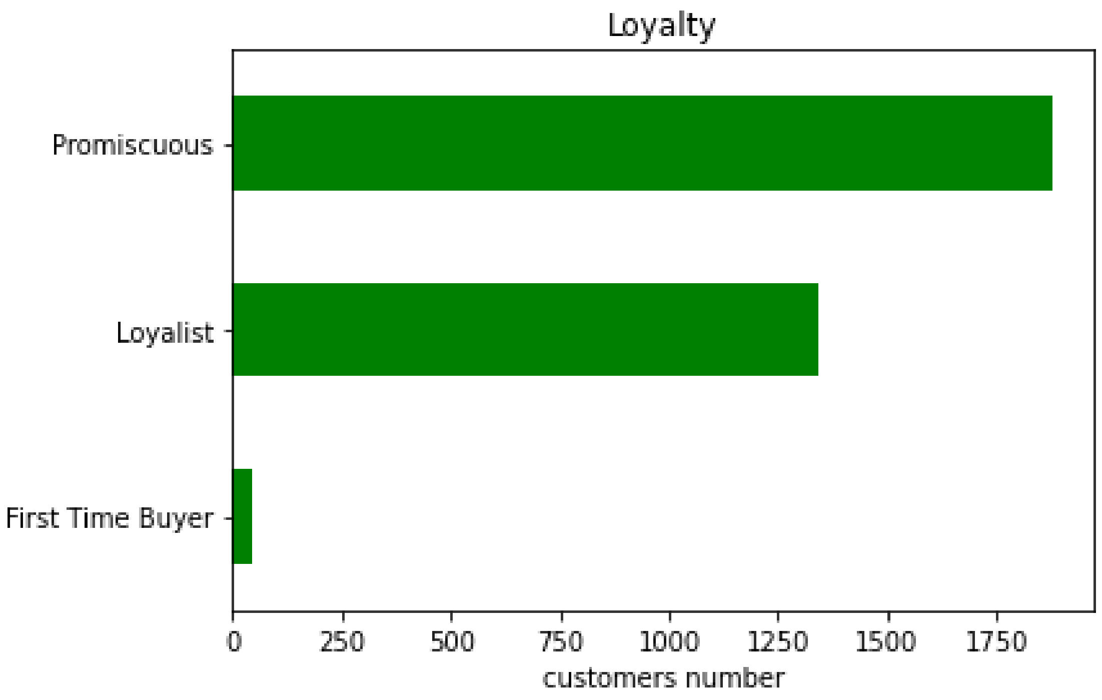
uyencode-Copy1

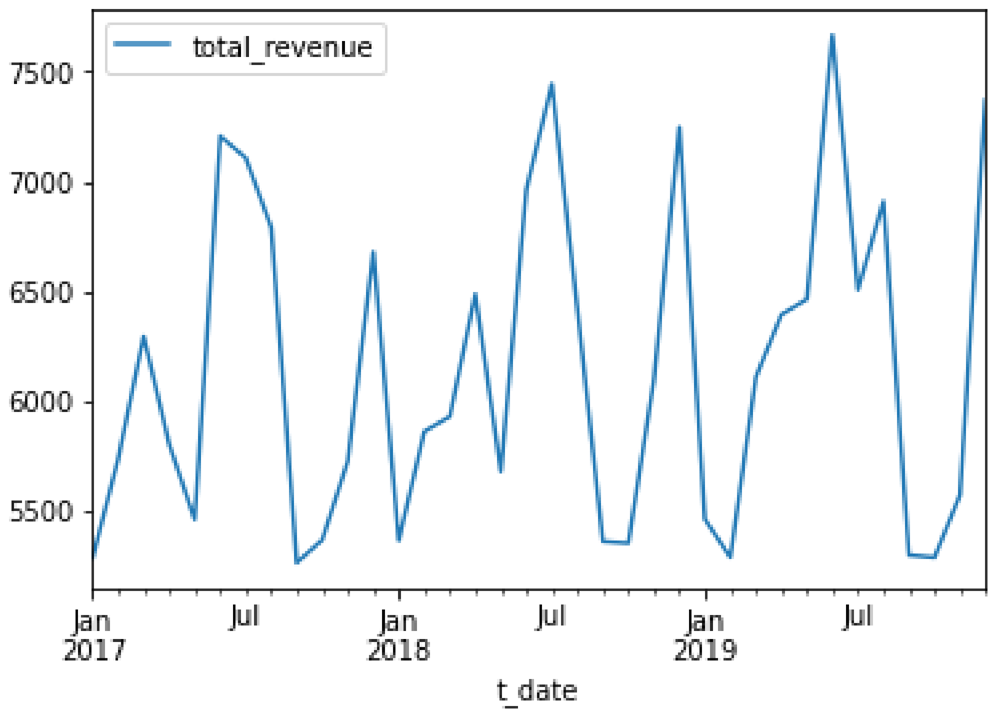
September 24, 2021

```
In [16]: import pandas as pd
         from pandas.plotting import autocorrelation_plot
         import numpy as np
         import matplotlib.pyplot as plt
         import seaborn as sns
         import plotly.graph_objects as go
         from statsmodels.tsa.seasonal import seasonal_decompose #library for time series anal
         from statsmodels.tsa.stattools import adfuller
         from statsmodels.tsa.arima_model import ARIMA
         import statsmodels
         statsmodels.__version__

Out[16]: '0.12.0'

In [22]: df = pd.read_csv('dataset_2017_2020.csv')

In [ ]: df.transaction_date = pd.to_datetime(df.transaction_date) #Have to change to datetime
        df["year"] = df.transaction_date.dt.year
        tmp = df.groupby(['year']).agg(number_baskets=('basket_id', pd.Series.nunique)).reset_
        tmp.head()
        fig = plt.figure() #from this point, we start plotting.
        plt.bar(tmp.year, tmp.number_baskets, color='blue') # color = ['green', 'yellow', 'blu
        plt.xticks(tmp.year)
        plt.xlabel('Year')
        plt.ylabel('Baskets number')
        plt.title('Baskets estimation')
        plt.show();

In [6]: df.groupby(['loyalty', 'transaction_date']).agg(revenue=('price', sum)).reset_index()
        data = []
        for d in df.loyalty.unique():
            tmp = df[df.loyalty==d].groupby(['transaction_date']).agg(revenue=('price', sum)).
            data.append(go.Scatter(x=tmp.transaction_date, y=tmp.revenue, name = d, line=dict(
        go.Figure(
            data=data,
            layout = go.Layout(
                title ='Loyalty trends',
                yaxis=dict(
                    title='Revenue'
                )
            )
        ).show(renderer = 'iframe')
```

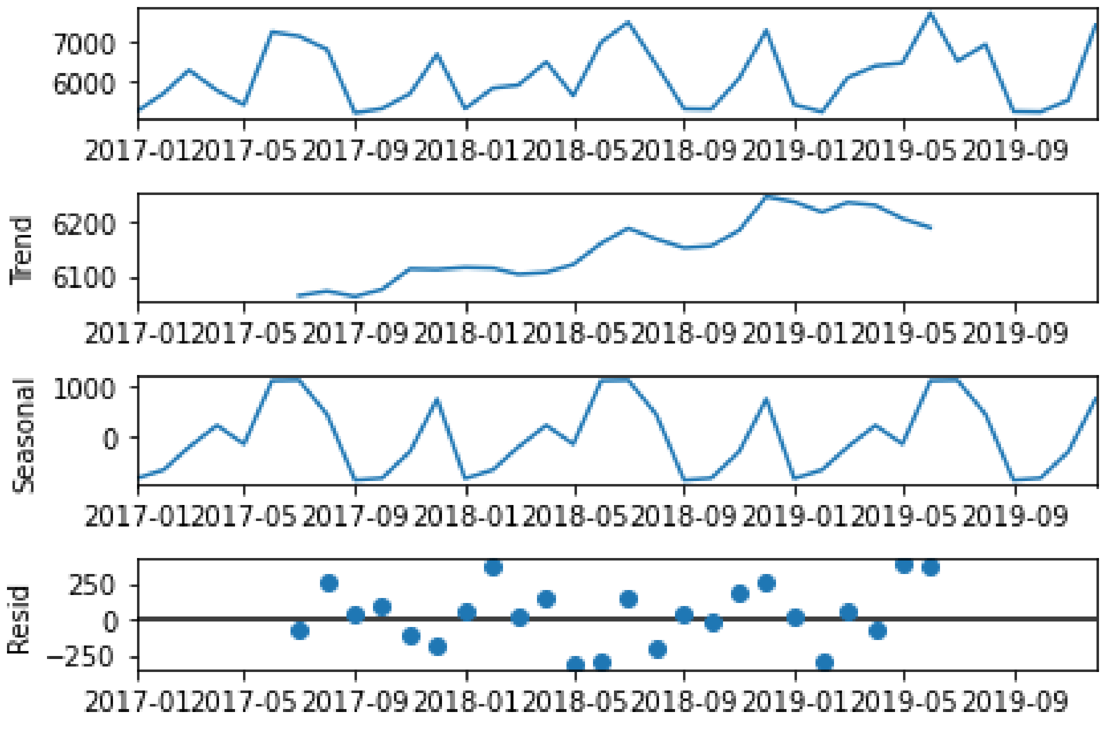
```
In [15]: from matplotlib import pyplot as plt
         df.groupby('loyalty').agg(totals=('customer_id',pd.Series.nunique)) \
             .plot(kind='barh', legend = False, title = 'Loyalty',color='green')
         plt.ylabel('')
         plt.xlabel('customers number');
```



```
In [24]: df['transaction_date'] = df.transaction_date.str[:10] #avoid time in date
         df['t_date'] = pd.to_datetime(df.transaction_date) #convert to date format
         df['t_date'] = df.t_date + pd.offsets.MonthBegin(-1) #send dates to first day of the

In [25]: ts = df.groupby(['t_date']).agg(total_revenue=('price', sum)).reset_index()

In [30]: yearstrendta = ts.loc[ts.t_date < '2020-01-01'].set_index('t_date')
         yearstrendta.shape
         yearstrendta.plot()

Out[30]: <AxesSubplot:xlabel='t_date'>
```



```
In [32]: ts_components = seasonal_decompose(yearstrendta)
         ts_components.plot();
```



```
In [ ]: test_adf = adfuller(yearstrendta)

        #Output the results:
        print('ADF test = ', test_adf[0])
        print('p-value = ', test_adf[1])
```

```
In [ ]: #13
        # Transform our data in a series, where the index is the time series
        whole = ts.set_index('t_date').squeeze().copy()
        # history is going to countain our training data as a time series
        history = whole.take(range(36))
        # future contains the test data, also as a time series
        future = test.squeeze().copy()
```