

Foodhub Order Analysis

Summer Boot Camp Project

June 2024

Presented by
- Ritu Raj Kumar

Index

List Of Tables

List Of Figures

✓ Problem Statement/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: 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

Basic Things

```
#importing libraries
import pandas as pd
import numpy as np
import matplotlib.pyplot as plt
import seaborn as sns
import plotly.express as px
from scipy import stats

# Load the dataset
df = pd.read_csv('2-foodhub_order_New.csv')
```

1. Display the First 5 Rows

```
df.head()
```

	order_id	customer_id	restaurant_name	cuisine_type	cost_of_the_order	day_of_th
0	1477147	337525	Hangawi	Korean	30.75	W
1	1477685	358141	Blue Ribbon Sushi Izakaya	Japanese	12.08	W
2	1477070	66393	Cafe Habana	Mexican	12.23	W
			Blue Ribbon Fried			

	0	1	2	3	4
order_id	1477147	1477685	1477070	1477334	1478249
customer_id	337525	358141	66393	106968	76942
restaurant_name	Hangawi	Blue Ribbon Sushi Izakaya	Cafe Habana	Blue Ribbon Fried Chicken	Dirty Bird to Go
cuisine_type	Korean	Japanese	Mexican	American	American
cost_of_the_order	30.75	12.08	12.23	29.2	11.59
day_of_the_week	Weekend	Weekend	Weekday	Weekend	Weekday
rating	Not given	Not given	5	3	4
food_preparation_time	25.0	25.0	23.0	25.0	25.0
delivery_time	20	?	28	15	24

Observations:

Rating - Not Given delivery time has error values (?).

2. Display the Last 5 Rows


```
df.tail()
```



	order_id	customer_id	restaurant_name	cuisine_type	cost_of_the_order	day_of
1893	1476701	292602	Chipotle Mexican Grill \$1.99 Delivery	Mexican	22.31	
1894	1477421	397537	The Smile	American	12.18	
1895	1477819	35309	Blue Ribbon Sushi	Japanese	25.22	



	1893	1894	1895	1896	1897
order_id	1476701	1477421	1477819	1477513	1478056
customer_id	292602	397537	35309	64151	120353
restaurant_name	Chipotle Mexican Grill \$1.99 Delivery	The Smile	Blue Ribbon Sushi	Jack's Wife Freda	Blue Ribbon Sushi
cuisine_type	Mexican	American	Japanese	Mediterranean	Japanese
cost_of_the_order	22.31	12.18	25.22	12.18	19.45
day_of_the_week	Weekend	Weekend	Weekday	Weekday	Weekend
rating	5	5	Not given	5	Not given
food_preparation_time	31.0	31.0	31.0	23.0	28.0
delivery_time	17	19	24	31	24




Observations:

Resturant name has \$

3. Check the Shape of the Dataset:

```
df.shape
```




```
(1898, 9)
```

Observations:

Shape of the data set is Rows: 1898 & Col: 9

4. Check the Data Types of Each Feature:

```
df.dtypes
```



```

order_id          int64
customer_id       int64
restaurant_name    object
cuisine_type       object
cost_of_the_order  float64
day_of_the_week    object
rating            object
food_preparation_time float64
delivery_time      object
dtype: object

```

Observations:

Here delivery_time is an object type that we need to change

5. Check the Statistical summary

```
df.describe(include='all')
```



	order_id	customer_id	restaurant_name	cuisine_type	cost_of_the_order
count	1.898000e+03	1898.000000	1898	1895	1898.000000
unique	NaN	NaN	178	14	NaN
top	NaN	NaN	Shake Shack	American	NaN
freq	NaN	NaN	219	582	NaN
mean	1.477496e+06	171168.478398	NaN	NaN	80.722007
std	5.480497e+02	113698.139743	NaN	NaN	2798.141333
min	1.476547e+06	1311.000000	NaN	NaN	0.000000
25%	1.477021e+06	77787.750000	NaN	NaN	12.080000
50%	1.477496e+06	128600.000000	NaN	NaN	14.160000
75%	1.477970e+06	270525.000000	NaN	NaN	22.310000

Observations:

Issue in delivery_time, cost_of_the_order - Holds a outlier

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          1895 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 1896 non-null   float64
8   delivery_time         1898 non-null   object
dtypes: float64(2), int64(2), object(5)
memory usage: 133.6+ KB
```

Observations:

Rating - Object

- delivery - (it should be int)
- time - object (it should be int)

6. Check the null values

df.isnull().sum()



```
order_id      0
customer_id   0
restaurant_name 0
cuisine_type   3
cost_of_the_order 0
day_of_the_week 0
rating         0
food_preparation_time 2
delivery_time  0
dtype: int64
```

Observations:

There are 3 Null values in cuisine_type

There are 2 Null values in food_preparation_time

7. Check the duplicate values

```
df.duplicated().sum()
```

```
↗ 0
```

Observations:

In This data set there is no null values

✓ 8. Check the anomalies or wrong entries.

```
df['day_of_the_week'].unique()
df['rating'].unique()
```

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

Observations:

In columnn of "rating" there are value - 5,4,3 which are considerable but also some exceptions like "Not given" values

✓ 9. Check the outliers and their authenticity.

```
def detect_iqr(df):
    Q1 = df.quantile(0.25)
    Q3 = df.quantile(0.75)
    IQR = Q3 - Q1
    lower_bound = Q1 - 1.5 * IQR
    upper_bound = Q3 + 1.5 * IQR
    outliers = (df < lower_bound) | (df > upper_bound)
    return outliers

# Identify outliers in numerical columns using IQR
numerical_columns = df.select_dtypes(include=['int', 'float']).columns
outliers_iqr = df[numerical_columns].apply(detect_iqr)

# Set up the plot
plt.figure(figsize=(12, 6))

# Create the boxplot
sns.boxplot(data=df[numerical_columns], orient='h', notch=True, palette='Set2')

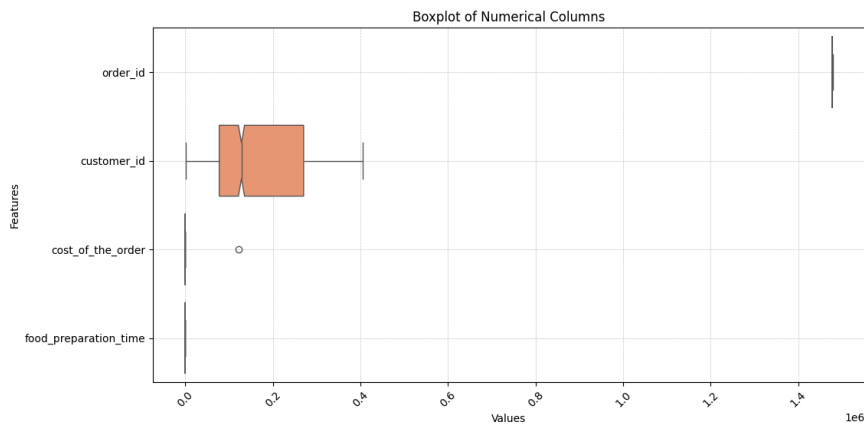
# Add title and labels
plt.title('Boxplot of Numerical Columns')
plt.xlabel('Values')
plt.ylabel('Features')

# Rotate x-axis labels for better readability
plt.xticks(rotation=45)

# Add grid for better visualization of values
plt.grid(True, linestyle='--', linewidth=0.5, alpha=0.7)

# Show the plot
plt.show()

# Display the outliers found using IQR
print("Outliers detected using IQR method:")
print(outliers_iqr.sum())
```



Outliers detected using IQR method:

```
order_id      0
customer_id   0
cost_of_the_order  1
food_preparation_time  0
dtype: int64
```

Observations:

In "cost_of_the_order", "customer_id " and "order_id" outliers present that we need correct

10. Data Cleaning

```
df.duplicated().sum()
df[df.duplicated()==True]
```



```
order_id  customer_id  restaurant_name  cuisine_type  cost_of_the_order  day_of_the
```

Invalid values

```
# we will check only for
df['delivery_time'].unique()
df['delivery_time'].value_counts()
df[df['delivery_time']=='?']
df['delivery_time']= df['delivery_time'].replace('?',np.nan)
df[df['delivery_time']=='?']
```



