databricks Food Analysis

(https://databricks.com)

Project Python Foundations: FoodHub Data Analysis

Business Problem Overview

The number of restaurants in New York is increasing day by day. Lots of students and busy professionals rely on those restaurants due to their hectic lifestyles. Online food delivery service is a great option for them. It provides them with good food from their favorite restaurants. A food aggregator company FoodHub offers access to multiple restaurants through a single smartphone app.

The app allows the restaurants to receive a direct online order from a customer. The app assigns a delivery person from the company to pick up the order after it is confirmed by the restaurant. The delivery person then uses the map to reach the restaurant and waits for the food package. Once the food package is handed over to the delivery person, he/she confirms the pick-up in the app and travels to the customer's location to deliver the food. The delivery person confirms the drop-off in the app after delivering the food package to the customer. The customer can rate the order in the app. The food aggregator earns money by collecting a fixed margin of the delivery order from the restaurants.

Objective

The food aggregator company has stored the data of the different orders made by the registered customers in their online portal. They want to analyze the data to get a fair idea about the demand of different restaurants which will help them in enhancing their customer experience. Suppose you are hired as a Data Scientist in this company and the Data Science team has shared some of the key questions that need to be answered. Perform the data analysis to find answers to these questions that will help the company to improve the business.

Solution Approach

- Understand the demand of restaurants in the FoodHub portal
- Cuisine preference of the New York customers
- Get an idea about the cost of the ordered food
- Understand the volume of the orders over weekdays and weekends
- Estimate the revenue generated by the company
- Help the company to take decision on promotional offers
- Order rating analysis

Data Description

The data contains the different data related to a food order. The detailed data dictionary is given below.

Data Dictionary

- order_id: Unique ID of the order
- customer_id: ID of the customer who ordered the food
- restaurant_name: Name of the restaurant
- cuisine_type: Cuisine ordered by the customer
- cost: Cost of the order
- day_of_the_week: Indicates whether the order is placed on a weekday or weekend (The weekday is from Monday to Friday and the weekend is Saturday and Sunday)
- rating: Rating given by the customer out of 5
- food_preparation_time: Time (in minutes) taken by the restaurant to prepare the food. This is calculated by taking the difference between the timestamps of the restaurant's order confirmation and the delivery person's pick-up confirmation.
- delivery_time: Time (in minutes) taken by the delivery person to deliver the food package.This is calculated by taking the difference between the timestamps of the delivery person's pick-up confirmation and drop-off information

Import the required libraries

```
from pyspark.sql import functions as f
import matplotlib.pyplot as plt
```

```
%fs ls
```

Table

	path	name	size	modificationTime
1	dbfs:/FileStore/	FileStore/	0	0
2	dbfs:/databricks-datasets/	databricks-datasets/	0	0
3	dbfs:/databricks-results/	databricks-results/	0	0
4	dbfs:/user/	user/	0	0

4 rows

```
%fs ls dbfs:/FileStore/
```

Table

	path	name	size	modificationTime
1	dbfs:/FileStore/Adani_Share.csv	Adani_Share.csv	94126	1687155176000
2	dbfs:/FileStore/tables/	tables/	0	0

2 rows

```
%fs ls dbfs:/FileStore/tables/
```

Table

	path	name	size	modificationTime
1	dbfs:/FileStore/tables/BigDataSet/	BigDataSet/	0	0
2	dbfs:/FileStore/tables/Covid-19/	Covid-19/	0	0
3	dbfs:/FileStore/tables/Datasets-1.csv	Datasets-1.csv	864813	1692099350000
4	dbfs:/FileStore/tables/Datasets-2.csv	Datasets-2.csv	864813	1692099758000
5	dbfs:/FileStore/tables/Datasets.csv	Datasets.csv	864813	1692096320000
6	dbfs:/FileStore/tables/Ecommerce.csv	Ecommerce.csv	231599	1684814738000
7	dbfs:/FileStore/tables/Financials.csv	Financials.csv	121839	1693897705000

18 rows

```
datasets = spark.read.csv(
    "dbfs:/FileStore/tables/foodhub_order.csv",
                          header = True,
                          inferSchema=True
                          )
```

```
df = datasets.alias("copy")
```

```
df.show()
```

```
+--------+-----------+--------------------+-------------+-----------------+---------------+---------+----------------
-----+-------------+
|order_id|customer_id|     restaurant_name| cuisine_type|cost_of_the_order|day_of_the_week|   rating|food_preparation
_time|delivery_time|
+--------+-----------+--------------------+-------------+-----------------+---------------+---------+----------------
-----+-------------+
| 1477147|     337525|             Hangawi|       Korean|            30.75|        Weekend|Not given|
25|           20|
| 1477685|     358141|Blue Ribbon Sushi...|     Japanese|            12.08|        Weekend|Not given|
25|           23|
| 1477070|      66393|         Cafe Habana|      Mexican|           12.23|        Weekday|        5|
23|          28|
| 1477334|     106968|Blue Ribbon Fried...|     American|            29.2|        Weekend|        3|
25|          15|
| 1478249|      76942|    Dirty Bird to Go|     American|           11.59|        Weekday|        4|
25|          24|
| 1477224|     147468|    Tamarind TriBeCa|       Indian|           25.22|        Weekday|        3|
20|          24|
| 1477894|     157711|   The Meatball Shop|      Italian|            6.07|        Weekend|Not given|
28|          21|
| 1477858|      89574|           Barbounia|Mediterranean|            5.97|        Weekday|        3|
```

```
df.head(5)
```

```
Out[5]: [Row(order_id=1477147, customer_id=337525, restaurant_name='Hangawi', cuisine_type='Korean', cost_of_the_orde
r=30.75, day_of_the_week='Weekend', rating='Not given', food_preparation_time=25, delivery_time=20),
 Row(order_id=1477685, customer_id=358141, restaurant_name='Blue Ribbon Sushi Izakaya', cuisine_type='Japanese', cost
_of_the_order=12.08, day_of_the_week='Weekend', rating='Not given', food_preparation_time=25, delivery_time=23),
 Row(order_id=1477070, customer_id=66393, restaurant_name='Cafe Habana', cuisine_type='Mexican', cost_of_the_order=1
2.23, day_of_the_week='Weekday', rating='5', food_preparation_time=23, delivery_time=28),
 Row(order_id=1477334, customer_id=106968, restaurant_name='Blue Ribbon Fried Chicken', cuisine_type='American', cost
_of_the_order=29.2, day_of_the_week='Weekend', rating='3', food_preparation_time=25, delivery_time=15),
 Row(order_id=1478249, customer_id=76942, restaurant_name='Dirty Bird to Go', cuisine_type='American', cost_of_the_or
der=11.59, day_of_the_week='Weekday', rating='4', food_preparation_time=25, delivery_time=24)]
```

```
df.tail(5)
```

```
Out[6]: [Row(order_id=1476701, customer_id=292602, restaurant_name='Chipotle Mexican Grill $1.99 Delivery', cuisine_t
ype='Mexican', cost_of_the_order=22.31, day_of_the_week='Weekend', rating='5', food_preparation_time=31, delivery_tim
e=17),
 Row(order_id=1477421, customer_id=397537, restaurant_name='The Smile', cuisine_type='American', cost_of_the_order=1
2.18, day_of_the_week='Weekend', rating='5', food_preparation_time=31, delivery_time=19),
 Row(order_id=1477819, customer_id=35309, restaurant_name='Blue Ribbon Sushi', cuisine_type='Japanese', cost_of_the_o
rder=25.22, day_of_the_week='Weekday', rating='Not given', food_preparation_time=31, delivery_time=24),
 Row(order_id=1477513, customer_id=64151, restaurant_name="Jack's Wife Freda", cuisine_type='Mediterranean', cost_of_
the_order=12.18, day_of_the_week='Weekday', rating='5', food_preparation_time=23, delivery_time=31),
 Row(order_id=1478056, customer_id=120353, restaurant_name='Blue Ribbon Sushi', cuisine_type='Japanese', cost_of_the_
order=19.45, day_of_the_week='Weekend', rating='Not given', food_preparation_time=28, delivery_time=24)]
```

```
df.columns
```

```
Out[7]: ['order_id',
 'customer_id',
 'restaurant_name',
 'cuisine_type',
 'cost_of_the_order',
 'day_of_the_week',
 'rating',
 'food_preparation_time',
 'delivery_time']
```

```
print(f"The length of Columns is: {len(df.columns)} and total size of datasets is: {df.count()}")
```

```
The length of Columns is: 9 and total size of datasets is: 1898
```

```
df.printSchema()
```

```
root
 |-- order_id: integer (nullable = true)
 |-- customer_id: integer (nullable = true)
 |-- restaurant_name: string (nullable = true)
 |-- cuisine_type: string (nullable = true)
 |-- cost_of_the_order: double (nullable = true)
 |-- day_of_the_week: string (nullable = true)
 |-- rating: string (nullable = true)
 |-- food_preparation_time: integer (nullable = true)
 |-- delivery_time: integer (nullable = true)
```

```
df = df.withColumn(
    "rating",
    f.regexp_replace(f.col("rating"),"Not given","0")
)
```

```
df = df.withColumn(
    'rating',
    f.col('rating').cast("int")
)
```

```
df.printSchema()
```

```
root
 |-- order_id: integer (nullable = true)
 |-- customer_id: integer (nullable = true)
 |-- restaurant_name: string (nullable = true)
 |-- cuisine_type: string (nullable = true)
 |-- cost_of_the_order: double (nullable = true)
 |-- day_of_the_week: string (nullable = true)
 |-- rating: integer (nullable = true)
 |-- food_preparation_time: integer (nullable = true)
 |-- delivery_time: integer (nullable = true)
```

```
df.select([f.sum(f.col(col).isNull().cast("int")).alias(col) for col in df.columns]).show()
```

```
+--------+-----------+---------------+------------+-----------------+---------------+------+---------------------+---
----------+
|order_id|customer_id|restaurant_name|cuisine_type|cost_of_the_order|day_of_the_week|rating|food_preparation_time|del
ivery_time|
+--------+-----------+---------------+------------+-----------------+---------------+------+---------------------+---
----------+
|       0|          0|              0|           0|                0|              0|     0|                    0|
0|
+--------+-----------+---------------+------------+-----------------+---------------+------+---------------------+---
----------+
```

```
null_check = df.select([f.sum(f.col(col).isNull().cast("int")).alias(col) for col in df.columns])
```

Observations:

- There are no missing values in the data and no duplicates either.

