# Project Python Foundations: FoodHub Data Analysis

## Context

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.

## 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_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

## Let us start by importing the required libraries

In [ ]:
```
# Installing the libraries with the specified version.
!pip install numpy==1.25.2 pandas==1.5.3 matplotlib==3.7.1 seaborn==0.13.1 -q
--user
```

```
   ━━━━━━━━━━━━━━━━━━━━━━━━━━━━━━━━━━━━━━━━ 18.2/18.2 MB 37.6 MB/s eta 0:00:0
0
   ━━━━━━━━━━━━━━━━━━━━━━━━━━━━━━━━━━━━━━━━ 12.0/12.0 MB 66.0 MB/s eta 0:00:0
0
   ━━━━━━━━━━━━━━━━━━━━━━━━━━━━━━━━━━━━━━━━ 11.6/11.6 MB 54.7 MB/s eta 0:00:0
0
   ━━━━━━━━━━━━━━━━━━━━━━━━━━━━━━━━━━━━━━━━ 294.8/294.8 kB 16.4 MB/s eta 0:0
0:00
  WARNING: The scripts f2py, f2py3 and f2py3.11 are installed in '/root/.loca
l/bin' which is not on PATH.
  Consider adding this directory to PATH or, if you prefer to suppress this w
arning, use --no-warn-script-location.
ERROR: pip's dependency resolver does not currently take into account all the
packages that are installed. This behaviour is the source of the following de
pendency conflicts.
google-colab 1.0.0 requires pandas==2.2.2, but you have pandas 1.5.3 which is
incompatible.
plotnine 0.14.5 requires matplotlib>=3.8.0, but you have matplotlib 3.7.1 whi
ch is incompatible.
plotnine 0.14.5 requires pandas>=2.2.0, but you have pandas 1.5.3 which is in
compatible.
tensorflow 2.18.0 requires numpy<2.1.0,>=1.26.0, but you have numpy 1.25.2 wh
ich is incompatible.
xarray 2025.1.2 requires pandas>=2.1, but you have pandas 1.5.3 which is inco
mpatible.
mizani 0.13.1 requires pandas>=2.2.0, but you have pandas 1.5.3 which is inco
mpatible.
cudf-cu12 24.12.0 requires pandas<2.2.4dev0,>=2.0, but you have pandas 1.5.3
which is incompatible.
```

**Note**: *After running the above cell, kindly restart the notebook kernel and run all cells sequentially from the start again.*

In [2]:
```
# import libraries for data manipulation
import numpy as np
import pandas as pd

# import libraries for data visualization
import matplotlib.pyplot as plt
import seaborn as sns
```

## Understanding the structure of the data

In [3]:
```
# uncomment and run the following lines for Google Colab
from google.colab import drive
drive.mount('/content/drive')
path = '/content/drive/MyDrive/AI Class/Projects/Project 1'
```

```
Mounted at /content/drive
```

In [4]:
```
# Write your code here to read the data
data = pd.read_csv('/content/drive/MyDrive/AI Class/Projects/Project 1/foodhub
_order.csv')
```

In [5]:
```
# Write your code here to view the first 5 rows
data.head(5)
```

Out[5]:

| | order_id | customer_id | restaurant_name | cuisine_type | cost_of_the_order | day_of_the_week | rat |
|---|---|---|---|---|---|---|---|
| **0** | 1477147 | 337525 | Hangawi | Korean | 30.75 | Weekend | gi |
| **1** | 1477685 | 358141 | Blue Ribbon Sushi Izakaya | Japanese | 12.08 | Weekend | gi |
| **2** | 1477070 | 66393 | Cafe Habana | Mexican | 12.23 | Weekday | |
| **3** | 1477334 | 106968 | Blue Ribbon Fried Chicken | American | 29.20 | Weekend | |
| **4** | 1478249 | 76942 | Dirty Bird to Go | American | 11.59 | Weekday | |

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

In [6]:
```
# Write your code here
data.shape
```

Out[6]:
```
(1898, 9)
```

#### Observations:

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

In [7]:
```
# Write your code here
data.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_week        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(4), object(4)
memory usage: 133.6+ KB
```

**Observations:**

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

In [8]:
```
data['rating'] = data['rating'].replace(['Not given'], np.nan)
data['rating'] = data['rating'].astype(float)
#Replacing all the not given entries with NaN, there are 736 rows with no rati
ng
#But they had all the other data so we cant remove it but just convert it to N
aN


nan_ratings_df = data[data['rating'].isna()]
nan_ratings_df
#Displaying all the rows with NaN


# unique_values = data['delivery_time'].unique()
# unique_values = data['delivery_time'].value_counts(dropna=False)
# unique_values = data[data['delivery_time'].isna()]

# Used the top 3 lines to go through every column to make sure there werent an
y missing values
# Only Ratings was off in any way, the math on all of them added up right
```

