Exploratory 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.

Data Description

The data contains the different data related to a food order.

- 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

In [1]:
```
# 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

# removing warnings
import warnings
warnings.filterwarnings("ignore")


# to restrict the float value to 2 decimal places (specially useful when looking at the statisti
pd.set_option('display.float_format', lambda x: '%.2f' % x)
```

```
C:\ProgramData\anaconda3\lib\site-packages\pandas\core\arrays\masked.py:60: UserWarning: Pandas
requires version '1.3.6' or newer of 'bottleneck' (version '1.3.5' currently installed).
  from pandas.core import (
```

Understanding the structure of the data

In [3]:
```
# Read the data from local drive
df = pd.read_csv('foodhub_order.csv')

# returns the first 5 rows
df.head()
```

Out[3]:

	order_id	customer_id	restaurant_name	cuisine_type	cost_of_the_order	day_of_the_week	rating	food_preparat
0	1477147	337525	Hangawi	Korean	30.75	Weekend	Not given	
1	1477685	358141	Blue Ribbon Sushi Izakaya	Japanese	12.08	Weekend	Not given	
2	1477070	66393	Cafe Habana	Mexican	12.23	Weekday	5	
3	1477334	106968	Blue Ribbon Fried Chicken	American	29.20	Weekend	3	
4	1478249	76942	Dirty Bird to Go	American	11.59	Weekday	4	

Observations:

The DataFrame has 9 columns as mentioned in the Data Dictionary. Data in each row corresponds to the order placed by a customer.

In [4]:
```
# Rows and column quantity
rows, columns = df.shape

print(f'The Dataset contains {rows} rows and {columns} columns')
```

```
The Dataset contains 1898 rows and 9 columns
```

In [5]:
```
# Use info() to print a concise summary of the DataFrame
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_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
```

Notes:

- From the 9 columns present in the dataset, four of them are integers, being these the "order_id", "customer_id", "food_preparation_time" and "delivery_time".
- Four other columns have datatypes as object, being these the "restaurant_name", "cuisine_type", "day_of_the_week" and "rating".
- One column is a float (or decimal) being this one the "cost_of_the_order".
- From this, we can see that column "rating" column could be changed to an integer to facilitate mathematical calculations, such as the average rating of each restaurant. To do this, we would need to replace the string values like "Not given" and any other possible value that does not represent a number.

In [6]:
```
# Summing the missing values
missing_values = df.isnull().sum()

print('Missing values per column:')
print(missing_values)
```

```
Missing values per column:
order_id                0
customer_id             0
restaurant_name         0
cuisine_type            0
cost_of_the_order       0
day_of_the_week         0
rating                  0
food_preparation_time   0
delivery_time           0
dtype: int64
```

Notes:

- As we can see from Question 2, when using info() we get the count of values per column, and all of them denotates the same number.
- Now, using the sum of isnull gives 0 values, denotating no missing values in any column of the dataset.
- However, as we saw from previous questions, the "rating" column contains nulls values labeled with the text "Not given", for cases in which the customer did not provide any rating. Therefore, despite prior

analysis, there are null values in "rating" column. We will treat the nulls in the below code.

In [7]:
```
# See unique options displayed in rating column
df['rating'].unique()
```

Out[7]:
```
array(['Not given', '5', '3', '4'], dtype=object)
```

In [8]:
```
# Replace value "Not given" for a 0 value and convert the column into integer data type
df['rating'] = df['rating'].replace('Not given', 0).astype(int)
```

In [9]:
```
# Confirm the change
df['rating'].unique()
```

Out[9]:
```
array([0, 5, 3, 4])
```

In [10]:
```
# Confirm the datatype change
df.dtypes
```

Out[10]:
```
order_id                  int64
customer_id               int64
restaurant_name          object
cuisine_type             object
cost_of_the_order       float64
day_of_the_week          object
rating                    int32
food_preparation_time     int64
delivery_time             int64
dtype: object
```

Note:

- We successfully treated the null values in the "rating" column by transforming the text "Not given" to value 0.
- It is important to mention that if we intend to perform any mathematical calculations on the "rating" column, we should exclude these values, as they could skew the results.

In [11]:
```
# Statistical summary
df.describe().T
```

Out[11]:

	count	mean	std	min	25%	50%	75%	max
order_id	1898.00	1477495.50	548.05	1476547.00	1477021.25	1477495.50	1477969.75	1478444.00
customer_id	1898.00	171168.48	113698.14	1311.00	77787.75	128600.00	270525.00	405334.00
cost_of_the_order	1898.00	16.50	7.48	4.47	12.08	14.14	22.30	35.41
rating	1898.00	2.66	2.20	0.00	0.00	4.00	5.00	5.00
food_preparation_time	1898.00	27.37	4.63	20.00	23.00	27.00	31.00	35.00
delivery_time	1898.00	24.16	4.97	15.00	20.00	25.00	28.00	33.00

Notes:

- The minimum time that it takes to prepare a food, once the order is placed, is 20 minutes and the maximum time that has taken is 35 minutes.

- The average time that it takes to prepare a food is 27 minutes, and approximately half of the orders are prepared in 27 minutes or less.

In [12]:
```
# Previously the "Not rated" ratings were transformed to 0 value.
# Therefore, filtering the "rating" column by 0 will give us the answer

no_rated_orders = df[df['rating']==0]
no_rated_orders_count = no_rated_orders.shape[0]

total_orders = df['rating'].shape[0]

no_rated_orders_perc = round((no_rated_orders_count / total_orders) * 100,1)

print(f'There are {no_rated_orders_count} no rated orders from {total_orders} \
orders in total, accounting for {no_rated_orders_perc}% of the total orders')
```

```
There are 736 no rated orders from 1898 orders in total, accounting for 38.8% of the total order
s
```

Notes:

- As we can see, 38.8% of the total orders did not recieve a rating, equivalent to 736 orders.
- For any analysis based on the "rating" field, it is advisable to change the marketing strategy to enforce ratings. One idea is to make this field obligatory or incentivize customers with points for their next order. This approach would help in better understanding customer behavior and preferences based on ratings.

Exploratory Data Analysis (EDA)

Univariate Analysis

In [13]:
```
# Confirming each order is displayed in just one row
df['order_id'].nunique()
```

Out[13]:
```
1898
```

In [14]:
```
#  Getting number of unique customers
df['customer_id'].nunique()
```

Out[14]:
```
1200
```

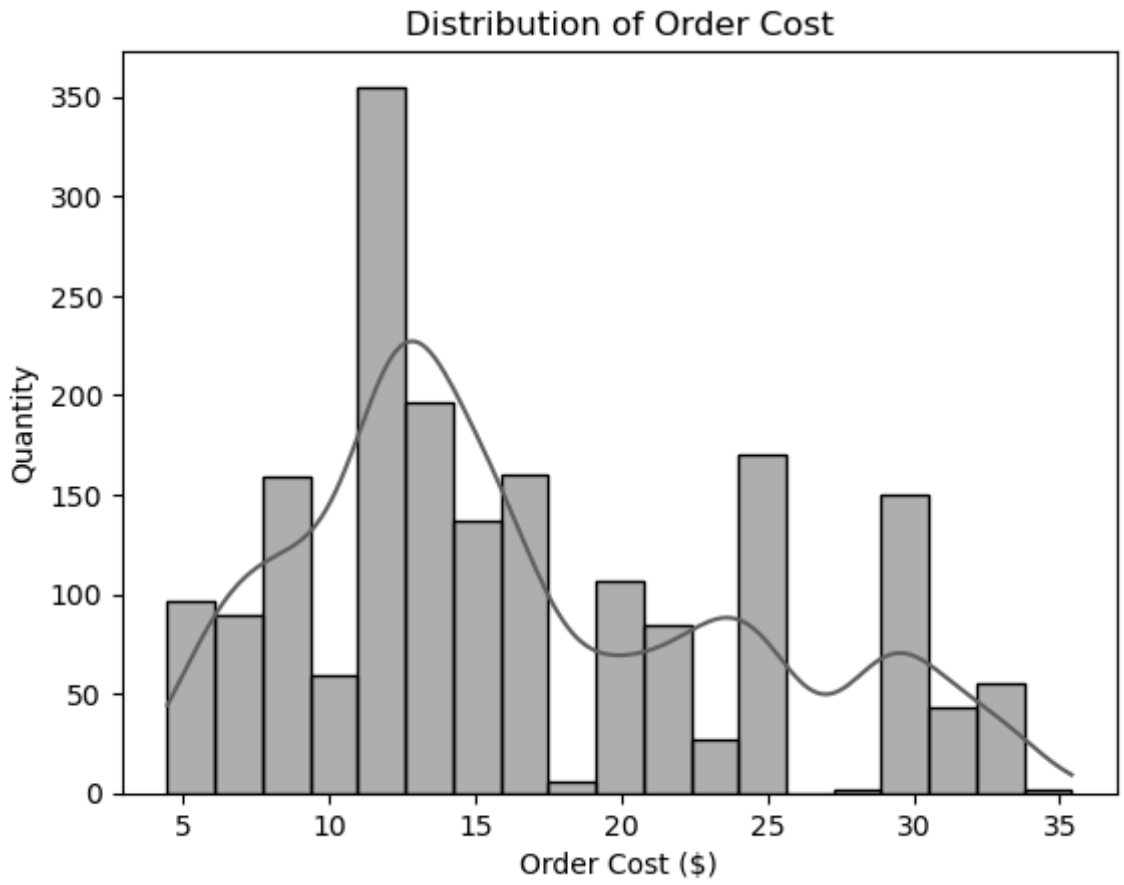
In [15]:
```
# Number of orders per restaurant
orders_per_restaurant = df['restaurant_name'].value_counts()
orders_per_restaurant.head(20)
```

Out[15]:
```
restaurant_name
Shake Shack                      219
The Meatball Shop                132
Blue Ribbon Sushi                119
Blue Ribbon Fried Chicken         96
Parm                              68
RedFarm Broadway                  59
RedFarm Hudson                    55
TAO                               49
Han Dynasty                       46
Blue Ribbon Sushi Bar & Grill     44
Nobu Next Door                    42
Rubirosa                          37
Sushi of Gari 46                  37
Momoya                            30
Five Guys Burgers and Fries       29
Blue Ribbon Sushi Izakaya         29
Bareburger                        27
Tamarind TriBeCa                  27
Jack's Wife Freda                 25
Sushi of Gari Tribeca             24
Name: count, dtype: int64
```

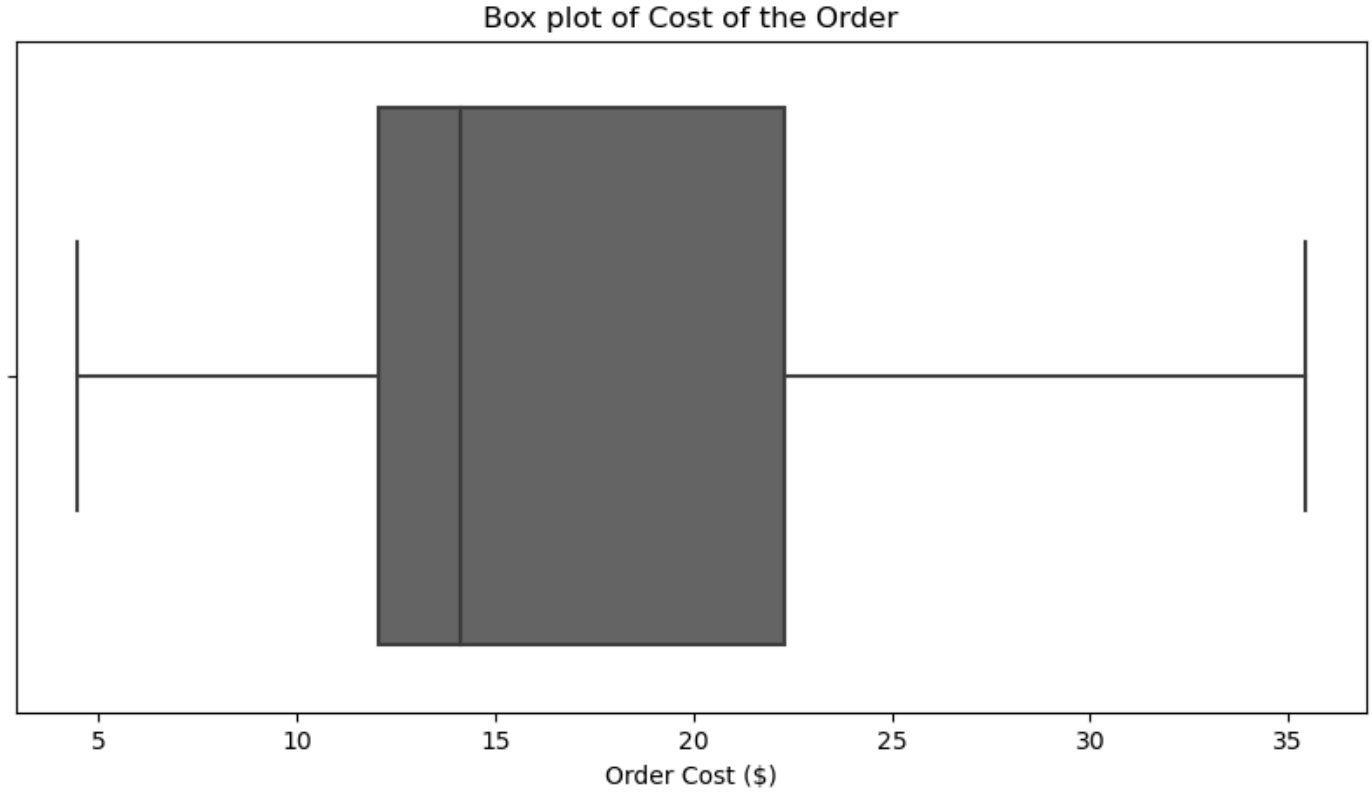
In [16]:
```
# Order cost
order_cost = df['cost_of_the_order'].describe()
order_cost
```

Out[16]:
```
count   1898.00
mean      16.50
std        7.48
min        4.47
25%       12.08
50%       14.14
75%       22.30
max       35.41
Name: cost_of_the_order, dtype: float64
```

In [17]:
```
# Order cost histogram
sns.histplot(data=df, x='cost_of_the_order', kde=True)
plt.title('Distribution of Order Cost')
plt.xlabel('Order Cost ($)')
plt.ylabel('Quantity')

plt.savefig('Images/Distribution of Order Cost.png', bbox_inches='tight')


plt.show()
```