```
order_id  customer_id  restaurant_name  cuisine_type  cost_of_the_order  day_of_the
```

	order_id	customer_id	restaurant_name	cuisine_type	cost_of_the_order	day_of_the_week	rating	food_preparation_time	delivery_time
1	1477685	358141	Blue Ribbon Sushi Izakaya	Japanese	12.08	Weekend	Not given	25.0	?
180	1476808	84700	Pepe Giallo	Italian	14.60	Weekday	3	32.0	?

Values - Corrected

MISSING VAUIES

```
# Check for rows with any missing values
missing_rows = df[df.isnull().sum(axis=1) > 0]

# Summarize the number of missing values per column
missing_values_per_column = df.isnull().sum()

# Calculate the percentage of missing values per column
missing_values_percentage = df.isnull().sum() / len(df) * 100

# Convert 'delivery_time' to float
df['delivery_time'] = df['delivery_time'].astype('float')

# Plot box plots for 'food_preparation_time' and 'delivery_time'
for column in ['food_preparation_time', 'delivery_time']:
    plt.figure(figsize=(10, 3))
    sns.boxplot(data=df, x=column)
    plt.title(f'Box Plot for {column}')
    plt.show()

# Define the function to calculate lower and upper bounds for outlier removal
def calculate_bounds(col):
    Q1, Q3 = col.quantile([0.25, 0.75])
    IQR = Q3 - Q1
    lower_bound = Q1 - (1.5 * IQR)
    upper_bound = Q3 + (1.5 * IQR)
    return lower_bound, upper_bound

# Calculate lower and upper bounds for 'cost_of_the_order'
lower_bound, upper_bound = calculate_bounds(df['cost_of_the_order'])

# Identify outliers in 'cost_of_the_order'
outliers_upper = df[df['cost_of_the_order'] > upper_bound]
outliers_lower = df[df['cost_of_the_order'] < lower_bound]

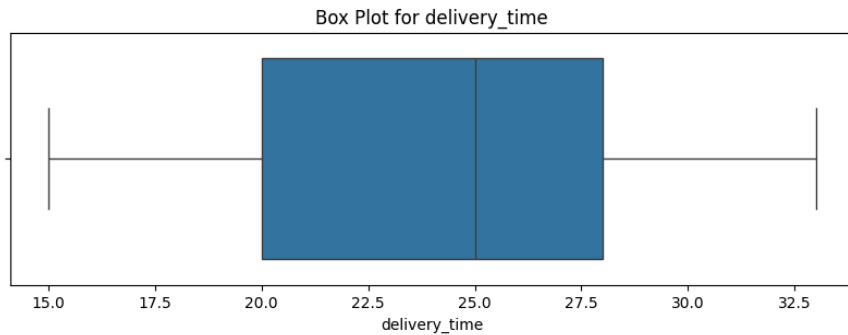
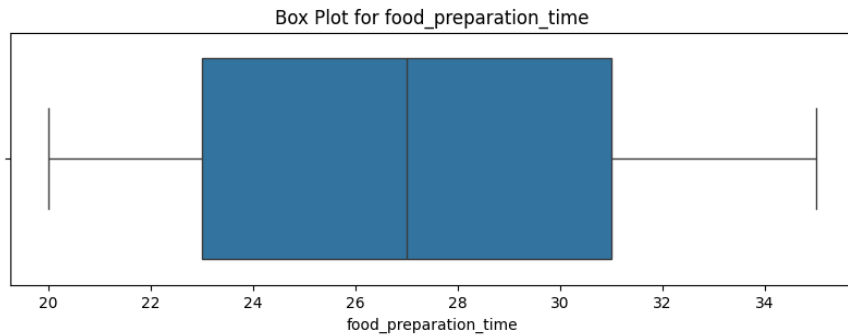
# Replace outliers in 'cost_of_the_order' with bounds
df['cost_of_the_order'] = np.where(df['cost_of_the_order'] > upper_bound, upper_bound, df['cost_of_the_order'])
df['cost_of_the_order'] = np.where(df['cost_of_the_order'] < lower_bound, lower_bound, df['cost_of_the_order'])

# Verify the changes
print(df['cost_of_the_order'].describe())

# Fill missing values for 'cuisine_type' with the mode
mode_cuisine = df['cuisine_type'].mode().values[0]
df['cuisine_type'] = df['cuisine_type'].replace(np.nan, mode_cuisine)

# Fill missing values for 'food_preparation_time' and 'delivery_time' with the mean
mean_food_preparation_time = df['food_preparation_time'].mean()
mean_delivery_time = df['delivery_time'].mean()

df['food_preparation_time'].fillna(mean_food_preparation_time, inplace=True)
df['delivery_time'].fillna(mean_delivery_time, inplace=True)
```



```
count    1898.000000
mean      16.505809
std        7.507834
min         0.000000
25%       12.080000
50%       14.160000
75%       22.310000
max       37.655000
Name: cost_of_the_order, dtype: float64
```

```
df.describe()
```



	order_id	customer_id	cost_of_the_order	food_preparation_time	delivery
count	1.898000e+03	1898.000000	1898.000000	1898.000000	1898.000000
mean	1.477496e+06	171168.478398	16.505809	27.371835	24.116839
std	5.480497e+02	113698.139743	7.507834	4.631768	4.580497
min	1.476547e+06	1311.000000	0.000000	20.000000	15.000000
25%	1.477021e+06	77787.750000	12.080000	23.000000	20.000000
50%	1.477496e+06	128600.000000	14.160000	27.000000	25.000000
75%	1.477970e+06	270525.000000	22.310000	31.000000	28.000000
max	1.478444e+06	405334.000000	37.655000	35.000000	33.000000

Observations:

Here as we can see all the data is managed outliers were removed and whole data were managed

Main

✓ 1. Order Analysis

- ✓ • Total number of orders in the dataset


```
total_orders = df['order_id'].nunique()
print(f'Total number of orders: {total_orders}')
```

```
df.shape
```

```
Total number of orders: 1898
(1898, 9)
```

Observations:

Total number of orders placed: 1898

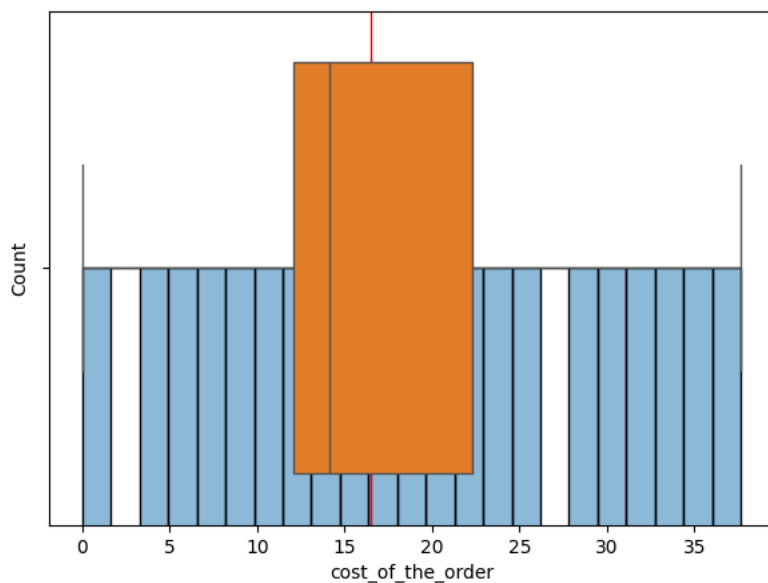
• The average cost of an order

```
plt.figure(figsize = (7,5))
plt.axvline(df['cost_of_the_order'].mean(), color='r', linewidth=1)
sns.histplot(data = df, x = 'cost_of_the_order', kde = True)
```

```
sns.boxplot(data = df, x = 'cost_of_the_order')
```

```
average_cost = df['cost_of_the_order'].mean()
print(f'Average cost of an order: ${average_cost:.2f}')
```

```
Average cost of an order: $16.51
```



Observations:

The Average FoodHub order cost is **\$16.51** and the *majority* of the order cost ranges from **11 - 13 USD**.

• Number Of unique customers have placed orders

```
unique_customers = df['customer_id'].nunique()
print(f'Number of unique customers: {unique_customers}')
```

```
df['customer_id'].value_counts().unique()
```

```
Number of unique customers: 1200
array([13, 10, 9, 8, 7, 6, 5, 4, 3, 2, 1])
```

Observations:

- No Of unique customers is **1200**
- Among these customers there is an order count range of **1 - 13**.

• Restaurant with highest number of orders

```
#The restaurant has received the highest number of orders with numbers
restaurant_id = df['restaurant_id'].value_counts().index[0]
print(f'Restaurant with highest number of orders: {restaurant_id}')
```

```
restaurant_orders = ut.groupby( restaurant_name )[ order_id ].nunique().sort_values(ascending=False)
print("Restaurant with the highest number of orders:")
print(restaurant_orders.head(1))
```

```
➦ Restaurant with the highest number of orders:
restaurant_name
Shake Shack      219
Name: order_id, dtype: int64
```

Observations:

Restaurant Shake Shack has received the highest number of orders: 219

✓ 2. Customer Behavior

✓ • The average rating given by customers?