Out[8]:

| | order_id | customer_id | restaurant_name | cuisine_type | cost_of_the_order | day_of_the_week |
|---|---|---|---|---|---|---|
| **0** | 1477147 | 337525 | Hangawi | Korean | 30.75 | Weekend |
| **1** | 1477685 | 358141 | Blue Ribbon Sushi Izakaya | Japanese | 12.08 | Weekend |
| **6** | 1477894 | 157711 | The Meatball Shop | Italian | 6.07 | Weekend |
| **10** | 1477895 | 143926 | Big Wong Restaurant □_¤¾Ñ¼ | Chinese | 5.92 | Weekday |
| **14** | 1478198 | 62667 | Lucky's Famous Burgers | American | 12.13 | Weekday |
| **...** | ... | ... | ... | ... | ... | ... |
| **1887** | 1476873 | 237616 | Shake Shack | American | 5.82 | Weekend |
| **1891** | 1476981 | 138586 | Shake Shack | American | 5.82 | Weekend |
| **1892** | 1477473 | 97838 | Han Dynasty | Chinese | 29.15 | Weekend |
| **1895** | 1477819 | 35309 | Blue Ribbon Sushi | Japanese | 25.22 | Weekday |
| **1897** | 1478056 | 120353 | Blue Ribbon Sushi | Japanese | 19.45 | Weekend |

736 rows × 9 columns

**Observations:**

### 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 marks]

```
In [9]: # Write your code here
        data['food_preparation_time'].min()
        data['food_preparation_time'].mean()
        data['food_preparation_time'].max()

        #fairly straightforward mathematics
```

```
Out[9]: 35
```

**Observations:**

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

```
In [10]: data['rating'].isna().sum()

         # Had been using isnull(), but since i am using nan it makes more sense to use
         this than that
```