Check the statistical summary of the data. What is the minimum, average, and maximum time it takes for food to be prepared once an order is placed?

```
df.describe().display()
```

Table

	summary	order_id	customer_id	restaurant_name	cuisine_type	cost_of_the_order	day_
1	count	1898	1898	1898	1898	1898	1898
2	mean	1477495.5	171168.478398314	null	null	16.498851422550082	null
3	stddev	548.0497240214318	113698.13974303957	null	null	7.483812110049568	null
4	min	1476547	1311	'wichcraft	American	4.47	Week
5	max	1478444	405334	indikitch	Vietnamese	35.41	Week

5 rows

Observations:

- Order ID and Customer ID are identifiers for each order.
- The minimum time it takes to for an order to be prepped is 20.0 minutes, the maximum time is 35.0 minutes, and the average is 27.0 minutes.
- The cost of an order ranges from 4.47 to 35.41 dollars, with an average order costing around 16 dollars and a standard deviation of 7.5 dollars.
- The cost of 75% of the orders are below 23 dollars.

- This indicates that most of the customers prefer low-cost food compared to the expensive ones.
- Delivery time ranges from 15 to 33 minutes, with an average of around 24 minutes and a standard deviation of 5 minutes.
- The spread is not too high for delivery time either

How many orders are not rated?

```
df.select('rating').show()
```

```
+-------+
|rating|
+-------+
|      0|
|      0|
|      5|
|      3|
|      4|
|      3|
|      0|
|      3|
|      5|
|      5|
|      0|
|      5|
|      5|
|      3|
|      0|
|      5|
|      0|
|      0|
```

```
not_rated = df.groupBy(
    f.col("rating")).count(

    )

not_rated.show()
```

```
+------+-----+
|rating|count|
+------+-----+
|     3|  188|
|     5|  588|
|     4|  386|
|     0|  736|
+------+-----+
```

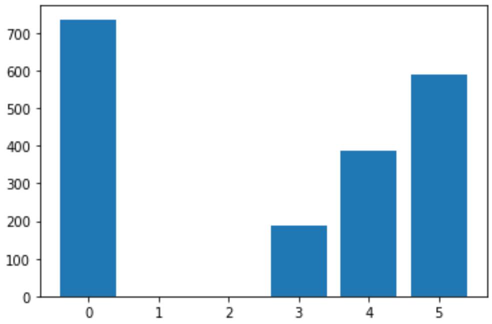
```
# Create the visualisation

data = not_rated.select("rating", "count").collect()

rate = [i['rating'] for i in data]
number = [i['count'] for i in data]

plt.bar(rate, number)
```

```
Out[18]: <BarContainer object of 4 artists>
```



```
print(f"The number of that who not rated 736")
```

```
The number of that who not rated 736
```

- Based off of the statistical summary of the data and the 4 unique results in the ratings column and 736 orders that are not rated.

Observations:

- The distribution of 'rating' shows that the most frequent rating category is 'not given', followed by a rating of 5.
- Only 188 orders have been rated 3.
- 386 orders have a rating of 4.
- 588 orders have a rating of 5.
- 736 orders have not been rated.

```
df.select('customer_id').distinct().count()
```

```
Out[20]: 1200
```

Exploratory Data Analysis (EDA)

Univariate Analysis

Explore all the variables and provide observations on their distributions.

```
df.select("cost_of_the_order").collect()
```

```
Out[21]: [Row(cost_of_the_order=30.75),
 Row(cost_of_the_order=12.08),
 Row(cost_of_the_order=12.23),
 Row(cost_of_the_order=29.2),
 Row(cost_of_the_order=11.59),
 Row(cost_of_the_order=25.22),
 Row(cost_of_the_order=6.07),
 Row(cost_of_the_order=5.97),
 Row(cost_of_the_order=16.44),
 Row(cost_of_the_order=7.18),
 Row(cost_of_the_order=5.92),
 Row(cost_of_the_order=8.1),
 Row(cost_of_the_order=24.3),
 Row(cost_of_the_order=11.3),
 Row(cost_of_the_order=12.13),
 Row(cost_of_the_order=16.2),
 Row(cost_of_the_order=16.98),
 Row(cost_of_the_order=33.03),
 Row(cost_of_the_order=14.12),
 Row(cost_of_the_order=16.2),
 Row(cost_of_the_order=24.2),
```

```
values = df.select("cost_of_the_order").rdd.flatMap(lambda x: x)
values
```

```
Out[22]: PythonRDD[68] at RDD at PythonRDD.scala:58
```

```
# Collect the values as a list
values_list = values.collect()
values_list
```

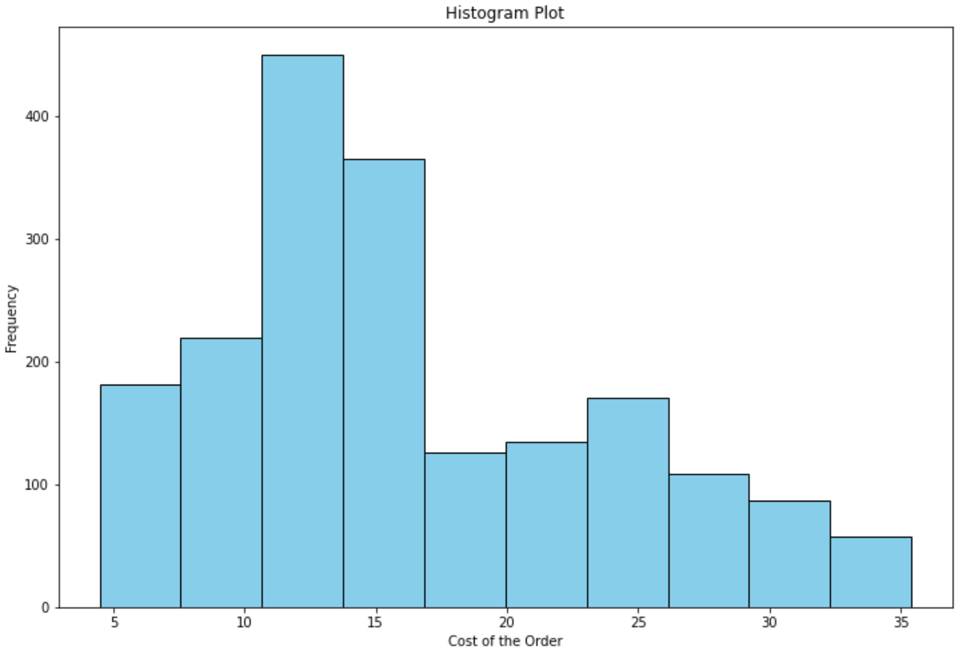
```
Out[23]: [30.75,
 12.08,
 12.23,
 29.2,
 11.59,
 25.22,
 6.07,
 5.97,
 16.44,
 7.18,
 5.92,
 8.1,
 24.3,
 11.3,
 12.13,
 16.2,
 16.98,
 33.03,
 14.12,
 16.2,
```

```
plt.figure(figsize=(12,8))

# Create a histogram
plt.hist(values_list, bins=10, color='skyblue', edgecolor='black')


# Add labels and a title
plt.xlabel("Cost of the Order")
plt.ylabel("Frequency")
plt.title("Histogram Plot")

# Show the plot
plt.show()
```

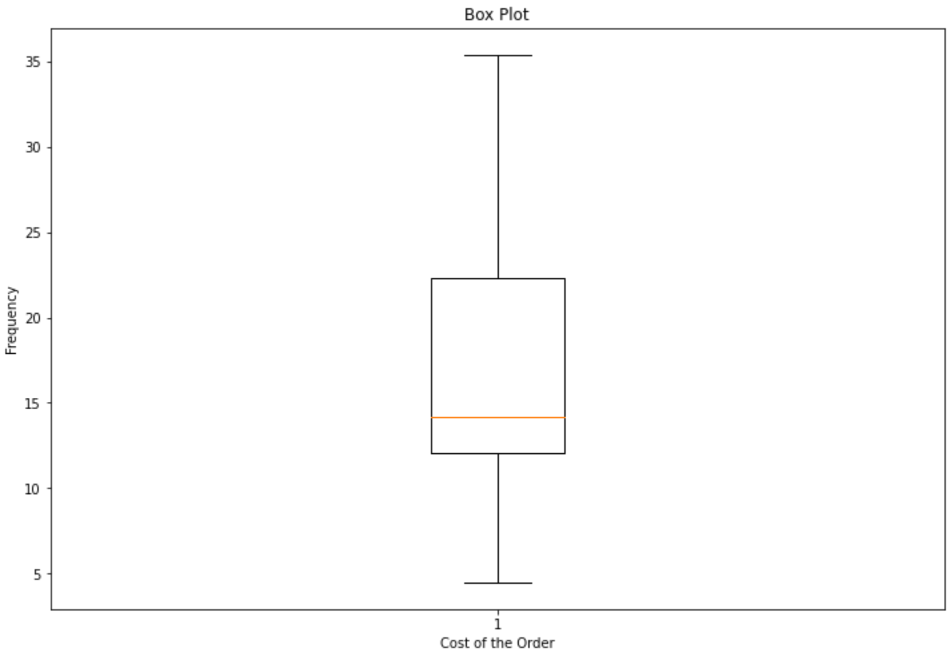


```
plt.figure(figsize=(12,8))

# Create a histogram
plt.boxplot(values_list)


# Add labels and a title
plt.xlabel("Cost of the Order")
plt.ylabel("Frequency")
plt.title("Box Plot")

# Show the plot
plt.show()
```



Observations:

- The average cost of the order is greater than the median cost indicating that the distribution for the cost of the order is right-skewed.
- The mode of the distribution indicates that a large chunk of people prefer to order food that costs around 10-12 dollars.
- There are few orders that cost greater than 30 dollars. These orders might be for some expensive meals.

```
df.select("day_of_the_week").show()
```

```
+---------------+
|day_of_the_week|
+---------------+
|        Weekend|
|        Weekend|
|        Weekday|
|        Weekend|
|        Weekday|
|        Weekday|
|        Weekend|
|        Weekday|
|        Weekday|
|        Weekday|
|        Weekday|
|        Weekend|
|        Weekend|
|        Weekend|
|        Weekday|
|        Weekend|
|        Weekend|
|        Weekend|
```

```
day_of_week = df.select("day_of_the_week").rdd.flatMap(lambda x : x)
day_of_week_list = day_of_week.collect()
day_of_week_list
```

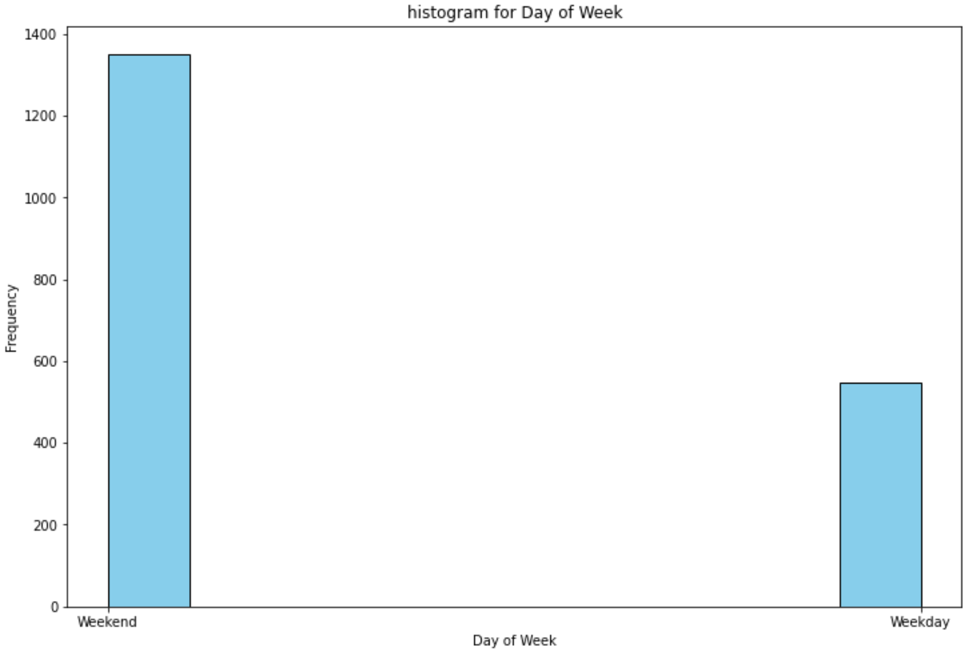
```
Out[27]: ['Weekend',
 'Weekend',
 'Weekday',
 'Weekend',
 'Weekday',
 'Weekday',
 'Weekend',
 'Weekday',
 'Weekday',
 'Weekday',
 'Weekday',
 'Weekend',
 'Weekend',
 'Weekend',
 'Weekday',
 'Weekend',
 'Weekend',
 'Weekend',
 'Weekend',
 'Weekend',
 'Weekend'
```

```
plt.figure(figsize=(12,8))

# Create a histogram
plt.hist(day_of_week_list, bins=10, color='skyblue', edgecolor='black')


# Add labels and a title
plt.xlabel("Day of Week")
plt.ylabel("Frequency")
plt.title("histogram for Day of Week")

# Show the plot
plt.show()
```



Observations:

- The 'day_of_the_week' columns consists of 2 unique values - Weekday and Weekend.
- The distribution shows that around 1300 orders are placed on weekends.
- The distribution shows that number of order placed on weekends is approximately double the number of orders placed on weekdays.

Food Preparation Time

```
preparation_time = df.select("food_preparation_time").rdd.flatMap(lambda x:x)
preparation_time_list = preparation_time.collect()


delivery_time = df.select("delivery_time").rdd.flatMap(lambda x:x)
delivery_time_list = delivery_time.collect()
```

```
preparation_time_list
```

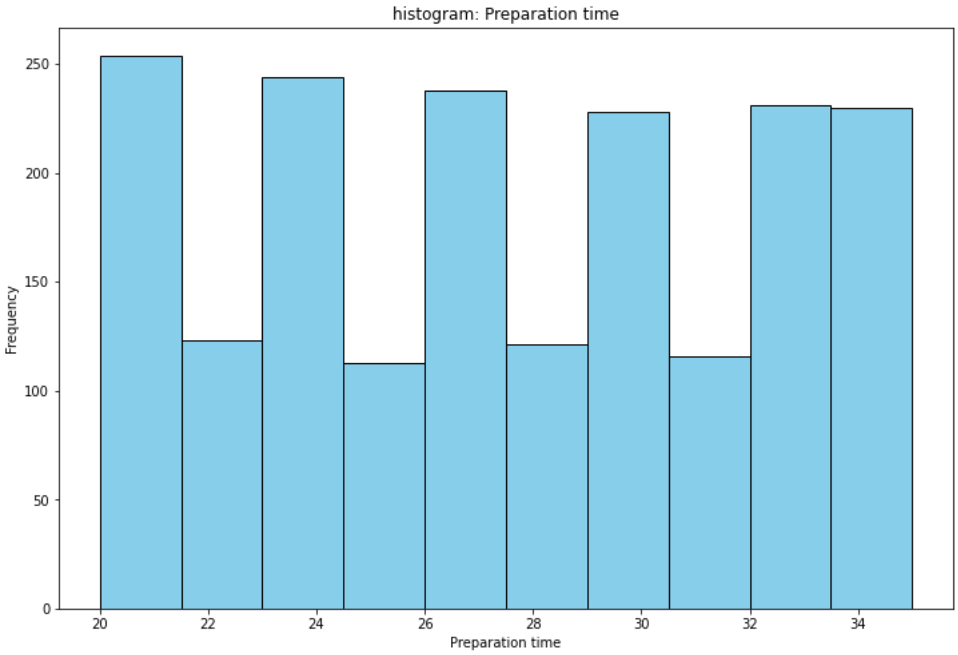
```
Out[30]: [25,
 25,
 23,
 25,
 25,
 20,
 28,
 33,
 21,
 29,
 34,
 23,
 23,
 24,
 23,
 33,
 30,
 21,
 25,
 35,
 21,
```

```
plt.figure(figsize=(12,8))

# Create a histogram
plt.hist(preparation_time_list, bins=10, color='skyblue', edgecolor='black')


# Add labels and a title
plt.xlabel("Preparation time")
plt.ylabel("Frequency")
plt.title("histogram: Preparation time")

# Show the plot
plt.show()
```

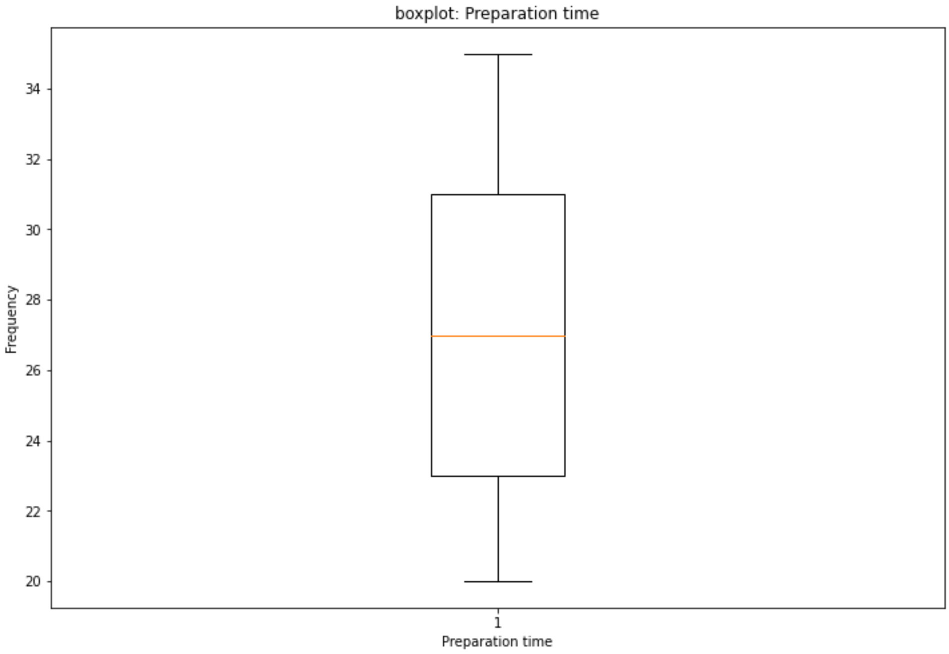


```
plt.figure(figsize=(12,8))

# Create a histogram
plt.boxplot(preparation_time_list)


# Add labels and a title
plt.xlabel("Preparation time")
plt.ylabel("Frequency")
plt.title("boxplot: Preparation time")

# Show the plot
plt.show()
```



Observations:

- The average food preparation time is almost equal to the median food preparation time indicating that the distribution is nearly symmetrical.
- The food preparation time is pretty evenly distributed between 20 and 35 minutes.
- There are no outliers in this column.

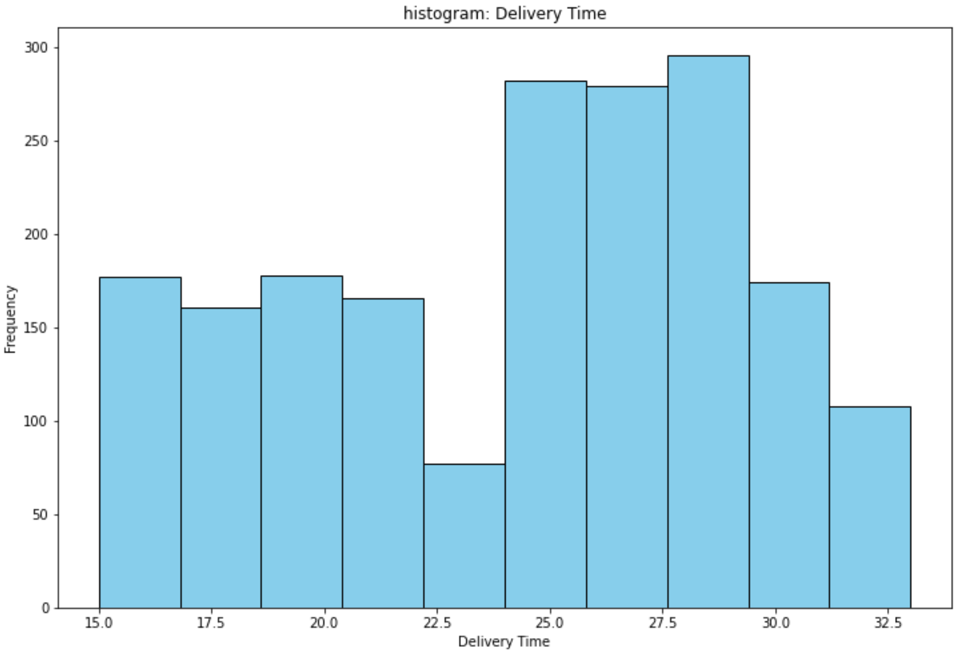
```
delivery_time_list
```

```
Out[33]: [20,
 23,
 28,
 15,
 24,
 24,
 21,
 30,
 26,
 26,
 28,
 22,
 17,
 23,
 30,
 25,
 16,
 22,
 24,
 26,
 24,
plt.figure(figsize=(12,8))

# Create a histogram
plt.hist(delivery_time_list, bins=10, color='skyblue', edgecolor='black')


# Add labels and a title
plt.xlabel("Delivery Time")
plt.ylabel("Frequency")
plt.title("histogram: Delivery Time")

# Show the plot
plt.show()
```

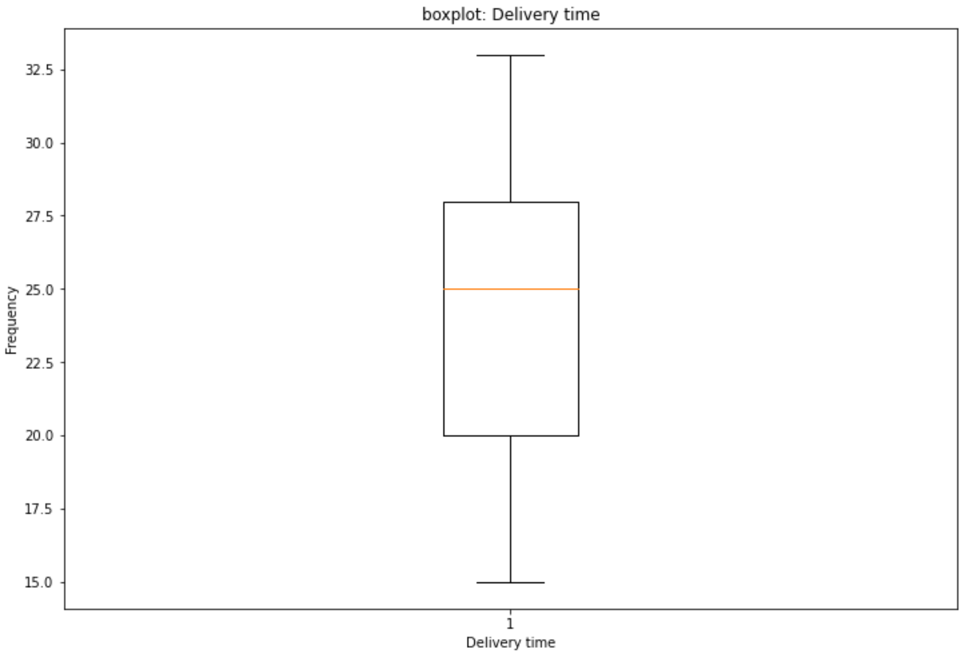


```
plt.figure(figsize=(12,8))

# Create a histogram
plt.boxplot(delivery_time_list)


# Add labels and a title
plt.xlabel("Delivery time")
plt.ylabel("Frequency")
plt.title("boxplot: Delivery time")

# Show the plot
plt.show()
```



Observations:

- The average delivery time is a bit smaller than the median delivery time indicating that the distribution is a bit left-skewed.
- Comparatively more number of orders have delivery time between 24 and 30 minutes.
- There are no outliers in this column.

```
df.select("cuisine_type").show()
```

```
+-------------+
| cuisine_type|
+-------------+
|       Korean|
|     Japanese|
|      Mexican|
|     American|
|     American|
|       Indian|
|      Italian|
|Mediterranean|
|       Indian|
|       Indian|
|      Chinese|
|      Mexican|
|     American|
|Mediterranean|
|     American|
|     American|
|     Japanese|
|     Japanese|
```

```
cuisine_type_count = df.groupBy("cuisine_type").agg(f.count("*").alias("counts"))
cuisine_type_count.show()
```

```
+--------------+------+
|  cuisine_type|counts|
+--------------+------+
|       Mexican|    77|
|          Thai|    19|
|        Indian|    73|
|      Southern|    17|
|       Chinese|   215|
|      Japanese|   470|
|       Spanish|    12|
|    Vietnamese|     7|
|       Italian|   298|
|        Korean|    13|
|        French|    18|
|Middle Eastern|    49|
| Mediterranean|    46|
|      American|   584|
+--------------+------+
```

```
cuisine_type_count_list = cuisine_type_count.select("cuisine_type", "counts").collect()
cuisine_type_count_list
```

```
Out[38]: [Row(cuisine_type='Mexican', counts=77),
 Row(cuisine_type='Thai', counts=19),
 Row(cuisine_type='Indian', counts=73),
 Row(cuisine_type='Southern', counts=17),
 Row(cuisine_type='Chinese', counts=215),
 Row(cuisine_type='Japanese', counts=470),
 Row(cuisine_type='Spanish', counts=12),
 Row(cuisine_type='Vietnamese', counts=7),
 Row(cuisine_type='Italian', counts=298),
 Row(cuisine_type='Korean', counts=13),
 Row(cuisine_type='French', counts=18),
 Row(cuisine_type='Middle Eastern', counts=49),
 Row(cuisine_type='Mediterranean', counts=46),
 Row(cuisine_type='American', counts=584)]
```

```
cuisine_type = [i["cuisine_type"] for i in cuisine_type_count_list]
counts = [i["counts"] for i in cuisine_type_count_list]
```

```
cuisine_type
```

```
Out[40]: ['Mexican',
 'Thai',
 'Indian',
 'Southern',
 'Chinese',
 'Japanese',
 'Spanish',
 'Vietnamese',
 'Italian',
 'Korean',
 'French',
 'Middle Eastern',
 'Mediterranean',
 'American']
```

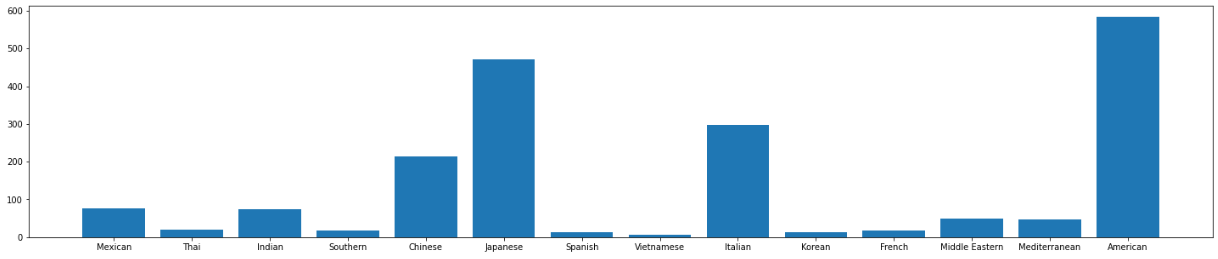
```
counts
```

```
Out[41]: [77, 19, 73, 17, 215, 470, 12, 7, 298, 13, 18, 49, 46, 584]
```

```
plt.figure(figsize=(25,5))

plt.bar(cuisine_type, counts)
```

```
Out[42]: <BarContainer object of 14 artists>
```



Observations:

- There are 14 unique cuisines in the dataset.
- The distribution of cuisine types show that cuisines types are not equally distributed.

- The most frequent cuisine type is American followed by Japanese and Italian.
- Vietnamese appears to be the least popular of all cuisine types.

```
df.select("restaurant_name").show()
```

```
+--------------------+
|     restaurant_name|
+--------------------+
|             Hangawi|
|Blue Ribbon Sushi...|
|         Cafe Habana|
|Blue Ribbon Fried...|
|    Dirty Bird to Go|
|    Tamarind TriBeCa|
|   The Meatball Shop|
|           Barbounia|
|  Anjappar Chettinad|
|       Bukhara Grill|
|Big Wong Restaura...|
|Empanada Mama (cl...|
|Blue Ribbon Fried...|
|               Pylos|
|Lucky's Famous Bu...|
|         Shake Shack|
|       Sushi of Gari|
|Blue Ribbon Sushi...|
```

```
restaurant_name_counts = df.groupBy("restaurant_name").agg(f.count("*").alias("Counts"))
restaurant_name_counts.show()
```

```
+--------------------+------+
|     restaurant_name|Counts|
+--------------------+------+
|         J. G. Melon|    15|
|          Taro Sushi|     1|
|                brgr|     2|
|Schnipper's Quali...|     3|
|           Cafeteria|     9|
|         Han Dynasty|    46|
|            Rubirosa|    37|
|           L'Express|     6|
|       Bukhara Grill|     2|
|           Hatsuhana|     6|
|           67 Burger|     1|
|             Hangawi|     2|
|                Amma|     4|
|  Haru Gramercy Park|     1|
|         Tres Carnes|     3|
|Le Zie 2000 Tratt...|     1|
|Balthazar Boulang...|    10|
|      Zero Otto Nove|     2|
```

```
restaurant_name_counts_list = restaurant_name_counts.select("restaurant_name", "counts").collect()
restaurant_name_counts_list
```

```
Out[45]: [Row(restaurant_name='J. G. Melon', counts=15),
 Row(restaurant_name='Taro Sushi', counts=1),
 Row(restaurant_name='brgr', counts=2),
 Row(restaurant_name="Schnipper's Quality Kitchen", counts=3),
 Row(restaurant_name='Cafeteria', counts=9),
 Row(restaurant_name='Han Dynasty', counts=46),
 Row(restaurant_name='Rubirosa', counts=37),
 Row(restaurant_name="L'Express", counts=6),
 Row(restaurant_name='Bukhara Grill', counts=2),
 Row(restaurant_name='Hatsuhana', counts=6),
 Row(restaurant_name='67 Burger', counts=1),
 Row(restaurant_name='Hangawi', counts=2),
 Row(restaurant_name='Amma', counts=4),
 Row(restaurant_name='Haru Gramercy Park', counts=1),
 Row(restaurant_name='Tres Carnes', counts=3),
 Row(restaurant_name='Le Zie 2000 Trattoria', counts=1),
 Row(restaurant_name='Balthazar Boulangerie', counts=10),
 Row(restaurant_name='Zero Otto Nove', counts=2),
 Row(restaurant_name='Empanada Mama (closed)', counts=13),
 Row(restaurant_name='Hibino', counts=1),
restaurant_name_col = [i["restaurant_name"] for i in restaurant_name_counts_list]
counts_col = [i["counts"] for i in restaurant_name_counts_list]
```

```
restaurant_name_col
```

```
Out[47]: ['J. G. Melon',
 'Taro Sushi',
 'brgr',
 "Schnipper's Quality Kitchen",
 'Cafeteria',
 'Han Dynasty',
 'Rubirosa',
 "L'Express",
 'Bukhara Grill',
 'Hatsuhana',
 '67 Burger',
 'Hangawi',
 'Amma',
 'Haru Gramercy Park',
 'Tres Carnes',
 'Le Zie 2000 Trattoria',
 'Balthazar Boulangerie',
 'Zero Otto Nove',
 'Empanada Mama (closed)',
 'Hibino',
 'UVA Wine Bar & Restaurant',
```

```
counts_col
```

```
Out[48]: [15,
 1,
 2,
 3,
 9,
 46,
 37,
 6,
 2,
 6,
 1,
 2,
 4,
 1,
 3,
 1,
 10,
 2,
 13,
 1,
 2,
```

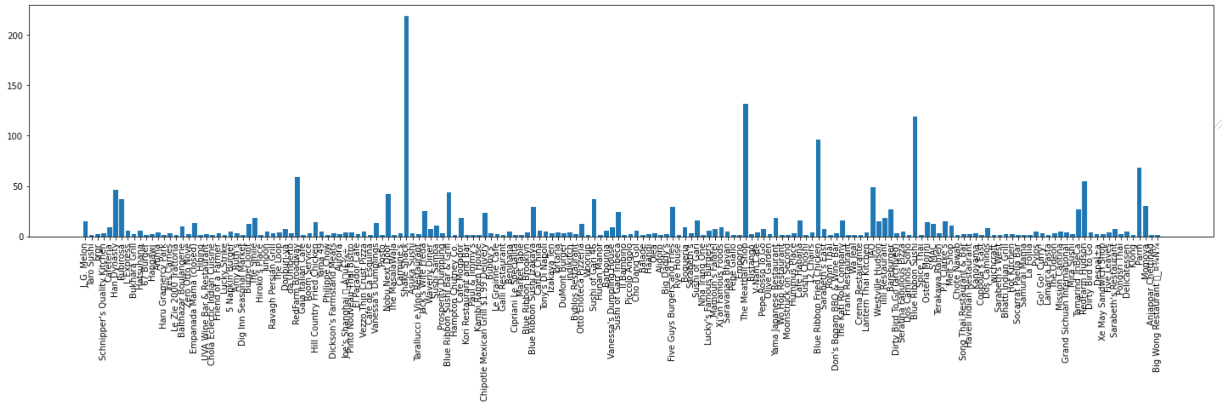
```
plt.figure(figsize=(25,5))

plt.bar(restaurant_name_col, counts_col)

plt.xticks(rotation=90)
```

```
Out[49]: ([0,
  1,
  2,
  3,
  4,
  5,
  6,
  7,
  8,
  9,
  10,
  11,
  12,
  13,
  14,
  15,
  16,
  17,
  18,
  19,
```

```
/databricks/python/lib/python3.9/site-packages/IPython/core/events.py:89: UserWarning: Glyph 142 (\x8e) missing from
current font.
  func(*args, **kwargs)
/databricks/python/lib/python3.9/site-packages/IPython/core/events.py:89: UserWarning: Glyph 140 (\x8c) missing from
current font.
  func(*args, **kwargs)
/databricks/python/lib/python3.9/site-packages/IPython/core/pylabtools.py:151: UserWarning: Glyph 142 (\x8e) missing
from current font.
  fig.canvas.print_figure(bytes_io, **kw)
/databricks/python/lib/python3.9/site-packages/IPython/core/pylabtools.py:151: UserWarning: Glyph 140 (\x8c) missing
from current font.
  fig.canvas.print_figure(bytes_io, **kw)
```



Which are the top 5 restaurants in terms of the number of orders received?

```
restaurant_name_counts.orderBy("counts", ascending=False).head(5)
```

```
Out[50]: [Row(restaurant_name='Shake Shack', Counts=219),
 Row(restaurant_name='The Meatball Shop', Counts=132),
 Row(restaurant_name='Blue Ribbon Sushi', Counts=119),
 Row(restaurant_name='Blue Ribbon Fried Chicken', Counts=96),
 Row(restaurant_name='Parm', Counts=68)]
```

```
restaurant_name_top5 = restaurant_name_counts.orderBy("counts", ascending=False).head(5)
```

```
top_5 = [i['restaurant_name'] for i in restaurant_name_top5]
top_5
```

```
Out[52]: ['Shake Shack',
 'The Meatball Shop',
 'Blue Ribbon Sushi',
 'Blue Ribbon Fried Chicken',
 'Parm']
```

Which is the most popular cuisine on weekends?

```
df_weekend = df.filter(f.col("day_of_the_week") == "Weekend")
df_weekend.show()
```

```
+--------+-----------+--------------------+-------------+-----------------+---------------+------+-------------------
--+-------------+
|order_id|customer_id|     restaurant_name| cuisine_type|cost_of_the_order|day_of_the_week|rating|food_preparation_ti
me|delivery_time|
+--------+-----------+--------------------+-------------+-----------------+---------------+------+-------------------
--+-------------+
| 1477147|     337525|             Hangawi|       Korean|            30.75|        Weekend|     0|
25|           20|
| 1477685|     358141|Blue Ribbon Sushi...|     Japanese|            12.08|        Weekend|     0|
25|           23|
| 1477334|     106968|Blue Ribbon Fried...|     American|             29.2|        Weekend|     3|
25|           15|
| 1477894|     157711|   The Meatball Shop|      Italian|             6.07|        Weekend|     0|
28|           21|
| 1478437|     221206|Empanada Mama (cl...|      Mexican|              8.1|        Weekend|     5|
23|           22|
| 1476966|     129969|Blue Ribbon Fried...|     American|             24.3|        Weekend|     5|
23|           17|
| 1477449|     104548|               Pylos|Mediterranean|             11.3|        Weekend|     3|
24|           23|
| 1477414|      66222|         Shake Shack|     American|             16.2|        Weekend|     5|
```

```
df_weekday = df.filter(f.col("day_of_the_week") == "Weekday")
df_weekday.show()
```

```
+--------+-----------+--------------------+--------------+-----------------+---------------+------+------------------
---+-------------+
|order_id|customer_id|     restaurant_name|  cuisine_type|cost_of_the_order|day_of_the_week|rating|food_preparation_t
ime|delivery_time|
+--------+-----------+--------------------+--------------+-----------------+---------------+------+------------------
---+-------------+
| 1477070|      66393|         Cafe Habana|       Mexican|            12.23|        Weekday|     5|
23|           28|
| 1478249|      76942|    Dirty Bird to Go|      American|            11.59|        Weekday|     4|
25|           24|
| 1477224|     147468|    Tamarind TriBeCa|        Indian|            25.22|        Weekday|     3|
20|           24|
| 1477859|      89574|           Barbounia| Mediterranean|             5.97|        Weekday|     3|
33|           30|
| 1477174|     121706|  Anjappar Chettinad|        Indian|            16.44|        Weekday|     5|
21|           26|
| 1477311|      39705|       Bukhara Grill|        Indian|             7.18|        Weekday|     5|
29|           26|
| 1477895|     143926|Big Wong Restaura...|       Chinese|             5.92|        Weekday|     0|
34|           28|
| 1478198|      62667|Lucky's Famous Bu...|      American|            12.13|        Weekday|     0|
```

```
count_cuisine = df.groupBy(f.col("cuisine_type")).count()
count_cuisine = count_cuisine.orderBy("count", ascending=False)
count_cuisine.show()
```

```
+--------------+-----+
|  cuisine_type|count|
+--------------+-----+
|      American|  584|
|      Japanese|  470|
|       Italian|  298|
|       Chinese|  215|
|       Mexican|   77|
|        Indian|   73|
|Middle Eastern|   49|
| Mediterranean|   46|
|          Thai|   19|
|        French|   18|
|      Southern|   17|
|        Korean|   13|
|       Spanish|   12|
|    Vietnamese|    7|
+--------------+-----+
```

```
count_cuisine_data = count_cuisine.select(count_cuisine.columns).collect()
count_cuisine_data
```

```
Out[66]: [Row(cuisine_type='American', count=584),
 Row(cuisine_type='Japanese', count=470),
 Row(cuisine_type='Italian', count=298),
 Row(cuisine_type='Chinese', count=215),
 Row(cuisine_type='Mexican', count=77),
 Row(cuisine_type='Indian', count=73),
 Row(cuisine_type='Middle Eastern', count=49),
 Row(cuisine_type='Mediterranean', count=46),
 Row(cuisine_type='Thai', count=19),
 Row(cuisine_type='French', count=18),
 Row(cuisine_type='Southern', count=17),
 Row(cuisine_type='Korean', count=13),
 Row(cuisine_type='Spanish', count=12),
 Row(cuisine_type='Vietnamese', count=7)]
```

```
types = [i['cuisine_type'] for i in count_cuisine_data]
count = [i['count'] for i in count_cuisine_data]
```

```
types
```

```
Out[71]: ['American',
 'Japanese',
 'Italian',
 'Chinese',
 'Mexican',
 'Indian',
 'Middle Eastern',
 'Mediterranean',
 'Thai',
 'French',
 'Southern',
 'Korean',
 'Spanish',
 'Vietnamese']
```

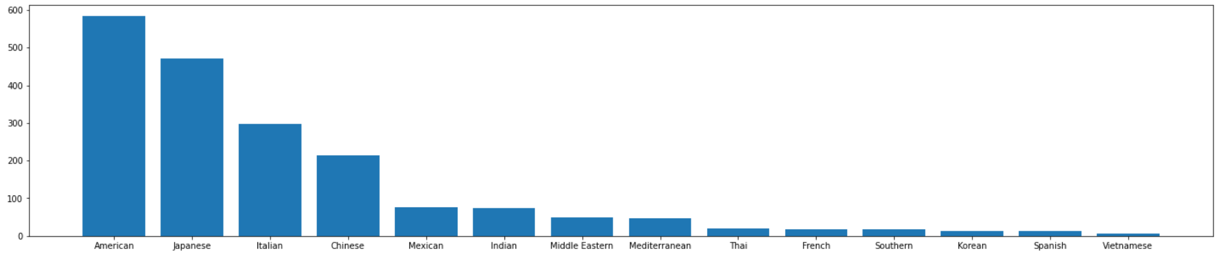
```
count
```

```
Out[72]: [584, 470, 298, 215, 77, 73, 49, 46, 19, 18, 17, 13, 12, 7]
```

```
plt.figure(figsize=(25,5))

plt.bar(types, count)
```

```
Out[80]: <BarContainer object of 14 artists>
```



```
day_of_week = df.groupBy(f.col("day_of_the_week")).count()

day_of_week.show()
```

```
+---------------+-----+
|day_of_the_week|count|
+---------------+-----+
|        Weekday|  547|
|        Weekend| 1351|
+---------------+-----+
```

```
day_of_week_data = day_of_week.select('day_of_the_week').collect()
day_of_week_data
```

```
Out[85]: [Row(day_of_the_week='Weekday'), Row(day_of_the_week='Weekend')]
```

```
column = [i["day_of_the_week"] for i in day_of_week_data]
column
```

```
Out[87]: ['Weekday', 'Weekend']
```

The most popular cuisine is American cuisine.

```
%md

# What percentage of the orders cost more than 20 dollars?
```

```
UsageError: Line magic function `%nd` not found.
  UsageError: Line magic function `%nd` not found.
```

```
cost=df.select("cost_of_the_order")
cost.show()
```

```
+-----------------+
|cost_of_the_order|
+-----------------+
|            30.75|
|            12.08|
|            12.23|
|             29.2|
|            11.59|
|            25.22|
|             6.07|
|             5.97|
|            16.44|
|             7.18|
|             5.92|
|              8.1|
|             24.3|
|             11.3|
|            12.13|
|             16.2|
|            16.98|
|            33.03|
```

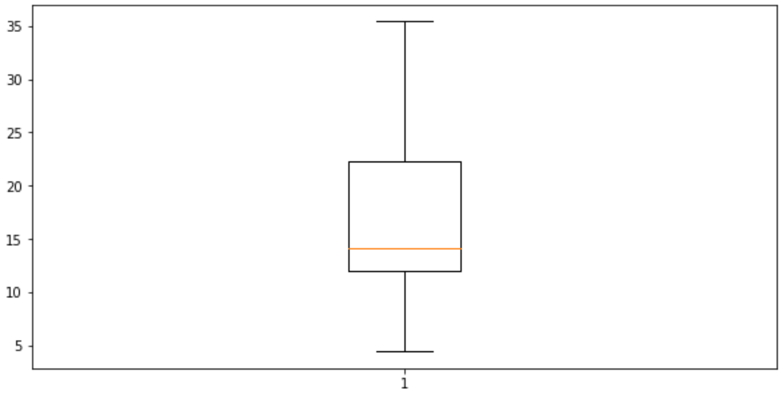
```
cost = cost.select(cost.columns).collect()
cost = [i["cost_of_the_order"] for i in cost]
cost
```

```
Out[93]: [30.75,
 12.08,
 12.23,
 29.2,
 11.59,
 25.22,
 6.07,
 5.97,
 16.44,
 7.18,
 5.92,
 8.1,
 24.3,
 11.3,
 12.13,
 16.2,
 16.98,
 33.03,
 14.12,
 16.2,
 24.2,
```

```
plt.figure(figsize=(10,5))
```

```
plt.boxplot(cost)
```

```
Out[96]: {'whiskers': [<matplotlib.lines.Line2D at 0x7fe78ae5f8b0>,
  <matplotlib.lines.Line2D at 0x7fe78ae5fb80>],
 'caps': [<matplotlib.lines.Line2D at 0x7fe78ae5fe50>,
  <matplotlib.lines.Line2D at 0x7fe78ae79160>],
 'boxes': [<matplotlib.lines.Line2D at 0x7fe78ae5f5e0>],
 'medians': [<matplotlib.lines.Line2D at 0x7fe78ae79430>],
 'fliers': [<matplotlib.lines.Line2D at 0x7fe78ae79700>],
 'means': []}
```



```
# Get orders that cost above 20 dollars
df_greater_than_20 = df.filter(df["cost_of_the_order"] > 20)
df_greater_than_20.show()
```

```
+--------+-----------+--------------------+--------------+-----------------+---------------+------+------------------
---+-------------+
|order_id|customer_id|     restaurant_name|  cuisine_type|cost_of_the_order|day_of_the_week|rating|food_preparation_t
ime|delivery_time|
+--------+-----------+--------------------+--------------+-----------------+---------------+------+------------------
---+-------------+
| 1477147|     337525|             Hangawi|        Korean|            30.75|        Weekend|     0|
25|           20|
| 1477334|     106968|Blue Ribbon Fried...|      American|             29.2|        Weekend|     3|
25|           15|
| 1477224|     147468|    Tamarind TriBeCa|        Indian|            25.22|        Weekday|     3|
20|           24|
| 1476966|     129969|Blue Ribbon Fried...|      American|             24.3|        Weekend|     5|
23|           17|
| 1477373|     139885|Blue Ribbon Sushi...|      Japanese|            33.03|        Weekend|     0|
21|           22|
| 1478296|     250494|Five Guys Burgers...|      American|             24.2|        Weekend|     4|
21|           24|
| 1478287|     150599|         Shake Shack|      American|             29.1|        Weekday|     5|
21|           30|
| 1476693|      41877|        Cafe Mogador|Middle Eastern|             29.1|        Weekday|     5|
```

```
print(f'The number of total orders that cost above 20 dollars is: {df_greater_than_20.count()}.')
```

```
The number of total orders that cost above 20 dollars is: 555.
```

```
df_greater_than_20
```

```
Out[107]: 555
```

```
# Calculate percentage of such orders in the dataset
percentage = (df_greater_than_20 / df.count()) * 100
percentage
```

```
Out[110]: 29.24130663856691
```

```
print(f"Percentage of orders above 20 dollars: {round(percentage,2)} %.")
```

```
Percentage of orders above 20 dollars: 29.24 %.
```

Observations:

- There are a total of 555 orders that cost above 20 dollars.
- The percentage of such orders in the dataset is around 29.24%

What is the mean order delivery time?

```
delivery=df.select("delivery_time")
delivery.show()
```

```
+-------------+
|delivery_time|
+-------------+
|           20|
|           23|
|           28|
|           15|
|           24|
|           24|
|           21|
|           30|
|           26|
|           26|
|           28|
|           22|
|           17|
|           23|
|           30|
|           25|
|           16|
|           22|
```

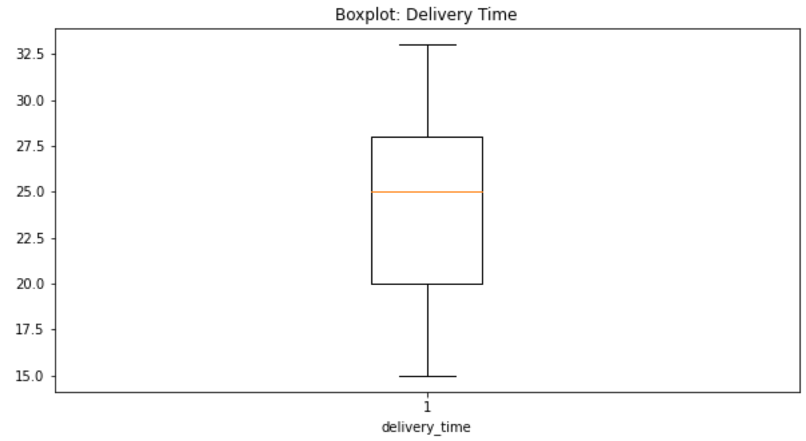
```
delivery = delivery.select(delivery.columns).collect()
delivery = [i['delivery_time'] for i in delivery]
delivery
```

```
Out[114]: [20,
 23,
 28,
 15,
 24,
 24,
 21,
 30,
 26,
 26,
 28,
 22,
 17,
 23,
 30,
 25,
 16,
 22,
 24,
 26,
 24,
```

```
plt.figure(figsize=(10,5))
plt.title('Boxplot: Delivery Time')
plt.xlabel('delivery_time')


plt.boxplot(delivery)
```

```
Out[116]: {'whiskers': [<matplotlib.lines.Line2D at 0x7fe788d77d00>,
  <matplotlib.lines.Line2D at 0x7fe788d77fd0>],
 'caps': [<matplotlib.lines.Line2D at 0x7fe788cd52e0>,
  <matplotlib.lines.Line2D at 0x7fe788cd55b0>],
 'boxes': [<matplotlib.lines.Line2D at 0x7fe788d77a30>],
 'medians': [<matplotlib.lines.Line2D at 0x7fe788cd5880>],
 'fliers': [<matplotlib.lines.Line2D at 0x7fe788cd5b50>],
 'means': []}
```



```
avrg = df.select("delivery_time").agg({
    "delivery_time" : "mean"
})
avrg.show()
```

```
+------------------+
|avg(delivery_time)|
+------------------+
|24.161749209694417|
+------------------+
```

```
# Mean delivery time
print(f'The mean delivery time for this dataset is: 26.16 minute')
```

```
The mean delivery time for this dataset is: 26.16 minute
```

Observations:

- Delivery time ranges from 15 to 33 minutes, with an average of around 24
- minutes and a standard deviation of 5 minutes. The mean order delivery time is about 24.16
- minutes. The spread is not too high for delivery time either and there are no outliers of delivery time.

The company has decided to give 20% discount vouchers to the top 3 most frequent customers. Find the IDs of these customers and the number of orders they placed.

```
customer_id = df.select("customer_id")
```

```
customer_id.show()
```

```
+-----------+
|customer_id|
+-----------+
|     337525|
|     358141|
|      66393|
|     106968|
|      76942|
|     147468|
|     157711|
|      89574|
|     121706|
|      39705|
|     143926|
|     221206|
|     129969|
|     104548|
|      62667|
|      66222|
|     104555|
```

```
customer_id = customer_id.groupBy(f.col("customer_id")).count()
customer_id = customer_id.orderBy("count", ascending = False)
customer_id.show()
```

```
+-----------+-----+
|customer_id|count|
+-----------+-----+
|      52832|   13|
|      47440|   10|
|      83287|    9|
|     250494|    8|
|      65009|    7|
|      82041|    7|
|     276192|    7|
|     259341|    7|
|     115213|    6|
|      97079|    6|
|     107909|    6|
|      97991|    6|
|     275689|    6|
|      60052|    6|
|      78939|    5|
|      94152|    5|
|      91817|    5|
|      62359|    5|
```

```
customer_id = customer_id.select(customer_id.columns).collect()
customer_id
```

```
Out[128]: [Row(customer_id=52832, count=13),
 Row(customer_id=47440, count=10),
 Row(customer_id=83287, count=9),
 Row(customer_id=250494, count=8),
 Row(customer_id=65009, count=7),
 Row(customer_id=82041, count=7),
 Row(customer_id=276192, count=7),
 Row(customer_id=259341, count=7),
 Row(customer_id=115213, count=6),
 Row(customer_id=97079, count=6),
 Row(customer_id=107909, count=6),
 Row(customer_id=97991, count=6),
 Row(customer_id=275689, count=6),
 Row(customer_id=60052, count=6),
 Row(customer_id=78939, count=5),
 Row(customer_id=94152, count=5),
 Row(customer_id=91817, count=5),
 Row(customer_id=81110, count=5),
 Row(customer_id=64153, count=5),
 Row(customer_id=142461, count=5),
 Row(customer_id=84087, count=5),
```

```
customer_id_col = [i['customer_id'] for i in customer_id]
customer_id_count = [i['count'] for i in customer_id]
```

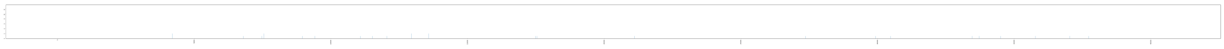
```
customer_id_col
```

```
Out[132]: [52832,
 47440,
 83287,
 250494,
 65009,
 82041,
 276192,
 259341,
 115213,
 97079,
 107909,
 97991,
 275689,
 60052,
 78939,
 94152,
 91817,
 81110,
 64153,
 142461,
```

```
customer_id_count
```

```
Out[133]: [13,
 10,
 9,
 8,
 7,
 7,
 7,
 7,
 6,
 6,
 6,
 6,
 6,
 6,
 5,
 5,
 5,
 5,
 5,
 5,
 5,
```

```
plt.figure(figsize=(180,5))
plt.bar(customer_id_col, customer_id_count);
plt.xticks(rotation=90)
plt.yticks(rotation=90)
plt.show()
```



```
# Observations:
* The highest number of orders was made by customer ID 52832 with 13 orders;
* the second highest is 47440 with 10 orders, and the third highest is 83287 with 9 orders
```

Multivariate Analysis

Perform a multivariate analysis to explore relationships between the important variables in the dataset.

```
column1 = "cost_of_the_order"
column2 = "food_preparation_time"

# Calculate the correlation between the two columns
correlation = df.select(f.corr(column1, column2))

# Show the correlation between the two columns
correlation.show()
```

```
+---------------------------------------------+
|corr(cost_of_the_order, food_preparation_time)|
+---------------------------------------------+
|                          0.04152747282774483|
+---------------------------------------------+
```

Cuisine vs Cost of the Order

```
df1 = df.groupBy("cuisine_type").agg(f.mean("cost_of_the_order").alias("Avrg"),
                                     f.min("cost_of_the_order").alias("Min"),
                                     f.max("cost_of_the_order").alias("Max"))
df1.show()
```

```
+--------------+------------------+-----+-----+
|  cuisine_type|              Avrg|  Min|  Max|
+--------------+------------------+-----+-----+
|       Mexican|16.933116883116877| 4.85|33.32|
|          Thai|19.207894736842103| 6.69|32.93|
|        Indian|16.919726027397267| 5.34|33.03|
|      Southern| 19.30058823529412| 7.38|31.43|
|       Chinese| 16.30520930232558| 4.75|34.19|
|      Japanese| 16.30453191489364| 4.47|33.37|
|       Spanish| 18.99416666666667|12.13| 29.1|
|    Vietnamese|12.882857142857143| 6.01|22.26|
|       Italian|16.418691275167788| 5.05|33.03|
|        Korean|14.001538461538464| 5.77|30.75|
|        French| 19.79388888888889|11.98|29.25|
|Middle Eastern|18.820612244897955| 5.77|32.93|
| Mediterranean|15.474782608695655| 5.67|35.41|
|      American|16.319828767123287| 4.71|33.18|
+--------------+------------------+-----+-----+
```

Observations:

- Vietnamese and Korean cuisines cost less compared to other cuisines.
- The boxplots for Italian, American, Chinese, Japanese cuisines are quite similar. This indicates
- that the quartile costs for these cuisines are quite similar.
- Outliers are present for the cost of Korean, Mediterranean and Vietnamese cuisines.
- French and Spanish cuisines are costlier compared to other cuisines

The company wants to analyze the delivery time of the orders on weekdays and weekends. How does the mean delivery time vary during weekdays and weekends?

```
means_time = df.groupBy("day_of_the_week").agg(f.mean("delivery_time").alias("Avrage Delivery time"))
means_time.show()
```

```
+---------------+--------------------+
|day_of_the_week|Avrage Delivery time|
+---------------+--------------------+
|        Weekday|  28.340036563071298|
|        Weekend|    22.4700222057735|
+---------------+--------------------+
```

```
data = df.createOrReplaceTempView("data")
```

```
%sql

select day_of_the_week, round(avg(delivery_time),2) from data
group by day_of_the_week
```

Table

	day_of_the_week	round(avg(delivery_time), 2)
1	Weekday	28.34
2	Weekend	22.47

```
2 rows
```

```
means_time_df = df.select("day_of_the_week", 'delivery_time')
means_time_df.show()
```

```
+---------------+-------------+
|day_of_the_week|delivery_time|
+---------------+-------------+
|        Weekend|           20|
|        Weekend|           23|
|        Weekday|           28|
|        Weekend|           15|
|        Weekday|           24|
|        Weekday|           24|
|        Weekend|           21|
|        Weekday|           30|
|        Weekday|           26|
|        Weekday|           26|
|        Weekday|           28|
|        Weekend|           22|
|        Weekend|           17|
|        Weekend|           23|
|        Weekday|           30|
|        Weekend|           25|
|        Weekend|           16|
|        Weekend|           22|
```

Observations:

- The mean delivery time on weekdays is around 28 minutes whereas the mean delivery time on weekends is around 22 minutes.
- This could be due to the dip of traffic volume in the weekends.

```
# The company wants to analyze the total time required to deliver the food. What percentage of orders take more than
60 minutes to get delivered from the time the order is placed? (The food has to be prepared and then delivered.)
```

```
df1 = df.withColumn('Total_time', df["food_preparation_time"] + df["delivery_time"])
df1.show()
```

```
+--------+-----------+--------------------+-------------+-----------------+---------------+------+-------------------
--+-------------+----------+
|order_id|customer_id|     restaurant_name| cuisine_type|cost_of_the_order|day_of_the_week|rating|food_preparation_ti
me|delivery_time|Total_time|
+--------+-----------+--------------------+-------------+-----------------+---------------+------+-------------------
--+-------------+----------+
| 1477147|     337525|             Hangawi|       Korean|            30.75|        Weekend|     0|
25|           20|        45|
| 1477685|     358141|Blue Ribbon Sushi...|     Japanese|            12.08|        Weekend|     0|
25|           23|        48|
| 1477070|      66393|         Cafe Habana|      Mexican|            12.23|        Weekday|     5|
23|           28|        51|
| 1477334|     106968|Blue Ribbon Fried...|     American|             29.2|        Weekend|     3|
25|           15|        40|
| 1478249|      76942|    Dirty Bird to Go|     American|            11.59|        Weekday|     4|
25|           24|        49|
| 1477224|     147468|    Tamarind TriBeCa|       Indian|            25.22|        Weekday|     3|
20|           24|        44|
| 1477894|     157711|   The Meatball Shop|      Italian|             6.07|        Weekend|     0|
28|           21|        49|
| 1477859|      89574|           Barbounia|Mediterranean|             5.97|        Weekday|     3|
```

```
time_df = df1.select("Total_time")
time_df.show()
```

```
+----------+
|Total_time|
+----------+
|        45|
|        48|
|        51|
|        40|
|        49|
|        44|
|        49|
|        63|
|        47|
|        55|
|        62|
|        45|
|        40|
|        47|
|        53|
|        58|
|        46|
```

```
time_df = time_df.select(time_df.columns).collect()
time_df
```

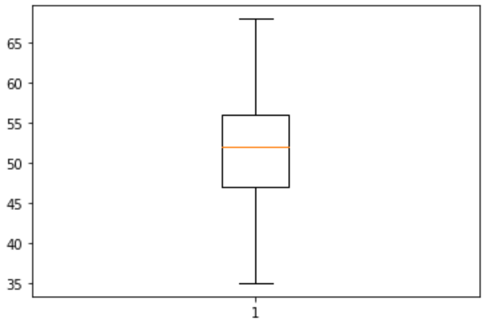
```
Out[173]: [Row(Total_time=45),
 Row(Total_time=48),
 Row(Total_time=51),
 Row(Total_time=40),
 Row(Total_time=49),
 Row(Total_time=44),
 Row(Total_time=49),
 Row(Total_time=63),
 Row(Total_time=47),
 Row(Total_time=55),
 Row(Total_time=62),
 Row(Total_time=45),
 Row(Total_time=40),
 Row(Total_time=47),
 Row(Total_time=53),
 Row(Total_time=58),
 Row(Total_time=46),
 Row(Total_time=43),
 Row(Total_time=49),
 Row(Total_time=61),
 Row(Total_time=45),
```

```
time_df = [i["Total_time"] for i in time_df]
time_df
```

```
Out[174]: [45,
 48,
 51,
 40,
 49,
 44,
 49,
 63,
 47,
 55,
 62,
 45,
 40,
 47,
 53,
 58,
 46,
 43,
 49,
 61,
 45,
```

```
plt.boxplot(time_df)
```

```
Out[175]: {'whiskers': [<matplotlib.lines.Line2D at 0x7fe788cfae20>,
  <matplotlib.lines.Line2D at 0x7fe787973dc0>],
 'caps': [<matplotlib.lines.Line2D at 0x7fe787973250>,
  <matplotlib.lines.Line2D at 0x7fe7879736d0>],
 'boxes': [<matplotlib.lines.Line2D at 0x7fe7879689d0>],
 'medians': [<matplotlib.lines.Line2D at 0x7fe78b879880>],
 'fliers': [<matplotlib.lines.Line2D at 0x7fe7877b2880>],
 'means': []}
```



```
%sql

select round(avg((food_preparation_time + delivery_time)),2) as Averge_time from data
```

Table

	Averge_time
1	51.53

1 row

Observations: Approximately 10.54 % of the total orders have more than 60 minutes of total delivery time.

```
+--------+-----------+--------------------+-------------+-----------------+---------------+------+-------------------
--+-------------+
|order_id|customer_id|     restaurant_name| cuisine_type|cost_of_the_order|day_of_the_week|rating|food_preparation_ti
me|delivery_time|
+--------+-----------+--------------------+-------------+-----------------+---------------+------+-------------------
--+-------------+
| 1477147|     337525|             Hangawi|       Korean|            30.75|        Weekend|     0|
25|           20|
| 1477685|     358141|Blue Ribbon Sushi...|     Japanese|            12.08|        Weekend|     0|
25|           23|
| 1477070|      66393|         Cafe Habana|      Mexican|            12.23|        Weekday|     5|
23|           28|
| 1477334|     106968|Blue Ribbon Fried...|     American|             29.2|        Weekend|     3|
25|           15|
| 1478249|      76942|    Dirty Bird to Go|     American|            11.59|        Weekday|     4|
25|           24|
| 1477224|     147468|    Tamarind TriBeCa|       Indian|            25.22|        Weekday|     3|
20|           24|
| 1477894|     157711|   The Meatball Shop|      Italian|             6.07|        Weekend|     0|
28|           21|
| 1477859|      89574|           Barbounia|Mediterranean|             5.97|        Weekday|     3|
```