

Priority Allocation in Railway Reservations: A Focus on Overnight Travelers with Waiting Tickets



```
In [1]: import numpy as np
import pandas as pd
import matplotlib.pyplot as plt
import seaborn as sns
```

```
In [2]: import warnings
warnings.filterwarnings('ignore')
```

```
In [3]: df = pd.read_csv("railway_price_data.csv")
```

```
In [4]: df.head()
```

```
Out[4]: baseFare reservationCharge superfastCharge fuelAmount totalConcession tatkalFare serviceTax otherCharge cateringCharge dynamicCh
```

	baseFare	reservationCharge	superfastCharge	fuelAmount	totalConcession	tatkalFare	serviceTax	otherCharge	cateringCharge	dynamicCh
0	1059	60	0	0	0	0	56	0	0	0
1	626	50	0	0	0	0	34	0	0	0
2	441	40	0	0	0	0	24	0	0	0
3	125	20	0	0	0	0	0	0	0	0
4	1059	60	0	0	0	0	56	0	0	0

```
In [5]: df.tail()
```

	baseFare	reservationCharge	superfastCharge	fuelAmount	totalConcession	tatkalFare	serviceTax	otherCharge	cateringCharge	dyr
326638	786	40	45	0	0	0	44	0	0	0
326639	786	40	45	0	0	0	44	0	0	0
326640	786	40	45	0	0	0	44	0	0	0
326641	786	40	45	0	0	0	44	0	0	0
326642	786	40	45	0	0	0	44	0	0	0

In [6]: `df.shape`

Out[6]: (326643, 19)

In [7]: `df.columns`

Out[7]: Index(['baseFare', 'reservationCharge', 'superfastCharge', 'fuelAmount', 'totalConcession', 'tatkalFare', 'serviceTax', 'otherCharge', 'cateringCharge', 'dynamicFare', 'totalFare', 'availability', 'trainNumber', 'timeStamp', 'fromStnCode', 'toStnCode', 'classCode', 'distance', 'duration'],
dtype='object')

In [8]: `df.duplicated().sum()`

Out[8]: 0

In [9]: `df.isnull().sum()`

Out[9]:

baseFare	0
reservationCharge	0
superfastCharge	0
fuelAmount	0
totalConcession	0
tatkalFare	0
serviceTax	0
otherCharge	0
cateringCharge	0
dynamicFare	0
totalFare	0
availability	0
trainNumber	0
timeStamp	0
fromStnCode	0
toStnCode	0
classCode	0
distance	0
duration	0

dtype: int64

In [10]: `df.info()`

```

<class 'pandas.core.frame.DataFrame'>
RangeIndex: 326643 entries, 0 to 326642
Data columns (total 19 columns):
 #   Column           Non-Null Count  Dtype  
--- 
 0   baseFare        326643 non-null   int64  
 1   reservationCharge 326643 non-null   int64  
 2   superfastCharge 326643 non-null   int64  
 3   fuelAmount       326643 non-null   int64  
 4   totalConcession 326643 non-null   int64  
 5   tatkalFare       326643 non-null   int64  
 6   serviceTax       326643 non-null   int64  
 7   otherCharge      326643 non-null   int64  
 8   cateringCharge   326643 non-null   int64  
 9   dynamicFare      326643 non-null   int64  
 10  totalFare        326643 non-null   int64  
 11  availability     326643 non-null   object  
 12  trainNumber      326643 non-null   int64  
 13  timeStamp         326643 non-null   object  
 14  fromStnCode      326643 non-null   object  
 15  toStnCode         326643 non-null   object  
 16  classCode         326643 non-null   object  
 17  distance          326643 non-null   int64  
 18  duration          326643 non-null   int64  
dtypes: int64(14), object(5)
memory usage: 47.3+ MB

```

In [11]: df.describe()

	baseFare	reservationCharge	superfastCharge	fuelAmount	totalConcession	tatkalFare	serviceTax	otherCharge	cateringCh
count	326643.000000	326643.000000	326643.000000	326643.0	326643.0	326643.0	326643.000000	326643.000000	326643.00
mean	890.075122	38.157805	20.058091	0.0	0.0	0.0	42.761510	0.053208	4.99
std	793.765711	14.324952	24.646072	0.0	0.0	0.0	45.080793	1.018403	42.58
min	30.000000	15.000000	0.000000	0.0	0.0	0.0	0.000000	0.000000	0.00
25%	362.000000	20.000000	0.000000	0.0	0.0	0.0	0.000000	0.000000	0.00
50%	626.000000	40.000000	0.000000	0.0	0.0	0.0	34.000000	0.000000	0.00
75%	1222.000000	50.000000	45.000000	0.0	0.0	0.0	65.000000	0.000000	0.00
max	6541.000000	60.000000	75.000000	0.0	0.0	0.0	334.000000	25.000000	1365.00

In [12]: df.nunique()

baseFare	1840
reservationCharge	5
superfastCharge	5
fuelAmount	1
totalConcession	1
tatkalFare	1
serviceTax	276
otherCharge	3
cateringCharge	69
dynamicFare	475
totalFare	976
availability	27639
trainNumber	533
timeStamp	573
fromStnCode	1480
toStnCode	1594
classCode	6
distance	2759
duration	2912
dtype:	int64

In [13]: object_columns = df.select_dtypes(include=['object', 'bool']).columns
print("Object type columns:")
print(object_columns)

```

numerical_columns = df.select_dtypes(include=['int64', 'float64']).columns
print("\nNumerical type columns:")
print(numerical_columns)

```

```

Object type columns:
Index(['availability', 'timeStamp', 'fromStnCode', 'toStnCode', 'classCode'], dtype='object')

Numerical type columns:
Index(['baseFare', 'reservationCharge', 'superfastCharge', 'fuelAmount',
       'totalConcession', 'tatkalFare', 'serviceTax', 'otherCharge',
       'cateringCharge', 'dynamicFare', 'totalFare', 'trainNumber', 'distance',
       'duration'],
      dtype='object')

```

In [14]: def classify_features(df):
 categorical_features = []

```

non_categorical_features = []
discrete_features = []
continuous_features = []

for column in df.columns:
    if df[column].dtype in ['object', 'bool']:
        if df[column].nunique() < 15:
            categorical_features.append(column)
        else:
            non_categorical_features.append(column)
    elif df[column].dtype in ['int64', 'float64']:
        if df[column].nunique() < 10:
            discrete_features.append(column)
        else:
            continuous_features.append(column)

return categorical_features, non_categorical_features, discrete_features, continuous_features

```

```
In [15]: categorical, non_categorical, discrete, continuous = classify_features(df)
```

```
In [16]: print("Categorical Features:", categorical)
print("Non-Categorical Features:", non_categorical)
print("Discrete Features:", discrete)
print("Continuous Features:", continuous)
```

Categorical Features: ['classCode']
 Non-Categorical Features: ['availability', 'timeStamp', 'fromStnCode', 'toStnCode']
 Discrete Features: ['reservationCharge', 'superfastCharge', 'fuelAmount', 'totalConcession', 'tatkalFare', 'otherCharge']
 Continuous Features: ['baseFare', 'serviceTax', 'cateringCharge', 'dynamicFare', 'totalFare', 'trainNumber', 'distance', 'duration']

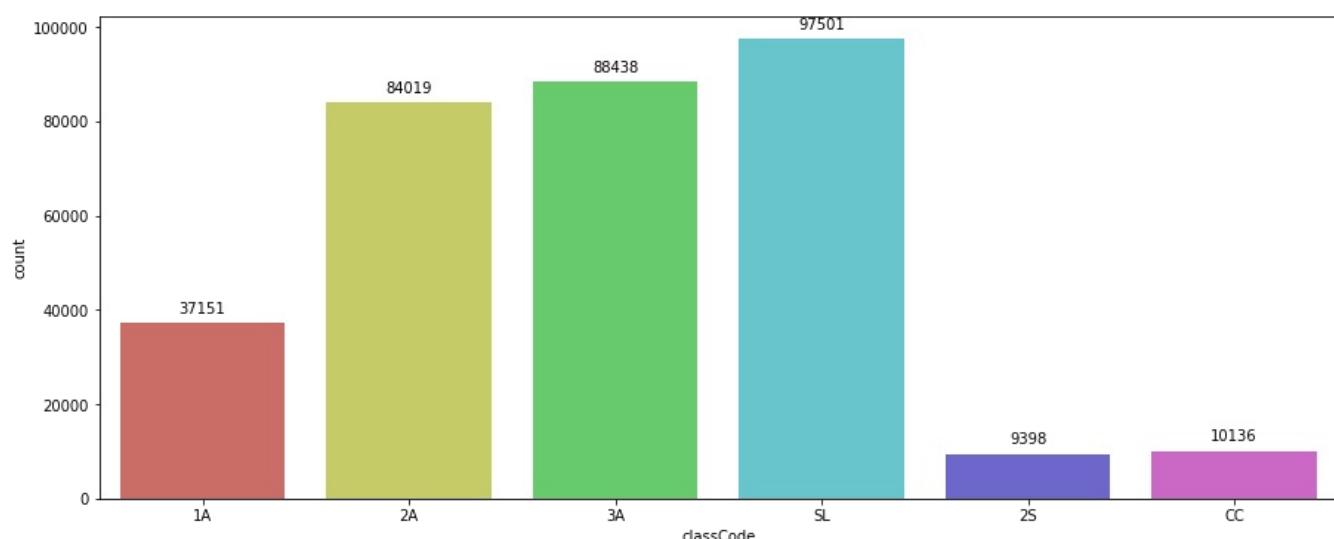
```
In [17]: for i in categorical:
    print(i, ':')
    print(df[i].unique())
    print()
```

classCode :
 ['1A' '2A' '3A' 'SL' '2S' 'CC']

```
In [18]: for i in categorical:
    print(i, ':')
    print(df[i].value_counts())
    print()
```

classCode :
 SL 97501
 3A 88438
 2A 84019
 1A 37151
 CC 10136
 2S 9398
 Name: classCode, dtype: int64

```
In [19]: for i in categorical:
    plt.figure(figsize=(15, 6))
    ax = sns.countplot(x=i, data=df, palette='hls')
    for p in ax.patches:
        ax.annotate(f'{p.get_height()}', (p.get_x() + p.get_width() / 2., p.get_height()),
                    ha='center', va='center', xytext=(0, 10), textcoords='offset points')
    plt.show()
```



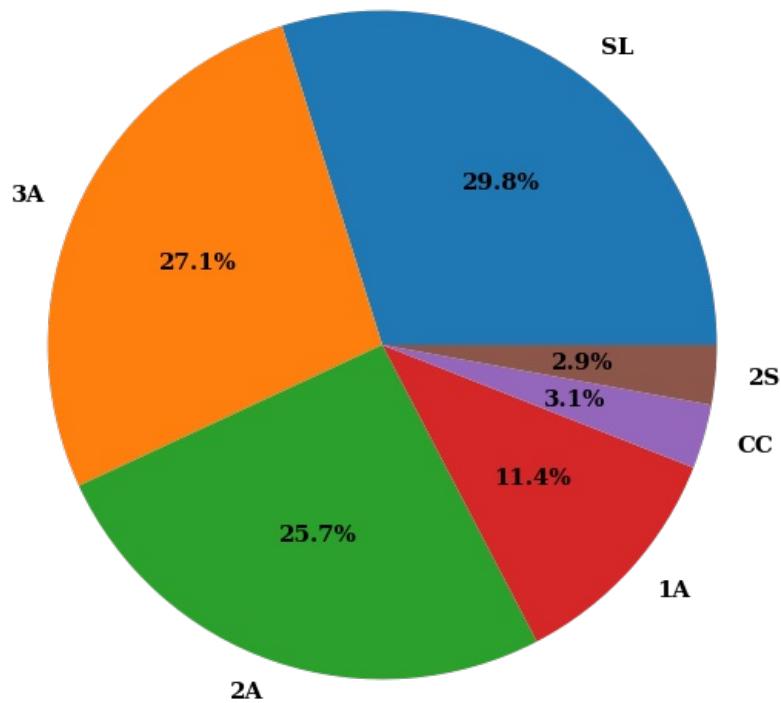
```
In [20]: for i in categorical:
```

```

plt.figure(figsize=(20,10))
plt.pie(df[i].value_counts(), labels=df[i].value_counts().index, autopct='%1.1f%%', textprops={'fontsize':
    'color': 'black',
    'weight': 'bold',
    'family': 'serif' })
hfont = {'fontname':'serif', 'weight': 'bold'}
plt.title(i, size=20, **hfont)
plt.show()

