Project Python Foundations: FoodHub Data Analysis

Marks: 60

Context

The number of restaurants in New York is increasing day by day. Lots of students and busy professionals rely on these 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 smart platform.

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 app to navigate the restaurant and route to the customer's place. Once the food package is handed over to the delivery person, he/she confirms the order on the app and starts the delivery. The customer can track the status of the order which is displayed on the app delivery partner has confirmed the pickup of the order and is en route to the customer's place.

Objective

The food aggregator company has recorded the data of the different orders made by the registered customers in their online portal. They want to analyze the data to get a first view of the data collected in different orders. The company wants to understand the customer base, the most popular cuisines, and the most and less busy days for the company and the customer base. The company also wants to understand the customer's ordering behavior. Perform the data analysis for the company to derive business insights that will help the company in the business.

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_of_the_order: Cost of the order
- day_of_the_order: Indicates whether the order was placed on a weekday or weekend (The weekday is from Monday to Friday and the weekend is Saturday and Sunday)
- 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 acceptance of the order and the order pick-up by the delivery partner.
- 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

Let us start by importing the required libraries

```
# Import numpy for array manipulation
import numpy as np
import numpy as np

# Import pandas for data manipulation
import pandas as pd
import pandas as pd
```

Understanding the structure of the data

```
# Load pandas and numpy libraries first before loading foodhub_order.csv
# Read the data into a variable called df
df = pd.read_csv('foodhub_order.csv')
df.head(5)
```

```
   order_id  customer_id  restaurant_name  cuisine_type  cost_of_the_order  \
0   1477147       337525          Hangawi       Korean              30.75
1   1477685       358141  Blue Ribbon Sushi     Japanese              12.08
2   1477070       66393  Cafe Mamoun express   Middle Eastern          5.92
3   1477334       106968  Blue Ribbon Fried     American               48.77
4   1478249       76942   Dirty Bird to Go      American               30.26

   day_of_the_order  food_preparation_time  delivery_time
0           Weekend                     25             20
1           Weekend                     23             28
2           Weekend                     25             15
3           Weekend                     25             24
4           Weekend                     20             24
```

Observations:

The DataFrame has 9 rows and 9 columns.

Question 1: How many rows and columns are present in the data? [0.5 mark]

```
df.shape
```

```
(1898, 9)
```

Observations:

The DataFrame has 1898 rows and 9 columns.

Question 2: What are the datatypes of the different columns in the dataset? (The info() function can be used) [0.5 mark]

```
# Use info() to print a concise summary of the dataset
df.info()
```

```
<class 'pandas.core.frame.DataFrame'>
RangeIndex: 1898 entries, 0 to 1897
Data columns (total 9 columns):
 #   Column                 Non-Null Count  Dtype
---  ------                 --------------  -----
 0   order_id               1898 non-null   int64
 1   customer_id            1898 non-null   int64
 2   restaurant_name        1898 non-null   object
 3   cuisine_type           1898 non-null   object
 4   cost_of_the_order      1898 non-null   float64
 5   day_of_the_order       1898 non-null   object
 6   rating                 1898 non-null   object
 7   food_preparation_time  1898 non-null   int64
 8   delivery_time          1898 non-null   int64
dtypes: float64(1), int64(5), object(3)
memory usage: 133.6 KB
```

Observations:

- The data type for rating is object. It contains decimals from zero 1898 non-null values for this dataset. There are 9 columns in the dataset
- There are integer type order_id, customer_id, food_preparation_time, and delivery_time
- There are 3 object type which are also considered string values (Restaurant_name, cuisine_type, day_of_the_order, and rating)
- There are 1 float type for cost_of_the_order. The data type for cost_of_the_order indicates that price values contains cent (integer).
- The memory usage is approximately 133.6 KB

Question 3: Are there any missing values in the data? If yes, treat them using an appropriate method. [1 mark]

```
# Check for missing values
df.isnull().sum()
```

```
order_id                 0
customer_id              0
restaurant_name          0
cuisine_type             0
cost_of_the_order        0
day_of_the_order         0
rating                   0
food_preparation_time    0
delivery_time            0
dtype: int64
```

Observations:

- The data does not have any missing values because all 1898 rows are non-null counts, as indicated with the df.info() code in question two.
- Additionally, the missing value for the column rating is not recorded as missing in the dataset.