```
# Convert 'rating' column to numeric, handling non-numeric values
df['rating'] = pd.to_numeric(df['rating'], errors='coerce')

#Calculate the average rating, ignoring missing values
average_rating = df['rating'].mean()
print(f'Average rating given by customers: {average_rating:.2f}')
```

```
➦ Average rating given by customers: 4.34
```

Observations:

Average rating given by customers is 4.34

✓ • How does the rating vary between weekdays and weekends

```
weekday_ratings = df[(df['day_of_the_week'] == 'Weekday') & (df['rating'] != 'Not given')]['rating'].astype(float)
weekend_ratings = df[(df['day_of_the_week'] == 'Weekend') & (df['rating'] != 'Not given')]['rating'].astype(float)

avg_weekday_rating = weekday_ratings.mean()
avg_weekend_rating = weekend_ratings.mean()

print(f'Average weekday rating: {avg_weekday_rating:.2f}')
print(f'Average weekend rating: {avg_weekend_rating:.2f}')
```

```
➦ Average weekday rating: 4.31
Average weekend rating: 4.36
```

Observations:

Average **weekday** rating by Customer: 4.31

Average **weekend** rating by Customer: 4.36

✓ • Cuisine type is ordered the most?

```
# Import necessary libraries
import pandas as pd
import plotly.express as px

# Assuming 'df' is your DataFrame and 'cuisine_type' and 'order_id' are your columns

# Get the count of orders for each cuisine type
cuisine_order_counts = df['cuisine_type'].value_counts().reset_index()
cuisine_order_counts.columns = ['Cuisine Type', 'Number of Orders']

# Create a Plotly bar plot
fig = px.bar(cuisine_order_counts,
             x='Cuisine Type',
             y='Number of Orders',
             labels={'x': 'Cuisine Type', 'y': 'Number of Orders'},
             title='Most Ordered Cuisine Types',
             hover_data={'Number of Orders': True})
```

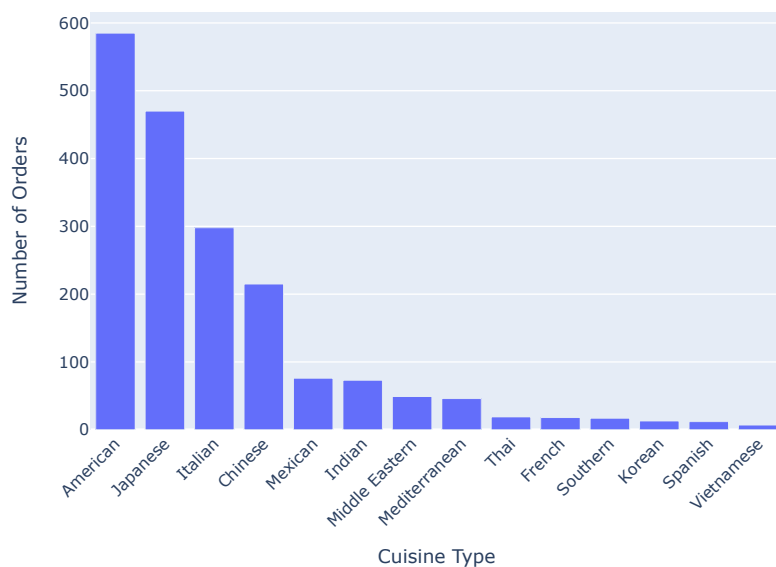
```
# Update layout for better appearance
fig.update_layout(
    xaxis_title='Cuisine Type',
    yaxis_title='Number of Orders',
    xaxis_tickangle=-45
)

# Show the interactive plot
fig.show()

# Print the most ordered cuisine type
cuisine_orders = df.groupby('cuisine_type')['order_id'].nunique().sort_values(ascending=False)
print("Most ordered cuisine type: ")
print(cuisine_orders.head(1))
```



Most Ordered Cuisine Types



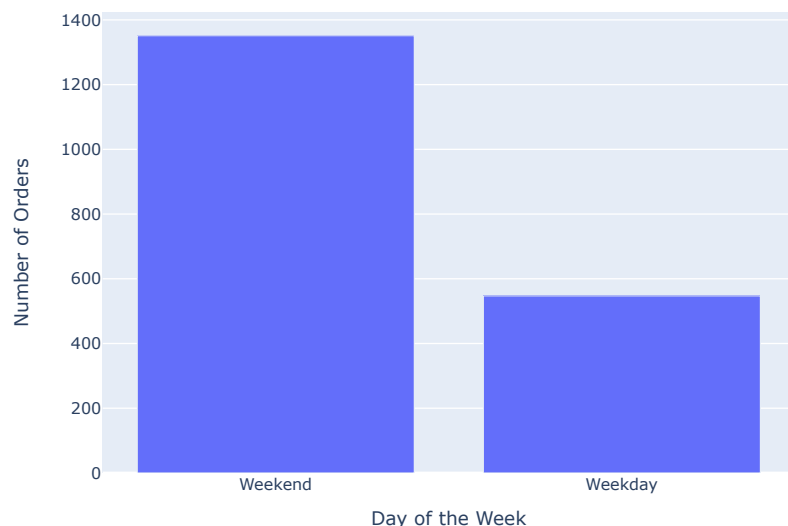
Most ordered cuisine type:

```
cuisine_type
American    585
```

```
order_id      0      1      2      3      4      5      6      7      8      9      10      11      12      13      14      15      16      17      18      19      20      21      22      23      24      25      26      27      28      29      30      31      32      33      34      35      36      37      38      39      40      41      42      43      44      45      46      47      48      49      50      51      52      53      54      55      56      57      58      59      60      61      62      63      64      65      66      67      68      69      70      71      72      73      74      75      76      77      78      79      80      81      82      83      84      85      86      87      88      89      90      91      92      93      94      95      96      97      98      99      100      101      102      103      104      105      106      107      108      109      110      111      112      113      114      115      116      117      118      119      120      121      122      123      124      125      126      127      128      129      130      131      132      133      134      135      136      137      138      139      140      141      142      143      144      145      146      147      148      149      150      151      152      153      154      155      156      157      158      159      160      161      162      163      164      165      166      167      168      169      170      171      172      173      174      175      176      177      178      179      180      181      182      183      184      185      186      187      188      189      190      191      192      193      194      195      196      197      198      199      200      201      202      203      204      205      206      207      208      209      210      211      212      213      214      215      216      217      218      219      220      221      222      223      224      225      226      227      228      229      230      231      232      233      234      235      236      237      238      239      240      241      242      243      244      245      246      247      248      249      250      251      252      253      254      255      256      257      258      259      260      261      262      263      264      265      266      267      268      269      270      271      272      273      274      275      276      277      278      279      280      281      282      283      284      285      286      287      288      289      290      291      292      293      294      295      296      297      298      299      300      301      302      303      304      305      306      307      308      309      310      311      312      313      314      315      316      317      318      319      320      321      322      323      324      325      326      327      328      329      330      331      332      333      334      335      336      337      338      339      340      341      342      343      344      345      346      347      348      349      350      351      352      353      354      355      356      357      358      359      360      361      362      363      364      365      366      367      368      369      370      371      372      373      374      375      376      377      378      379      380      381      382      383      384      385      386      387      388      389      390      391      392      393      394      395      396      397      398      399      400      401      402      403      404      405      406      407      408      409      410      411      412      413      414      415      416      417      418      419      420      421      422      423      424      425      426      427      428      429      430      431      432      433      434      435      436      437      438      439      440      441      442      443      444      445      446      447      448      449      450      451      452      453      454      455      456      457      458      459      460      461      462      463      464      465      466      467      468      469      470      471      472      473      474      475      476      477      478      479      480      481      482      483      484      485      486      487      488      489      490      491      492      493      494      495      496      497      498      499      500      501      502      503      504      505      506      507      508      509      510      511      512      513      514      515      516      517      518      519      520      521      522      523      524      525      526      527      528      529      530      531      532      533      534      535      536      537      538      539      540      541      542      543      544      545      546      547      548      549      550      551      552      553      554      555      556      557      558      559      560      561      562      563      564      565      566      567      568      569      570      571      572      573      574      575      576      577      578      579      580      581      582      583      584      585      586      587      588      589      590      591      592      593      594      595      596      597      598      599      600      601      602      603      604      605      606      607      608      609      610      611      612      613      614      615      616      617      618      619      620      621      622      623      624      625      626      627      628      629      630      631      632      633      634      635      636      637      638      639      640      641      642      643      644      645      646      647      648      649      650      651      652      653      654      655      656      657      658      659      660      661      662      663      664      665      666      667      668      669      670      671      672      673      674      675      676      677      678      679      680      681      682      683      684      685      686      687      688      689      690      691      692      693      694      695      696      697      698      699      700      701      702      703      704      705      706      707      708      709      710      711      712      713      714      715      716      717      718      719      720      721      722      723      724      725      726      727      728      729      730      731      732      733      734      735      736      737      738      739      740      741      742      743      744      745      746      747      748      749      750      751      752      753      754      755      756      757      758      759      760      761      762      763      764      765      766      767      768      769      770      771      772      773      774      775      776      777      778      779      780      781      782      783      784      785      786      787      788      789      790      791      792      793      794      795      796      797      798      799      800      801      802      803      804      805      806      807      808      809      810      811      812      813      814      815      816      817      818      819      820      821      822      823      824      825      826      827      828      829      830      831      832      833      834      835      836      837      838      839      840      841      842      843      844      845      846      847      848      849      850      851      852      853      854      855      856      857      858      859      860      861      862      863      864      865      866      867      868      869      870      871      872      873      874      875      876      877      878      879      880      881      882      883      884      885      886      887      888      889      890      891      892      893      894      895      896      897      898      899      900      901      902      903      904      905      906      907      908      909      910      911      912      913      914      915      916      917      918      919      920      921      922      923      924      925      926      927      928      929      930      931      932      933      934      935      936      937      938      939      940      941      942      943      944      945      946      947      948      949      950      951      952      953      954      955      956      957      958      959      960      961      962      963      964      965      966      967      968      969      970      971      972      973      974      975      976      977      978      979      980      981      982      983      984      985      986      987      988      989      990      991      992      993      994      995      996      997      998      999      1000      1001      1002      1003      1004      1005      1006      1007      1008      1009      1010      1011      1012      1013      1014      1015      1016      1017      1018      1019      1020      1021      1022      1023      1024      1025      1026      1027      1028      1029      1030      1031      1032      1033      1034      1035      1036      1037      1038      1039      1040      1041      1042      1043      1044      1045      1046      1047      1048      1049      1050      1051      1052      1053      1054      1055      1056      1057      1058      1059      1060      1061      1062      1063      1064      1065      1066      1067      1068      1069      1070      1071      1072      1073      1074      1075      1076      1077      1078      1079      1080      1081      1082      1083      1084      1085      1086      1087      1088      1089      1090      1091      1092      1093      1094      1095      1096      1097      1098      1099      1100      1101      1102      1103      1104      1105      1106      1107      1108      1109      1110      1111      1112      1113      1114      1115      1116      1117      1118      1119      1120      1121      1122      1123      1124      1125      1126      1127      1128      1129      1130      1131      1132      1133      1134      1135      1136      1137      1138      1139      1140      1141      1142      1143      1144      1145      1146      1147      1148      1149      1150      1151      1152      1153      1154      1155      1156      1157      1158      1159      1160      1161      1162      1163      1164      1165      1166      1167      1168      1169      1170      1171      1172      1173      1174      1175      1176      1177      1178      1179      1180      1181      1182      1183      1184      1185      1186      1187      1188      1189      1190      1191      1192      1193      1194      1195      1196      1197      1198      1199      1200      1201      1202      1203      1204      1205      1206      1207      1208      1209      1210      1211      1212      1213      1214      1215      1216      1217      1218      1219      1220      1221      1222      1223      1224      1225      1226      1227      1228      1229      1230      1231      1232      1233      1234      1235      1236      1237      1238      1239      1240      1241      1242      1243      1244      1245      1246      1247      1248      1249      1250      1251      1252      1253      1254      1255      1256      1257      1258      1259      1260      1261      1262      1263      1264      1265      1266      1267      1268      1269      1270      1271      1272      1273      1274      1275      1276      1277      1278      1279      1280      1281      1282      1283      1284      1285      1286      1287      1288      1289      1290      1291      1292      1293      1294      1295      1296      1297      1298      1299      1300      1301      1302      1303      1304      1305      1306      1307      1308      1309      1310      1311      1312      1313      1314      1315      1316      1317      1318      1319      1320      1321      1322      1323      1324      1325      1326      1327      1328      1329      1330      1331      1332      1333      1334      1335      1336      1337      1338      1339      1340      1341      1342      1343      1344      1345      1346      1347      1348      1349      1350      1351      1352      1353      1354      1355      1356      1357      1358      1359      1360      1361      1362      1363      1364      1365      1366      1367      1368      1369      1370      1371      1372      1373      1374      1375      1376      1377      1378      1379      1380      1381      1382      1383      1384      1385      1386      1387      1388      1389      1390      1391      1392      1393      1394      1395      1396      1397      1398      1399      1400      1401      1402      1403      1404      1405      1406      1407      1408      1409      1410      1411      1412      1413      1414      1415      1416      1417      1418      1419      1420      1421      1422      1423      1424      1425      1426      1427      1428      1429      1430      1431      1432      1433      1434      1435      1436      1437      1438      1439      1440      1441      1442      1443      1444      1445      1446      1447      1448      1449      1450      1451      1452      1453      1454      1455      1456      1457      1458      1459      1460      1461      1462      1463      1464      1465      1466      1467      1468      1469      1470      1471      1472      1473      1474      1475      1476      1477      1478      1479      1480      1481      1482      1483      1484      1485      1486      1487      1488      1489      1490      1491      1492      1493      1494      1495      1496      1497      1498      1499      1500      1501      1502      1503      1504      1505      1506      1507      1508      1509      1510      1511      1512      1513      1514      1515      1516      1517      1518      1519      1520      1521      1522      1523      1524      1525      1526      1527      1528      1529      1530      1531      1532      1533      1534      1535      1536      1537      1538      1539      1540      1541      1542      1543      1544      1545      1546      1547      1548      1549      1550      1551      1552      1553      1554      1555      1556      1557      1558      1559      1560      1561      1562      1563      1564      1565      1566      1567      1568      1569      1570      1571      1572      1573      1574      1575      1576      1577      1578      1579      1580      1581      1582      1583      1584      1585      1586      1587      1588      1589      1590      1591      1592      1593      1594      1595      1596      1597      1598      1599      1600      1601      1602      1603      1604      1605      1606      1607      1608      1609      1610      1611      1612      1613      1614      1615      1616      1617      1618      1619      1620      1621      1622      1623      1624      1625      1626      1627      1628      1629      1630      1631      1632      1633      1634      1635      1636      1637      1638      1639      1640      1641      1642      1643      1644      1645      1646      1647      1648      1649      1650      1651      1652      1653      1654      1655      1656      1657      1658      1659      1660      1661      1662      1663      1664      1665      1666      1667      1668      1669      1670      1671      1672      1673      1674      1675      1676      1677      1678      1679      1680      1681      1682      1683      1684      1685      1686      1687      1688      1689      1690      1691      1692      1693      1694      1695      1696      1697      1698      1699      1700      1701      1702      1703      1704      1705      1706      1707      1708      1709      1710      1711      1712      1713      1714      1715      1716      1717      1718      1719      1720      1721      1722      1723      1724      1725      1726      1727      1728      1729      1730      1731      1732      1733      1734      1735      1736      1737      1738      1739      1740      1741      1742      1743      1744      1745      1746      1747      1748      1749      1750      1751      1752      1753      1754      1755      1756      1757      1758      1759      1760      1761      1762      1763      1764      1765      1766      1767      1768      1769      1770      1771      1772      1773      1774      1775      1776      1777      1778      1779      1780      1781      1782      1783      1784      1785      1786      1787      1788      1789      1790      1791      1792      1793      1794      1795      1796      1797      1798      1799      1800      1801      1802      1803      1804      1805      1806      1807      1808      1809      1810      1811      1812      1813      1814      1815      1816      1817      1818      1819      1820      1821      1822      1823      1824      1825      1826      1827      1828      1829      1830      1831      1832      1833      1834      1835      1836      1837      1838      1839      1840      1841      1842      1843      1844      1845      1846      1847      1848      1849      1850      1851      1852      1853      1854      1855      1856      1857      1858      1859      1860      1861      1862      1863      1864      1865      1866      1867      1868      1869      1870      1871      1872      1873      1874      1875      1876      1877      1878      1879      1880      1881      1882      1883      1884      1885      1886      1887      1888      1889      1890      1891      1892      1893      1894      1895      1896      1897      1898      1899      1900      1901      1902      1903      1904      1905      1906      1907      1908      1909      1910      1911      1912      1913      1914      1915      1916      1917      1918      1919      1920      1921      1922      1923      1924      1925      1926      1927      1928      1929      1930      1931      1932      1933      1934      1935      1936      1937      1938      1939      1940      1941      1942      1943      1944      1945      1946      1947      1948      1949      1950      1951      1952      1953      1954      1955      1956      1957      1958      1959      1960      1961      1962      1963      1964      1965      1966      1967      1968      1969      1970      1971      1972      1973      1974      1975      1976      1977      1978      1979      1980      1981      1982      1983      1984      1985      1986      1987      1988      1989      1990      1991      1992      1993      1994      1995      1996      1997      1998      1999      2000      2001      2002      2003      2004      2005      2006      2007      2008      2009      2010      2011      2012      2013      2014      2015      2016      2017      2018      2019      2020      2021      2022      2023      2024      2025      2026      2027      2028      2029      2030      2031      2032      2033      2034      2035      2036      2037      2038      2039      2040      2041      2042      2043      2044      2045      2046      2047      2048      2049      2050      2051      2052      2053      2054      2055      2056      2057      2058      2059      2060      2061      2062      2063      2064      2065      2066      2067      2068      2069      2070      2071      2072      2073      2074      2075      2076      2077      2078      2079      2080      2081      2082      2083      2084      2085      2086      2087      2088      2089      2090      2091      2092      2093      2094      2095      2
```



Distribution of Orders Across Different Days of the Week



Observations:

Number Order Placed On **Weekend** is *Higher* than **Weekday** : 1351 > 547

✓ 3. Restaurant Performance

✓ • The average food preparation time for each restaurant?