```

classCode



```
In [21]: for i in discrete:
    print(i, ':')
    print(df[i].unique())
    print()
```

```
reservationCharge :
[60 50 40 20 15]
```

```
superfastCharge :
[ 0 45 30 15 75]
```

```
fuelAmount :
[0]
```

```
totalConcession :
[0]
```

```
tatkalFare :
[0]
```

```
otherCharge :
[ 0 25 15]
```

```
In [22]: for i in discrete:
    print(i, ':')
    print(df[i].value_counts())
    print()
```

```

reservationCharge :
40    98573
20    97501
50    84019
60    37152
15    9398
Name: reservationCharge, dtype: int64

superfastCharge :
0     181900
45    80829
30    35932
75    23614
15    4368
Name: superfastCharge, dtype: int64

fuelAmount :
0    326643
Name: fuelAmount, dtype: int64

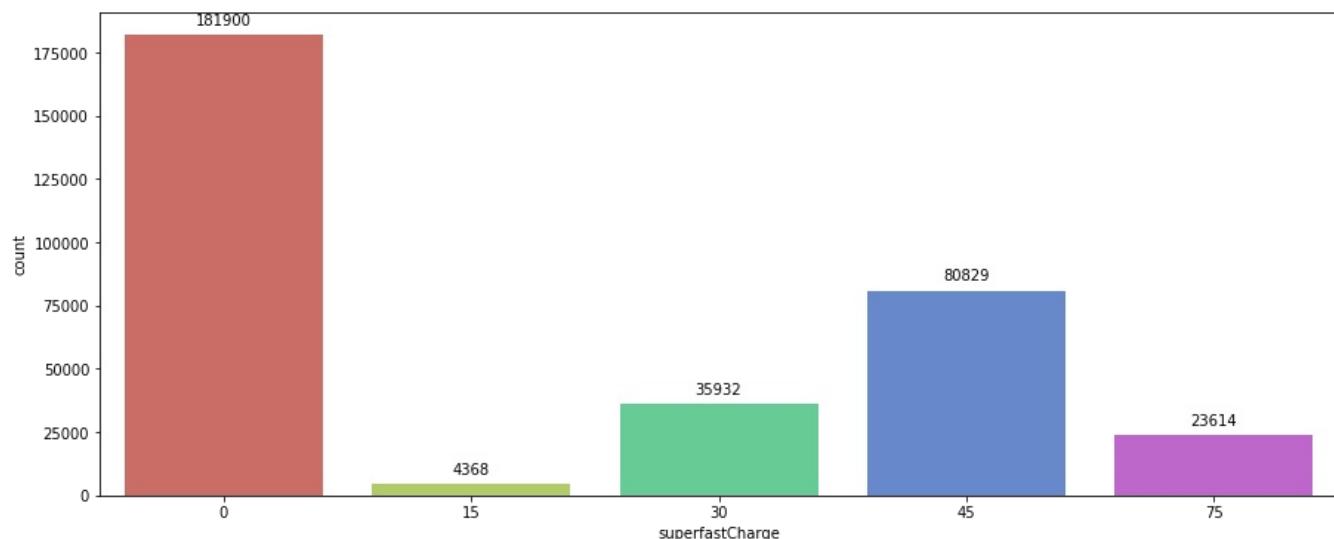
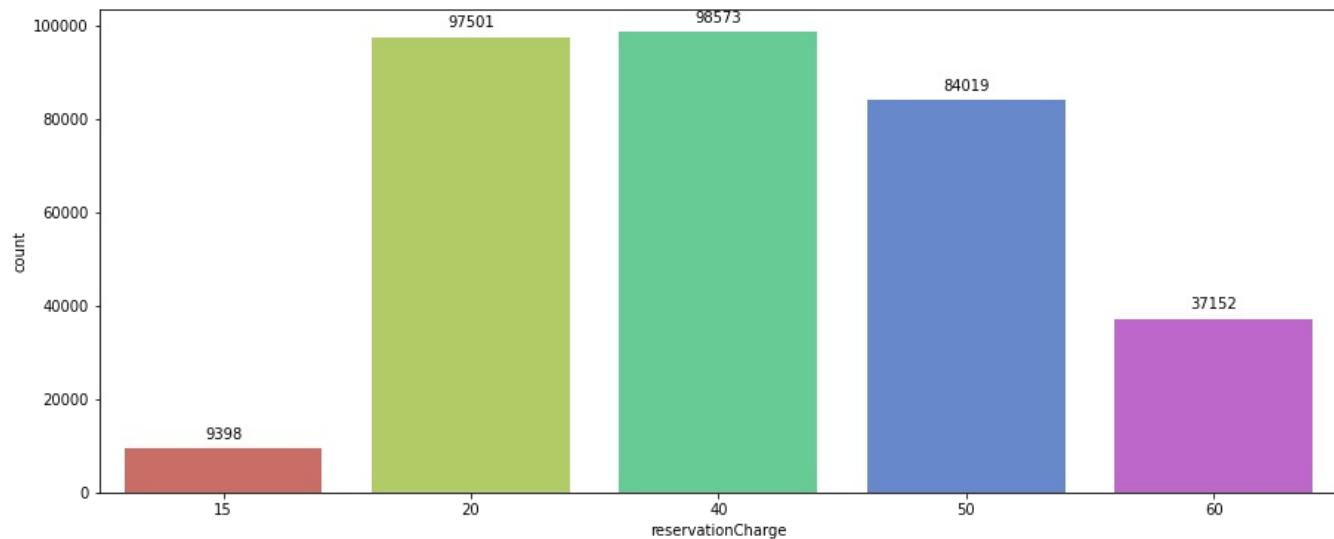
totalConcession :
0    326643
Name: totalConcession, dtype: int64

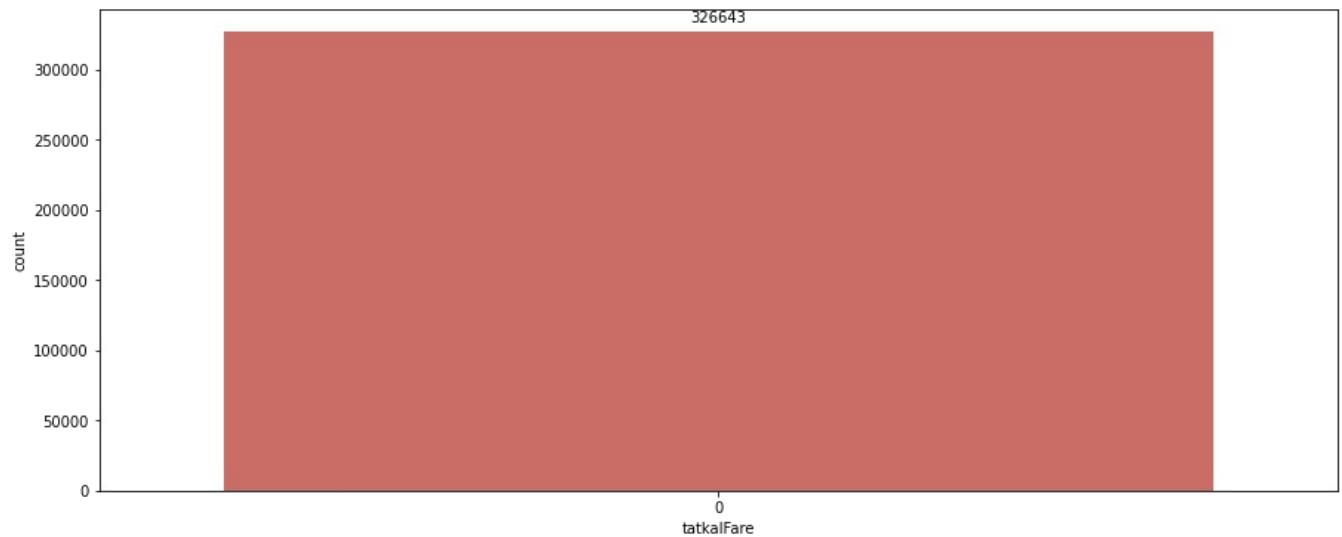
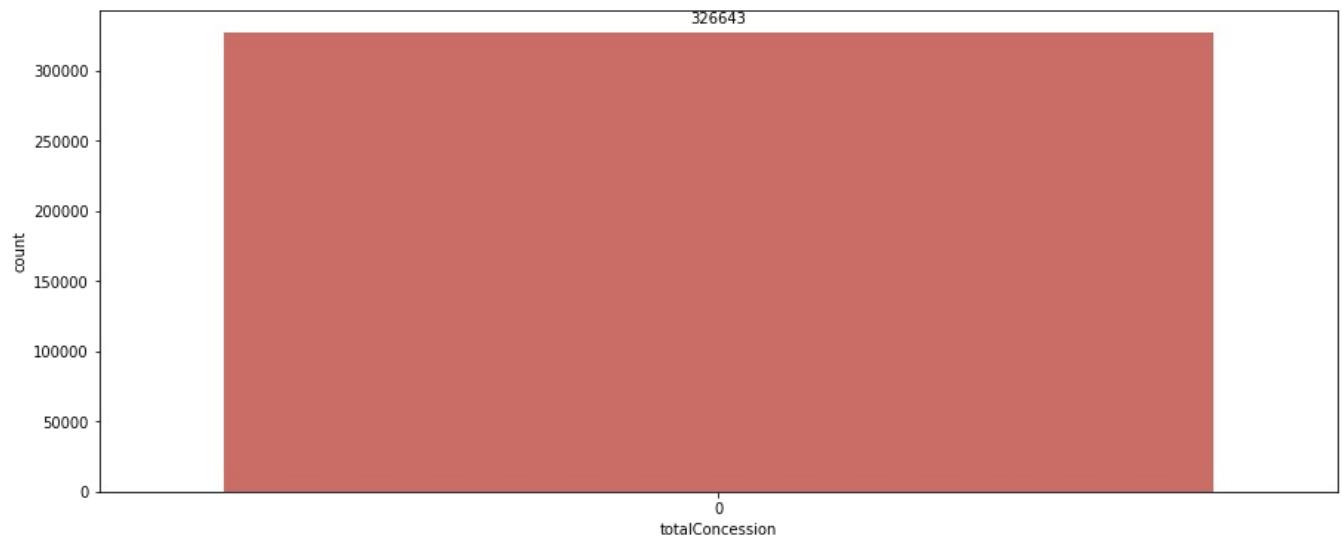
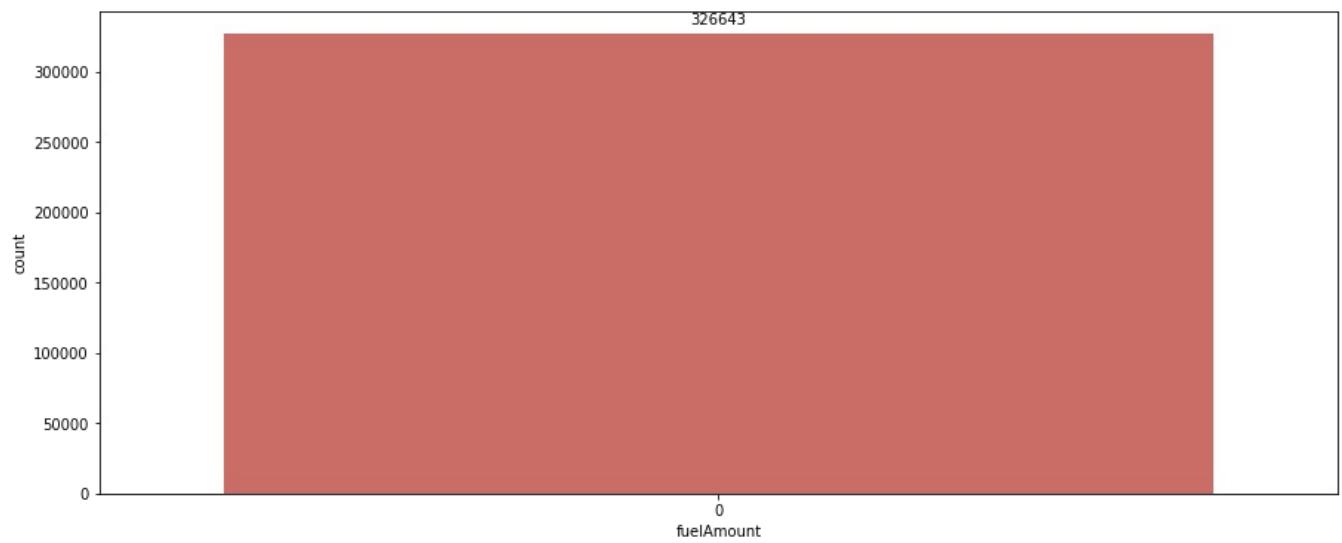
tatkalFare :
0    326643
Name: tatkalFare, dtype: int64

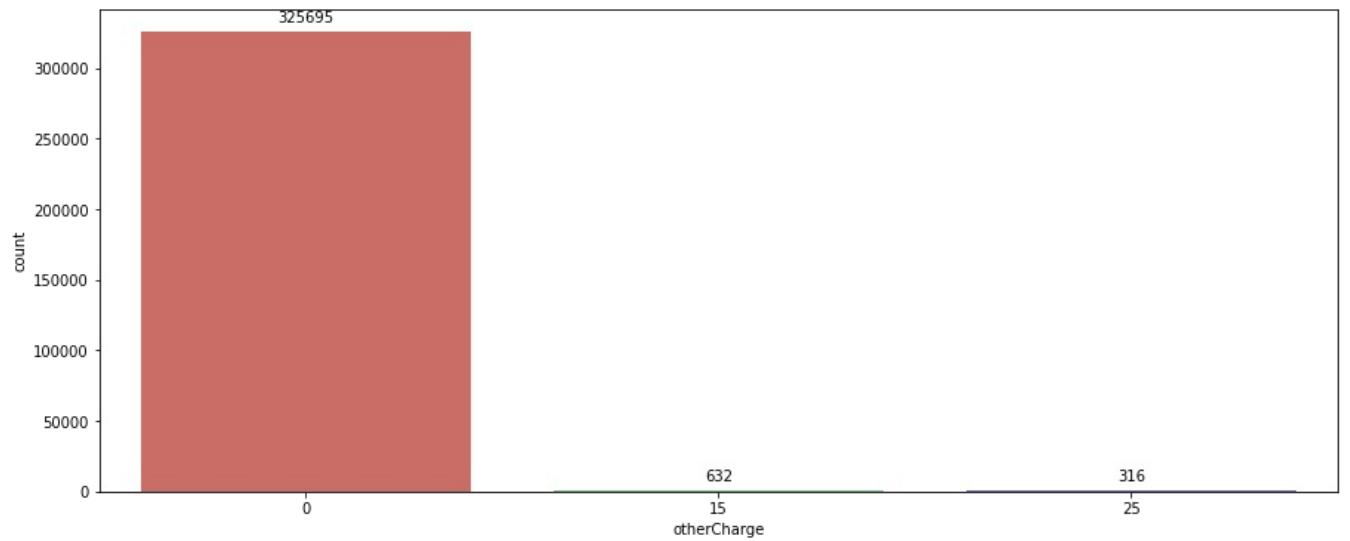
otherCharge :
0    325695
15   632
25   316
Name: otherCharge, dtype: int64

```

```
In [23]: for i in discrete:
    plt.figure(figsize=(15, 6))
    ax = sns.countplot(x=i, data=df, palette='hls')
    for p in ax.patches:
        ax.annotate(f'{p.get_height()}', (p.get_x() + p.get_width() / 2., p.get_height()),
                    ha='center', va='center', xytext=(0, 10), textcoords='offset points')
    plt.show()
```

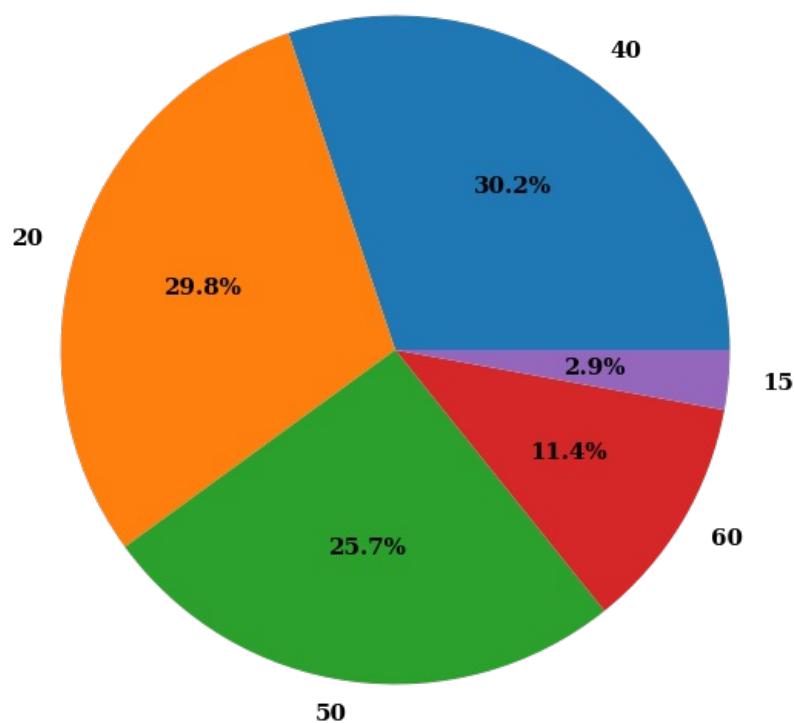




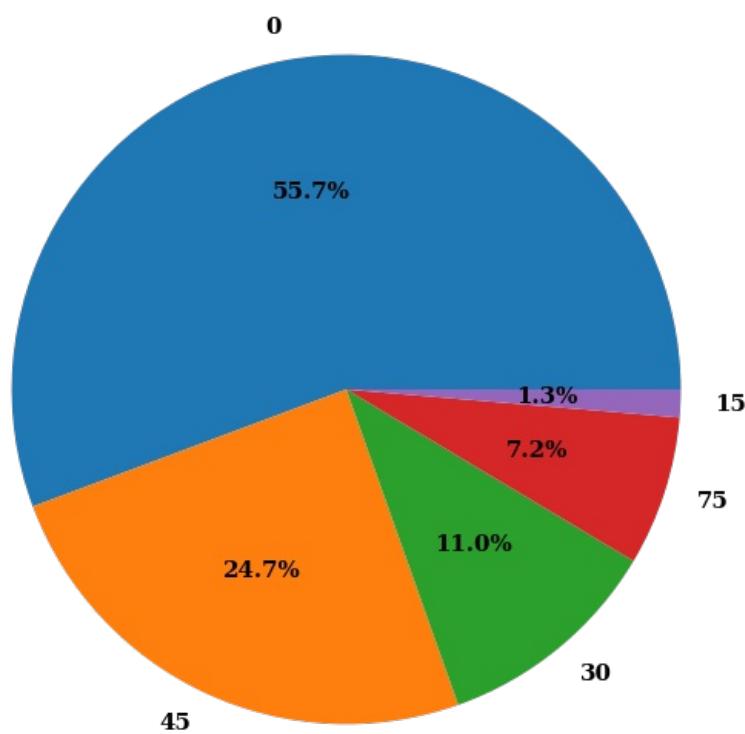


```
In [24]: for i in discrete:
    plt.figure(figsize=(20,10))
    plt.pie(df[i].value_counts(), labels=df[i].value_counts().index, autopct='%1.1f%%', textprops={'fontsize':
        'color': 'black',
        'weight': 'bold',
        'family': 'serif' })
    hfont = {'fontname':'serif', 'weight': 'bold'}
    plt.title(i, size=20, **hfont)
    plt.show()
```

