Data Understanding

In [1]:

```python
#importing libraries
import matplotlib.pyplot as plt
import seaborn as sns
import warnings
warnings.filterwarnings('ignore')
import numpy as np
import pandas as pd
import boto3

from sklearn.preprocessing import MinMaxScaler
from scipy.stats import norm
from sklearn.preprocessing import StandardScaler
from sklearn.model_selection import train_test_split
from sklearn.preprocessing import LabelEncoder
from sklearn.preprocessing import RobustScaler
from sklearn.metrics import r2_score
from sklearn.metrics import mean_absolute_error
```

In [2]:

```python
#pd.options.display.max_columns = None
pd.set_option('display.max_columns', None)
```

In [3]:

```python
s3_csv_path = f's3://group5-porter-delivery-estimation/data/dataset.csv'

df = pd.read_csv(s3_csv_path)
```

In [4]:

```python
#Information of the data
print(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
None
```

In [5]:

```python
#First 3 rows of table

df.loc[:df.index[2]]
```

Out[5]:

	market_id	created_at	actual_delivery_time	store_id	store_primary_c
0	1.0	2015-02-06 22:24:17	2015-02-06 23:27:16	df263d996281d984952c07998dc54358	a
1	2.0	2015-02-10 21:49:25	2015-02-10 22:56:29	f0ade77b43923b38237db569b016ba25	
2	3.0	2015-01-22 20:39:28	2015-01-22 21:09:09	f0ade77b43923b38237db569b016ba25	

In [6]:

```python
#year,month and day for 'created_at'

df[['year', 'month', 'day']] = df['created_at'].str.split('-', expand=True)
df['day'] = df['day'].str.split(' ', expand=True)[0]
df[['year', 'month', 'day']] = df[['year', 'month', 'day']].astype(int)
```

In [7]:

```python
#created_at & actual_delivery_time into date_time format conversion
df['created_at'] = pd.to_datetime(df['created_at'])
df['actual_delivery_time'] = pd.to_datetime(df['actual_delivery_time'])
```

In [8]:

```python
#feature'time_taken(mins)' created to store the time taken for delivery in minutes
df['time_taken(mins)'] = (df['actual_delivery_time'] - df['created_at']).astype('ti
```

In [9]:

```python
#created_at and actual_deivery_time dropping
df = df.drop(columns=['created_at', 'actual_delivery_time'])
```

In [10]:

```python
#make a copy for exploration
df_=df.copy()
```

Exploratory Data Analysis

In [11]:

```python
#missing values percentage in each category
percent_missing = df_.isnull().sum() * 100 / len(df_)
missing_value_df = pd.DataFrame({'%age of missing value': percent_missing})
missing_value_df.index.name = 'feature'
missing_value_df = missing_value_df.reset_index()
```