```
avg_prep_time = df.groupby('restaurant_name')['food_preparation_time'].mean().sort_values()
print(avg_prep_time)
"""
Only use When Need
avg_prep_time_per_restaurant = df.groupby('restaurant_name')['food_preparation_time'].mean()
print(avg_prep_time_per_restaurant)

# Visualization
avg_prep_time_per_restaurant.plot(kind='bar', figsize=(15, 5), title='Average Food Preparation Time by Restaurant')
plt.xlabel('Restaurant Name')
plt.ylabel('Average Food Preparation Time (minutes)')
plt.show()"""
```



```
restaurant_name
Haru Gramercy Park      20.0
67 Burger               20.0
Frank Restaurant        20.0
Despaña                 20.5
Sarabeth's West        21.0
...
Taro Sushi              35.0
Cipriani Le Specialita  35.0
Kambi Ramen House       35.0
Klong                   35.0
Sushi Choshi            35.0
Name: food_preparation_time, Length: 178, dtype: float64
'\nOnly use When Need\navg_prep_time_per_restaurant = df.groupby('restaurant_name')
['food_preparation_time'].mean()\nprint(avg_prep_time_per_restaurant)\n\n# Visualiza
tion\navg_prep_time_per_restaurant.plot(kind='bar', figsize=(15, 5), title='Average
```

Observations:

Restaurants show varying average food preparation times, impacting customer wait times and operational efficiency.

Haru Gramercy Park: 20.0 minutes

67 Burger: 20.0 minutes

Frank Restaurant: 20.0 minutes

Despaña: 20.5 minutes

Sarabeth's West: 21.0 minutes

Taro Sushi: 35.0 minutes

Cipriani Le Specialita: 35.0 minutes

✓ • Resturant has the shortest average food preparation time

```
# Calculate the average preparation time by restaurant name
avg_prep_time = df.groupby('restaurant_name')['food_preparation_time'].mean().reset_index()

# Sort the data by average preparation time in ascending order
avg_prep_time = avg_prep_time.sort_values(by='food_preparation_time')

# Find the top 5 restaurants with the shortest average preparation time
top_5_restaurants = avg_prep_time.head(5)
print("Top 5 Restaurants with the Shortest Average Preparation Time:")
for index, row in top_5_restaurants.iterrows():
    print(f"{row['restaurant_name']}: {row['food_preparation_time']:.2f} minutes")

# Create the bar chart
fig = px.bar(avg_prep_time, x='restaurant_name', y='food_preparation_time',
             title='Average Preparation Time by Restaurant',
             labels={'restaurant_name': 'Restaurant', 'food_preparation_time': 'Average Preparation Time'},
             hover_data={'food_preparation_time': ':.2f'})

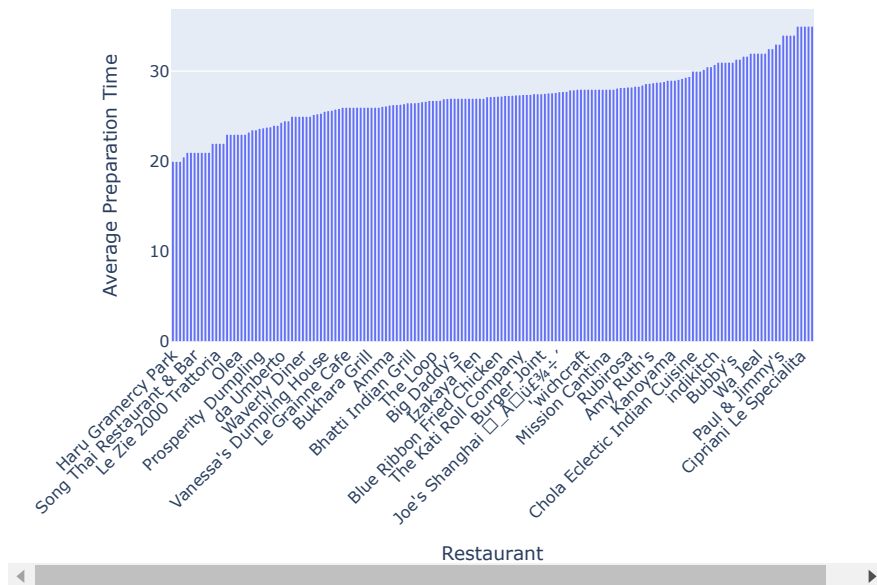
# Update the layout for better visualization
fig.update_layout(xaxis_tickangle=-45)

# Show the figure
fig.show()
```

➡ Top 5 Restaurants with the Shortest Average Preparation Time:

- Haru Gramercy Park: 20.00 minutes
- 67 Burger: 20.00 minutes
- Frank Restaurant: 20.00 minutes
- DespaIta: 20.50 minutes
- Sarabeth's West: 21.00 minutes

Average Preparation Time by Restaurant



Observations:

Restaurants with the Shortest Average Preparation Time:

- Haru Gramercy Park: 20.00 minutes
- 67 Burger: 20.00 minutes
- Frank Restaurant: 20.00 minutes
- DespaIta: 20.50 minutes
- Sarabeth's West: 21.00 minutes

- The average delivery time compare across different restaurants?