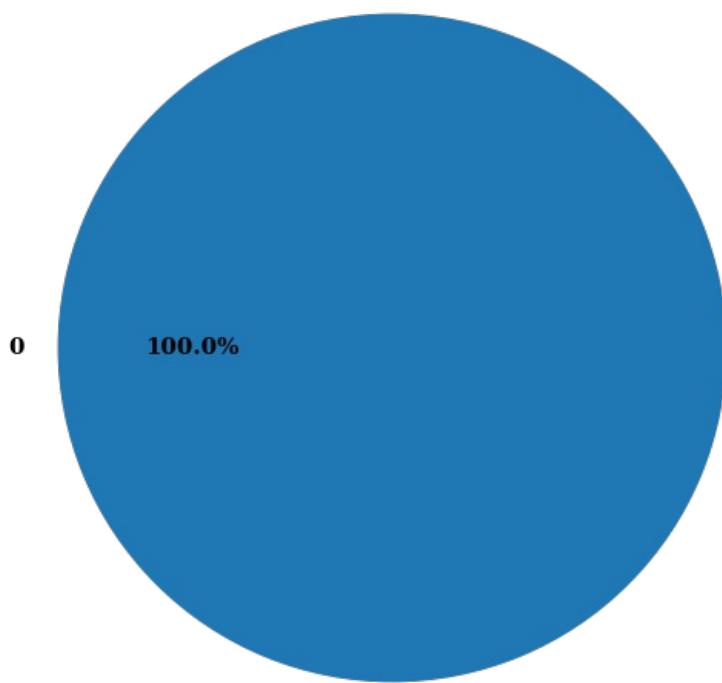
reservationCharge



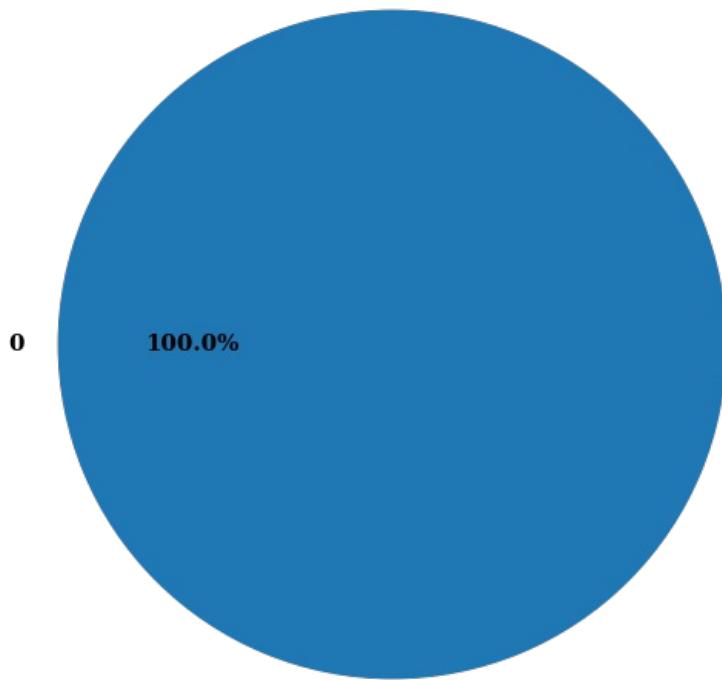
superfastCharge



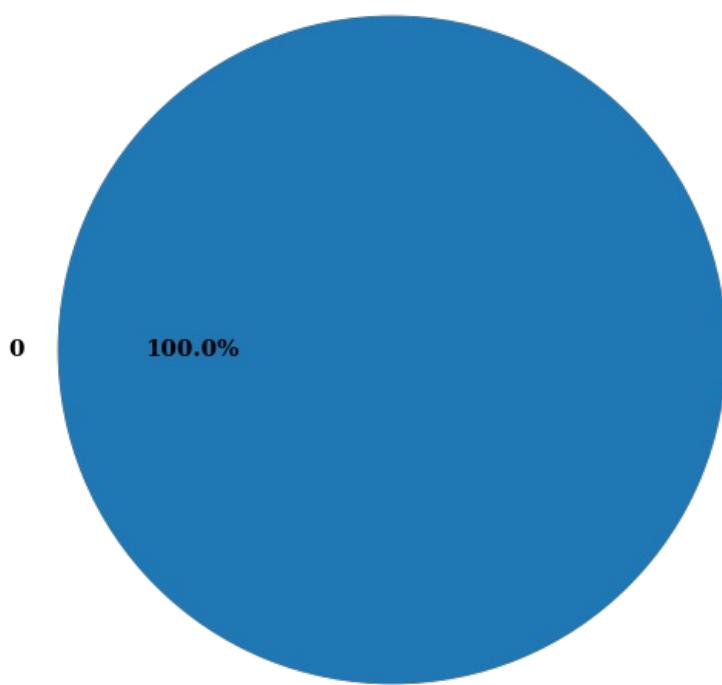
fuelAmount



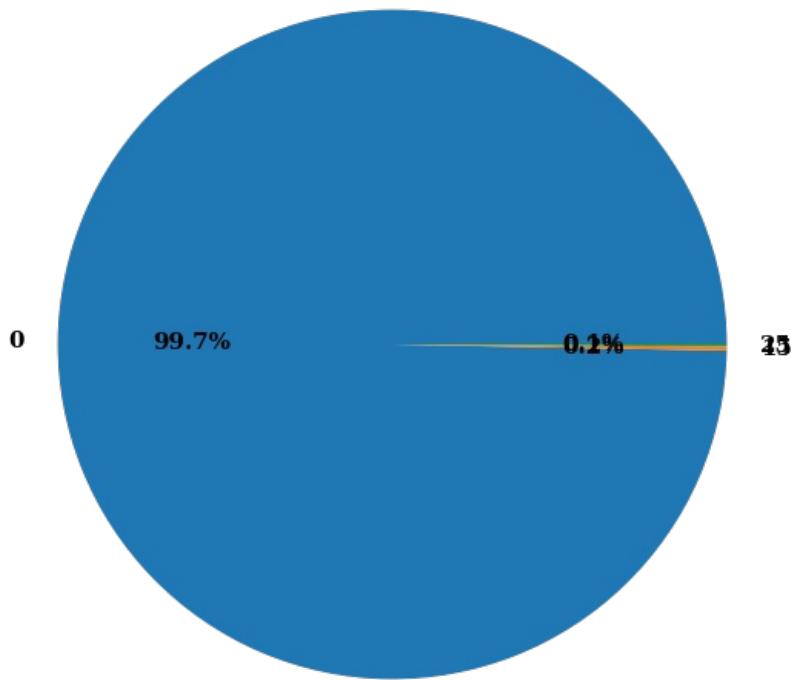
totalConcession



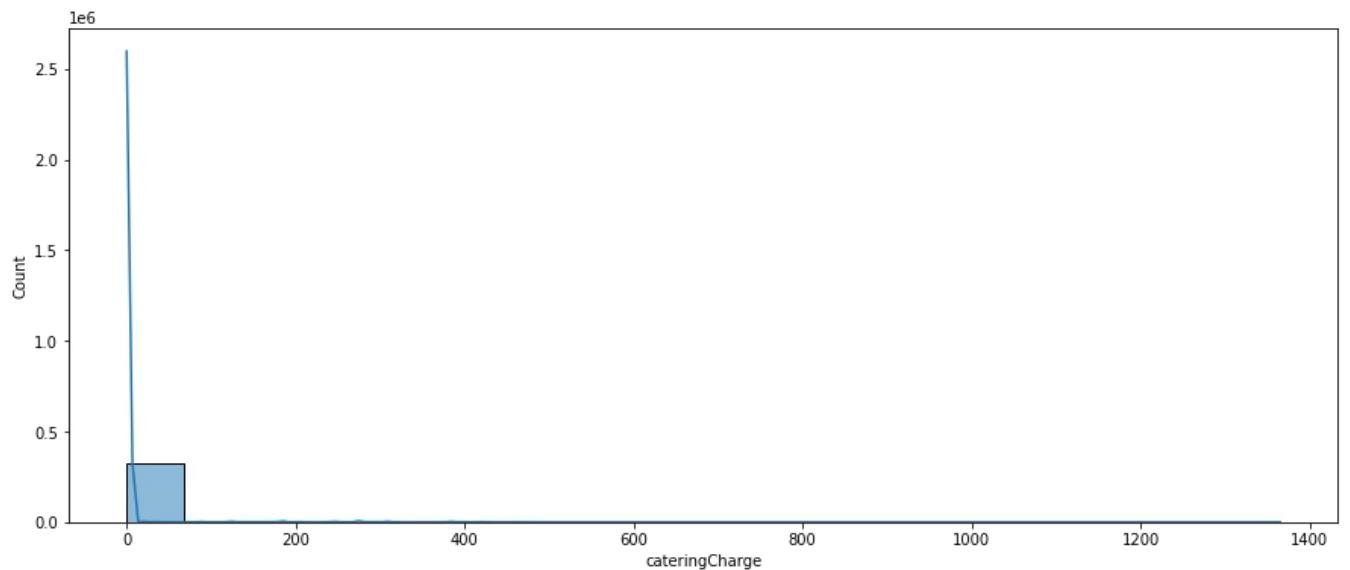
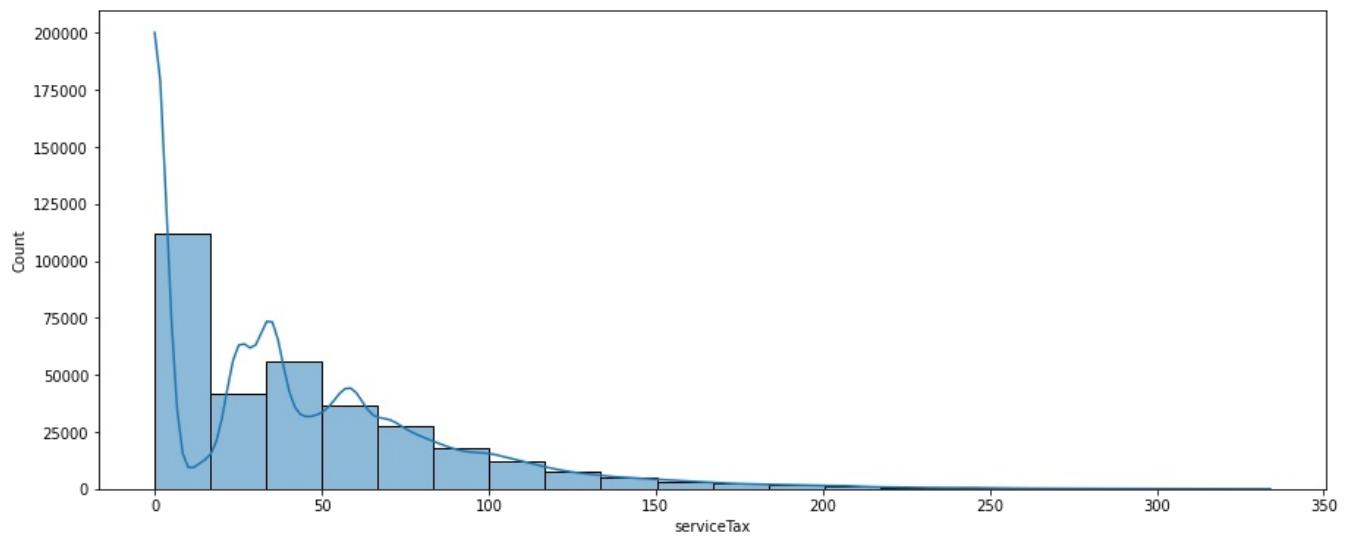
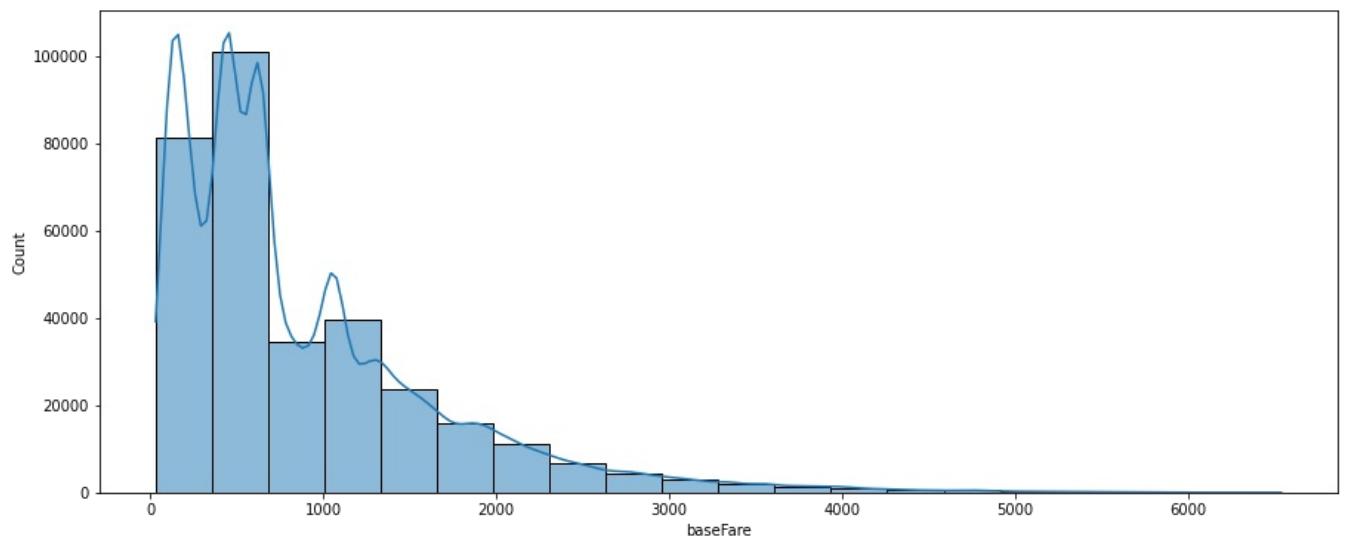
tatkalFare