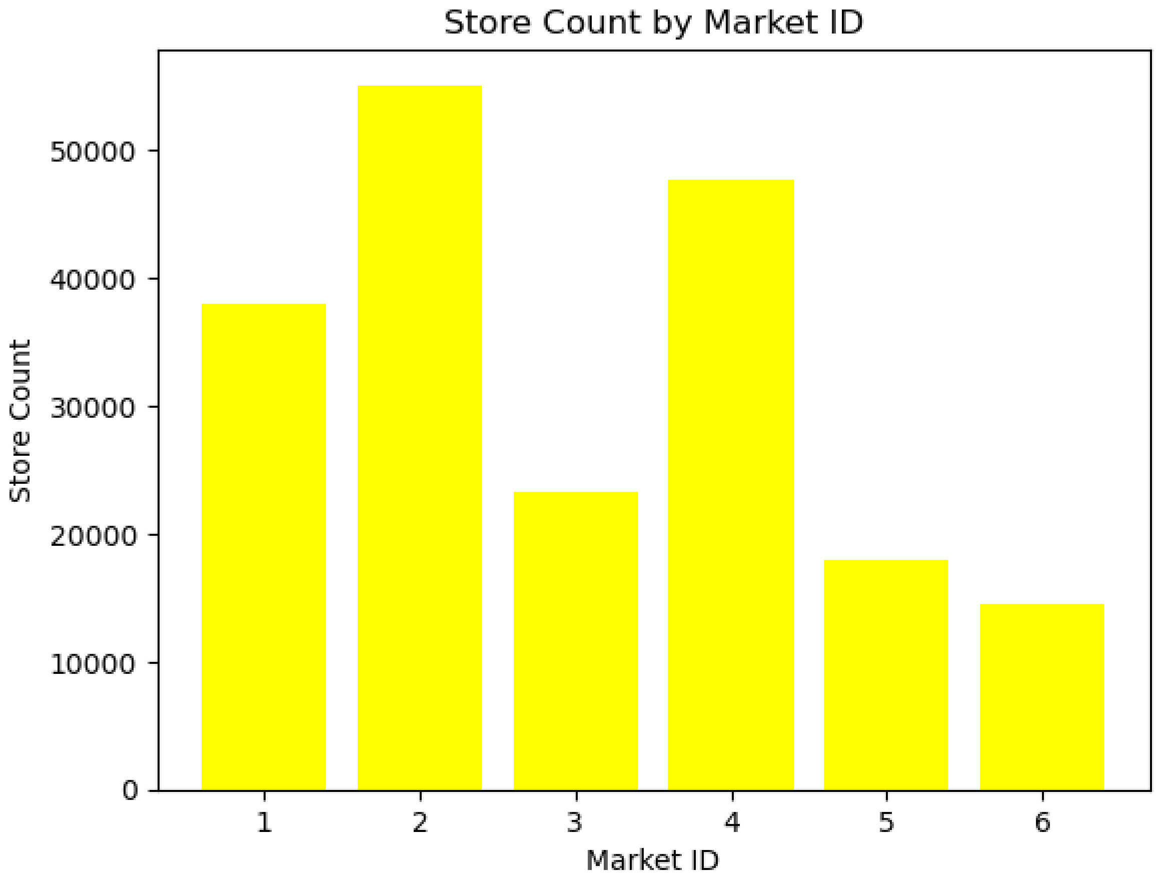
In [12]:

```python
#categorical and numerical features splitting
categorical_feature = []
numerical_feature = []

for col in df.columns:
    if df[col].dtype == 'object' or df[col].dtype == 'category':
        categorical_feature.append(col)
    else:
        numerical_feature.append(col)
```

In [13]:

```
# total number of order from each market
store_count = df_.groupby('market_id')['store_id'].count()
plt.bar(store_count.index, store_count.values, color='yellow')
plt.xlabel('Market ID')
plt.ylabel('Store Count')
plt.title('Store Count by Market ID')
plt.show()
```

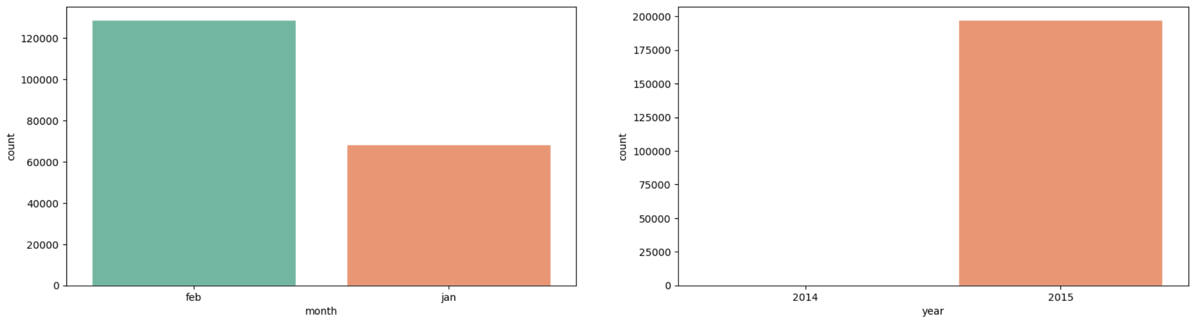


In [14]:

```
#Analysing what was the frequency of orders in each month and in year 2014 and 2015
df_['month']=df_['month'].map({1:'jan',2:'feb',3:'oct'})
fig=plt.figure(figsize=(20,5))
ax=[None for _ in range(2)]
ax[0]=plt.subplot2grid((1,2),(0,0))
ax[1]=plt.subplot2grid((1,2),(0,1))
sns.set_style('darkgrid')
sns.countplot(x='month',data=df_,palette='Set2',ax=ax[0])
sns.countplot(x='year',data=df_,palette='Set2',ax=ax[1])
```