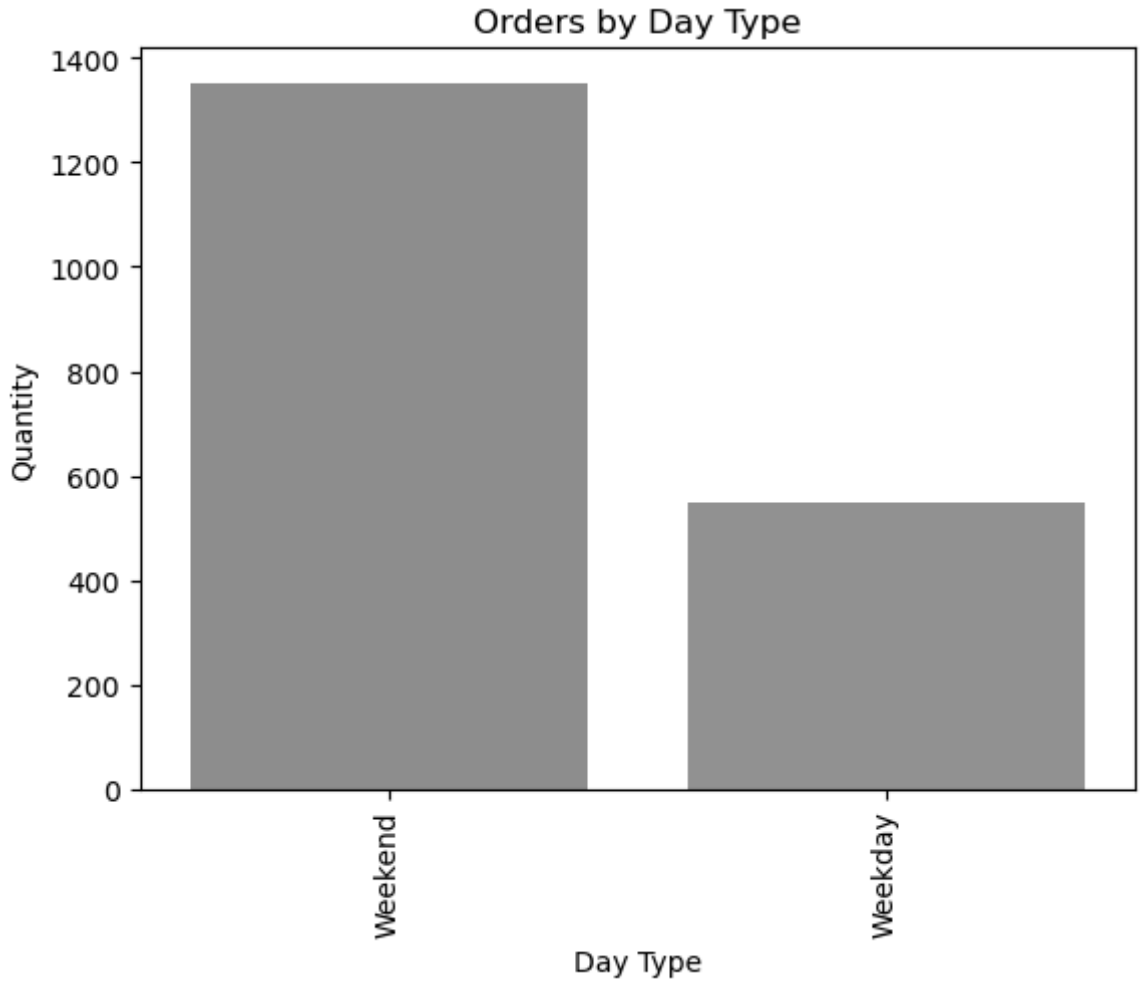
In [18]:
```
# Boxplot for order cost
plt.figure(figsize=(10,5))
sns.boxplot(data=df, x='cost_of_the_order')
plt.title('Box plot of Cost of the Order')
plt.xlabel('Order Cost ($)')
plt.savefig('Images/Box plot of Cost of the Order.png', bbox_inches='tight')
plt.show()
```



Note:

- 75% of the orders costs $22.30 dollars or less
- Half of the orders costs $14.14 dollars or less
- The maximum an order costs is $35 dollars

In [19]:
```
# Custom colors for weekend or weekday, considering colorblinded
colorblind_palette = sns.color_palette("colorblind", 8)

custom_color_day_of_week = {
    'Weekday' : colorblind_palette[7],
    'Weekend' : colorblind_palette[1]
}



#  Countplot for day of the week
sns.countplot(data=df, x='day_of_the_week', palette=custom_color_day_of_week)
plt.xticks(rotation=90)
plt.title('Orders by Day Type')
plt.xlabel('Day Type')
plt.ylabel('Quantity')

plt.savefig('Images/Orders by Day Type.png', bbox_inches='tight')
plt.show()
```



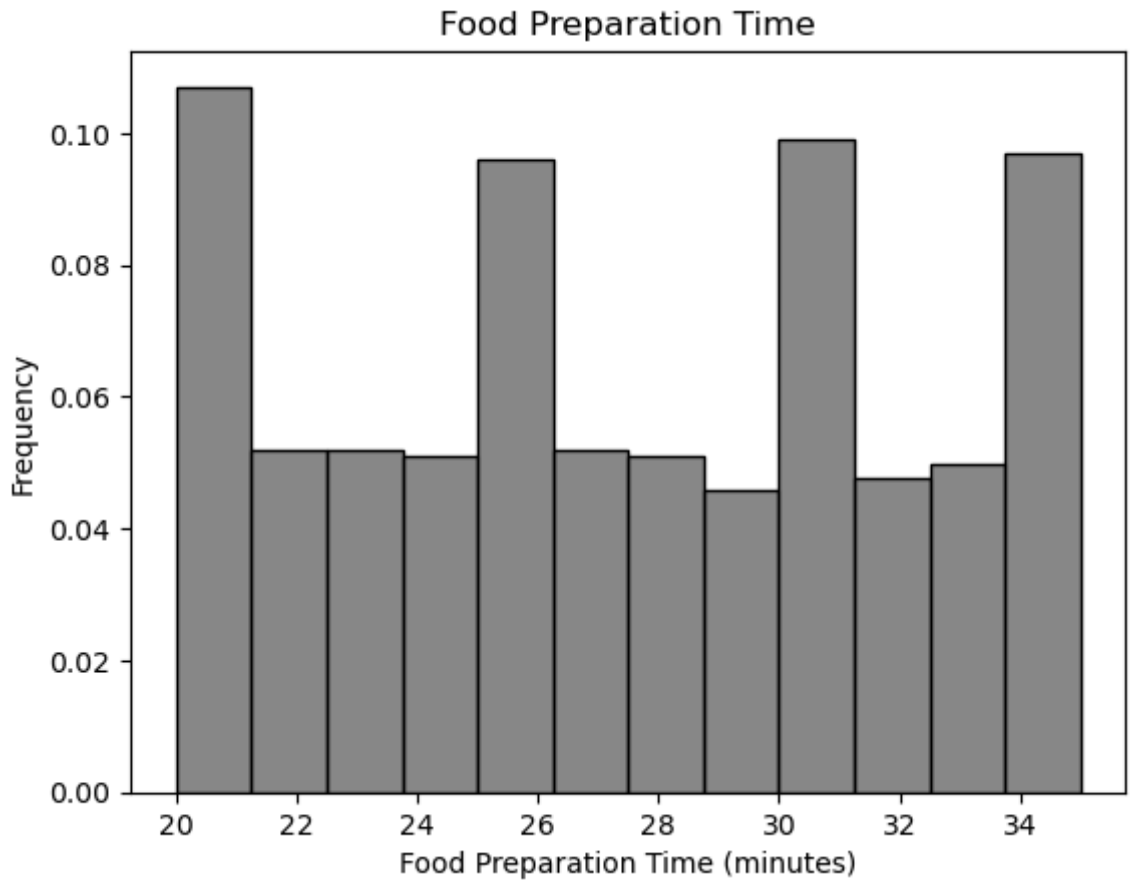
Note:

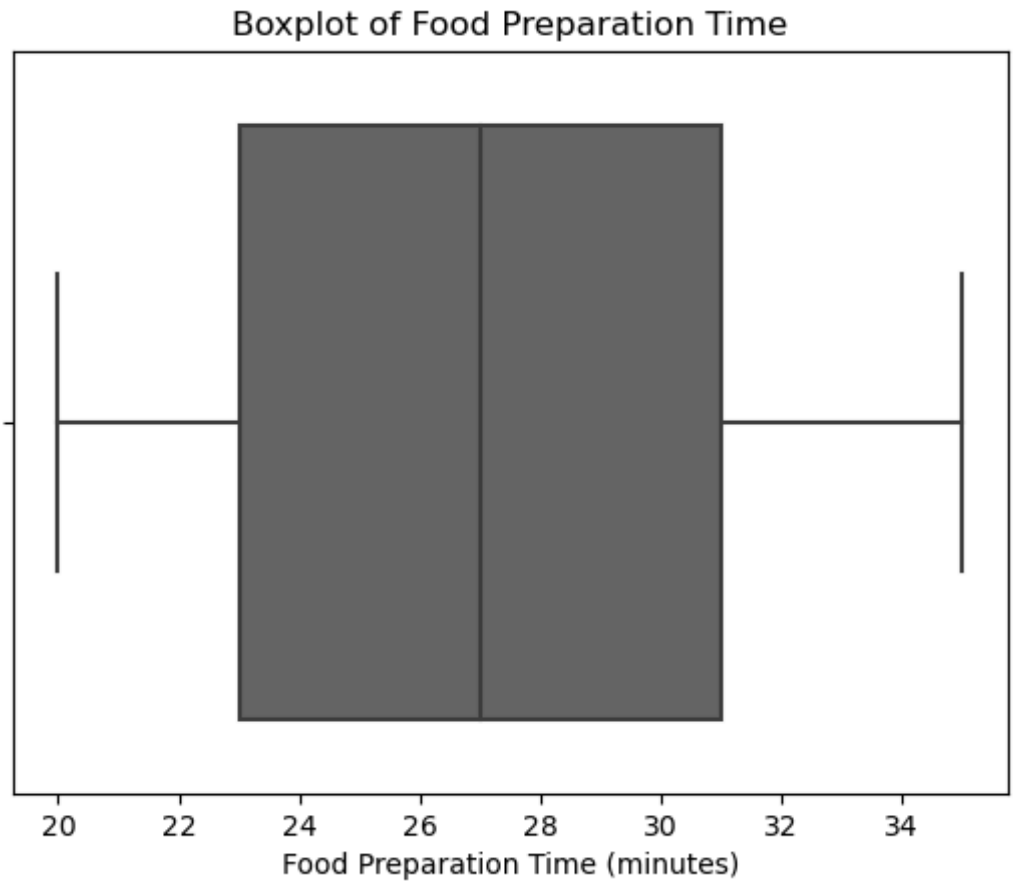
- Majority of orders come from the weekend

In [20]:
```
# Analyzing food preparation time
df['food_preparation_time'].describe()
```

Out[20]:
```
count   1898.00
mean      27.37
std        4.63
min       20.00
25%       23.00
50%       27.00
75%       31.00
max       35.00
Name: food_preparation_time, dtype: float64
```

In [21]:
```
# Histogram of food preparation time
sns.histplot(data=df, x='food_preparation_time', stat='density')
plt.title('Food Preparation Time')
plt.xlabel('Food Preparation Time (minutes)')
plt.ylabel('Frequency')
plt.show()

# Boxplot of food preparation time
sns.boxplot(data=df, x='food_preparation_time')
plt.title('Boxplot of Food Preparation Time')
plt.xlabel('Food Preparation Time (minutes)')
plt.savefig('Images/Boxplot of Food Preparation Time.png', bbox_inches='tight')
plt.show()
```



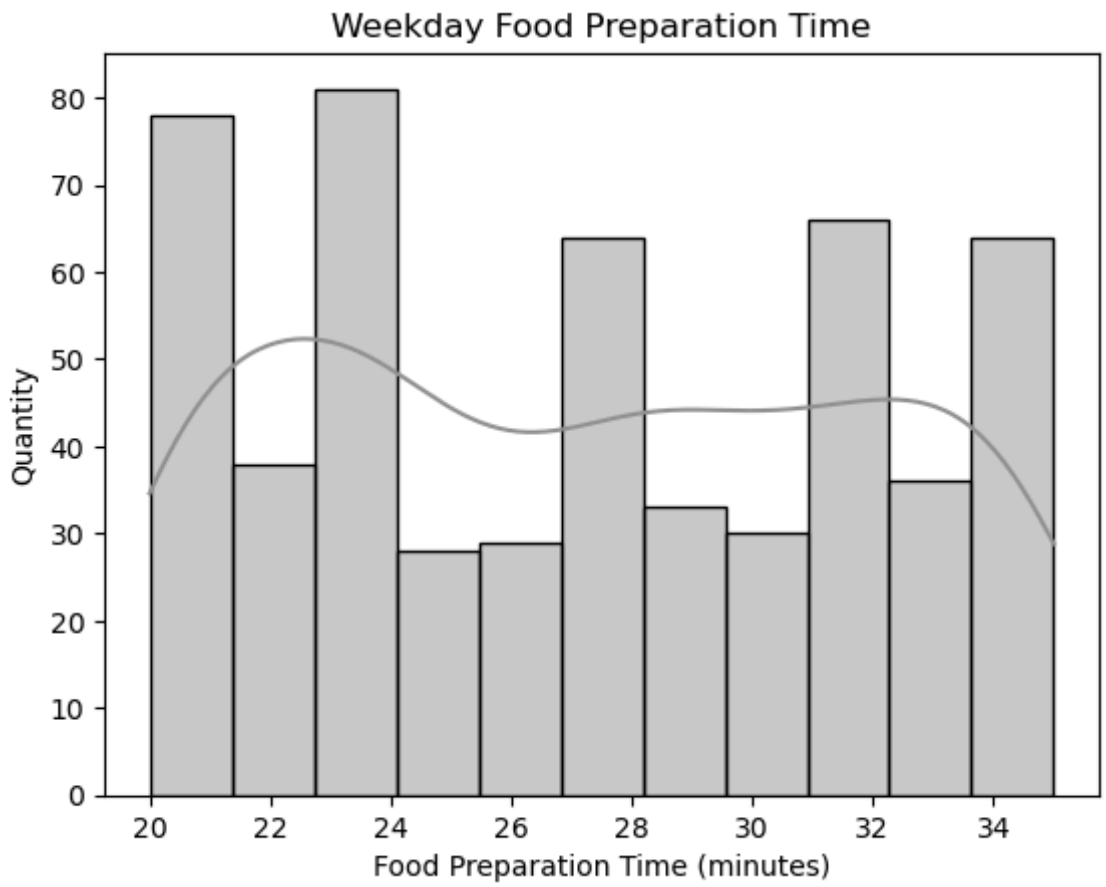


Notes:

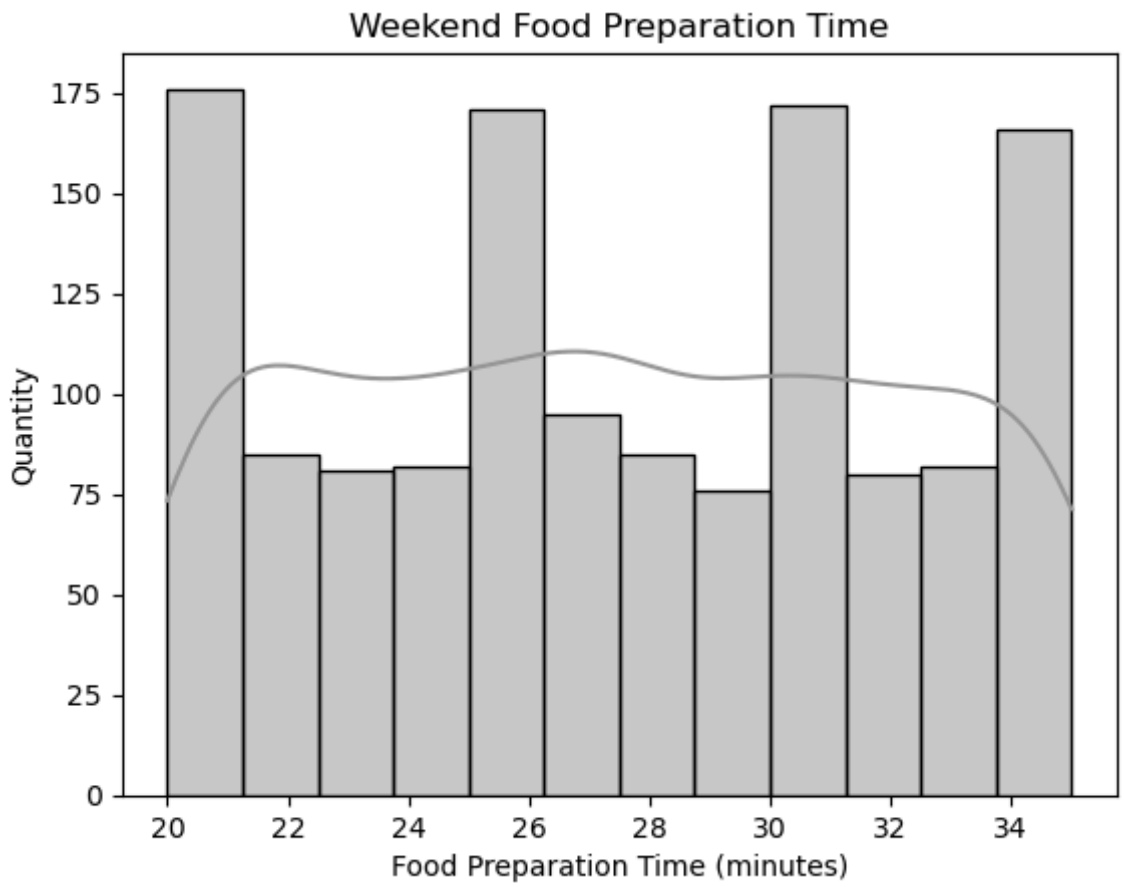
- Histogram displays multimodal distributio when looking at food preparation time.
- There is higher density in some intervals like 20-, 26-, 30- and 34-minute marks of preparation time, which can be due to having more orders to prepare in certain times of the day that makes the preparation to spike in these intervals.
- For the smaller density around 22, 24, 28 and 32 minutes; this could suggest is less frequent to have these preparation time. Would be beneficial to have in the dataset the time of the day the orders are places to combine with this analysis in trying to understand better customer behavior and restaurant performance.
- It would be also beneficial to understand what specific day of the week the orders are placed.

In [22]:
```
# Filtering dataset on day type (weekday or weekend) to explore the data further
weekday_preparation_time = df[df['day_of_the_week']=='Weekday']
```

In [23]:
```
# Histogram of weekday food prepation time
sns.histplot(data=weekday_preparation_time['food_preparation_time'], kde=True,  color=colorblind
plt.title('Weekday Food Preparation Time')
plt.xlabel('Food Preparation Time (minutes)')
plt.ylabel('Quantity')

plt.savefig('Images/Weekday Food Preparation Time.png', bbox_inches='tight')

plt.show()
```



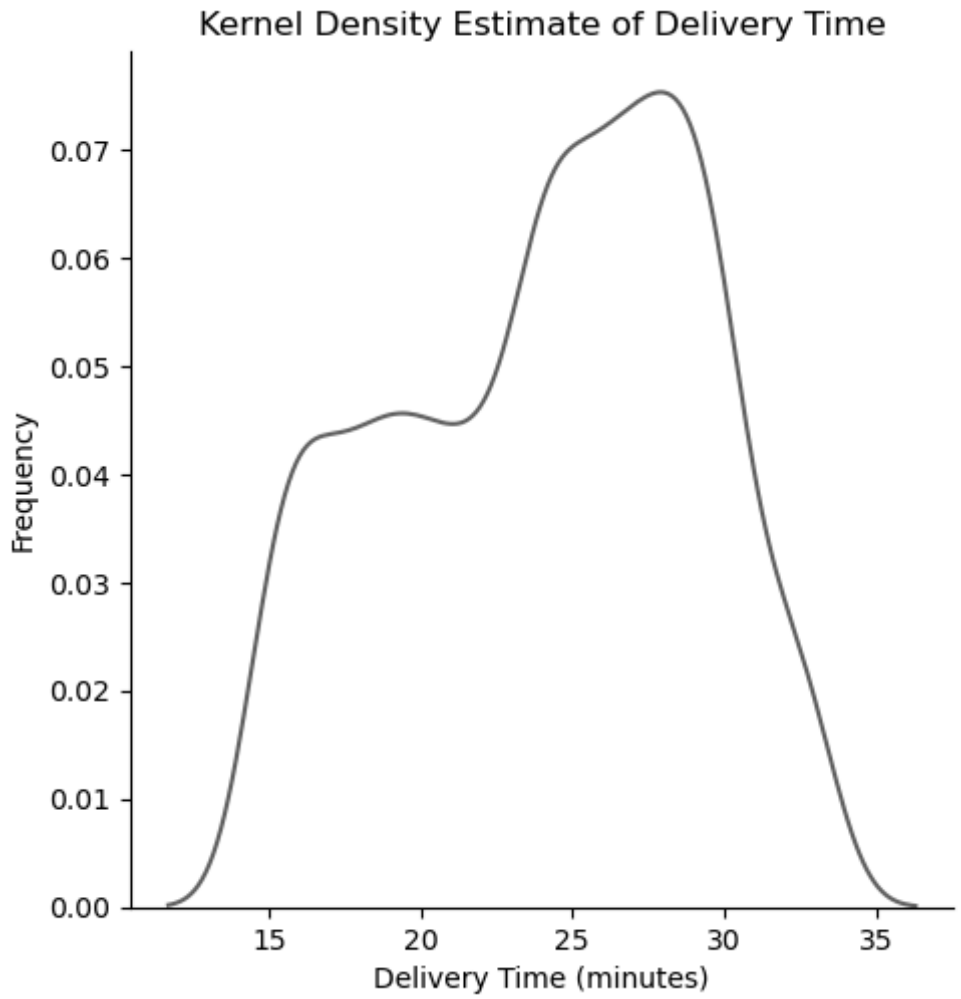
In [24]:
```
# Histogram of weekend food prepation time
weekend_preparation_time = df[df['day_of_the_week']=='Weekend']

sns.histplot(data=weekend_preparation_time['food_preparation_time'], kde=True, color=colorblind_
plt.title('Weekend Food Preparation Time')
plt.xlabel('Food Preparation Time (minutes)')
plt.ylabel('Quantity')


plt.savefig('Images/Weekend Food Preparation Time.png', bbox_inches='tight')
plt.show()
```



Notes:

- Food preparation time does not present considerable difference in terms of distribution between weekday or weekend.
- Main difference between weekday or weekend we can see in these graphs, is that despite on weekends there are more orders, the food preparation time remains similar, ranging from 20 to 43 minutes.

In [25]:
```
# Distribution plot for delivery time (Kernel Density Estimate)
sns.displot(data=df, x='delivery_time', kind='kde')
plt.title('Kernel Density Estimate of Delivery Time')
plt.xlabel('Delivery Time (minutes)')
plt.ylabel('Frequency')


plt.savefig('Images/Kernel Density Estimate of Delivery Time.png', bbox_inches='tight')
plt.show()

# Delivery time boxplot
sns.boxplot(data=df, x='delivery_time')
plt.title('Boxplot of Deilvery Time')
plt.xlabel('Delivery Time (minutes)')
plt.show()
```



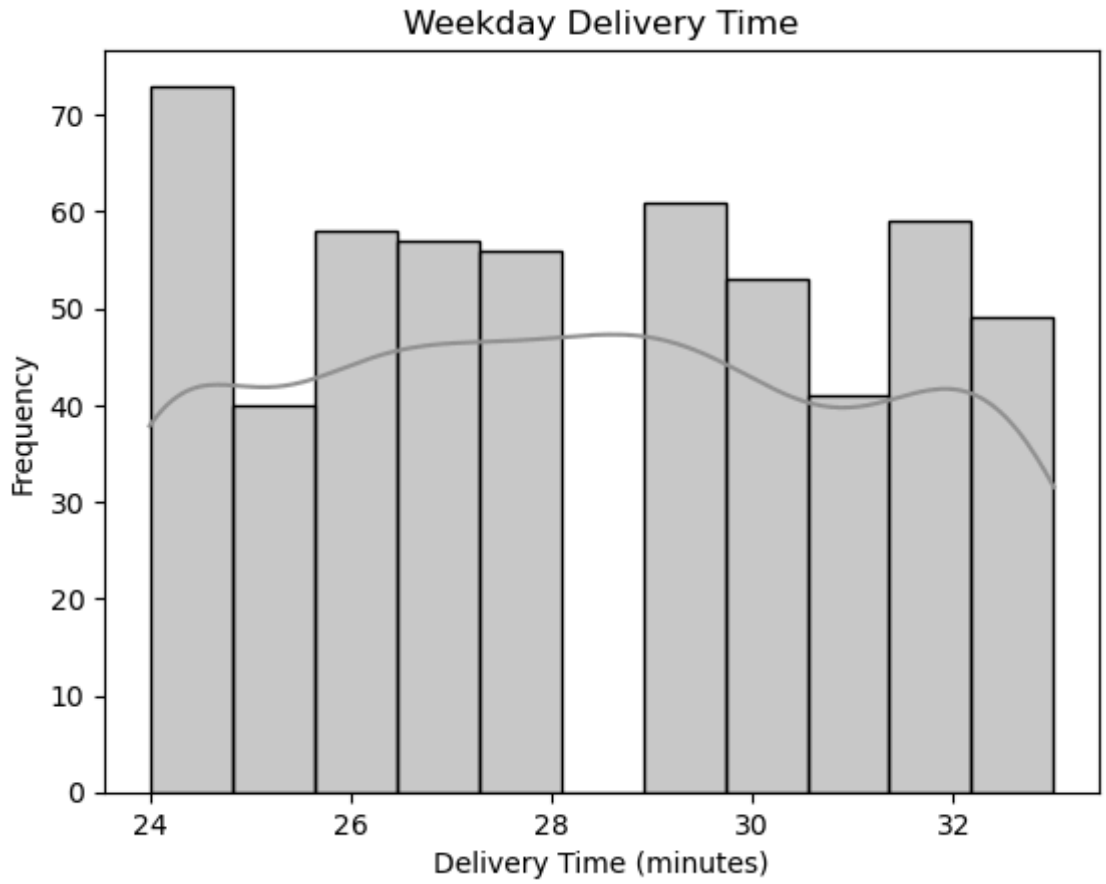
kde: kernel density estimate that smooths the data for us

Notes:

- Majority of deliveries are within 25-30 minutes range
- There is a spread in delivery time, with some orders being delivered in 15 minutes and others taking up to 35 minutes, although fewer.
- The delivery variability is likely affected by other factors like customer location, restaurant distance, other operational conditions like amount of orders received by the restaurant, for example.
- It would be beneficial to add to the dataset the restaurant and customers address to better understand these variations