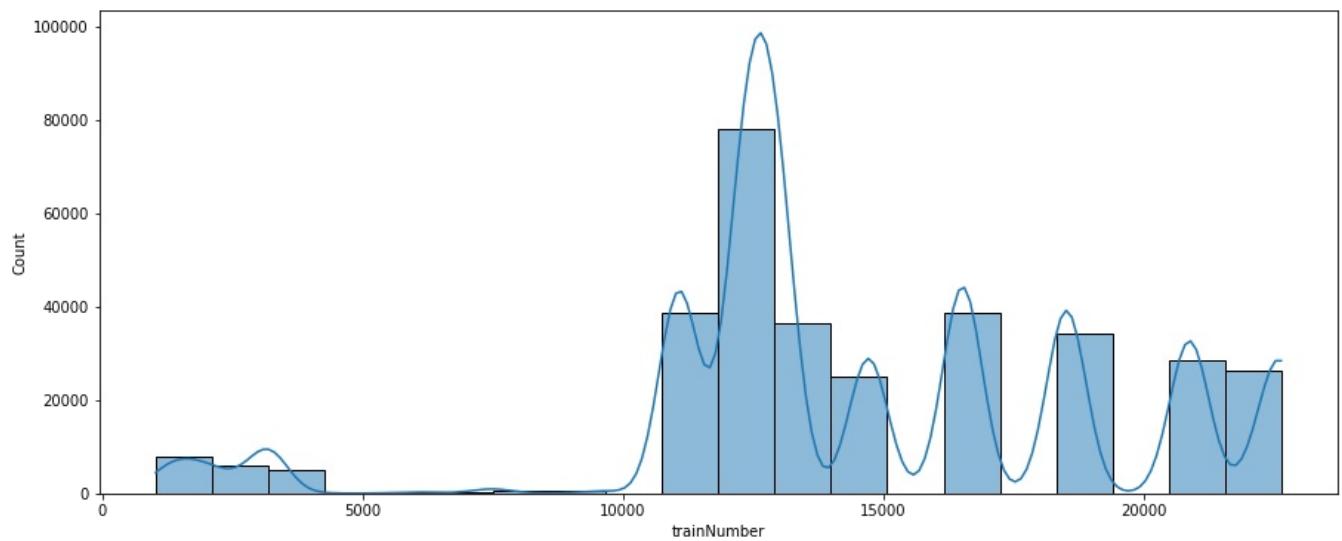
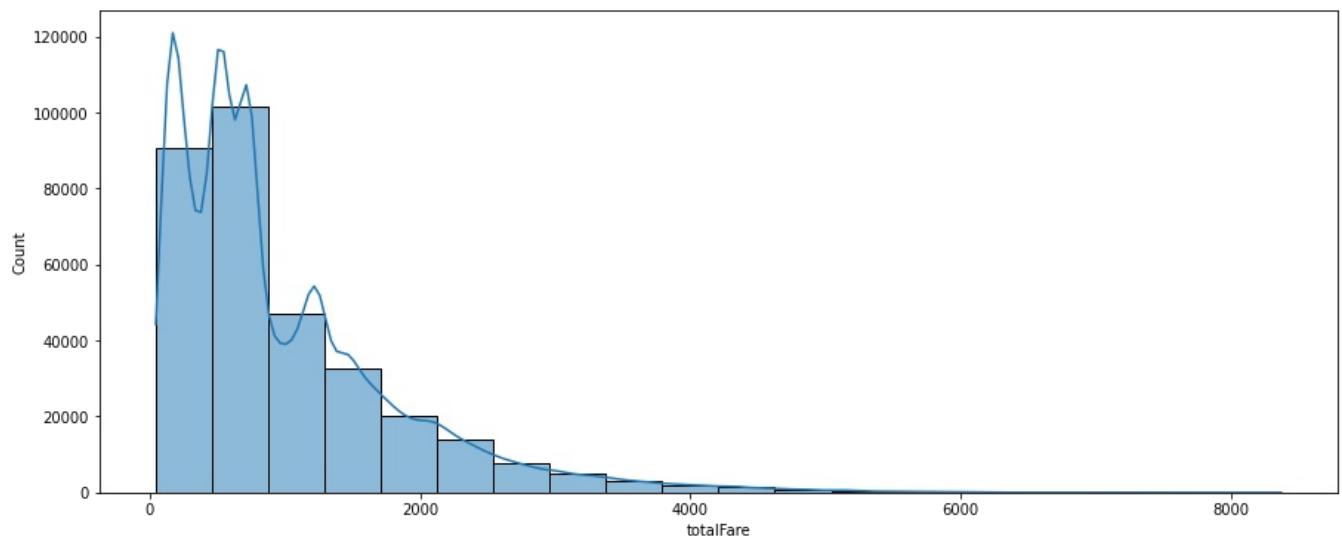
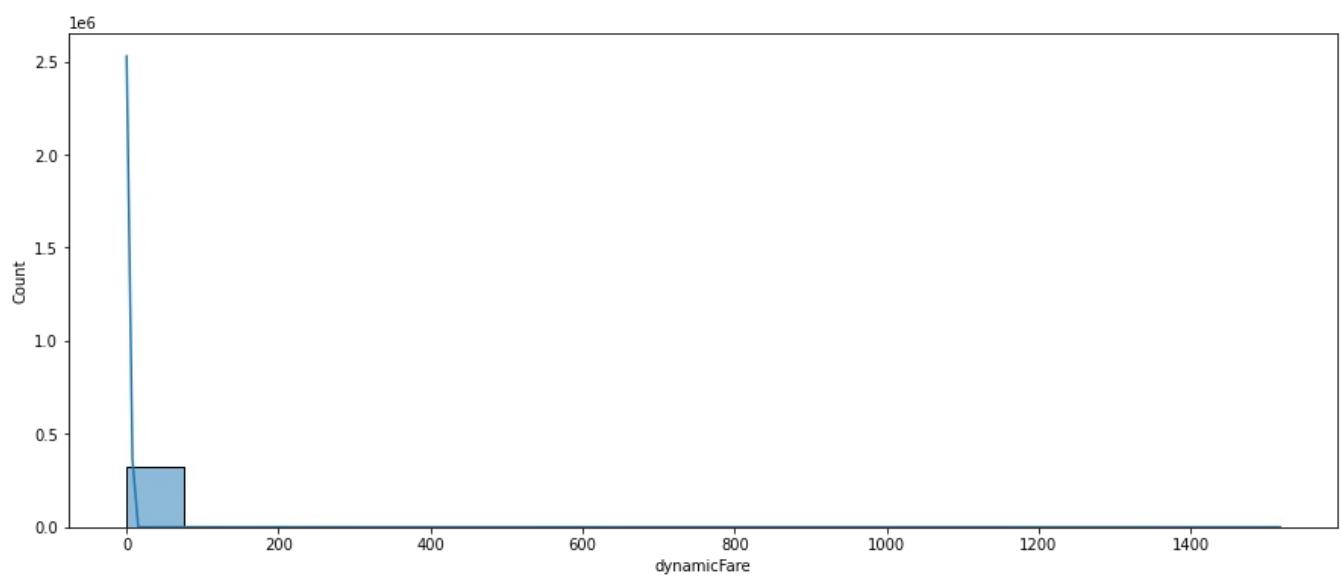


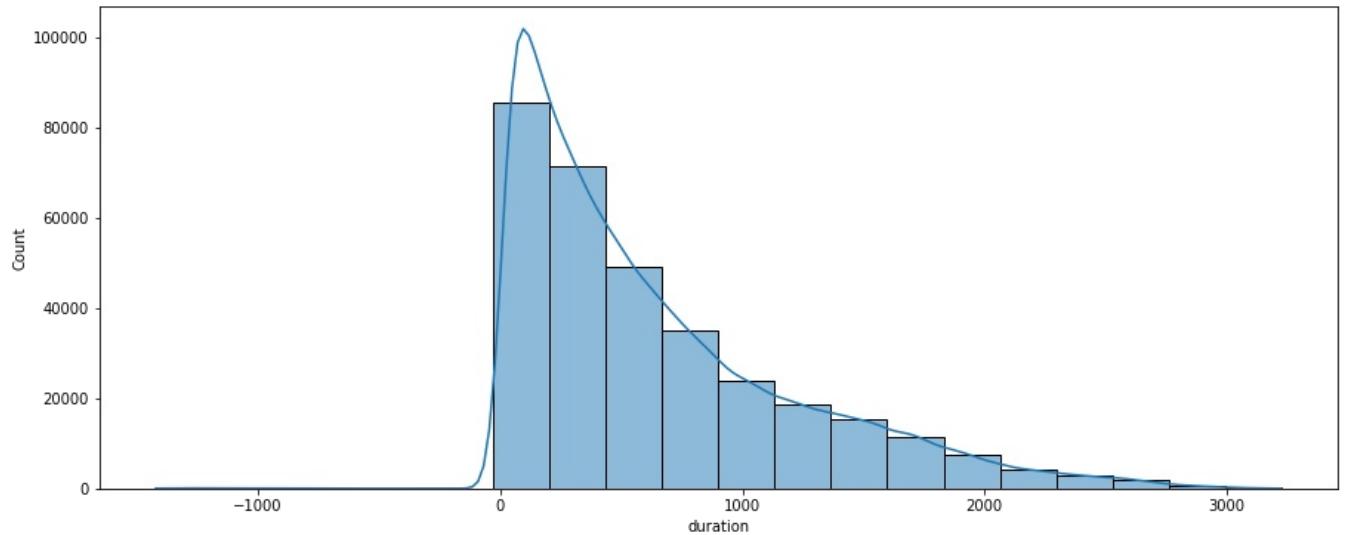
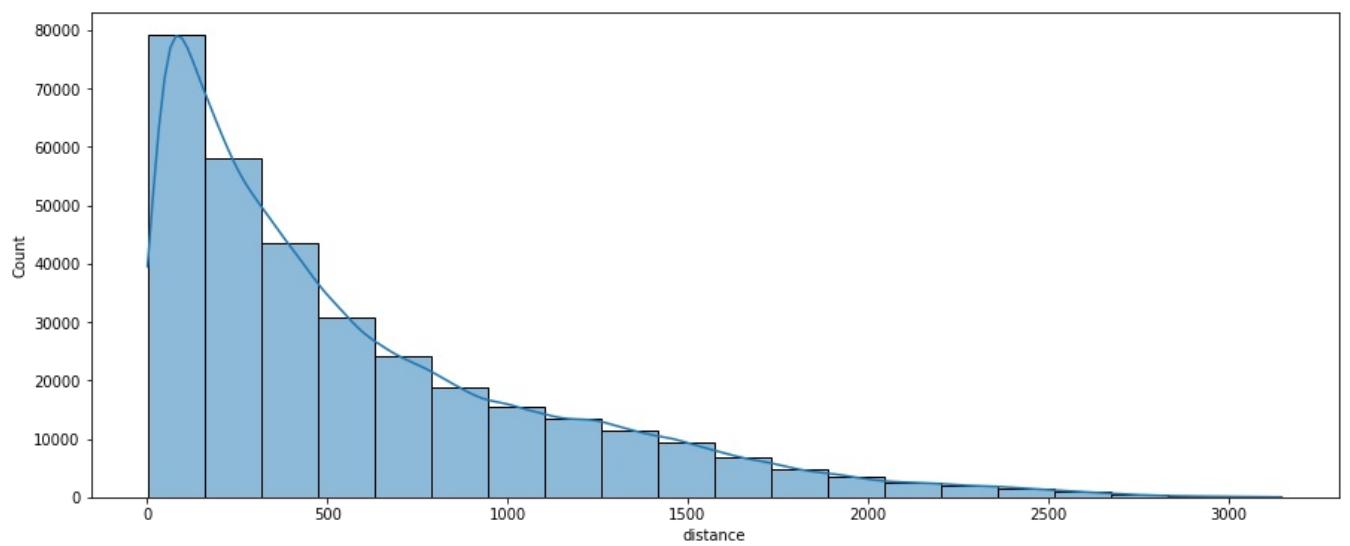
otherCharge



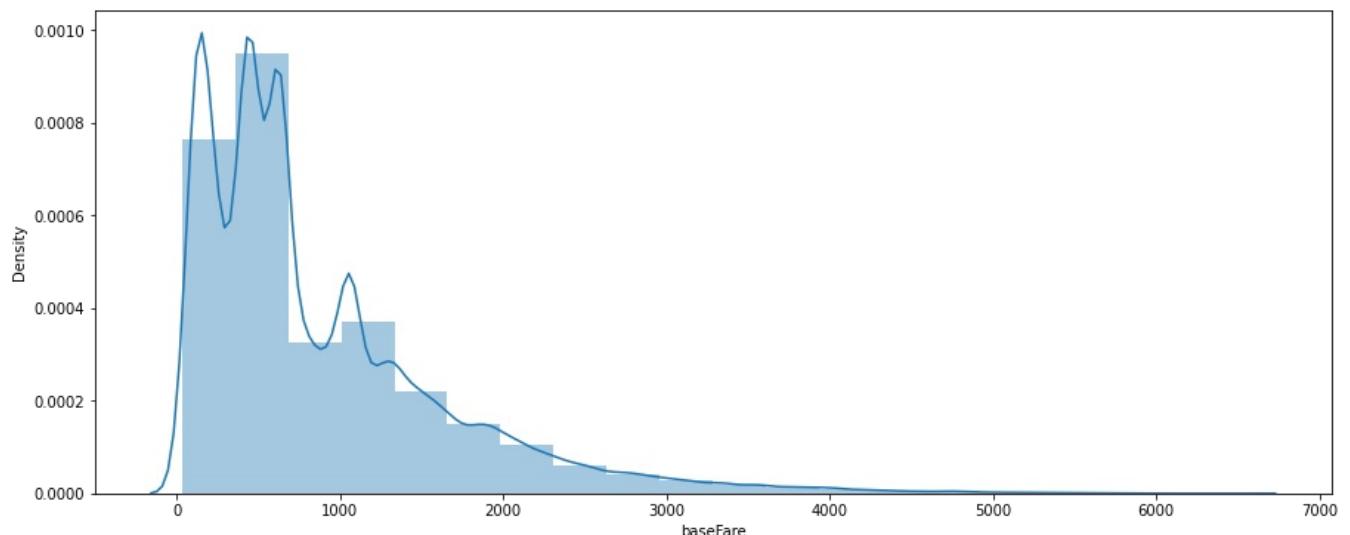
```
In [25]: for i in continuous:  
    plt.figure(figsize=(15,6))  
    sns.histplot(df[i], bins = 20, kde = True, palette='hls')  
    plt.show()
```

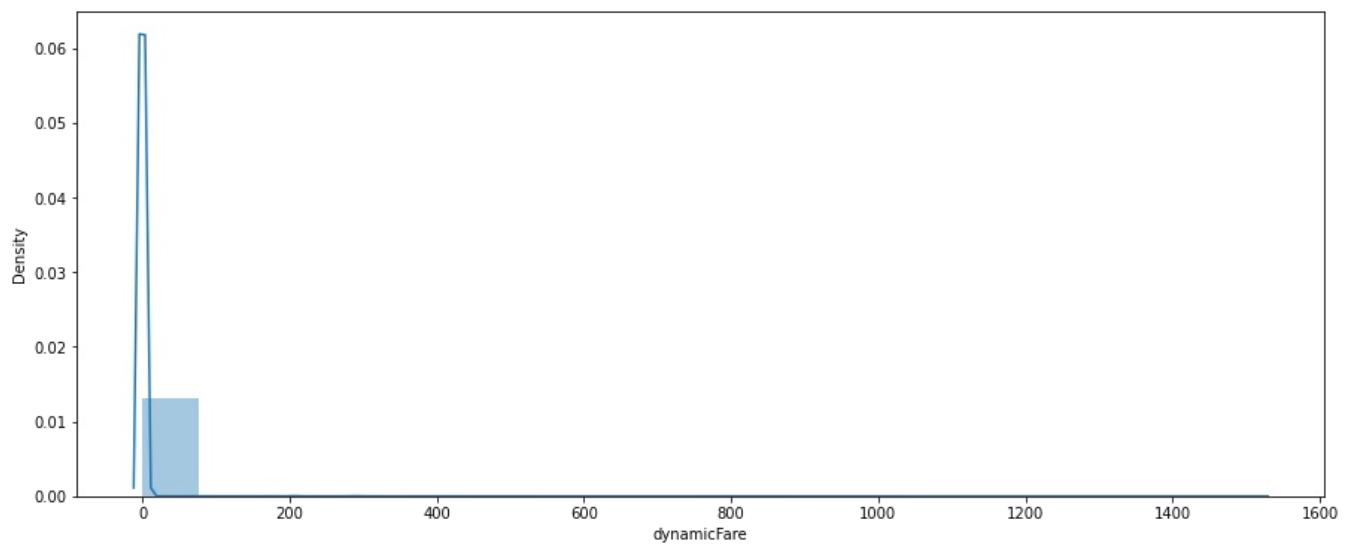
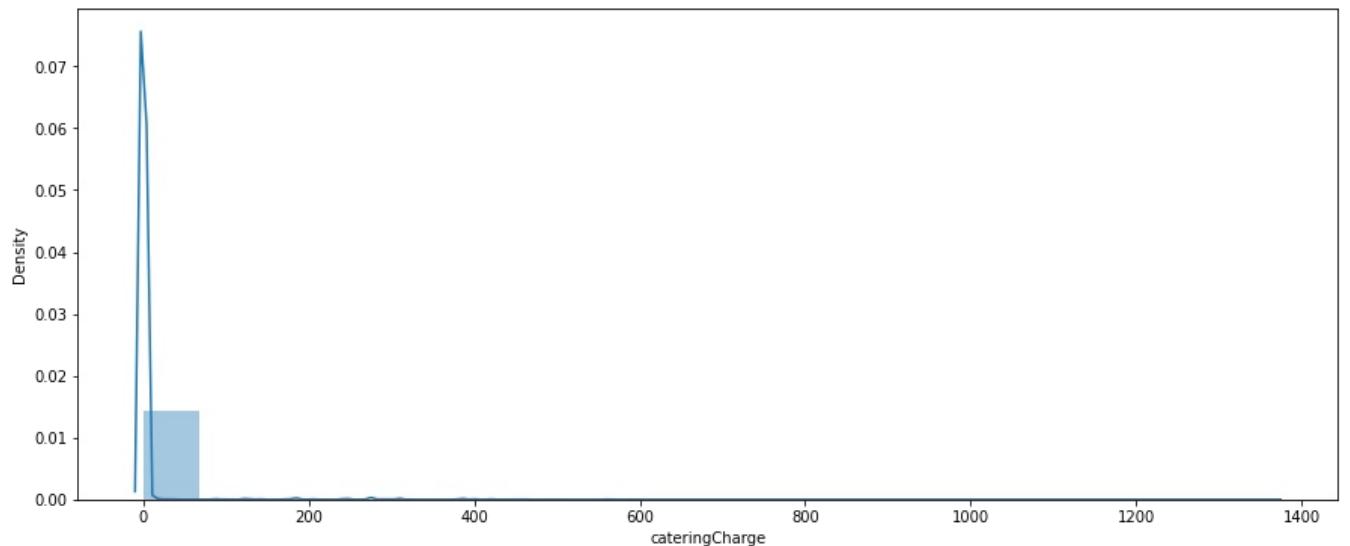
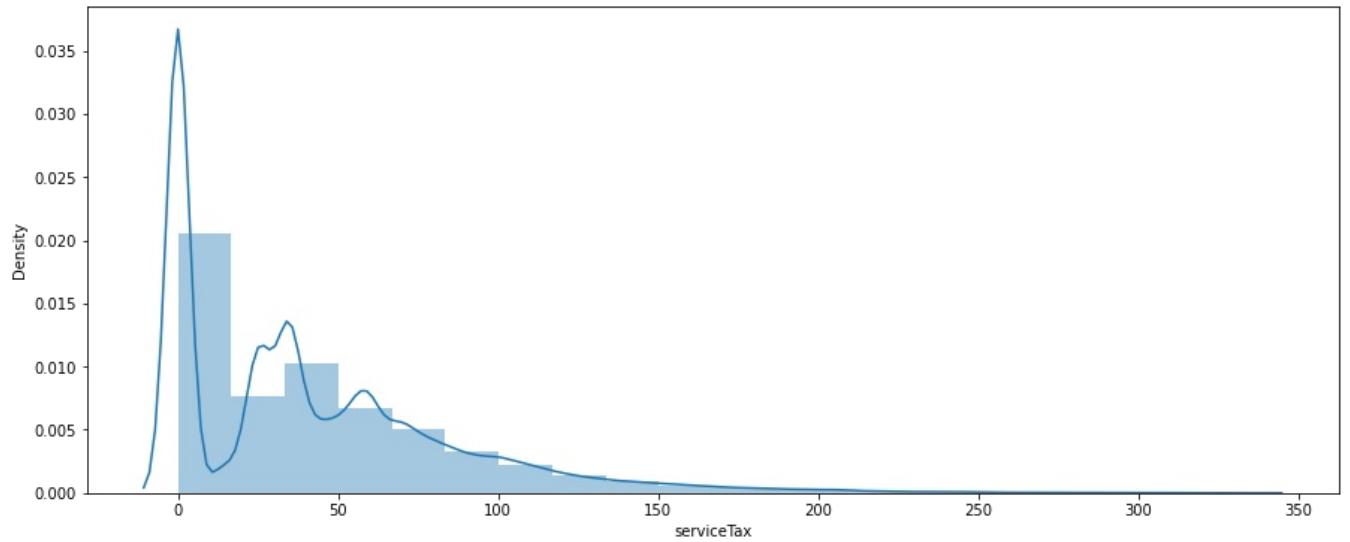