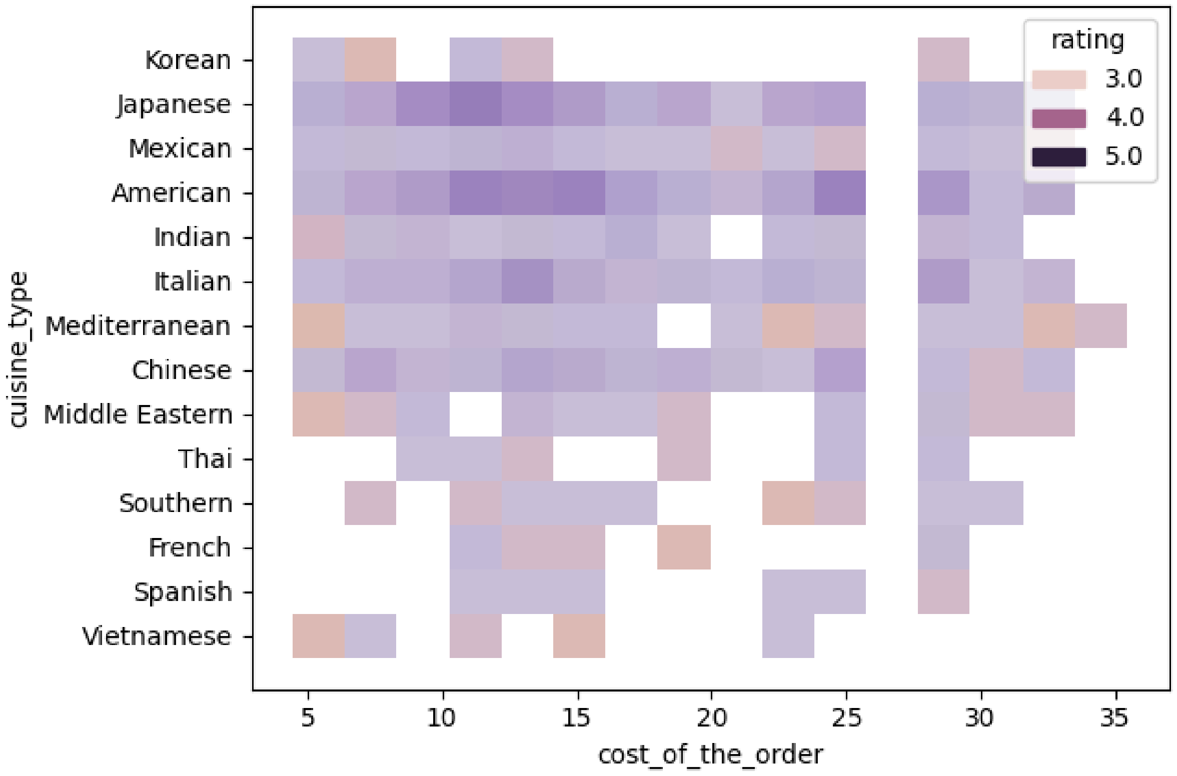
```
Out[10]: 736
```

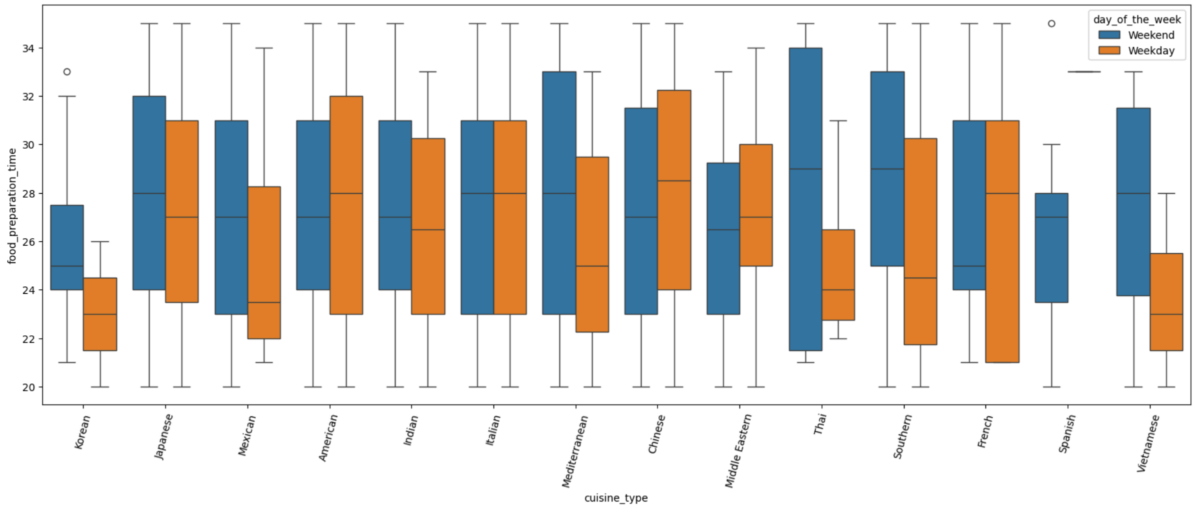
## Exploratory Data Analysis (EDA)