```
# Calculate the average delivery time by restaurant name
avg_delivery_time = df.groupby('restaurant_name')['delivery_time'].mean().reset_index().sort_values(by='delivery_time')

# Print the average delivery time for each restaurant
print("Average Delivery Time by Restaurant:")
print(avg_delivery_time)

# Create the bar chart
fig = px.bar(avg_delivery_time, x='restaurant_name', y='delivery_time',
             title='Average Delivery Time by Restaurant',
             labels={'restaurant_name': 'Restaurant', 'delivery_time': 'Average Delivery Time (minutes)'},
             hover_data={'delivery_time': ':.2f'})

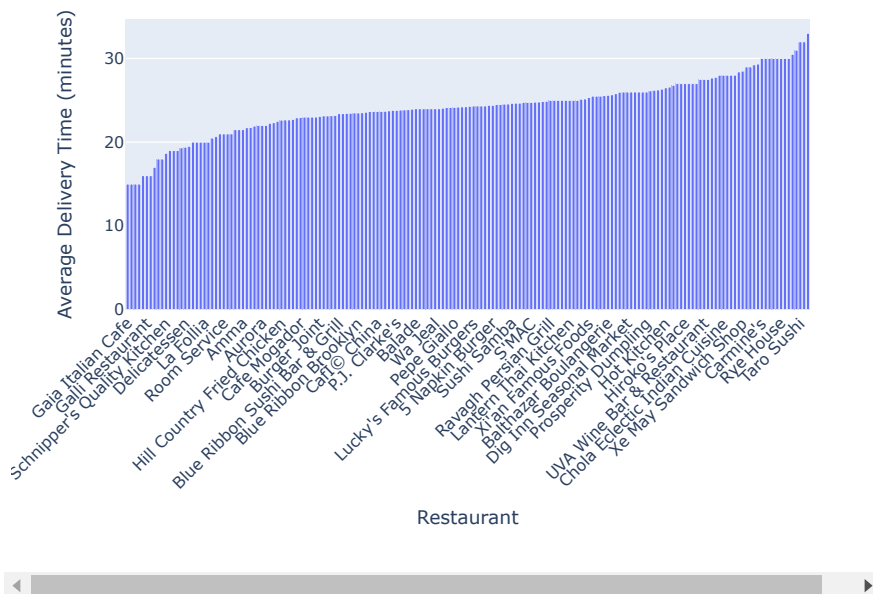
# Update the layout for better visualization
fig.update_layout(xaxis_tickangle=-45)

# Show the figure
fig.show()
```

```
➦ Average Delivery Time by Restaurant:
  restaurant_name  delivery_time
60    Gaia Italian Cafe         15.0
110   Paul & Jimmy's         15.0
152   The MasalaWala         15.0
71      Hibino              15.0
40   Coppola's East         16.0
..          ...            ...
64      Haandi             30.5
58   Frank Restaurant       31.0
148    Taro Sushi           32.0
68   Haru Gramercy Park     32.0
132  Sarabeth's West       33.0
```

[178 rows x 2 columns]

Average Delivery Time by Restaurant



Observations:

Restaurants have varied average delivery times, with some delivering in 15 minutes and others taking over 30 minutes.

- Checking is there a correlation between the cost of the order and the rating given

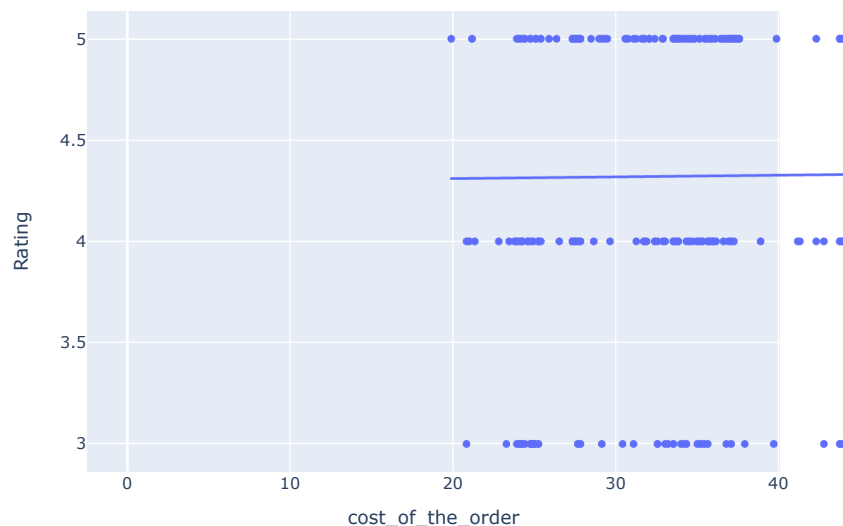
```
# Calculate the correlation between cost and rating
correlation = df[['cost_of_the_order', 'rating']].corr().iloc[0, 1]
print(f'Correlation between cost of the order and the rating given: {correlation:.2f}')

# Create a scatter plot with a trendline
fig = px.scatter(df, x='cost_of_the_order', y='rating', trendline='ols',
                 title=f'Correlation Between Cost of Order and Rating (Correlation: {correlation:.2f})',
                 labels={'cost': 'Cost of Order', 'rating': 'Rating'})

# Show the figure
fig.show()
```

Correlation between cost of the order and the rating given: 0.03

Correlation Between Cost of Order and Rating (Correlation: 0.03)



Observations:

The scatter plot indicates a very weak positive correlation (0.03) between the cost of the order and the rating given, suggesting minimal relationship.

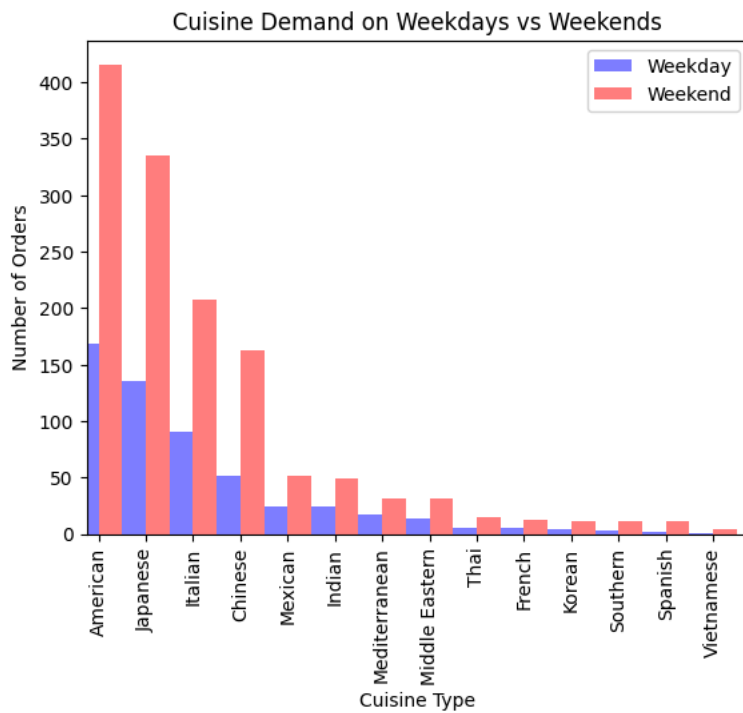
✓ 4. Demand Patterns

✓ • Howing the demand for different cuisine types vary on weekdays versus weekends?

```
cuisine_demand_weekday = df[df['day_of_the_week'] == 'Weekday']['cuisine_type'].value_counts()
cuisine_demand_weekend = df[df['day_of_the_week'] == 'Weekend']['cuisine_type'].value_counts()

"""print("Cuisine demand on weekdays:")
print(cuisine_demand_weekday)
print("\nCuisine demand on weekends:")
print(cuisine_demand_weekend)"""

# Visualization
cuisine_demand_weekday.plot(kind='bar', alpha=0.5, color='blue', position=1, label='Weekday')
cuisine_demand_weekend.plot(kind='bar', alpha=0.5, color='red', position=0, label='Weekend')
plt.legend()
plt.title('Cuisine Demand on Weekdays vs Weekends')
plt.xlabel('Cuisine Type')
plt.ylabel('Number of Orders')
plt.show()
```



Observations:

The bar chart reveals that demand for all cuisine types is generally higher on weekends compared to weekdays, with American and Japanese cuisines being the most popular.

• Which day of the week has the highest average order cost?