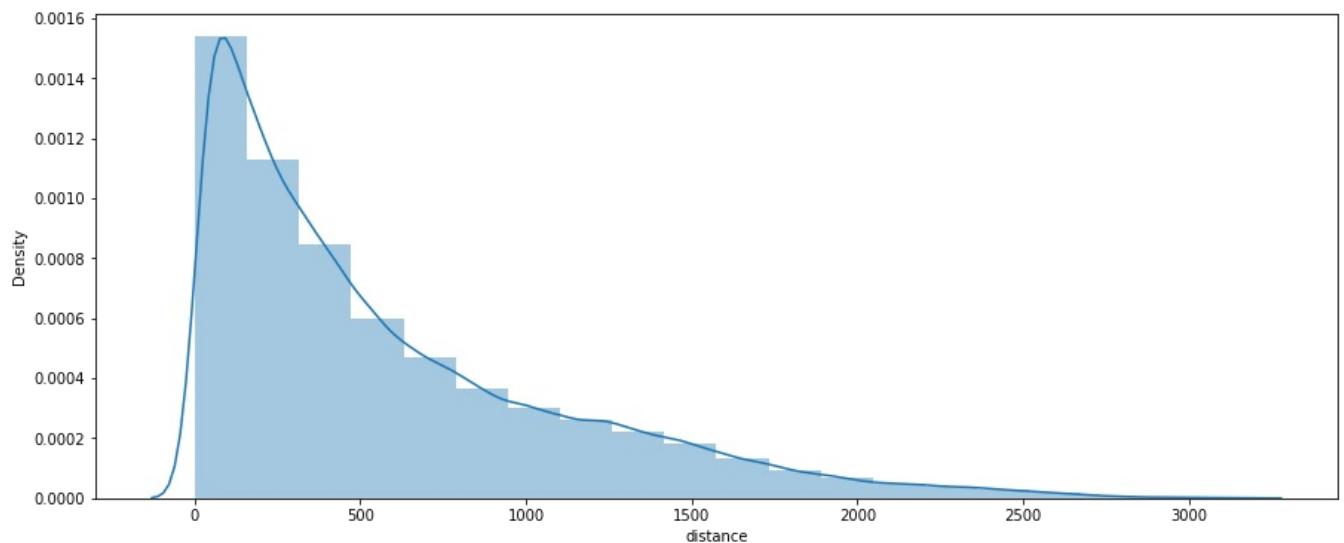
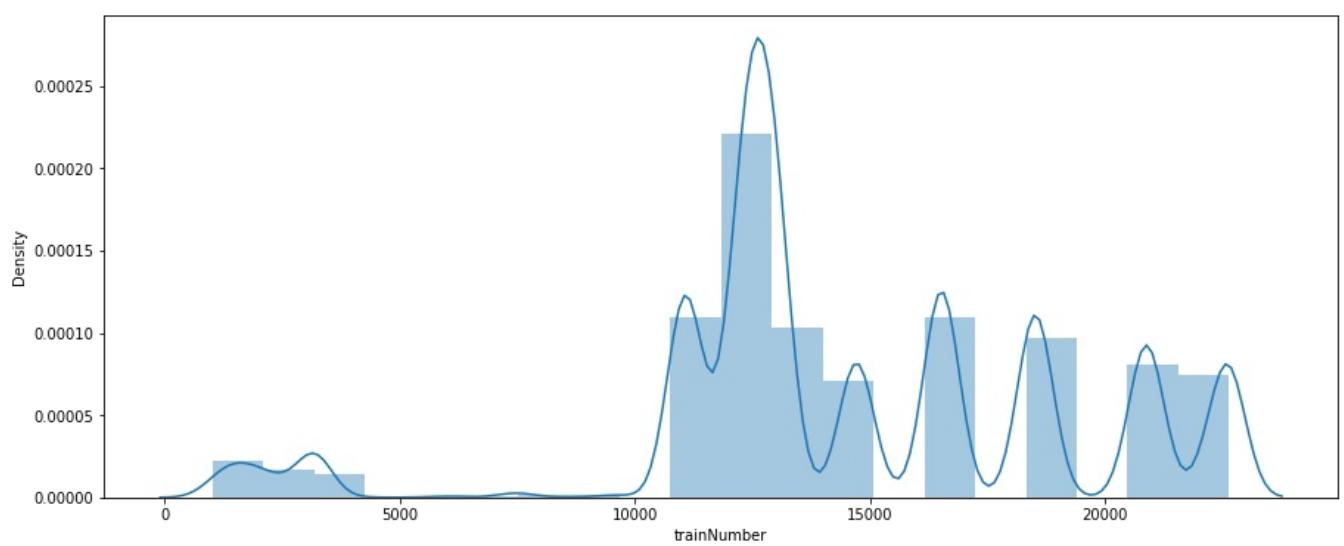
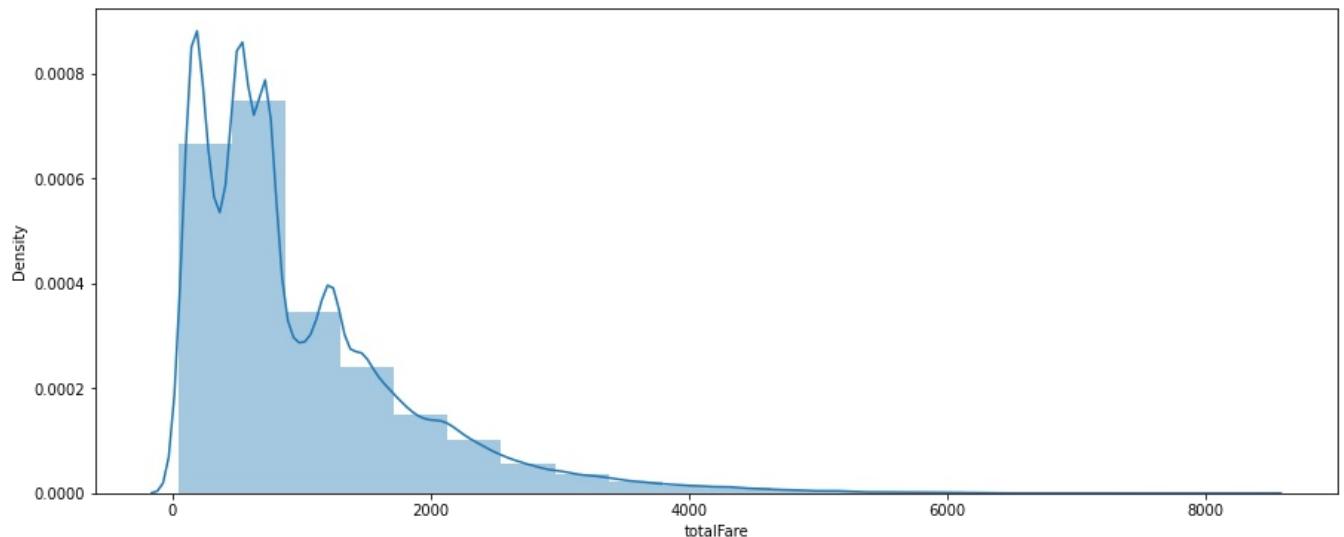


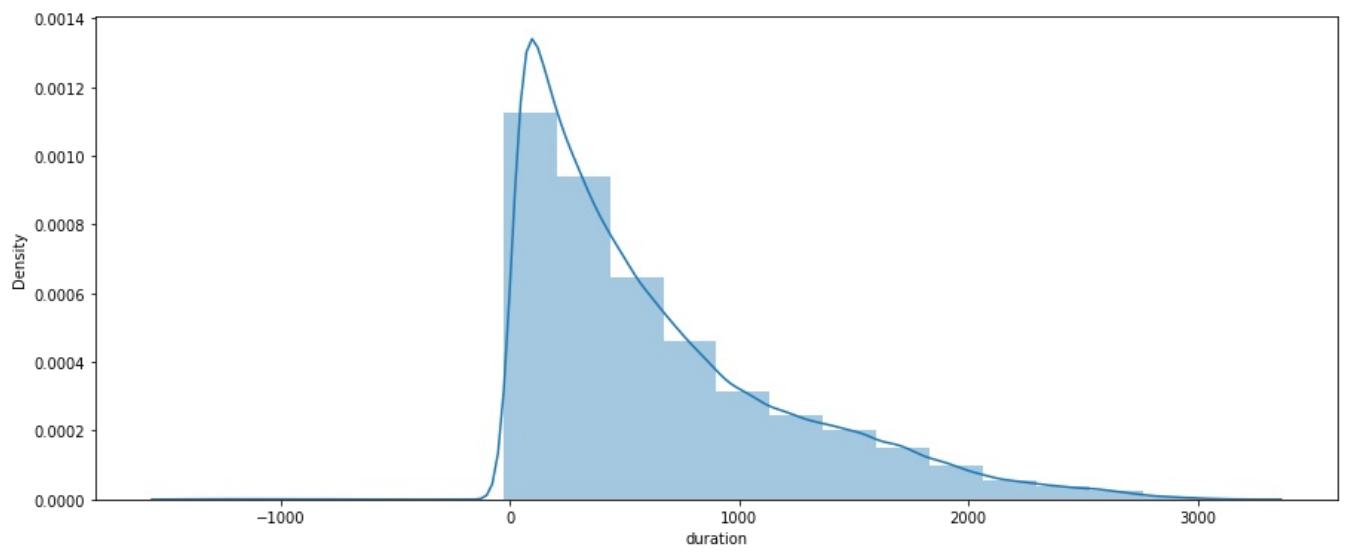


```
In [26]: for i in continuous:  
    plt.figure(figsize=(15,6))  
    sns.distplot(df[i], bins = 20, kde = True)  
    plt.show()
```

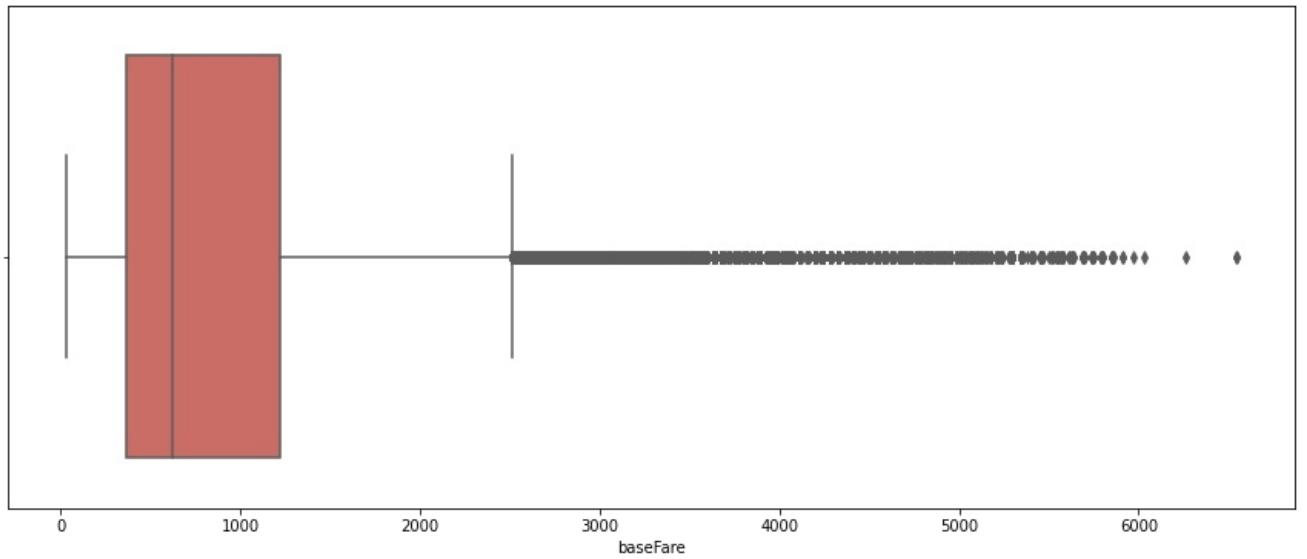


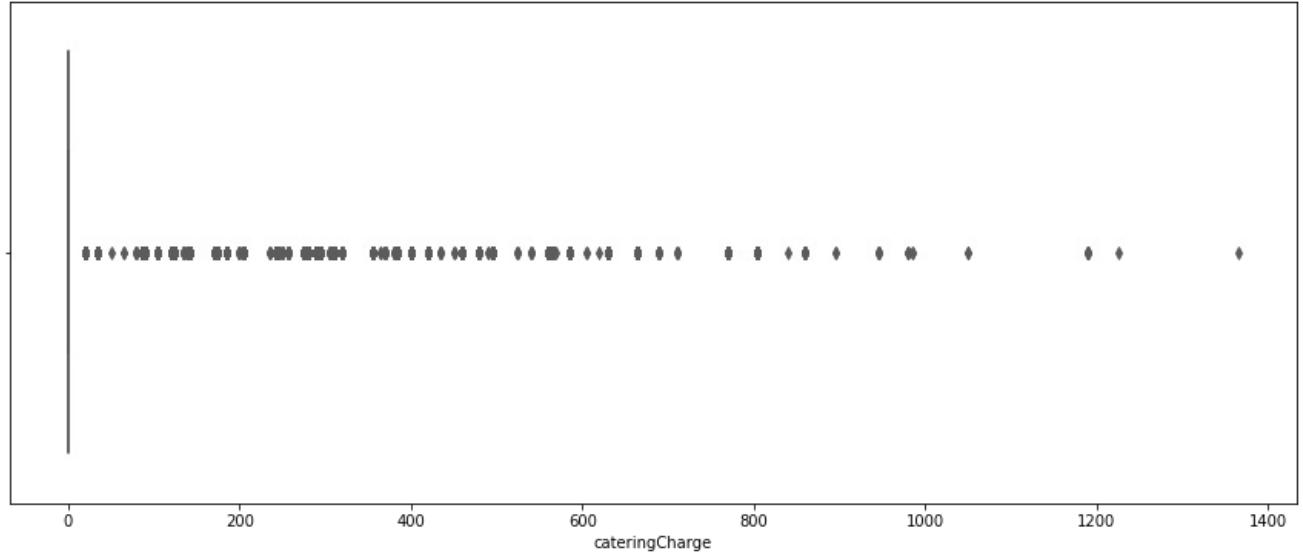
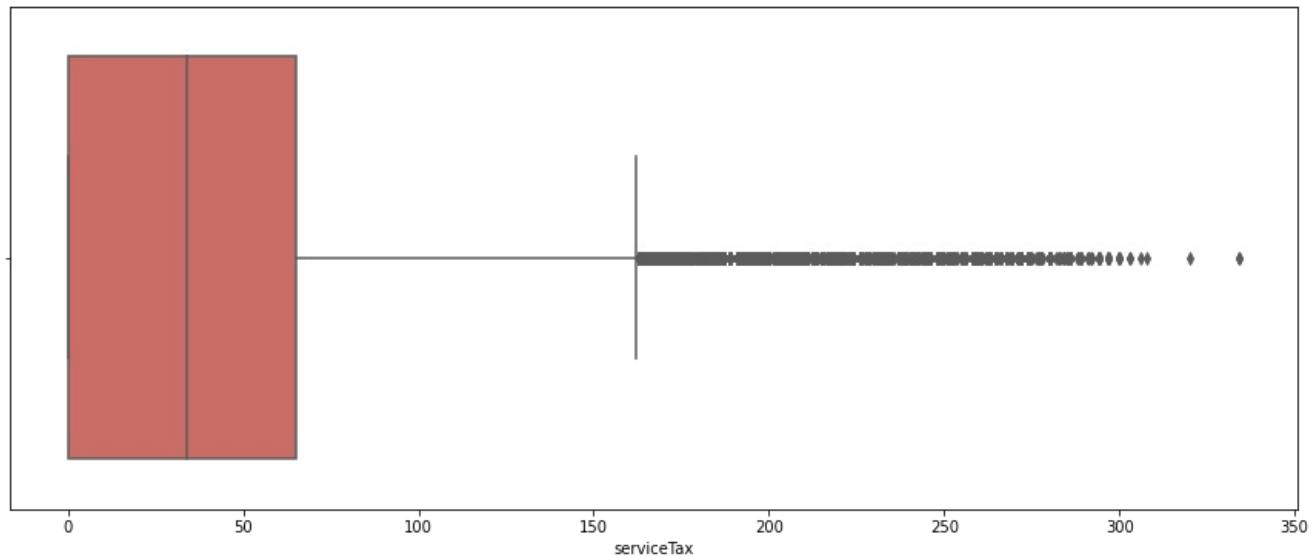