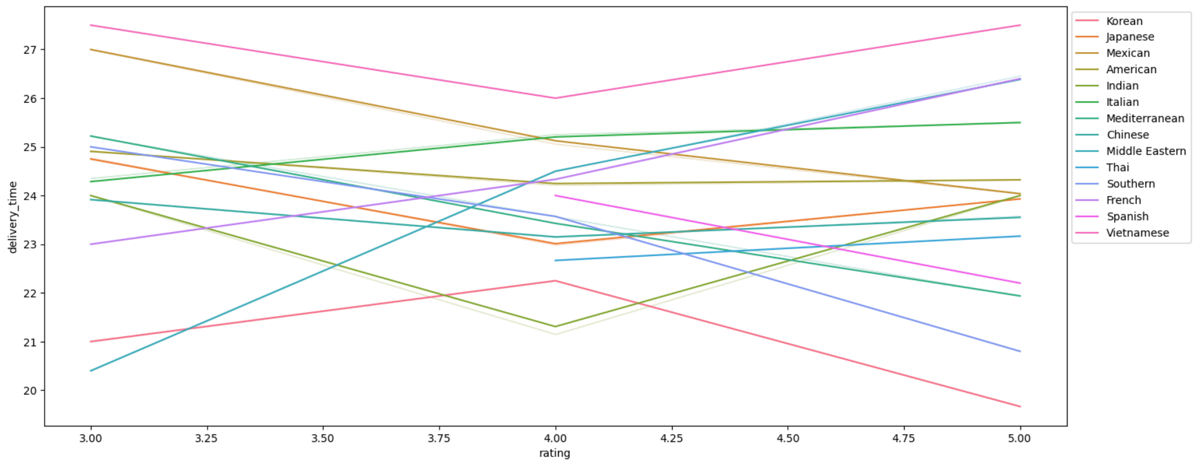
### Univariate Analysis

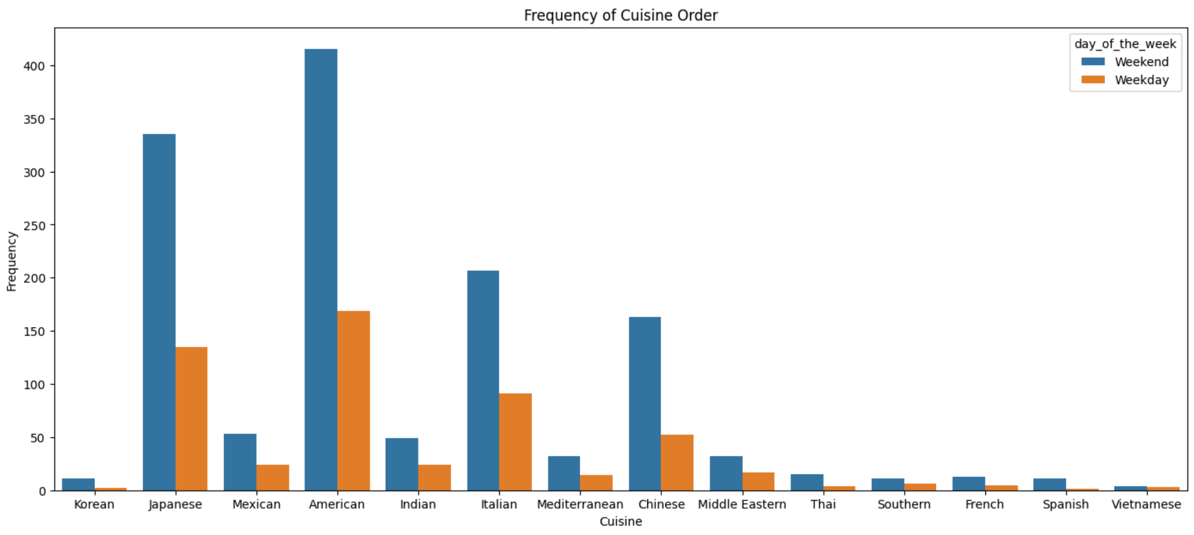
### Question 6: Explore all the variables and provide observations on their distributions. (Generally, histograms, boxplots, countplots, etc. are used for univariate exploration.) [9 marks]

```
In [48]: # Write the code here

         sns.histplot(data=data, x = 'cost_of_the_order', y = 'cuisine_type', hue = 'ra
         ting')

         # showed the correlation of cuisine type, cost and rating
         # -------------------------------------------------------------------




         plt.figure(figsize=(20,7))
         sns.boxplot(data = data, x = data['cuisine_type'], y = 'food_preparation_tim
         e', hue = data['day_of_the_week']);
         plt.xticks(rotation = 75);

         # using a boxplot, I showed the bounds of the food prep time of each cuisine f
         or each time of week, showing the upper, lower and outliers of each cuisine


         # -------------------------------------------------------------------



         plt.figure(figsize=(17,7));
         sns.lineplot(data = data, x = data['rating'], y = data['delivery_time'], hue =
         data['cuisine_type'], errorbar=('ci', False));
         plt.legend(bbox_to_anchor=[1,1]);

         # using a lineplot, I showed the correlation of ratings for each cuisine based
         on their delivery time


         # -------------------------------------------------------------------


         data['cuisine_order_count'] = data.groupby('cuisine_type')['order_id'].count()
         data = data.drop('cuisine_order_count', axis = 1)

         plt.figure(figsize=(17,7));
         sns.countplot(data = data, x = 'cuisine_type', hue = 'day_of_the_week');
         plt.title('Frequency of Cuisine Order');
         plt.xlabel('Cuisine');
         plt.ylabel('Frequency');

         # using a countplot, I showed the frequency of order for each cuisine dependin
         g on the time of week, very lope sided
```


rating
3.0
4.0
5.0
Korean
Japanese
Mexican
American
Indian
Italian
Mediterranean
Chinese
Middle Eastern
Thai
Southern
French
Spanish
Vietnamese
cuisine_type
5
10
15
20
25
30
35
cost_of_the_order


day_of_the_week
Weekend
Weekday
food_preparation_time
cuisine_type
Korean
Japanese
Mexican
American
Indian
Italian
Mediterranean
Chinese
Middle Eastern
Thai
Southern
French
Spanish
Vietnamese


Korean
Japanese
Mexican
American
Indian
Italian
Mediterranean
Chinese
Middle Eastern
Thai
Southern
French
Spanish
Vietnamese
delivery_time
rating



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

In [12]:
```
# Write the code here
rest_orders = data.groupby(['restaurant_name'])['order_id'].count()
top_5_rests = rest_orders.sort_values(ascending=False).head(5)
top_5_rests

# Grouped order ID and restaurant name and then counted them, but made sure to
do asc = false, then used head(5) to get top 5
```

Out[12]:

| | order_id |
|---|---|
| **restaurant_name** | |
| **Shake Shack** | 219 |
| **The Meatball Shop** | 132 |
| **Blue Ribbon Sushi** | 119 |
| **Blue Ribbon Fried Chicken** | 96 |
| **Parm** | 68 |

**dtype:** int64

**Observations:**

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

In [13]:
```python
# Write the code here
filtered_data = data[data['day_of_the_week'] == 'Weekend']

pop_cus = filtered_data.groupby(['cuisine_type'])['order_id'].count()

pop_cus_sorted = pop_cus.sort_values(ascending=False).head(1)

pop_cus_sorted

# Assigned the filtered data to df, so only weekend would show up
# Since I did groupby cuisine type and order ID it would return a series, whic
h I counted
# Took all the popular cuisines and sorted them, not ascending and took the he
ad(1), gets me the top cuisine
```

Out[13]:

|  | order_id |
|---|---|
| **cuisine_type** |  |
| **American** | 415 |

**dtype:** int64

**Observations:**

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

In [14]:
```
from itertools import count
# Write the code here

filtered_data = data[data['cost_of_the_order'] > 20.0]
# filtered_data

price_set = filtered_data.groupby(['order_id'])['cost_of_the_order'].count()
# price_set

count_price_set = price_set.count()
count_price_set

tot_cols = data['order_id'].count()
final_percent = (count_price_set / tot_cols)

final_percent = round(final_percent, 2)
final_result = str(final_percent) + '% of the orders cost more than $20'
final_result

# Returned a df that had all the orders with cost > $20
# Grouped the order id and cost columns to get the total number of orders > $2
0, that returned 555 orders
# Got the total number of rows then divided that by qualifying orders > $20 to
get the % of orders
# Then formatted it to return only 2 decimal places to look nicer and turned i
t into a string to add % char
```

Out[14]:
```
'0.29% of the orders cost more than $20'
```

**Observations:**

## Question 10: What is the mean order delivery time? [1 mark]

In [15]:
```
# Write the code here
mean_time = data['delivery_time'].mean()
formatted_percent = f"{mean_time:.2f} mins"
formatted_percent

# Got the mean of the whole delivery time column
# formatted it and returned it
```

Out[15]:
```
'24.16 mins'
```

**Observations:**

## 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]

In [16]:
```
# Write the code here

top_custs = data.groupby(['customer_id'])['order_id'].count()

top_3_custs = top_custs.sort_values(ascending=False).head(3)

top_3_custs

# Grouped the customer ID and order ID cols and counted them, this would show
the frequency of orders for each customer
# Took that series and sorted it, got descending values and returned top 3 cus
tomers
```

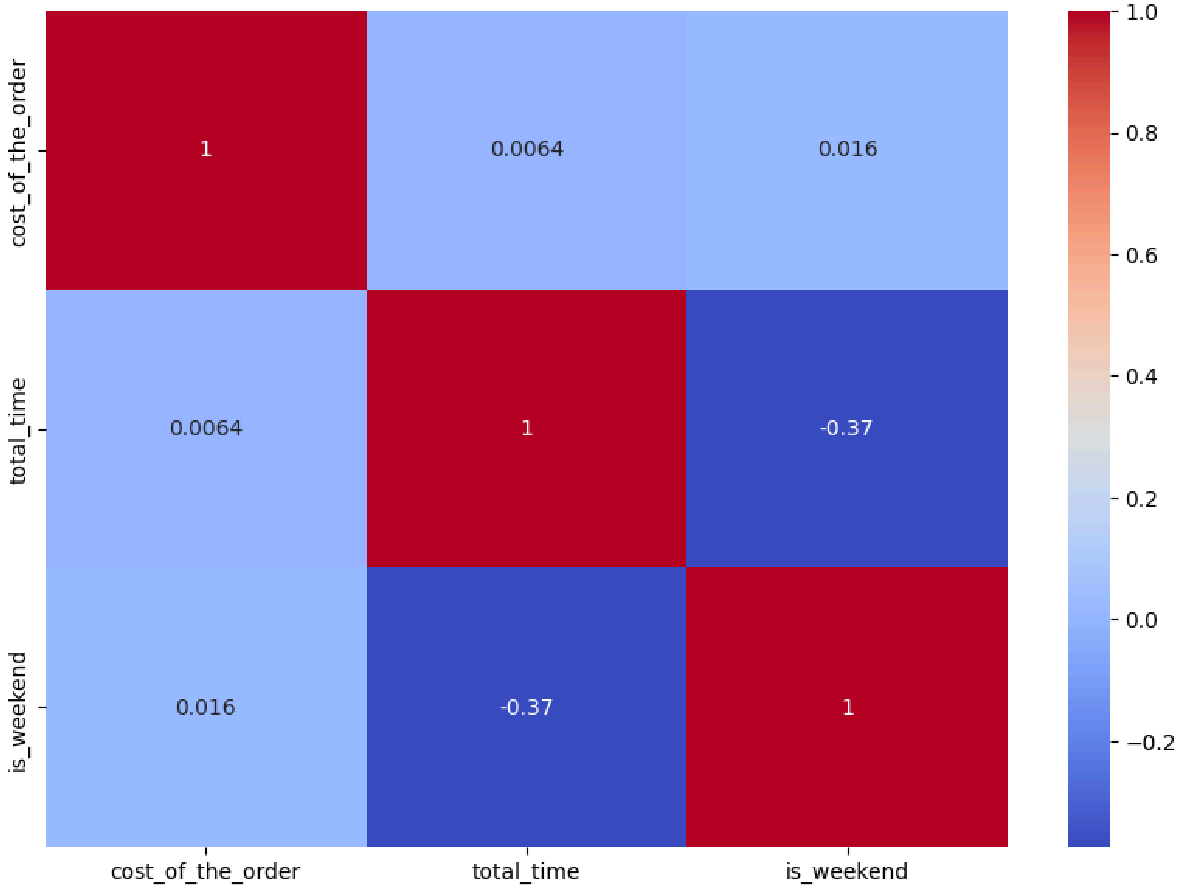
Out[16]:
```
             order_id
customer_id
      52832        13
      47440        10
      83287         9

dtype: int64
```

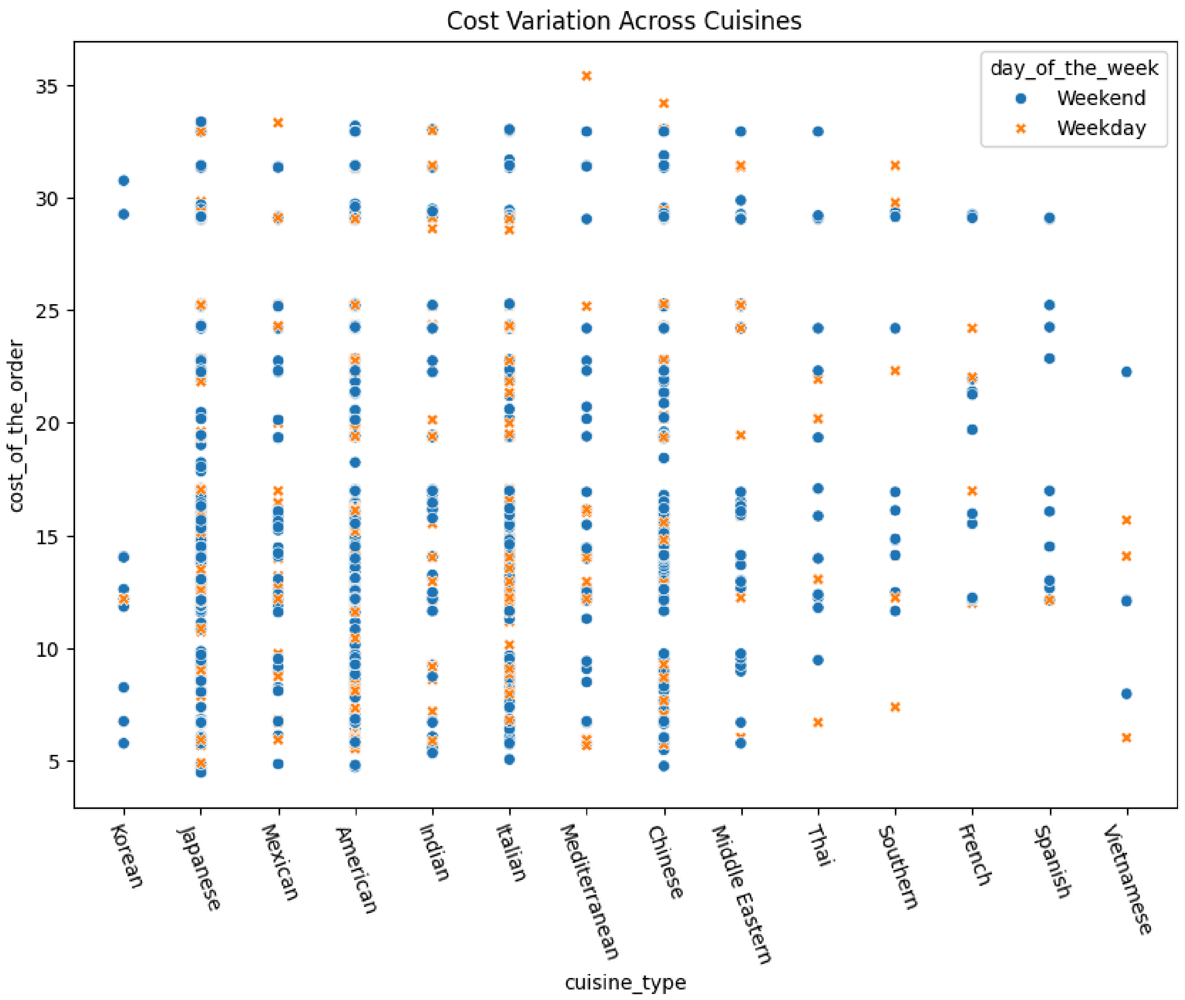
**Observations:**

## Multivariate Analysis

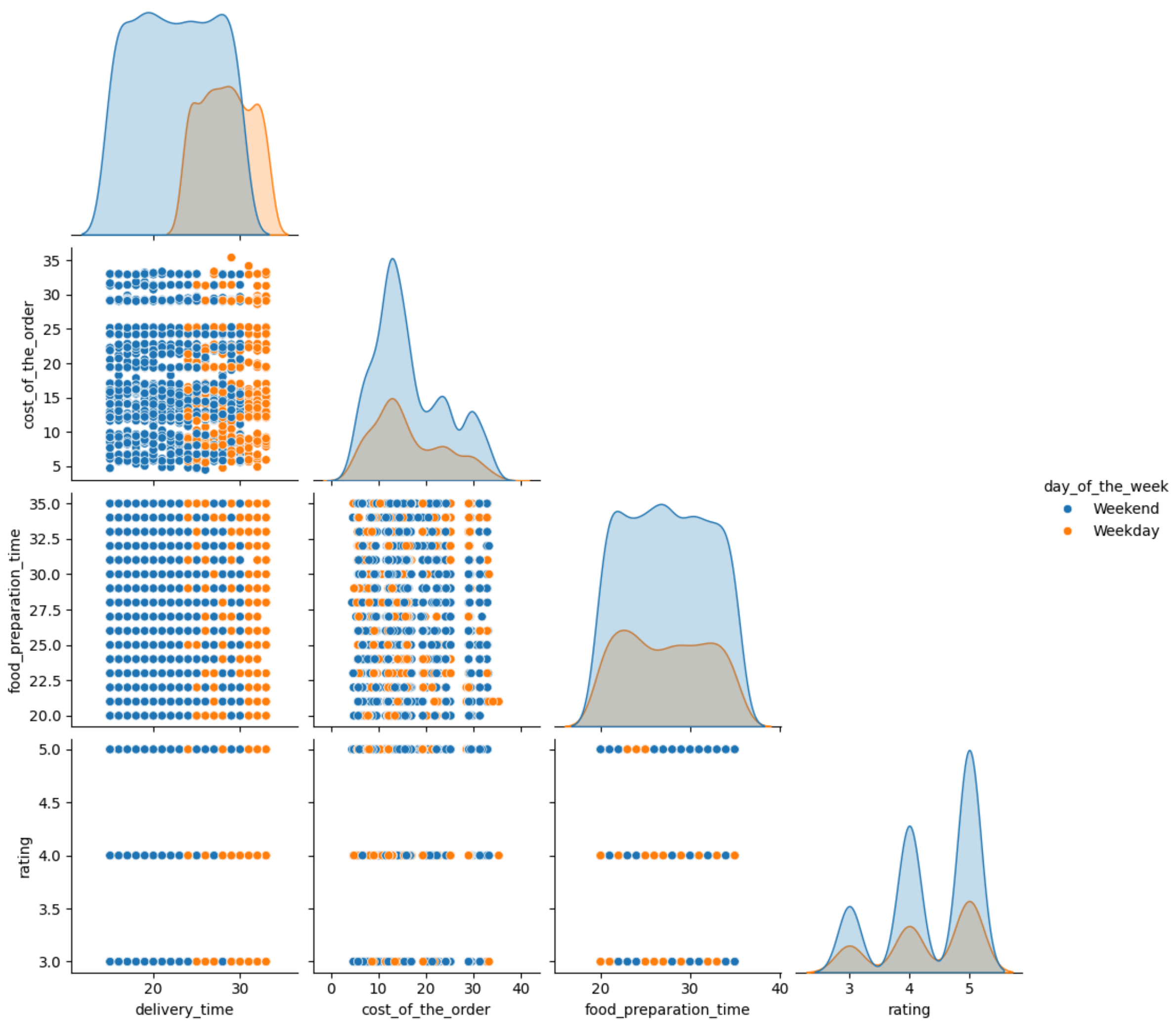
## Question 12: Perform a multivariate analysis to explore relationships between the important variables in the dataset. (It is a good idea to explore relations between numerical variables as well as relations between numerical and categorical variables) [10 marks]

In [49]:
```
data['total_time'] = data['food_preparation_time'] + data['delivery_time']
data['is_weekend'] = (data['day_of_the_week'] == 'Weekend').astype(int)
corr_square = data[['cost_of_the_order', 'total_time', 'is_weekend']].corr()
plt.figure(figsize=(10,7))
sns.heatmap(corr_square, annot = True, cmap = 'coolwarm')
plt.show()

# Combined the time columns to get a total time
# Converted the day of the week to 0 and 1
# made a heatmap to show the correlation
# -----------------------------------------------


plt.figure(figsize=(10,7))
sns.scatterplot(data = data, x = 'cuisine_type', y = 'cost_of_the_order', hue
= 'day_of_the_week', style = 'day_of_the_week');
plt.xticks(rotation = -70)
plt.title('Cost Variation Across Cuisines');

# scatterplot to show the correlation of cost of the order over the cuisines o
n day of the week

# -----------------------------------------------


plt.figure(figsize=(15,10))
sns.pairplot(data = data, vars=['delivery_time','cost_of_the_order','food_prep
aration_time','rating'], hue = 'day_of_the_week', corner = True);

# pairplot to show correlation between numerical vars with the hues being day
of week to show the difference
```


cost_of_the_order
total_time
is_weekend
1
0.0064
0.016
-0.37
1.0
0.8
0.6
0.4
0.2
0.0
−0.2


Cost Variation Across Cuisines
day_of_the_week
● Weekend
× Weekday
cost_of_the_order
35
30
25
20
15
10
5
Korean
Japanese
Mexican
American
Indian
Italian
Mediterranean
Chinese
Middle Eastern
Thai
Southern
French
Spanish
Vietnamese
cuisine_type

```
<Figure size 1500x1000 with 0 Axes>
```


cost_of_the_order
food_preparation_time
rating
delivery_time
cost_of_the_order
food_preparation_time
rating
day_of_the_week
Weekend
Weekday

## 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 [18]:
```
# Write the code here
rest_with_50_ratings = data.groupby(['restaurant_name'])['rating'].count()
rest_with_50_ratings = rest_with_50_ratings[rest_with_50_ratings > 50.0]

rest_with_4_stars = data.groupby(['restaurant_name'])['rating'].mean()
rest_with_4_stars = rest_with_50_ratings[rest_with_4_stars > 4.0]

qualified_rests = rest_with_4_stars.loc[rest_with_50_ratings.index]
qualified_rests


# grouped the restaurants with their ratings and counted them
# sorted through them to only return ones with more than 50 ratings
# got the mean for all the restaurants with ratings
# went through the series with 50 ratings that had above a 4.0 rating and retu
rned it
# using loc, returned only the restaurants that fulfilled both requirements an
d returned that series
```

Out[18]:
```
                         rating
restaurant_name
Blue Ribbon Fried Chicken    64
       Blue Ribbon Sushi     73
             Shake Shack    133
        The Meatball Shop    84

dtype: int64
```

**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 [19]:
```
# Write the code here9
orders_20_over = data[data['cost_of_the_order'] > 20.0]
total_sum_for_25_percent = orders_20_over['cost_of_the_order'].sum()
total_sum_for_25_percent = total_sum_for_25_percent * 0.25

orders_5_over = data[(data['cost_of_the_order'] > 5.0) & (data['cost_of_the_or
der'] <= 20.0)]
total_sum_for_15_percent = orders_5_over['cost_of_the_order'].sum()
total_sum_for_15_percent = total_sum_for_15_percent * 0.15

net_revenue = total_sum_for_15_percent + total_sum_for_25_percent
formatted_revenue = f"The net revenue is ${net_revenue:.2f} after combining th
e company charges"
formatted_revenue

# went through the cost of order column to return a df with orders > $20
# got the sum of those orders and multiplied it with 25%
# did the same thing for orders greater than $5 but less than $20 to follow th
e requirement
# got the sum for ordes >$5
# added them together to get net revenue
```

Out[19]:
```
'The net revenue is $6166.30 after combining the company charges'
```

**Observations:**

## 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 [20]:
```
# Write the code here
food_prep_time = data['food_preparation_time']
delivery_time = data['delivery_time']

total_time = food_prep_time + delivery_time
filtered_orders = data[total_time > 60]

filtered_count = filtered_orders.shape[0]

total_orders = data.shape[0]

percent = (filtered_count / total_orders) * 100
formatted_percent = f"Only {percent:.2f}% of the orders take more than 60 minu
tes to get to the customer once order is placed"
formatted_percent

# got the food prep and delivery times, assigned them to a var and added them
up
# filtered that var through the df to get only the ones that are > 60 mins
# got the total number of rows for the order > 60 mins
# got the total number of rows for the entire column
# divided those 2 numbers to get percentage
# formatted it and returned to look nice
```

Out[20]:
```
'Only 10.54% of the orders take more than 60 minutes to get to the customer o
nce order is placed'
```

### 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 [27]:
```
# Write the code here

weekend = data[data['day_of_the_week'] == "Weekend"]['delivery_time'].mean()
weekday = data[data['day_of_the_week'] == "Weekday"]['delivery_time'].mean()

difference_delivery_time = weekday - weekend

f"During the week, it takes {weekday:.0f} minutes for the food to get to the c
ustomer. While on the weekend it takes {weekend:.0f} minutes. Making it {diffe
rence_delivery_time:.0f} minutes longer on the weekday than the weekend."

# got the weekend mean time and the weekday time and assigned them to vars
# got the difference
# then displayed it while showing the mean time difference between the two and
which time of week is faster with deliveries
```

Out[27]:
```
'During the week, it takes 28 minutes for the food to get to the customer. Wh
ile on the weekend it takes 22 minutes. Making it 6 minutes longer on the wee
kday than the weekend.'
```

#### Observations:

In [ ]:

## 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:

- For Q15, there is a small percentage of orders that took longer than an hr to complete, which should be addressed.
- There is a noticable difference between weekday and weekend orders being placed. It is odd to find that weekend deliveries take shorter time to complete compared to weekday orders.
- There is a good chunk of orders that dont have ratings, it would be ideal for customers to give one for businesses to know how they're doing and get more customers.
- There is a clear customer favorite when ordering food online, shake shack.

- Even on weekends, the cuisine most people are going to seems to be something customers can rely on, they're not taking risks trying other cuisines.
- There were a few customers that ordered several times, if the company was able to offer more coupons or deals that would increase engagement and revenue.
- To get customers to rate the transaction, offer a coupon if rating is completed.
- Try to get more engagement from lower performing restaurants to boost their business.
- Try to close the gap between weekend and weekday delivery times, people would notice instantly.
- Create a baseline for certain cuisines to prepare their food within a certain time window, so customers are happy to return.
- Introduce a loyalty program for frequent customers as well as spotty customers, it would keep them in the loop and incline them to order more.
- Have coupons for certain restaurants/cuisines for when they're slow to increase revenue and to get their name out.
- Have more push notifs on the app during certain times of the week to entice customers to try a new restaurant.
- Its all about getting new customers and keeping them and while also retaining old ones. What can FoodHub do to keep everyone happy.

## Recommendations:

-