# PORTER DATA ANALYSIS

```
from IPython.display import Image
Image(filename="M:\Porter Case Study\Images\Porter.png",width=600)
```

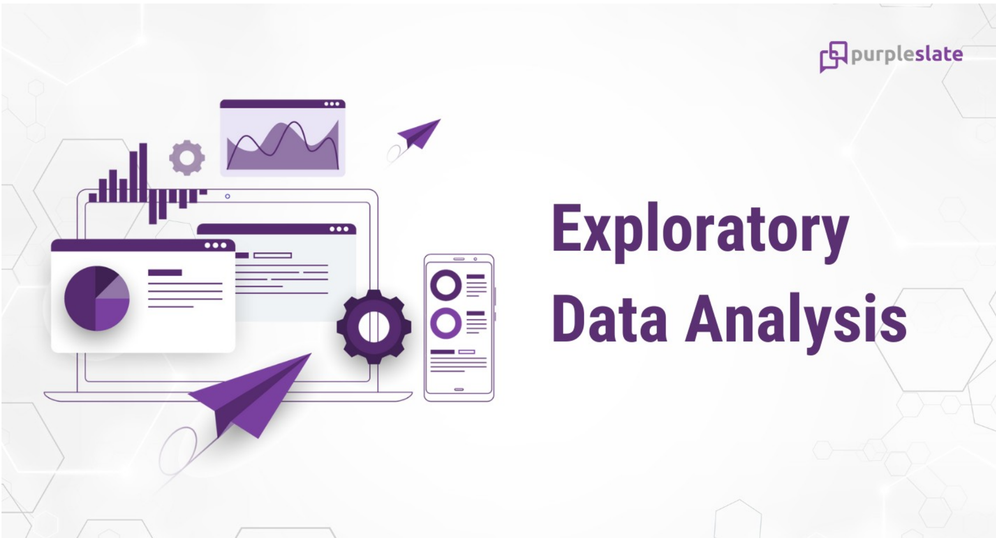
PORTER

## Important Library for Analysis

```
import pandas as pd
import numpy as np
import matplotlib.pyplot as plt
import seaborn as sns
import warnings
warnings.filterwarnings("ignore")
%matplotlib inline
from scipy.stats import zscore

from IPython.display import Image
Image(filename="M:\Porter Case Study\Images\
ProblemStatment.jpeg",width=300)
```

Porter, India's largest marketplace for intra-city logistics, works with a wide range of restaurants to deliver their items directly to customers. The company wants to estimate the delivery time for each order based on various features, such as the items ordered, the restaurant, and the availability of delivery partners. An accurate estimation of delivery time will enhance customer satisfaction and optimize the delivery process.

```
from IPython.display import Image
Image(filename="M:\Porter Case Study\Images\EDA.png",width=500)
```




## Import the datasets

```
df=pd.read_csv(r"M:\Porter Case Study\dataset.csv")

#Top 5 Rows
df.head()
```

```
   market_id           created_at actual_delivery_time  \
0        1.0  2015-02-06 22:24:17  2015-02-06 23:27:16
1        2.0  2015-02-10 21:49:25  2015-02-10 22:56:29
2        3.0  2015-01-22 20:39:28  2015-01-22 21:09:09
3        3.0  2015-02-03 21:21:45  2015-02-03 22:13:00
4        3.0  2015-02-15 02:40:36  2015-02-15 03:20:26

                           store_id store_primary_category
order_protocol  \
0  df263d996281d984952c07998dc54358               american
1.0
1  f0ade77b43923b38237db569b016ba25                mexican
2.0
2  f0ade77b43923b38237db569b016ba25                    NaN
1.0
3  f0ade77b43923b38237db569b016ba25                    NaN
```

```
1.0
4  f0ade77b43923b38237db569b016ba25                    NaN
1.0

   total_items  subtotal  num_distinct_items  min_item_price
max_item_price  \
0            4      3441                   4             557
1239
1            1      1900                   1            1400
1400
2            1      1900                   1            1900
1900
3            6      6900                   5             600
1800
4            3      3900                   3            1100
1600

   total_onshift_partners  total_busy_partners
total_outstanding_orders
0                    33.0                 14.0
21.0
1                     1.0                  2.0
2.0
2                     1.0                  0.0
0.0
3                     1.0                  1.0
2.0
4                     6.0                  6.0
9.0

#Bottom 5 Rows
df.tail()

        market_id           created_at actual_delivery_time  \
197423        1.0  2015-02-17 00:19:41  2015-02-17 01:24:48
197424        1.0  2015-02-13 00:01:59  2015-02-13 00:58:22
197425        1.0  2015-01-24 04:46:08  2015-01-24 05:36:16
197426        1.0  2015-02-01 18:18:15  2015-02-01 19:23:22
197427        1.0  2015-02-08 19:24:33  2015-02-08 20:01:41

                                store_id store_primary_category  \
197423  a914ecef9c12ffdb9bede64bb703d877                   fast
197424  a914ecef9c12ffdb9bede64bb703d877                   fast
197425  a914ecef9c12ffdb9bede64bb703d877                   fast
197426  c81e155d85dae5430a8cee6f2242e82c               sandwich
197427  c81e155d85dae5430a8cee6f2242e82c               sandwich

        order_protocol  total_items  subtotal  num_distinct_items  \
197423             4.0            3      1389                   3
197424             4.0            6      3010                   4
```

```
197425             4.0             5      1836                   3
197426             1.0             1      1175                   1
197427             1.0             4      2605                   4

        min_item_price  max_item_price  total_onshift_partners  \
197423             345             649                    17.0
197424             405             825                    12.0
197425             300             399                    39.0
197426             535             535                     7.0
197427             425             750                    20.0

        total_busy_partners  total_outstanding_orders
197423                 17.0                      23.0
197424                 11.0                      14.0
197425                 41.0                      40.0
197426                  7.0                      12.0
197427                 20.0                      23.0
```

```
# Statistical view of datasets of numerical data
df.describe()
```

```
           market_id  order_protocol    total_items       subtotal  \
count  196441.000000   196433.000000  197428.000000  197428.000000
mean        2.978706        2.882352       3.196391    2682.331402
std         1.524867        1.503771       2.666546    1823.093688
min         1.000000        1.000000       1.000000       0.000000
25%         2.000000        1.000000       2.000000    1400.000000
50%         3.000000        3.000000       3.000000    2200.000000
75%         4.000000        4.000000       4.000000    3395.000000
max         6.000000        7.000000     411.000000   27100.000000

       num_distinct_items  min_item_price  max_item_price  \
count       197428.000000   197428.000000   197428.000000
mean             2.670791      686.218470     1159.588630
std              1.630255      522.038648      558.411377
min              1.000000      -86.000000        0.000000
25%              1.000000      299.000000      800.000000
50%              2.000000      595.000000     1095.000000
75%              3.000000      949.000000     1395.000000
max             20.000000    14700.000000    14700.000000

       total_onshift_partners  total_busy_partners
total_outstanding_orders
count           181166.000000        181166.000000
181166.000000
mean                44.808093            41.739747
58.050065
std                 34.526783            32.145733
52.661830
min                 -4.000000            -5.000000                    -
```

```
6.000000
25%                   17.000000            15.000000
17.000000
50%                   37.000000            34.000000
41.000000
75%                   65.000000            62.000000
85.000000
max                  171.000000           154.000000
285.000000
```

```python
# Statistical view of datasets of categorical data
df.describe(include=object)
```

```
                 created_at actual_delivery_time  \
count                197428               197421
unique               180985               178110
top     2015-02-11 19:50:43  2015-02-11 20:40:45
freq                      6                    5

                                store_id store_primary_category
count                             197428                 192668
unique                              6743                     74
top     d43ab110ab2489d6b9b2caa394bf920f               american
freq                                 937                  19399
```

## Understanding Data Structure

```python
df.dtypes
```

```
market_id                   float64
created_at                   object
actual_delivery_time         object
store_id                     object
store_primary_category       object
order_protocol              float64
total_items                   int64
subtotal                      int64
num_distinct_items            int64
min_item_price                int64
max_item_price                int64
total_onshift_partners      float64
total_busy_partners         float64
total_outstanding_orders    float64
dtype: object
```

```python
df.info()
```

```
<class 'pandas.core.frame.DataFrame'>
RangeIndex: 197428 entries, 0 to 197427
Data columns (total 14 columns):
```

```
 #   Column                      Non-Null Count   Dtype
---  ------                      --------------   -----
 0   market_id                   196441 non-null  float64
 1   created_at                  197428 non-null  object
 2   actual_delivery_time        197421 non-null  object
 3   store_id                    197428 non-null  object
 4   store_primary_category      192668 non-null  object
 5   order_protocol              196433 non-null  float64
 6   total_items                 197428 non-null  int64
 7   subtotal                    197428 non-null  int64
 8   num_distinct_items          197428 non-null  int64
 9   min_item_price              197428 non-null  int64
 10  max_item_price              197428 non-null  int64
 11  total_onshift_partners      181166 non-null  float64
 12  total_busy_partners         181166 non-null  float64
 13  total_outstanding_orders    181166 non-null  float64
dtypes: float64(5), int64(5), object(4)
memory usage: 21.1+ MB
```

```
#Checking null values
df.isnull().sum()
```

```
market_id                     987
created_at                      0
actual_delivery_time            7
store_id                        0
store_primary_category       4760
order_protocol                995
total_items                     0
subtotal                        0
num_distinct_items              0
min_item_price                  0
max_item_price                  0
total_onshift_partners      16262
total_busy_partners         16262
total_outstanding_orders    16262
dtype: int64
```

```
Image(filename="M:\Porter Case Study\Images\DataProcessing and featureEngg.webp",width=700)
```


We got missing values in multiple columns we have to impute those

```
df['market_id'].value_counts()

market_id
2.0    55058
4.0    47599
1.0    38037
3.0    23297
5.0    18000
6.0    14450
Name: count, dtype: int64
```

Since our market_id has the 6 unique market id and the missing value in the 987 rows which are very less if we compare it with our whole datasets where all 197428 so we can use the random method to impute the missing values.

```
nonnull_market_id=df.market_id.dropna().values
df['market_id']=df['market_id'].apply(lambda
x:np.random.choice(nonnull_market_id) if pd.isnull(x) else x)

# we imputed missing values in the 'market_id' column
df['market_id'].isnull().sum()

0
```

```python
df['actual_delivery_time'].isnull().sum()
```

```
7
```

imputting missing values in the "actual_delivery_time" column Here we have only 7 rows containing null values in the 'actual_delivery_time' column and also the data is in TimeSeries and Continuous so we can impute it with ffill or bfill.

```python
df['actual_delivery_time']=df['actual_delivery_time'].ffill()

df['actual_delivery_time'].isnull().sum()
```

```
0
```

```python
df['store_primary_category'].isnull().sum()
```

```
4760
```

```python
df['store_primary_category'].value_counts()
```

```
store_primary_category
american             19399
pizza                17321
mexican              17099
burger               10958
sandwich             10060
                     ...
lebanese                 9
belgian                  2
indonesian               2
chocolate                1
alcohol-plus-food        1
Name: count, Length: 74, dtype: int64
```

Imputting missing values in the 'store_primary_category' column. Since our store_primary_category column has 4760 missing rows and the 74 unique we compare it with our whole database i.e, the 197428 rows and 14 columns these missing values are very few so we can impute it with the high occurence of the data i.e, the mode() method.

```python
mode_value=df['store_primary_category'].mode()[0]
df['store_primary_category'].fillna(mode_value,inplace=True)

df['store_primary_category'].isnull().sum()
```

```
0
```

```python
# Imputting missing values in 'order_protocol'Column
df['order_protocol'].value_counts()
```

```
order_protocol
1.0    54725
3.0    53199
5.0    44290
2.0    24052
4.0    19354
6.0      794
7.0       19
Name: count, dtype: int64
```

```
df['order_protocol'].isnull().sum()
```

```
995
```

Order_protocol is has 995 missing values so we can impute with ffill or bfill

```
df['order_protocol']=df['order_protocol'].ffill()
```

```
df['order_protocol'].isnull().sum()
```

```
0
```

```
# Imputting missing values in
total_onshift_partners,total_outstanding_orders & total_busy_partners
```

```
print("Total null values in total_onshift_partners
column:",df.total_onshift_partners.isnull().sum())
print("-------------------------------------------------------")
print(df.total_onshift_partners.value_counts())
```

```
Total null values in total_onshift_partners column: 16262
-------------------------------------------------------
total_onshift_partners
 0.0      3615
 18.0     2924
 15.0     2912
 21.0     2841
 19.0     2824
          ...
 164.0       1
 159.0       1
 169.0       1
-4.0         1
 168.0       1
Name: count, Length: 172, dtype: int64
```

```
print("Total null values in total_outstanding_order
column:",df.total_outstanding_orders.isnull().sum())
print("-------------------------------------------------------")
print(df.total_outstanding_orders.value_counts())
```

```
Total null values in total_outstanding_order column: 16262
-------------------------------------------------------
total_outstanding_orders
0.0      4111
9.0      2744
10.0     2705
8.0      2685
6.0      2672
         ...
268.0       1
264.0       1
277.0       1
265.0       1
260.0       1
Name: count, Length: 281, dtype: int64
```

```
print("Total null values in total_busy_partners
column:",df.total_busy_partners.isnull().sum())
print("-------------------------------------------------------")
print(df.total_busy_partners.value_counts())
```

```
Total null values in total_busy_partners column: 16262
-------------------------------------------------------
total_busy_partners
 0.0      4171
 10.0     3114
 13.0     3052
 6.0      3040
 18.0     3001
          ...
 152.0       2
 153.0       1
 154.0       1
 149.0       1
-5.0         1
Name: count, Length: 159, dtype: int64
```

```
#Statistical View of the columns
df[["total_onshift_partners","total_outstanding_orders","total_busy_pa
rtners"]].describe()
```

```
       total_onshift_partners  total_outstanding_orders
total_busy_partners
count           181166.000000             181166.000000
181166.000000
mean                44.808093                 58.050065
41.739747
std                 34.526783                 52.661830
32.145733
min                 -4.000000                 -6.000000            -
```

```
5.000000
25%                    17.000000                  17.000000
15.000000
50%                    37.000000                  41.000000
34.000000
75%                    65.000000                  85.000000
62.000000
max                   171.000000                 285.000000
154.000000
```

Since our above all three columns have the 16262 rows contain the null values and also these are the numerical columns and rows containing values like

By the above Statical view Observation we can say that several similarites between the columns like each column has 181166 entries, mean vary between the 41-58, standard deviation also vary between 32-52, min vary between -4 to -6, max vary between 154 to 285 and the Quartile range also the same. So We Can Impute missing values all three columns with same method i.e, the random method.

```
def random_sampling(col):
    non_null=col.dropna().values
    return col.apply(lambda x:np.random.choice(non_null) if
pd.isnull(x) else x)

df['total_busy_partners']=random_sampling(df['total_busy_partners'])
df['total_onshift_partners']=random_sampling(df['total_onshift_partner
s'])
df['total_outstanding_orders']=random_sampling(df['total_outstanding_o
rders'])


# we imputed the all null values from our datasets

df['total_busy_partners'].isnull().sum()

0

df['total_onshift_partners'].isnull().sum()

0

df['total_outstanding_orders'].isnull().sum()

0

df.isnull().sum()

market_id                   0
created_at                  0
actual_delivery_time        0
store_id                    0
```

```
store_primary_category      0
order_protocol              0
total_items                 0
subtotal                    0
num_distinct_items          0
min_item_price              0
max_item_price              0
total_onshift_partners      0
total_busy_partners         0
total_outstanding_orders    0
dtype: int64
```

Creating new features "TimeTakenForDelivery" with help of "Created_at" and "actual_delivery_time" columns

```
df['created_at']=pd.to_datetime(df.created_at,errors='coerce')
df['actual_delivery_time']=pd.to_datetime(df.actual_delivery_time,erro
rs='coerce')

df['time_taken_for_delivery']=df['actual_delivery_time']-
df['created_at']
df['time_taken_for_delivery']

0        0 days 01:02:59
1        0 days 01:07:04
2        0 days 00:29:41
3        0 days 00:51:15
4        0 days 00:39:50
               ...
197423   0 days 01:05:07
197424   0 days 00:56:23
197425   0 days 00:50:08
197426   0 days 01:05:07
197427   0 days 00:37:08
Name: time_taken_for_delivery, Length: 197428, dtype: timedelta64[ns]

df['time_taken_for_delivery'].describe()

count                         197428
mean     0 days 00:47:50.302201308
std      0 days 05:41:18.561739175
min             -23 days +04:12:56
25%                0 days 00:35:04
50%                0 days 00:44:20
75%                0 days 00:56:21
max               98 days 13:47:39
Name: time_taken_for_delivery, dtype: object
```

By the statistical view of "TimeTakenForDelivery" column observation are minimum time to delivered to any product is -23 days and maximum time is 98 days. So it is impossible to get time in negative. so there has to be some outliers in our column. Need to remove that.

```
# We are removing the ouliers with Quarantile range
df['time_taken_for_delivery_seconds']=df['time_taken_for_delivery'].dt
.total_seconds()
Q1=df['time_taken_for_delivery_seconds'].quantile(0.25)
Q3=df['time_taken_for_delivery_seconds'].quantile(0.75)

IQR=Q3-Q1
lb=Q1-1.5 *IQR
ub=Q3+1.5 *IQR

df['time_taken_for_delivery_seconds']=np.where(
    (df['time_taken_for_delivery_seconds']<lb)|
(df['time_taken_for_delivery_seconds']>ub),
    np.nan,df['time_taken_for_delivery_seconds']
)
df['time_taken_for_delivery']=pd.to_timedelta(df['time_taken_for_deliv
ery_seconds'],unit='s')

df['time_taken_for_delivery'].isnull().sum()

6285
```

After removing outliers we got 6285 missing values

Since our dataset is datetime format and best approach to handle missing in datetime is ffill or bfill

```
df['time_taken_for_delivery'].ffill(inplace=True)

df['time_taken_for_delivery'].isnull().sum()

0

df['time_taken_for_delivery_seconds'].ffill(inplace=True)

df['time_taken_for_delivery_seconds'].isnull().sum()

0

# Statical View of "TimeTakenForDelivery" Column.
df['time_taken_for_delivery'].describe()

count                       197428
mean     0 days 00:45:49.712872540
std      0 days 00:14:48.545796823
min                0 days 00:03:43
25%                0 days 00:34:51
```

```
50%                    0 days 00:43:51
75%                    0 days 00:55:02
max                    0 days 01:28:16
Name: time_taken_for_delivery, dtype: object
```

Observations are

Avg time taken to delivered per product is ~45 min

Minimum time taken to delivered per product is ~4 min

Maximum time taken to delivered per product is ~89 min

75% delivered ~55 min

## Creating new features "HourOfDay", "DayOfWeek" and "Month"

```
df['HourOfDay']=df['created_at'].dt.hour
df['DayOfWeek']=df['created_at'].dt.dayofweek
df['Month']=df['created_at'].dt.month
df['Week']=df['created_at'].dt.isocalendar().week
df['Year']=df['created_at'].dt.year

#Extracting new features like "TimeTakenforDeilvery_Minutes" and
"TimeTakenforDeilvery_Hours"
df['time_taken_for_delivery_minutes']=df['time_taken_for_delivery'].dt
.total_seconds()/60
df['time_taken_for_delivery_hours']=df['time_taken_for_delivery'].dt.t
otal_seconds()/3600
```

Extracting new features from "Delivery_categories" as how fast and slow are we able to deliver the products

```
df['Delivery_categories']=pd.cut(df['time_taken_for_delivery_minutes']
,
                                 bins=[3,30,40,60,90],

labels=['fast','Moderate','Slow','Very Slow'])

df['Delivery_categories'].value_counts()

Delivery_categories
Slow          85291
Moderate      51013
Very Slow     34582
fast          26542
Name: count, dtype: int64
```

How much time taken to delivered to item

```
df['DeliverySpeedperItem']=df['time_taken_for_delivery_minutes']/
df['total_items']
df['DeliverySpeedperItem'].value_counts()
```

```
DeliverySpeedperItem
16.200000    67
19.000000    65
16.050000    65
15.750000    61
14.350000    61
             ..
6.361538      1
3.334127      1
2.242857      1
1.338506      1
3.527778      1
Name: count, Length: 24743, dtype: int64
```

```
# Here we extracting AVGITEMPRICE
df['AvgItemPrice']=df['subtotal']/df['total_items']

df['AvgItemPrice'].value_counts()
```

```
AvgItemPrice
1095.000000    1508
995.000000     1421
895.000000     1364
795.000000     1313
1200.000000    1280
               ...
2664.500000       1
2011.666667       1
1352.666667       1
544.600000        1
367.200000        1
Name: count, Length: 18803, dtype: int64
```

# Encoding Categorical Columns

```
Categorical_df=df.select_dtypes(include='object')
Encoded_categorical_df=pd.get_dummies(Categorical_df,drop_first=True)

Encoded_categorical_df.info()
```

```
<class 'pandas.core.frame.DataFrame'>
RangeIndex: 197428 entries, 0 to 197427
Columns: 6815 entries, store_id_00053f5e11d1fe4e49a221165b39abc9 to
store_primary_category_vietnamese
```

```
dtypes: bool(6815)
memory usage: 1.3 GB

Encoded_categorical_df

        store_id_00053f5e11d1fe4e49a221165b39abc9  \
0                                           False
1                                           False
2                                           False
3                                           False
4                                           False
...                                           ...
197423                                      False
197424                                      False
197425                                      False
197426                                      False
197427                                      False

        store_id_0006aabe0ba47a35c0b0bf6596f85159  \
0                                           False
1                                           False
2                                           False
3                                           False
4                                           False
...                                           ...
197423                                      False
197424                                      False
197425                                      False
197426                                      False
197427                                      False

        store_id_000a91f3e374e6147d58ed1814247508  \
0                                           False
1                                           False
2                                           False
3                                           False
4                                           False
...                                           ...
197423                                      False
197424                                      False
197425                                      False
197426                                      False
197427                                      False

        store_id_0029f088c57ad3b6ec589f9ba4f7a057  \
0                                           False
1                                           False
2                                           False
3                                           False
4                                           False
```

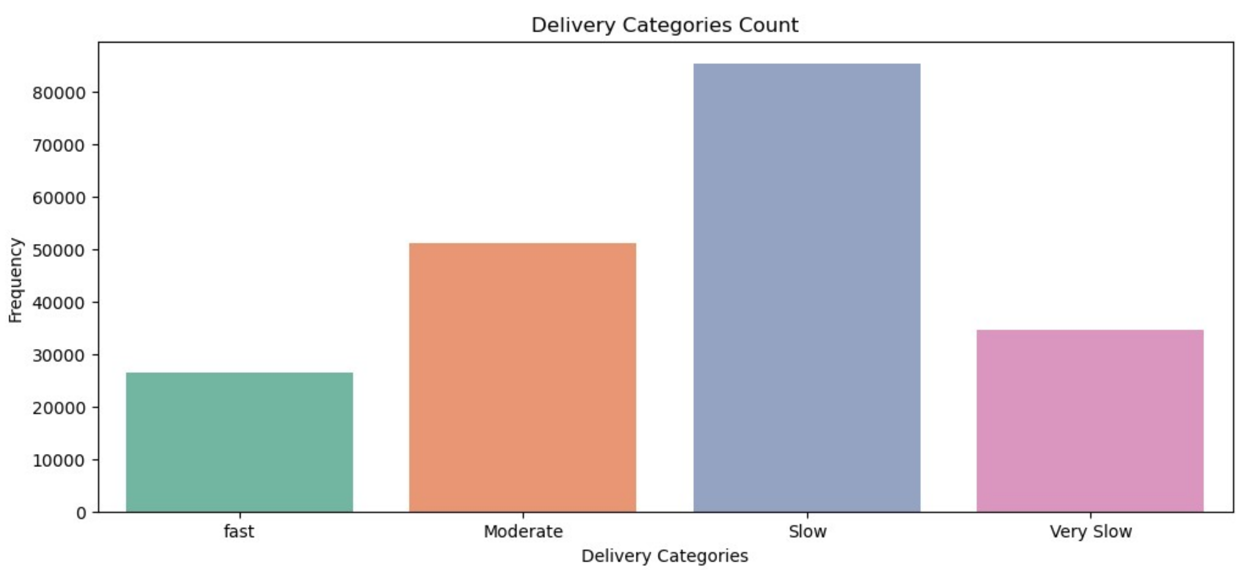
```
...                                                  ...
197423                                             False
197424                                             False
197425                                             False
197426                                             False
197427                                             False

        store_id_002f9c8cee878b64a747a2c211da7d83  \
0                                                  False
1                                                  False
2                                                  False
3                                                  False
4                                                  False
...                                                  ...
197423                                             False
197424                                             False
197425                                             False
197426                                             False
197427                                             False

        store_id_00430c0c1fae276c9713ab5f21167882  \
0                                                  False
1                                                  False
2                                                  False
3                                                  False
4                                                  False
...                                                  ...
197423                                             False
197424                                             False
197425                                             False
197426                                             False
197427                                             False

        store_id_0044deeec43ded19b952125079eb1781  \
0                                                  False
1                                                  False
2                                                  False
3                                                  False
4                                                  False
...                                                  ...
197423                                             False
197424                                             False
197425                                             False
197426                                             False
197427                                             False

        store_id_00482b9bed15a272730fcb590ffebddd  \
0                                                  False
1                                                  False
2                                                  False
```

```
3                                             False
4                                             False
...                                             ...
197423                                        False
197424                                        False
197425                                        False
197426                                        False
197427                                        False

        store_id_004a68efcee088ddeaaca5c5a3afaa2f  \
0                                             False
1                                             False
2                                             False
3                                             False
4                                             False
...                                             ...
197423                                        False
197424                                        False
197425                                        False
197426                                        False
197427                                        False

        store_id_005b0c27e7224dabb8c1c7346ceea228  ...  \
0                                             False  ...
1                                             False  ...
2                                             False  ...
3                                             False  ...
4                                             False  ...
...                                             ...  ...
197423                                        False  ...
197424                                        False  ...
197425                                        False  ...
197426                                        False  ...
197427                                        False  ...

        store_primary_category_southern
store_primary_category_spanish  \
0                                  False
False
1                                  False
False
2                                  False
False
3                                  False
False
4                                  False
False
...                                  ...                              ..
.
197423                             False
```

```
False
197424                                 False
False
197425                                 False
False
197426                                 False
False
197427                                 False
False

        store_primary_category_steak  store_primary_category_sushi  \
0                              False                         False
1                              False                         False
2                              False                         False
3                              False                         False
4                              False                         False
...                              ...                           ...
197423                         False                         False
197424                         False                         False
197425                         False                         False
197426                         False                         False
197427                         False                         False

        store_primary_category_tapas  store_primary_category_thai  \
0                              False                        False
1                              False                        False
2                              False                        False
3                              False                        False
4                              False                        False
...                              ...                          ...
197423                         False                        False
197424                         False                        False
197425                         False                        False
197426                         False                        False
197427                         False                        False

        store_primary_category_turkish
store_primary_category_vegan  \
0                                False                         False

1                                False                         False

2                                False                         False

3                                False                         False

4                                False                         False

...                                ...                           ...
```

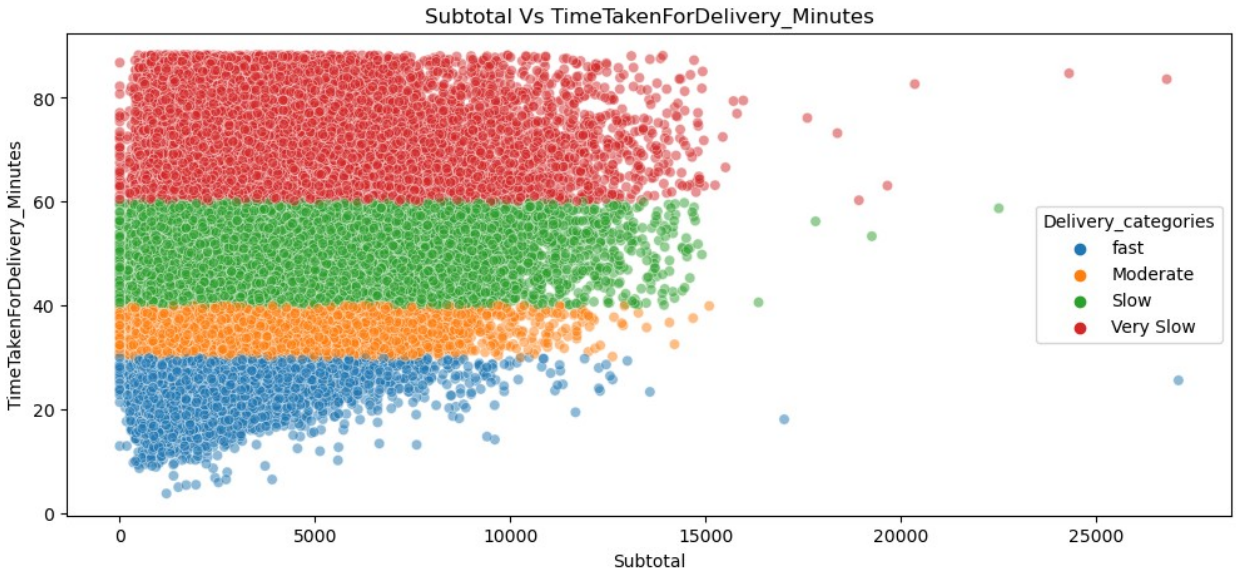
```
197423                                 False                              False

197424                                 False                              False

197425                                 False                              False

197426                                 False                              False

197427                                 False                              False


        store_primary_category_vegetarian
store_primary_category_vietnamese
0                                    False
False
1                                    False
False
2                                    False
False
3                                    False
False
4                                    False
False
...                                    ...
...
197423                               False
False
197424                               False
False
197425                               False
False
197426                               False
False
197427                               False
False

[197428 rows x 6815 columns]

Image(filename="M:\Porter Case Study\Images\DVAC.webp",width=400)
```

```
# How delivery categories is distributed
plt.figure(figsize=(12,5))

sns.countplot(x='Delivery_categories',data=df,palette='Set2')
plt.title('Delivery Categories Count')
plt.xlabel('Delivery Categories')
plt.ylabel('Frequency')
plt.show()
```

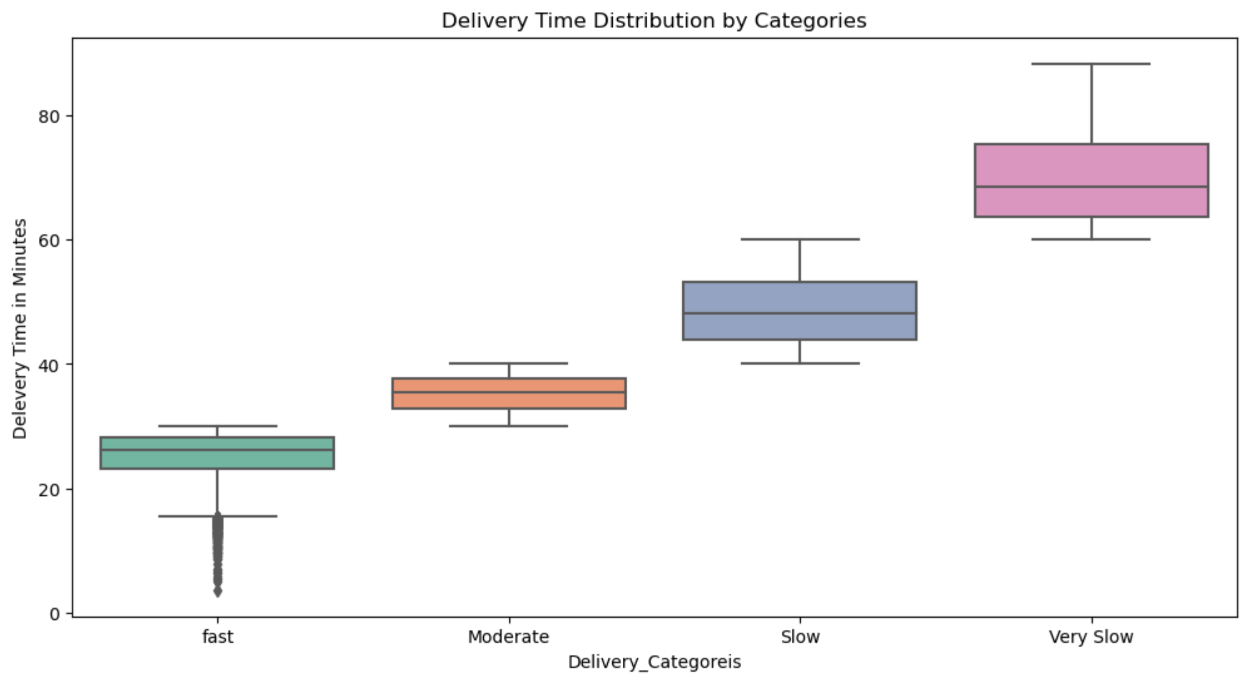



```
df['subtotal']=pd.to_numeric(df['subtotal'],errors='coerce')
df['time_taken_for_delivery_minutes']=pd.to_numeric(df['time_taken_for_delivery_minutes'],errors='coerce')

plt.figure(figsize=(12,5))
```

```
sns.scatterplot(data=df,x='subtotal',y='time_taken_for_delivery_minute
s',hue='Delivery_categories',alpha=0.5)
plt.title("Subtotal Vs TimeTakenForDelivery_Minutes")
plt.xlabel("Subtotal")
plt.ylabel("TimeTakenForDelivery_Minutes")
plt.show()
```




```
plt.figure(figsize=(12,6))

sns.boxplot(data=df,x='Delivery_categories',y='time_taken_for_delivery
_minutes',palette='Set2')
plt.title("Delivery Time Distribution by Categories")
plt.xlabel("Delivery_Categoreis")
plt.ylabel("Delevery Time in Minutes")
plt.show()
```




```
df['time_taken_for_delivery_minutes'].describe()

count    197428.000000
mean         45.828548
std          14.809097
min           3.716667
25%          34.850000
50%          43.850000
75%          55.033333
max          88.266667
Name: time_taken_for_delivery_minutes, dtype: float64
```

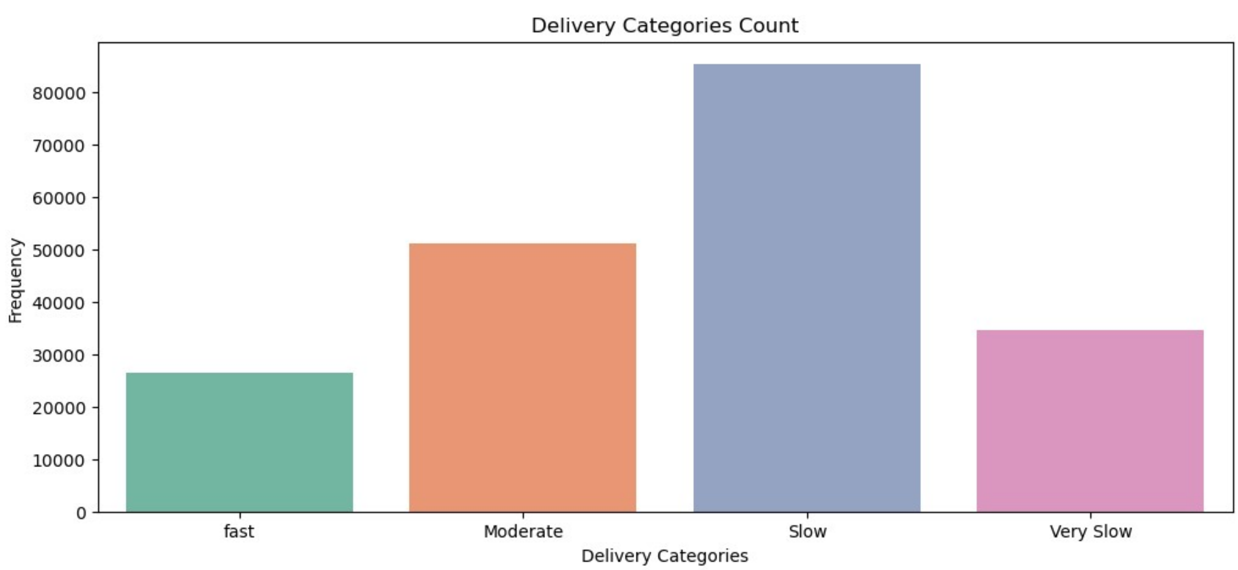
After the Analysis scatterplot and boxplot we can say that the some of the outliers in 'time_taken_for_delivery_minutes' Column and also statiscal view indicates

```
#Removing outliers
Q1=df['time_taken_for_delivery_minutes'].quantile(0.25)
Q3=df['time_taken_for_delivery_minutes'].quantile(0.75)

IQR=Q3-Q1
lb=Q1-1.5 * IQR
ub=Q3+1.5 * IQR

df['time_taken_for_delivery_minutes']=np.where(
    (df['time_taken_for_delivery_minutes']<lb) |
(df['time_taken_for_delivery_minutes']>ub),
    np.nan,df['time_taken_for_delivery_minutes'])
```

```
df['time_taken_for_delivery_minutes']=pd.to_timedelta(df['time_taken_for_delivery_minutes'],unit='m')

df['time_taken_for_delivery_minutes'].isnull().sum()
# Since we removed outliers from the colmns that's the reason we receving missing values in 1335 columns we have to impute it.

1335

# Since we containing time series data we impute missing values with ffill or bfill
df['time_taken_for_delivery_minutes'].bfill(inplace=True)

df['time_taken_for_delivery_minutes'].isnull().sum()

0
```

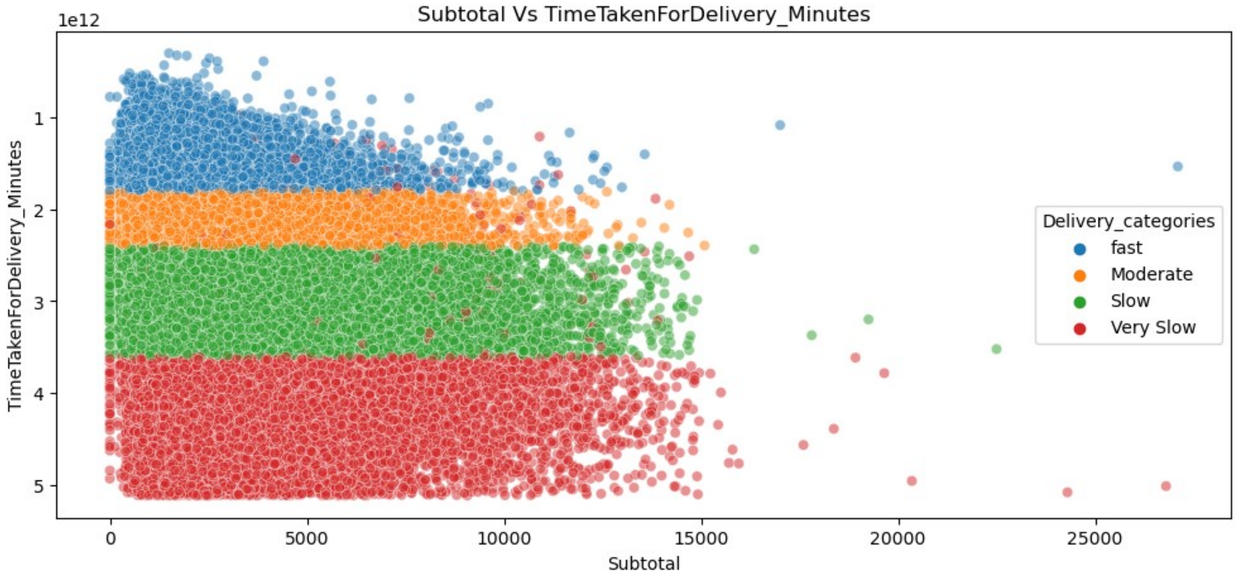
## Plotting Graph Again

```
plt.figure(figsize=(12,5))

sns.countplot(x='Delivery_categories',data=df,palette='Set2')
plt.title('Delivery Categories Count')
plt.xlabel('Delivery Categories')
plt.ylabel('Frequency')
plt.show()
```




```
plt.figure(figsize=(12,5))

sns.scatterplot(data=df,x='subtotal',y='time_taken_for_delivery_minutes',hue='Delivery_categories',alpha=0.5)
plt.title("Subtotal Vs TimeTakenForDelivery_Minutes")
plt.xlabel("Subtotal")
```

```
plt.ylabel("TimeTakenForDelivery_Minutes")
plt.show()
```




```
df['TimeTakenForDelivery_Minutes_Numeric'] =
df['time_taken_for_delivery_minutes'].dt.total_seconds() / 60

plt.figure(figsize=(10, 5))
sns.boxplot(
    x="Delivery_categories",
    y="TimeTakenForDelivery_Minutes_Numeric",
    data=df,
    palette="twilight"
)
plt.title("Delivery Categories VS Time Taken for Delivery (Minutes)")
plt.xlabel("Delivery Categories")
plt.ylabel("Time Taken for Delivery (Minutes)")
plt.xticks(rotation=45)
plt.show()
```


```
df['order_protocol'].value_counts()
```

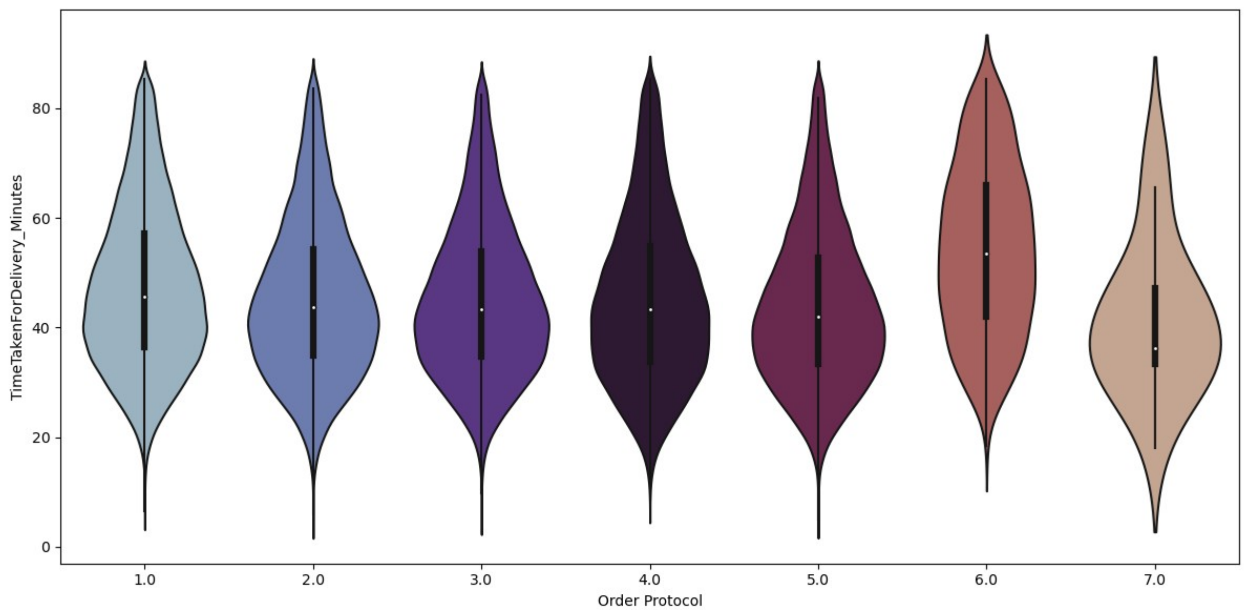
```
order_protocol
1.0    54987
3.0    53452
5.0    44516
2.0    24192
4.0    19460
6.0      802
7.0       19
Name: count, dtype: int64
```

```
df['time_taken_for_delivery_minutes'].value_counts()
```

```
time_taken_for_delivery_minutes
0 days 00:41:22.999999998    133
0 days 00:38:00              127
0 days 00:38:01.000000002    122
0 days 00:43:30              121
0 days 00:37:43.000000002    121
                            ...
0 days 00:11:42                1
0 days 00:13:48                1
0 days 00:14:36                1
0 days 00:10:30                1
0 days 00:11:43.000000002      1
Name: count, Length: 4408, dtype: int64
```

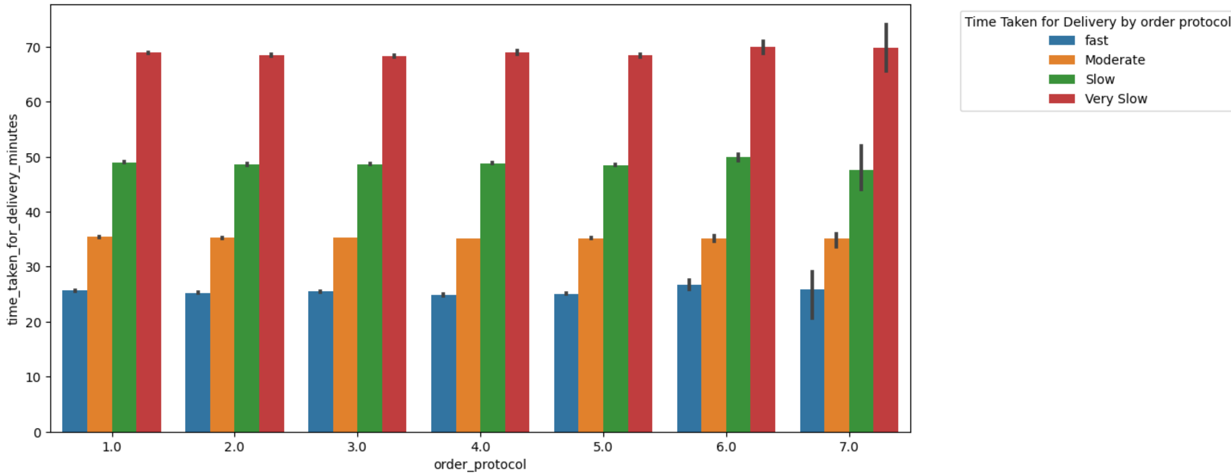
Here we find how order Protocol and time taken for delivery in minutes affect our buisness

```
df['time_taken_for_delivery_minutes']=df['time_taken_for_delivery_minutes'].dt.total_seconds() / 60

plt.figure(figsize=(12, 6))
sns.violinplot(
    x="order_protocol",
    y='time_taken_for_delivery_minutes',
    data=df,
    palette="twilight"
)
plt.xlabel("Order Protocol")
plt.ylabel("TimeTakenForDelivery_Minutes")
plt.tight_layout()
plt.show()
```

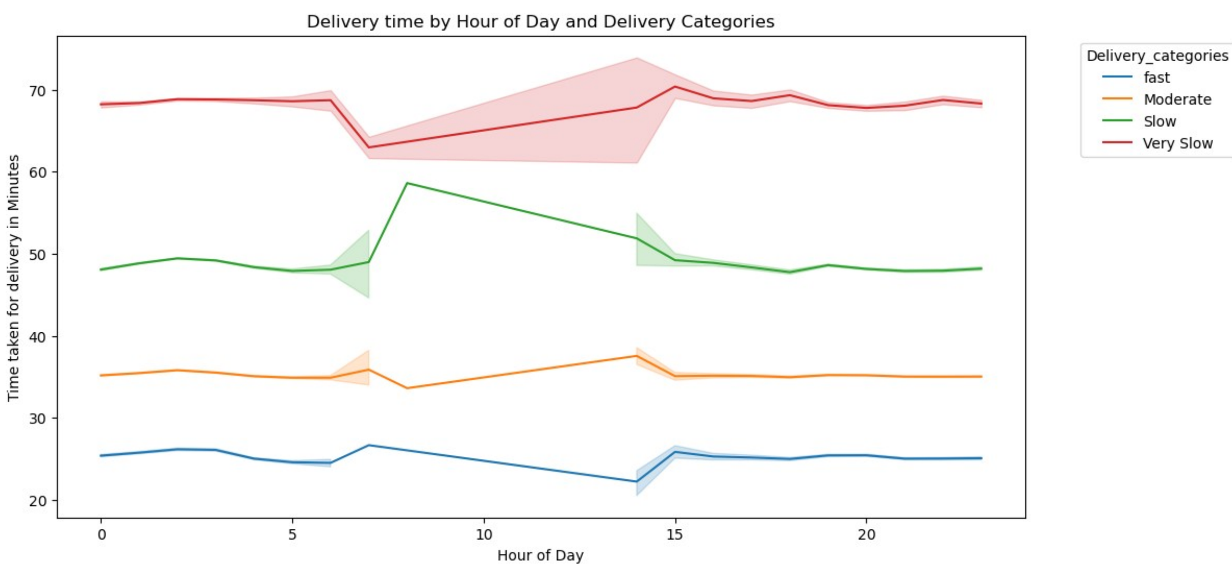



In our case maximum order delivered between the 40 to 50 minutes and the order protocol 6.0 shows versalilty.

```
# we are trying to find how order_protocol and timetaken in minutes variition as per Delivery categories

plt.figure(figsize=(12,6))
sns.barplot(x='order_protocol',y='time_taken_for_delivery_minutes',hue='Delivery_categories',data=df)
plt.legend(title='Time Taken for Delivery by order protocol',bbox_to_anchor=(1.05,1),loc='upper left')
plt.show()
```

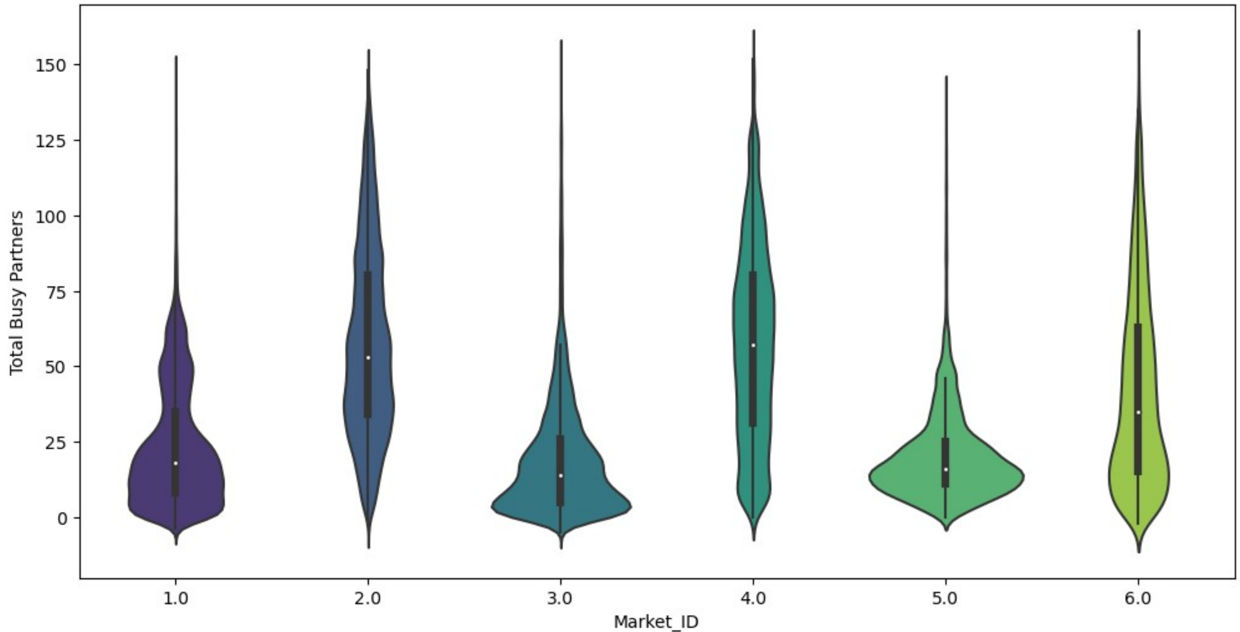



```python
#patterns and trends
plt.figure(figsize=(12,6))
sns.lineplot(x='HourOfDay',y='time_taken_for_delivery_minutes',data=df
,hue='Delivery_categories')
plt.title("Delivery time by Hour of Day and Delivery Categories")
plt.xlabel("Hour of Day")
plt.ylabel("Time taken for delivery in Minutes")
plt.legend(title='Delivery_categories',bbox_to_anchor=(1.05,1),loc='up
per left')
plt.show()
```




```python
# here we are looking for the how "total_busy_partners" and
"market_id" columns are related to each other.
plt.figure(figsize=(12,6))
sns.violinplot(x='market_id',y='total_busy_partners',data=df,palette='
viridis')
```

```
plt.xlabel('Market_ID')
plt.ylabel('Total Busy Partners')
plt.show()
```

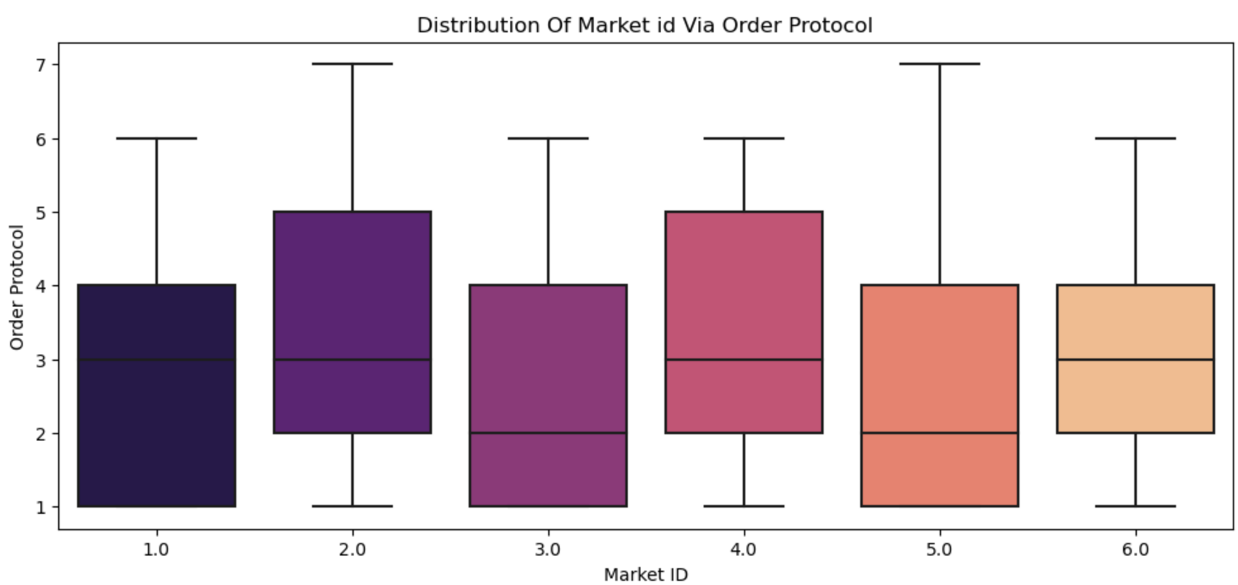



Trying to find the how our service/product related to on 'min_item_price','AvgItemPrice' and 'MaxItemPrice'

```
fig,axes=plt.subplots(1,3,figsize=(15,5))
sns.histplot(x='AvgItemPrice',data=df,palette='viridis',ax=axes[0],kde
=True,bins=5)
axes[0].set_title('Average Item Price')

sns.histplot(x='min_item_price',data=df,color='red',ax=axes[1],kde=Tru
e,bins=5)
axes[1].set_title('Minimum Item Price')

sns.histplot(x='max_item_price',data=df,color='green',ax=axes[2],kde=T
rue,bins=5)
axes[2].set_title('Maximum Item Price')

plt.show()
```


Here we trying to find the distribution of market id by order protocol

```
plt.figure(figsize=(12,5))

sns.boxplot(x='market_id',y='order_protocol',data=df,palette='magma')
plt.title("Distribution Of Market id Via Order Protocol")
plt.xlabel("Market ID")
plt.ylabel("Order Protocol")
plt.show()
```

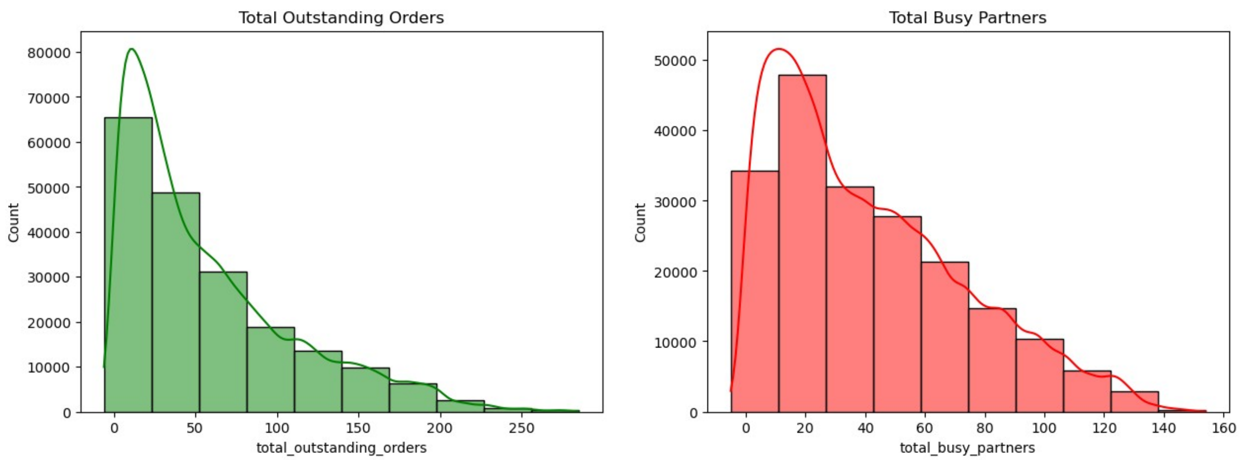



Here we are trying 'total_outstanding_orders' and 'total_busy_partners' are affecting our services

```
fig,axes=plt.subplots(1,2,figsize=(15,5))

sns.histplot(x='total_outstanding_orders',data=df,ax=axes[0],kde=True,
color='green',bins=10)
axes[0].set_title('Total Outstanding Orders')
```

```
sns.histplot(x='total_busy_partners',data=df,ax=axes[1],kde=True,color
='red',bins=10)
axes[1].set_title('Total Busy Partners')
```

```
Text(0.5, 1.0, 'Total Busy Partners')
```




Observation are following of the 'TotalBusyPartners' and 'TotalOutstandingOrders'

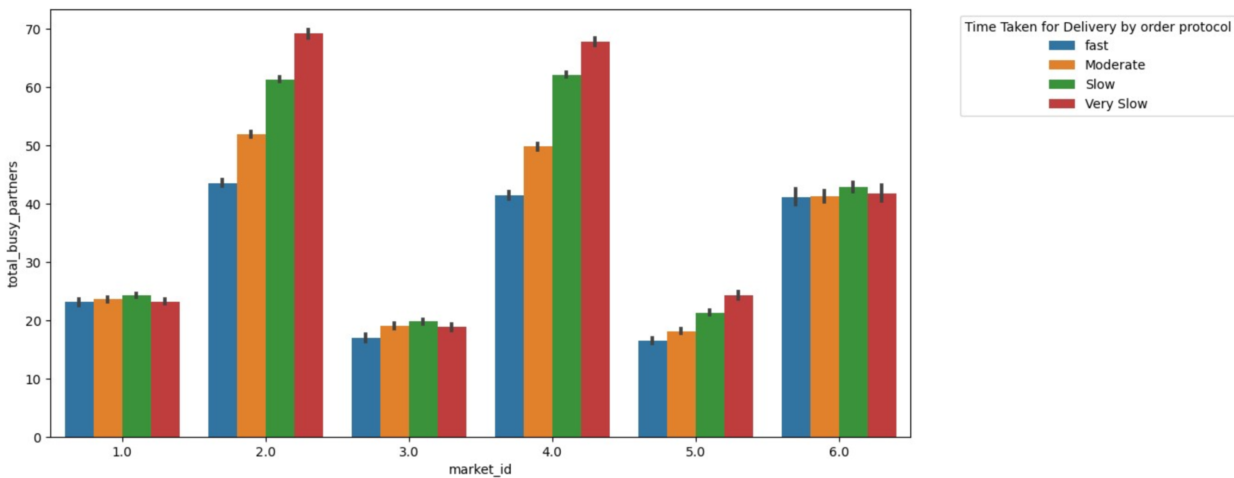
Total Busy Partners: Number of Delivery Partners attending the other tasks.

```
Since Our "TotalBusyPartners" falls within 0-160 and peak is 20 and
count more than 50000. After that second most is ~15 and count is
~35000
```

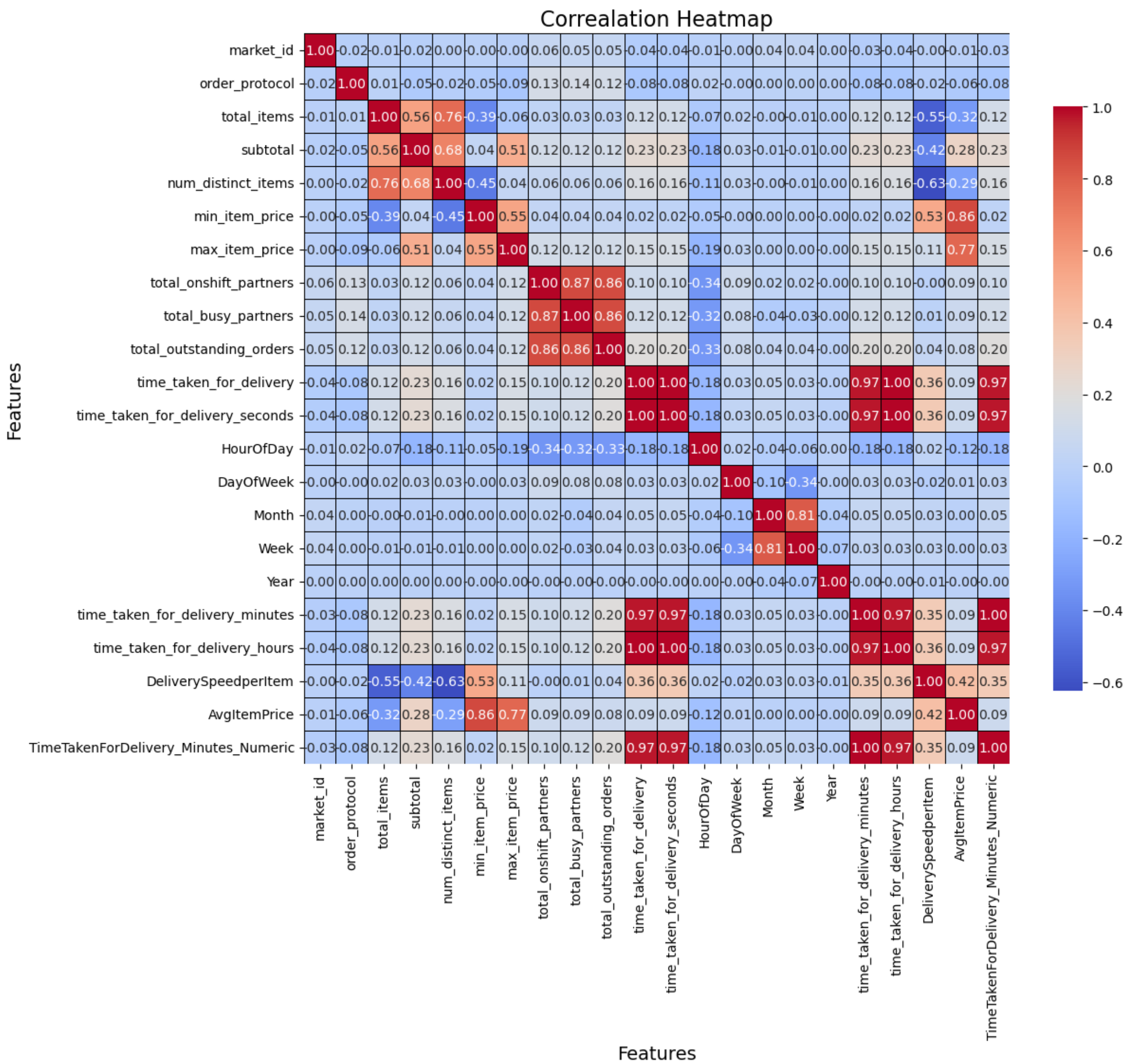
Total Outstanding Orders: Total number of orders to be fulfilled at the moment

```
Since our "TotalOutstandingOrders" fall between  0-250 and maximum
"TotalOutstandingOrders"  deliver at ≈30 and count ≈65000 and second
most "TotalOutstandingOrders" at  ≈ 40 and frequency  is ≈ 50000.
```

```
plt.figure(figsize=(12,6))
sns.barplot(x='market_id',y='total_busy_partners',hue='Delivery_catego
ries',data=df)
plt.legend(title='Time Taken for Delivery by order
protocol',bbox_to_anchor=(1.05,1),loc='upper left')
plt.show()
```




```
numeric_df=df.select_dtypes(include=[np.number])
corr=numeric_df.corr()

plt.figure(figsize=(12,10))

sns.heatmap(corr,annot=True,cmap='coolwarm',fmt='0.2f',linewidth=0.5,l
inecolor='black',cbar_kws={'shrink':0.8},annot_kws={'size':10})
plt.title("Correalation Heatmap", fontsize = 16)
plt.xlabel("Features", fontsize = 14)
plt.ylabel("Features", fontsize = 14)
plt.show()
```




```
Image(filename="M:\Porter Case Study\Images\Insights.jpeg",width=900)
```


INSIGHT

A) Delivery Speed:- Our Most delivery categories fall inside the SLOW, MODERATE and very less come inside the FAST and VERY SLOW categories. Also HOUROFDAY at peak time not able to handle the customers. Having NUMBER OF DISINCT ITEMS IN THE ORDER takes more time to deliver the product.

B) Total Busy Partners and Total Outstanding Orders:- At the peak performance TOTAL BUSY PARTNERS are less and other time there much more PARTNERS. Also the TOTAL OUTSTANDING ORDERS comes between 0-40

C) Customer Spending and Item trends:- Our maximum customer are in the minimum categories and services they are using which is Budget Friendly, So we can say that our service is generally use by middle class family.

D) Market and Total Busy Partners : in market id "2.0" and "4.0" high freqency and the servieces in the high density these two market have high demand and rest of all are the in the general but we are not able to full fill the demand bcz here we delivered maximum order in "moderate" and "slow" Categories.

E) Store and Market :store are releated on the market we observed that the market highly coreleated to the market bcz in the bifarcation of "Delivery_Categories" we got the maximum order are fall in the "slow" categories.

F) Order Proctol and Partner Effciency : In some protocol we are able to handle the delivery but in some we are not bcz we our onshift parners are less and not full fill the demand.

G) Operational Efficency : A high number of "total_OutSatanding_Orders" are releated to delayed delevires specially in the "TotalBusyPartners" during the peak hour we don't have enough work force to handle the volume of orders.There are Seven "Orde_protcol" that's the reason we have verify Orders coming from where like through Porter,call to restaurant, pre-booked, third-party, etc. all these thing taking to much time.

```python
Image(filename="M:\Porter Case Study\Images\letter-recommendation-7580900.webp",width=600)
```

Delivery Speed Optimization : most of our fall in "moderate" and "slow" category and in peak hour we are not able to full fill the demand. We can increase the staff during the peak hour and ensure there are enough delivery partners at peak hours we adjust this to give over time to the delivery partners. Implement incentive program for those delivery who frequently delivered product in the fastCategory, this can boost overall speed and keep partners to motivate and redirect their root.

Flexible workhour : Offers flexible work hour to the delivery partners, allow them to work at the peak hours handle the work and reduce the totalOutstandingOrders.

Customer Segmentation and Targeting : Since Our customer is budget consious and specially middle class. Introduce the premium services for premium customer those who pay for the superfast delivery orders. Provide the some discount in the premium services during the non peak hours. we can introduce the customer loyality program for the customer those who are frequently orders and refer their freinds give them the rewards or offer premiumservices at more discount.

Market Specific Starategdy : in market 4.0 and market 6.0 high demand and we are not able full fill it also our Delivery Category is slow. Assign more delivery partners and resources during the peak hours partnership with local retailsto take geographic advantages.

Store Level Improvement : Stores are highly corelated with the speed of delivery in slow Categories.Regulary audit the high volume orders and why is it slow that meansin preparation time, inventory management, order processing or something for causing the delays.work with stream line and give the clear instuction and provide training if it necessary,introduce the incentive for who those delivered the high volume product in fast Category.

Improving Order protocol Efficency : Some order protocols types are more efficent and some are stuggling to fullfil their orders due to the lack of partners.introduce streamline protocols and assign the protocols types like directly pre-booked,and third party protocols automatically ordered no need to manual verification andconfiramtion. introduce the AI which assign the assigment of Orders equally or less dependent on a special Store.

Customer Satisfacion Initiatives : Delay and slower services in some category leading to lower customer satisfaction.Implement a feedback mechanism where customer can give the real time feedback andafter use this data we can identify what is most concern of customeras delivey speed or the better communication.provide real time tracking system and estimated time to deliverd the product alsoinform them if in case the product will deliver late, this all above mentioned thingsreduce the customer frustration during the peak workhour.

Long-Term Growth Strategies : As demand grows particulary high market invest in growing force of delivery partners.We can use predictive model to to predict where the demand will increase and accordingto them we will prepare.strengthen releationships with stores and delivery partners provide the analyis and insightof their performance this will help to improve their work and as well as customer behaviour and improve customer satifaction rate.

```
Image(filename="M:\Porter Case Study\Images\questions.jpeg",width=500)
```


Data Structure and Overview

```
# What is the shape of the dataset (number of rows and columns)?
df.shape

(197428, 27)

#What are the data types of each column?
df.info()

<class 'pandas.core.frame.DataFrame'>
RangeIndex: 197428 entries, 0 to 197427
Data columns (total 27 columns):
 #   Column                                  Non-Null Count   Dtype

---  ------                                  --------------   -----

 0   market_id                               197428 non-null  float64

 1   created_at                              197428 non-null
datetime64[ns]
 2   actual_delivery_time                    197428 non-null
datetime64[ns]
```

```
 3   store_id                              197428 non-null  object
 4   store_primary_category                197428 non-null  object
 5   order_protocol                        197428 non-null  float64
 6   total_items                           197428 non-null  int64
 7   subtotal                              197428 non-null  int64
 8   num_distinct_items                    197428 non-null  int64
 9   min_item_price                        197428 non-null  int64
 10  max_item_price                        197428 non-null  int64
 11  total_onshift_partners                197428 non-null  float64
 12  total_busy_partners                   197428 non-null  float64
 13  total_outstanding_orders              197428 non-null  float64
 14  time_taken_for_delivery               197428 non-null
timedelta64[ns]
 15  time_taken_for_delivery_seconds       197428 non-null  float64
 16  HourOfDay                             197428 non-null  int32
 17  DayOfWeek                             197428 non-null  int32
 18  Month                                 197428 non-null  int32
 19  Week                                  197428 non-null  UInt32
 20  Year                                  197428 non-null  int32
 21  time_taken_for_delivery_minutes       197428 non-null  float64
 22  time_taken_for_delivery_hours         197428 non-null  float64
 23  Delivery_categories                   197428 non-null  category
 24  DeliverySpeedperItem                  197428 non-null  float64
 25  AvgItemPrice                          197428 non-null  float64
 26  TimeTakenForDelivery_Minutes_Numeric  197428 non-null  float64
dtypes: UInt32(1), category(1), datetime64[ns](2), float64(11),
int32(4), int64(5), object(2), timedelta64[ns](1)
memory usage: 35.8+ MB
```

```
#Are there any missing values in the dataset? If so, how many and in which columns?
print(df.isnull().sum())
'''
At the initial investigation some null values present in our dataset but we already impute it
at present there no missing in our dataset.
'''
```

```
market_id                                  0
created_at                                 0
actual_delivery_time                       0
store_id                                   0
store_primary_category                     0
order_protocol                             0
total_items                                0
subtotal                                   0
num_distinct_items                         0
min_item_price                             0
max_item_price                             0
total_onshift_partners                     0
total_busy_partners                        0
total_outstanding_orders                   0
time_taken_for_delivery                    0
time_taken_for_delivery_seconds            0
HourOfDay                                  0
DayOfWeek                                  0
Month                                      0
Week                                       0
Year                                       0
time_taken_for_delivery_minutes            0
time_taken_for_delivery_hours              0
Delivery_categories                        0
DeliverySpeedperItem                       0
AvgItemPrice                               0
TimeTakenForDelivery_Minutes_Numeric       0
dtype: int64

'\nAt the initial investigation some null values present in our dataset but we already impute it\nat present there no missing in our dataset.\n'
```

Descriptive Statistics

```
# 1. What are the basic statistical summaries (mean, median, standard deviation) for the numerical features?
df.describe()
```

```
           market_id                    created_at  \
count  197428.000000                        197428
```

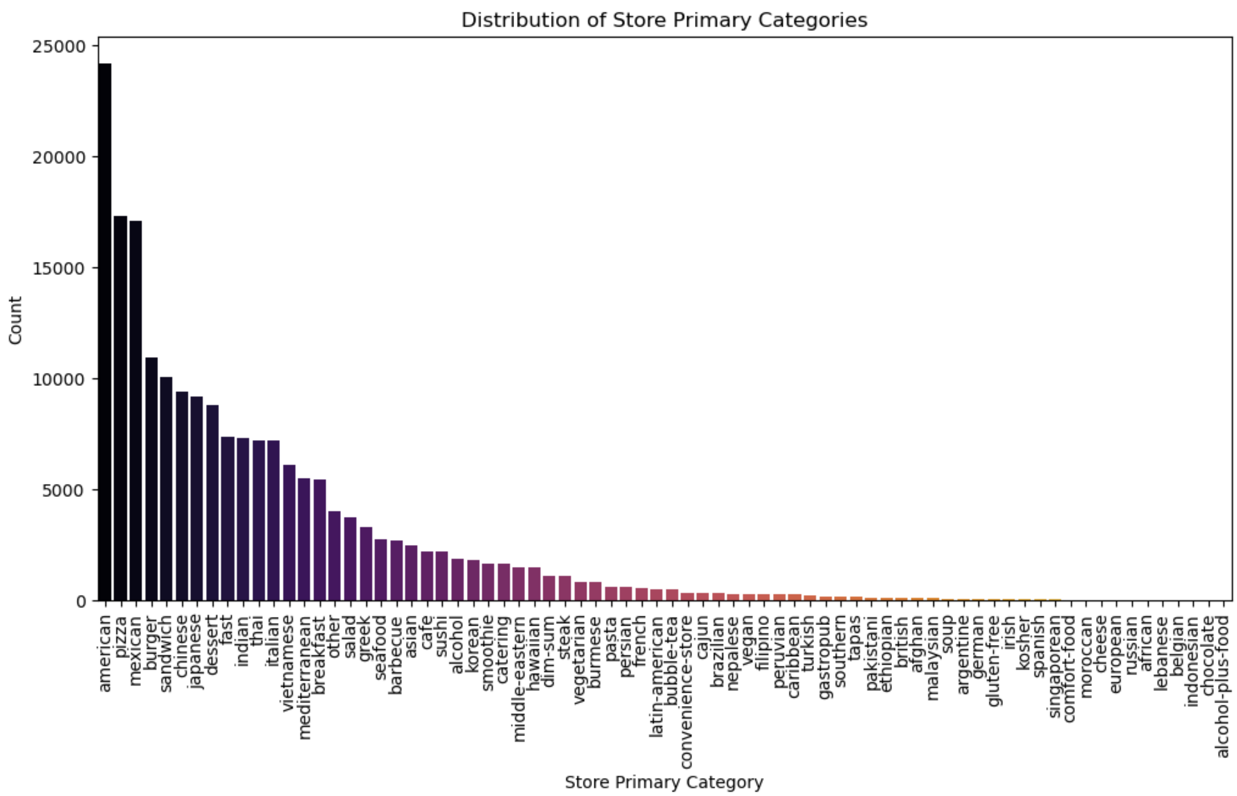
```
mean       2.978443  2015-02-04 22:00:09.537962752
min        1.000000            2014-10-19 05:24:15
25%        2.000000            2015-01-29 02:32:42
50%        3.000000     2015-02-05 03:29:09.500000
75%        4.000000     2015-02-12 01:39:18.500000
max        6.000000            2015-02-18 06:00:44
std        1.524676                            NaN

                actual_delivery_time  order_protocol    total_items  \
count                         197428   197428.000000  197428.000000
mean   2015-02-04 22:47:59.840164608        2.882529       3.196391
min              2015-01-21 15:58:11        1.000000       1.000000
25%    2015-01-29 03:22:23.750000128        1.000000       2.000000
50%       2015-02-05 04:40:28.500000        3.000000       3.000000
75%    2015-02-12 02:25:17.750000128        4.000000       4.000000
max              2015-02-19 22:45:31        7.000000     411.000000
std                              NaN        1.503796       2.666546

            subtotal  num_distinct_items  min_item_price
max_item_price  \
count  197428.000000       197428.000000   197428.000000
197428.000000
mean     2682.331402            2.670791      686.218470
1159.588630
min         0.000000            1.000000      -86.000000
0.000000
25%      1400.000000            1.000000      299.000000
800.000000
50%      2200.000000            2.000000      595.000000
1095.000000
75%      3395.000000            3.000000      949.000000
1395.000000
max     27100.000000           20.000000    14700.000000
14700.000000
std      1823.093688            1.630255      522.038648
558.411377

       total_onshift_partners  ...      HourOfDay      DayOfWeek  \
count           197428.000000  ...  197428.000000  197428.000000
mean                44.826468  ...       8.467213       3.218966
min                 -4.000000  ...       0.000000       0.000000
25%                 17.000000  ...       2.000000       1.000000
50%                 37.000000  ...       3.000000       3.000000
75%                 65.000000  ...      19.000000       5.000000
max                171.000000  ...      23.000000       6.000000
std                 34.518204  ...       8.658759       2.045789

               Month      Week           Year  \
count  197428.000000  197428.0  197428.000000
mean        1.653170  5.903712    2014.999995
```

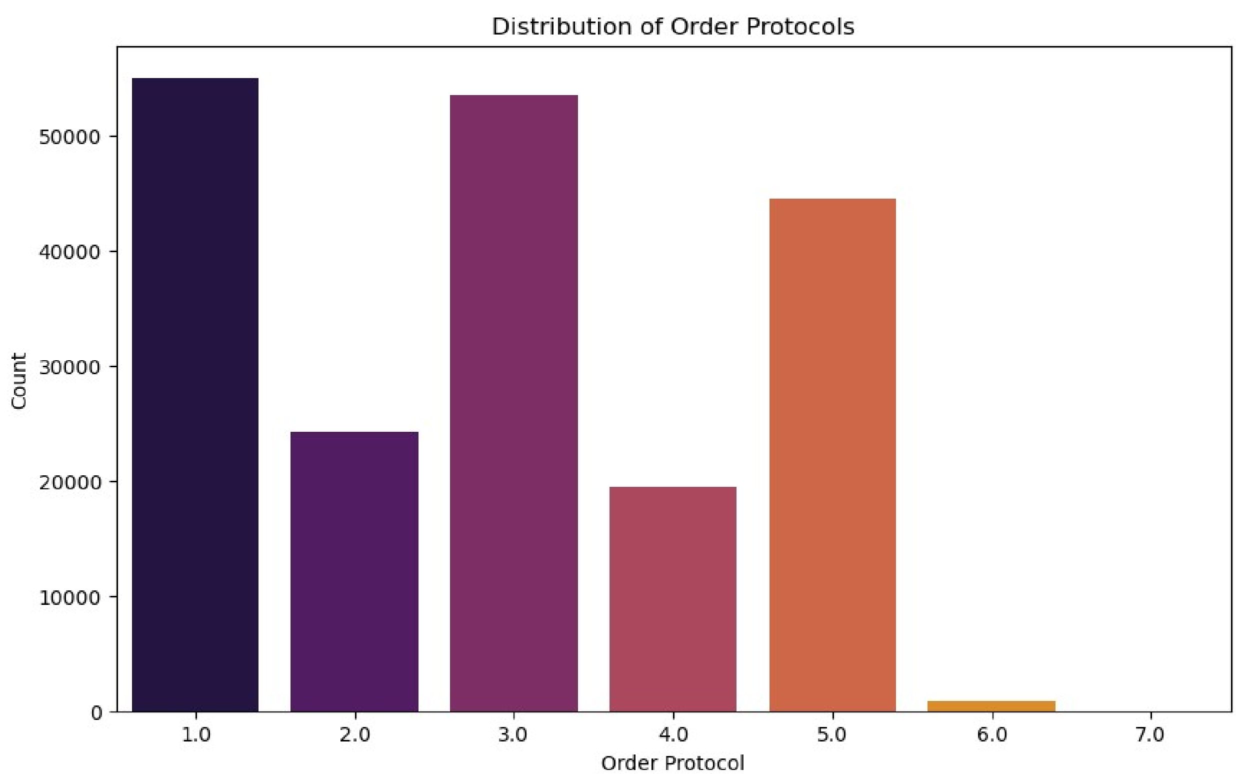
```
min         1.000000       4.0    2014.000000
25%         1.000000       5.0    2015.000000
50%         2.000000       6.0    2015.000000
75%         2.000000       7.0    2015.000000
max        10.000000      42.0    2015.000000
std         0.476345  1.216714       0.002251

       time_taken_for_delivery_minutes  time_taken_for_delivery_hours
\
count                    197428.000000                  197428.000000

mean                         45.571782                       0.763809

min                           4.950000                       0.061944

25%                          34.800000                       0.580833

50%                          43.733333                       0.730833

75%                          54.783333                       0.917222

max                          85.300000                       1.471111

std                          14.473399                       0.246818


       DeliverySpeedperItem   AvgItemPrice  \
count         197428.000000  197428.000000
mean              21.135179     975.322997
min                0.123885       0.000000
25%               10.670000     647.666667
50%               16.725000     895.000000
75%               27.108333    1195.000000
max               88.250000   14700.000000
std               14.823381     517.244403

       TimeTakenForDelivery_Minutes_Numeric
count                         197428.000000
mean                              45.571782
min                                4.950000
25%                               34.800000
50%                               43.733333
75%                               54.783333
max                               85.300000
std                               14.473399

[8 rows x 24 columns]

# 2. What is the distribution of the categorical variables like
store_primary_category and order_protocol?
```

```python
categorical_df=df.select_dtypes(include='object')
categorical_df.describe()
```

| | store_id | store_primary_category |
|---|---|---|
| count | 197428 | 197428 |
| unique | 6743 | 74 |
| top | d43ab110ab2489d6b9b2caa394bf920f | american |
| freq | 937 | 24159 |

```python
# 3. What is the distribution of the categorical variables like
store_primary_category and order_protocol?
Store_primary_category_dist=df['store_primary_category'].value_counts(
)

plt.figure(figsize=(12,6))
sns.barplot(x=Store_primary_category_dist.index,y=Store_primary_catego
ry_dist.values, palette='inferno')
plt.title('Distribution of Store Primary Categories')
plt.xlabel('Store Primary Category')
plt.ylabel('Count')
plt.xticks(rotation=90)
plt.show()

order_protocol_dist=df['order_protocol'].value_counts()
plt.figure(figsize=(10,6))
sns.barplot(x=order_protocol_dist.index,y=order_protocol_dist.values,p
alette='inferno')
plt.title('Distribution of Order Protocols')
plt.xlabel('Order Protocol')
plt.ylabel('Count')
plt.show()
```


Distribution of Store Primary Categories
Count
Store Primary Category
american
pizza
mexican
burger
sandwich
chinese
japanese
dessert
fast
indian
thai
italian
vietnamese
mediterranean
breakfast
other
salad
greek
seafood
barbecue
asian
cafe
sushi
alcohol
korean
smoothie
catering
middle-eastern
hawaiian
dim-sum
steak
vegetarian
burmese
pasta
persian
french
latin-american
bubble-tea
convenience-store
cajun
brazilian
nepalese
vegan
filipino
peruvian
caribbean
turkish
gastropub
southern
tapas
pakistani
ethiopian
british
afghan
malaysian
soup
argentine
german
gluten-free
irish
kosher
spanish
singaporean
comfort-food
moroccan
cheese
european
russian
african
lebanese
belgian
indonesian
chocolate
alcohol-plus-food

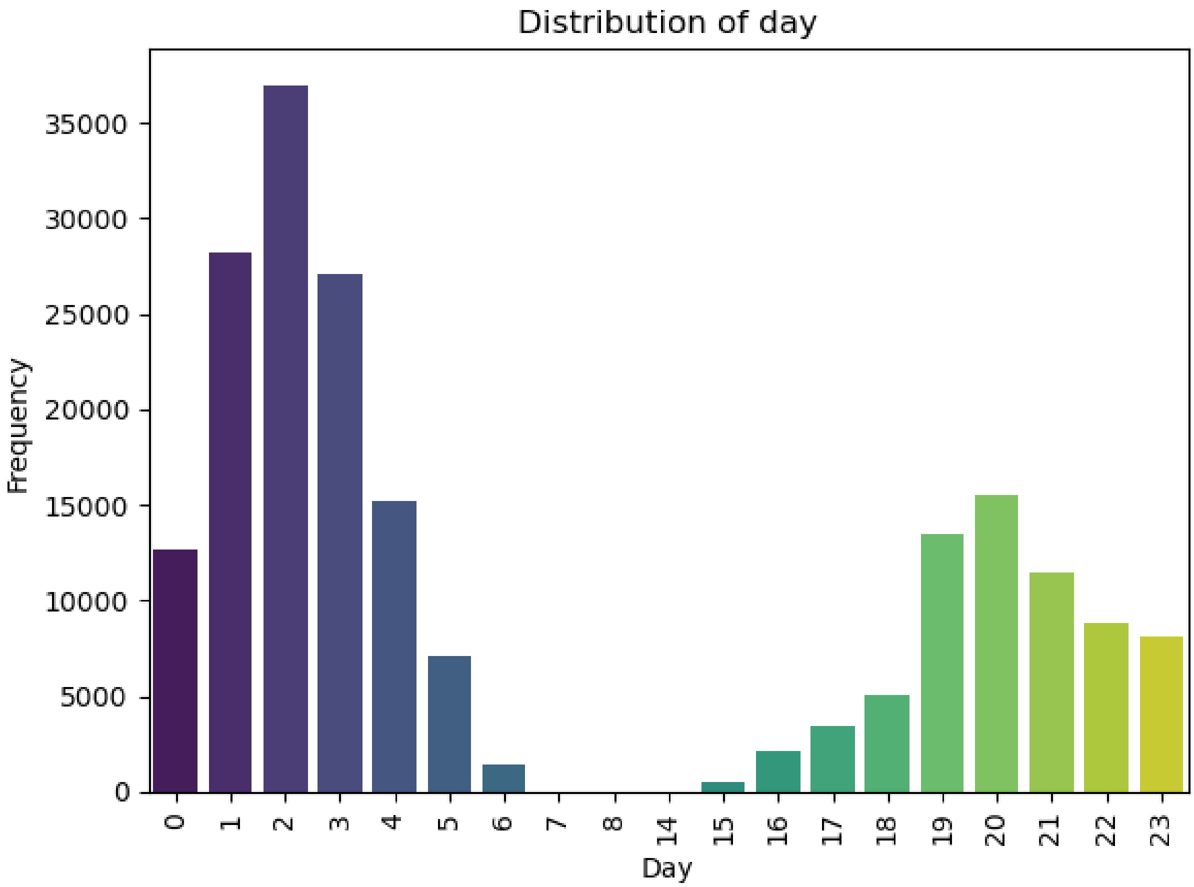

Distribution of Order Protocols
Count
Order Protocol
1.0
2.0
3.0
4.0
5.0
6.0
7.0

Datetime Features

```
# 1. How many orders were placed each day/week/month?
print("Total Order placed in each day :",
df["DayOfWeek"].value_counts())
print("="*80)

print("Total Order Placed in each week :", df["Week"].value_counts())
print("*"*80)


print("Total Order Placed in each month :",
df["Month"].value_counts())
```

```
Total Order placed in each day : DayOfWeek
5    34541
6    33620
4    27875
0    27403
3    25673
2    24254
1    24062
Name: count, dtype: int64
======================================================================
==========
Total Order Placed in each week : Week
7     52042
6     51188
5     45342
4     30864
8     17991
42        1
Name: count, dtype: Int64
**********************************************************************
**********
Total Order Placed in each month : Month
2     128945
1      68482
10         1
Name: count, dtype: int64
```

```
# 2.What is the distribution of order times throughout the day?
Hourofday_dist=df['HourOfDay'].value_counts().sort_index()
sns.barplot(x=Hourofday_dist.index,y=Hourofday_dist.values,palette='vi
ridis')
plt.title("Distribution of day")
plt.xlabel("Day")
plt.ylabel("Frequency")
plt.xticks(rotation = 90)
plt.tight_layout()
plt.show()
```




Feature Engineering

```
# Q1. How can we create a new feature for the time taken for each delivery?

# So we already created the "TimeTakenForEachDelivery" so don't need to create again
df['DeliverySpeedperItem']
```

```
0         15.745833
1         67.066667
2         29.683333
3          8.541667
4         13.277778
            ...
197423    21.705556
197424     9.397222
197425    10.026667
197426    65.116667
197427     9.283333
Name: DeliverySpeedperItem, Length: 197428, dtype: float64
```

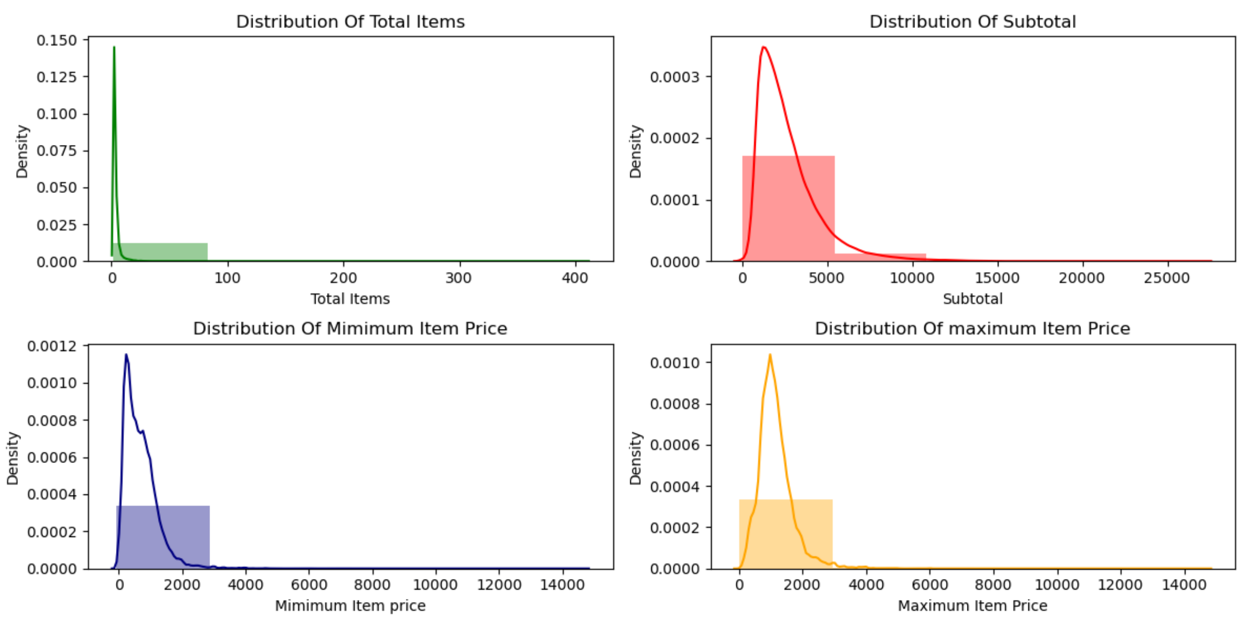
```
# Q2. How can we extract additional features from the datetime
columns, such as the hour of the day or the day of the week?
# Already created these features

print(df['HourOfDay'])
print("="*80)
print(df['DayOfWeek'])
```

```
0         22
1         21
2         20
3         21
4          2
          ..
197423     0
197424     0
197425     4
197426    18
197427    19
Name: HourOfDay, Length: 197428, dtype: int32
======================================================================
==========
0         4
1         1
2         3
3         1
4         6
         ..
197423    1
197424    4
197425    5
197426    6
197427    6
Name: DayOfWeek, Length: 197428, dtype: int32
```

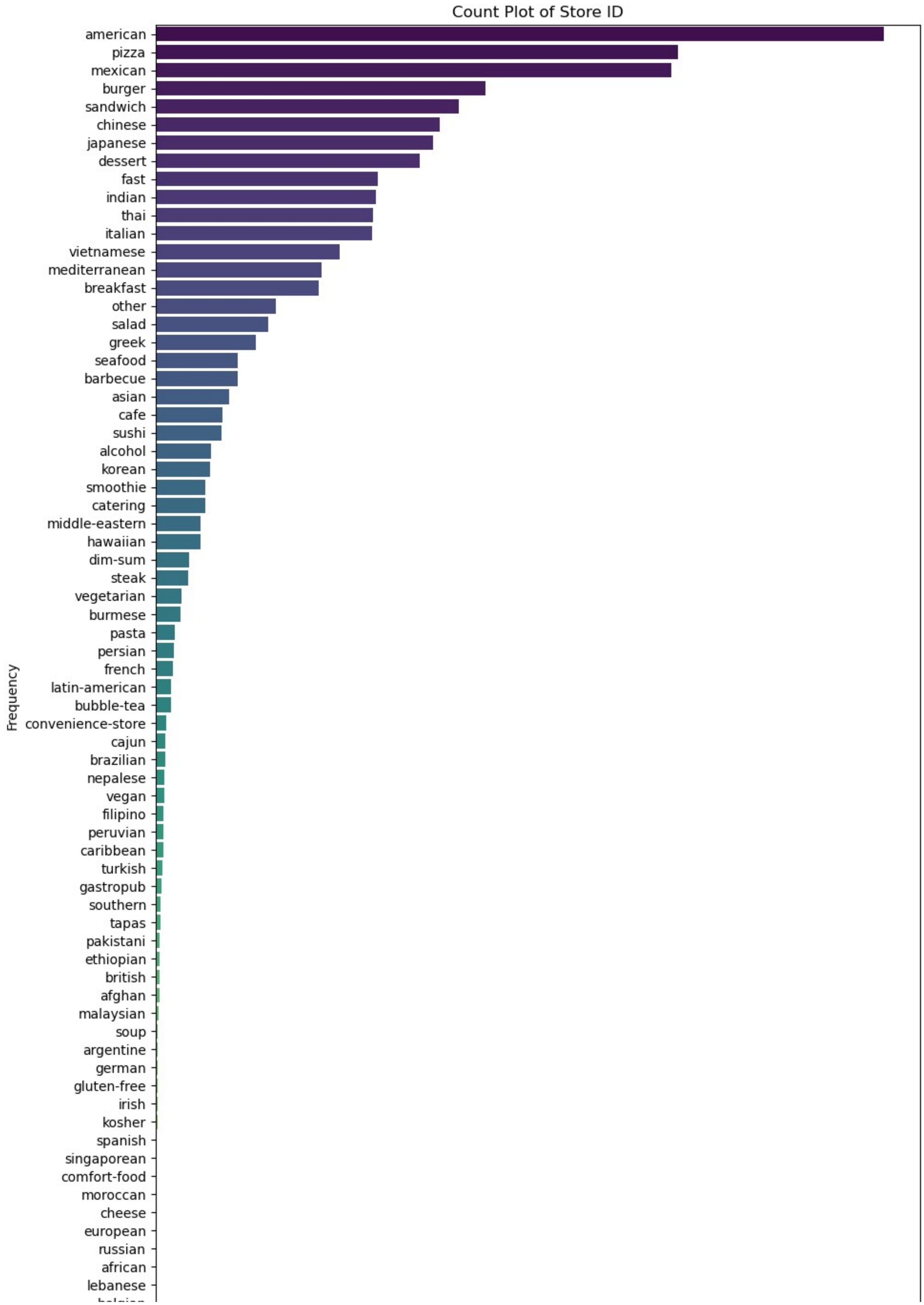
Exploratory Data Analysis (EDA)

```
#What are the distribution plots for continuous variables like
total_items, subtotal, min_item_price, and max_item_price?

fig,axes=plt.subplots(2,2,figsize=(12,6))

sns.distplot(x=df['total_items'],ax=axes[0,0],color='green',bins=5)
axes[0,0].set_title("Distribution Of Total Items")
axes[0,0].set_xlabel("Total Items")
axes[0,0].set_ylabel("Density")

sns.distplot(x=df['subtotal'],ax=axes[0,1],color='red',bins=5)
axes[0, 1].set_title("Distribution Of Subtotal")
axes[0, 1].set_xlabel("Subtotal")
```

```
axes[0, 1].set_ylabel("Density")

sns.distplot(x=df['min_item_price'],ax=axes[1,0],color='navy',bins=5)
axes[1, 0].set_title("Distribution Of Mimimum Item Price")
axes[1, 0].set_xlabel("Mimimum Item price")
axes[1, 0].set_ylabel("Density")

sns.distplot(x=df['max_item_price'],ax=axes[1,1],color='orange',bins=5
)
axes[1, 1].set_title("Distribution Of maximum Item Price")
axes[1, 1].set_xlabel("Maximum Item Price")
axes[1, 1].set_ylabel("Density")

plt.tight_layout()
plt.show()
```




```
#What are the count plots for categorical variables like
store_primary_category and order_protocol?

plt.figure(figsize=(10,15))
sns.countplot(y='store_primary_category',data=df,order=df['store_prima
ry_category'].value_counts().index,palette='viridis')
plt.title("Count Plot of Store ID")
plt.xlabel("Store ID")
plt.ylabel("Frequency")
plt.tight_layout()
plt.show()
```


Count Plot of Store ID
Frequency
american
pizza
mexican
burger
sandwich
chinese
japanese
dessert
fast
indian
thai
italian
vietnamese
mediterranean
breakfast
other
salad
greek
seafood
barbecue
asian
cafe
sushi
alcohol
korean
smoothie
catering
middle-eastern
hawaiian
dim-sum
steak
vegetarian
burmese
pasta
persian
french
latin-american
bubble-tea
convenience-store
cajun
brazilian
nepalese
vegan
filipino
peruvian
caribbean
turkish
gastropub
southern
tapas
pakistani
ethiopian
british
afghan
malaysian
soup
argentine
german
gluten-free
irish
kosher
spanish
singaporean
comfort-food
moroccan
cheese
european
russian
african
lebanese

Handling Missing Values

```
#How can we handle missing values in the dataset, especially for
important columns like store_primary_category?

print(df['store_primary_category'].isnull().sum())
print(df['store_primary_category'].value_counts())

0
store_primary_category
american              24159
pizza                 17321
mexican               17099
burger                10958
sandwich              10060
                      ...
lebanese                  9
belgian                   2
indonesian                2
chocolate                 1
alcohol-plus-food         1
Name: count, Length: 74, dtype: int64
```

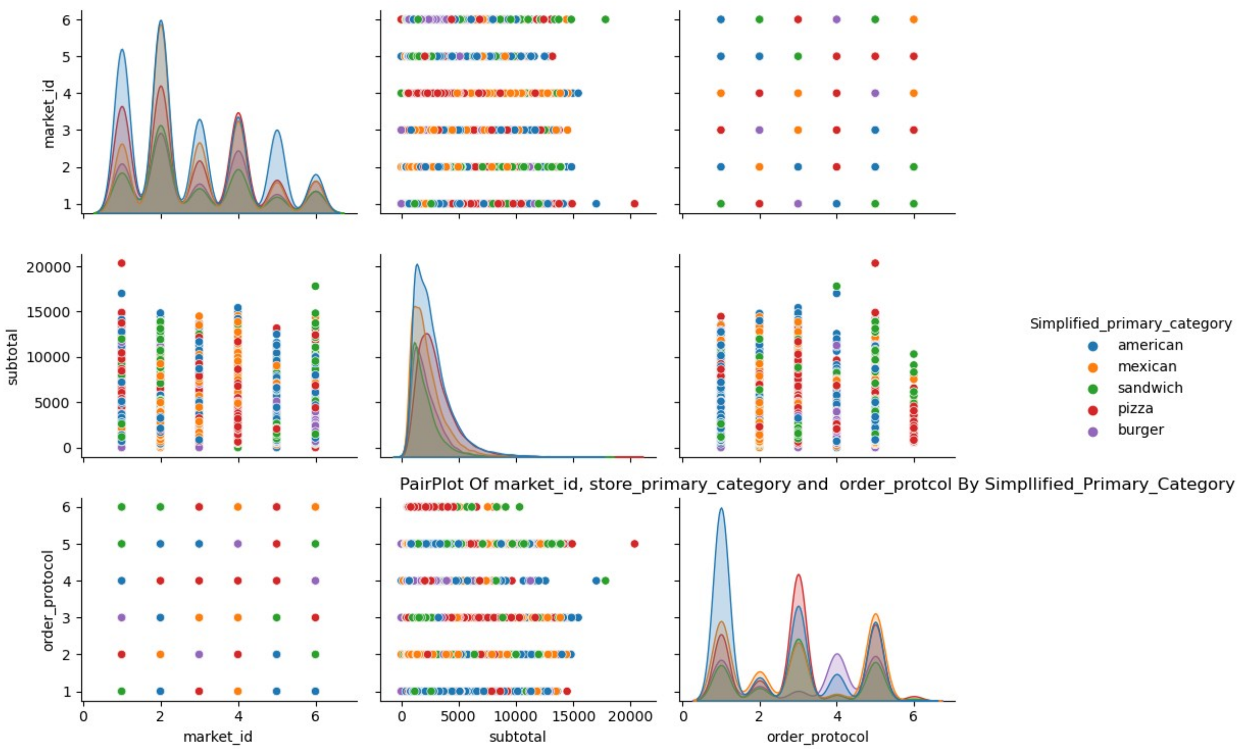
Correlation Analysis

```
#What are the Pearson and Spearman correlation coefficients between
numerical features (e.g., total_items, subtotal, min_item_price,
max_item_price)?

numerical_features=df[['total_items','subtotal','min_item_price','max_
item_price']]
Pearson_corr=numerical_features.corr(method='pearson')
print(Pearson_corr)

print('='*80)

Spearman_corr=numerical_features.corr(method='spearman')
print(Spearman_corr)

                total_items  subtotal  min_item_price  max_item_price
total_items        1.000000  0.558067       -0.393149       -0.058233
subtotal           0.558067  1.000000        0.037038        0.505547
min_item_price    -0.393149  0.037038        1.000000        0.545484
max_item_price    -0.058233  0.505547        0.545484        1.000000
======================================================================
==========
                total_items  subtotal  min_item_price  max_item_price
total_items        1.000000  0.664301       -0.590844       -0.006598
subtotal           0.664301  1.000000        0.027429        0.592247
min_item_price    -0.590844  0.027429        1.000000        0.429658
max_item_price    -0.006598  0.592247        0.429658        1.000000
```

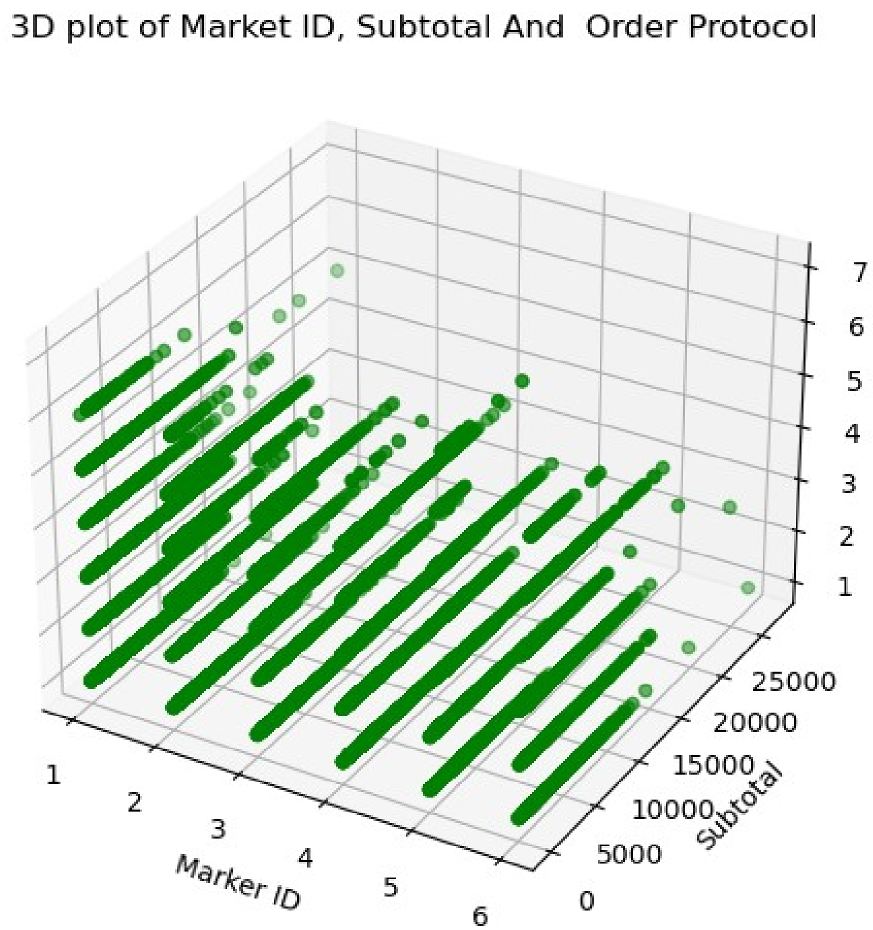
```
#What do these correlations suggest?
```

Multivariate Analysis

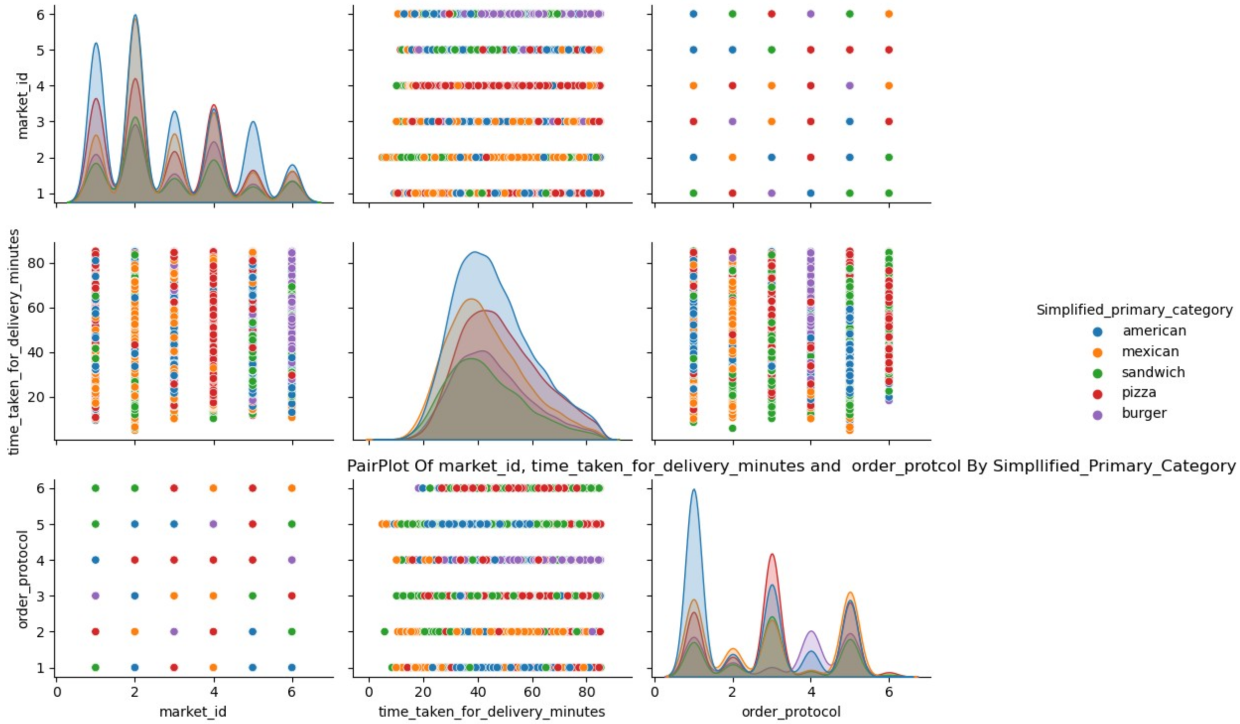
```
#How do multiple factors (e.g., market_id, store_primary_category,
order_protocol) together influence the subtotal?

#Extracting top 5 'store_primary_category bcz we have total 174 unique
values might be our plot will cluttered for clear pictures we use top
5
Category_counts=df['store_primary_category'].value_counts()
top5=Category_counts.nlargest(5).index

#creating temp df we don't our dataset to be affect
temp_df=df.copy()

#creating new feature in temp_df
temp_df['Simplified_primary_category']=temp_df['store_primary_category
'].apply(lambda x: x if x in top5 else np.nan)

filtered_df=temp_df[temp_df['Simplified_primary_category'].notnull()]

sns.pairplot(data=temp_df,vars=['market_id','subtotal','order_protocol
'],hue='Simplified_primary_category')
plt.title("PairPlot Of market_id, store_primary_category and
order_protcol By Simpllified_Primary_Category")
plt.tight_layout()
plt.show()
```



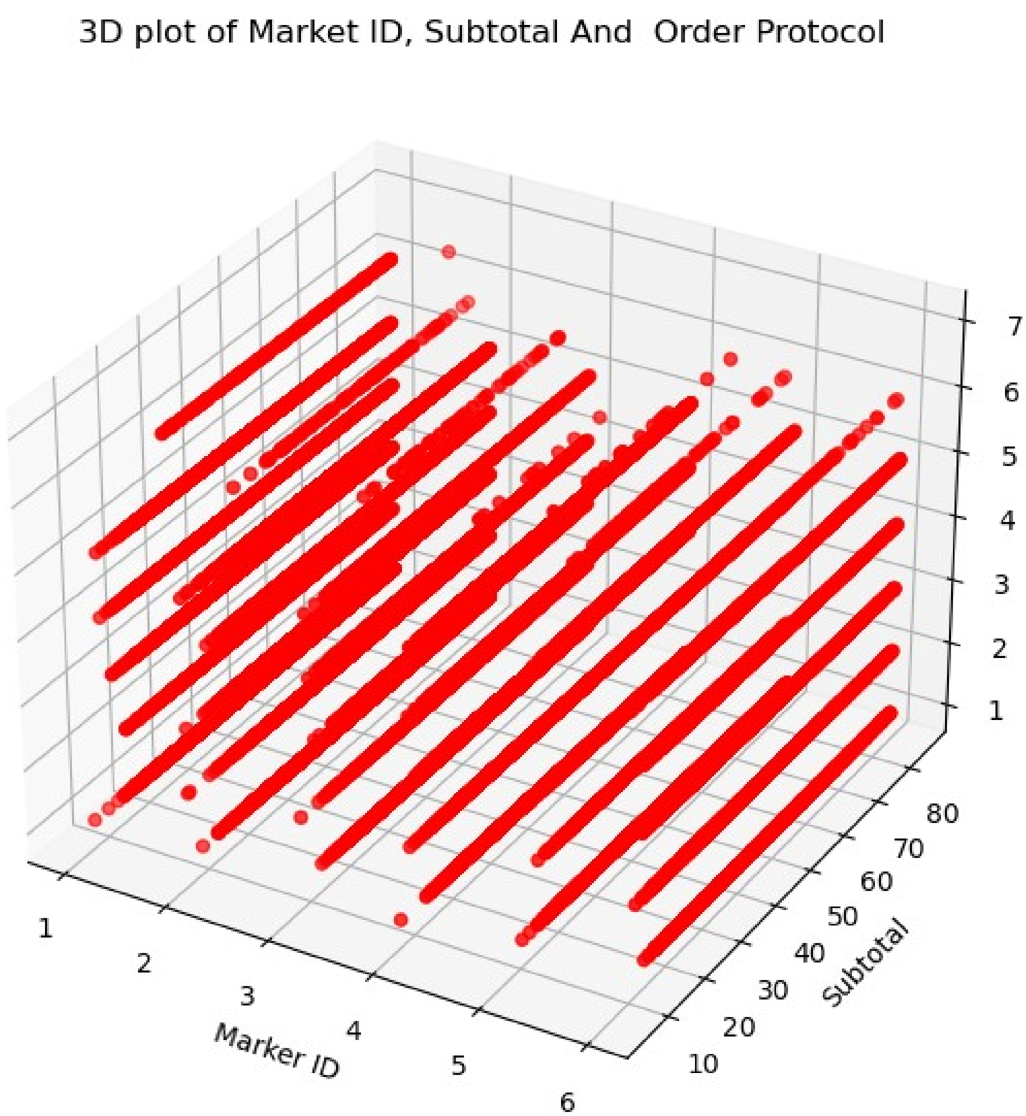

```
# Ploting 3d plot to see how 'market_id' , 'subtotal' and
'order_protocol' related to each other

from mpl_toolkits.mplot3d import Axes3D

fig=plt.figure(figsize=(10,5))
ax=plt.subplot(111,projection="3d")

ax.scatter(df['market_id'],df['subtotal'],df['order_protocol'],color='
green',marker='o')
ax.set_title("3D plot of Market ID, Subtotal And  Order Protocol")
ax.set_xlabel("Marker ID")
ax.set_ylabel("Subtotal")
ax.set_zlabel("Order Protocol")

plt.tight_layout()
plt.show()
```




```
#How do multiple factors (e.g., market_id, store_primary_category, order_protocol) together influence the delivery time?

sns.pairplot(data=temp_df,vars=['market_id','time_taken_for_delivery_minutes','order_protocol'],hue='Simplified_primary_category')
plt.title("PairPlot Of market_id, time_taken_for_delivery_minutes and order_protcol By Simpllified_Primary_Category")
plt.tight_layout()
plt.show()
```




```
plt.figure(figsize=(10,6))
ax=plt.subplot(111,projection='3d')

ax.scatter(df['market_id'],df['time_taken_for_delivery_minutes'],df['o
rder_protocol'],color='red',marker='o')
ax.set_title("3D plot of Market ID, Subtotal And  Order Protocol")
ax.set_xlabel("Marker ID")
ax.set_ylabel("Subtotal")
ax.set_zlabel("Order Protocol")

plt.tight_layout()
plt.show()
```




Outliers Analysis

```
#Are there any outliers in the dataset? Which method can be used to
identify and handle these outliers?

df.describe()

          market_id                     created_at  \
count  197428.000000                        197428
mean        2.978443  2015-02-04 22:00:09.537962752
min         1.000000            2014-10-19 05:24:15
25%         2.000000            2015-01-29 02:32:42
50%         3.000000     2015-02-05 03:29:09.500000
75%         4.000000     2015-02-12 01:39:18.500000
max         6.000000            2015-02-18 06:00:44
std         1.524676                           NaN
```

```
                actual_delivery_time  order_protocol    total_items  \
count                         197428   197428.000000  197428.000000
mean   2015-02-04 22:47:59.840164608        2.882529       3.196391
min              2015-01-21 15:58:11        1.000000       1.000000
25%    2015-01-29 03:22:23.750000128        1.000000       2.000000
50%       2015-02-05 04:40:28.500000        3.000000       3.000000
75%    2015-02-12 02:25:17.750000128        4.000000       4.000000
max              2015-02-19 22:45:31        7.000000     411.000000
std                              NaN        1.503796       2.666546

            subtotal  num_distinct_items  min_item_price
max_item_price  \
count  197428.000000       197428.000000   197428.000000
197428.000000
mean     2682.331402            2.670791      686.218470
1159.588630
min         0.000000            1.000000      -86.000000
0.000000
25%      1400.000000            1.000000      299.000000
800.000000
50%      2200.000000            2.000000      595.000000
1095.000000
75%      3395.000000            3.000000      949.000000
1395.000000
max     27100.000000           20.000000    14700.000000
14700.000000
std      1823.093688            1.630255      522.038648
558.411377

       total_onshift_partners  ...      HourOfDay      DayOfWeek  \
count           197428.000000  ...  197428.000000  197428.000000
mean                44.826468  ...       8.467213       3.218966
min                 -4.000000  ...       0.000000       0.000000
25%                 17.000000  ...       2.000000       1.000000
50%                 37.000000  ...       3.000000       3.000000
75%                 65.000000  ...      19.000000       5.000000
max                171.000000  ...      23.000000       6.000000
std                 34.518204  ...       8.658759       2.045789

               Month      Week           Year  \
count  197428.000000  197428.0  197428.000000
mean        1.653170  5.903712    2014.999995
min         1.000000       4.0    2014.000000
25%         1.000000       5.0    2015.000000
50%         2.000000       6.0    2015.000000
75%         2.000000       7.0    2015.000000
max        10.000000      42.0    2015.000000
std         0.476345  1.216714       0.002251
```

```
       time_taken_for_delivery_minutes  time_taken_for_delivery_hours
\
count                    197428.000000                  197428.000000

mean                         45.571782                       0.763809

min                           4.950000                       0.061944

25%                          34.800000                       0.580833

50%                          43.733333                       0.730833

75%                          54.783333                       0.917222

max                          85.300000                       1.471111

std                          14.473399                       0.246818


       DeliverySpeedperItem   AvgItemPrice  \
count         197428.000000  197428.000000
mean              21.135179     975.322997
min                0.123885       0.000000
25%               10.670000     647.666667
50%               16.725000     895.000000
75%               27.108333    1195.000000
max               88.250000   14700.000000
std               14.823381     517.244403

       TimeTakenForDelivery_Minutes_Numeric
count                         197428.000000
mean                              45.571782
min                                4.950000
25%                               34.800000
50%                               43.733333
75%                               54.783333
max                               85.300000
std                               14.473399

[8 rows x 24 columns]
```

```
df=df[df['min_item_price']>=0]
df=df[df['total_onshift_partners']>=0]

'''Here we can see that the no any negative value in our dataset.'''
df.describe()
```

```
           market_id                      created_at  \
count  197394.000000                          197394
mean        2.978571  2015-02-04 22:00:23.545401856
min         1.000000             2014-10-19 05:24:15
```

```
25%        2.000000  2015-01-29 02:32:43.750000128
50%        3.000000     2015-02-05 03:29:12.500000
75%        4.000000  2015-02-12 01:39:32.249999872
max        6.000000            2015-02-18 06:00:44
std        1.524680                            NaN

                actual_delivery_time  order_protocol    total_items  \
count                         197394   197394.000000  197394.000000
mean   2015-02-04 22:48:13.818221312        2.882463       3.195700
min              2015-01-21 15:58:11        1.000000       1.000000
25%              2015-01-29 03:22:25        1.000000       2.000000
50%       2015-02-05 04:40:29.500000        3.000000       3.000000
75%              2015-02-12 02:25:47        4.000000       4.000000
max              2015-02-19 22:45:31        7.000000     411.000000
std                              NaN        1.503808       2.663999

            subtotal  num_distinct_items  min_item_price
max_item_price  \
count  197394.000000       197394.000000   197394.000000
197394.000000
mean     2682.372863            2.670679      686.260509
1159.638459
min         0.000000            1.000000        0.000000
0.000000
25%      1400.000000            1.000000      299.000000
800.000000
50%      2200.000000            2.000000      595.000000
1095.000000
75%      3395.000000            3.000000      949.000000
1395.000000
max     27100.000000           20.000000    14700.000000
14700.000000
std      1823.126645            1.630147      522.025047
558.407462

       total_onshift_partners  ...      HourOfDay      DayOfWeek  \
count           197394.000000  ...  197394.000000  197394.000000
mean                44.832204  ...       8.466422       3.218958
min                  0.000000  ...       0.000000       0.000000
25%                 17.000000  ...       2.000000       1.000000
50%                 37.000000  ...       3.000000       3.000000
75%                 65.000000  ...      19.000000       5.000000
max                171.000000  ...      23.000000       6.000000
std                 34.517407  ...       8.658576       2.045744

               Month      Week           Year  \
count  197394.000000  197394.0  197394.000000
mean        1.653161  5.903741    2014.999995
min         1.000000       4.0    2014.000000
25%         1.000000       5.0    2015.000000
```

```
50%          2.000000       6.0    2015.000000
75%          2.000000       7.0    2015.000000
max         10.000000      42.0    2015.000000
std          0.476348  1.216736       0.002251

       time_taken_for_delivery_minutes  time_taken_for_delivery_hours
\
count                      197394.000000                  197394.000000

mean                           45.571176                       0.763800

min                             4.950000                       0.061944

25%                            34.800000                       0.580833

50%                            43.733333                       0.730833

75%                            54.783333                       0.917222

max                            85.300000                       1.471111

std                            14.473465                       0.246820


       DeliverySpeedperItem   AvgItemPrice  \
count         197394.000000  197394.000000
mean              21.135660     975.356363
min                0.123885       0.000000
25%               10.670833     647.783333
50%               16.725000     895.000000
75%               27.108333    1195.000000
max               88.250000   14700.000000
std               14.823199     517.201870

       TimeTakenForDelivery_Minutes_Numeric
count                         197394.000000
mean                              45.571176
min                                4.950000
25%                               34.800000
50%                               43.733333
75%                               54.783333
max                               85.300000
std                               14.473465

[8 rows x 24 columns]
```

Categorical Feature Encoding

```
#How can we encode categorical variables like store_primary_category
and order_protocol for further analysis?
```

```
Categorical_df=df[['store_primary_category','order_protocol']]
Encoded_categorical_df=pd.get_dummies(Categorical_df, drop_first=True)
Encoded_categorical_df
```

```
        order_protocol  store_primary_category_african  \
0                  1.0                           False
1                  2.0                           False
2                  1.0                           False
3                  1.0                           False
4                  1.0                           False
...                ...                             ...
197423             4.0                           False
197424             4.0                           False
197425             4.0                           False
197426             1.0                           False
197427             1.0                           False

        store_primary_category_alcohol  \
0                                False
1                                False
2                                False
3                                False
4                                False
...                                ...
197423                           False
197424                           False
197425                           False
197426                           False
197427                           False

        store_primary_category_alcohol-plus-food  \
0                                          False
1                                          False
2                                          False
3                                          False
4                                          False
...                                          ...
197423                                     False
197424                                     False
197425                                     False
197426                                     False
197427                                     False

        store_primary_category_american
store_primary_category_argentine  \
0                                   True
False
1                                  False
False
2                                   True
```

```
False
3                                  True
False
4                                  True
False
...                                 ...
...
197423                            False
False
197424                            False
False
197425                            False
False
197426                            False
False
197427                            False
False

        store_primary_category_asian  store_primary_category_barbecue
\
0                              False                            False

1                              False                            False

2                              False                            False

3                              False                            False

4                              False                            False

...                              ...                              ...

197423                         False                            False

197424                         False                            False

197425                         False                            False

197426                         False                            False

197427                         False                            False


        store_primary_category_belgian
store_primary_category_brazilian  ...  \
0                                False
False  ...
1                                False
False  ...
2                                False
```

```
False  ...
3                                 False
False  ...
4                                 False
False  ...
...                                 ...                                 .
..  ...
197423                            False
False  ...
197424                            False
False  ...
197425                            False
False  ...
197426                            False
False  ...
197427                            False
False  ...

        store_primary_category_southern
store_primary_category_spanish  \
0                                  False
False
1                                  False
False
2                                  False
False
3                                  False
False
4                                  False
False
...                                  ...                             ..
.
197423                             False
False
197424                             False
False
197425                             False
False
197426                             False
False
197427                             False
False

        store_primary_category_steak  store_primary_category_sushi  \
0                              False                         False
1                              False                         False
2                              False                         False
3                              False                         False
4                              False                         False
```

```
...                                ...                              ...
197423                           False                            False
197424                           False                            False
197425                           False                            False
197426                           False                            False
197427                           False                            False

        store_primary_category_tapas  store_primary_category_thai  \
0                              False                        False
1                              False                        False
2                              False                        False
3                              False                        False
4                              False                        False
...                              ...                          ...
197423                         False                        False
197424                         False                        False
197425                         False                        False
197426                         False                        False
197427                         False                        False

        store_primary_category_turkish
store_primary_category_vegan  \
0                                False                        False

1                                False                        False

2                                False                        False

3                                False                        False

4                                False                        False

...                                ...                          ...

197423                           False                        False

197424                           False                        False

197425                           False                        False

197426                           False                        False

197427                           False                        False


        store_primary_category_vegetarian
store_primary_category_vietnamese
0                                   False
False
1                                   False
False
```

```
2                                       False
False
3                                       False
False
4                                       False
False
...                                       ...
...
197423                                  False
False
197424                                  False
False
197425                                  False
False
197426                                  False
False
197427                                  False
False

[197394 rows x 74 columns]
```

Advanced Feature Engineering

```
#Can we create a feature based on the availability of delivery
partners, such as a ratio of total_busy_partners to
total_onshift_partners?

df['AvailabilityofDeliveryPartners']=np.where(df['total_onshift_partne
rs']!=0,

df['total_busy_partners']/df['total_onshift_partners'],np.nan)

df['AvailabilityofDeliveryPartners']

0         0.424242
1         2.000000
2         0.000000
3         1.000000
4         1.000000
            ...
197423    1.000000
197424    0.916667
197425    1.051282
197426    1.000000
197427    1.000000
Name: AvailabilityofDeliveryPartners, Length: 197394, dtype: float64

#How do engineered features like order time of day or week enhance the
predictive power or insights of the analysis?
```

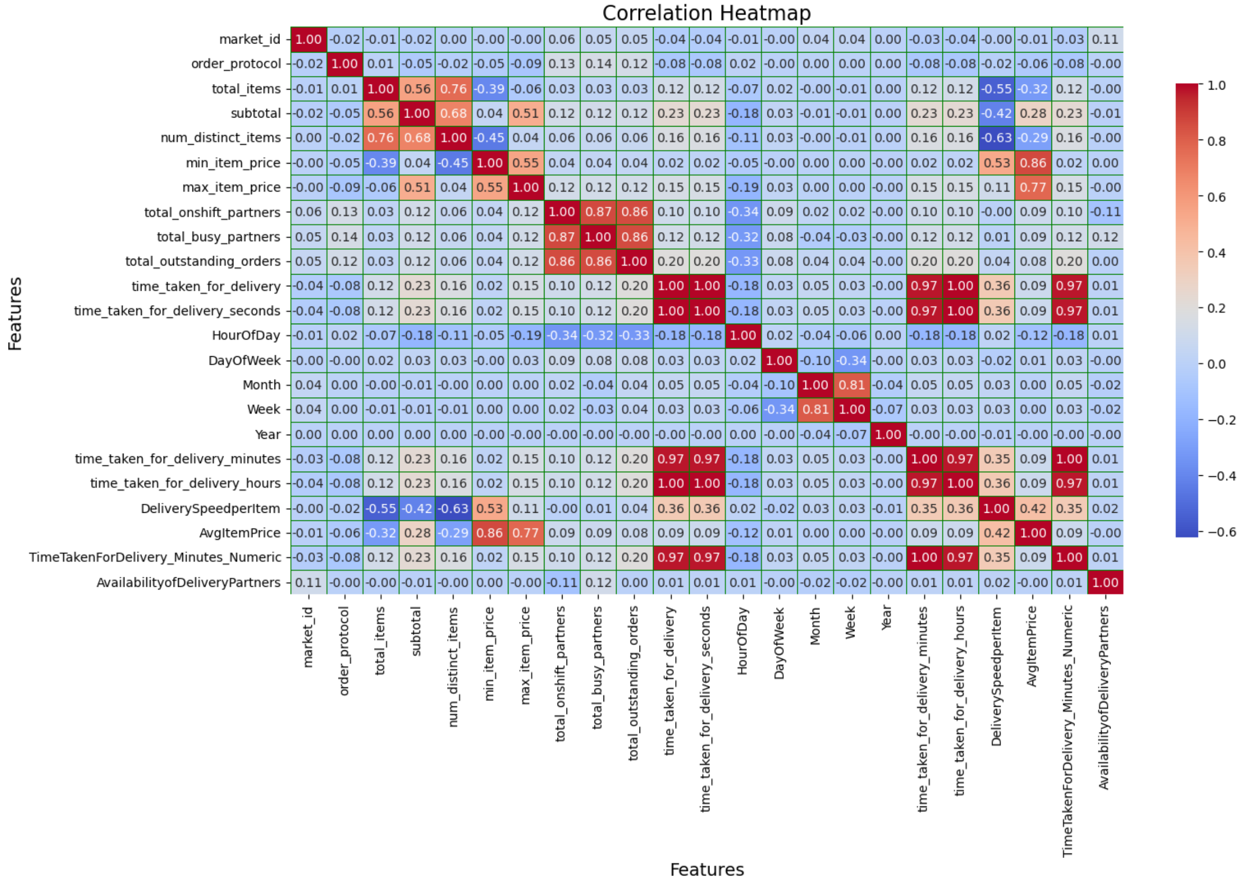
```python
df['OrderTimeofDay']=df['created_at'].dt.time
print(df['OrderTimeofDay'])
print('='*80)

print(df['OrderTimeofDay'].value_counts())
print('='*80)

''' Since we already made dayOfWeek so don't need to do again'''
print(df['DayOfWeek'])
print('='*80)
print(df['DayOfWeek'].value_counts())
```

```
0         22:24:17
1         21:49:25
2         20:39:28
3         21:21:45
4         02:40:36
            ...
197423    00:19:41
197424    00:01:59
197425    04:46:08
197426    18:18:15
197427    19:24:33
Name: OrderTimeofDay, Length: 197394, dtype: object
======================================================================
==========
OrderTimeofDay
02:27:40    23
02:12:02    21
02:10:03    21
02:21:13    21
02:17:37    21
            ..
23:20:25     1
16:29:43     1
17:05:03     1
16:38:32     1
17:50:23     1
Name: count, Length: 46074, dtype: int64
======================================================================
==========
0         4
1         1
2         3
3         1
4         6
         ..
197423    1
197424    4
197425    5
```

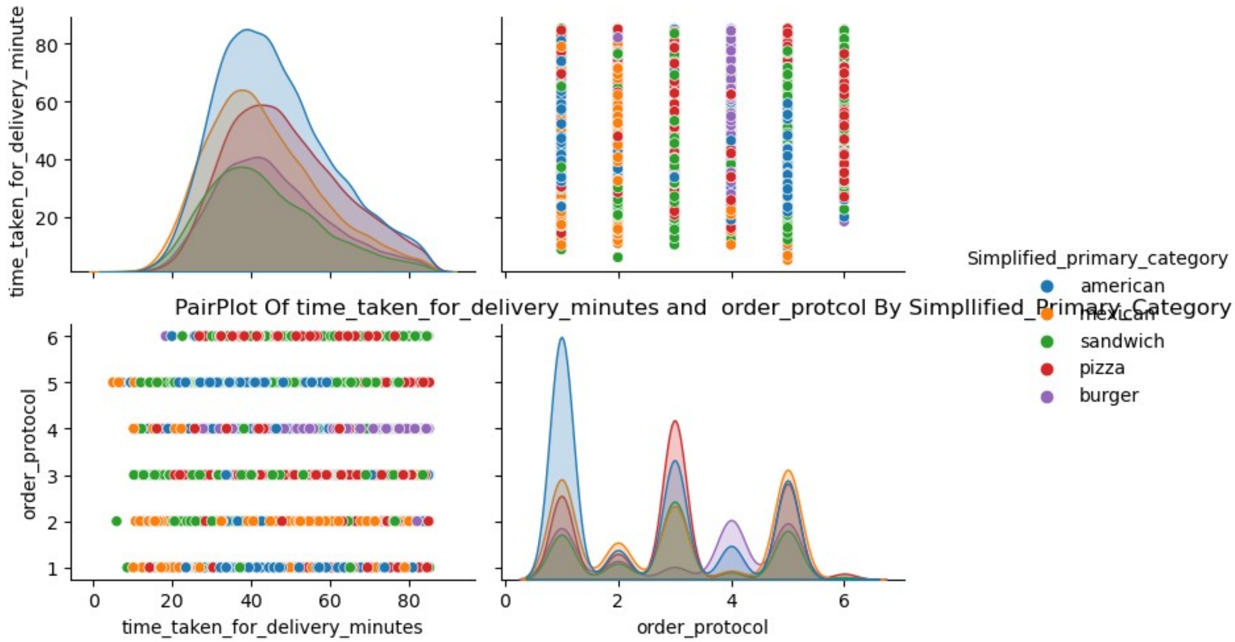
```
197426    6
197427    6
Name: DayOfWeek, Length: 197394, dtype: int32
======================================================================
==========
DayOfWeek
5    34535
6    33611
4    27874
0    27397
3    25670
2    24249
1    24058
Name: count, dtype: int64
```

Advanced Visualization

```
#Use advanced visualization techniques (e.g., heatmaps, pair plots) to
explore relationships between multiple variables simultaneously.

plt.figure(figsize=(15,10))
numeric_df=df.select_dtypes([np.number])
corr=numeric_df.corr()

sns.heatmap(corr,annot=True,cmap='coolwarm',fmt='0.2f',linewidths=0.5,
linecolor='green',cbar_kws={'shrink':0.8},annot_kws={'size':10})
plt.title("Correlation Heatmap", fontsize = 16)
plt.xlabel("Features", fontsize = 14)
plt.ylabel("Features", fontsize = 14)
plt.tight_layout()
plt.show()
```

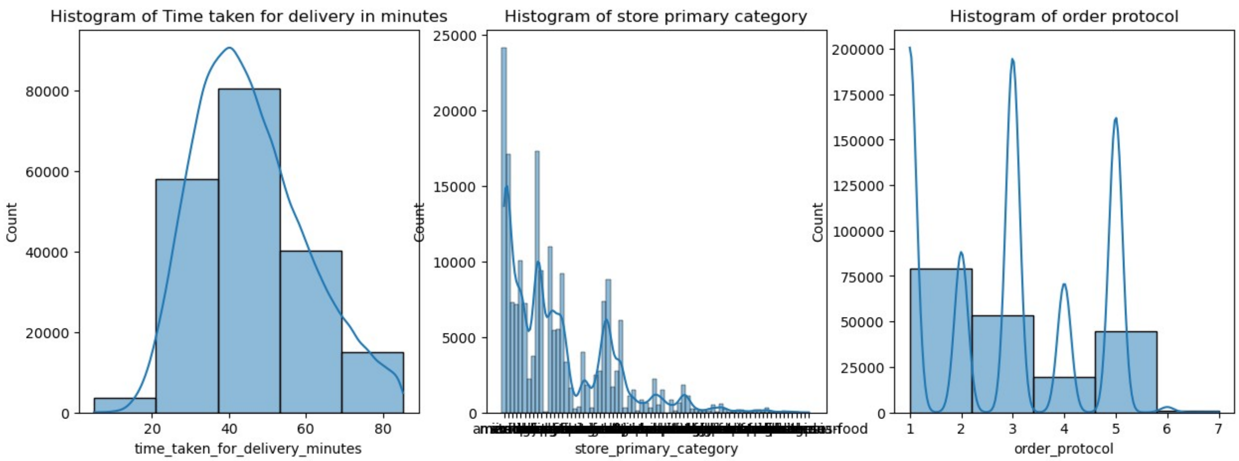



```
#How do interactions between categorical variables (e.g.,
store_primary_category * order_protocol) affect the delivery time?

sns.pairplot(data=filtered_df,vars=['time_taken_for_delivery_minutes',
'order_protocol'],hue='Simplified_primary_category')
plt.title("PairPlot Of time_taken_for_delivery_minutes and
order_protcol By Simpllified_Primary_Category")
plt.tight_layout()
plt.show()
```




Statistical Tests

```
#Trying to find the 'timetakenfordeliveryminutes',
'store_primary_category' and 'order_protocol' and tapers off towards
both sides
fig,axes=plt.subplots(1,3,figsize=(15,5))

sns.histplot(x=df['time_taken_for_delivery_minutes'],kde=True,bins=5,a
x=axes[0])
axes[0].set_title('Histogram of Time taken for delivery in minutes')

sns.histplot(x=df['store_primary_category'],kde=True,bins=5,ax=axes[1]
)
axes[1].set_title('Histogram of store primary category')

sns.histplot(x=df['order_protocol'],kde=True,bins=5,ax=axes[2])
axes[2].set_title('Histogram of order protocol')

Text(0.5, 1.0, 'Histogram of order protocol')
```




```
#Time_taken_delivery_mintues looks normally distributed it's seems
peak around 40 minutes mark and tapers off towards both sides
#Both tails for histogram are releatively balances which is another of
normality

# "store_primary_category" historgram appear quite irregular and not
resemble the normal distribution data contain multiple
# peaks it's shows data are multimodal distribution rather than normal
distribution.

# it's also indicate the data is not normally distributed containing
multiple distict peak representing multimodal distribution.

#TimeTakenfordeliveryminutes might not be normally distributed let's
find out with SHAPIRO TEST

from scipy.stats import shapiro

stat,p=shapiro(df['time_taken_for_delivery_minutes'])
print(f'statics: {stat:.3f},p-value: {p:.3f}')

if p<0.05:
    print("Sample doesn't look normally distruted (reject H0)")
else:
    print("Sample does look normally distruted (fail to reject H0)")

statics: 0.978,p-value: 0.000
Sample doesn't look normally distruted (reject H0)

#Now we confirmed it "TimeTakenForDelivery_Minutes" column doesn't
normally distributed.

#Perform statistical tests to determine if there are significant
differences in delivery times between different groups (e.g.,
different restaurant categories or order protocols).
```

```python
#Since our all three columns are not normally distributed so we can go Kruskal-Wallis test'

from scipy.stats import kruskal
# for store_primary_categeory
kruskal_res=kruskal(
    *(df[df['store_primary_category']==category]
['time_taken_for_delivery_minutes']
      for category in df['store_primary_category'].unique())
)
print(kruskal_res)
print('='*80)

if kruskal_res.pvalue < 0.05:
    print('sample look normally distributed : (fail to reject H0)')
else:
    print("sample doesn't look normally distributed : (reject H0)")
```

```
KruskalResult(statistic=3974.1769352725955, pvalue=0.0)
================================================================================
sample look normally distributed : (fail to reject H0)
```

```python
#performing for order_protocol

from scipy.stats import kruskal

kruskal_res=kruskal(
    *(df[df['order_protocol']==category]
['time_taken_for_delivery_minutes']
      for category in df['order_protocol'].unique())
)
print(kruskal_res)
print('='*80)

if kruskal_res.pvalue < 0.05:
    print('sample look normally distributed : (fail to reject H0)')
else:
    print("sample doesn't look normally distributed : (reject H0)")
```

```
KruskalResult(statistic=1768.440230482327, pvalue=0.0)
================================================================================
sample look normally distributed : (fail to reject H0)
```

```python
import os
os.getcwd()
```

```
'M:\\Porter Case Study'
```

```
df.to_csv("Cleaned Porter Datasets.csv")

df.to_csv("Cleaned Porter Datasets.xlsx")

!pip install openpyxl

Requirement already satisfied: openpyxl in c:\users\anils\anaconda3\lib\site-packages (3.0.10)
Requirement already satisfied: et_xmlfile in c:\users\anils\anaconda3\lib\site-packages (from openpyxl) (1.1.0)

df.to_excel("Cleaned Porter Dataset.xlsx",index=False)
```