```
avg_cost_per_day = df.groupby('day_of_the_week')['cost_of_the_order'].mean()
highest_avg_cost_day = avg_cost_per_day.idxmax()
highest_avg_cost = avg_cost_per_day.max()
print(f'Day with the highest average order cost: {highest_avg_cost_day} (${highest_avg_cost:.2f})')
```

Day with the highest average order cost: Weekend (\$16.58)

Observations:

Day with the highest average order cost: **Weekend \$16.58**

• What is the most common day for orders to be placed?

```
most_common_order_day = df['day_of_the_week'].value_counts().idxmax()
print(f'Most common day for orders: {most_common_order_day}')
```

Most common day for orders: Weekend

Observations:

Most common day for orders: **Weekend**

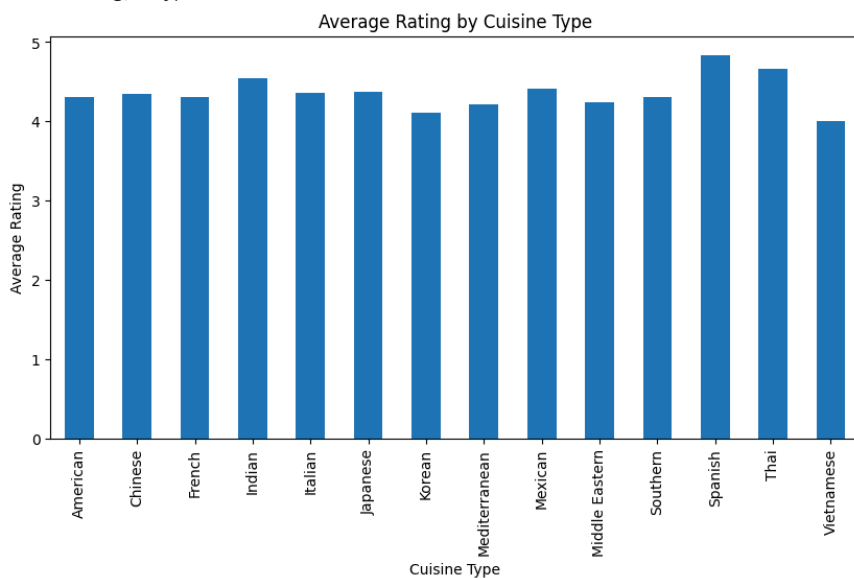
• The average rating vary by cuisine type?

```
avg_rating_per_cuisine = df[df['rating'] != 'Not given'].groupby('cuisine_type')['rating'].mean()
print(avg_rating_per_cuisine)
```

```
# Visualization
avg_rating_per_cuisine.plot(kind='bar', figsize=(10, 5), title='Average Rating by Cuisine Type')
plt.xlabel('Cuisine Type')
plt.ylabel('Average Rating')
plt.show()
```



```
cuisine_type
American      4.300813
Chinese       4.338346
French        4.300000
Indian        4.540000
Italian       4.360465
Japanese      4.373626
Korean        4.111111
Mediterranean 4.218750
Mexican       4.404255
Middle Eastern 4.235294
Southern      4.307692
Spanish       4.833333
Thai          4.666667
Vietnamese    4.000000
Name: rating, dtype: float64
```



Observations:

Average ratings are generally high across all cuisine types, with Spanish and Indian cuisines receiving the highest average ratings, while Vietnamese cuisine has the lowest.

✓ 5. Operational Efficiency

✓ • Average Delivery Time for All Orders

```
average_delivery_time = df['delivery_time'].mean()
print(f'Average delivery time for all orders: {average_delivery_time:.2f} minutes')
```

```
✓ Average delivery time for all orders: 24.16 minutes
```

Observations:

Average delivery time for all orders: 24.16 minutes

✓ • Restaurant with Longest Average Delivery Time

```
# Calculate average delivery time per restaurant
avg_delivery_time_per_restaurant = df.groupby('restaurant_name')['delivery_time'].mean()

# Now you can proceed with the rest of your code
longest_delivery_time_restaurant = avg_delivery_time_per_restaurant.idxmax()
longest_delivery_time = avg_delivery_time_per_restaurant.max()
print(f'Restaurant with the longest average delivery time: {longest_delivery_time_restaurant} ({longest_delivery_time:.2f} minutes)')

↗ Restaurant with the longest average delivery time: Sarabeth's West (33.00 minutes)
```

Observations:

The restaurant with the longest average delivery time is Sarabeth's West, taking an average of 33.00 minutes for delivery.

• Relationship Between Food Preparation Time and Delivery Time

```
prep_delivery_correlation = df[['food_preparation_time', 'delivery_time']].corr().iloc[0, 1]
print(f'Relationship between food preparation time and delivery time: {prep_delivery_correlation:.2f}')

# Compute correlation
prep_delivery_correlation = df[['food_preparation_time', 'delivery_time']].corr().iloc[0, 1]

# Print correlation
print(f'Relationship between food preparation time and delivery time: {prep_delivery_correlation:.2f}')

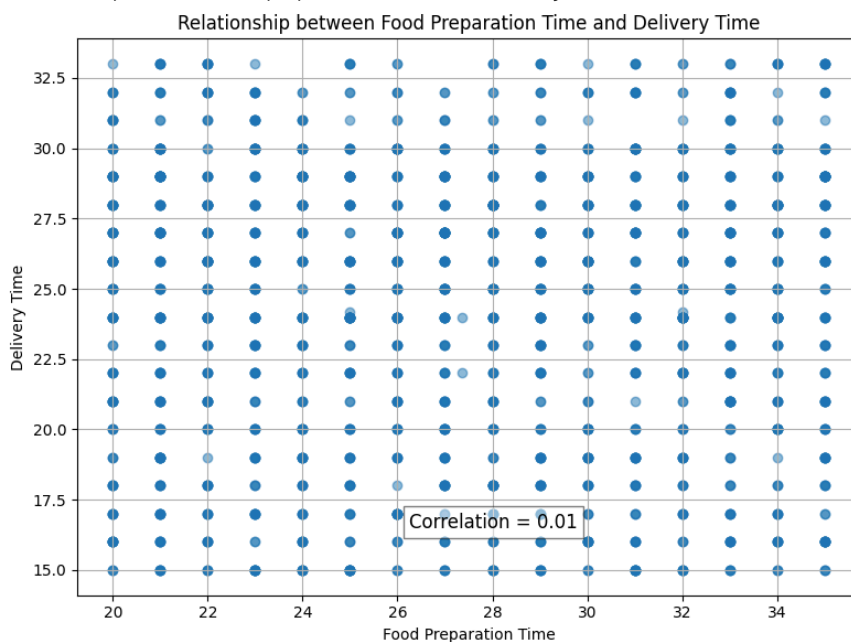
# Visualize the relationship with a scatter plot
plt.figure(figsize=(8, 6))
plt.scatter(df['food_preparation_time'], df['delivery_time'], alpha=0.5)
plt.title('Relationship between Food Preparation Time and Delivery Time')
plt.xlabel('Food Preparation Time')
plt.ylabel('Delivery Time')
plt.grid(True)
plt.tight_layout()

# Optionally, show correlation coefficient on the plot
plt.text(df['food_preparation_time'].max() * 0.75, df['delivery_time'].min() * 1.1,
         f'Correlation = {prep_delivery_correlation:.2f}', fontsize=12, bbox=dict(facecolor='white', alpha=0.5))

plt.show()
```

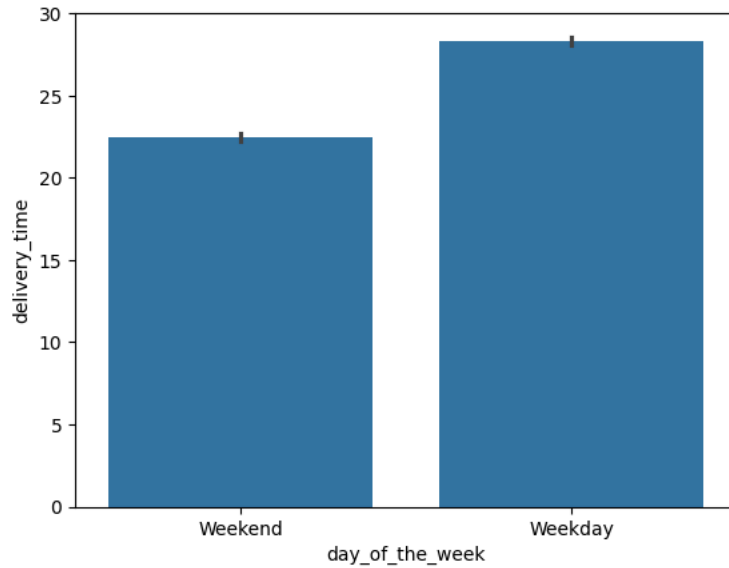
#delivery time and food preparation - to mean