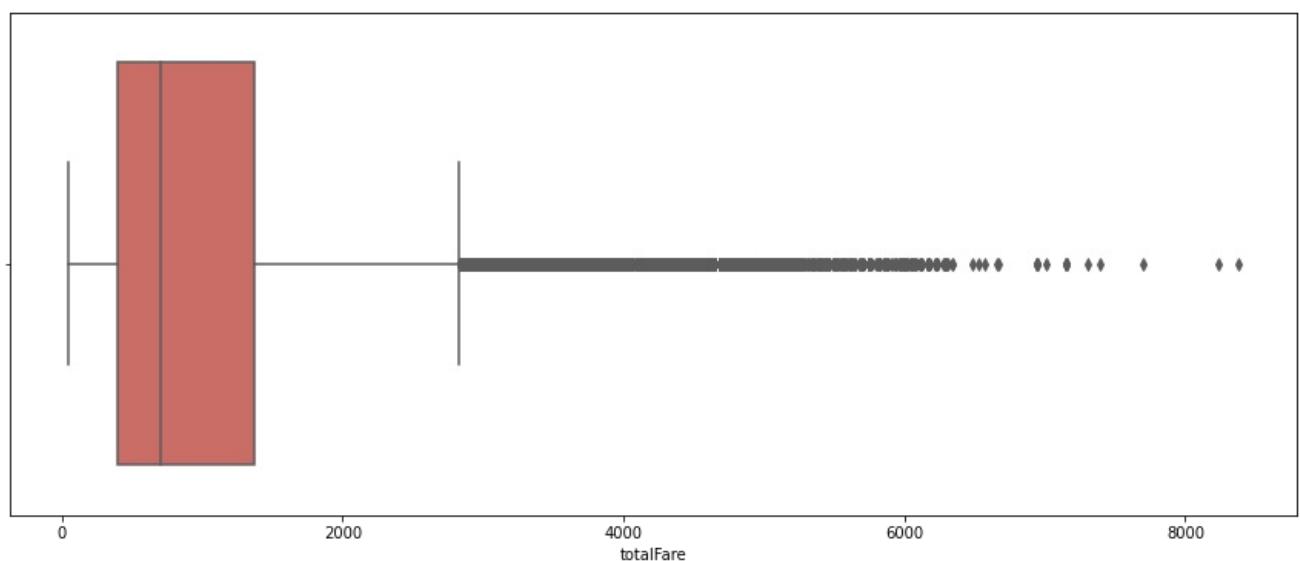
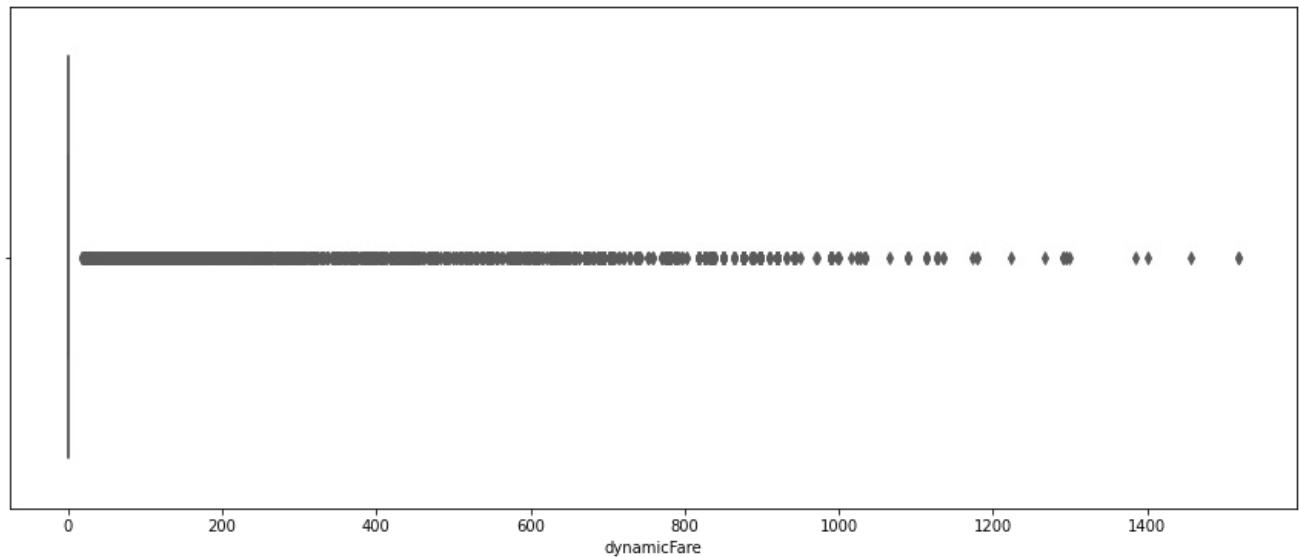


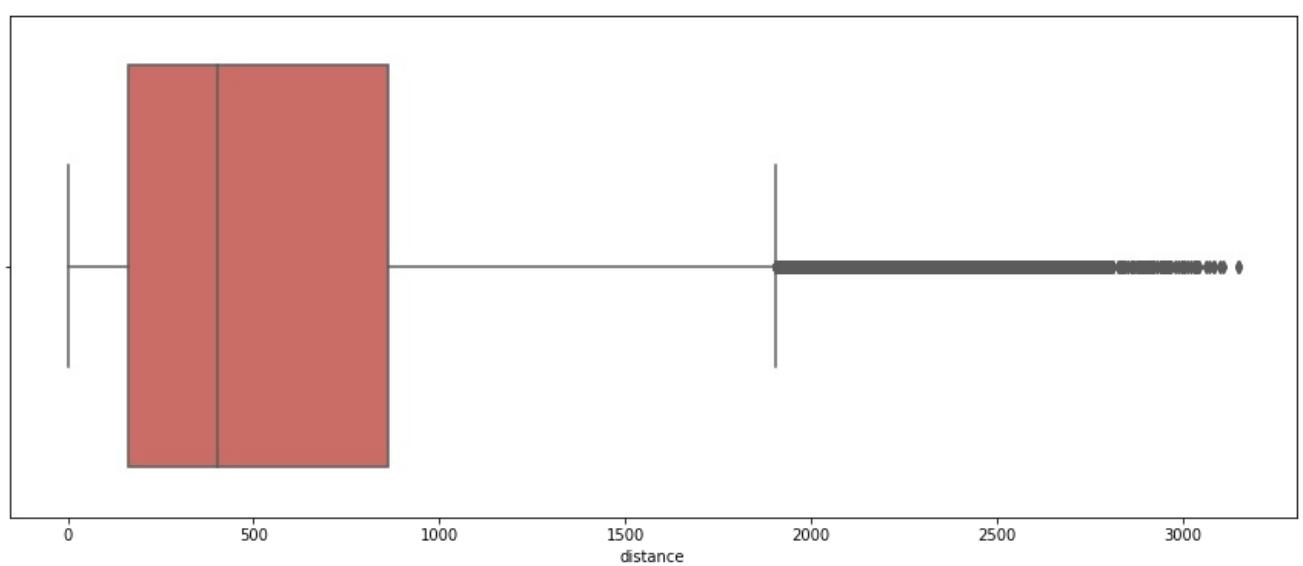
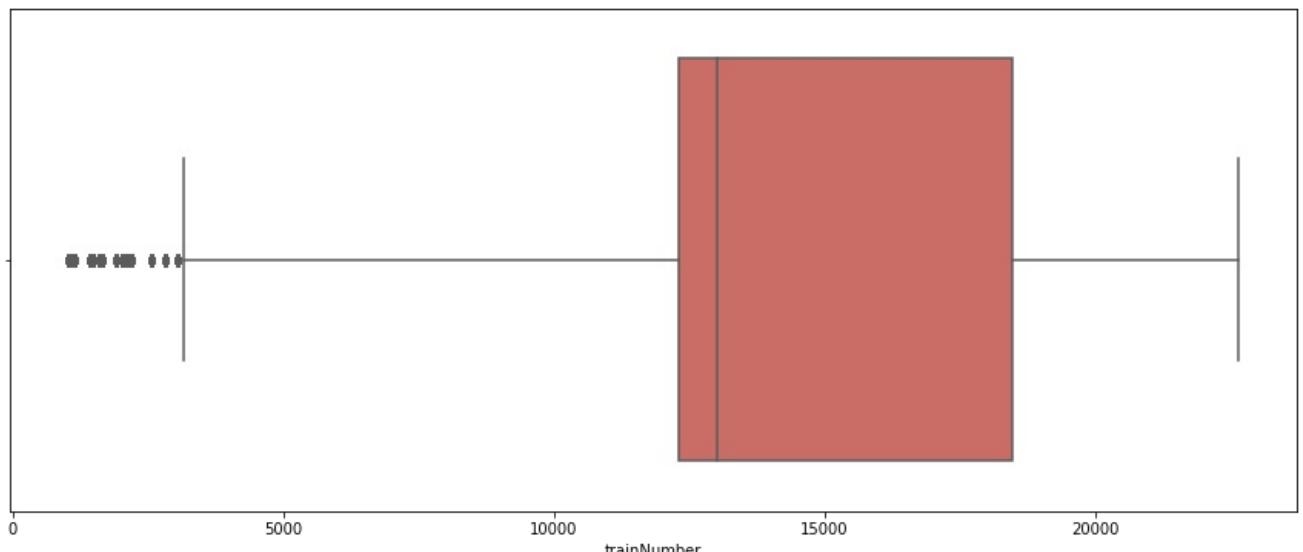


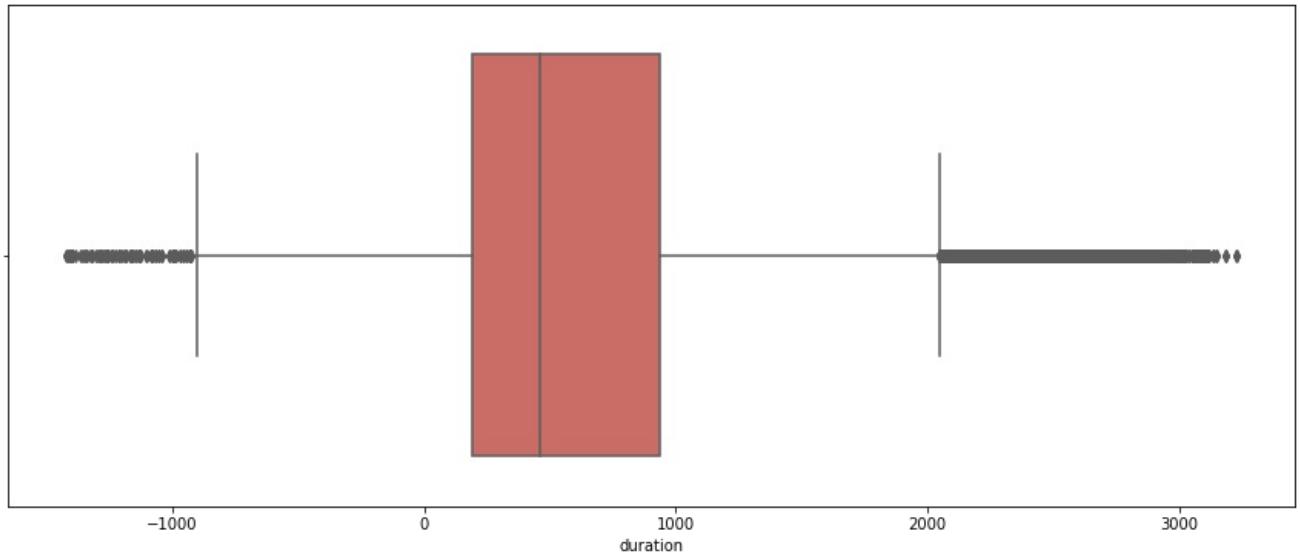
```
In [27]: for i in continuous:  
    plt.figure(figsize=(15,6))  
    sns.boxplot(i, data = df, palette='hls')  
    plt.show()
```