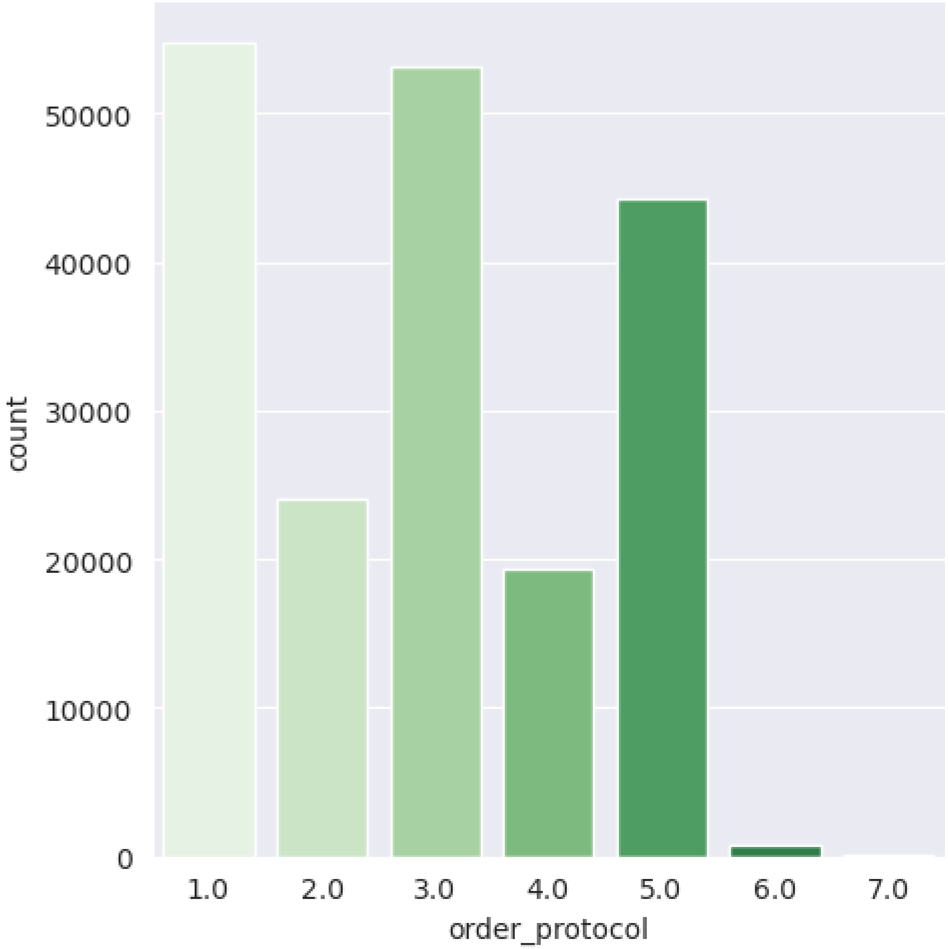
Out[14]:

```
<AxesSubplot: xlabel='year', ylabel='count'>
```

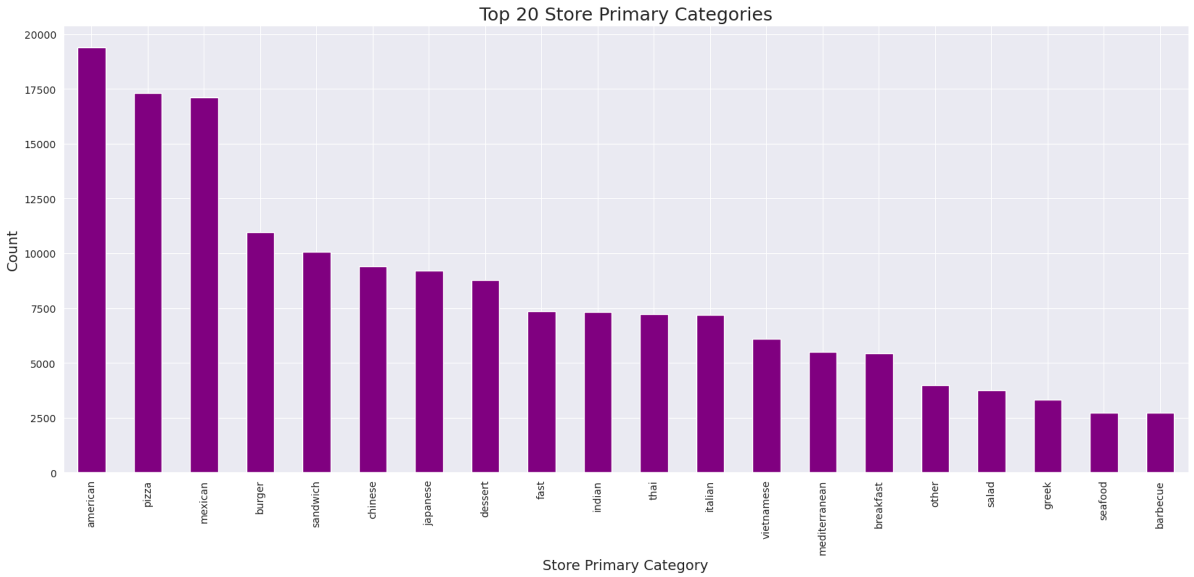


In [15]:
```
#ways in which different orders are placed(i.e., order protocol with most number of
sns.catplot(x='order_protocol', kind='count', data=df_, palette='Greens')
```

Out[15]:
```
<seaborn.axisgrid.FacetGrid at 0x7fe0e586a0e0>
```

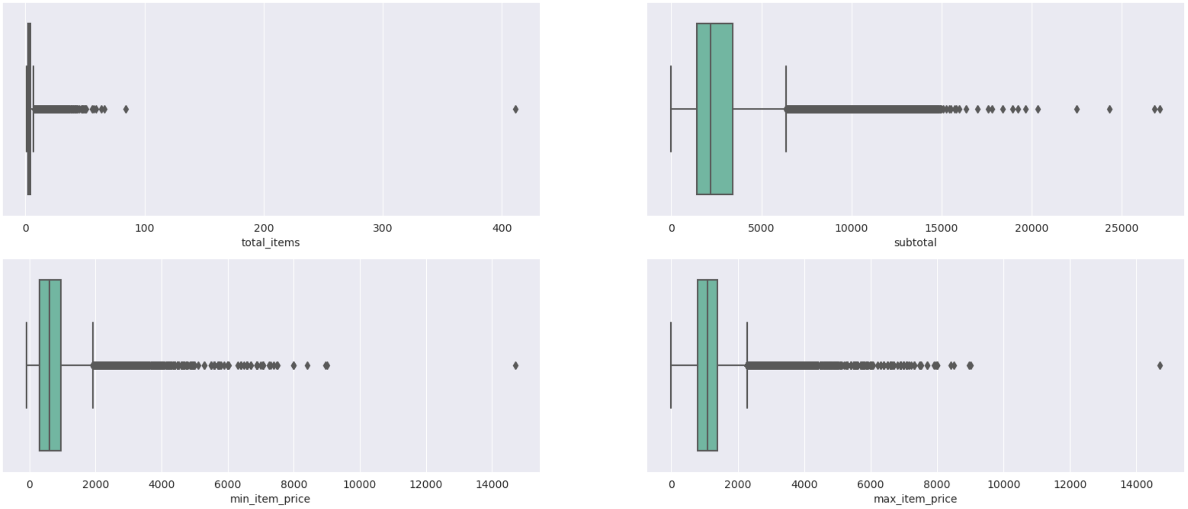


In [16]:
```
#top 20 i.e., most ordered category
fig, ax = plt.subplots(figsize=(20, 8))
df_['store_primary_category'].value_counts().sort_values(ascending=False)[:20].plot
ax.set_xlabel('Store Primary Category', fontsize=14)
ax.set_ylabel('Count', fontsize=14)
ax.set_title('Top 20 Store Primary Categories', fontsize=18)
plt.show()
```