```
↗ Relationship between food preparation time and delivery time: 0.01
Relationship between food preparation time and delivery time: 0.01
```



```
sns.barplot(data = df, x = 'day_of_the_week', y = 'delivery_time')
df.groupby(['day_of_the_week'])['delivery_time'].mean()
```

```
day_of_the_week
Weekday    28.340334
Weekend    22.470883
Name: delivery_time, dtype: float64
```




Observations:

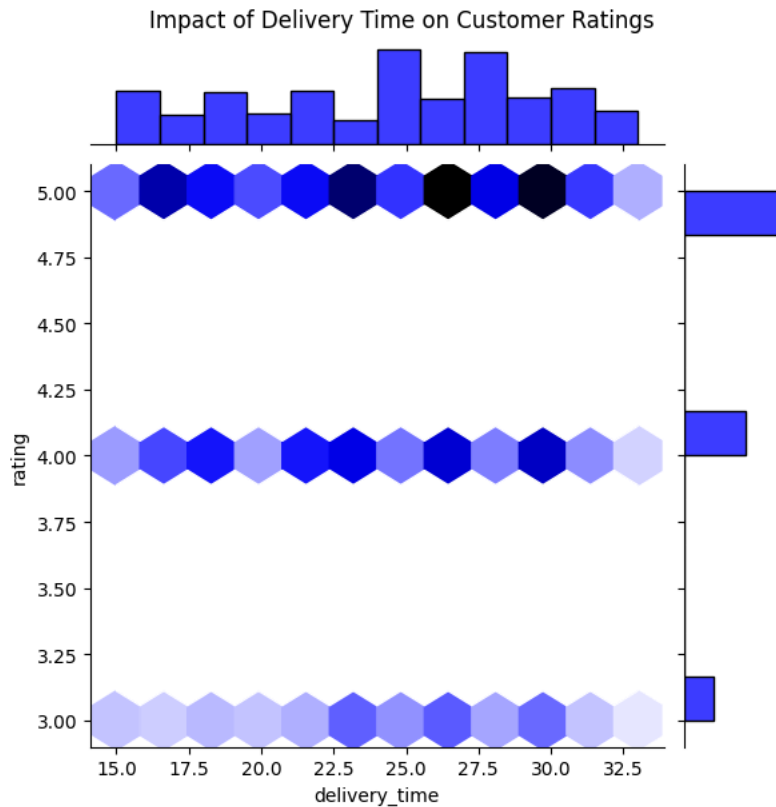
On average delivery times are higher during the weekday than on the weekend.

1. Weekday - 28 minutes
2. Weekend - 22 minutes

• Impact of Delivery Time on Customer Ratings

```
"""plt.figure(figsize=(10, 8))
sns.jointplot(data=df, x='delivery_time', y='rating', kind='reg')
plt.suptitle('Impact of Delivery Time on Customer Ratings', y=1.02)
plt.show()"""
plt.figure(figsize=(10, 8))
sns.jointplot(data=df, x='delivery_time', y='rating', kind='hex', color='b')
plt.suptitle('Impact of Delivery Time on Customer Ratings', y=1.02)
plt.show()
```

 <Figure size 1000x800 with 0 Axes>



Observations:

Customer ratings are evenly spread across various delivery times, indicating that delivery time has little to no effect on the ratings given.

6. Customer Insights

• Repeat Order Rate

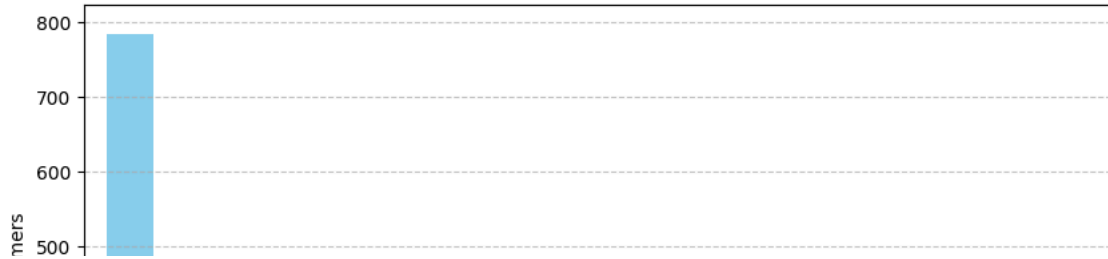
```
import matplotlib.pyplot as plt

# Calculate repeat customer counts
repeat_customers = df['customer_id'].value_counts()

# Plotting
plt.figure(figsize=(10, 6))
repeat_customers.value_counts().sort_index().plot(kind='bar', color='skyblue')
plt.title('Distribution of Repeat Orders per Customer')
plt.xlabel('Number of Repeat Orders')
plt.ylabel('Number of Customers')
plt.xticks(rotation=0)
plt.grid(axis='y', linestyle='--', alpha=0.7)
plt.show()
```



Distribution of Repeat Orders per Customer



```
# Get top 5 customers by customer_id
top_5_customers = repeat_customers.head(5)
```

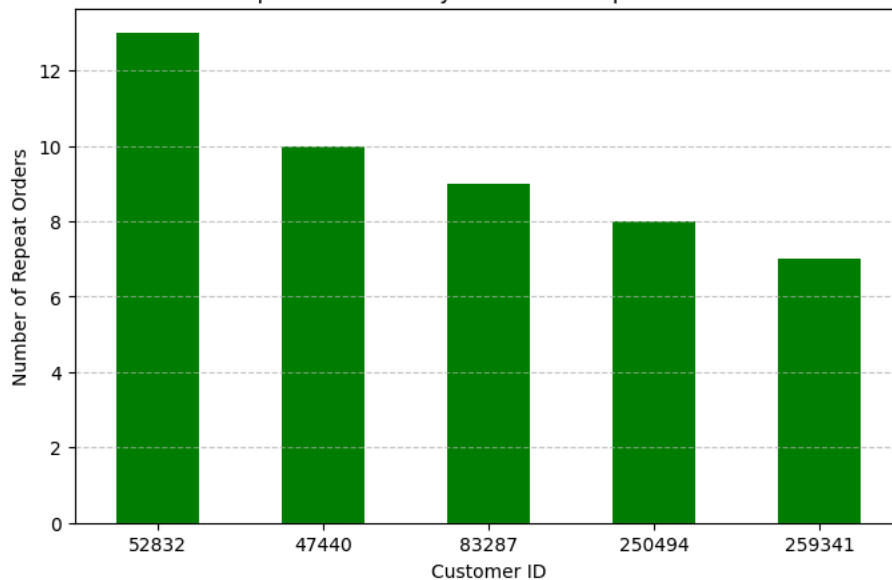
```
# Displaying customer_id of the top 5
print("Top 5 Customers by customer_id:")
print(top_5_customers)
```

```
# Optionally, visualize the top 5 customers
plt.figure(figsize=(8, 5))
top_5_customers.plot(kind='bar', color='green')
plt.title('Top 5 Customers by Number of Repeat Orders')
plt.xlabel('Customer ID')
plt.ylabel('Number of Repeat Orders')
plt.xticks(rotation=0)
plt.grid(axis='y', linestyle='--', alpha=0.7)
plt.show()
```



```
Top 5 Customers by customer_id:
customer_id
52832      13
47440      10
83287       9
250494       8
259341       7
Name: count, dtype: int64
```

Top 5 Customers by Number of Repeat Orders



Observations: