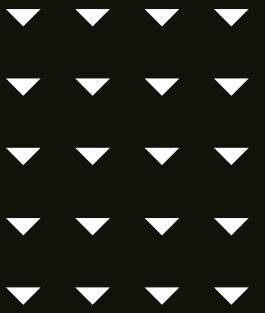




We Are Open

RESTAURANT ORDER DATASET *And* SQL ANALYSIS

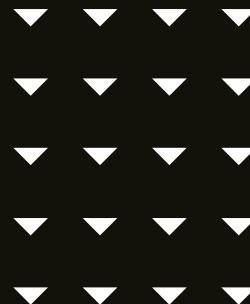
Presentation



ABOUT ME

Hi, my name is Dwi Winda Fitrika and I am a student from MSIB CAKAP from the Data Scientist class. Here I want to present the results of the SQL analysis project with PostgreSQL and visualization using Python for the Restaurant Order dataset from Maven Analytics.

We Are Open



01. DATA PREPARATION

Downloading, Cleaning, and Merging
Dataset using Postgre SQL

02. SQL QUERY

Answered 4 recommended analysis
and 2 additional analysis questions
with SQL

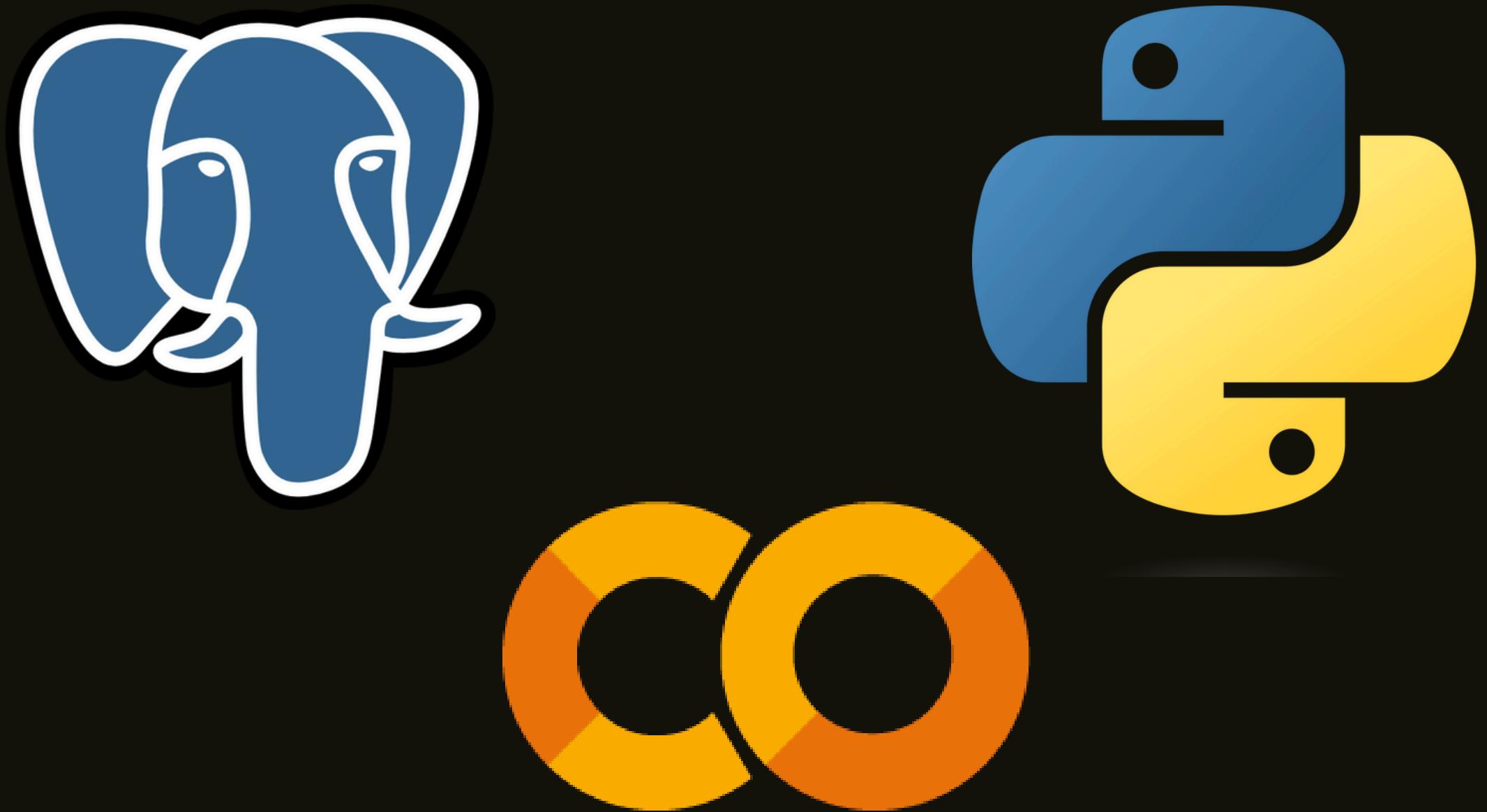
03. INTEGRATION WITH PYTHON AND VISUALIZATION

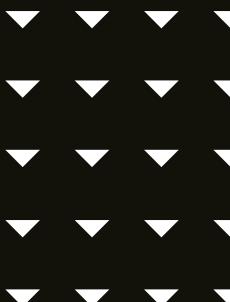
Connecting postgresql with python and
visualizing with python libraries

OUTLINE



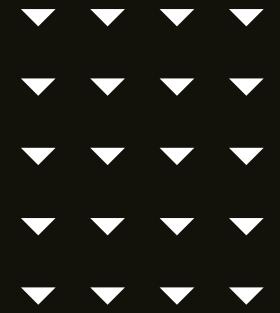
TOOLS USED





01. DATA PREPARATION

DOWNLOAD DATASET



The screenshot shows a user interface for downloading datasets. At the top, there are three cards representing different datasets:

- Restaurant Orders**: A quarter's worth of orders from a fictitious restaurant. File types: Excel, MySQL, CSV. Tags: Human Resources, Government.
- Coffee Shop Sales**: Transaction records from a fictitious coffee shop. File types: Excel. Tags: Food & Beverage, Geospatial.
- Restaurant Orders**: Another card for the same dataset as the first, with file types MySQL and CSV, and tags Food & Beverage, Business, Time Series.

The central part of the interface is a detailed view of the second "Restaurant Orders" dataset:

Restaurant Orders

A quarter's worth of orders from a fictitious restaurant serving international cuisine, including the date and time of each order, the items ordered, and additional details on the type, name and price of the items.

Recommended Analysis

1. What were the least and most ordered items? What categories were they in?
2. What do the highest spend orders look like? Which items did they buy and how much did they spend?
3. Were there certain times that had more or less orders?
4. Which cuisines should we focus on developing more menu items for based on the data?

Want feedback on your solutions?

On the right side of the detailed view, there is a download section with a dropdown menu:

- Download (dropdown menu)
- [SQL \(551 KB\)](#) (highlighted with a yellow box and arrow)
- [CSV \(412 KB\)](#)

Below the download section are dataset statistics:

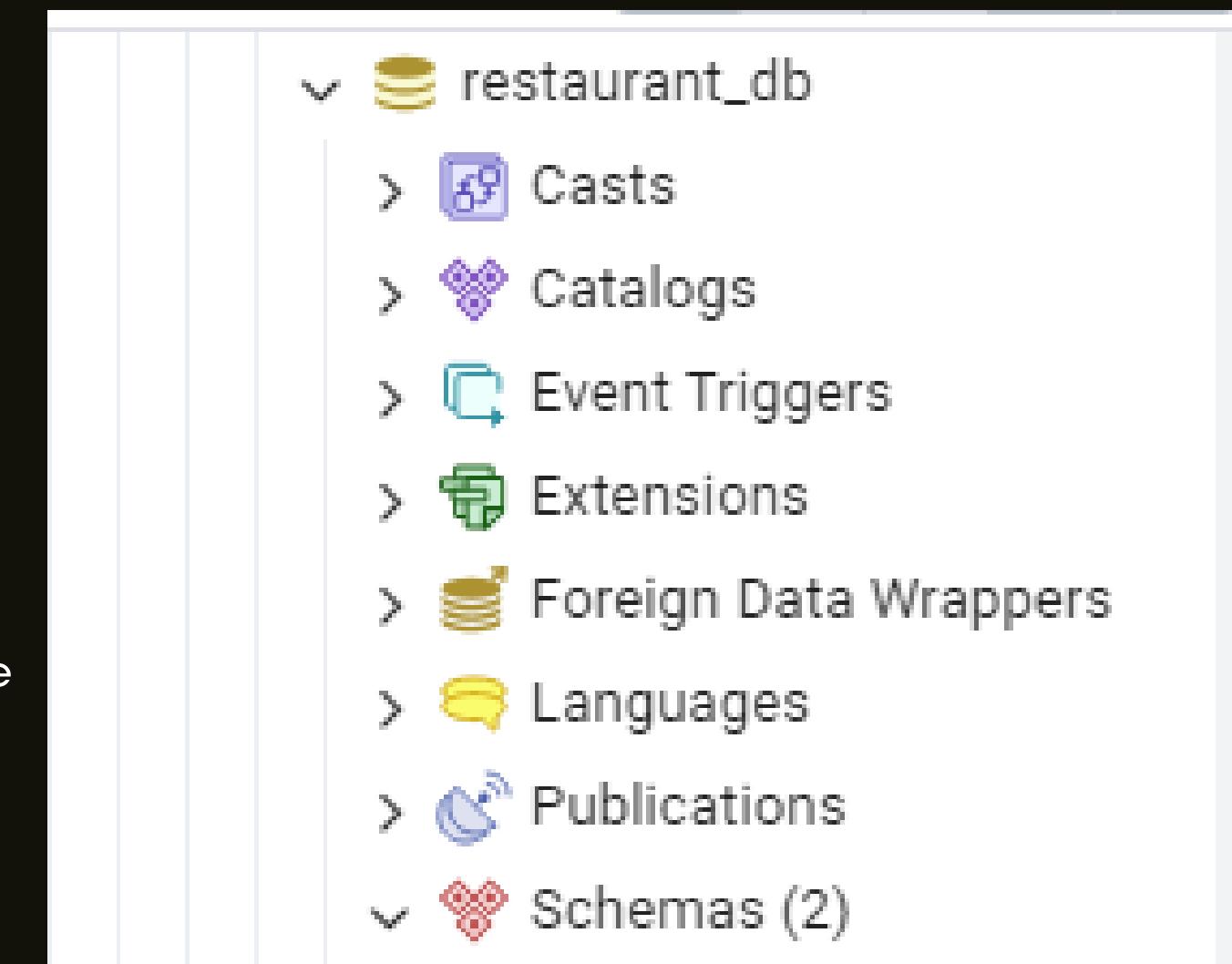
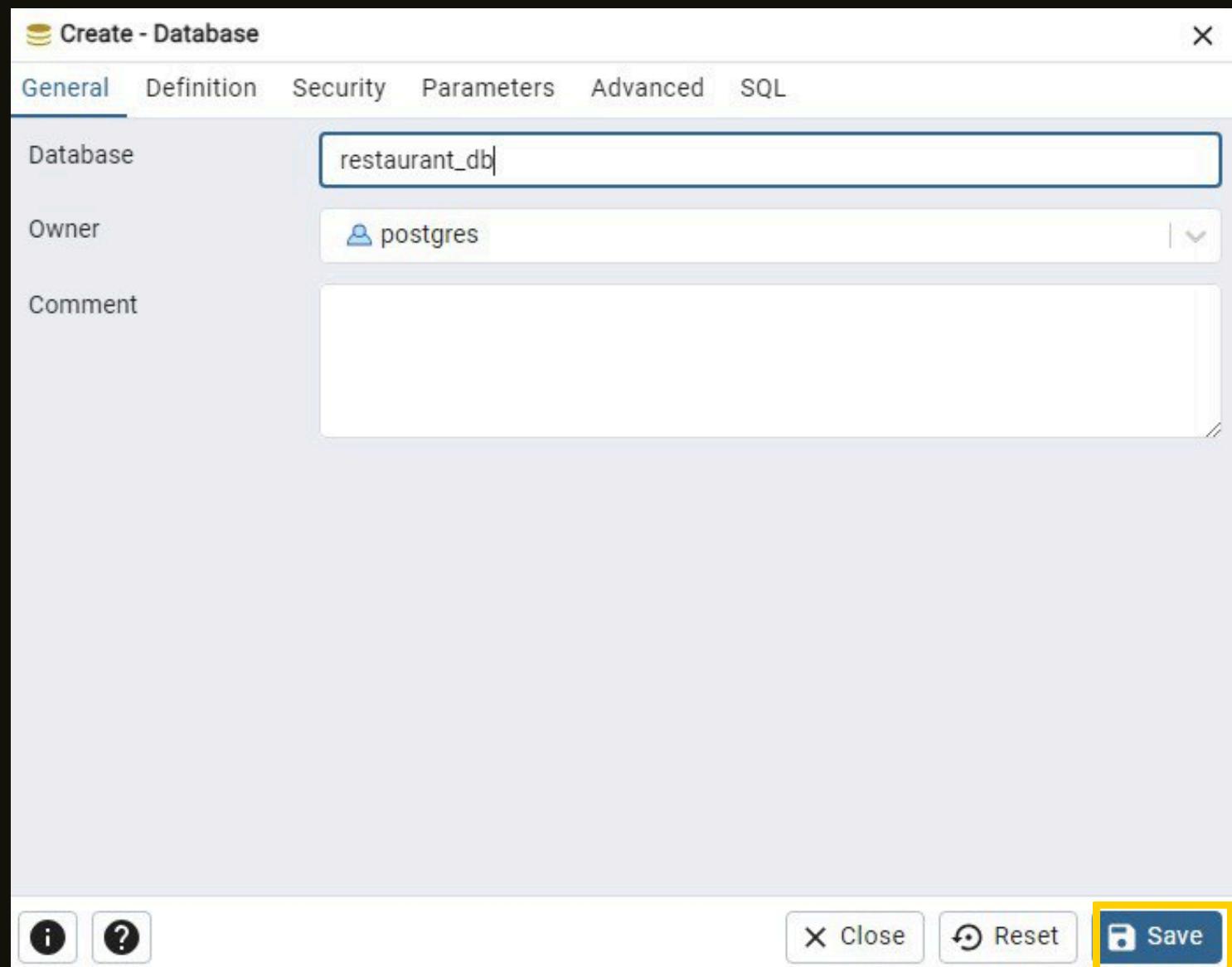
DATA STRUCTURE	# OF RECORDS
Multiple tables	12,266

# OF FIELDS	DATE ADDED
8	10/24/2023

Download SQL
Dataset for import
Process in Postgre
SQL Pgadmin

DATASET IMPORT (SQL)

1. Create Database



name and save the
database

DATASET IMPORT (SQL)

2. Load Data to SQL by Open File in Pgadmin

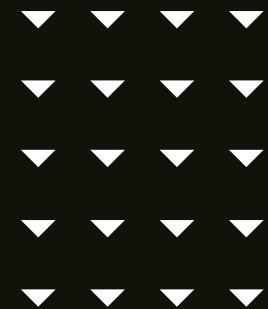
This is tools for open file SQL

```
1 DROP SCHEMA IF EXISTS restaurant_db;
2 CREATE SCHEMA restaurant_db;
3 USE restaurant_db;          Delete USE Query because in pgadmin the USE query is not used
4
5 --
6 -- Table structure for table `order_details`
7 --
8
9 CREATE TABLE order_details (
10    order_details_id SMALLINT NOT NULL,
11    order_id SMALLINT NOT NULL,
12    order_date DATE,
13    order_time TIME,
14    item_id SMALLINT,
15    PRIMARY KEY (order_details_id)
16 );
17
18 --
19 -- Table structure for table `menu_items`
20 --
21
22 CREATE TABLE menu_items (
23    menu_item_id SMALLINT NOT NULL,
24    item_name VARCHAR(45),
```

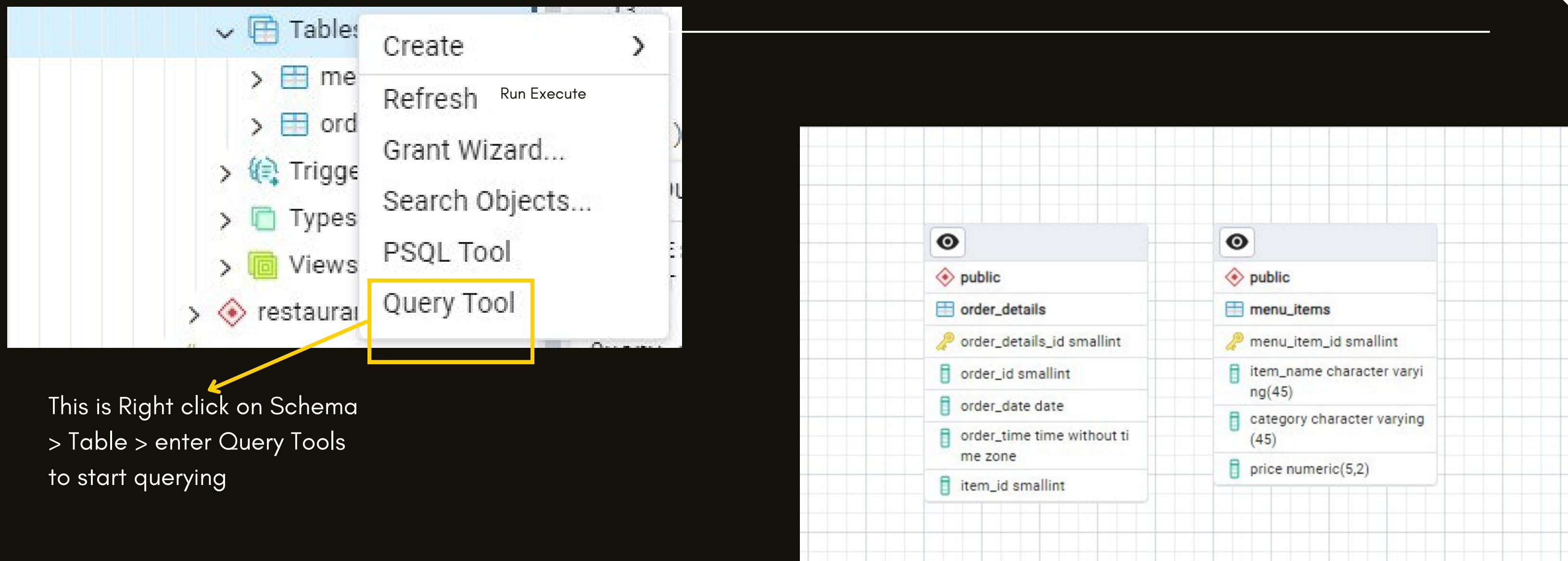
Data Output	Messages	Notifications
	NOTICE: schema "restaurant_db" does not exist, skipping INSERT 0 32	
	Query returned successfully in 1 secs 892 msec.	

Run Query until Query Returned Successfully

DATA UNDERSTANDING (SQL)



3. Start Query



This is Right click on Schema
> Table > enter Query Tools
to start querying

This is ERD for Database has two tables
with the table names menu_items and
order_details columns and their data types

DATA QUERY (SQL)

4. Check Table

```
1 -- CEK TABLE menu_items
2 SELECT * FROM menu_items;
```

Data Output Messages Notifications

	menu_item_id [PK] smallint	item_name character varying (45)	category character varying (45)	price numeric (5,2)
1	101	Hamburger	American	12.95
2	102	Cheeseburger	American	13.95
3	103	Hot Dog	American	9.00
4	104	Veggie Burger	American	10.50
5	105	Mac & Cheese	American	7.00
6	106	French Fries	American	7.00
7	107	Orange Chicken	Asian	16.50
8	108	Tofu Pad Thai	Asian	14.50
9	109	Korean Beef Bowl	Asian	17.95
10	110	Pork Ramen	Asian	17.95
11	111	California Roll	Asian	11.95
12	112	Salmon Roll	Asian	14.95
13	113	Edamame	Asian	5.00
14	114	Potstickers	Asian	9.00
15	115	Chicken Tacos	Mexican	11.95

Total rows: 32 of 32 Query complete 00:00:00.266 Ln 2, Col 26

check the menu_items table record using the query as shown and there are rows with a total of 32,

```
4 -- CEK TABLE order_details
5 SELECT * FROM order_details;
```

Data Output Messages Notifications

	order_details_id [PK] smallint	order_id smallint	order_date date	order_time time without time zone	item_id smallint
1		1	1	2023-01-01 11:38:36	109
2		2	2	2023-01-01 11:57:40	108
3		3	2	2023-01-01 11:57:40	124
4		4	2	2023-01-01 11:57:40	117
5		5	2	2023-01-01 11:57:40	129
6		6	2	2023-01-01 11:57:40	106
7		7	3	2023-01-01 12:12:28	117
8		8	3	2023-01-01 12:12:28	119
9		9	4	2023-01-01 12:16:31	117
10		10	5	2023-01-01 12:21:30	117
11		11	6	2023-01-01 12:29:36	101
12		12	6	2023-01-01 12:29:36	114
13		13	7	2023-01-01 12:50:37	123
14		14	8	2023-01-01 12:51:37	123

Total rows: 1000 of 12234 Query complete 00:00:00.460 Ln 5, Col 29

Check the order_details table record using a query like the image and there are rows with a total of 12234.

DATA CLEANSING (SQL)

5. Check NULL Value

```
7 -- CEK Nilai NULL di table menu_items
8 SELECT *
9 FROM menu_items
10 WHERE item_name IS NULL
11 OR category IS NULL
12 OR price IS NULL;
```

Total rows: 0 of 0 | Query complete 00:00:00.316

check NULL VALUE in each table, for table menu_items like the query shown, and there are 0 NULL VALUE

```
22 SELECT COUNT(*) AS null_count
23 FROM order_details
24 WHERE item_id IS NULL;
25
```

Total rows: 1 of 1 | Query complete 00:00:00.252

Count the NULL VALUE in the item_id column and there are 137 NULL VALUE.

```
14 -- CEK Nilai NULL di table order_details
15 SELECT *
16 FROM order_details
17 WHERE order_id IS NULL
18 OR order_date IS NULL
19 OR order_time IS NULL
20 OR item_id IS NULL;
```

	order_details_id [PK] smallint	order_id smallint	order_date date	order_time time without time zone	item_id smallint
1		122	50	2023-01-01	18:41:01
2		298	125	2023-01-02	20:31:06
3		358	147	2023-01-03	14:32:51
4		387	161	2023-01-03	16:43:46
5		470	200	2023-01-03	22:24:05
6		474	201	2023-01-03	22:29:59
7		779	338	2023-01-06	15:18:26
8		833	364	2023-01-06	19:27:24
9		854	376	2023-01-07	12:01:17
10		941	410	2023-01-07	17:33:49

Total rows: 137 of 137 | Query complete 00:00:00.252 | Ln 20, Col 23

Check NULL VALUE for the order_details table as shown in the query, and there are 137 NULL VALUE in the item_id column.

DATA CLEANSING (SQL)

6. Handling NULL VALUE

Because a NULL value was found in the order_details table, column item_id, let's handle the NULL value based on the data type in both tables. The missing null value is of the categorical type, so handling this NULL is by determining the value that appears most often (mode) in item_id, to fill in the value. -

```
27 --HANDLING NULL VALUE--  
28  
29 -- Karena ditemukan Nilai NULL pada tabel order_details, column item_id mari kita handling nilai NULL tersebut  
30 -- -berdasarkan jenis data pada kedua table nilai null yang hilang merupakan jenis categorical, maka handling NULL ini  
31 -- dengan menentukan nilai yang paling banyak muncul (modus) pada item_id, untuk mengisi nilai null tersebut--  
32  
33 -- MELIHAT NILAI MODUS YANG PALING BANYAK MUNCUL DI item_id  
34 SELECT item_id, COUNT(*) AS jumlah_kemunculan  
35 FROM order_details  
36 GROUP BY item_id  
37 ORDER BY jumlah_kemunculan DESC  
38 LIMIT 1;  
39
```

Data Output Messages Notifications

item_id	jumlah_kemunculan
101	622

Total rows: 1 of 1 Query complete 00:00:00.305 Ln 38, Col 8

Look at the Mode value that appears the most in the item_id column. From the query as shown, it is known that the value that appears the most is 101, 622, so fill in the null value with item_id 101

Update the item_id column based on the value that appears the most. The query is as shown and the update process is successful.

```
40 --dari query ini diketahui, nilai paling banyak muncul adalah 101, maka untuk mengisi null value dengan item_id 101  
41  
42 --MELAKUKAN UPDATE UNTUK KOLOM item_id BERDASARKAN NILAI YANG PALING BANYAK MUNCUL  
43 UPDATE order_details AS od  
44 SET item_id = (  
45   SELECT subquery.item_id  
46   FROM (  
47     SELECT item_id, COUNT(*) AS count  
48     FROM order_details  
49     WHERE item_id IS NOT NULL  
50     GROUP BY item_id  
51     ORDER BY count DESC  
52     LIMIT 1  
53   ) AS subquery  
54 )  
55 WHERE item_id IS NULL;  
Data Output    Messages    Notifications  
UPDATE 137  
Query returned successfully in 515 msec.
```

Total rows: 1 of 1 Query complete 00:00:00.515 Ln 51, Col 28

DATA CLEANSING (SQL)

7. Re-Check NULL VALUE

```
57 --CEK KEMBALI UNTUK MELIHAT NULL VALUE
58 SELECT *
59 FROM menu_items, order_details
60 WHERE item_name IS NULL
61 OR category IS NULL
62 OR price IS NULL
63 OR order_id IS NULL
64 OR order_date IS NULL
65 OR order_time IS NULL
66 OR item_id IS NULL;
--
```

Data Output Messages Notifications

menu_item_id	item_name	category	price	order_details_id	order_id	order_date	order_time	item_id
smallint	character varying (45)	character varying (45)	numeric (5,2)	smallint	smallint	date	time without time zone	smallint

Total rows: 0 of 0 Query complete 00:00:00.297 Ln 63, Col 7

Check the value again to see the null value, and the process shows the NULL value no longer exists.

```
68 SELECT COUNT(*) AS null_count
69 FROM menu_items, order_details
70 WHERE item_name IS NULL
71 OR category IS NULL
72 OR price IS NULL
73 OR order_id IS NULL
74 OR order_date IS NULL
75 OR order_time IS NULL
76 OR item_id IS NULL;
--
```

Data Output Messages Notifications

null_count
1

Total rows: 1 of 1 Query complete 00:00:00.320

Calculate the NULL value with the query as in the picture, with the result being that the Null value is 0.

HANDLING NULL VALUES HAS BEEN COMPLETED

DATA MERGER (SQL)

8. Join Table order_details and menu_items

```
80 ---MELAKUKAN LEFT JOIN UNTUK MENGGABUNGKAN TABLE menu_items dan order_details---
81 SELECT od.order_details_id,
82     od.order_id,
83     od.order_date,
84     od.order_time,
85     od.item_id,
86     mi.item_name,
87     mi.category,
88     mi.price
89 FROM order_details od
90 LEFT JOIN menu_items mi ON od.item_id = mi.menu_item_id;
91 --menu_item_id tidak digabungkan karena sama dengan item_id, untuk mempermudah table di analisis--
```

Data Output Messages Notifications

	order_details_id smallint	order_id smallint	order_date date	order_time time without time zone	item_id smallint	item_name character varying (45)	category character varying (45)	price numeric (5,2)	
1		1	2023-01-01	11:38:36	109	Korean Beef Bowl	Asian	17.95	
2		2	2023-01-01	11:57:40	108	Tofu Pad Thai	Asian	14.50	
3		2	2023-01-01	11:57:40	124	Spaghetti	Italian	14.50	
4		2	2023-01-01	11:57:40	117	Chicken Burrito	Mexican	12.95	
5		2	2023-01-01	11:57:40	129	Mushroom Ravioli	Italian	15.50	
6		2	2023-01-01	11:57:40	106	French Fries	American	7.00	
7		3	2023-01-01	12:12:28	117	Chicken Burrito	Mexican	12.95	

Total rows: 1000 of 12234 Query complete 00:00:00.601

```
93 ---COMBINE TABLE menu_items dan order_details jadi 1 table--
94 CREATE TABLE combined_table AS
95 SELECT od.order_details_id,
96     od.order_id,
97     od.order_date,
98     od.order_time,
99     od.item_id,
100    mi.item_name,
101    mi.category,
102    mi.price
103   FROM order_details od
104  JOIN menu_items mi ON od.item_id = mi.menu_item_id;
105
```

Data Output Messages Notifications

```
SELECT 12234
```

Query returned successfully in 156 msec.

Left join to combine the menu_items and order details tables, the menu_item_id is not combined because it is the same as the item_id, to make the table easier to analyze

COMBINE TABLE menu_items and order_details into 1 table with the name combined_table with the query as shown and the query has been successful

DATA MERGER (SQL)

9. See Record from new Table combined_table

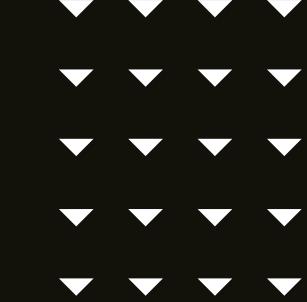
The screenshot shows a PostgreSQL SQL editor interface. The top navigation bar includes tabs for 'Query' (which is selected), 'Query History', 'Data Output' (selected), 'Messages', and 'Notifications'. Below the tabs is a toolbar with various icons for file operations like new, open, save, and export. The main area displays a table with 15 rows of data. The table has columns: order_details_id, order_id, order_date, order_time, item_id, item_name, category, and price. The data represents 15 different food items ordered on January 1, 2023, at various times between 11:38:36 and 12:52:01. The items include Korean Beef Bowl, Tofu Pad Thai, Spaghetti, Chicken Burrito, Mushroom Ravioli, French Fries, Chicken Burrito, Chicken Torta, Chicken Burrito, Chicken Burrito, Hamburger, Potstickers, Chips & Guacamole, Chips & Guacamole, and Tofu Pad Thai. Prices range from \$7.00 to \$17.95. The bottom status bar shows the path 'restaurant_db > Schemas > public > Tables > combined_table' and the text 'Ln 107, Col 30'.

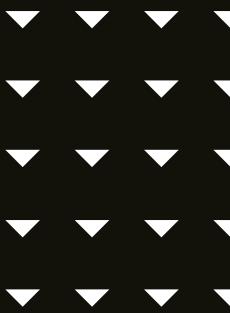
	order_details_id	order_id	order_date	order_time	item_id	item_name	category	price
1	1	1	2023-01-01	11:38:36	109	Korean Beef Bowl	Asian	17.95
2	2	2	2023-01-01	11:57:40	108	Tofu Pad Thai	Asian	14.50
3	3	2	2023-01-01	11:57:40	124	Spaghetti	Italian	14.50
4	4	2	2023-01-01	11:57:40	117	Chicken Burrito	Mexican	12.95
5	5	2	2023-01-01	11:57:40	129	Mushroom Ravioli	Italian	15.50
6	6	2	2023-01-01	11:57:40	106	French Fries	American	7.00
7	7	3	2023-01-01	12:12:28	117	Chicken Burrito	Mexican	12.95
8	8	3	2023-01-01	12:12:28	119	Chicken Torta	Mexican	11.95
9	9	4	2023-01-01	12:16:31	117	Chicken Burrito	Mexican	12.95
10	10	5	2023-01-01	12:21:30	117	Chicken Burrito	Mexican	12.95
11	11	6	2023-01-01	12:29:36	101	Hamburger	American	12.95
12	12	6	2023-01-01	12:29:36	114	Potstickers	Asian	9.00
13	13	7	2023-01-01	12:50:37	123	Chips & Guacamole	Mexican	9.00
14	14	8	2023-01-01	12:51:37	123	Chips & Guacamole	Mexican	9.00
15	15	9	2023-01-01	12:52:01	108	Tofu Pad Thai	Asian	14.50

View all records from combined_table with a total of 12234 rows.



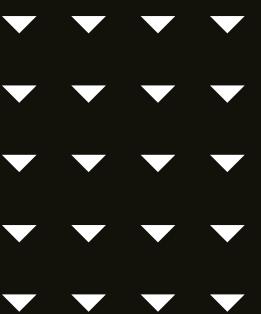
**THE DATA IS
NOW CLEAN
AND READY
TO BE
ANALYZED**





02. SQL QUERY ANALYSIS

SQL ANALYSIS



1. What were the least and most ordered items? What categories were they in?

```
111 --Recommended Analysis
112 -- 1. What were the least and most ordered items? What categories were they in?
113 -- Barang apa yang paling sedikit dan paling banyak dipesan? Di kategori apa mereka berada?
114
115 WITH CountsItem AS (
116     SELECT
117         item_id,
118         item_name,
119         category,
120         COUNT(*) AS order_count,
121         RANK() OVER (ORDER BY COUNT(*) ASC) AS min_rank,
122         RANK() OVER (ORDER BY COUNT(*) DESC) AS max_rank
123     FROM combined_table
124     GROUP BY item_id, item_name, category
125 )
126 SELECT
127     item_id,
128     item_name,
129     category,
130     order_count
131 FROM CountsItem
132 WHERE min_rank = 1 OR max_rank = 1;
```

Query

Shows Query results with the result item_name Chicken Tacos and the Mexican category with a total of 123 orders is the menu that was ordered the least. While, item_name Hamburger and the American category with a total of 759 orders were the most ordered menu items



```
111 --Recommended Analysis
112 -- 1. What were the least and most ordered items? What categories were they in?
113 -- Barang apa yang paling sedikit dan paling banyak dipesan? Di kategori apa mereka berada?
114
```

Data Output Messages Notifications

	item_id smallint	item_name character varying (45)	category character varying (45)	order_count bigint
1	115	Chicken Tacos	Mexican	123
2	101	Hamburger	American	759

SQL ANALYSIS

2. What do the highest spend orders look like? Which items did they buy and how much did they spend?

The screenshot shows a SQL query editor with the following details:

- Query Text:**

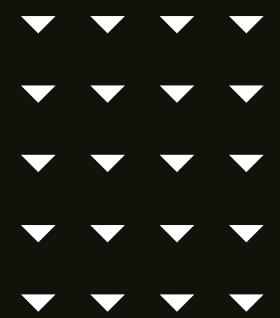
```
-- 2. What do the highest spend orders look like? Which items did they buy and how much did they spend?  
-- Seperti apa pesanan pembelanjaan tertinggi? Barang apa yang mereka beli dan berapa banyak yang mereka belanjakan?  
SELECT  
    order_id,  
    item_name,  
    COUNT(order_id) AS total_pembelian,  
    SUM(price) AS total_pembelian_harga  
FROM  
    combined_table  
GROUP BY  
    order_id, item_name  
ORDER BY  
    total_pembelian_harga DESC;
```
- Data Output:** The results show 5 rows of data, ordered by total purchase price in descending order.
- Table Headers:** order_id, item_name, total_pembelian, total_pembelian_harga.
- Sample Data:**

	order_id	item_name	total_pembelian	total_pembelian_harga
1	1639	Korean Beef Bowl	3	53.85
2	2364	Meat Lasagna	3	53.85
3	740	Chicken Parmesan	3	53.85
4	4314	Eggplant Parmesan	3	50.85
5	3940	Eggplant Parmesan	3	50.85
- Message Bar:** Total rows: 1000 of 11917 | Query complete 00:00:00.418 | Ln 146, Col 31

Query

Shows there are 11917 rows with total purchases from the highest and lowest prices. This image displays the top 5 data showing order_id, item_name, total_purchase_and total_purchase_price by customers with each order_id.

SQL ANALYSIS



2. What do the highest spend orders look like? Which items did they buy and how much did they spend?

```
147 LIMIT 1;
```

Data Output Messages Notifications

order_id item_name total_pembelian total_pembelian_harga

	order_id	item_name	total_pembelian	total_pembelian_harga
1	1639	Korean Beef Bowl	3	53.85

Use Query LIMIT 1 to find the largest 1 data record.

```
147 LIMIT 5;
```

Data Output Messages Notifications

order_id item_name total_pembelian total_pembelian_harga

	order_id	item_name	total_pembelian	total_pembelian_harga
1	2364	Meat Lasagna	3	53.85
2	1639	Korean Beef Bowl	3	53.85
3	740	Chicken Parmesan	3	53.85
4	4314	Eggplant Parmesan	3	50.85
5	3940	Eggplant Parmesan	3	50.85

Use Query LIMIT 5 to find the largest 5 data record.

```
147 LIMIT 3;
```

Data Output Messages Notifications

order_id item_name total_pembelian total_pembelian_harga

	order_id	item_name	total_pembelian	total_pembelian_harga
1	1639	Korean Beef Bowl	3	53.85
2	740	Chicken Parmesan	3	53.85
3	2364	Meat Lasagna	3	53.85

Use Query LIMIT 3 to find the largest 3 data record.

SQL ANALYSIS

3. Were there certain times that had more or less orders?

```
149 -- 3. Were there certain times that had more or less orders?  
150 -- Apakah ada waktu-waktu tertentu yang pesanannya lebih banyak atau lebih sedikit?  
151  
152 SELECT  
153     EXTRACT(HOUR FROM order_time) AS jam,  
154     COUNT(order_id) AS jumlah_pesanan  
155 FROM  
156     combined_table  
157 GROUP BY  
158     jam  
159 ORDER BY  
160     jumlah_pesanan DESC;
```

Query

A screenshot of a SQL query results window. The results are displayed in a table with two columns: 'jam' (hour) and 'jumlah_pesanan' (number of orders). The data shows the following rows:

jam	jumlah_pesanan
12	1672
13	1575
17	1370
18	1307
19	1085
16	1054

Total rows: 14 of 14 Query complete 00:00:00.133 Ln 161, Col 1

Shows that there are 14 data records with the highest number of orders at 12 o'clock and the number of orders is 1672 orders

A screenshot of a SQL query results window. The results are displayed in a table with two columns: 'jam' (hour) and 'jumlah_pesanan' (number of orders). The data shows the following rows:

jam	jumlah_pesanan
10	5
23	11
22	309
21	608
11	630
15	751

Total rows: 14 of 14 Query complete 00:00:00.167

Meanwhile, the minimum order is at 10 o'clock with a total of 5 orders using ASC Query

SQL ANALYSIS

4. Which cuisines should we focus on developing more menu items for based on the data?

```
173 -- 4. Which cuisines should we focus on developing more menu items for based on the data?  
174 -- Masakan mana yang harus kami fokuskan untuk mengembangkan lebih banyak item menu berdasarkan data?  
175 SELECT  
176     category,  
177     COUNT(order_id) AS jumlah_pesanan  
178 FROM  
179     combined_table  
180 GROUP BY  
181     category  
182 ORDER BY  
183     jumlah_pesanan DESC;  
184
```

Query for category then calculates order_id as the number of orders and orders them from largest.

	category	jumlah_pesanan
1	Asian	3470
2	Italian	2948
3	Mexican	2945
4	American	2871

Shows that the Asia category received the largest number of orders with a total of 3470 orders. This category can be developed into more types of menus from the Asian category because most customers like Asian cuisine.

SQL ANALYSIS

5. View the details of the highest spend order. Which specific items were purchased?

```
195 -- TAMBAHAN 2 ANALISIS --
196 -- 5. View the details of the highest spend order. Which specific items were purchased?
197 -- Lihat detail pesanan pembelanjaan tertinggi. Barang spesifik apa yang dibeli?
198 SELECT
199   od.order_id,
200   mi.item_name,
201   od.order_date,
202   od.order_time,
203   mi.category,
204   mi.price,
205   (
206     SELECT
207       SUM(price)
208     FROM
209       combined_table
210     WHERE
211       order_id = od.order_id
212   ) AS total_cost
213 FROM
214   combined_table AS od
215 JOIN
216   menu_items AS mi ON od.item_id = mi.menu_item_id
217 WHERE
218   od.order_id = (
219     SELECT
220       od.order_id
221     FROM
222       combined_table
223     GROUP BY
224       order_id
225     ORDER BY
226       SUM(price) DESC
227     LIMIT 1
228   );
229
```

Query

	order_id	item_name	order_date	order_time	category	price	total_cost
	smallint	character varying (45)	date	time without time zone	character varying (45)	numeric (5,2)	numeric
1	4482	Cheeseburger	2023-03-17	13:57:38	American	13.95	197.45
2	4482	Tofu Pad Thai	2023-03-17	13:57:38	Asian	14.50	197.45
3	4482	Spaghetti & Meatballs	2023-03-17	13:57:38	Italian	17.95	197.45
4	4482	Korean Beef Bowl	2023-03-17	13:57:38	Asian	17.95	197.45
5	4482	Chicken Burrito	2023-03-17	13:57:38	Mexican	12.95	197.45
6	4482	Chicken Burrito	2023-03-17	13:57:38	Mexican	12.95	197.45
7	4482	Mushroom Ravioli	2023-03-17	13:57:38	Italian	15.50	197.45
8	4482	California Roll	2023-03-17	13:57:38	Asian	11.95	197.45
9	4482	Steak Burrito	2023-03-17	13:57:38	Mexican	14.95	197.45
10	4482	Steak Torta	2023-03-17	13:57:38	Mexican	13.95	197.45
11	4482	Steak Torta	2023-03-17	13:57:38	Mexican	13.95	197.45
12	4482	Mac & Cheese	2023-03-17	13:57:38	American	7.00	197.45
13	4482	Eggplant Parmesan	2023-03-17	13:57:38	Italian	16.95	197.45
14	4482	Hamburger	2023-03-17	13:57:38	American	12.95	197.45

The customer with order_id 4482 is the customer with the highest total purchase is 197.45 in one order, ordering 14 different menus from various categories. From these results it can be seen that there are customers who are loyal to the restaurant and to develop this, you can make these customers become members of the restaurant and get special discount.

SQL ANALYSIS

6. How many orders were placed on each date?

Query

```
274 --6. How many orders were placed on each date?  
275 -- Berapa jumlah pesanan yang dilakukan pada setiap tanggal?  
276 SELECT  
277     DATE(order_date) AS tanggal,  
278     COUNT(DISTINCT order_id) AS jumlah_pesanan  
279 FROM  
280     combined_table  
281 GROUP BY  
282     tanggal  
283 ORDER BY  
284     tanggal;  
---
```

Data Output Messages Notifications

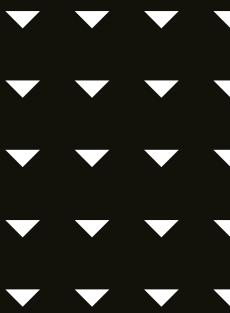


	tanggal date	jumlah_pesanan bigint
1	2023-01-01	69
2	2023-01-02	67
3	2023-01-03	66
4	2023-01-04	52
5	2023-01-05	54
6	2023-01-06	64
7	2023-01-07	58
~ ~ ~ ~ ~		

Total rows: 90 of 90 Query complete 00:00:00.182

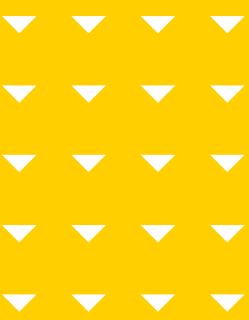
Ln 274, Col 1

There are 90 rows consisting of dates and order amounts sequentially by date.



03. INTEGRATION WITH PYTHON AND VISUALIZATION

PYTHON



1. Create a file in .ipynb format, for example (analysis_restaurant.ipynb), install the necessary packages such as: psycopg2-binary, matplotlib, seaborn, pandas, etc. Then do the import like then run cell

```
import pandas as pd
import numpy as np
import psycopg2
import matplotlib.pyplot as plt
import seaborn as sns
```

[1] Python

2. Create a connection to connect to the database, make sure the username and others are correct then run.

```
conn = psycopg2.connect(
    dbname="restaurant_db",
    user="postgres",
    password="",
    host="localhost",
    port="5432"
)

if conn:
    print("Connected to the database")
```

[2] Python

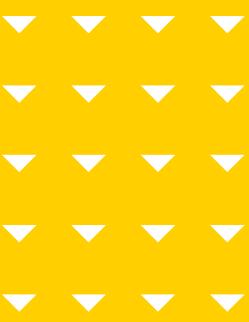
... Connected to the database

3. Create a python function to run SQL queries.

```
##function for execute sql
def execute_sql_query(sql, conn):
    return pd.read_sql_query(sql, conn)
```

[3] Python

PYTHON



4. Read the table contained in the database with a query

```
##search table name from database

sql = """ SELECT table_name
    FROM information_schema.tables
    WHERE table_schema = 'public';
"""

data_table = execute_sql_query(sql,conn)
data_table
```

[5] C:\Users\DWI Winda Fitrika\AppData\Local\Temp\ipykernel_5980\1194220348.py:3: UserWarning: pandas only supports SQLAlchemy connectable (engine) objects, not Connection objects. This warning is displayed once per session.

```
...     return pd.read_sql_query(sql, conn)
```

[6] table_name

	table_name
0	order_details
1	menu_items
2	combined_table

There are 3 tables and the one that will be used is the combined_table table which was previously created in pgadmin.

5. View data records in the combined_table table. There are 12234 rows with 8 columns

```
## select table combined_table from join between table order_details and menu_items
sql = """ SELECT * FROM combined_table;
"""

data = execute_sql_query(sql,conn)
data
```

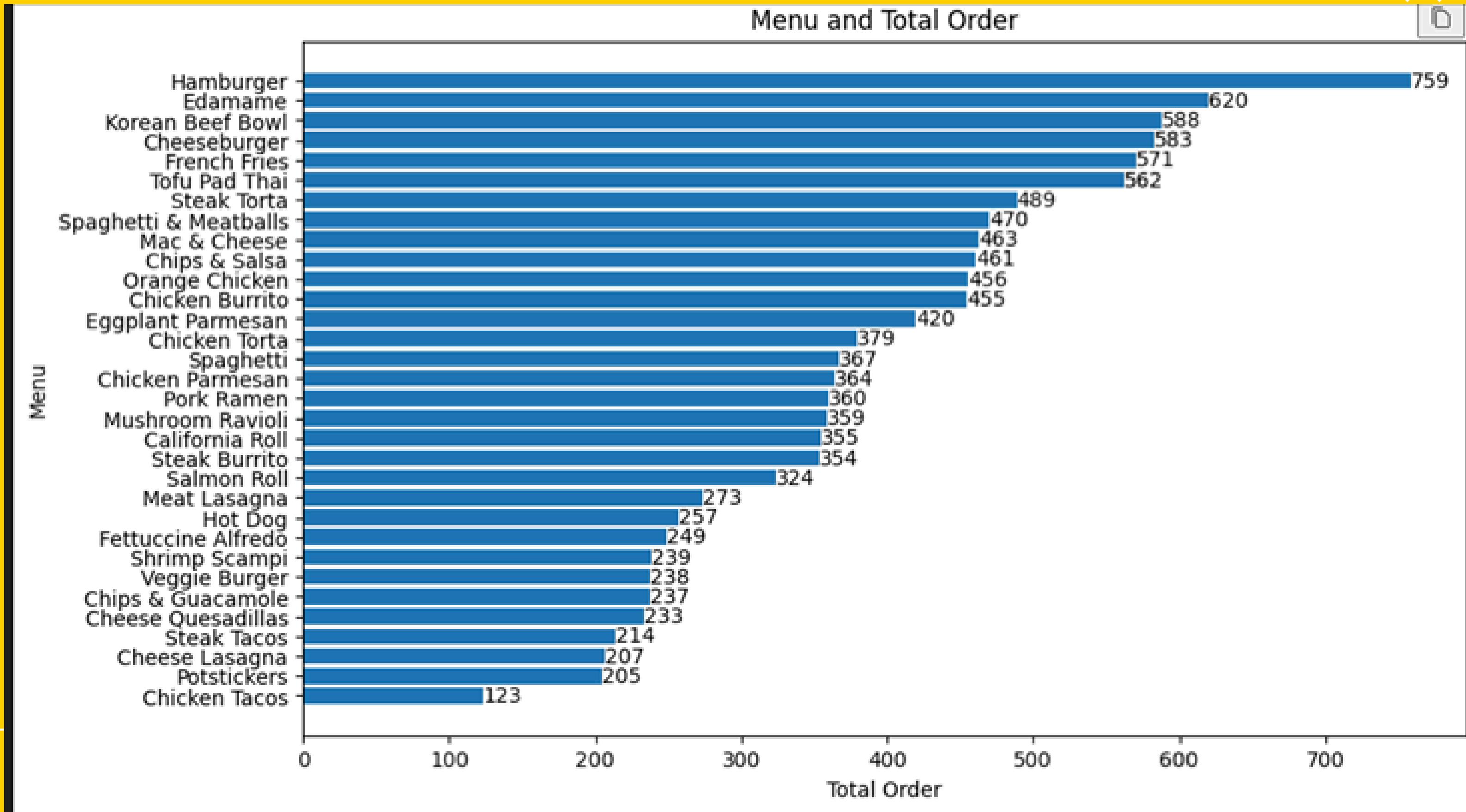
[6] C:\Users\DWI Winda Fitrika\AppData\Local\Temp\ipykernel_5980\1194220348.py:3: UserWarning: pandas only supports SQLAlchemy connectable (engine) objects, not Connection objects. This warning is displayed once per session.

```
...     return pd.read_sql_query(sql, conn)
```

order_details_id	order_id	order_date	order_time	item_id	item_name	category	price	
0	1	2023-01-01	11:38:36	109	Korean Beef Bowl	Asian	17.95	
1	2	2023-01-01	11:57:40	108	Tofu Pad Thai	Asian	14.50	
2	3	2023-01-01	11:57:40	124	Spaghetti	Italian	14.50	
3	4	2023-01-01	11:57:40	117	Chicken Burrito	Mexican	12.95	
4	5	2023-01-01	11:57:40	129	Mushroom Ravioli	Italian	15.50	
...	
12229	11717	5149	2023-03-28	14:48:50	101	Hamburger	American	12.95
12230	11904	5225	2023-03-29	17:40:52	101	Hamburger	American	12.95
12231	11907	5226	2023-03-29	17:43:56	101	Hamburger	American	12.95
12232	12022	5281	2023-03-30	16:56:04	101	Hamburger	American	12.95
12233	12064	5303	2023-03-30	21:15:27	101	Hamburger	American	12.95

12234 rows × 8 columns

VISUALIZATION



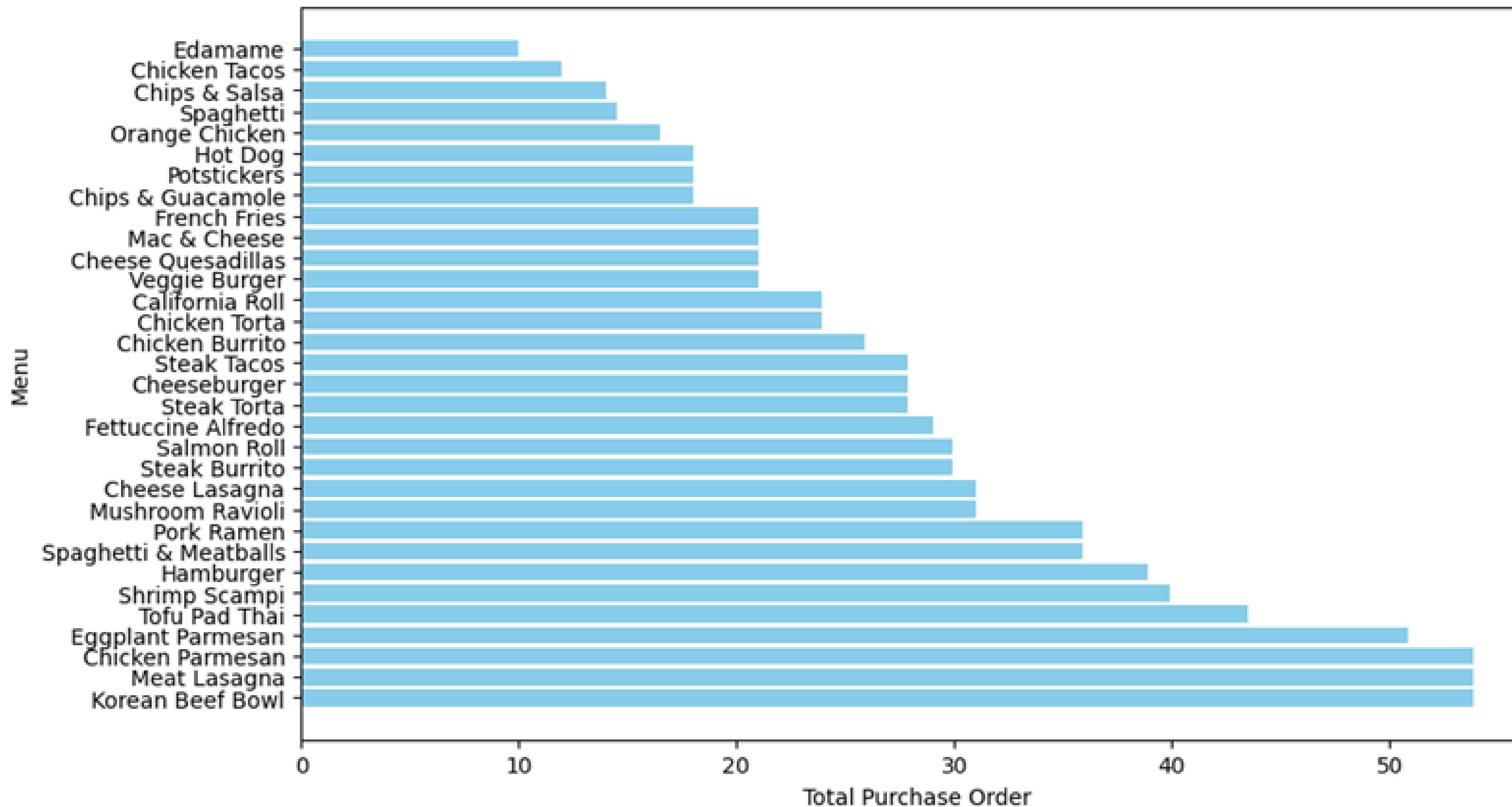
CONCLUSION

ITEM_NAME CHICKEN TACOS AND THE MEXICAN CATEGORY WITH A TOTAL OF 123 ORDERS ARE THE MENU ITEMS ORDERED THE LEAST.

MEANWHILE, ITEM_NAME HAMBURGER AND THE AMERICAN CATEGORY WITH A TOTAL OF 759 ORDERS WERE THE MOST ORDERED MENU ITEMS.

VISUALIZATION

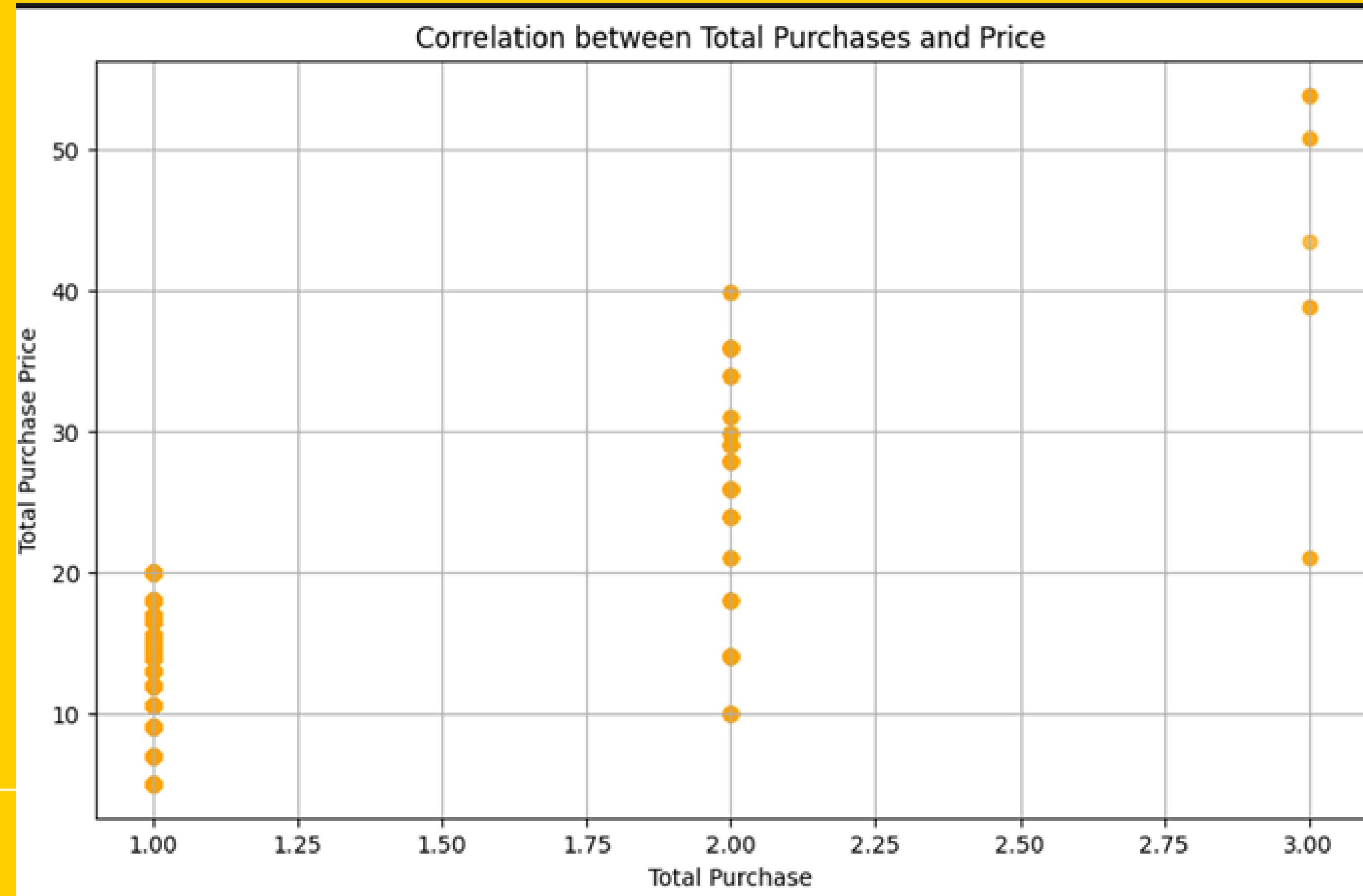
Menu and Total Purchase Order



CONCLUSION

**THERE ARE 3 MENUS WITH THE
SAME TOTAL PURCHASE PRICE BY
DIFFERENT CUSTOMERS, NAMELY
CHICKEN PARMESAN, MEAT
LASAGNA, KOREAN BEEF BOWL
WITH A TOTAL PRICE OF MORE
THAN 50.**

VISUALIZATION

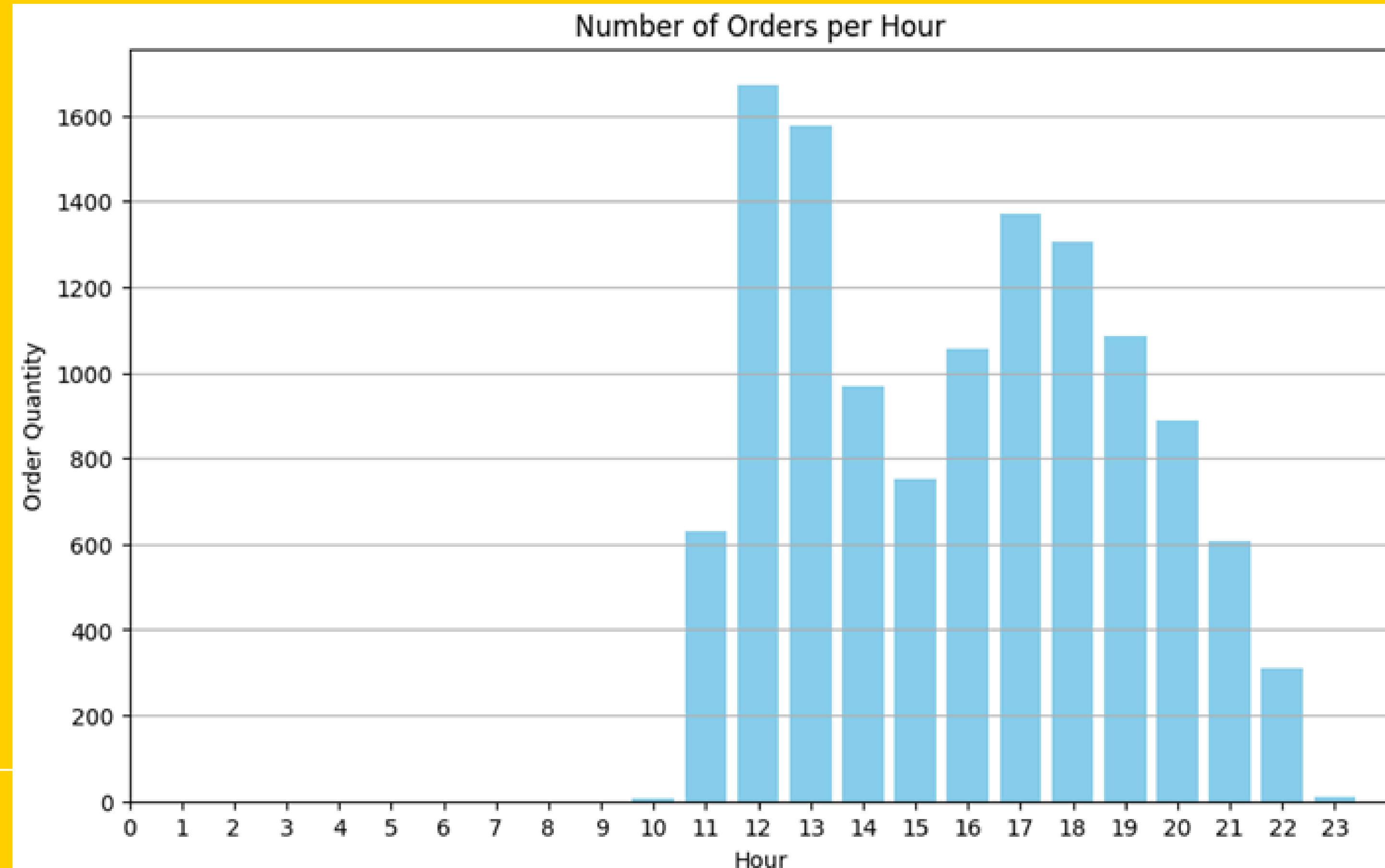


CONCLUSION

THIS VISUALIZATION ILLUSTRATES THE CORRELATION BETWEEN TOTAL PURCHASES OF A PRODUCT AND THE TOTAL PURCHASE PRICE. ON THE X-AXIS, WE HAVE TOTAL PURCHASES, WHICH IS THE NUMBER OF UNITS OF THE PRODUCT PURCHASED BY CUSTOMERS. MEANWHILE, THE Y-AXIS SHOWS THE TOTAL PURCHASE PRICE, WHICH REFLECTS THE AMOUNT OF MONEY CUSTOMERS SPEND TO BUY THESE PRODUCTS.

FROM THIS GRAPH, IT CAN BE SEEN THAT THERE IS A STRONG POSITIVE TREND BETWEEN TOTAL PURCHASES AND TOTAL PURCHASE PRICE. THIS MEANS THAT THE MORE PRODUCT UNITS PURCHASED, THE HIGHER THE TOTAL PURCHASE PRICE. THIS CAN BE EXPLAINED BY THE ASSUMPTION THAT THE MORE PRODUCTS PURCHASED, THE MORE LIKELY CUSTOMERS ARE TO SPEND MORE MONEY.

VISUALIZATION

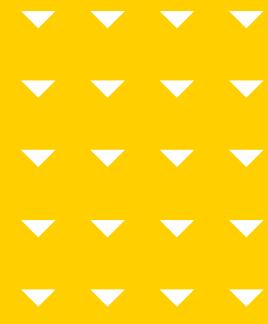




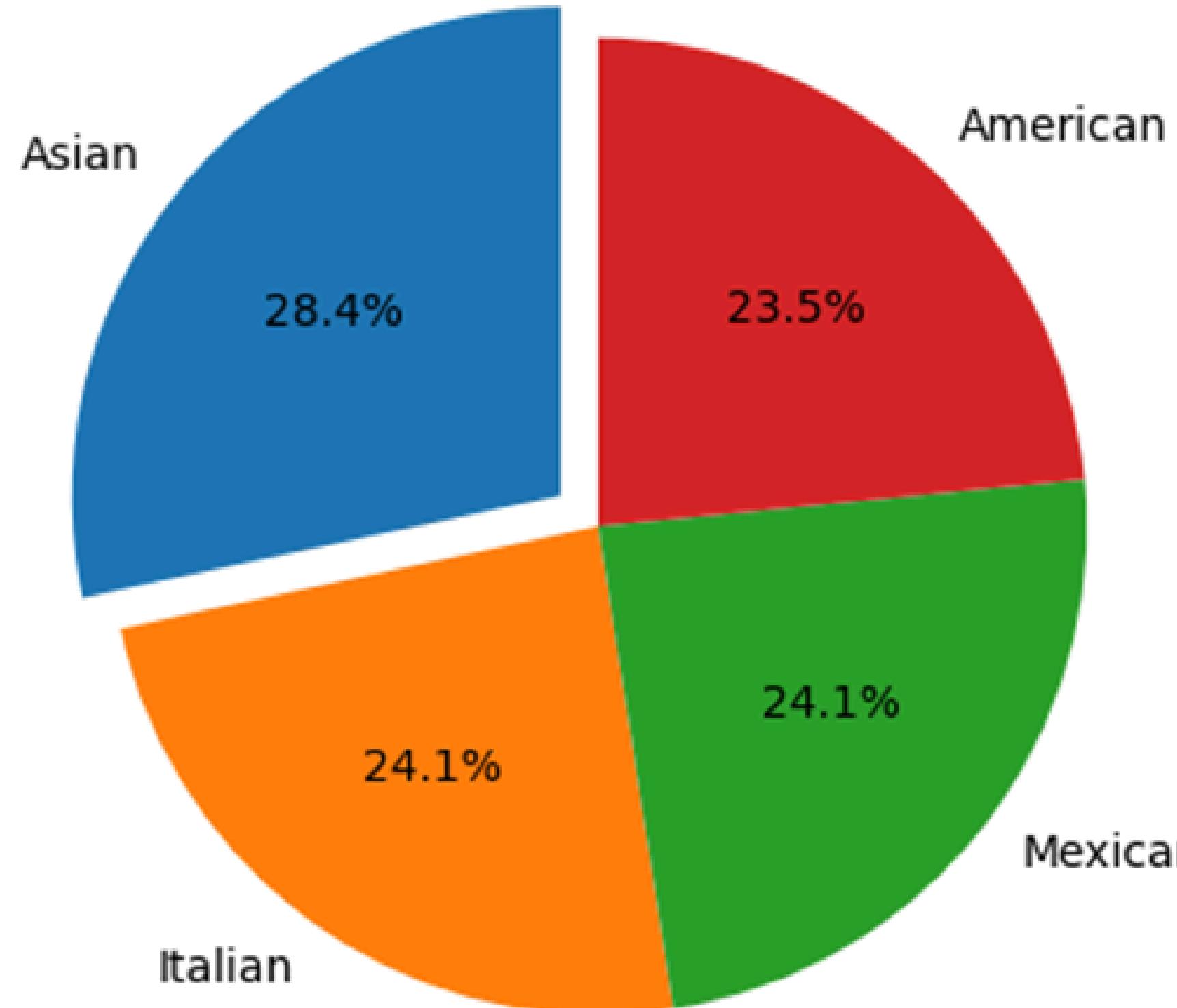
CONCLUSION

THE TIME THAT GETS THE MOST ORDERS IS BETWEEN 12 AND 13 NOON, THIS IS PROBABLY LUNCH TIME SO THERE ARE MORE ORDERS AT THAT TIME.

VISUALIZATION



Total Order by Category

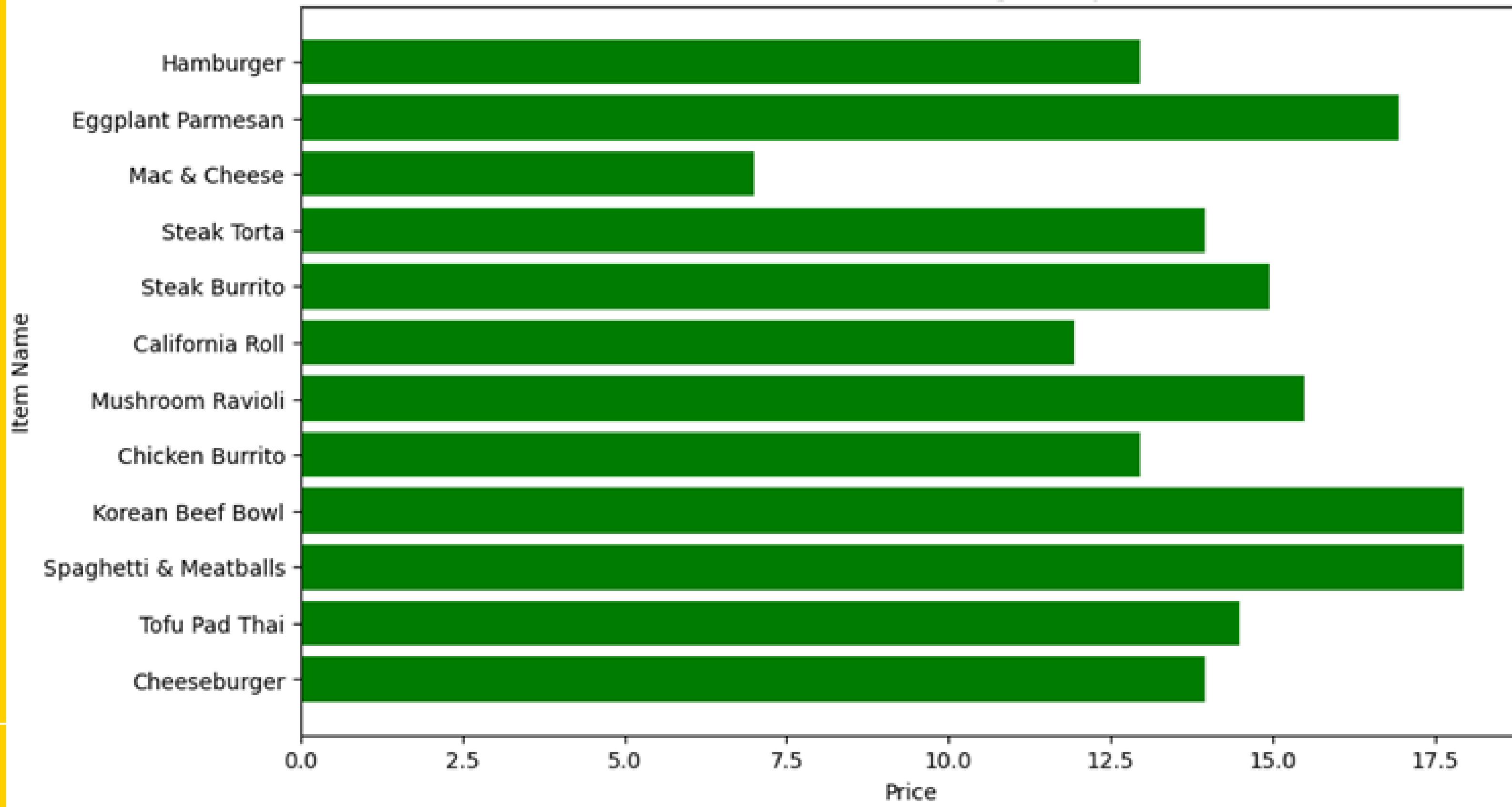


CONCLUSION

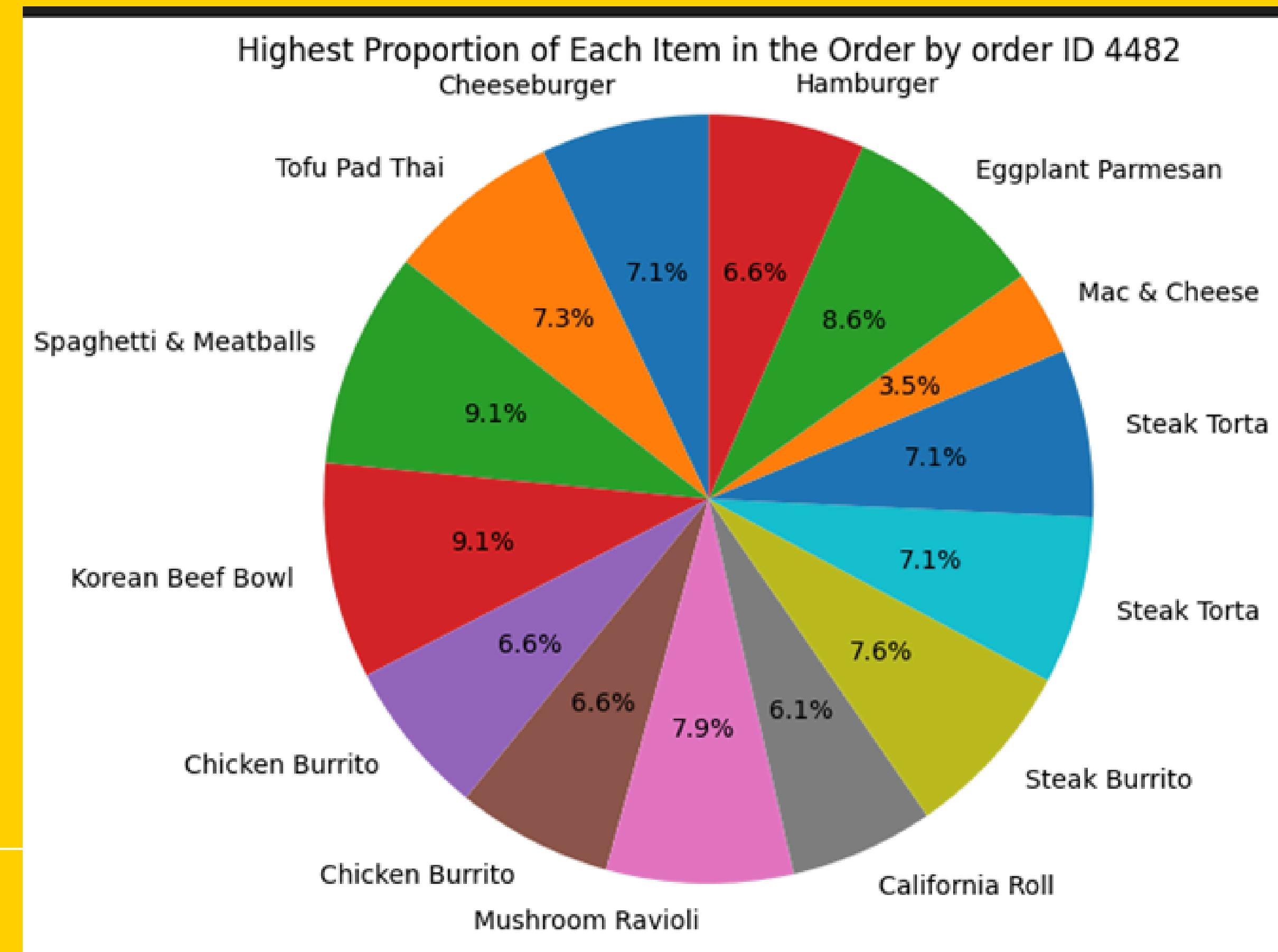
SHOWS THAT THE ASIA CATEGORY RECEIVED THE LARGEST NUMBER OF ORDERS WITH A TOTAL OF 3470 ORDERS. THIS CATEGORY CAN BE DEVELOPED INTO MORE TYPES OF MENUS FROM THE ASIAN CATEGORY BECAUSE MOST CUSTOMERS LIKE ASIAN CUISINE. VISUALIZATION USING PIE CHART.

VISUALIZATION

Price of Each Item in the Order is Highest by order ID 4482



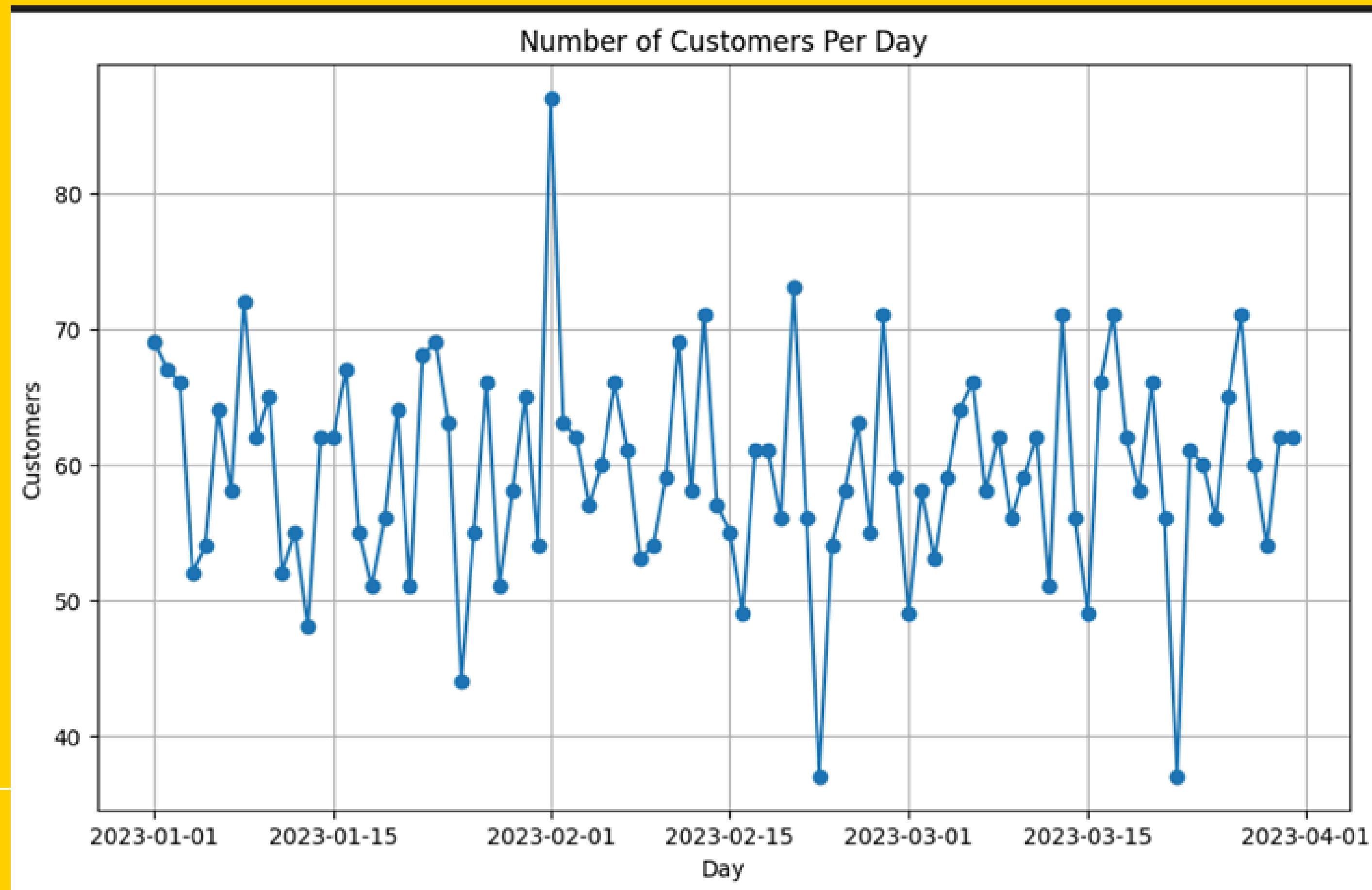
VISUALIZATION



CONCLUSION

SHOW THE PRICE OF EACH ITEM PURCHASED BY CUSTOMERS WITH THE MOST ORDER_ID 4482, THE HIGHEST PRICES ARE 2 ITEMS, NAMELY SPAGHETTI & MEATBALLS AND KOREAN BEEF BOWL, THE LOWEST PRICE IS ON THE STEAK TORTA MENU USING THE BAR CHART AND PIE CHART.

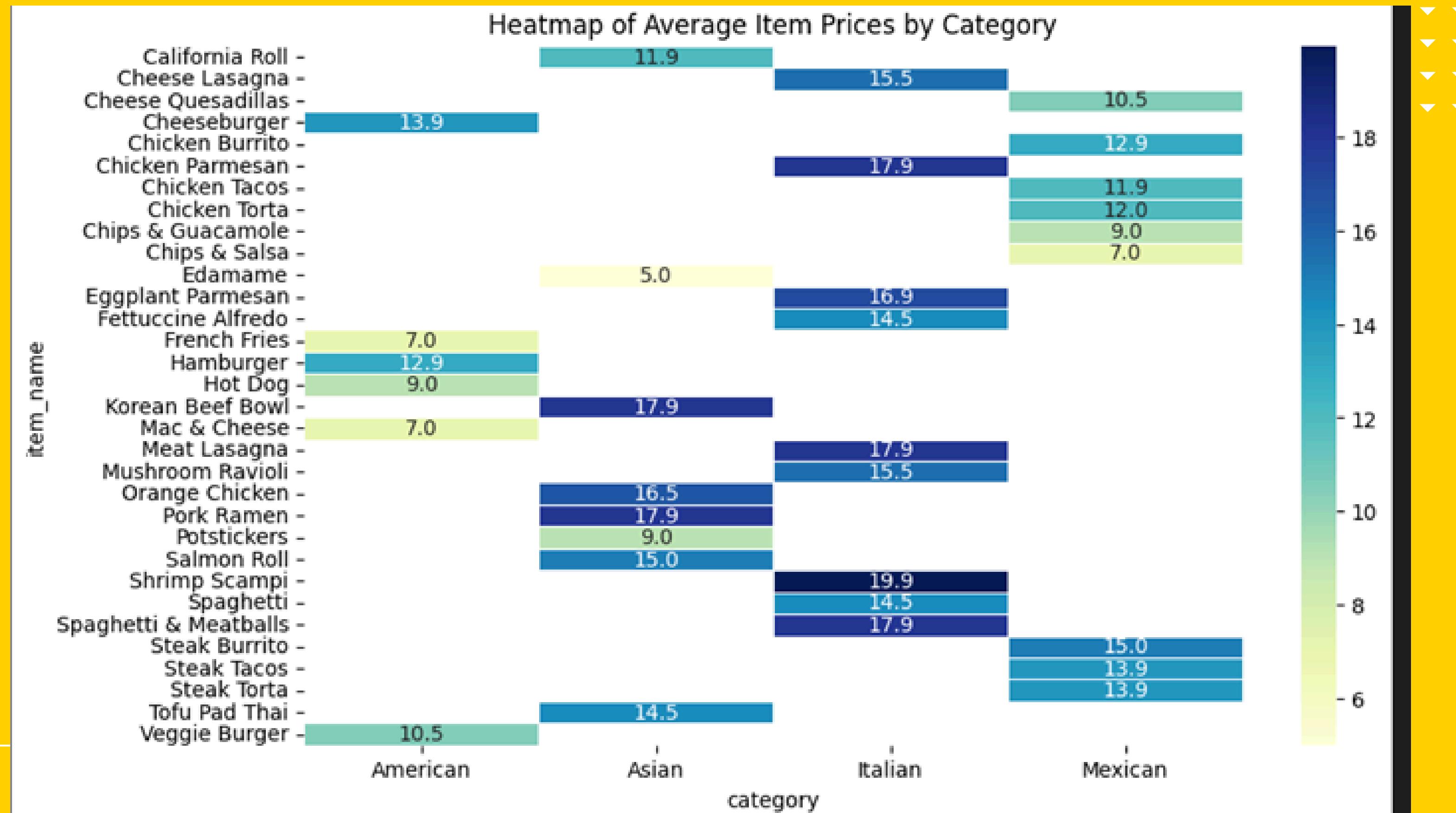
VISUALIZATION



CONCLUSION

THE HIGHEST ORDERS OCCUR ON THE 1ST OF FEBRUARY. THE LOWEST ORDERS OCCUR IN THE MIDDLE OF FEBRUARY AND MARCH

VISUALIZATION



CONCLUSION

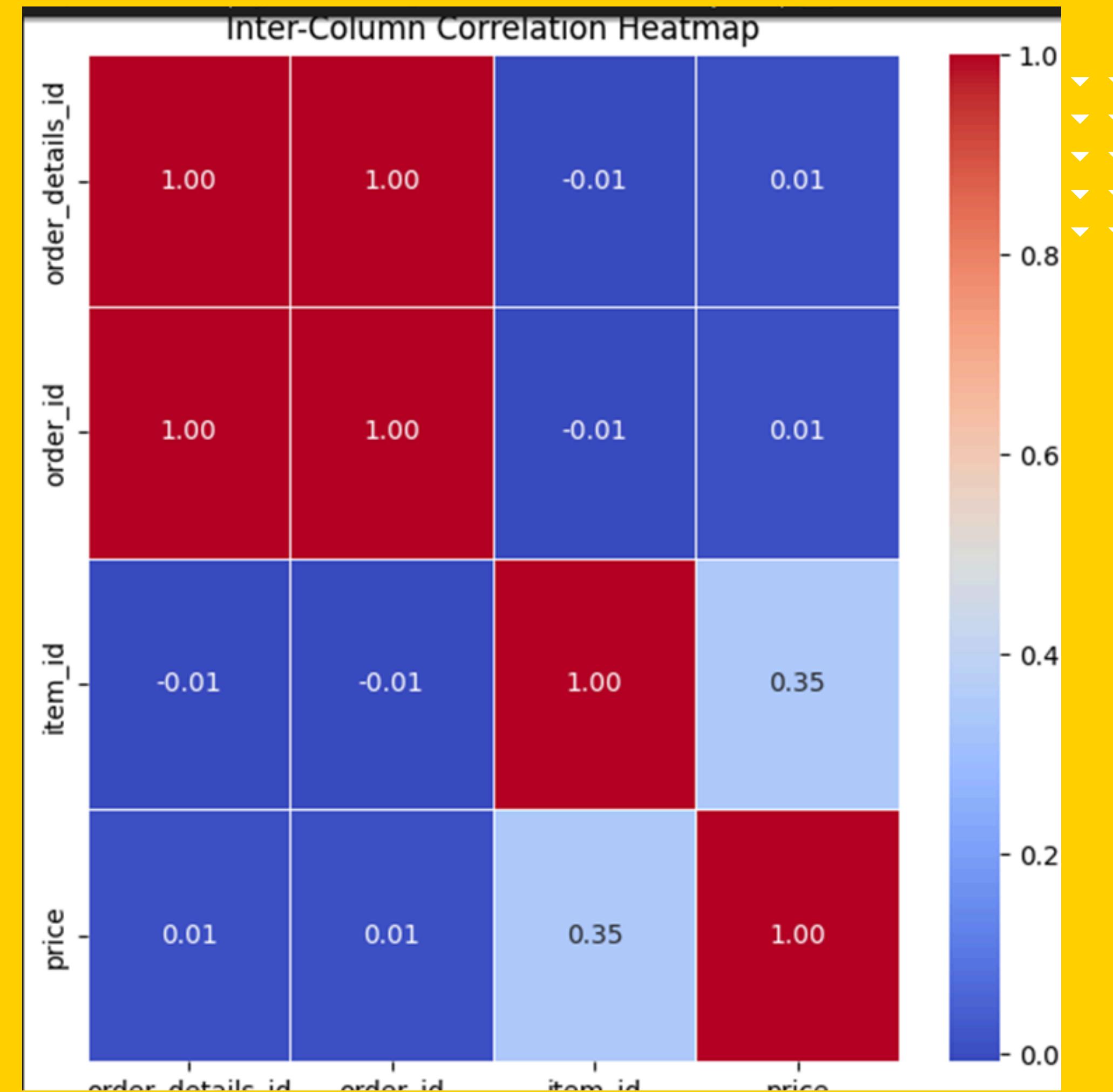
THIS VISUALIZATION IS A HEATMAP THAT DISPLAYS THE AVERAGE PRICE OF ITEMS BASED ON THEIR CATEGORY.

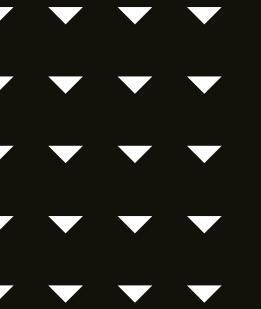
EACH ROW ON THE HEATMAP REPRESENTS AN ITEM, WHILE EACH COLUMN REPRESENTS THE CATEGORY OF THAT ITEM. THE INTENSITY OF THE COLOR IN EACH CELL INDICATES THE AVERAGE PRICE OF ITEMS IN THAT CATEGORY, WITH LIGHTER COLORS INDICATING HIGHER PRICES.

WITH THIS VISUALIZATION, WE CAN QUICKLY IDENTIFY WHICH CATEGORIES HAVE HIGHER OR LOWER AVERAGE ITEM PRICES COMPARED TO OTHER CATEGORIES.

VISUALIZATION

This visualization is a heatmap that displays the correlation matrix between numerical columns in the dataset. Each cell in the heatmap depicts the degree of correlation between two columns, with the correlation value displayed as a number within each cell.





THANK YOU

Download Dataset :

<https://maven-datasets.s3.amazonaws.com/Restaurant+Orders/Restaurant+Orders+MySQL.zip>

Google Colaboratory :

https://colab.research.google.com/drive/10-ECPWWNPr1FtLVunVvY6FoJT4zaOxVn?usp=drive_link