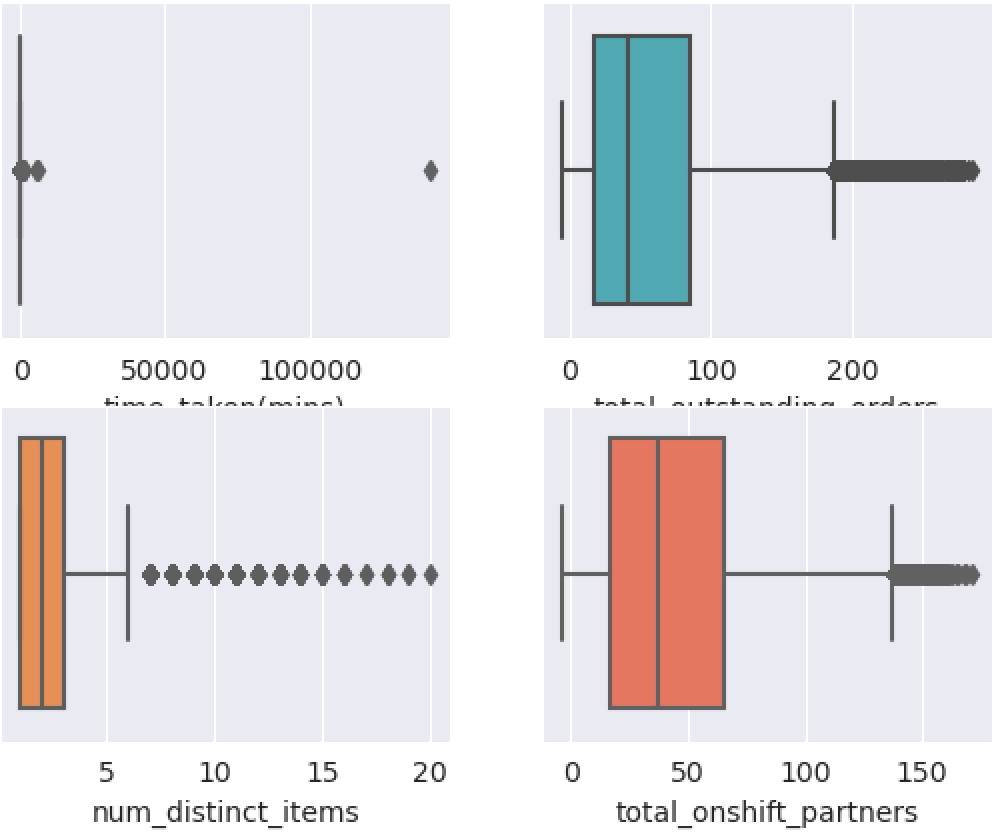
In [17]:
```
#Analysis of the numerical features
fig, axes = plt.subplots(nrows=2, ncols=2, figsize=(20, 8))
sns.boxplot(x='total_items', data=df_, palette='Set2', ax=axes[0, 0])
sns.boxplot(x='subtotal', data=df_, palette='Set2', ax=axes[0, 1])
sns.boxplot(x='min_item_price', data=df_, palette='Set2', ax=axes[1, 0])
sns.boxplot(x='max_item_price', data=df_, palette='Set2', ax=axes[1, 1])
```

Out[17]:
```
<AxesSubplot: xlabel='max_item_price'>
```



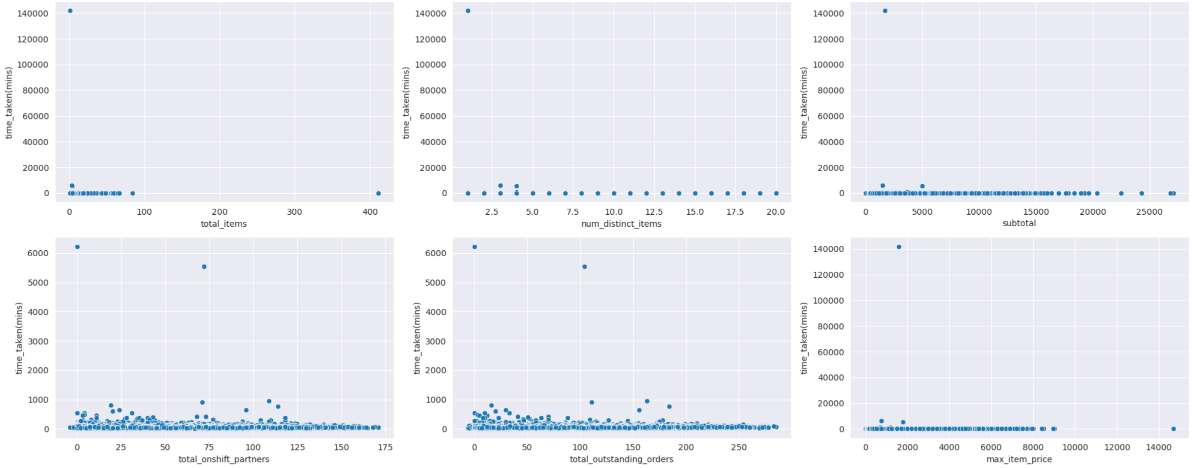
In [18]:
```
#fig=plt.figure(figsize=(20,8))
ax=[None for _ in range(4)]
ax[0]=plt.subplot2grid((2,2),(0,0))
ax[1]=plt.subplot2grid((2,2),(0,1))
ax[2]=plt.subplot2grid((2,2),(1,0))
ax[3]=plt.subplot2grid((2,2),(1,1))
sns.boxplot(x='time_taken(mins)',data=df_,palette='PuBu',ax=ax[0])
sns.boxplot(x='total_outstanding_orders',data=df_,palette='YlGnBu',ax=ax[1])
sns.boxplot(x='num_distinct_items',data=df_,palette='Oranges',ax=ax[2])
sns.boxplot(x='total_onshift_partners',data=df_,palette='Reds',ax=ax[3])
```

Out[18]:
```
<AxesSubplot: xlabel='total_onshift_partners'>
```



In [19]:

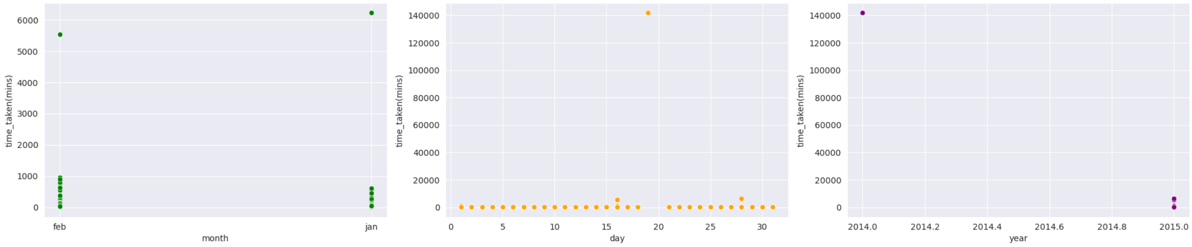
```
#bivariate analysis: to see how the other features are correlated with delivery tim
fig, axes = plt.subplots(nrows=2, ncols=3, figsize=(20, 8))
sns.scatterplot(x='total_items', y='time_taken(mins)', data=df_, ax=axes[0, 0])
sns.scatterplot(x='num_distinct_items', y='time_taken(mins)', data=df_, ax=axes[0,
sns.scatterplot(x='subtotal', y='time_taken(mins)', data=df_, ax=axes[0, 2])
sns.scatterplot(x='total_onshift_partners', y='time_taken(mins)', data=df_, ax=axes
sns.scatterplot(x='total_outstanding_orders', y='time_taken(mins)', data=df_, ax=ax
sns.scatterplot(x='max_item_price', y='time_taken(mins)', data=df_, ax=axes[1, 2])
plt.tight_layout()
```



In [20]:

```
fig=plt.figure(figsize=(20,8))
ax=[None for _ in range(3)]
ax[0]=plt.subplot2grid((2,3),(0,0))
ax[1]=plt.subplot2grid((2,3),(0,1))
ax[2]=plt.subplot2grid((2,3),(0,2))
```

```
sns.scatterplot(x='month',y='time_taken(mins)',data=df_,color='green',ax=ax[0])
sns.scatterplot(x='day',y='time_taken(mins)',data=df_,color='orange',ax=ax[1])
sns.scatterplot(x='year',y='time_taken(mins)',data=df_,color='purple',ax=ax[2])
plt.tight_layout()
```

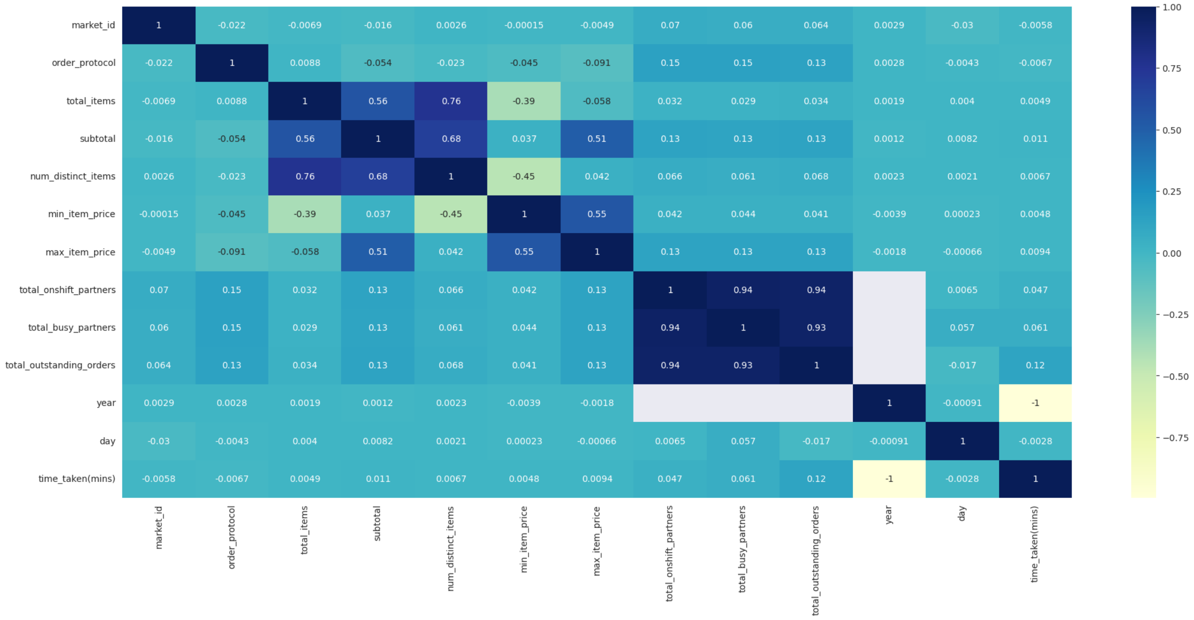


In [21]:

```
#correlation heatmap
plt.figure(figsize=(24,10))
sns.heatmap(df_.corr(), cmap='YlGnBu', annot=True, annot_kws={'size':10})
```

Out[21]:

```
<AxesSubplot: >
```



In [22]:

```
abs_corr = abs(df_.corr()['time_taken(mins)'])
sorted_corr = abs_corr.sort_values(ascending=False)
print(sorted_corr)
```

```
time_taken(mins)            1.000000
year                        0.996474
total_outstanding_orders    0.122261
total_busy_partners         0.060615
total_onshift_partners      0.046952
subtotal                    0.011203
max_item_price              0.009411
num_distinct_items          0.006743
order_protocol              0.006662
market_id                   0.005781
total_items                 0.004906
min_item_price              0.004764
day                         0.002843
Name: time_taken(mins), dtype: float64
```