable = true)
 |-- payDate: string (nullable = true)
 |-- purchasePrice: double (nullable = true)
 |-- quantity: integer (nullable = true)
 |-- dollars: double (nullable = true)
 |-- classification: integer (nullable = true)
```

```
3. Explore Purchase dataset key details:
            Column Data Type  total count  Unique Count  \
0      inventoryId    object      1048575        194912
1            store     int64      1048575            79
2            brand     int64      1048575          8512
3      description    object      1048575          7657
4             size    object      1048575            45
5     vendorNumber     int64      1048575           117
6       vendorName    object      1048575           118
7         pONumber     int64      1048575          2640
8           pODate    object      1048575           162
9    receivingDate    object      1048575           181
10     invoiceDate    object      1048575           190
11         payDate    object      1048575           198
12   purchasePrice   float64      1048575          1873
13        quantity     int64      1048575           448
14         dollars   float64      1048575         23971
15  classification     int64      1048575             2
```

Figures 1.5.4 Purchase Table

The figure 1.5.4 shows the data schema of purchase table. The tables have 16 columns and data type of string, integer and float.

Purchase Price Table

```
# Load the csv into a dataframe
purchase_price = spark.read.csv("purchase_price.csv", header=True, inferSchema=True)
```

```
[15] purchase_price.printSchema()
```

```
root
 |-- brand: integer (nullable = true)
 |-- description: string (nullable = true)
 |-- price: double (nullable = true)
 |-- size: string (nullable = true)
 |-- volume: double (nullable = true)
 |-- classification: integer (nullable = true)
 |-- purchasePrice: double (nullable = true)
 |-- vendorNumber: integer (nullable = true)
 |-- vendorName: string (nullable = true)
```

```
datasets = [("Beginning Inventory", beg_inv),
            ("Ending Inventory", end_inv),
            ("Purchase", purchase),
            ("Purchase Invoices", invoice),
            ("Purchase Price", purchase_price),
            ("Sales", sales)]

for index, (dataset_name, dataset) in enumerate(datasets, start=1):
    print(f"{index}. Explore {dataset_name} dataset key details:")
    print(explore_dataset(dataset))
    print("\n")
```

```
5. Explore Purchase Price dataset key details:
           Column Data Type  total count  Unique Count  \
0           brand     int64        12261         12261
1     description    object        12261         11114
2           price   float64        12261           380
3            size    object        12261            55
4          volume   float64        12261            32
5  classification     int64        12261             2
6   purchasePrice   float64        12261          2314
7    vendorNumber     int64        12261           131
8      vendorName    object        12261           136
```

Figures 1.5.5 Purchase Price Table

The figure 1.5.5 shows the data schema of purchase price table. The tables have 9 columns and data type of string, integer and float.

Sales Table

```
[16] # Load the csv into a dataframe
     sales = spark.read.csv("sales.csv", header=True, inferSchema=True)
```

```
[17] sales.printSchema()
```

```
root
 |-- inventoryId: string (nullable = true)
 |-- store: integer (nullable = true)
 |-- brand: integer (nullable = true)
 |-- description: string (nullable = true)
 |-- size: string (nullable = true)
 |-- salesQuantity: integer (nullable = true)
 |-- salesDollars: double (nullable = true)
 |-- salesPrice: double (nullable = true)
 |-- salesDate: string (nullable = true)
 |-- volume: integer (nullable = true)
 |-- classification: integer (nullable = true)
 |-- exciseTax: double (nullable = true)
 |-- vendorNumber: integer (nullable = true)
 |-- vendorName: string (nullable = true)
```

```
6. Explore Sales dataset key details:
            Column Data Type  total count  Unique Count  \
0      inventoryId    object      1048575        170131
1            store     int64      1048575            79
2            brand     int64      1048575          7658
3      description    object      1048575          6890
4             size    object      1048575            40
5    salesQuantity     int64      1048575           141
6     salesDollars   float64      1048575          3426
7       salesPrice   float64      1048575           273
8        salesDate    object      1048575            60
9           volume     int64      1048575            22
10  classification     int64      1048575             2
11       exciseTax   float64      1048575           468
12    vendorNumber     int64      1048575           116
13      vendorName    object      1048575           117
```

Figures 1.5.6 Purchase Price Table

The figure 1.5.6 shows the data schema of sales table. The tables have 14 columns and data type of string, integer and float.

2.0 ARCHITECTURE AND ETL PIPELINE

2.1 Pipeline Structure

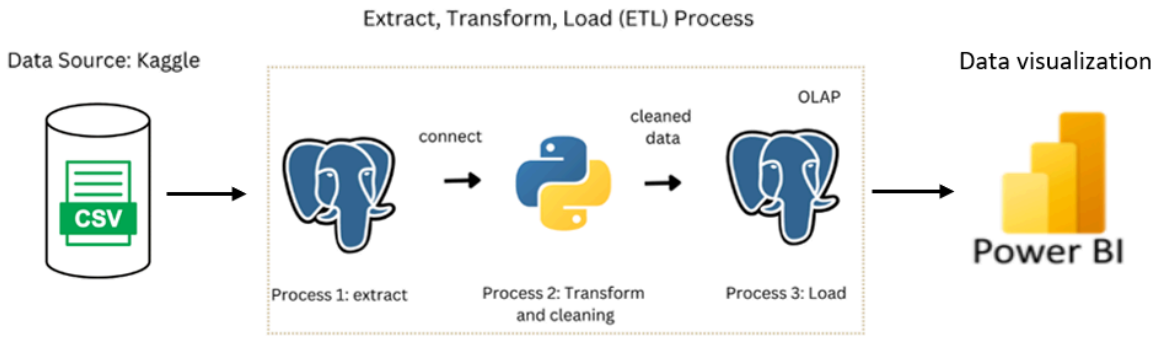


Figure 2.1.1 Pipeline Structure

The data set for this project consists of multiple tables, including Beginning Inventory, Ending Inventory, Invoice, Purchase, Purchase Price, and Sales. These tables were sourced from the company's internal database. The tables were then imported into PostgreSQL for centralized data management and querying. After successfully importing all the tables into PostgreSQL, the following process connects Python to Jupyter Notebook. Libraries such as psycopg2 and SQLAlchemy must be installed first to extract data from PostgreSQL.

The data integration process combines all the tables to provide a unified, single-data view. To improve the view and facilitate the analysis steps, OLAP operations like slicing, roll-up, and dicing were performed on the cleaned data. After the OLAP operations, the results are imported into Power BI for visualization. Various studies are performed in Power BI to observe needed outcomes.

The last step would be the analysis of those visualizations and report generation. In turn, this step would be able to identify critical trends and patterns that go into recommendations for optimization of inventory. The reports will also identify the impact the project has on sustainability and align with the goals of responsible consumption and production arising from SDG12. The pipeline will lead to a systematic acquisition of data, cleaning, integration, OLAP operations, and visualization, which when implemented supports the all-encompassing analytical and decision operations for inventory management optimization.

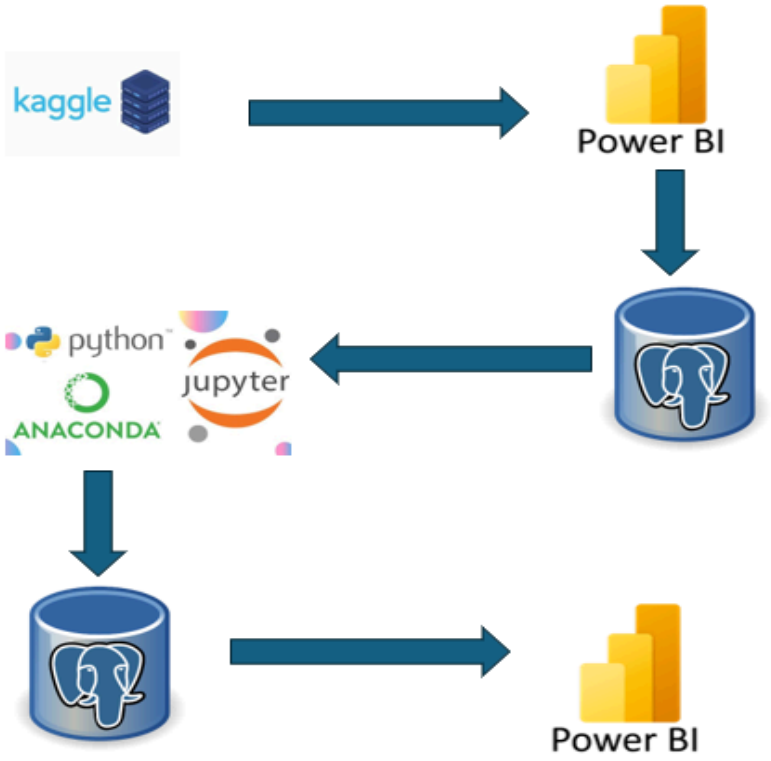


Figure 2.1.2 Project Structure

1. The data source is from kaggle in csv format.
2. Next the dataset import in Power BI to check relational model between dataset by table
3. Import data in PostgreSQL by table with six tables.
4. Next, import the dataset by table in Jupyter for the cleaning process.
5. Download the cleaned dataset from Jupyter and import it again in PostgreSQL for the OLAP (Online Analytical Process) process.
6. Import a cleaned dataset in Power BI again to check the relational model between all tables and then create data visualization.

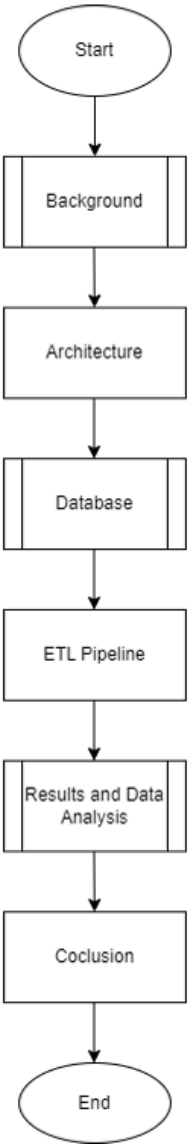


Figure 2.1.3 Flow of this project

Figure 2.1.3 shows the overall procedure of the project, which our project will complete in six stages.

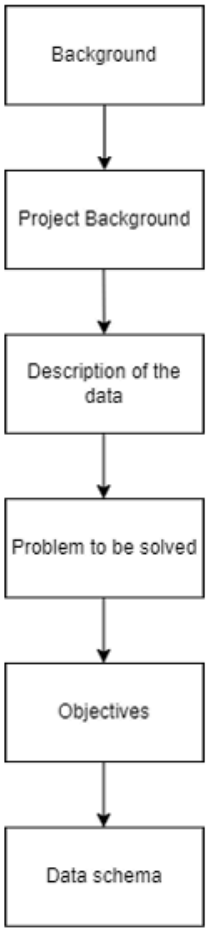


Figure 2.1.4 Process of the background

Figure 2.1.4 shows the background process, which covers the project's background, the data's description, the problem to be solved, the goals of the project, and the data schema. The Information used in this project was obtained through Kaggle.The architecture was thus developed to ensure that project execution would go smoothly as planned.

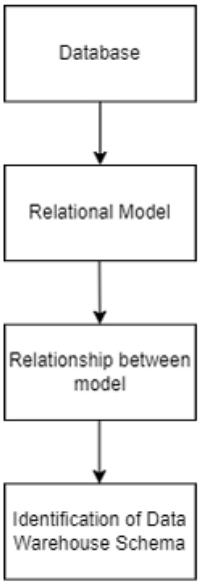


Figure 2.1.5 Process of Database

Figure 2.1.5 the database process, including the relational model and the relationships among the model and identification of the data warehouse schema. In this project, our team utilizes Microsoft Power BI and pgAdmin to make the relational model. After this, the process goes through the Extract, ETL pipeline by using Jupyter Notebook and pgAdmin. The raw data will be extracted into the Jupyter Notebook by doing data cleaning and merging. We then load the Deduplicate data to the pgAdmin for the OLAP process, Tableau, and Power BI for the data visualization.

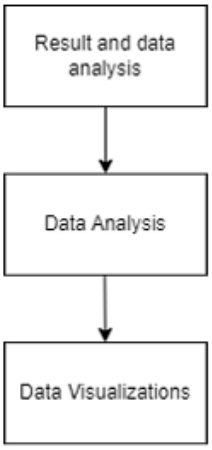


Figure 2.1.6 Process for the results and data analysis

Figure 2.1.6 presents the results and data analysis, which also includes data visualization. We would use pgAdmin to perform the data analysis functions such as roll-up and also slicing. After that we have been using Microsoft Power BI to finish the data visualization. Based on the analysis of data. Based on the results of these visualizations, we will draw a conclusion.

3.0 ETL PIPELINE

3.1 ETL Pipeline

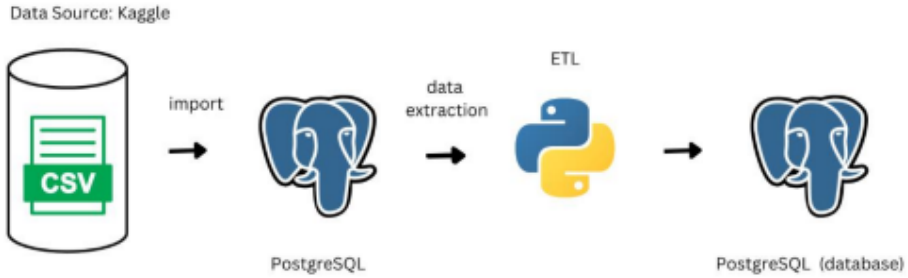


Figure 2.2.1 ETL pipeline

The data pipeline is the backbone of the data integration process that data flows from different sources to a central database. This project has three stages extraction, transformation and loading. Extraction phase involves pulling out data from various sources such as CSV files, databases and API responses. Data is in the form of Pandas Dataframes representing structured datasets. Data is then transformed to fit the schema of the destination database as well as improve its quality. Loading stage includes inserting the extracted data into PostgreSQL database using the 'load' function which connects using SQLAchemy and creates temporary tables for incoming data.The function takes care of any errors during the loading process. The ETL Pipeline serves as an efficient method for transferring software. It can do extraction and manipulation with Pandas library and connectivity/management with SQLAlchemy.

3.2 ETL Process

3.2.1 Extract

Before starting the ETL process the datasets need to be stored in a database in PostgreSQL. Firstly, create a new database and use database connectors to fetch the relevant data from each table.

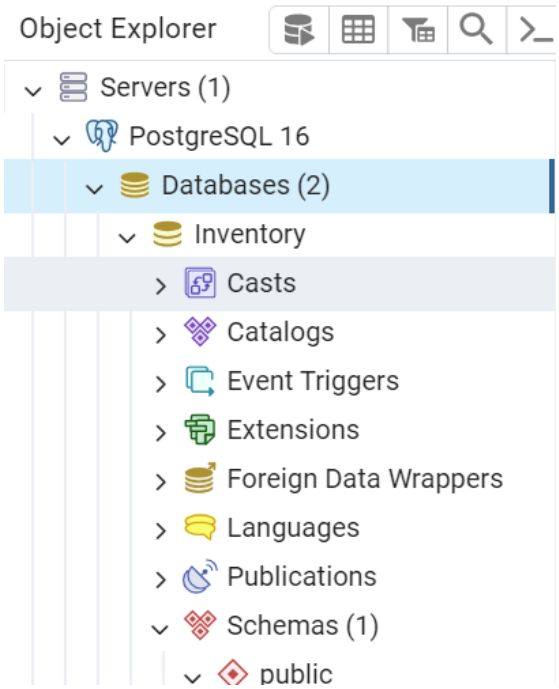


Figure 2.3.1 Database in PostgreSQL

Figure 2.3.1 shows that we have created a database named ‘Inventory’ in PostgreSQL.

Import csv file into table

	PID	Type	Server	Object	Start Time	Status	Time Taken (sec)
	14664	Import Data	PostgreSQL 16 (localhost:54...	Food Price/public.sales	04/06/2024, 04:01:39	Finished	8.05
	1688	Import Data	PostgreSQL 16 (localhost:54...	Food Price/public.purchase_...	04/06/2024, 03:27:38	Finished	0.19
	15916	Import Data	PostgreSQL 16 (localhost:54...	Food Price/public.purchase	04/06/2024, 03:15:01	Finished	12.53
	17116	Import Data	PostgreSQL 16 (localhost:54...	Food Price/public.invoice	04/06/2024, 03:14:48	Finished	0.18
	25468	Import Data	PostgreSQL 16 (localhost:54...	Food Price/public.end_inv	04/06/2024, 03:14:26	Finished	3.35
	19440	Import Data	PostgreSQL 16 (localhost:54...	Food Price/public.beg_inv	04/06/2024, 03:14:12	Finished	1.21

Figure 2.3.2 Import Files

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

SELECT * FROM beg_inv

```
SELECT * FROM beg_inv
```

Data Output | Messages | Notifications

	inventoryid text	store integer	city text	brand integer	description text	size text	onhand integer	price numeric	startdate timestamp witho
1	1_HARDERSFIELD_58	1	HARDERSFIELD	58	Gekkeikan Black & Gold Sake	750mL	8	12.99	2016-01-01 00:0
2	1_HARDERSFIELD_60	1	HARDERSFIELD	60	Canadian Club 1858 VAP	750mL	7	10.99	2016-01-01 00:0
3	1_HARDERSFIELD_62	1	HARDERSFIELD	62	Herradura Silver Tequila	750mL	6	36.99	2016-01-01 00:0
4	1_HARDERSFIELD_63	1	HARDERSFIELD	63	Herradura Reposado Tequila	750mL	3	38.99	2016-01-01 00:0
5	1_HARDERSFIELD_72	1	HARDERSFIELD	72	No. 3 London Dry Gin	750mL	6	34.99	2016-01-01 00:0
6	1_HARDERSFIELD_75	1	HARDERSFIELD	75	Three Olives Tomato Vodka	750mL	18	14.99	2016-01-01 00:0
7	1_HARDERSFIELD_77	1	HARDERSFIELD	77	Three Olives Espresso Vodka	750mL	7	14.99	2016-01-01 00:0
8	1_HARDERSFIELD_79	1	HARDERSFIELD	79	Three Olives Loopy Vodka	750mL	2	14.99	2016-01-01 00:0
9	1_HARDERSFIELD_115	1	HARDERSFIELD	115	Belvedere Vodka	Liter	5	27.99	2016-01-01 00:0
10	1_HARDERSFIELD_120	1	HARDERSFIELD	120	Tarantula Azul Tequila Gift	750mL	11	13.99	2016-01-01 00:0
11	1_HARDERSFIELD_126	1	HARDERSFIELD	126	Grey Goose Vodka	Liter	17	29.99	2016-01-01 00:0
12	1_HARDERSFIELD_165	1	HARDERSFIELD	165	Gentleman Jack Gift Pack	750mL	0	26.99	2016-01-01 00:0
13	1_HARDERSFIELD_171	1	HARDERSFIELD	171	Gentleman Jack	1.75L	12	49.99	2016-01-01 00:0

SELECT * FROM end_inv

```
SELECT * FROM end_inv
```

Data Output | Messages | Notifications

	inventoryid text	store integer	city text	brand integer	description text	size text	onhand integer	price numeric	enddate timestamp with
1	1_HARDERSFIELD_58	1	HARDERSFIELD	58	Gekkeikan Black & Gold Sake	750mL	11	12.99	2016-12-31 00
2	1_HARDERSFIELD_62	1	HARDERSFIELD	62	Herradura Silver Tequila	750mL	7	36.99	2016-12-31 00
3	1_HARDERSFIELD_63	1	HARDERSFIELD	63	Herradura Reposado Tequila	750mL	7	38.99	2016-12-31 00
4	1_HARDERSFIELD_72	1	HARDERSFIELD	72	No. 3 London Dry Gin	750mL	4	34.99	2016-12-31 00
5	1_HARDERSFIELD_75	1	HARDERSFIELD	75	Three Olives Tomato Vodka	750mL	7	14.99	2016-12-31 00
6	1_HARDERSFIELD_77	1	HARDERSFIELD	77	Three Olives Espresso Vodka	750mL	18	14.99	2016-12-31 00
7	1_HARDERSFIELD_79	1	HARDERSFIELD	79	Three Olives Loopy Vodka	750mL	7	14.99	2016-12-31 00
8	1_HARDERSFIELD_115	1	HARDERSFIELD	115	Belvedere Vodka	Liter	35	27.99	2016-12-31 00
9	1_HARDERSFIELD_126	1	HARDERSFIELD	126	Grey Goose Vodka	Liter	36	29.99	2016-12-31 00
10	1_HARDERSFIELD_159	1	HARDERSFIELD	159	Glenmorangie Original VAP	750mL + 2/	8	34.99	2016-12-31 00
11	1_HARDERSFIELD_171	1	HARDERSFIELD	171	Gentleman Jack	1.75L	24	49.99	2016-12-31 00
12	1_HARDERSFIELD_175	1	HARDERSFIELD	175	1800 Anejo Tequila	750mL	10	41.99	2016-12-31 00
13	1_HARDERSFIELD_178	1	HARDERSFIELD	178	W Turkey Russell's RSV 10 Yr	750mL	6	26.99	2016-12-31 00
14	1_HARDERSFIELD_192	1	HARDERSFIELD	192	Milagro Anejo Tequila	750mL	5	35.99	2016-12-31 00
15	1_HARDERSFIELD_200	1	HARDERSFIELD	200	Luxardo Maraschino Liqueur	750mL	3	29.99	2016-12-31 00

SELECT * FROM invoice

```
SELECT * FROM invoice
```

Data Output | Messages | Notifications

	vendornumber integer	vendorname text	invoicedate timestamp without time zone	ponumber integer	podate timestamp without time zone	paydate timestamp without tim
1	105	ALTAMAR BRANDS LLC	2016-01-04 00:00:00	8124	2015-12-21 00:00:00	2016-02-16 00:00:00
2	4466	AMERICAN VINTAGE BEVERAGE	2016-01-07 00:00:00	8137	2015-12-22 00:00:00	2016-02-21 00:00:00
3	388	ATLANTIC IMPORTING COMPANY	2016-01-09 00:00:00	8169	2015-12-24 00:00:00	2016-02-16 00:00:00
4	480	BACARDI USA INC	2016-01-12 00:00:00	8106	2015-12-20 00:00:00	2016-02-05 00:00:00
5	516	BANFI PRODUCTS CORP	2016-01-07 00:00:00	8170	2015-12-24 00:00:00	2016-02-12 00:00:00
6	2396	BLACK PRINCE DISTILLERY INC	2016-01-08 00:00:00	8191	2015-12-25 00:00:00	2016-02-06 00:00:00
7	1128	BROWN-FORMAN CORP	2016-01-09 00:00:00	8150	2015-12-23 00:00:00	2016-02-19 00:00:00
8	1189	BULLY BOY DISTILLERS	2016-01-09 00:00:00	8171	2015-12-24 00:00:00	2016-02-04 00:00:00
9	1273	CALEDONIA SPIRITS INC	2016-01-06 00:00:00	8172	2015-12-24 00:00:00	2016-02-15 00:00:00
10	11567	CAMPARI AMERICA	2016-01-06 00:00:00	8151	2015-12-23 00:00:00	2016-02-20 00:00:00
11	90046	CANDIA VINEYARDS	2016-01-05 00:00:00	8107	2015-12-20 00:00:00	2016-02-10 00:00:00
12	1485	CASTLE BRANDS CORP.	2016-01-08 00:00:00	8152	2015-12-23 00:00:00	2016-02-19 00:00:00
13	2876	CENTEUR IMPORTS LLC	2016-01-08 00:00:00	8125	2015-12-21 00:00:00	2016-02-07 00:00:00
14	4380	CHARLES JACQUIN ET CIE INC	2016-01-06 00:00:00	8153	2015-12-23 00:00:00	2016-02-12 00:00:00
15	1392	CONSTELLATION BRANDS INC	2016-01-11 00:00:00	8108	2015-12-20 00:00:00	2016-02-10 00:00:00

SELECT * FROM purchase

```
SELECT * FROM purchase
```

Data Output | Messages | Notifications

	inventoryid text	store integer	brand integer	description text	size text	vendornumber integer	vendorname text	ponumb integer
1	69_MOUNTMEND_8412	69	8412	Tequila Ocho Plata Fresno	750mL	105	ALTAMAR BRANDS LLC	
2	30_CULCHETH_5255	30	5255	TGI Fridays Ultimte Mudslide	1.75L	4466	AMERICAN VINTAGE BEVERAG...	
3	34_PITMERDEN_5215	34	5215	TGI Fridays Long Island Iced	1.75L	4466	AMERICAN VINTAGE BEVERAG...	
4	1_HARDERSFIELD_5255	1	5255	TGI Fridays Ultimte Mudslide	1.75L	4466	AMERICAN VINTAGE BEVERAG...	
5	76_DONCASTER_2034	76	2034	Glendalough Double Barrel	750mL	388	ATLANTIC IMPORTING COMPA...	
6	5_SUTTON_3348	5	3348	Bombay Sapphire Gin	1.75L	480	BACARDI USA INC	
7	1_HARDERSFIELD_8358	1	8358	Bacardi 151 Proof	750mL	480	BACARDI USA INC	
8	30_CULCHETH_4903	30	4903	Bacardi Superior Rum	200mL	480	BACARDI USA INC	
9	34_PITMERDEN_3782	34	3782	Grey Goose Le Citron Vodka	750mL	480	BACARDI USA INC	
10	1_HARDERSFIELD_4233	1	4233	Castillo Silver Label Rum	1.75L	480	BACARDI USA INC	
11	1_HARDERSFIELD_3830	1	3830	Grey Goose L'Orange Vodka	750mL	480	BACARDI USA INC	
12	78_EASTHAVEN_2628	78	2628	Dewars Special RSV 12-Yr	750mL	480	BACARDI USA INC	
13	30_CULCHETH_4196	30	4196	Bacardi Dragon Berry Rum	750mL	480	BACARDI USA INC	
14	22_SHARNWICK_3830	22	3830	Grey Goose L'Orange Vodka	750mL	480	BACARDI USA INC	
15	38_GOULCREST_8358	38	8358	Bacardi 151 Proof	750mL	480	BACARDI USA INC	

SELECT * FROM purchase_price

```
SELECT * FROM purchase_price
```

Data Output | Messages | Notifications

	brand integer	description text	price numeric	size text	volume numeric	classification integer	purchaseprice numeric	vendornumber integer	vendorname text
1	2993	Angostura Bitters	7.49	[null]	[null]	1	5.39	5895	Mizkan Americas, Inc
2	9908	Tito's Copper Mug 2 Pack	21.01	[null]	[null]	1	16.15	4425	MARTIGNETTI COMP
3	8992	Group 92	1.99	[null]	[null]	1	1.43	1703	ALISA CARR BEVERA
4	90590	Overture Champagne 2Glas...	19.95	[null]	[null]	2	13.12	4425	MARTIGNETTI COMP
5	25457	GH Mumm Cordon Rouge	472.49	9000mL	9000	2	308.82	17035	PERNOD RICARD USA
6	18633	Viberti Mixed Wooden Box 6	347.99	750mL 6 Pk	750	2	221.47	9165	ULTRA BEVERAGE CO
7	24553	Ch Lassegue 2/05-09-10 6 ...	467.99	750mL 6 Pk	750	2	301.93	9552	M S WALKER INC
8	1009	Rebel Yell Variety Pack	49.99	750mL 3 Pk	750	1	38.75	8352	LUXCO INC
9	26671	Catena Zapata 3pk 09-10-1...	359.99	750mL 3 Pk	750	2	244.89	9552	M S WALKER INC
10	27626	LaBelle Winery Holiday 3 Pak	40.99	750mL 3 Pk	750	2	27.7	90032	LABELLE VYDS AND
11	3341	Bombay Sapphire & East 2 ...	34.99	750mL 2 Pk	750	1	25.17	480	BACARDI USA INC
12	4881	Bacardi Twin Pack 2/750mls	19.99	750mL 2 Pk	750	1	14.81	480	BACARDI USA INC
13	22800	Alexander Vly Wicked Week...	15.99	750mL 2 Pk	750	2	10.59	7153	PINE STATE TRADING
14	4535	Bacardi Limon & Mango 2 P...	19.99	750mL 2 Pk	750	1	14.49	480	BACARDI USA INC
15	2502	Dewars Wh Label & Honey 2...	34.99	750mL 2 Pk	750	1	26.92	480	BACARDI USA INC

SELECT * FROM sales

```
SELECT * FROM sales
```

Data Output | Messages | Notifications

	inventoryid text	store integer	brand integer	description text	size text	salesquantity integer	salesdollars numeric	salesprice numeric	salesdate timestamp
1	1_HARDERSFIELD_1004	1	1004	Jim Beam w/2 Rocks Glass...	750mL	1	16.49	16.49	2016-01-0
2	1_HARDERSFIELD_1004	1	1004	Jim Beam w/2 Rocks Glass...	750mL	2	32.98	16.49	2016-01-0
3	1_HARDERSFIELD_1004	1	1004	Jim Beam w/2 Rocks Glass...	750mL	1	16.49	16.49	2016-01-0
4	1_HARDERSFIELD_1004	1	1004	Jim Beam w/2 Rocks Glass...	750mL	1	14.49	14.49	2016-01-0
5	1_HARDERSFIELD_1005	1	1005	Maker's Mark Combo Pack	375mL 2 Pk	2	69.98	34.99	2016-01-0
6	1_HARDERSFIELD_1005	1	1005	Maker's Mark Combo Pack	375mL 2 Pk	1	34.99	34.99	2016-01-1
7	1_HARDERSFIELD_1005	1	1005	Maker's Mark Combo Pack	375mL 2 Pk	1	34.99	34.99	2016-01-2
8	1_HARDERSFIELD_1005	1	1005	Maker's Mark Combo Pack	375mL 2 Pk	1	34.99	34.99	2016-01-3
9	1_HARDERSFIELD_10058	1	10058	F Coppola Dmd Ivry Cab Svgn	750mL	4	59.96	14.99	2016-01-0
10	1_HARDERSFIELD_10058	1	10058	F Coppola Dmd Ivry Cab Svgn	750mL	1	14.99	14.99	2016-01-0
11	1_HARDERSFIELD_10058	1	10058	F Coppola Dmd Ivry Cab Svgn	750mL	1	14.99	14.99	2016-01-0
12	1_HARDERSFIELD_10058	1	10058	F Coppola Dmd Ivry Cab Svgn	750mL	1	14.99	14.99	2016-01-1
13	1_HARDERSFIELD_10058	1	10058	F Coppola Dmd Ivry Cab Svgn	750mL	1	14.99	14.99	2016-01-1
14	1_HARDERSFIELD_10058	1	10058	F Coppola Dmd Ivry Cab Svgn	750mL	3	44.97	14.99	2016-01-2
15	1_HARDERSFIELD_10058	1	10058	F Coppola Dmd Ivry Cab Svgn	750mL	2	29.98	14.99	2016-01-2

Steps to extract the dataset using Jupyter Notebook

Before starting the process, we are required to install a few packages.

- `! pip install ipython-sql`
- `! pip install sqlalchemy`
- `! pip install psycopg2`
- `! pip install python-sql`
- `! pip install pandas-sql`
- `! pip install sql-queries`

After installing all these packages, we need to load ipython-sql using the following command:

```
[75]: %reload_ext sql
```

Figure 2.4.1 Load ipython-sql

Then, Call the create engine function:

```
[73]: from sqlalchemy import create_engine
```

Figure 2.4.2 Create engine function

Next, import the necessary libraries for the ETL process:

```
[74]: import psycopg2
      import pandas as pd
```

Figure 2.4.3 Import Libraries

```
[76]: conn = psycopg2.connect(
      dbname = 'Inventory',
      user = 'postgres',
      password = '1234',
      host = 'localhost' ,
      port = '5432'
      )
```

Figure 2.4.4 Connect the PgAdmin with Jupyter Notebook

Extract Data from pgAdmin to Jupyter Notebook

Coding

```
[124]: query = "SELECT * FROM beg_inv;"

[125]: df=sqlio.read_sql_query(query,conn)
       df
```

Output

[125]:

	inventoryid	store	city	brand	description	size	onhand	price	startdate
0	1_HARDERSFIELD_58	1	HARDERSFIELD	58	Gekkeikan Black & Gold Sake	750mL	8	12.99	2016-01-01
1	1_HARDERSFIELD_60	1	HARDERSFIELD	60	Canadian Club 1858 VAP	750mL	7	10.99	2016-01-01
2	1_HARDERSFIELD_62	1	HARDERSFIELD	62	Herradura Silver Tequila	750mL	6	36.99	2016-01-01
3	1_HARDERSFIELD_63	1	HARDERSFIELD	63	Herradura Reposado Tequila	750mL	3	38.99	2016-01-01
4	1_HARDERSFIELD_72	1	HARDERSFIELD	72	No. 3 London Dry Gin	750mL	6	34.99	2016-01-01
...	...	...	...	...	...	...	...	...	...
206524	79_BALLYMENA_46985	79	BALLYMENA	46985	Rodney Strong Cab Svgn Alexa	750mL	13	22.99	2016-01-01
206525	79_BALLYMENA_47014	79	BALLYMENA	47014	Juan Gil Jumilla Rd	750mL	13	13.99	2016-01-01
206526	79_BALLYMENA_47090	79	BALLYMENA	47090	Napa Cellars Cab Svgn Napa	750mL	19	23.99	2016-01-01
206527	79_BALLYMENA_90011	79	BALLYMENA	90011	Ch Pichon Longville 12 Pauil	750mL	12	144.99	2016-01-01
206528	79_BALLYMENA_90089	79	BALLYMENA	90089	Ch Lynch Bages 12 Pauillac	750mL	24	119.99	2016-01-01

206529 rows × 9 columns

```
[128]: query1 = "SELECT * FROM end_inv;"

[129]: df1=sqlio.read_sql_query(query1,conn)
       df1
```

[129]:

	inventoryid	store	city	brand	description	size	onhand	price	enddate
0	1_HARDERSFIELD_58	1	HARDERSFIELD	58	Gekkeikan Black & Gold Sake	750mL	11	12.99	2016-12-31
1	1_HARDERSFIELD_62	1	HARDERSFIELD	62	Herradura Silver Tequila	750mL	7	36.99	2016-12-31
2	1_HARDERSFIELD_63	1	HARDERSFIELD	63	Herradura Reposado Tequila	750mL	7	38.99	2016-12-31
3	1_HARDERSFIELD_72	1	HARDERSFIELD	72	No. 3 London Dry Gin	750mL	4	34.99	2016-12-31
4	1_HARDERSFIELD_75	1	HARDERSFIELD	75	Three Olives Tomato Vodka	750mL	7	14.99	2016-12-31
...	...	...	...	...	...	...	...	...	...
224484	81_PEMBROKE_90087	81	PEMBROKE	90087	Ch Mouton Rothschild 12 Paui	750mL	3	469.99	2016-12-31
224485	81_PEMBROKE_90088	81	PEMBROKE	90088	Ch Le Petite Mouton 12 Pauil	750mL	3	134.99	2016-12-31
224486	81_PEMBROKE_90089	81	PEMBROKE	90089	Ch Lynch Bages 12 Pauillac	750mL	3	119.99	2016-12-31
224487	81_PEMBROKE_90090	81	PEMBROKE	90090	Ch Lafite Rothschild 12	750mL	3	649.99	2016-12-31
224488	81_PEMBROKE_90604	81	PEMBROKE	90604	Ch Lynch Bages Pauilac	750mL	2	119.99	2016-12-31

224489 rows × 9 columns

```
[132]: query2 = "SELECT * FROM invoice;"

[133]: df2=sqlio.read_sql_query(query2,conn)
       df2
```

[133]:

	vendornumber	vendorname	invoicedate	ponumber	podate	paydate	quantity	dollars	freight	approval
0	105	ALTAMAR BRANDS LLC	2016-01-04	8124	2015-12-21	2016-02-16	6	214.26	3.47	None
1	4466	AMERICAN VINTAGE BEVERAGE	2016-01-07	8137	2015-12-22	2016-02-21	15	140.55	8.57	None
2	388	ATLANTIC IMPORTING COMPANY	2016-01-09	8169	2015-12-24	2016-02-16	5	106.60	4.61	None
3	480	BACARDI USA INC	2016-01-12	8106	2015-12-20	2016-02-05	10100	137483.78	2935.20	None
4	516	BANFI PRODUCTS CORP	2016-01-07	8170	2015-12-24	2016-02-12	1935	15527.25	429.20	None
...	...	...	...	...	...	...	...	...	...	...
5538	9622	WEIN BAUER INC	2017-01-06	13626	2016-12-21	2017-02-10	90	1563.00	8.60	None
5539	9625	WESTERN SPIRITS BEVERAGE CO	2017-01-10	13661	2016-12-23	2017-02-18	4617	37300.48	186.50	None
5540	3664	WILLIAM GRANT & SONS INC	2017-01-02	13643	2016-12-22	2017-02-04	9848	202815.78	932.95	None
5541	9815	WINE GROUP INC	2017-01-03	13602	2016-12-20	2017-02-08	24747	149007.56	819.54	None
5542	90058	ZORVINO VINEYARDS	2017-01-05	13574	2016-12-18	2017-02-12	437	3608.11	16.60	None

5543 rows × 10 columns

```
[139]: query3 = "SELECT * FROM purchase;"
       df3=sqlio.read_sql_query(query3,conn)
       df3
```

[149]:

	inventoryid	store	brand	description	size	vendornumber	vendorname	ponumber	podate	receivingdate	invoicedate	paydate	purchasepric
0	69_MOUNTMEND_8412	69	8412	Tequila Ocho Plata Fresno	750mL	105	ALTAMAR BRANDS LLC	8124	2015-12-21	2016-01-02	2016-01-04	2016-02-16	35.7
1	30_CULCHETH_5255	30	5255	TGI Fridays Ultimte Mudslide	1.75L	4466	AMERICAN VINTAGE BEVERAGE	8137	2015-12-22	2016-01-01	2016-01-07	2016-02-21	9.3
2	34_PITMERDEN_5215	34	5215	TGI Fridays Long Island Iced	1.75L	4466	AMERICAN VINTAGE BEVERAGE	8137	2015-12-22	2016-01-02	2016-01-07	2016-02-21	9.4
3	1_HARDERSFIELD_5255	1	5255	TGI Fridays Ultimte Mudslide	1.75L	4466	AMERICAN VINTAGE BEVERAGE	8137	2015-12-22	2016-01-01	2016-01-07	2016-02-21	9.3
4	76_DONCASTER_2034	76	2034	Glendalough Double Barrel	750mL	388	ATLANTIC IMPORTING COMPANY	8169	2015-12-24	2016-01-02	2016-01-09	2016-02-16	21.3
...	...	...	...	...	...	...	...	...	...	...	...	...	...
1048570	74_PAENTMARWY_8450	74	8450	E & J Brandy VS	1.75L	3252	E & J GALLO WINERY	10781	2016-06-22	2016-06-27	2016-07-07	2016-08-08	13.8
1048571	53_HILLFAR_7374	53	7374	New Amsterdam Mango Vodka	750mL	3252	E & J GALLO WINERY	10781	2016-06-22	2016-06-29	2016-07-07	2016-08-08	6.8

[141]:
```
query4 = "SELECT * FROM purchase_price;"
df4=sqlio.read_sql_query(query4,conn)
df4
```

[141]:

	brand	description	price	size	volume	classification	purchaseprice	vendornumber	vendorname
0	2993	Angostura Bitters	7.49	None	NaN	1	5.39	5895	Mizkan Americas, Inc.
1	9908	Tito's Copper Mug 2 Pack	21.01	None	NaN	1	16.15	4425	MARTIGNETTI COMPANIES
2	8992	Group 92	1.99	None	NaN	1	1.43	1703	ALISA CARR BEVERAGES
3	90590	Overture Champagne 2Glass Pk	19.95	None	NaN	2	13.12	4425	MARTIGNETTI COMPANIES
4	25457	GH Mumm Cordon Rouge	472.49	9000mL	9000.0	2	308.82	17035	PERNOD RICARD USA
...	...	...	...	...	...	...	...	...	...
12256	15775	Paul D Gruner Veltliner	14.49	1000mL	1000.0	2	9.35	90047	CRUSH WINES
12257	25221	Gnarly Head OV Zinfandel	9.99	1000mL	1000.0	2	6.89	2242	DELICATO VINEYARDS INC
12258	26645	Darting Rsl Kab	16.99	1000mL	1000.0	2	11.72	10754	PERFECTA WINES
12259	27440	Achaia Cellars Retsina	9.49	1000mL	1000.0	2	6.37	12331	STELLAR IMPORTING CO LLC
12260	4202	None	0.00	None	NaN	1	11.19	480	BACARDI USA INC

12261 rows × 9 columns

[143]:
```
query5 = "SELECT * FROM sales;"
df5=sqlio.read_sql_query(query5,conn)
df5
```

[143]:

	inventoryid	store	brand	description	size	salesquantity	salesdollars	salesprice	salesdate	volume	classification	excisetax	vendornumber	ve
0	1_HARDERSFIELD_1004	1	1004	Jim Beam w/2 Rocks Glasses	750mL	1	16.49	16.49	2016-01-01	750	1	0.79	12546	
1	1_HARDERSFIELD_1004	1	1004	Jim Beam w/2 Rocks Glasses	750mL	2	32.98	16.49	2016-01-02	750	1	1.57	12546	
2	1_HARDERSFIELD_1004	1	1004	Jim Beam w/2 Rocks Glasses	750mL	1	16.49	16.49	2016-01-03	750	1	0.79	12546	
3	1_HARDERSFIELD_1004	1	1004	Jim Beam w/2 Rocks Glasses	750mL	1	14.49	14.49	2016-01-08	750	1	0.79	12546	
4	1_HARDERSFIELD_1005	1	1005	Maker's Mark Combo Pack	375mL 2 Pk	2	69.98	34.99	2016-01-09	375	1	0.79	12546	
...	...	...	...	...	...	...	...	...	...	...	...	...	...	
1048570	19_WINTERVALE_39384	19	39384	F Coppola Diamond Pnt Nr	750mL	4	51.80	12.95	2016-02-12	750	2	0.45	2000	
1048571	19_WINTERVALE_39384	19	39384	F Coppola Diamond	750mL	8	103.60	12.95	2016-02-13	750	2	0.90	2000	

3.2.2 Transform

After extracting the dataset using Jupyter Notebook we do the Data transformation in the data analysis pipeline. It involves converting raw data into a format that is suitable for analysis. This process ensures that the data is clean, consistent, and ready for further processing or modeling.

We can start the cleaning process by checking the missing and duplicates values first:

```
[212]: dfs = [df, df1, df2, df3, df4, df5]
       missing_values_per_df = [df.isnull().sum() for df in dfs]
       for i, missing_values in enumerate(missing_values_per_df):
           print(f"Missing values in df{i}:")
           print(missing_values)
           print("\n")
```

Figure 2.4.1.1 Checking Missing Value

```
[194]: dfs = [df1, df2, df3, df4, df5]
       duplicate_values = [df.duplicated().sum() for df in dfs]

       for i, duplicate_values in enumerate(duplicate_values, start=1):
           print(f"Duplicate values in df{i}:")
           print(duplicate_values)
           print("\n")
```

Figure 2.4.1.2 Checking Duplicate Value

```
Duplicate values in df0:
0


Duplicate values in df1:
0


Duplicate values in df2:
0


Duplicate values in df3:
0


Duplicate values in df4:
0


Duplicate values in df5:
0
```

There are no duplicate values in this data.
After we check for the missing and duplicate values, if the dataset contains the missing and duplicate values, drop the values and check whether the values have been dropped.

[264]:
```
# Handling missing values for end_inv dataset
df1["city"] = df1["city"].fillna("TYWARDREATH")
# Handling missing values for invoice dataset
df2 = df2.drop(['approval'], axis=1)
# Handling missing values for purchase_prices dataset
cols_to_check = ['description', 'size', 'volume']
for col in cols_to_check:
    df4 = df4[df4[col].notna()]
```

Figure 2.4.1.3 Handle Missing Values

3.2.3 Load

Before we load the data, we need to check for missing values in several pandas DataFrames, each representing a different inventory-related table.It iterates through each DataFrame, calculates the amount of missing values per column, and outputs a summary that shows which columns have missing values and how many.If a DataFrame contains no missing value, it prints a message, making it easy to discover and fix data quality concerns in individual datasets.

Checking Missing Values

[48]:
```
print("Summary of missing values")

#Get a summary of missing values for each table
datasets = [dframe1, dframe2, dframe3, dframe4, dframe5, dframe6]
dataset_names = ["Beginning Inventory Table", "Ending Inventory Table", "Purchases Table",
"Purchases Invoice Table", "Purchase Price Table", "Sales Table"]

for name, data in zip(dataset_names, datasets):
    missing_values = data.isnull().sum()
    non_zero_missing_values = missing_values[missing_values > 0]

    if not non_zero_missing_values.empty:
        print(f"\nMissing values in {name}:")
        print(non_zero_missing_values)
    else:
        print(f"\nNo missing values in {name}.")
```

Figure 2.5.1 Checking Missing Values

```
Summary of missing values

No missing values in Beginning Inventory Table.

No missing values in Ending Inventory Table.

No missing values in Purchases Table.

No missing values in Purchases Invoice Table.

No missing values in Purchase Price Table.

No missing values in Sales Table.
```

Figure 2.5.2 Outputs Missing Values

Then, we can identify any duplicate data in the DataFrame, then calculate the total number of these duplicate rows using the 'len' function and store this count in 'total_duplicates', which is the output.

```
[49]: duplicateRows = dframe6[dframe6.duplicated()]
      total_duplicates = len(duplicateRows)
      total_duplicates

[49]: 0
```

Figure 2.5.3 Checking Duplicates Values

The code saves six DataFrames ('dframe1' to 'dframe6') to CSV files named 'beg_inv2.csv' , 'end_inv2.csv' , 'invoice2.csv', 'purchase2.csv', 'purchase_price2.csv' and 'sales2.csv' without including the indices.

```
[50]: dframe1.to_csv('beg_inv2.csv', index=False)
      dframe2.to_csv('end_inv2.csv', index=False)
      dframe3.to_csv('invoice2.csv', index=False)
      dframe4.to_csv('purchase2.csv', index=False)
      dframe5.to_csv('purchase_price2.csv', index=False)
      dframe6.to_csv('sales2.csv', index=False)
```

Figure 2.5.4 Saves six DataFrames

Finally, we need to load the cleaned data back into a PostgreSQL table. We can use SQLAlchemy to manage this:

```
[51]: pip install pandas sqlalchemy psycopg2-binary

Requirement already satisfied: pandas in c:\users\alisyusri\anaconda3\lib\site-packages (2.1.4)
Requirement already satisfied: sqlalchemy in c:\users\alisyusri\anaconda3\lib\site-packages (2.0.25)
Requirement already satisfied: psycopg2-binary in c:\users\alisyusri\anaconda3\lib\site-packages (2.9.9)
Requirement already satisfied: numpy<2,>=1.23.2 in c:\users\alisyusri\anaconda3\lib\site-packages (from pandas) (1.26.4)
Requirement already satisfied: python-dateutil>=2.8.2 in c:\users\alisyusri\anaconda3\lib\site-packages (from pandas) (2.8.2)
Requirement already satisfied: pytz>=2020.1 in c:\users\alisyusri\anaconda3\lib\site-packages (from pandas) (2023.3.post1)
Requirement already satisfied: tzdata>=2022.1 in c:\users\alisyusri\anaconda3\lib\site-packages (from pandas) (2023.3)
Requirement already satisfied: typing-extensions>=4.6.0 in c:\users\alisyusri\anaconda3\lib\site-packages (from sqlalchemy) (4.9.0)
Requirement already satisfied: greenlet!=0.4.17 in c:\users\alisyusri\anaconda3\lib\site-packages (from sqlalchemy) (3.0.1)
Requirement already satisfied: six>=1.5 in c:\users\alisyusri\anaconda3\lib\site-packages (from python-dateutil>=2.8.2->pandas) (1.16.0)
Note: you may need to restart the kernel to use updated packages.
```

Figure 2.5.5 Install SQLAlchemy

Uses the ‘extract’ function to go over each DataFrame and its related table name before using the ‘load’ function to do the actual loading. The ‘load’ function connects to the database via SQLAlchemy, prints the range of rows being imported, and loads each DataFrame into a table prefixed with ‘stg_’ using the ‘to_sql’ method. Calling the ‘extract’ function initiates the entire process.It allows for the extraction of data from each DataFrame and loading it into a corresponding table in a PostgreSQL database.

[52]:
```
import pandas as pd
from sqlalchemy import create_engine

# List of dataframes and corresponding table names
dataframes = [dframe1, dframe2, dframe3, dframe4, dframe5, dframe6]
table_names = ['beg_inv2', 'end_inv2', 'invoice2', 'purchase2', 'purchase_price2', 'sales2']

def extract():
    try:
        for df, tbl in zip(dataframes, table_names):
            load(df, tbl)
    except Exception as e:
        print("Data extract error: " + str(e))

def load(df, tbl):
    try:
        rows_imported = 0
        engine = create_engine('postgresql://postgres:1234@localhost:5432/inventory')
        print(f'importing rows {rows_imported} to {rows_imported + len(df)}... for table {tbl}')
        # save df to postgres
        df.to_sql(f'stg_{tbl}', engine, if_exists='replace', index=False, chunksize=100000)
        rows_imported += len(df)
        # add elapsed time to final print out
        print("Data imported successful")
    except Exception as e:
        print("Data load error: " + str(e))
```

Figure 2.5.6 Loads the Data into PostgreSQL

4.0 DATABASE

4.1 Relational Model

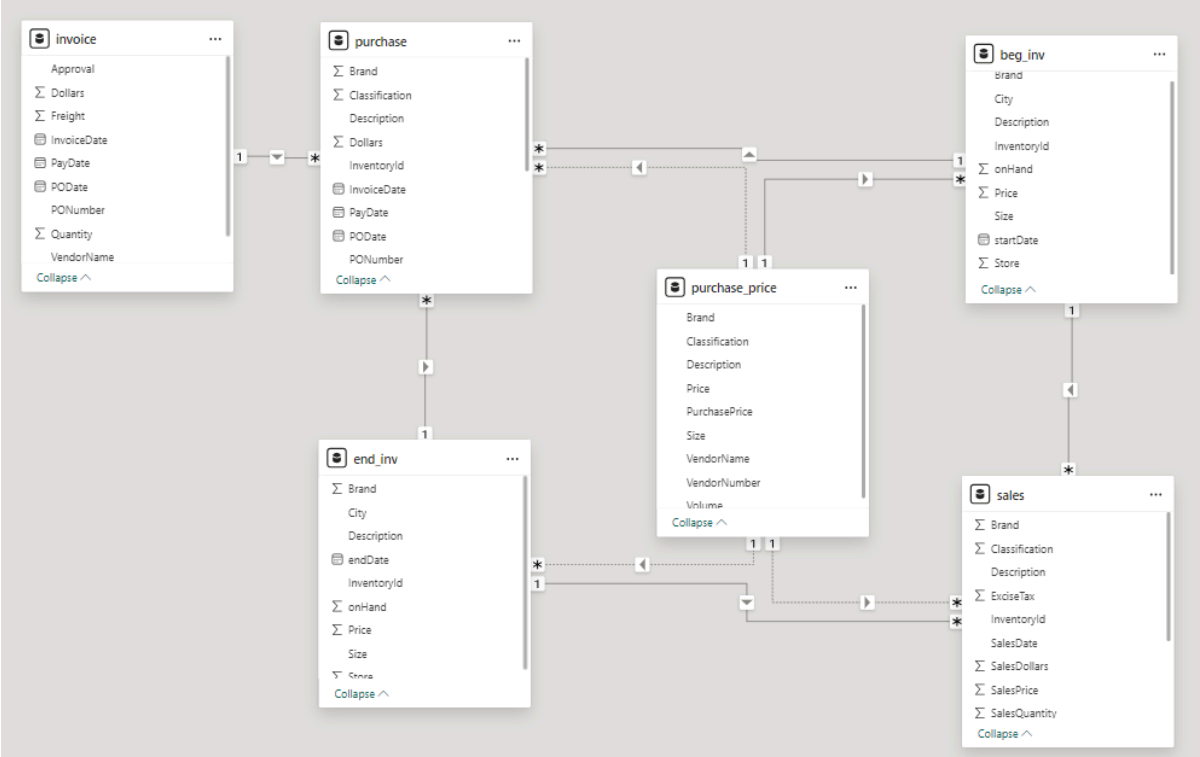


Figure 4.1 Relational Model using Power BI

Relational model using Power BI. There are six tables, sales, purchase, purchase price, invoice, beginning invoice, and ending invoice. All tables are relational models. Table purchase price is relational to all tables. This model effectively tracks inventory from purchase to sales, manages invoice details, and maintains accurate inventory records. The relational model which connects these tables through the key fields provides a well integrated data architecture for detailed analysis and reporting on sales, purchases and inventory positions. This is the systematic way that aids companies to observe their inventory lifecycle starting from the first purchase up to the ultimate sale so as to give very important points for decision-making and operational efficiency.

4.2 Relational Data

Data	Relationship
Beg_inv -> inventoryID	One to Many
End_inv -> inventoryID	One to Many
Invoice -> poNumber	One to Many
Purchase_price -> vendorNumber	One to Many
Sales -> inventoryID	One to Many
Purchase -> poNumber	Many to One
Purchase -> vendorNumber	Many to One

4.3 Data Warehouse Schema

Based on Figure 4.1 above, the data warehouse schema of these datasets is star Schema because it has three fact tables, and each fact table is connected to three dimensions table.This star schema includes fact tables for sales and purchases, and dimension tables for inventory, vendors, and products.

5.0 RESULTS AND DATA ANALYSIS

5.1 OLAP Coding

Cube

```
SELECT
  city,
  brand,
  AVG(price) AS AveragePrice
FROM stg_beg_inv2
GROUP BY city, brand;
```

Data Output | Messages | Notifications

	city text	brand bigint	averageprice double precision
1	HORNSEY	6081	1.99
2	ASHBORNE	23389	11.99
3	LEESIDE	36771	8.95
4	HORNSEY	98	14.49
5	STANMORE	15891	11.99
6	HORNSEY	2270	74.99
7	GOULCREST	22362	9.99
8	BLACKPOOL	46826	14.99
9	CARDEND	1896	99.99
10	CARDEND	25808	11.99
11	GOULCREST	3776	25.99
12	HORNSEY	34429	16.99
13	CARDEND	27735	9.99
14	HARDERSFIELD	37926	13.99

Figure 4.1.1 Cube

Analyzing average price by store or by brand within a city can provide valuable insights for businesses.Understanding these average prices you can have more of an insight into the overall strategies of pricing across the different locations or different brands. This knowledge enables you to spot outlets that within the same brand have larger margins or outlets that are charging really high prices compared to their neighboring stores. It shows in the figure 4.1.1 that Cardnend is the highest value of average price and Blackpool is the highest value of brand. Cardnend may have a higher average price because of market demand and affluence. The city can have better sales in the premiums or high-end bits and pieces that are generally more expensive. This is also because Blackpool may offer a more diverse clientele, so while some may find luxury leather goods satisfactory, other customers may prefer different brands for different prices. This could be because if a store is located in Blackpool, and if the area boasts a heavy tourist trade or high numbers of pedestrians, then the retailer is likely to have a higher number of brands on offer to appeal to as wide a demographic as possible. Thus, average price data are converted into useful information that enables informed decisions on promos, discounts, and campaigns, which mean efficient use of resources and overall business performance increase.

Roll up

```
SELECT brand, EXTRACT(MONTH FROM salesdate) AS month, SUM(salesquantity) AS total_quantity, SUM(salesdollars) AS total_sales
FROM stg_sales2
GROUP BY brand, EXTRACT(MONTH FROM salesdate);
```

Data Output | Messages | Notifications

	brand bigint	month numeric	total_quantity numeric	total_sales double precision
1	58	1	244	3169.5600000000013
2	58	2	44	571.5600000000001
3	60	1	120	1318.8000000000006
4	60	2	4	39.96
5	61	1	24	335.76
6	62	1	127	5257.729999999999
7	62	2	35	1294.65
8	63	1	106	4577.9400000000005
9	63	2	25	974.7500000000001
10	72	1	19	739.8100000000001
11	75	1	2	29.98
12	75	2	1	14.99

Figure 4.1.2 Roll Up

The SQL query created to aggregate sales data by brand and month, providing a summarized view of sales performance over time.The query chooses the ‘brand’ column, extracts the month from the ‘salesdate’ field, and calculate the total number of items sold (‘SUM(salesquantity)’) and total sales in dollars (‘SUM(salesdollars)’). It groups the results by ‘brand’ and the extracted month from ‘salesdate’ resulting in a single summary of sales activity for each brand across many months.

Consider a hypothetical result set. In January (month=1), Brand 58 sold 244 units, for a total of $3169.5600000000013. In February (month=2), Brand 58’s total quantity sold decreased to 44 units, with total sales reaching $571.560000000000. Similarly, in January, Brand 60 sold 120 units, totaling $1318.8000000000004, whereas in February, the total quantity sold was 4 units, totaling $39.96. In January, Brand 61 sold 24 units with total sales of $335.76. There were no sales in February.

This data contains several key insights. First, it highlights each brand’s monthly sales performance, showing how sales quantity and total sales change from month to month.For example, Brand 58’s sales decrease from January to February suggests poor performance. Second, the data allows a comparison of multiple brands within the same month, showing that in both January and February, Brand 58 beats Brands 60 and 61 in terms of units sold and total sales. Third, by analyzing these data, organizations can uncover patterns and make accurate predictions regarding future sales performance.For example, if Brand 58’s sales continue to rise in March, it would indicate a positive growth trend.

These insights are extremely useful for making strategic business decisions.Companies may better optimize their inventory levels, alter their marketing tactics, and plan promotional efforts more effectively if they understand the sales success of each brand over time. This OLAP analysis provides a comprehensive perspective of sales patterns, allowing for data-driven decision making to improve overall sales performance and business growth.

Slicing

Query Query History

```
1  SELECT
2      vendornumber,
3      SUM(dollars) AS total_dollars
4  FROM
5      stg_invoice2
6  WHERE
7      invoicedate = '04/10/2016'
8  GROUP BY
9      vendornumber;
```

Data Output Messages Notifications

SQL

	vendornumber bigint	total_dollars double precision
1	60	48.81
2	388	383.76
3	1265	84.3
4	1655	4136.51
5	2242	4090.83
6	2876	42.21
7	3089	27827.79
8	3252	287968.39
9	5455	1849.62
10	6830	1793.06
11	7239	45417.88
12	7240	959.32
13	8004	2617.32
14	8112	176327.58

Figure 4.1.3 Slicing

Analyze the spending trends in a store on a specific date. The dollars column is the representation of the query's measure. It tells how much money participated in transactions. For dimension the vendor number is selected which is a unique identifier for each vendor or entity that measures will be aggregated across their entities. This slice condition divide the purchase orders on the basis of their PO date and all purchasing transactions on April 10, 2016 only are separated out. In the sense of meaning conveyed by the SUM() function you get an opportunity to do an aggregation concerning the dollars spent with each vendor quantitatively on a certain

day. In figure 4.1.3 output of total dollar highest value is 27827.79 for the vendor number 3252 .Vendor 3252 probably affiliated with major purchase orders or bulk purchasing systems. This vendor may be involved in the supply of high-value goods or a huge quantity, which ultimately results in a significant total dollar amount. The result, when analyzed, adds value by showing distribution of expenditure among different merchants, and it focuses simply on trades transacted on April 10, 2016.

Drill down

```
SELECT
    brand,
    vendorName,
    AVG(purchasePrice) AS averagePurchasePrice
FROM stg_purchase_price2
GROUP BY brand, vendorName;
```

Data Output | Messages | Notifications

	brand bigint	vendorname text	averagepurchaseprice double precision
1	13558	MARTIGNETTI COMPANIES	32.67
2	11420	VINILANDIA USA	3.39
3	3267	WILLIAM GRANT & SONS INC	2.51
4	32065	MARTIGNETTI COMPANIES	4.76
5	3707	DIAGEO NORTH AMERICA INC	0.74
6	3239	ULTRA BEVERAGE COMPANY ...	7.8
7	20218	STATE WINE & SPIRITS	7.84
8	34395	PERFECTA WINES	7.99
9	20507	PERFECTA WINES	20.64
10	26403	CONSTELLATION BRANDS IN...	9.86
11	24981	ULTRA BEVERAGE COMPANY ...	15.99

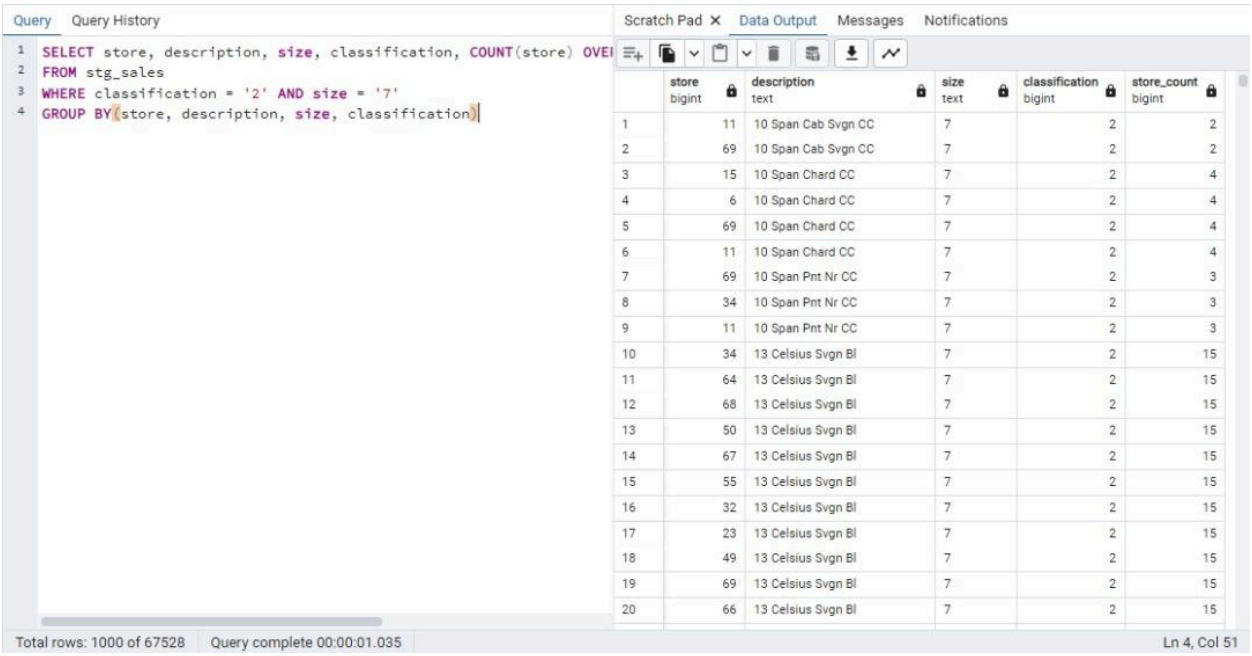
Total rows: 1000 of 12256 Query complete 00:00:00.377

Figure 4.1.4 Drill Down

According to the provided picture, the OLAP (Online Analytical Processing) drill-down technique was used to implement the given SQL query to consider the data elaborately. Drill-down is the process that provides the opportunity for the user to go from summarized to detailed data. For users, as a result, the general view of datasets becomes possible. In this context, the query starts from the data that is extracted from the stg_purchase_price2 table and drills down to the lowest level of detailed information by the data's grouping on brand and vendorName. The average purchase price amount is calculated for each brand-vendor combination and the purchase price is compared with an average purchase price for that brand across all its vendors.

For example the average purchase price for brand 13558 with MARTIGNETTI COMPANIES is 32.67, and for brand 11420, the average purchase price for brand 11420 with VINILANDIA USA equals 3.39. In such a way, with the help of breaking down the information, the drill-down process gives thorough insight into the various types of pricing patterns for each vendor for different brands. This detail is key for any business that wants to fine-tune its purchasing strategy, drive better negotiations, and eventually find areas where potential cost savings are possible. The drill-down approach aids in making a more informed decision since all underlying details of summary data are placed in view, giving a complete view of vendor performance and pricing efficiency.

Dicing



This is a dicing process from table sales to analyze how many stores and which descriptions display from to category size with seven only, classification with two only. There are 51 columns displayed. There also count how many stores have the same categories. The SQL query filtered and grouped data from the stg_sales table, resulting in a comprehensive study of individual segments. It chooses the columns store, description, size, and classification, as well as the computed column store_count, which reflects the number of stores for each description. The

filtering rule WHERE classification = 2 and size = 7 guarantees that only rows with these particular properties are included.

The GROUP BY clause then groups the filtered rows according to a combination of store, description, size, and categorization, ensuring that each group is distinct. The COUNT(store) OVER (PARTITION BY description) as store_count section of the query is a window function that determines how many times each description occurs in all groups. This count is added to the output as a new column called store_count.

For instance, if the data has stores with descriptions like 10 Span Cab Svgn CC, or 10 Span Chard CC, the query will summarize the total number of shops with each description and add them to the return set. The concept of dicing, which is to select some subcollection of data based on filters across given dimensions, can be seen here. By filtering the dataset to contain all the rows where classification = 2 and size = 7, the aggregation across various dimensions to represent aggregate metrics, the query allows for a detailed comparison across the diverse segments of data. This is a technique that provides distribution information for things based on their description, size, and classification with their features.

Pivot

Query | Query History

```
SELECT vendorname,
       EXTRACT(YEAR FROM invoicedate) AS year,
       EXTRACT(MONTH FROM invoicedate) AS month,
       SUM(dollars) AS monthly_sales
FROM stg_purchase2
GROUP BY vendorname, year, month
ORDER BY vendorname, year, month;
```

Data Output | Messages | Notifications

	vendorname text	year numeric	month numeric	monthly_sales double precision
1	ADAMBA IMPORTS INTL INC	2016	5	3319.079999999998
2	ADAMBA IMPORTS INTL INC	2016	6	5710.769999999998
3	ADAMBA IMPORTS INTL INC	2016	7	3449.239999999999
4	ALISA CARR BEVERAGES	2016	2	1024.95
5	ALISA CARR BEVERAGES	2016	3	1895.2
6	ALISA CARR BEVERAGES	2016	4	6660.2
7	ALISA CARR BEVERAGES	2016	5	5529.7
8	ALISA CARR BEVERAGES	2016	6	6930.1799999999985
9	ALTAMAR BRANDS LLC	2016	1	642.78

The SQL queries in figures 4.1.6 retrieves the monthly sales data for each vendor name by extracting the year and month. From the analysis, we can understanding the sales trend by identify the sales trends over the year for each vendor. We can identify the seasonal trends for the products since some beverages might have sell more during holidays or summer. Through this factor we can prepare for the product demand so they have sufficient stock during the peak periods.

This queries also shows the company performance so we can track their performance each time. Vendors that have a consisten sales over the month have positive growth and indicates a strong partnership. We also can address the issue for the low sales vendors. This insights helps the company to negotiate better terms or promotional deals during high-sales periods and ensuring the company can meet its financial obligations and invest in growth opportunities.

5.2 Data Visualisation

Invoice

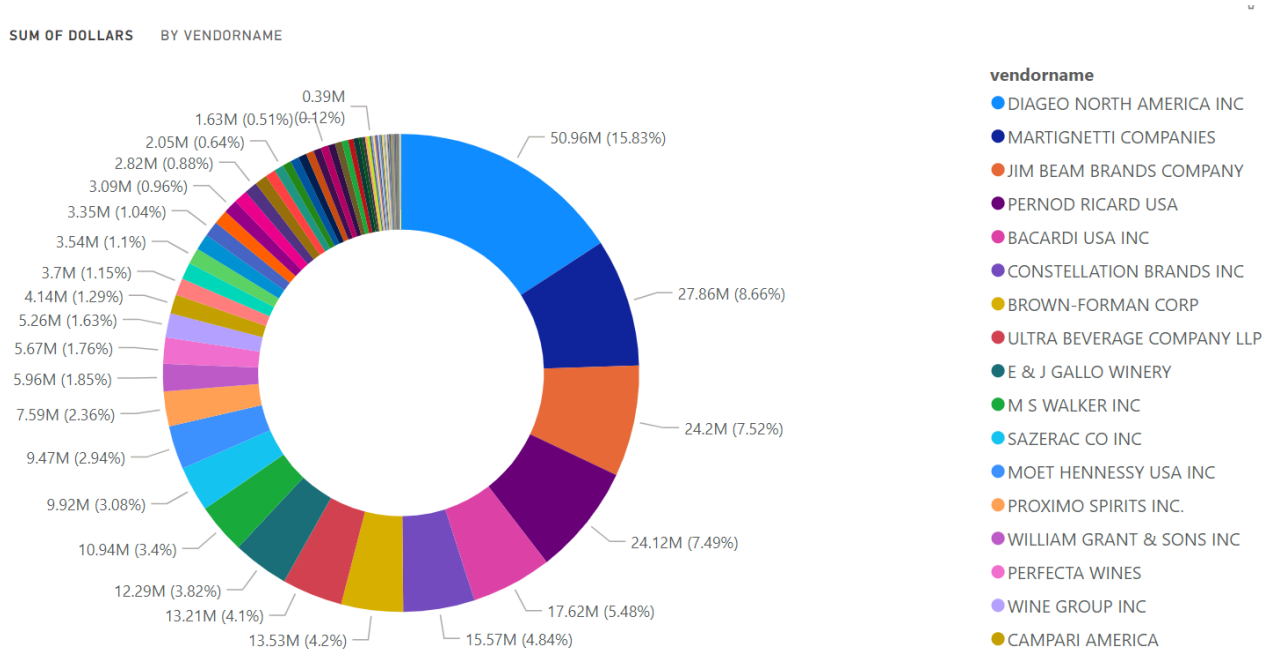


Figure 4.2.1 Sum Of Dollars By Vendorname

Donut chart summarizing the dollar sales in the beverage industry across the various vendors. Diageo North America Inc commands the largest share at $50.96 million, contributing to 15.83% of the total. The immediate runner-up is Martignetti Companies with a share of $27.86 million and an 8.66% contribution, making it the second-largest share. Jim Beam Brands Company is the third major player with its share at $24.2 million (7.52%).

The other vendors with major shares include Pernod Ricard USA and Bacardi USA Inc, which have their presence almost equal at $24.12 million (7.49%) and $17.62 million (5.48%) in the market. E & J Gallo Winery and M S Walker Inc offer shares worth $9.47 million (2.94%) and $9.92 million (3.08%), respectively, which is still a decent sum in the market.

Constellation Brands Inc, Brown-Forman Corp, and Ultra Beverage Company LLP also have their share of presence with sales of $15.57 million (4.84%), $13.53 million (4.2%), and $13.21 million (4.1%). Smaller shares are found in the market with vendors such as Sazerac Co Inc ($7.59 million, 2.36%) and Moet Hennessy USA Inc ($5.96 million, 1.85%).

Among the other vendors in the chart, 0.12% to 1.85% of dollar sales are completed by Proximo Spirits Inc., William Grant & Sons Inc, Perfecta Wines, Wine Group Inc, and Campari America. This shows that the dollar sales are spread across most other vendors, large and small.

Sales

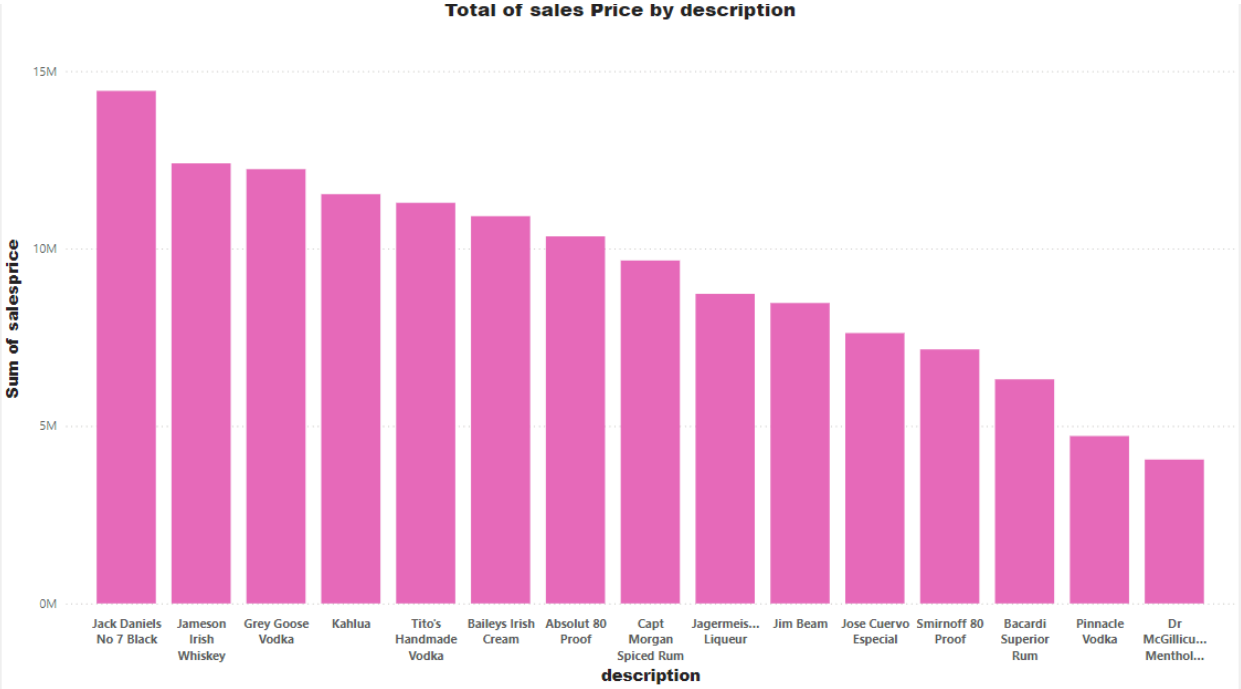


Figure 4.2.2 Total sales price by description product

This bar chart shows the total sales price of various types of beverage. The description of Jack Daniels No 7 Black has a total sales price of around $5 million, while description Bacardi Superior Vodka has the total sale price around $3 million. They may increase the sales price of Jack Daniels No 7 Black if, for instance, there is increased market demand for the product in a specified geographic location. It could be due to local preference or prevailing culture at the period under consideration or the region being considered. There could be a shortage of Jack Daniels No 7 Black to give it a prestige-like feel, this would make the price of the product skyrocket. Superior Vodka, there may be lesser total volume of the kind of beverage described in contrast with other booze. This could be because of regionalism, seasonality or latest marketing strategies that may be new to most people. Possibly the beverage is cheaper than its competitors and even though it gains high sales per unit it gains relatively less overall sales revenue.

Ending Inventory

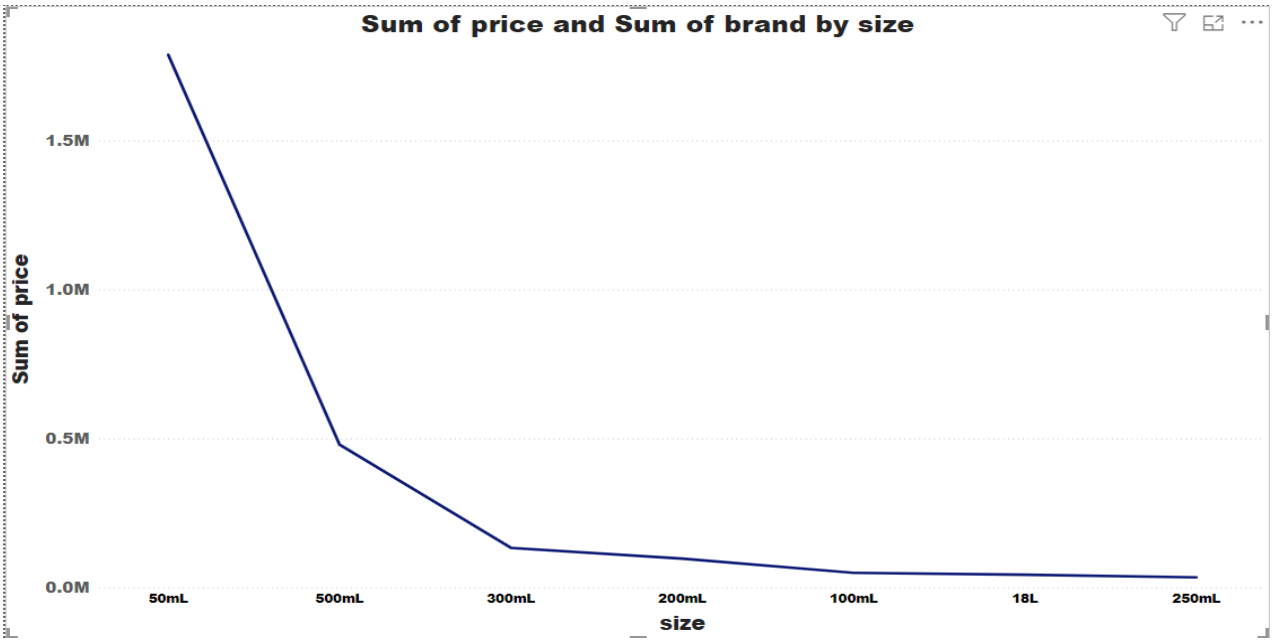


Figure 4.2.3 Ending Inventory Sum of price and Sum of brand by size

The provided line chart shows the 'Sum of price' and the 'Sum of brand' across various size categories, showing important trends in pricing and brand distribution. The Sum of price shows an inverse relationship with product size where smaller sizes, particularly the 50ml category, have the highest cumulative price, exceeding 1.5 million units. When the size increases to 500ml, the total price decreases a lot, showing a big drop in either how much is sold or the price of each item. This decrease keeps happening slowly for smaller sizes like 300ml, 200ml, and 100ml. The 18l and 250ml sizes have the lowest total prices.

In contrast to the obvious changes in pricing, the 'Sum of brand' appears to be quite consistent across different size categories. The data shows no major changes in brand count, suggesting that the number of brands offering products in each size category is largely consistent. This consistency shows that, whereas the income contribution from different sizes changes dramatically, the market presence in terms of brand variety does not change much with size.

Overall, the visualization shows that smaller product sizes, especially the 50ml size,make major contributions to the total price, most likely due to larger sales volumes or per-unit prices. Bigger

sizes add less to the overall cost, which might mean fewer sales or lower prices. Even with these price changes, having the same brand availability in all sizes shows that brands keep a steady place in the market, no matter the size. This information helps in making choices about prices, advertising, and what products to sell, showing how smaller sizes are important for making money while keeping a broad company image.

Purchase

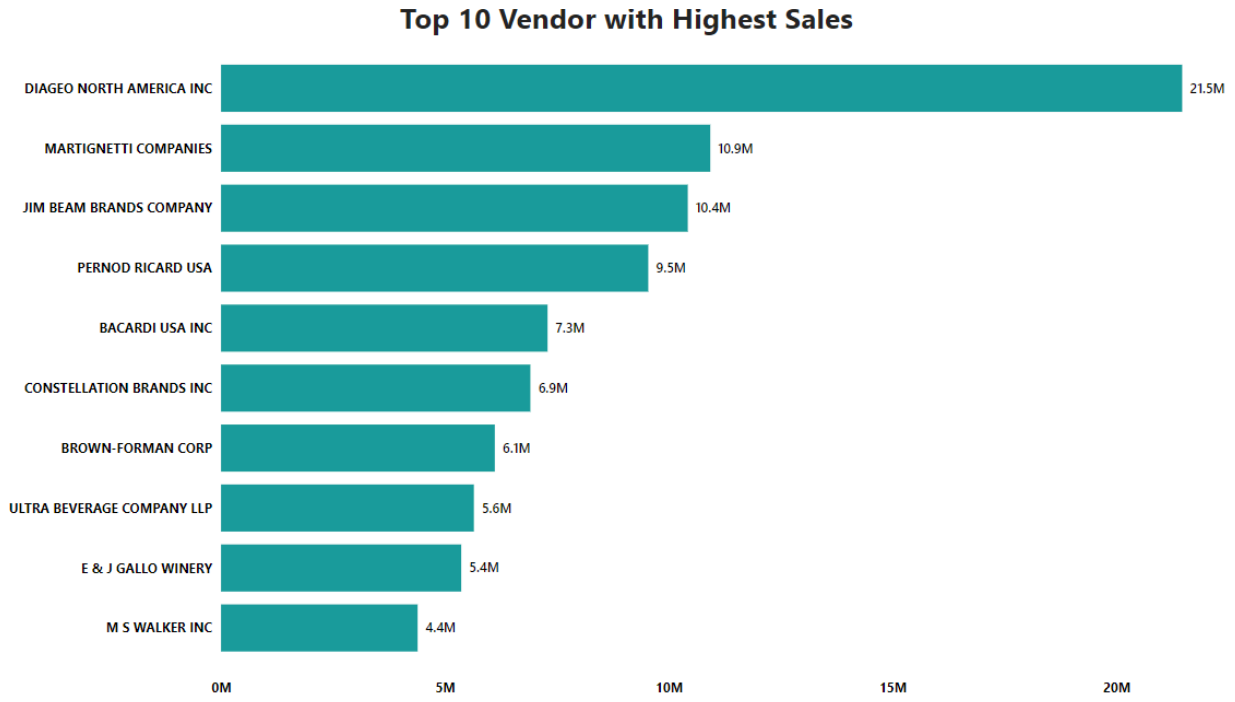


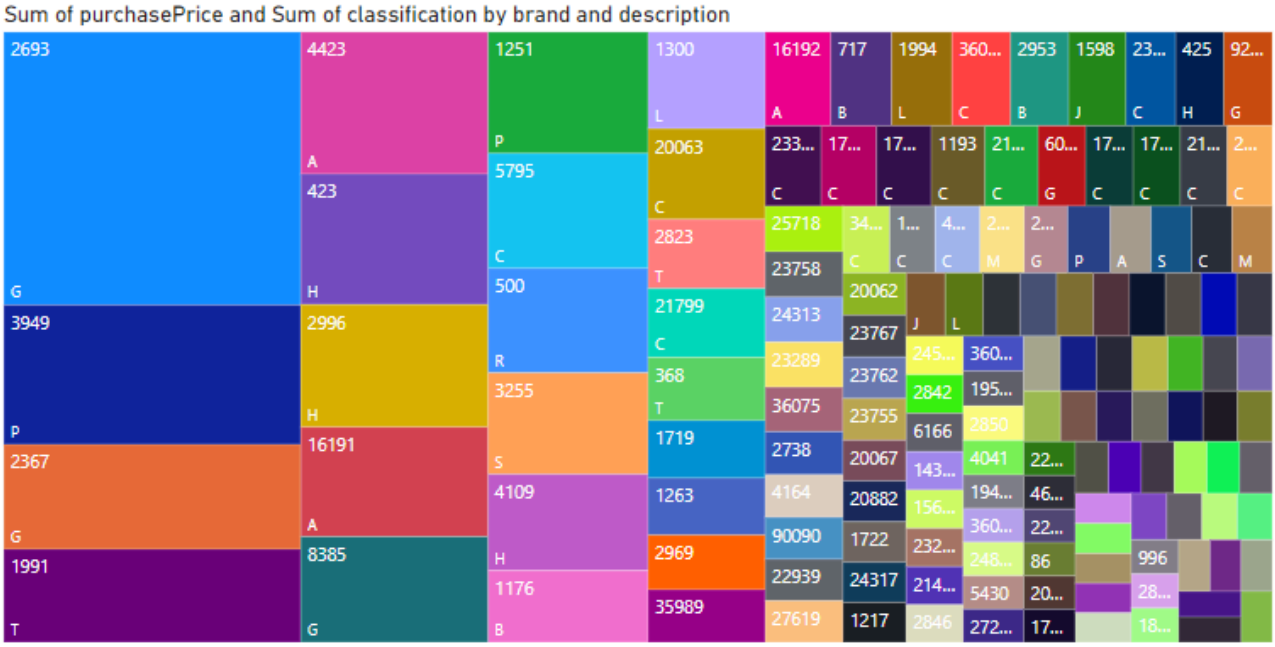
Figure 4.2.4 Top 10 vendor with highest sales

Figure 4.2.4 shows that Diageo North America Inc have the highest sales among other vendors which is $21.5 million. Diageo North America Inc stand out among other vendors since have the double amount of sales compared to the second and third place vendors. From the result, it shows that Diageo North America Inc has a strong market presence and have diverse popular product. This lead to high demand in the market and being one of the key player of LLC'c inventory. Martignetti Companies and Jim Beams Brands Company only have a slightly difference of sales and got the second and third place with sales of $10.9 million and $10.4 million perspectively. Eventhough both sales behind Diageo, its also shown that they solid customers based and strong performance. The small difference of sales indicates that both vendors are very competitive and have popular products that significantly contributes to overall sales.

Through this top 10 ranking, we can focus on the high selling vendors which is Diageo since it dominate others vendor. We need to make sure their products are well stocked to avoids

stockouts since it is high demand. We also need to focus on Martignetti and Jim Beam Brands since both have the potential to increase their sales. Besides maintaining the high seller vendors, we need to maintain the balanced inventory to the others vendors to make sure we have diverse products of customers preferences. Lastly, we can utilized the sales data to predict demand trends to ensure optimal stocks and minimize excess inventory. Thus we can optimized the inventory management, reduce cost and enhance costumes satisfaction throughout the product availability.

Purchase Price



The provided treemap chart visually represents the sum of purchase price and sum of classification values, segmented by brand and description. The largest contributions come from brands 2693 and 2349. The most largest block is brand 2693 with sum purchase price ($11111.03) and from classification 1. Another notable large block is brand 2349 with sum purchase price ($5681.81) and from classification 1 too. Next, a large block brand is 4423 indicating a substantial contribution to be overall sum with sum purchase price is $5791.65 and classification with 2. Fourth highest block brand is 423 with a sum purchase price $4409.24 and classification 1. Fifth highest block brand is 2996 with a sum purchase price $3601.88 and classification 3. Next brand is 3949 with a sum purchase price $39500 and classification 1. Followed by brand 2367 with a sum purchase price is $3652.79 and classification 4. Next brand is 16191 with a sum purchase price $2857.60 and classification 2. Following by brand 1991 with a sum purchase price $2380 and classification 5. The last top highest brand is 8385 with a sum purchase price $1950 and classification 1.

6.0 CONCLUSION

In conclusion, from all visualization the dollar sales in the beverage industry across the various vendors. Diageo North America Inc commands the largest share at $50.96 million, contributing to 15.83% of the total. Next, a description of Jack Daniels No 7 Black has a total sales price of around $5 million, while description Bacardi Superior Vodka has the total sale price around $3 million. Besides, a relationship with product size where smaller sizes, particularly the 50ml category, have the highest cumulative price, exceeding 1.5 million units. From the purchase the most highest sales is Diageo North America INC with 21.5M. the sum of purchase price and sum of classification values, segmented by brand and description. The largest contributions come from brands 2693 and 2349. The most largest block is brand 2693 with sum purchase price ($11111.03) and from classification 1. Another notable large block is brand 2349 with sum purchase price ($5681.81) and from classification 1 too.

However, it revealed the crucial need of using data analysis and OLAP (Online Analytical Process) methodologies to optimize inventory management. The project's rigorous recording and analysis of sales, purchases, and inventory data across several dimensions gave useful insights into pricing strategies, sales performance, and brand distribution. Implementing a structured data warehouse environment permitted thorough data integrity and deep analysis, allowing for the detection of key patterns and developments for strategic decision-making. However, based on the tool, using a Lakehouse architecture not just helps the analytical objectives of inventory analysis, but also provides a solid foundation to enhancing operational effectiveness, driving strategic decision-making, and fostering ongoing improvements in inventory management practices.

7.0 REFERENCES

Galaktikasoft, & Galaktikasoft. (2019b, February 7). *OLAP operations in data mining*. Galaktikasoft. https://galaktika-soft.com/blog/olap-operations-in-data-mining.html

Goal 12 | Department of Economic and Social Affairs. (n.d.). https://sdgs.un.org/goals/goal12

Inc, R. T. S. (2023, November 28). *ETL testing*. QuerySurge. https://www.querysurge.com/solutions/etl-testing?utm_source=QS-Site&utm_medium=web-page&utm_campaign=ETL-Testing-Everything-you-need-to-know&msclkid=53416ecb84e91323d88608ac9424d410

Inventory Analysis Case Study. (2023b, July 13). Kaggle. https://www.kaggle.com/datasets/bhanupratapbiswas/inventory-analysis-case-study

OLAP operations. (n.d.). https://athena.ecs.csus.edu/~mei/olap/OLAPoperations.php

Ramdhanhidayat. (2023, July 26). *Inventory analysis - data preprocessing*. Kaggle. https://www.kaggle.com/code/ramdhanhidayat/inventory-analysis-data-preprocessing

What is ETL (Extract, Transform, Load)? | IBM. (n.d.). https://www.ibm.com/topics/etl

8.0 APPENDIX

Query to create tables

```
CREATE TABLE beg_inv (
            inventoryId TEXT,
            store INT,
            city TEXT,
            brand INT,
            description TEXT,
            size TEXT,
            onHand INT,
            price NUMERIC,
            startDate TIMESTAMP
);

CREATE TABLE end_inv (
            inventoryId TEXT,
            store INT,
            city TEXT,
            brand INT,
            description TEXT,
            size TEXT,
            onHand INT,
            price NUMERIC,
            endDate TIMESTAMP
);

CREATE TABLE invoice (
            vendorNumber INT,
            vendorName TEXT,
            invoiceDate TIMESTAMP,
            pONumber INT,
            pODate TIMESTAMP,
            payDate TIMESTAMP,
            quantity INT,
            dollars NUMERIC,
            freight NUMERIC,
            approval TEXT
);

CREATE TABLE purchase (
            inventoryId TEXT,
            store INT,
```

```
            brand INT,
            description TEXT,
             size TEXT,
             vendorNumber INT,
             vendorName TEXT,
             pONumber INT,
             pODate TIMESTAMP,
             receivingDate TIMESTAMP,
             invoiceDate TIMESTAMP,
      payDate TIMESTAMP,
      purchasePrice NUMERIC,
      quantity INT,
      dollars NUMERIC,
      classification INT
);

CREATE TABLE purchase_price (
      brand INT,
      description TEXT,
      price NUMERIC,
      size TEXT,
      volume NUMERIC,
      classification INT,
      purchasePrice NUMERIC,
      vendorNumber INT,
             vendorName TEXT
);

CREATE TABLE sales (
             inventoryId TEXT,
             store INT,
      brand INT,
      description TEXT,
      size TEXT,
      salesQuantity INT,
      salesDollars NUMERIC,
      salesPrice NUMERIC,
             salesDate TIMESTAMP,
      volume INT,
      classification INT,
      exciseTax NUMERIC,
      vendorNumber INT,
             vendorName TEXT
);
```