Question 4: 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? [2 mark]

```
# Use describe() to print the statistical summary
df.describe
```

```
                order_id  customer_id  ...  food_preparation_time  delivery_time
count  1.898000e+03  1898.000000  ...            1898.000000    1898.000000
mean   1.477496e+06   171168.278602  ...              27.371970      24.161749
std    548.049724    113776.975671  ...               4.632481       4.972637
min    1.476547e+06    1311.000000  ...              20.000000      15.000000
25%    1.476971e+06   77787.750000  ...              23.000000      20.000000
50%    1.477494e+06  128520.000000  ...              27.000000      25.000000
75%    1.478064e+06  270525.000000  ...              31.000000      28.000000
max    1.478615e+06  405334.000000  ...              35.000000      33.000000

[8 rows x 6 rows]
```

Observations:

- The food preparation time ranges from a minimum of 20 minutes to a maximum of 35 minutes. The average time to prepare the food is approximately 27 minutes. The food prep time does not seem to be long.
- The delivery time ranges from a minimum of 15 minutes to a maximum of 33 minutes. The average delivery time is approximately 24 minutes.
- The delivery time is 3 times longer than the food preparation time.

Question 5: How many orders are not rated? [1 mark]

```
df[df['rating'] == 'Not given']
df['rating'].value_counts()
```

```
order_id, customer_id, restaurant_name, cuisine_type, cost_of_the_order, day_of_the_order, rating, food_preparation_time, delivery_time
Not given    736
Name: count, dtype: int64
```

Observations:

- 736 orders are not rated.

Exploratory Data Analysis (EDA)

Univariate Analysis

Question 6: Explore all the variables and provide observations on their distributions. (Generally, histograms, boxplots, countplots, etc. are used for univariate exploration.) [10 marks]

Order ID

```
# Number of unique values in order_id
df['order_id'].nunique()
```

```
1898
```

Observations:

- There are 1898 unique orders.

Customer ID

```
# Number of unique customers id
df['customer_id'].nunique()
```

```
1200
```

Observations:

- There are 1200 unique customers.

Restaurant Name

```
# Number of unique restaurant name
df['restaurant_name'].nunique()
```

```
178
```

Observations:

- There are 178 restaurants in the dataset.

Number of Orders

```
# Count number of orders from each restaurant
df['restaurant_name'].value_counts()
```

```
restaurant_name
Shake Shack                  219
The Meatball Shop            132
Blue Ribbon Sushi            119
Blue Ribbon Fried Chicken     96
Parm                          68
                            ...
Sushi Yasaki                   1
Soto                           1
Pylos                          1
Pump & Pantry                  1
Zero Otto                      1
Name: count, Length: 178, dtype: int64
```

Observations:

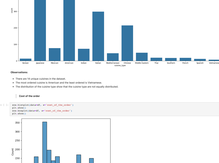
- The top 5 restaurants in the list receive most orders. With Shake Shack receiving the highest and Blue Ribbon receiving the lowest number of orders.

Cuisine Type

```
# Count number of cuisine type
df['cuisine_type'].nunique()
```

```
14
```

```
# Count plot for the cuisine type and the order id for the cuisine
plt.figure(figure=(8,4))
sns.countplot(data=df, x='cuisine_type')
```



Observations:

- The top three cuisine type in the dataset are American, Japanese, and Italian.
- The distribution of the cuisine type shows that the cuisine type are not equally distributed.