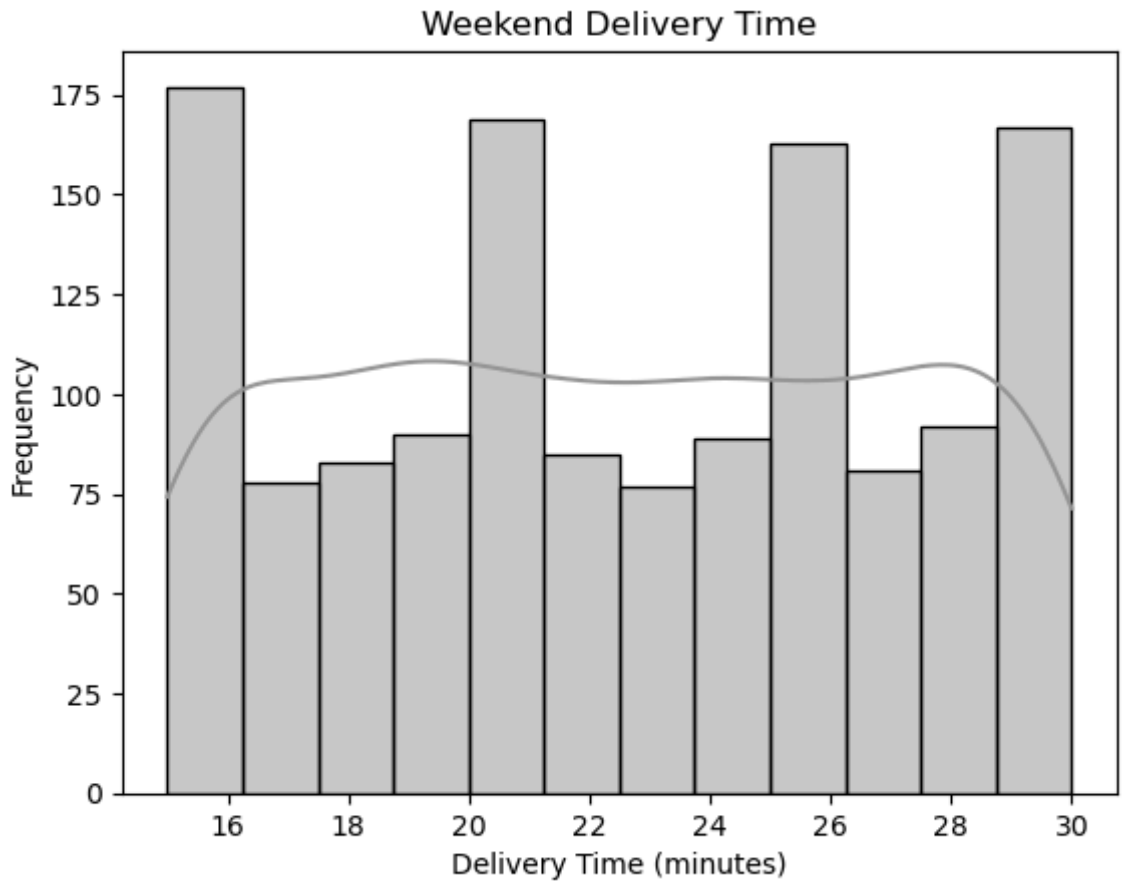
In [26]:
```
# Histogram with kernel distribution for weekday delivery time
sns.histplot(data=weekday_preparation_time['delivery_time'], kde=True, color=colorblind_palette[
plt.title('Weekday Delivery Time')
plt.xlabel('Delivery Time (minutes)')
plt.ylabel('Frequency')
```

Out[26]:
```
Text(0, 0.5, 'Frequency')
```

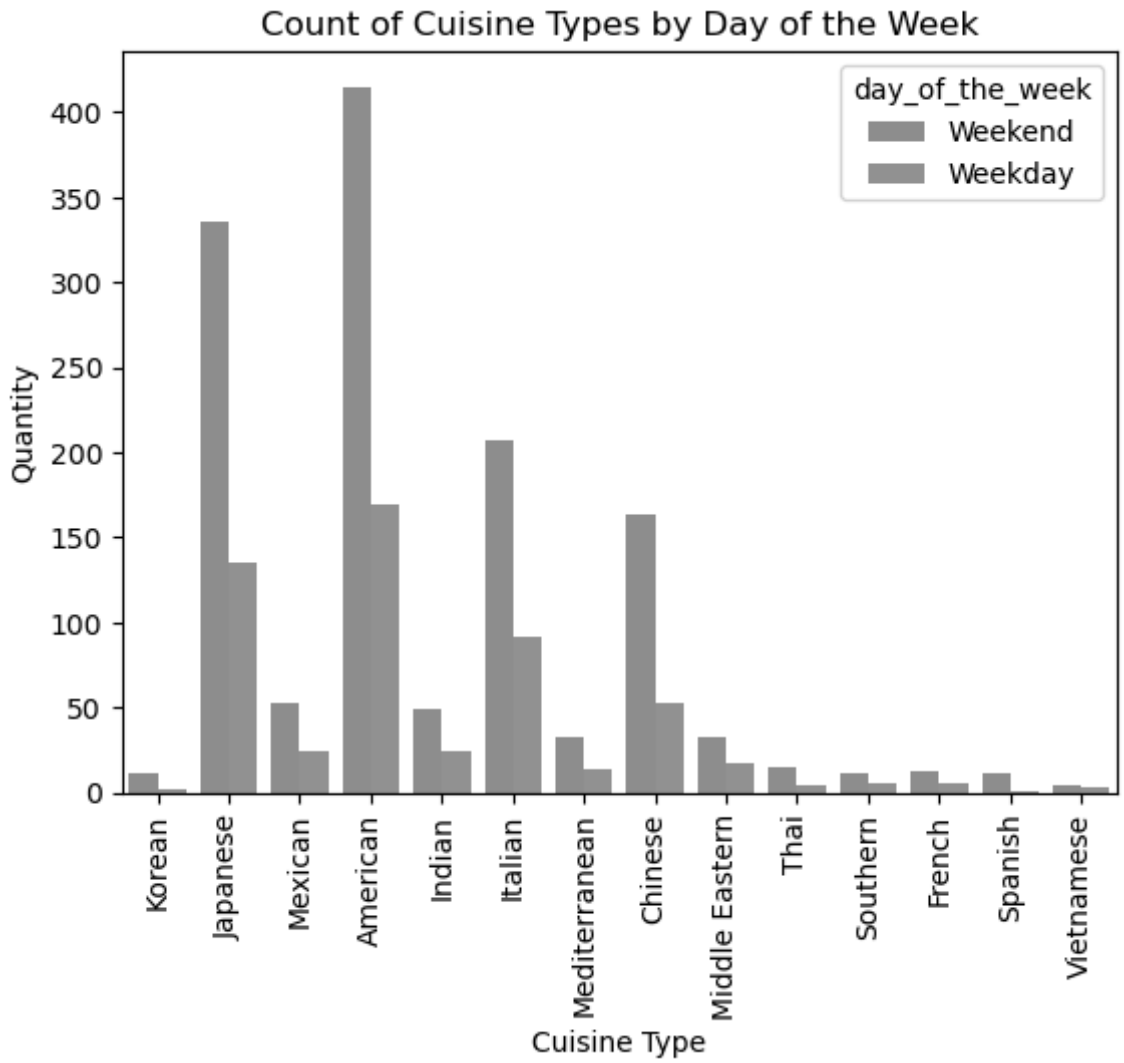


In [27]:
```
# Histogram with kernel distribution for weekendy delivery time
sns.histplot(data=weekend_preparation_time['delivery_time'], kde=True, color=colorblind_palette[
plt.title('Weekend Delivery Time')
plt.xlabel('Delivery Time (minutes)')
plt.ylabel('Frequency')
```

Out[27]:
```
Text(0, 0.5, 'Frequency')
```



In [28]:
```
# Count of Cuisine Types by Day of the Week
sns.countplot(data=df, x='cuisine_type', hue=df['day_of_the_week'], palette=custom_color_day_of_
plt.xticks(rotation=90)
plt.title('Count of Cuisine Types by Day of the Week')
plt.xlabel('Cuisine Type')
plt.ylabel('Quantity')

#Save the plot to a folder called Images. Added bbox_inches to be able to see the full plot in t
plt.savefig('Images/Count of Cuisine Types by Day of the Week.png', bbox_inches='tight')

plt.show()
```



Note:

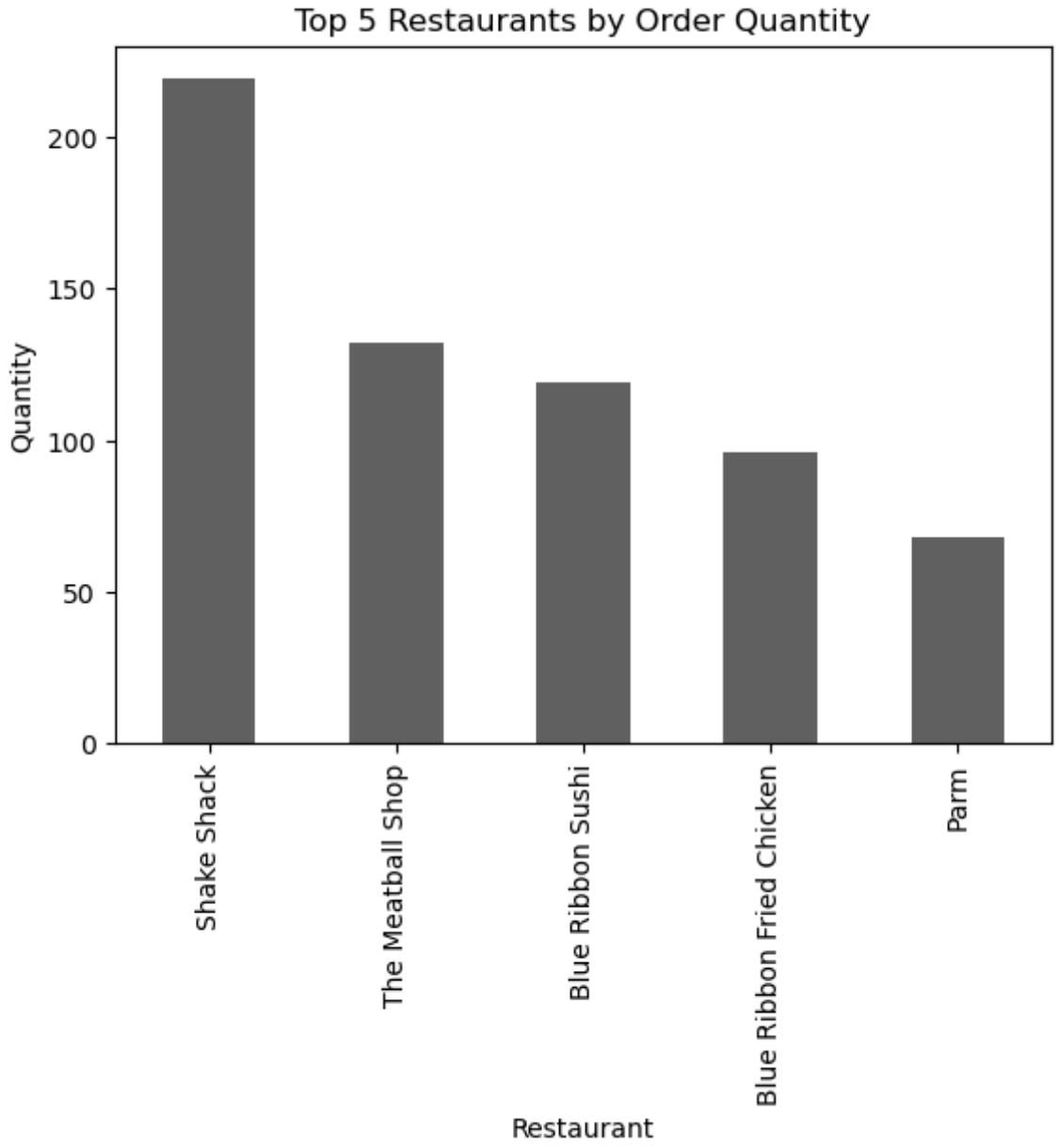
- American Cuisine is the most ordered during week and weekends
- Following American cuisine, Japanese, Italian and Chinese follows.
- The least ordered food is Vietnamese.

In [29]:
```
# Top restaurant by order count
orders_per_restaurant.nlargest(100)
```

Out[29]:
```
restaurant_name
Shake Shack                  219
The Meatball Shop            132
Blue Ribbon Sushi            119
Blue Ribbon Fried Chicken     96
Parm                          68
                            ...
Izakaya Ten                    3
Carmine's                      3
The Odeon                      3
Tres Carnes                    3
Amy Ruth's                     3
Name: count, Length: 100, dtype: int64
```

In [30]:
```
# Getting top 5 restaurants by order quantity
top_restaurants = orders_per_restaurant.nlargest(5)

top_restaurants.plot(kind='bar')
```

```
plt.xticks(rotation=90)
plt.title('Top 5 Restaurants by Order Quantity')
plt.xlabel('Restaurant')
plt.ylabel('Quantity')

#Save the plot to a folder called Images. Added bbox_inches to be able to see the full plot in t
plt.savefig('Images/Top 5 Restaurants by Order Quantity.png', bbox_inches='tight')

plt.show()
```



In [31]:
```
# Count the number of orders per restaurant
orders_per_restaurant = df['restaurant_name'].value_counts().reset_index()
orders_per_restaurant.columns = ['restaurant_name', 'order_count']

unique_restaurant_cuisine = df[['restaurant_name', 'cuisine_type']].drop_duplicates()

top_restaurants_cuisine = orders_per_restaurant.merge(unique_restaurant_cuisine, on='restaurant_

top_5_restaurants_cuisine = top_restaurants_cuisine.nlargest(5, 'order_count')

print(top_5_restaurants_cuisine)
```

```
             restaurant_name  order_count cuisine_type
0                Shake Shack          219     American
1          The Meatball Shop          132      Italian
2          The Meatball Shop          132     American
3          Blue Ribbon Sushi          119     Japanese
4  Blue Ribbon Fried Chicken           96     American
```

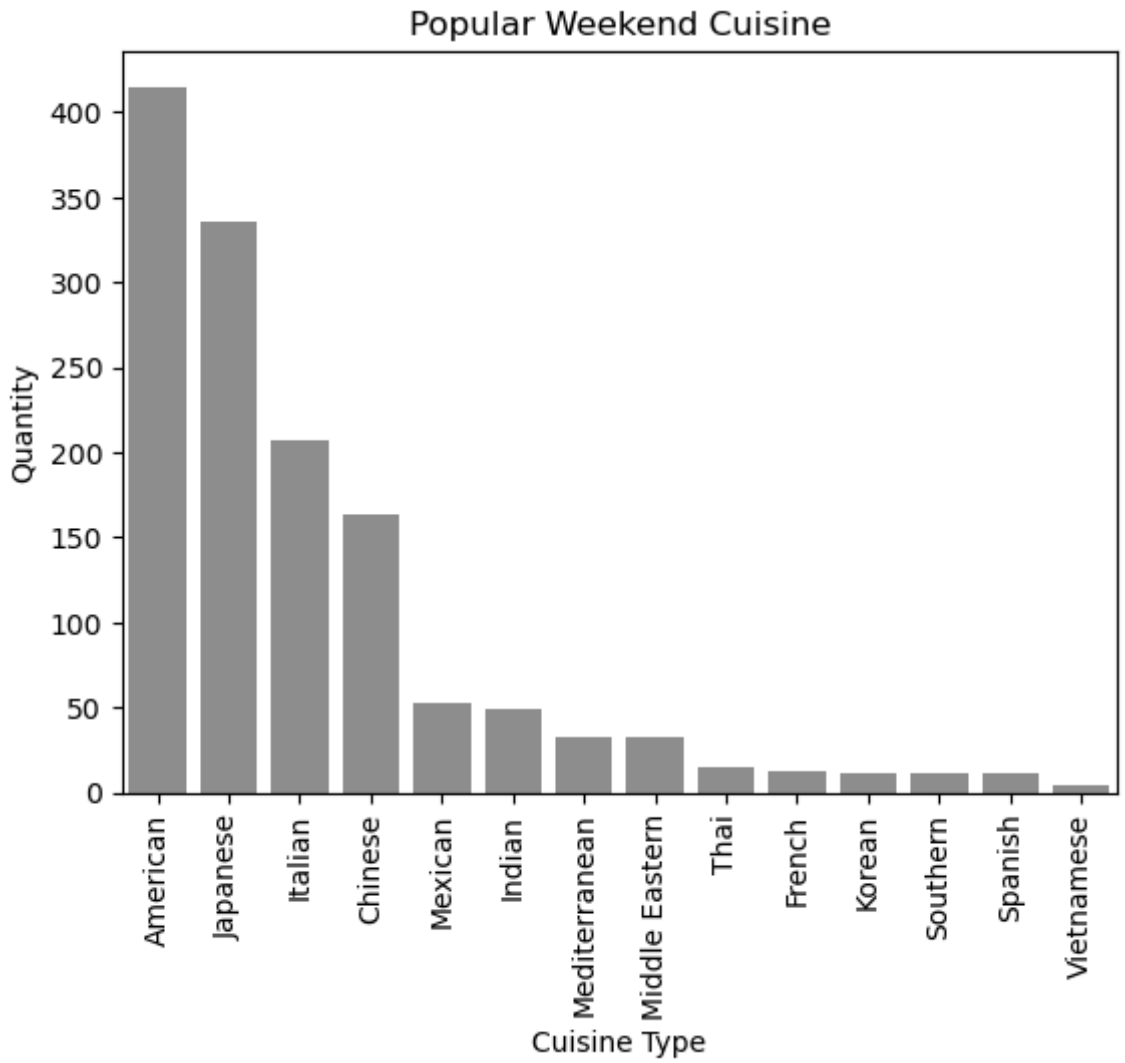
Note:

- From the top 5 restaurants by order count, three of them are American cuisine, one Italian and one is Japanese.

In [32]:
```
# Filtering by weekend orders
weekend = df.loc[df['day_of_the_week']=='Weekend']

# counting orders by restaurant to ge tthe most popular cuisine during the weekend
weekend_cuisine = weekend['cuisine_type'].value_counts()
weekend_cuisine
```

Out[32]:
```
cuisine_type
American          415
Japanese          335
Italian           207
Chinese           163
Mexican            53
Indian             49
Mediterranean      32
Middle Eastern     32
Thai               15
French             13
Korean             11
Southern           11
Spanish            11
Vietnamese          4
Name: count, dtype: int64
```

In [33]:
```
# Getting top 5 restaurants by order quantity
sns.countplot(data = weekend, x = 'cuisine_type', order = weekend['cuisine_type'].value_counts()
plt.xticks(rotation=90)
plt.title('Popular Weekend Cuisine')
plt.xlabel('Cuisine Type')
plt.ylabel('Quantity')

#Save the plot to a folder called Images. Added bbox_inches to be able to see the full plot in t
plt.savefig('Images/Popular Weekend Cuisine.png', bbox_inches='tight')

plt.show()
```

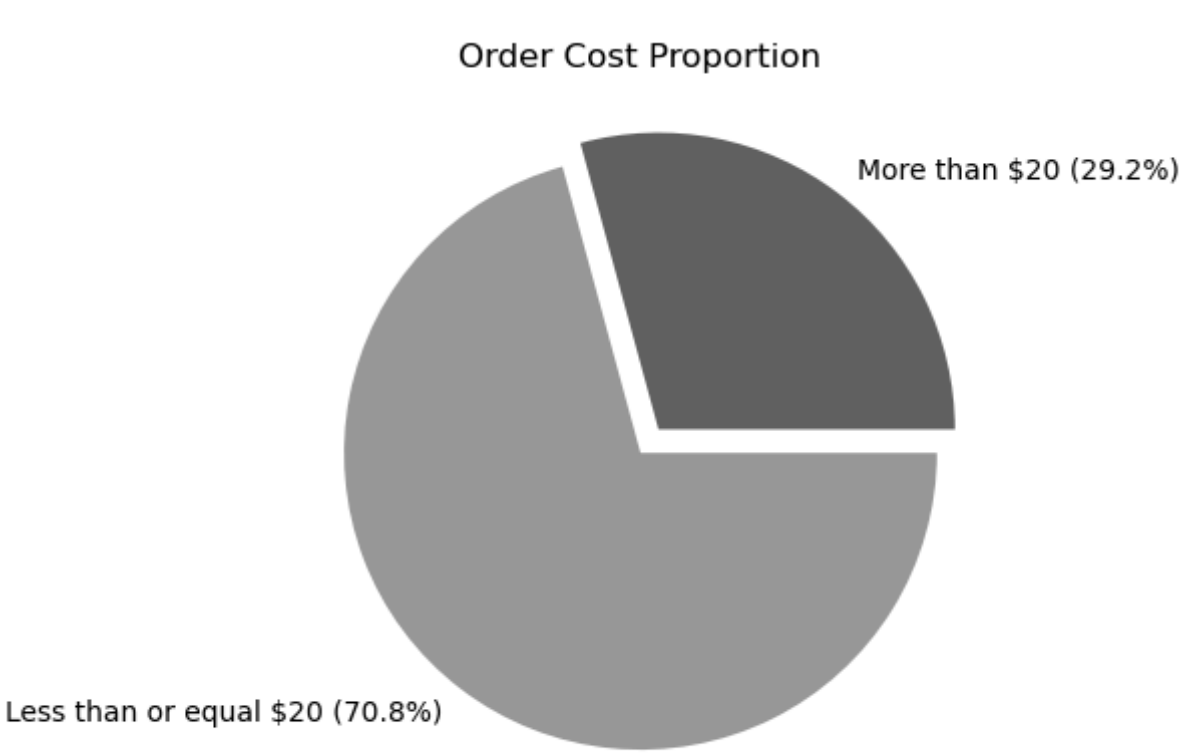


The most popular cuisine in weekends is the American food, with 415 orders in total.

In [34]:
```
# Write the code here
orders_cost_more_20_count = df.loc[df['cost_of_the_order']>20].shape[0]

orders_cost_less_equal_20 = df.loc[df['cost_of_the_order']<=20].shape[0]

all_orders_count = df.shape[0]

orders_cost_more_20_perc = round((orders_cost_more_20_count /  all_orders_count)*100,1)

print(f'From the total of {all_orders_count} orders, {orders_cost_more_20_count} cost more than
```

```
From the total of 1898 orders, 555 cost more than $20, which represents 29.2%
```

In [35]:
```
sizes = [orders_cost_more_20_count, orders_cost_less_equal_20]
labels = [f'More than $20 ({orders_cost_more_20_perc}%)', f'Less than or equal $20 ({100 - order
explode = (0.1,0)
plt.title('Order Cost Proportion')
plt.pie(sizes, explode=explode, labels = labels)

#Save the plot to a folder called Images. Added bbox_inches to be able to see the full plot in t
plt.savefig('Images/Order Cost Proportion.png', bbox_inches='tight')
```



In [36]:
```
# Mean of delivery time
delivery_mean = round(df['delivery_time'].mean(),2)
print(f'The mean order delivery time is {delivery_mean} minutes')
```

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

In [37]:
```
# Mean of delivery time
df['delivery_time'].describe()
```

Out[37]:
```
count   1898.00
mean      24.16
std        4.97
min       15.00
25%       20.00
50%       25.00
75%       28.00
max       33.00
Name: delivery_time, dtype: float64
```

In [38]:
```
# Getting top 5 restaurants by order quantity
sns.boxplot(data = df, x = 'delivery_time')
# plt.xticks(rotation=90)
plt.title('Delivery Time')
plt.xlabel('Delivery Time (minutes)')

#Save the plot to a folder called Images. Added bbox_inches to be able to see the full plot in t
plt.savefig('Images/Delivery Time.png', bbox_inches='tight')
```

In [39]:
```
# Write the code here
customer_count = df['customer_id'].value_counts()
customer_count

top_3_customers = customer_count.nlargest(3)
top_3_customers

print(f'The top 3 customers that should receive the 20% voucher discount are: {top_3_customers}'
```

```
The top 3 customers that should receive the 20% voucher discount are: customer_id
52832    13
47440    10
83287     9
Name: count, dtype: int64
```

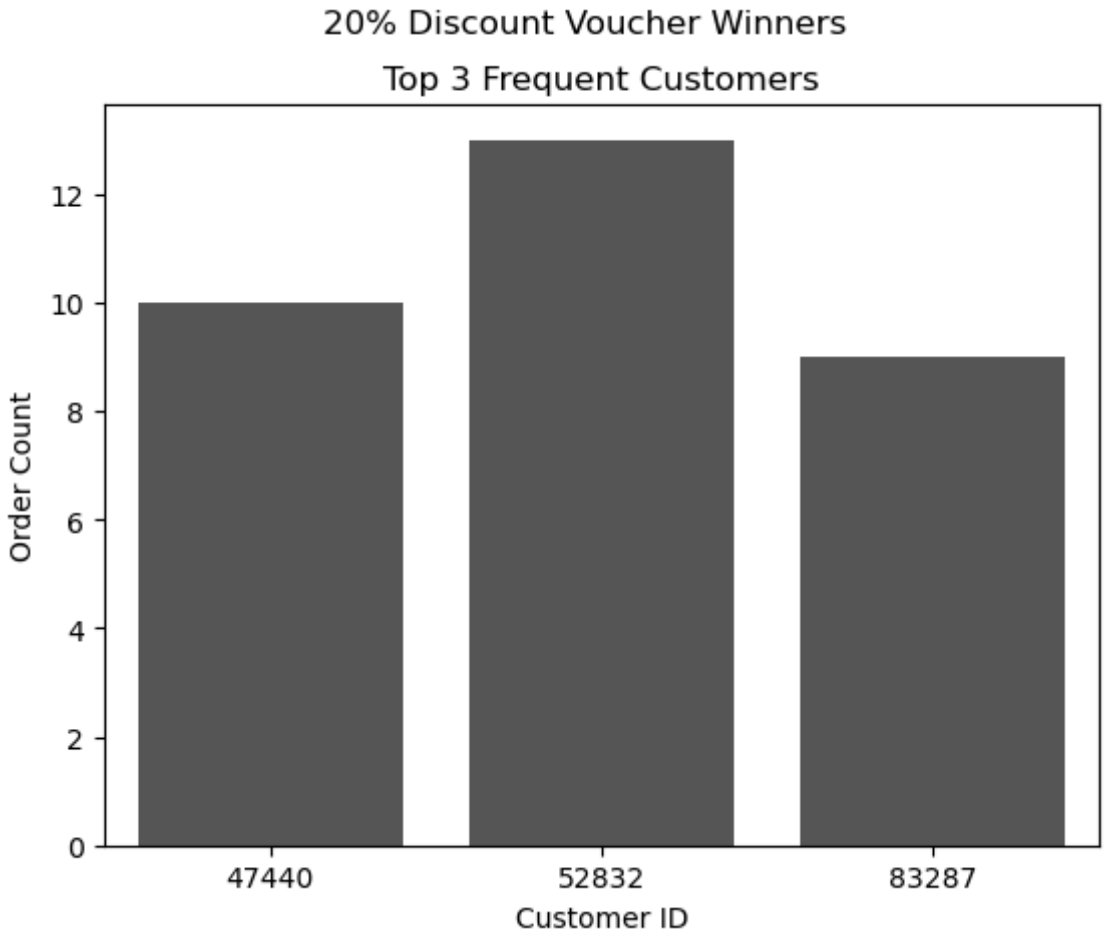
In [40]:
```
top_3_customers_df = top_3_customers.reset_index()
top_3_customers_df

top_3_customers_df.columns = ['customer_id', 'order_count']
top_3_customers_df = top_3_customers_df.sort_values(by='order_count', ascending=False)
top_3_customers_df
```

Out[40]:

	customer_id	order_count
0	52832	13
1	47440	10
2	83287	9

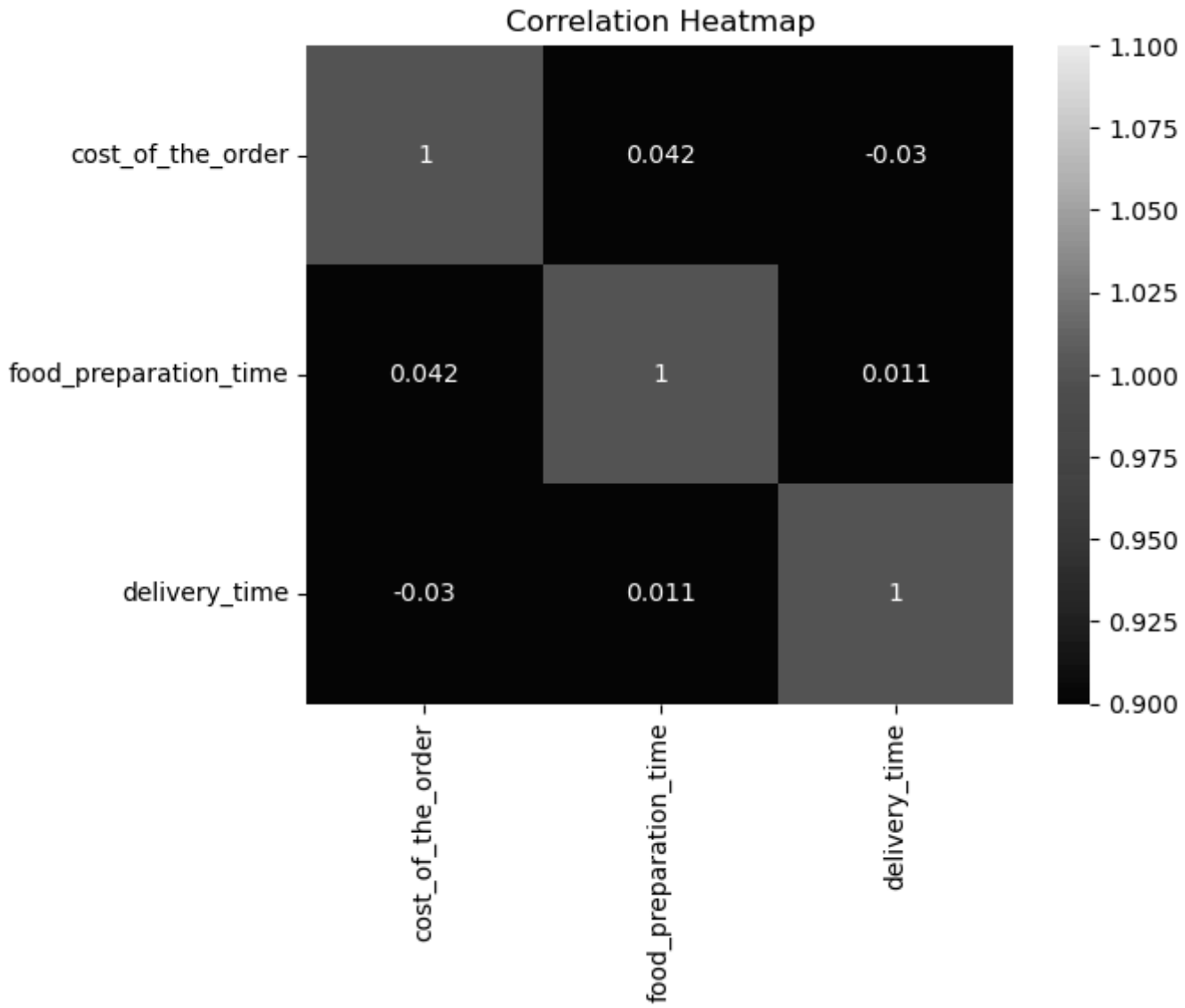
In [41]:
```
sns.barplot(x='customer_id', y= 'order_count', data = top_3_customers_df, color=colorblind_palet
plt.title('Top 3 Frequent Customers')
plt.suptitle('20% Discount Voucher Winners')
```

```
plt.xlabel('Customer ID')
plt.ylabel('Order Count')


#Save the plot to a folder called Images. Added bbox_inches to be able to see the full plot in t
plt.savefig('Images/20perc Discount Voucher Winners.png', bbox_inches='tight')
```



Multivariate Analysis

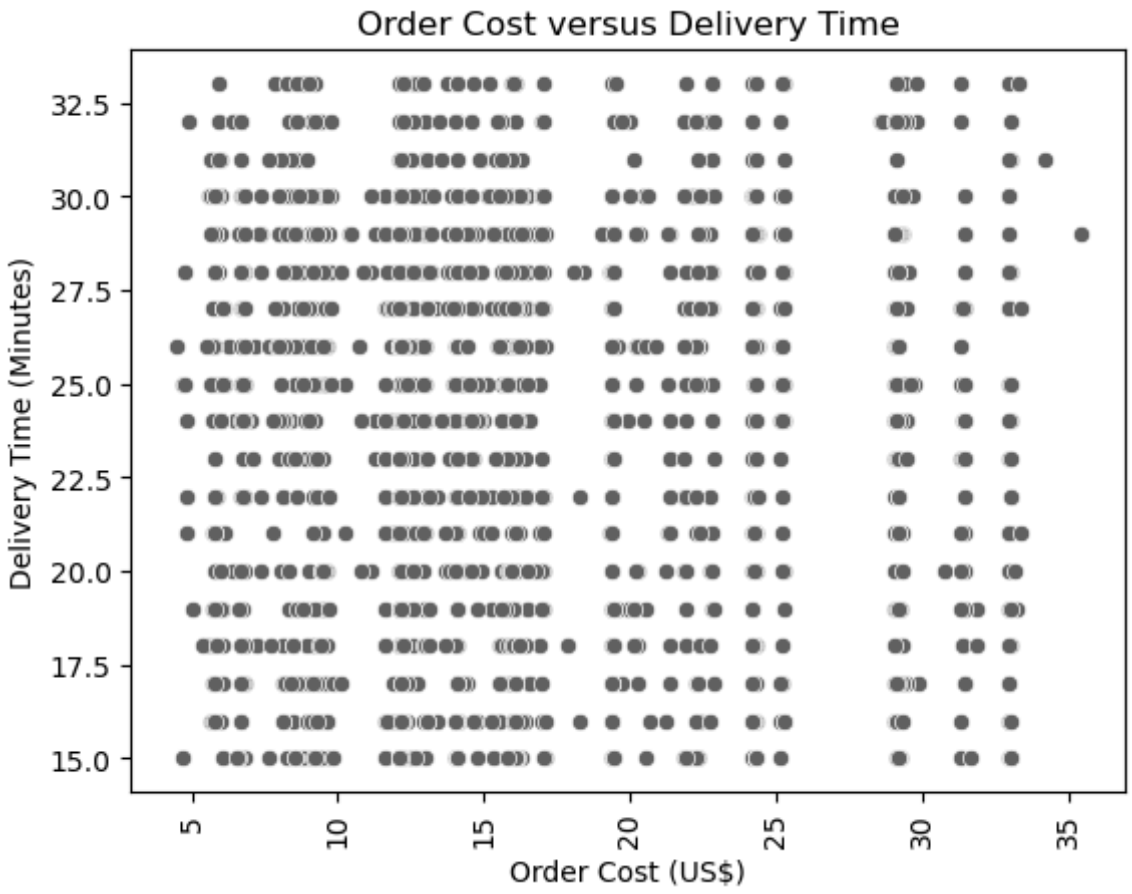
In [42]:
```
correlation_matrix = df[['cost_of_the_order', 'food_preparation_time', 'delivery_time']].corr()

sns.heatmap(correlation_matrix, annot=True, vmin=1, vmax=1 )
plt.title('Correlation Heatmap')
plt.savefig('Images/Correlation Heatmap.png', bbox_inches='tight')
plt.show()
```



Note:

- As we can see from the map heat, the correlation between order cost and food preparation time of 0.042 is very low, indicating that there is not relationship between them. So even is the order cost increases, the food preparation time does not change.
- Relationship between order cost and delivery time is -0.03, indicating that when the order cost is higher, the delivery time decreases but the amount of minutes is almost nothing, therefore, there is no relationship between these variables.
- Relationship between food preparation time and delivery time is 0.011, indicating that if the food preparation time increases, there is no change in the delivery time, therefore, there is no relationship between them.

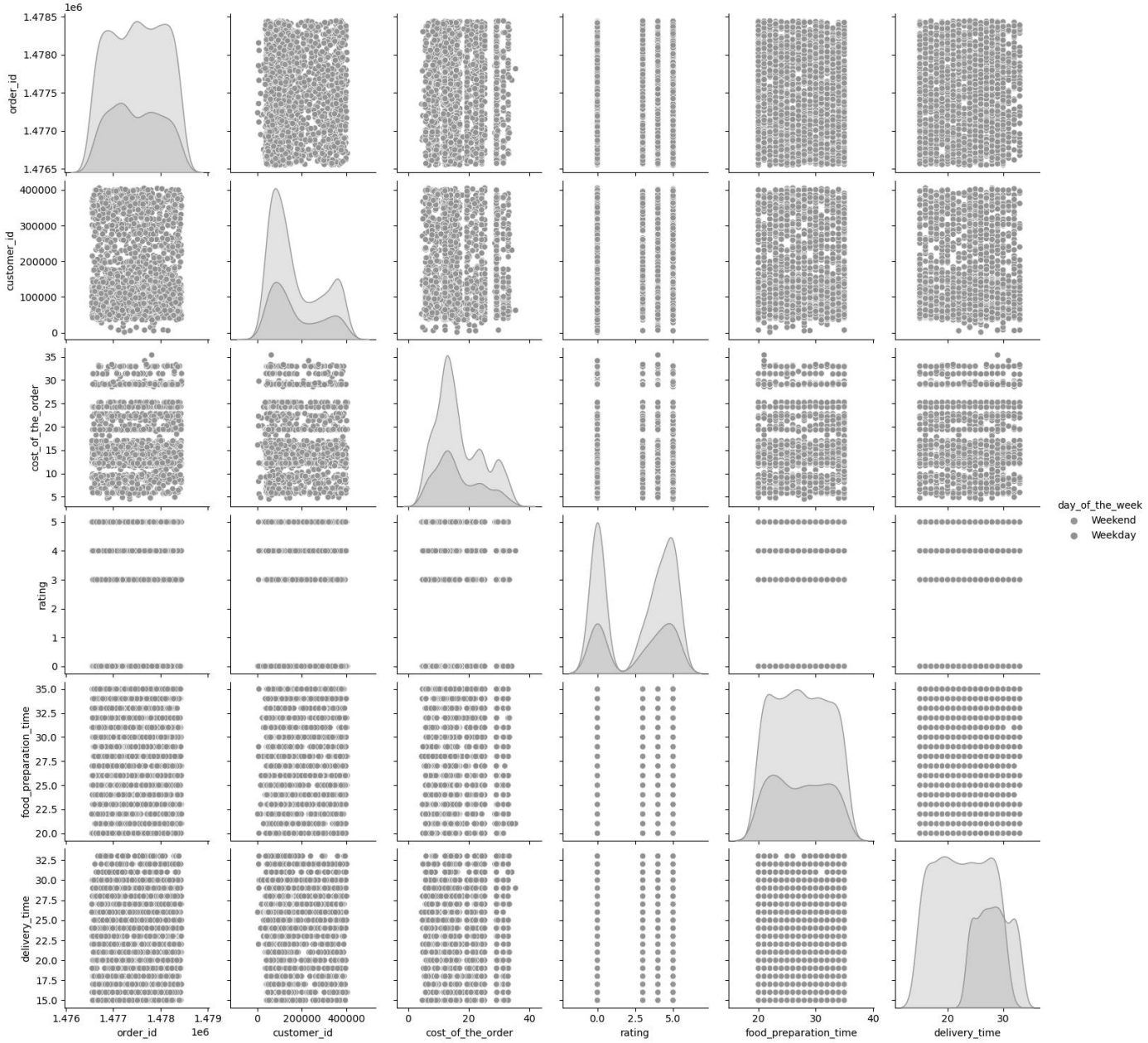
In [43]:
```
# Scatter plot to see relation between order cost and food preparation time
sns.scatterplot(data=df, x='cost_of_the_order', y='delivery_time')
plt.xticks(rotation=90)
plt.title('Order Cost versus Delivery Time')
plt.xlabel('Order Cost (US$)')
plt.ylabel('Delivery Time (Minutes)')

plt.savefig('Images/Order Cost versus Delivery Time.png', bbox_inches='tight')
plt.show()
```



Note:

- Confirming what we analyzed in the heat map, there is no clear trend between Order Cost and Delivery Time, regardless of the cost we can see delivery time vary widely.
- There are more deliveries for order between `$5` and `$16`. This is likely due to the number of orders within this cost range is higher.

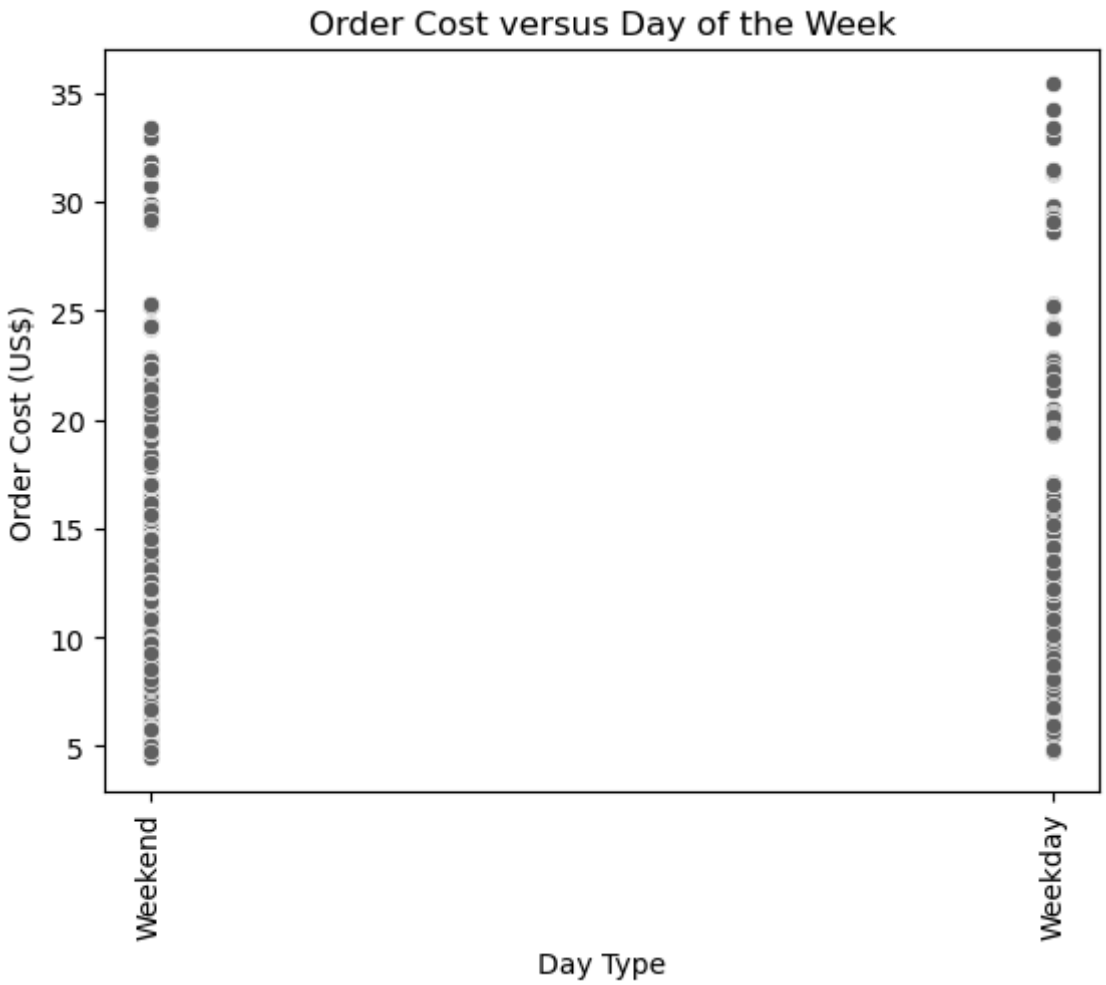
In [44]:
```
sns.pairplot(df, hue='day_of_the_week', palette=custom_color_day_of_week)
plt.savefig('Images/Pairplot.png', bbox_inches='tight')

plt.show()
```



Note:

- From above pairplot we can see that order id and customer id show uniform distributions
- Ratings have values from 0 to 5, showing more ratings for values between 3 and 5. 0 rating means the customer did not provide a rating.
- No clear relationship between order cost and food preparation, as well as order cost and delivery time.

In [45]:
```
# Scatter plot between day of the week and cost of order
sns.scatterplot(data=df, x='day_of_the_week', y='cost_of_the_order')
plt.xticks(rotation=90)
plt.title('Order Cost versus Day of the Week')
plt.xlabel('Day Type')
plt.ylabel('Order Cost (US$)')

plt.savefig('Images/Order Cost versus Day of the Week.png', bbox_inches='tight')
plt.show()
```



Note:

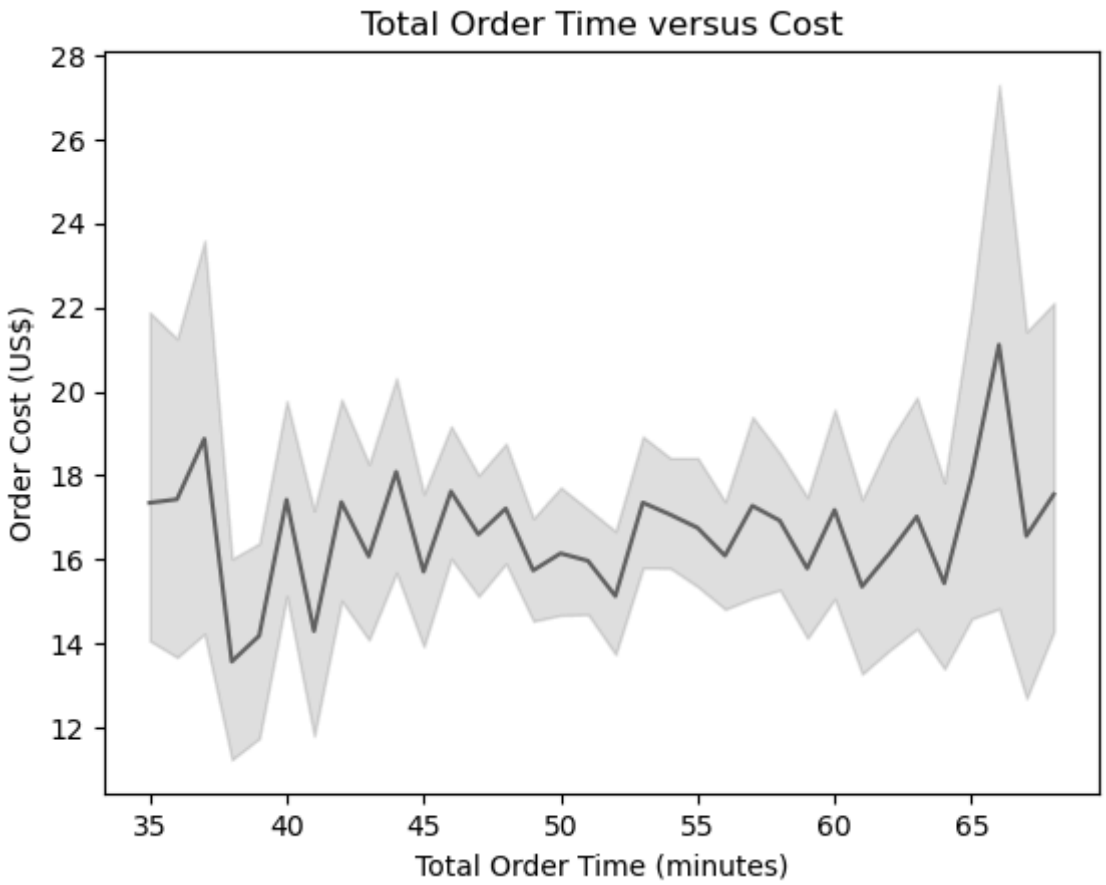
- There is no clear trend between day time and order cost. Indicating the costs remains the same during the weekend.

In [46]:
```
# Scatter plot between cuisine type and delivery time
sns.scatterplot(data=df, x='cuisine_type', y='delivery_time')
plt.xticks(rotation=90)
plt.title('Delivery Time by Cuisime')
plt.xlabel('Cuisine Type')
plt.ylabel('Delivery Time (minutes)')

plt.savefig('Images/Delivery Time by Cuisime.png', bbox_inches='tight')
plt.show()
```

Note:

- We can see in general all types of cuisine have a similar distribution in delivery time, being between 15minutes to 32 minutes
- Korean cuisine seems to have the shortest deliveries, ranging from around 16 minutes up to 26 minutes in some cases
- the Vietnamese cuisine displays deliveries between 18 minutes and around 31 minutes.

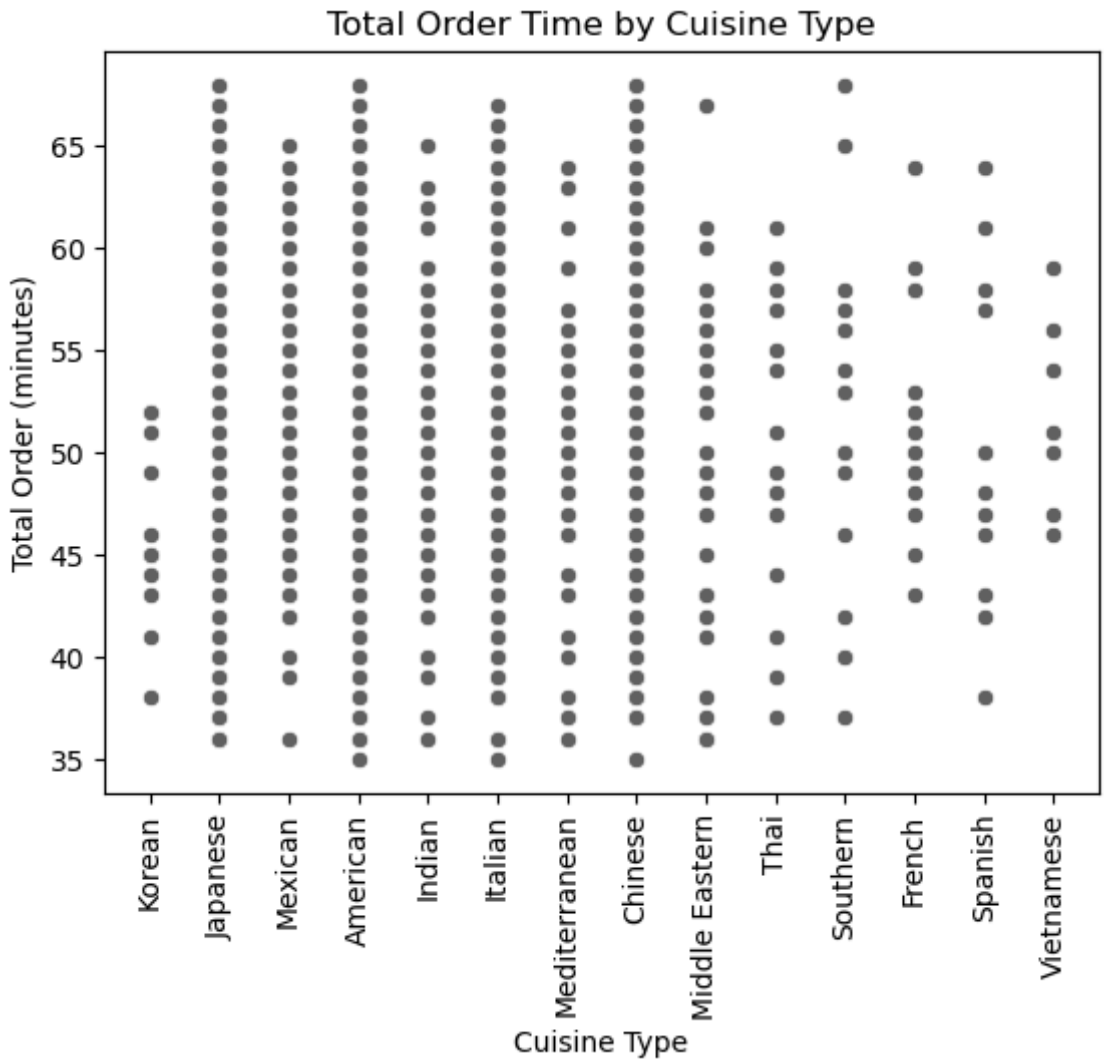
In [47]:
```
# combining food preparation time and delivery time to analyze
df['total_food_time'] = df['food_preparation_time'] + df['delivery_time']

sns.lineplot(data=df, x='total_food_time', y='cost_of_the_order')
plt.title('Total Order Time versus Cost')
plt.xlabel('Total Order Time (minutes)')
plt.ylabel('Order Cost (US$)')

plt.savefig('Images/Total Order Time versus Cost.png', bbox_inches='tight')
plt.show()
```



Note:

- Total order time does not vary much, independent of the order cost

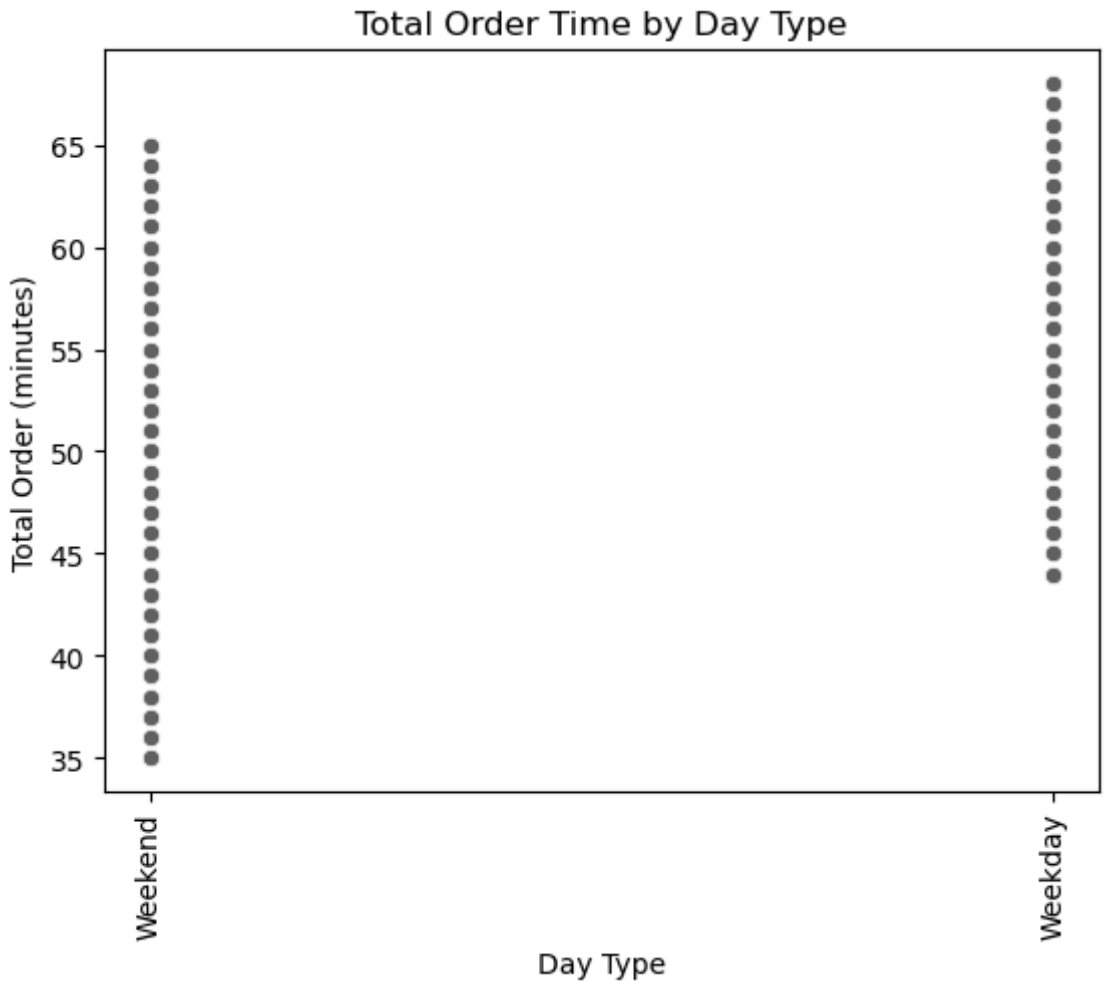
In [48]:
```
# Scatter plot between cuisine type and delivery time
sns.scatterplot(data=df, x='cuisine_type', y='total_food_time')
plt.xticks(rotation=90)
plt.title('Total Order Time by Cuisine Type')
plt.xlabel('Cuisine Type')
plt.ylabel('Total Order (minutes)')

plt.savefig('Images/Total Order Time by Cuisine Type.png', bbox_inches='tight')
plt.show()
```



Note:

- When combining the food preparation and delivery time, we can see that again, in general, there is no much variation between different cuisine types
- Korean cuisine displays shorter times in general.

In [49]:
```
# Scatter plot between day type and total food time
sns.scatterplot(data=df, x='day_of_the_week', y='total_food_time')
plt.xticks(rotation=90)
plt.title('Total Order Time by Day Type')
plt.xlabel('Day Type')
plt.ylabel('Total Order (minutes)')

plt.savefig('Images/Total Order Time by Day Type.png', bbox_inches='tight')
plt.show()
```



Note:

- When combining food preparation time and food delivery time, we observe that during weekdays, orders generally take between 45 and more than 65 minutes to be completed. However, during weekends, orders take between 35 minutes and 65 minutes.
- This variation, where some orders are completed faster on weekends compared to weekdays, might be explained by the increased staffing levels in restaurants and delivery services during weekends to manage the higher demand, helping to reduce the total order time.

In [50]:
```
# Gettign count and mean of orders delivery time by day type
# restaurant_delivery_time = df.groupby('day_of_the_week')['delivery_time'].agg(['count', 'mean'
# restaurant_delivery_time.sort_values(by='mean', ascending=False)


df_filtered = df[df['rating'] != 0]

df_filtered_count = df_filtered.shape[0]

# Gettign count and mean of orders delivery time by day type
restaurant_rating = df_filtered.groupby('cuisine_type')['rating'].agg(['count', 'mean']).reset_i
restaurant_rating.sort_values(by='mean', ascending=False)
```

Out[50]:

	cuisine_type	count	mean
11	Spanish	6	4.83
12	Thai	9	4.67
3	Indian	50	4.54
8	Mexican	48	4.42
5	Japanese	273	4.37
4	Italian	172	4.36
1	Chinese	133	4.34
10	Southern	13	4.31
2	French	10	4.30
0	American	368	4.30
9	Middle Eastern	34	4.24
7	Mediterranean	32	4.22
6	Korean	9	4.11
13	Vietnamese	5	4.00

In [51]:
```
restaurant_rating['count_perc (%)'] = (restaurant_rating['count'] / df_filtered_count)*100
restaurant_rating
```

Out[51]:

	cuisine_type	count	mean	count_perc (%)
0	American	368	4.30	31.67
1	Chinese	133	4.34	11.45
2	French	10	4.30	0.86
3	Indian	50	4.54	4.30
4	Italian	172	4.36	14.80
5	Japanese	273	4.37	23.49
6	Korean	9	4.11	0.77
7	Mediterranean	32	4.22	2.75
8	Mexican	48	4.42	4.13
9	Middle Eastern	34	4.24	2.93
10	Southern	13	4.31	1.12
11	Spanish	6	4.83	0.52
12	Thai	9	4.67	0.77
13	Vietnamese	5	4.00	0.43

Note:

- All cuisine types display a rating between 4 to 5 in average. This indicates in general, when a customer rates a restaurant, is because they liked it.

In [52]:
```
# Separate data for weekdays and weekends
df_weekday = df_filtered[df_filtered['day_of_the_week'] == 'Weekday']
df_weekend = df_filtered[df_filtered['day_of_the_week'] == 'Weekend']

# Group by "cuisine_type" to calculate the average rating by day type
weekday_ratings = df_weekday.groupby('cuisine_type')['rating'].mean().reset_index()
weekday_ratings.columns = ['cuisine_type', 'average_weekday_rating']

# For weekends
weekend_ratings = df_weekend.groupby('cuisine_type')['rating'].mean().reset_index()
weekend_ratings.columns = ['cuisine_type', 'average_weekend_rating']

# Getting information in one df
average_ratings_combined = pd.merge(weekday_ratings, weekend_ratings, on='cuisine_type', how='ou

average_ratings_combined = average_ratings_combined.sort_values(by='average_weekday_rating', asc

print(average_ratings_combined)
```

	cuisine_type	average_weekday_rating	average_weekend_rating
2	French	4.50	4.25
6	Korean	4.50	4.00
9	Middle Eastern	4.47	4.05
3	Indian	4.43	4.58
4	Italian	4.40	4.34
8	Mexican	4.38	4.44
5	Japanese	4.36	4.38
0	American	4.25	4.32
1	Chinese	4.23	4.36
7	Mediterranean	4.00	4.33
10	Southern	4.00	4.44
12	Thai	4.00	4.75
13	Vietnamese	3.00	4.67
11	Spanish	NaN	4.83

Note:

- Whenlooking at rating by cuisine type, there is no a big change in rating between weekday or weekend, except for Vietnamese food, that lowers the average rating to 3 during weekday and Spanish that there is no rating for weekday.
- Despite these changes, we can see both cuisine types, Vietnamese and Spanish, account for a little less than 1% of all the orders with rating.

In [53]:
```
# Filtering out the restaurants with no rating to avoid skewing the results
df_filtered = df[df['rating'] != 0]

# Getting the count and mean of the restaurants
restaurant_rating = df_filtered.groupby('restaurant_name')['rating'].agg(['count', 'mean']).rese
restaurant_rating.sort_values(by='count', ascending=False)
```

Out[53]:

	restaurant_name	count	mean
117	Shake Shack	133	4.28
132	The Meatball Shop	84	4.51
17	Blue Ribbon Sushi	73	4.22
16	Blue Ribbon Fried Chicken	64	4.33
104	RedFarm Broadway	41	4.24
...	...	...	...
51	Frank Restaurant	1	4.00
118	Socarrat Paella Bar	1	5.00
47	El Parador Cafe	1	5.00
79	Lucky Strike	1	4.00
0	'wichcraft	1	5.00

156 rows × 3 columns

In [54]:
```
# Applying the promo criteria of rating count of more than 50 and the average rating should be g
rest_promo = restaurant_rating[(restaurant_rating['count'] > 50) & (restaurant_rating['mean'] > 
rest_promo.reset_index(inplace=True)
rest_promo.sort_values(by='count', ascending=False)

print('The restaurants that deserves the promotional offer in the advertisement are:')
print(rest_promo.sort_values(by='count', ascending=False))
```

```
The restaurants that deserves the promotional offer in the advertisement are:
   index            restaurant_name  count  mean
2    117                Shake Shack    133  4.28
3    132          The Meatball Shop     84  4.51
1     17          Blue Ribbon Sushi     73  4.22
0     16  Blue Ribbon Fried Chicken     64  4.33
```

Notes:

- As we can see, there are 4 restaurants with more than 50 ratings and average ratings more than 4
- The four restaurants that won the promotional offer in the advertisement are: "Shake Shak", "The Meatball Shop", "Blue Ribbon Sushi" and "Blue Ribbon Fried Chicken".

In [55]:
```
# Making the revenue function
def company_net_revenue(cost):
    if cost > 20:
        return cost * 0.25
    elif cost > 5:
        return cost * 0.15
    else:
        return 0

# Applying revenue function to all orders
df['net_revenue'] = df['cost_of_the_order'].apply(company_net_revenue)
total_net_revenue = df['net_revenue'].sum()
total_net_revenue

print(f'The total net revenue of the company, when charging 25% on the orders having cost greate
```

```
The total net revenue of the company, when charging 25% on the orders having cost greater than 2
0 dollars and 15% on the orders having cost greater than 5 dollars is $6166.303 dollars
```

In [56]:
```
# Usign the total_food_time previously calculated as the food preparation time plus delivery tim
# Filtering by orders that take more than 60 minutes
long_total_delivery = df[df['total_food_time'] > 60].shape[0]

total_orders = df.shape[0]

perc_long_total_delivery = round((long_total_delivery / total_orders)*100,2)

print(f'From the {total_orders} total orders, {long_total_delivery} has a total order time (prep
```

```
From the 1898 total orders, 200 has a total order time (preparation + delivery) above 60 minute
s, which represents the 10.54%
```

In [57]:
```
# Gettign count and mean of orders delivery time by day type
restaurant_delivery_time = df.groupby('day_of_the_week')['delivery_time'].agg(['count', 'mean'])
restaurant_delivery_time.sort_values(by='mean', ascending=False)
```

Out[57]:

	day_of_the_week	count	mean
0	Weekday	547	28.34
1	Weekend	1351	22.47

In [58]:
```
print(f'The delivery time during week or weekend days is the following: ')
print(restaurant_delivery_time)
```

```
The delivery time during week or weekend days is the following:
  day_of_the_week  count  mean
0         Weekday    547 28.34
1         Weekend   1351 22.47
```

Conclusions:

Data

- Dataset contains 9 features for each of the 1898 orders listed

Order Cost

- Regardless of the day an order is placed or the cuisine type, all orders cost fall between `$5` and `$35` dollars, being 75% of them `$22.30` dollars or less.
- There is not relationship between order cost and food preparation time. So even is the order cost increases, the food preparation time does not change.In the same way, when the order cost is higher, the delivery time decreases but the amount of minutes is almost nothing, therefore, there is no relationship between these variables.
- There is no clear trend between day time and order cost. Indicating the costs remains the same during the weekend.

Restaurant

- The most popular cuisine in weekends is the American food, with 415 orders in total.
- From the top 5 restaurants by order count, three of them are American cuisine, one Italian and one is Japanese.

- Korean cuisine seems to have the shortest deliveries, ranging from around 16 minutes up to 26 minutes in some cases.
- The Vietnamese cuisine displays deliveries between 18 minutes and around 31 minutes.

Net Revenue

- The total net revenue of the company, when charging 25% on the orders having cost greater than `$20` dollars and 15% on the orders having cost greater than 5 dollars is `$6166.303` dollars

Rating

- Rating varies from 1 to 5, being 5 the best rating.
- Rating column is the only one with nulls. There are 736 no rated orders from 1898 orders in total, accounting for 38.8% of the total orders.
- All cuisine types display a rating between 4 to 5 in average. This indicates in general, when a customer rates a restaurant, is because they liked it.
- When looking at rating by cuisine type, there is no a big change in rating between weekday or weekend, except for Vietnamese food, that lowers the average rating to 3 during weekday and Spanish that there is no rating for weekday.
- Despite these changes, both cuisine types, Vietnamese and Spanish, account for a little less than 1% of all the orders with rating.
- There are 4 restaurants with more than 50 ratings and average ratings more than 4. These restaurants won the promotional offer in the advertisement and they are : "Shake Shak", "The Meatball Shop", "Blue Ribbon Sushi" and "Blue Ribbon Fried Chicken".

Order Time

- The minimun time that it takes to prepare a food, once the order is placed, is 20 minutes and the maximum time that has taken is 35 minutes. In average, food preparation is 27 minutes.
- Food preparation time does not present considerable difference in terms of distribution between weekday or weekend.
- Main difference between weekday or weekend is that despite on weekends there are more orders, the food preparation time remains similar, ranging from 20 to 43 minutes.
- Total order time does not vary much, independent of the order cost

- Majority of deliveries are within 25-30 minutes range, with an average of 24.16 minutes.

- The delivery variability is likely affected by other factors like customer location, restaurant distance, other operational conditions like amount of orders received by the restaurant, for example.
- There is no relationship between food preparation time and food delivery time. If food preparation time increases, there is no change in the delivery time. This can be explained due to the fact that our company, FoodHub, which is in charge of the deliveries is independent of the restaurants we deliver from. This indepence in relatoinship between food preparaiton and delivery is expected.
- In general all types of cuisine have a similar distribution in delivery time, being between 15minutes to 32 minutes
- When combining the food preparation and delivery time, we can see that again, in general, there is no much variation between different cuisine types

- When combining food preparation time and food delivery time, we observe that during weekdays, orders generally take between 45 and more than 65 minutes to be completed. However, during weekends, orders take between 35 minutes and 65 minutes.
- This variation, where some orders are completed faster on weekends compared to weekdays, might be explained by the increased staffing levels in restaurants and delivery services during weekends to manage the higher demand, helping to reduce the total order time.
- From the 1898 total orders, 200 has a delivery time (preparation + delivery) above 60 minutes, which represents the 10.54%

Recommendations:

Data collection

- **Customer Location:** Getting additional data points, such as customer location, could greatly enhance the analysis. Since the restaurant addresses can be found online, integrating this information would be beneficial to understand delivery times and customer distribution based on location.
    - **Order Time Tracking:** Including the day and time the order is placed could help to better understand order demand and performance. This additional data could also uncover potential seasonal trends and their impact on ratings.
    - **Day of the Week:** Adding the exact day of the week the orders are placed would provide a more granular view of customer behavior and restaurant performance.

Order Cost

- **Restaurant population:** Since order costs range between `$5` and `$35` with 75% being `$22.30` or less, consider including other restaurants with higher priced menu to open the target audience that would like to use the app.
- **Order cost versus food preparation time:** Would bebenefitial for the restaurants and overall order delivery time to investigate further why there is no significant relationship between order cost and food preparation.

Restaurant Popularity

- **Promotional Focus**: Given the popularity of certain American, Italian, and Japanese restaurants, focus marketing efforts on these cuisines during peak hours.
- **Optimizing Popular Cuisine Delivery**:
    - **American Cuisine Focus**: Given the popularity of American cuisine on weekends, coordinate with these restaurants to optimize delivery capacities during peak times.
    - **Highlight Efficient Restaurants**: Promote restaurants with consistently shorter delivery times (e.g., Korean and Vietnamese) to showcase efficient service and attract more customers.

Ratings

- **Encourage Customer Ratings**:
    - **Mandatory Feedback:** Implement a mandatory rating system to gather more comprehensive feedback.
    - **Incentives for Ratings:** Offer discounts, loyalty points, or other incentives to encourage customers to leave ratings.

    - **Promote High-Rated Restaurants:** Use top-rated restaurants in marketing campaigns to highlight quality and reliability.

Order and Delivery

- **Operational Efficiency:**
    - **Reduce Preparation Time:** Collaborate with restaurants to identify ways to reduce food preparation times without compromising quality.
    - **Manage Peak Times:** Use historical data to predict peak times and ensure adequate staffing and resources are available to meet demand efficiently.
- **Delivery Time Optimization:**
    - **Improve Delivery Accuracy:** Leverage technology to optimize delivery routes and reduce overall delivery times.
    - **Address Long Delivery Times:** Investigate orders with delivery times above 60 minutes to identify and mitigate common bottlenecks.
- **Staffing and Capacity Coordination:**
    - **Weekend Staffing:** Identify best practices from weekend operations, such as increased staffing, that can be applied during peak weekday times to improve efficiency.
    - **Monitor & Adjust Staffing Levels:** Work with restaurants to manage staffing levels dynamically based on demand predictions.
- **Consistency in Service:**
    - **Standardize Delivery Times:** Aim to keep delivery times within the 25-30 minute range as much as possible to maintain customer satisfaction.

Summary

To enhance the delivery company's operations:

- **Data Enhancement**: Gather more detailed data on customer locations, order times, and specific days to streamline logistics and marketing efforts.
- **Optimize Delivery**: Focus on improving delivery times, especially during peak hours, through better staffing, route optimization, and restaurant coordination.
- **Increase Customer Feedback**: Encourage mandatory ratings and incentivize feedback to gain insights for continuous improvement.
- **Focus on High-Rated Restaurants**: Promote top-rated restaurants to leverage their popularity and reliability.
- **Improve Efficiency**: Standardize preparation and delivery times across different cuisines and manage peak times effectively to maintain high service quality.