Cost of the order

```
# Count plot for the cost of the order
plt.figure(figure=(8,4))
sns.countplot(data=df, x='cost_of_the_order')
```

Observations:

- The highest count of the order is 300 orders.
- The lowest cost is 5 dollar.
- The distribution of the cost of the order ranges from 15-25 dollars.

Day of the week

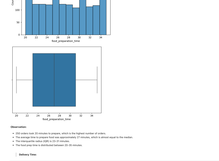
```
# Count number of the day of the week
df['day_of_the_order'].value_counts()
```

```
day_of_the_order
Weekend    1351
Weekday     547
Name: count, dtype: int64
```

Observations:

- The weekend count is more compared to the weekday count during the week. There are over twice as many orders on the weekends compared to the weekdays.
- The data is visualized in the countplot graph below.

```
sns.countplot(data=df, x='day_of_the_order')
```



Rating

```
# Count number of the rating
df['rating'].value_counts()
```

```
rating
Not given    736
5            588
4            386
3            188
Name: count, dtype: int64
```

Observations:

- There are 4 rating 1, 2, 3, and not given.
- The rating value of not given, and the lowest number of rating was 3.

Food Prep Time

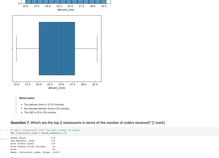
```
# Count number of food preparation time
df['food_preparation_time'].nunique()
```

```
16
```

```
# Count number of food preparation time
df['food_preparation_time'].value_counts()
```

```
food_preparation_time
20    187
21    172
22    159
23    148
24    132
25    123
26    116
27    114
28    112
29    111
30    109
31    100
32    100
33    100
34    100
35    100
Name: count, dtype: int64
```

```
# Count plot for the food preparation time
plt.figure(figure=(8,4))
sns.countplot(data=df, x='food_preparation_time')
```



Observations:

- 20 orders took 20 minutes to prepare, which is the highest number of orders.
- The average time it prepared food was approximately 27 minutes, which is almost equal to the median.
- The food prep time ranges from 20-35 minutes.
- The food prep time is distributed between 20-35 minutes.

Delivery Time

```
# Count number of the delivery time
df['delivery_time'].nunique()
```

```
11
```

```
# Count number of the delivery time
df['delivery_time'].value_counts()
```

```
delivery_time
20    215
21    207
22    201
23    196
24    188
25    179
26    171
27    163
28    156
29    148
30    140
31    132
32    124
33    116
Name: count, dtype: int64
```

```
# Count plot for the delivery time
plt.figure(figure=(8,4))
sns.countplot(data=df, x='delivery_time')
```

Observations:

- The delivery time is 15-33 minutes.
- The average delivery time is 24 minutes.
- The 20 min is the most frequent.

Question 7: Which are the top 5 restaurants in terms of the number of orders received? [1 mark]

```
# Count number of the order per restaurant
df['restaurant_name'].value_counts()
```

```
restaurant_name
Shake Shack                  219
The Meatball Shop            132
Blue Ribbon Sushi            119
Blue Ribbon Fried Chicken     96
Parm                          68
                            ...
Sushi Yasaki                   1
Soto                           1
Pylos                          1
Pump & Pantry                  1
Zero Otto                      1
Name: count, Length: 178, dtype: int64
```

Observations:

- The top 5 restaurants are Shake Shack, Blue Ribbon Sushi, Blue Ribbon Fried Chicken, Parm, and The Meatball Shop.
- With Shake Shack receiving the highest number of orders and Blue Ribbon receiving the lowest number of orders (shake shack, the meatball shop, blue ribbon sushi, blue ribbon fried chicken, Parm).

Question 8: Which is the most popular cuisine on weekends? [1 mark]

```
# Weekend popular cuisine
df[df['day_of_the_order'] == 'Weekend']
df['cuisine_type'].value_counts()
```

```
cuisine_type
American       415
Japanese       335
Italian        207
Chinese        163
Mexican         53
Indian          49
Middle Eastern  32
Mediterranean   30
French          24
Thai            19
Southern        17
Korean          11
Spanish         11
Vietnamese       4
Name: count, dtyp: int64
```

Observations:

- American cuisine is the most popular cuisine on weekends.
- Followed by Japanese, Italian, Chinese, Mexican, Indian, Middle Eastern, Mediterranean, French, Thai, Southern, Korean, Spanish, and Vietnamese.

Question 9: What percentage of the orders cost more than 20 dollars? [2 marks]

```
# Percentage of orders cost more than 20 dollars
df[df['cost_of_the_order'] > 20].shape[0]
# Divide the number of orders cost more than 20 dollars by the total number of orders
# Multiply the percentage of orders cost more than 20 dollars by 100
# Round the percentage of orders cost more than 20 dollars to 2 decimal
# Print the percentage of orders cost more than 20 dollars
print('The percentage of orders cost more than 20 dollars is', round(percentage, 2), '%')
```

```
The percentage of orders cost more than 20 dollars is 29.24 %
```

Observations:

- 555 orders cost more than 20 dollars.
- 29.24 % of orders cost more than 20 dollars.

Question 10: What is the mean order delivery time? [2 mark]

```
# Mean order delivery time
df['delivery_time'].mean()
# Round the mean order delivery time to 2 decimal
print('The mean order delivery time is', round(df['delivery_time'].mean(), 2), 'minutes')
```

```
The mean order delivery time is 24.16 minutes
```

Observations:

- The mean delivery time for this dataset is 24.16 minutes.

Question 11: 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. [1 mark]

```
# Find top 3 customers order count
df['customer_id'].value_counts()
```

```
customer_id
52832    13
47440    10
83287     9
250494    8
259341    7
                 ...
377306    1
376707    1
376537    1
376431    1
405334    1
Name: count, Length: 1200, dtype: int64
```

Observations:

- The most popular customer ids the customer's 52832.

Question 12: Perform a multivariate analysis to explore relationships between the important variables in the dataset. (It is a good idea to explore relations between numerical variables as well as relations between numerical and categorical variables) [10 marks]

Relationship between cuisine type and cost

```
# Relationship between cuisine type and cost
# Create a variable called cuisine_cost_rel
cuisine_cost_rel = df[['cuisine_type', 'cost_of_the_order']]
# Group the cuisine type and calculate the average cost of the order
cuisine_cost_rel.groupby('cuisine_type').mean()
```

```
               cost_of_the_order
cuisine_type
American               29.393027
Chinese                16.445767
Indian                 33.037551
Italian                28.191691
Japanese               32.159970
Mediterranean          30.314667
Mexican                31.077358
Middle Eastern         30.192250
French                 37.075833
Korean                 28.632727
Southern               28.296471
Spanish                31.284545
Thai                   30.746842
Vietnamese             28.225000
```

Observations:

- Korean food has the lowest average cost, and French food has the highest average cost.
- Mexican food is the second lowest.
- Japanese and Chinese food have the largest range of cost for food orders.
- Korean, Mediterranean, and Vietnamese food have similar.

Relationship between cuisine type and food prep time

```
# Create a variable called cuisine_prep_time
cuisine_prep_time = df[['cuisine_type', 'food_preparation_time']]
# Group the cuisine type and calculate the average food preparation time
cuisine_prep_time.groupby('cuisine_type').mean()
```

```
               food_preparation_time
cuisine_type
American                   27.398533
Chinese                    27.411043
Indian                     27.510204
Italian                    27.277778
Japanese                   27.386364
Mediterranean              27.333333
Mexican                    27.433962
Middle Eastern             27.375000
French                     27.666667
Korean                     27.272727
Southern                   27.294118
Spanish                    27.272727
Thai                       27.333333
Vietnamese                 27.000000
```

Observations:

- Vietnamese food has the shortest average food preparation time.
- French food has the longest.
- Japanese and Chinese food have the largest range of cost for food orders.
- Korean, Mediterranean, and Vietnamese food have similar.

Relationship between customer type and food prep time

```
# Create a variable called customer_prep_time
customer_prep_time = df[['customer_id', 'food_preparation_time']]
# Group the customer id and calculate the average food preparation time
customer_prep_time.groupby('customer_id').mean()
```

```
             food_preparation_time
customer_id
1311                         20.0
1320                         20.0
1335                         20.0
1339                         20.0
1350                         20.0
                             ...
405334                       27.0
406654                       27.0
406705                       27.0
406805                       27.0
407074                       27.0
[1200 rows x 1 rows]
```

Observations:

- Customer id 1311, 1320, 1335, 1339, 1350 have the shortest average food preparation time.
- Customer id 405334, 406654, 406705, 406805, 407074 have the longest average food preparation time.

Observations:

- Italian food has the highest average prep time, and Mexican, American, Indian, and Italian have the same IQR.
- 9 restaurants have a 20-35 minute food prep time.
- Korean food has the lowest food prep time, which is 20-29 minutes, and Korean food has the smallest IQR.
- Korean and Vietnames food have a 25 minute average food prep time, which is the lowest average food prep time amongst the 14 restaurants.
- Korean food has outliers with their food prep time.

Question 13: The company wants to provide a promotional offer in the advertisement of the restaurants. The condition to get the offer is that the restaurants must have a rating count of more than 50 and the average rating should be greater than 4. Find the restaurants fulfilling the criteria to get the promotional offer. [3 marks]

In [ ]:
```
# filter the rated restaurants
df_rated = df[df['rating'] != 'Not given'].copy()

# convert rating column from object to integer
df_rated['rating'] = df_rated['rating'].astype('int')

# create a dataframe that contains the restaurant names with their rating counts
df_rating_count = df_rated.groupby(['restaurant_name'])['rating'].count().sort_values(ascending = False).reset_index()
df_rating_count.head()
```

Out[ ]:

	restaurant_name	rating
0	Shake Shack	133
1	The Meatball Shop	84
2	Blue Ribbon Sushi	73
3	Blue Ribbon Fried Chicken	64
4	RedFarm Broadway	41

In [ ]:
```
# rating count more than 50
rest_names = df_rating_count[df_rating_count['rating'] > 50]['restaurant_name']
# filter to get the data of restaurants that have rating count more than 50
df_mean_4 = df_rated[df_rated['restaurant_name'].isin(rest_names)].copy()
# find the mean rating of the restaurants
df_mean_4.groupby(df_mean_4['restaurant_name'])['rating'].mean().sort_values(ascending = False).reset_index()
```

Out[ ]:

	restaurant_name	rating
0	The Meatball Shop	4.511905
1	Blue Ribbon Fried Chicken	4.328125
2	Shake Shack	4.278195
3	Blue Ribbon Sushi	4.219178

- The Meatball Shop, Blue Ribbon Fried Chicken, Shake Shack and Blue Ribbon Sushi meet the conditions to receive the promotional offer.

Observations:

Question 14: The company charges the restaurant 25% on the orders having cost greater than 20 dollars and 15% on the orders having cost greater than 5 dollars. Find the net revenue generated by the company across all orders. [3 marks]

In [6]:
```
# determine net revenue
def compute_rev(x):
    if x > 20:
        return x*.25
    elif x > 5:
        return x*.15

    else:
        return x*0

df['Revenue'] = df['cost_of_the_order'].apply(compute_rev)
df.head()
```

Out[6]:

	order_id	customer_id	restaurant_name	cuisine_type	cost_of_the_order	day_of_the_week	rating	food_preparation_time	delivery_time	Revenue
0	1477147	337525	Hangawi	Korean	30.75	Weekend	Not given	25	20	7.6875
1	1477685	358141	Blue Ribbon Sushi Izakaya	Japanese	12.08	Weekend	Not given	25	23	1.8120
2	1477070	66393	Cafe Habana	Mexican	12.23	Weekday	5	23	28	1.8345
3	1477334	106968	Blue Ribbon Fried Chicken	American	29.20	Weekend	3	25	15	7.3000
4	1478249	76942	Dirty Bird to Go	American	11.59	Weekday	4	25	24	1.7385

In [8]:
```
# total revenue
total_rev = df['Revenue'].sum()
print('The net revenue is around', round(total_rev, 2), 'dollars')
```

```
The net revenue is around 6166.3 dollars
```

Observations:

The net revenue is $6166.3.

Question 15: 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.) [2 marks]

In [12]:
```
#add column to datafram to store total delivery time
df['total_time'] = df['food_preparation_time'] + df['delivery_time']

# find the percentage taking longer than 60 minutes
print ('The percentage of orders that have more than 60 minutes of total delivery time is',
        round(df[df['total_time'] > 60].shape[0] / df.shape[0] * 100, 2), '%')
```

```
The percentage of orders that have more than 60 minutes of total delivery time is 10.54 %
```

Observations:

Question 16: 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? [2 marks]

In [ ]:
```
# Write the code here
```

Observations:

10.54% of total orders have more than 60 minutes of total delivery time.

Conclusion and Recommendations

Question 17: What are your conclusions from the analysis? What recommendations would you like to share to help improve the business? (You can use cuisine type and feedback ratings to drive your business recommendations.) [6 marks]

Conclusions:

- The net revenue is $6166.3.
- The Meatball Shop, Blue Ribbon Fried Chicken, Shake Shack and Blue Ribbon Sushi have more than 50 ratings and are rated higher than 4.
- The top 5 restaurants are shake shack, the meatball shop, blue ribbon sushi, blue ribbon fried chicken, Parm.
- More than double of the orders occur on the weekend.
- American, Japanese, Italian, and Chinese food have the most orders.
- 737 orders out of 1898 orders are not rated.

Recommendations:

- FoodHub needs to give an incentive to customers who provide ratings so there is not a high missing rating gap with the orders.
  - Suggestion: Customers provide a rating within 24 hours of delivery, every 3 ratings provide a 10-20% discount on the next order.
- FoodHub needs to look into the data for poor performing and high performing restaurants, and can share the data with the restaurants which details possible improvements (could be like a partnership with these restaurants to help understand customers/improve business operations).
  - This data that is shared can improve the customer experience/food if the improvements are made by the restaurants. This will be like a business partnership where FoodHub has data experts that can share business improvement plans/suggestions with restaurants monthly or quarterly. This is good for the bottom line for both the restaurants and FoodHub, and will positively impact the customer experience.
- Incentives for top performing restaurants.
  - Suggestion: Each month restaurants who receive 20+ ratings with an average of 4+ are promoted on the app and FoodHub allows customers with 10% off orders on certain days of the week.
  - Suggestion: Restaurants with average orders of $20+ and an average of 4 or more stars receive recognition from FoodHub as a "Fan Favorite Restaurant" (or whatever fancy/catchy award will attract more customers and more restaurants to achieve this goal. This award can be displayed by companies on their front door/window to show to customers and on their website).
- To drive more business on the weekdays FoodHub can provide discounts for larger orders.
  - 10% off $25 or more on the weekdays.
- American, Japanese, Italian, and Chinese food have the most orders, so FoodHub needs to incorporate more restaurants within their app so they gain more orders.

_ FoodHub needs to provide incentives for low performing food deliveries (such as Vietnamese, Southern, French, Thai, and etc.). How can FoodHub gain more interest for these foods? Could there be "Thai tuesday" themed deals, or "Southern Saturday" themed deals? This is an opportunity to create excitement for customers to try new restaurants.