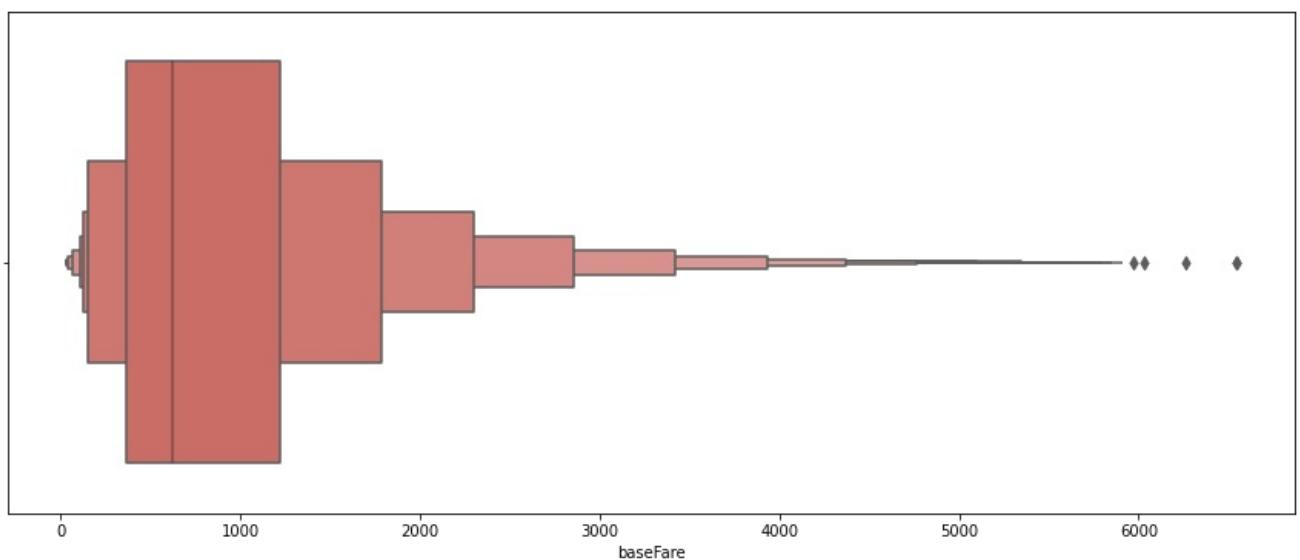


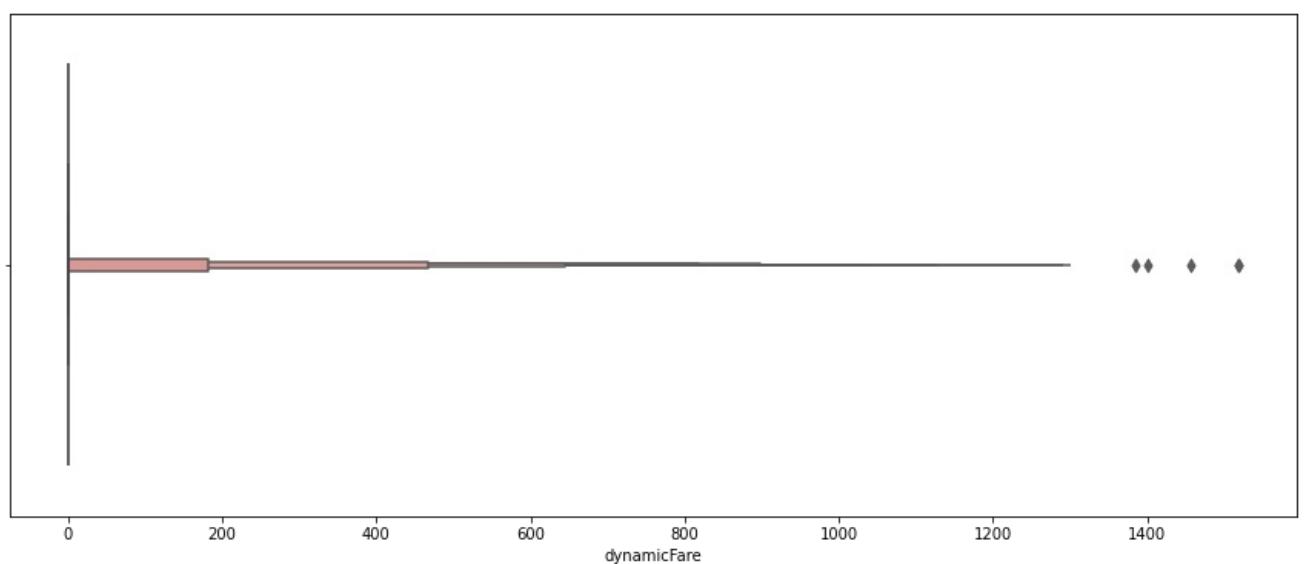
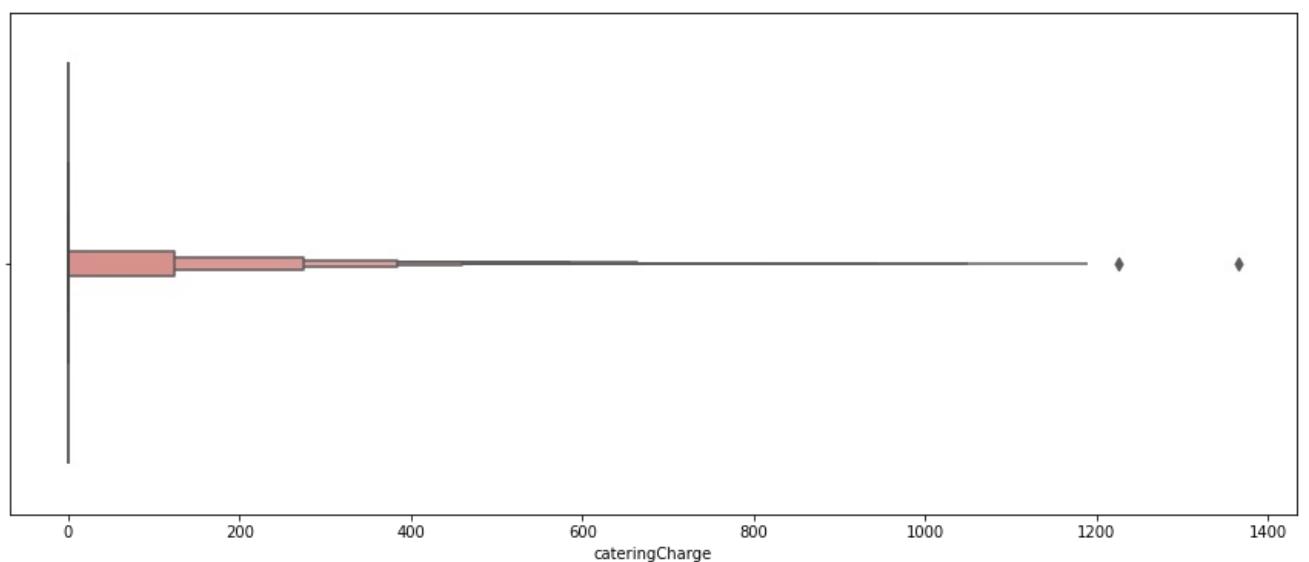
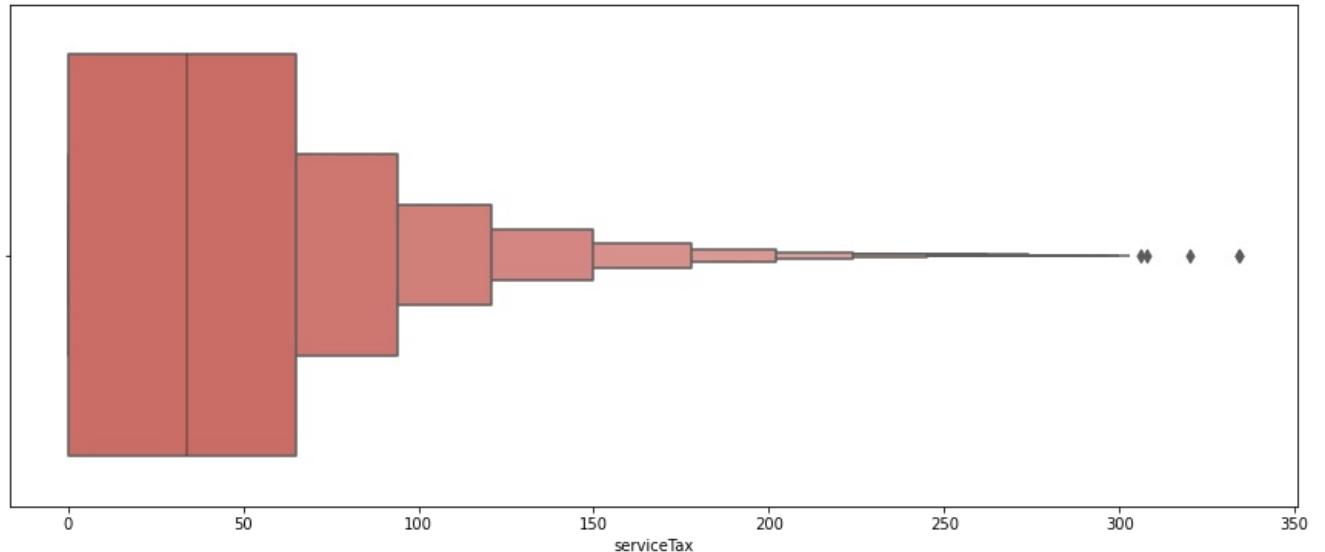


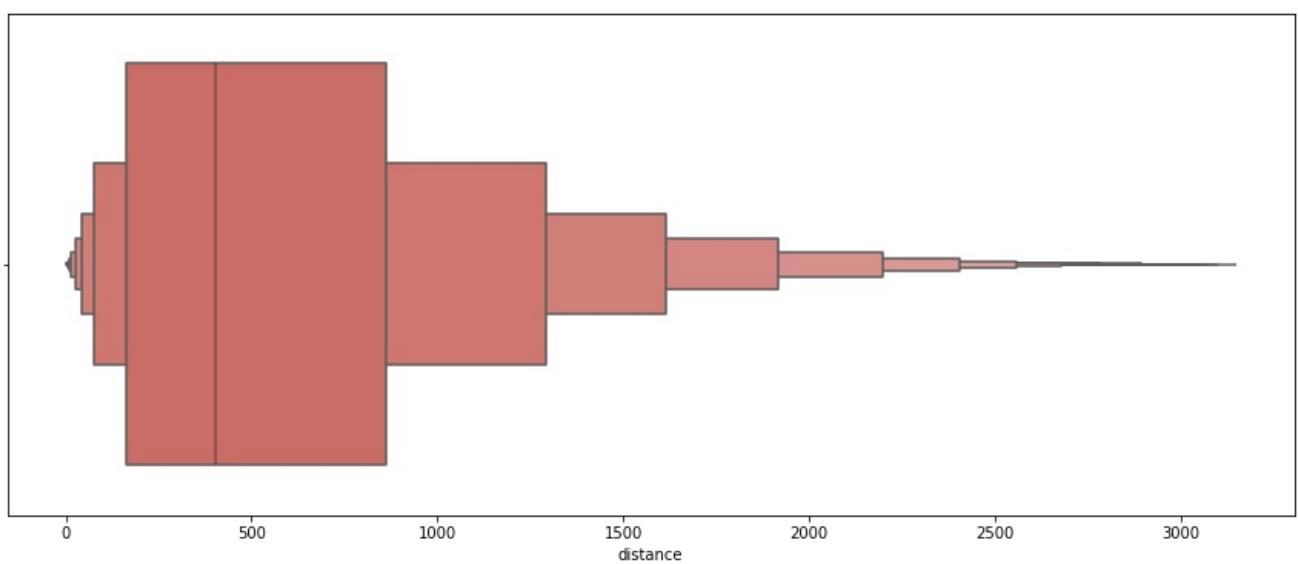
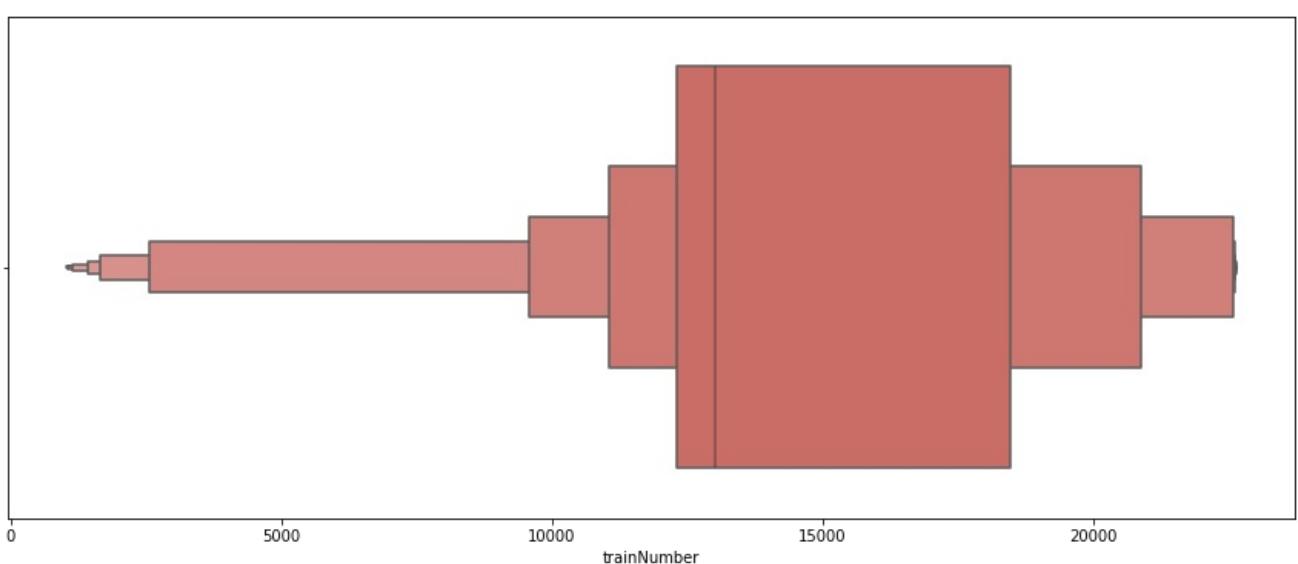
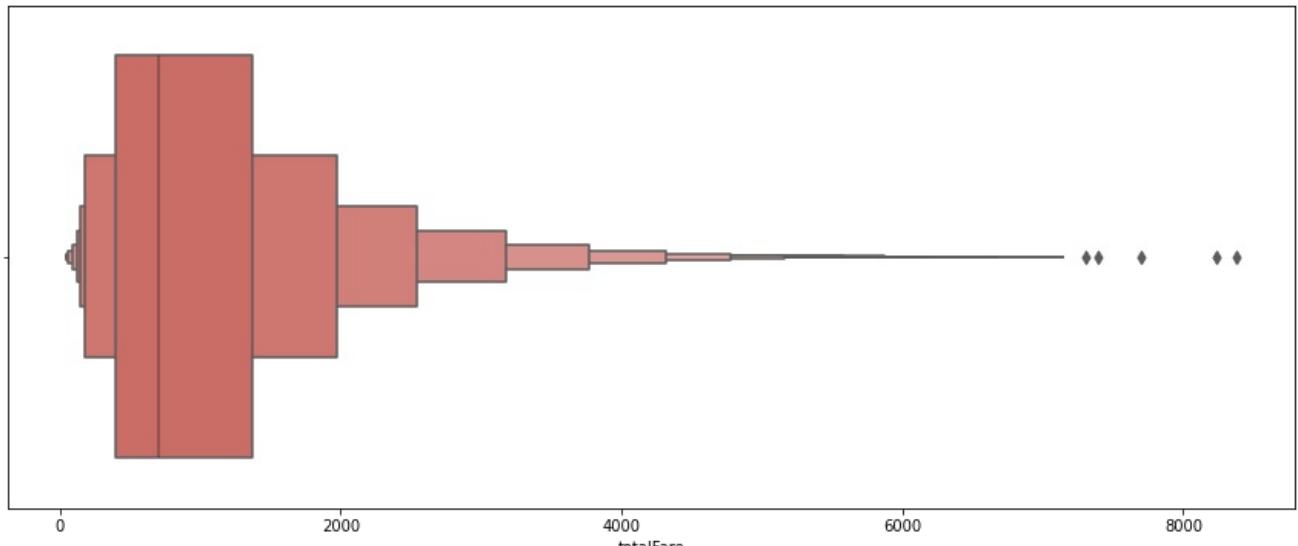


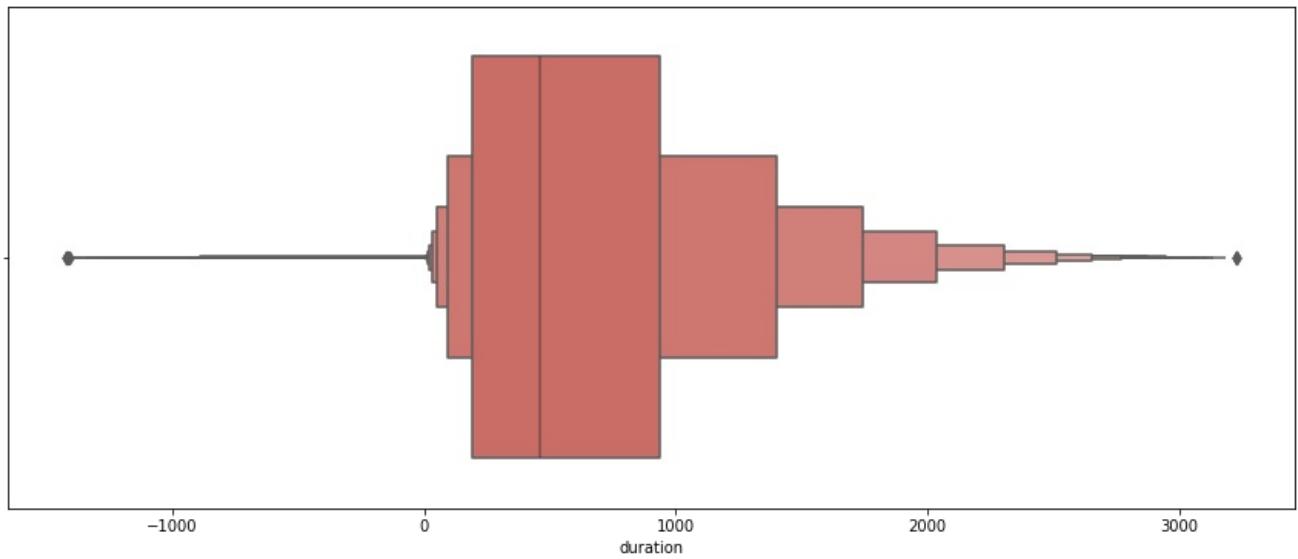


```
In [28]: for i in continuous:  
    plt.figure(figsize=(15,6))  
    sns.boxenplot(i, data = df, palette='hls')  
    plt.show()
```

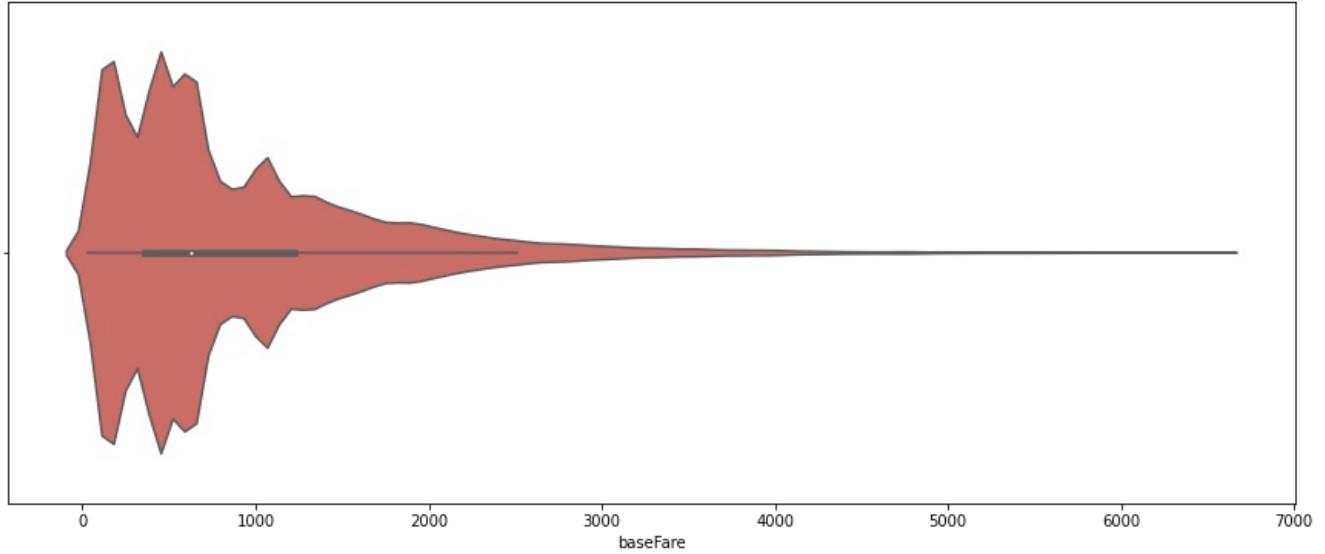


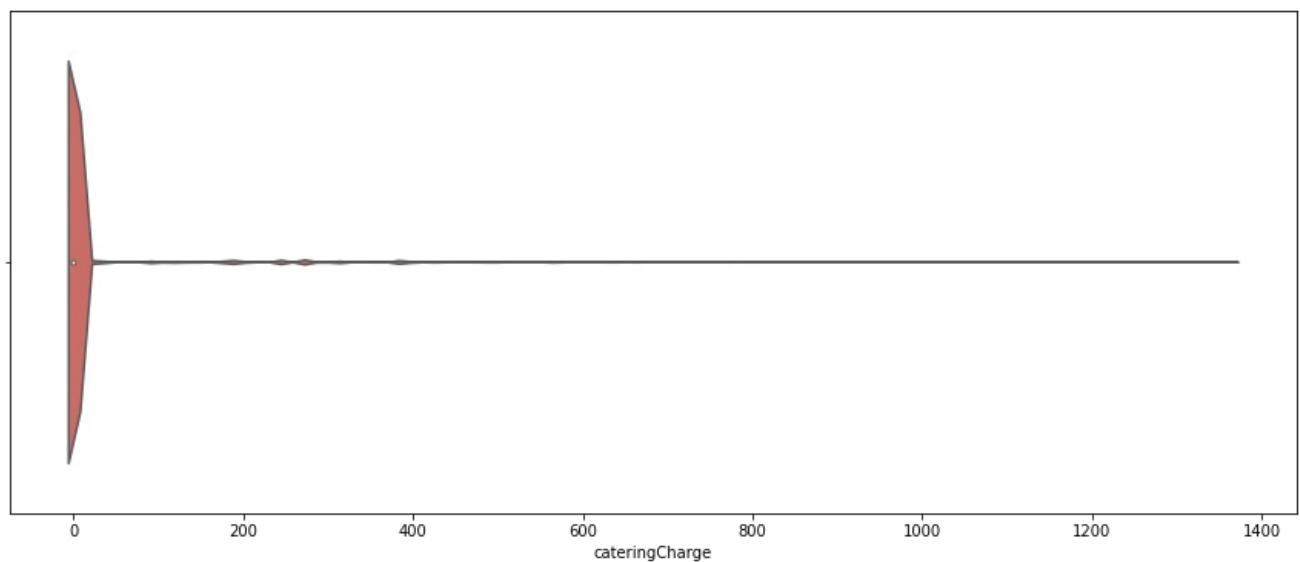
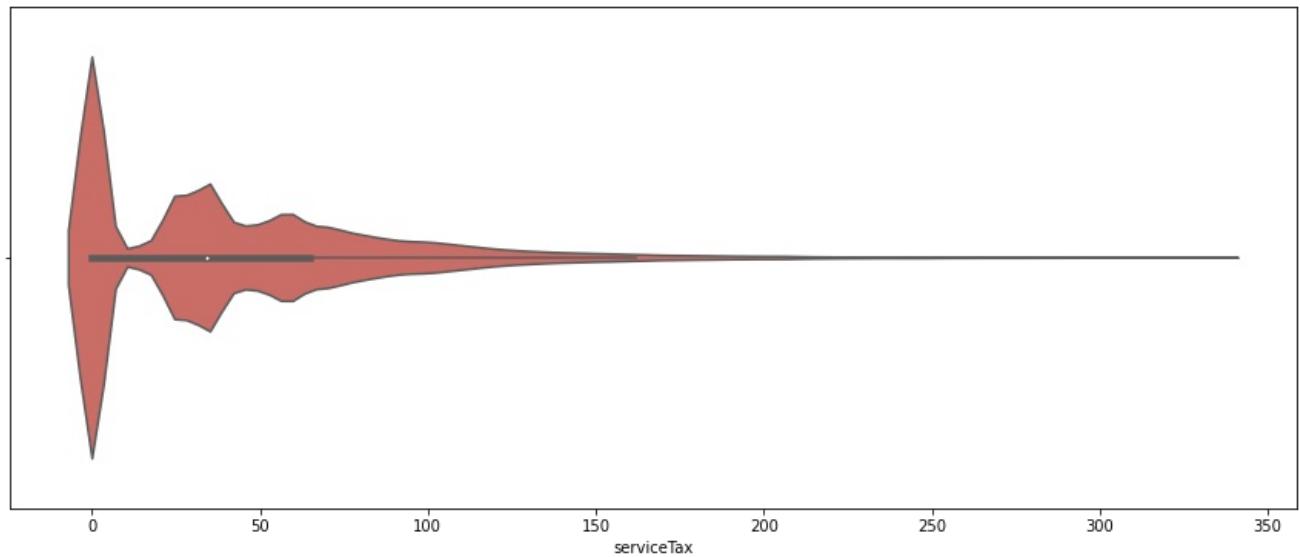


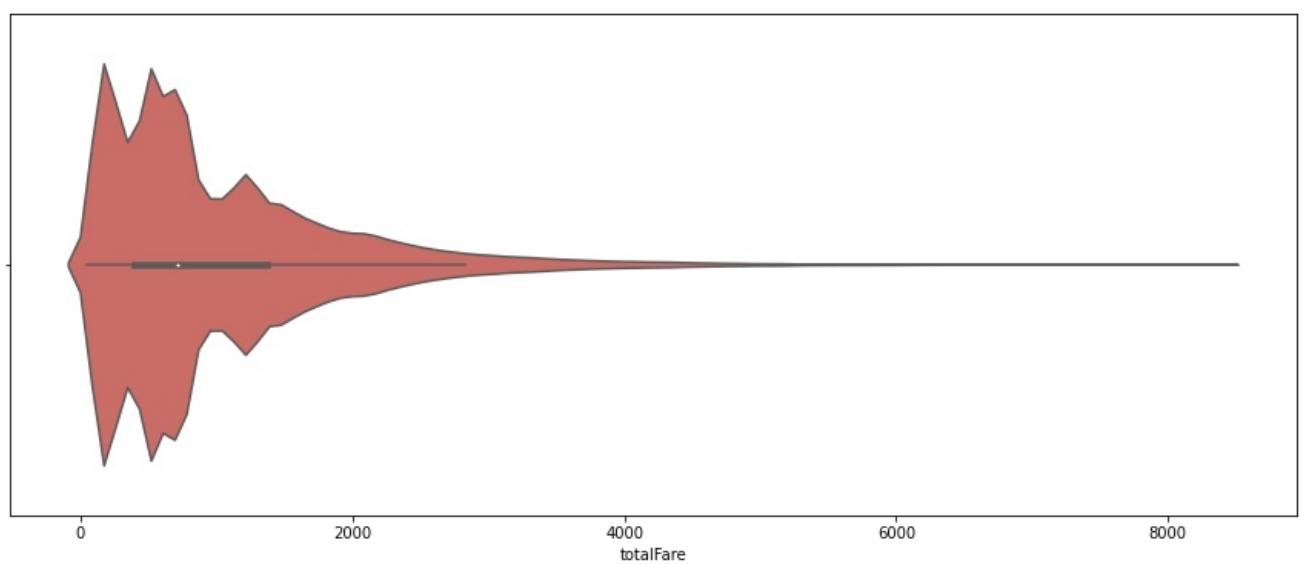
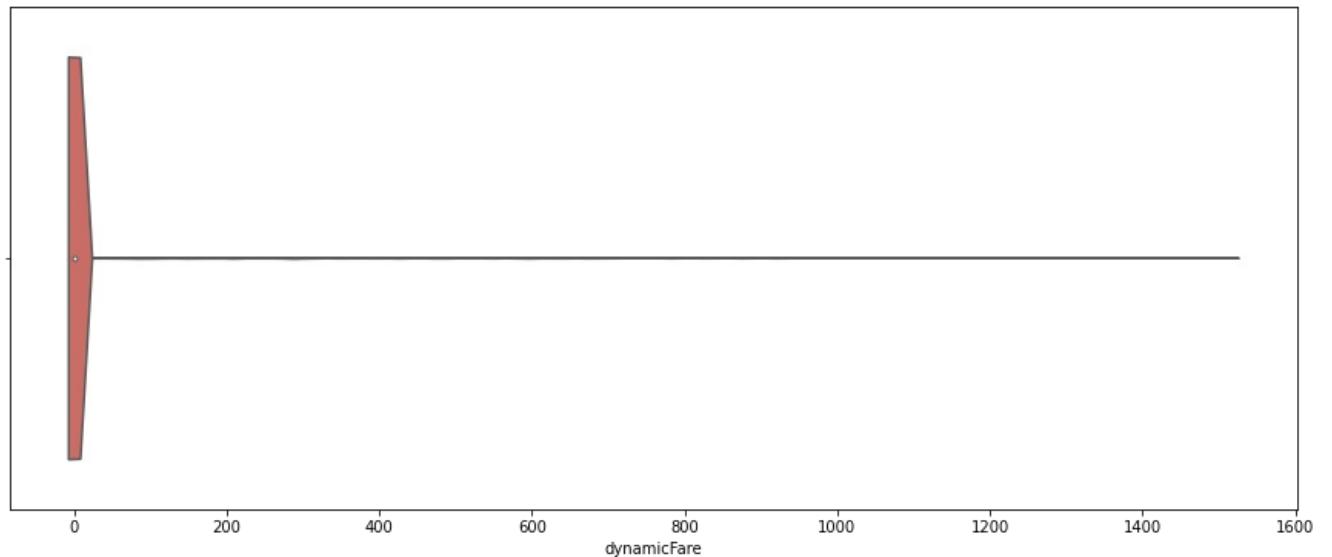


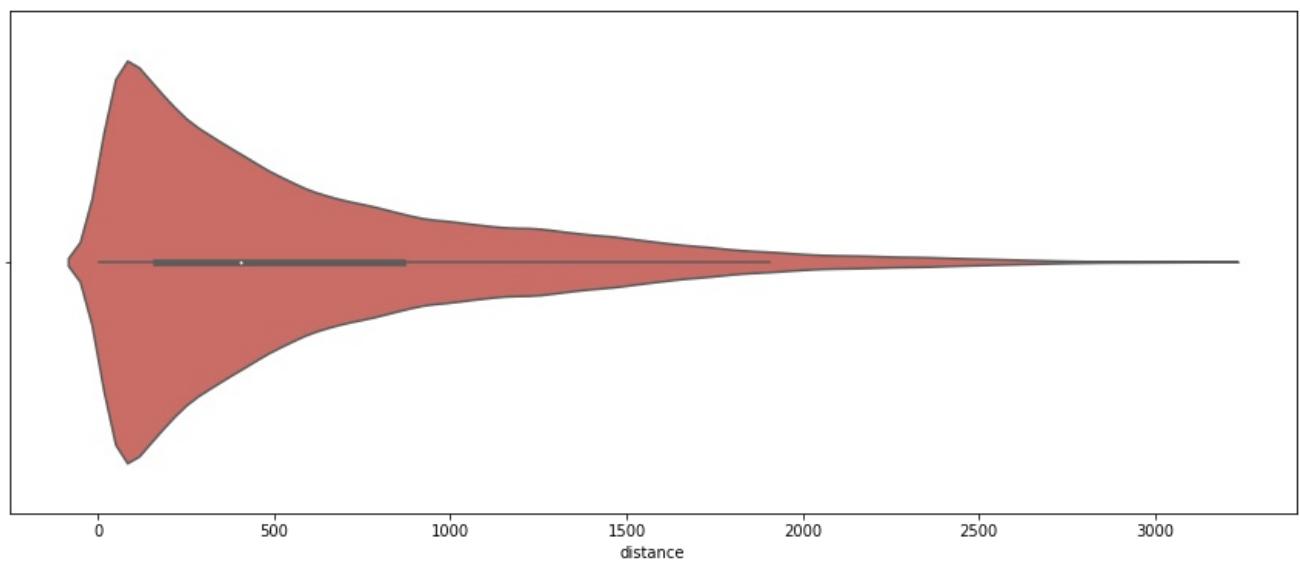
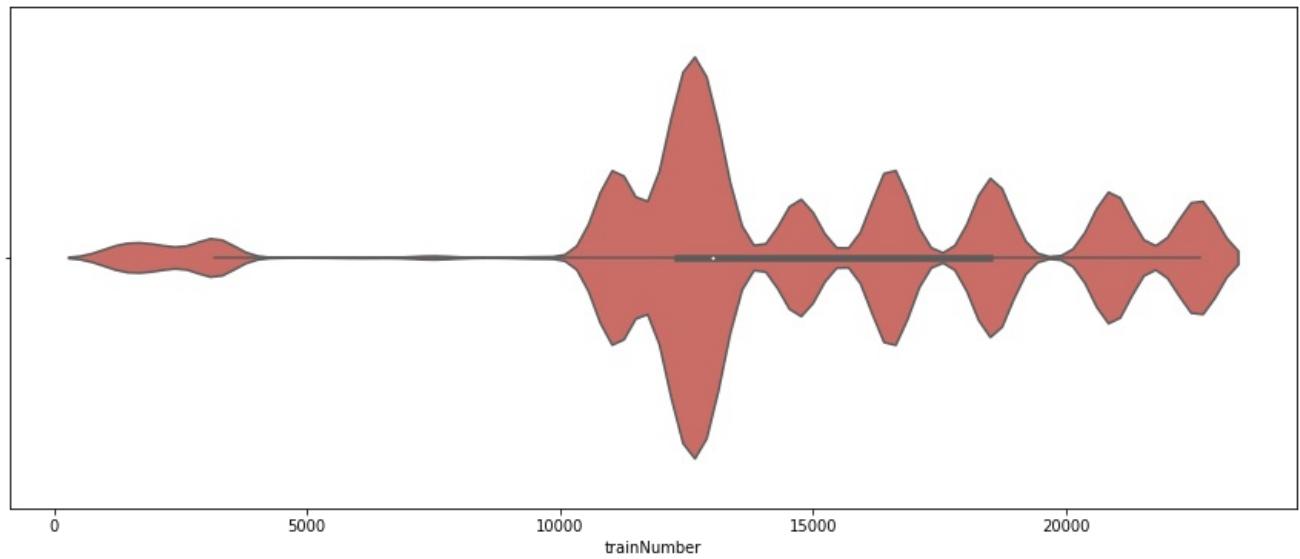


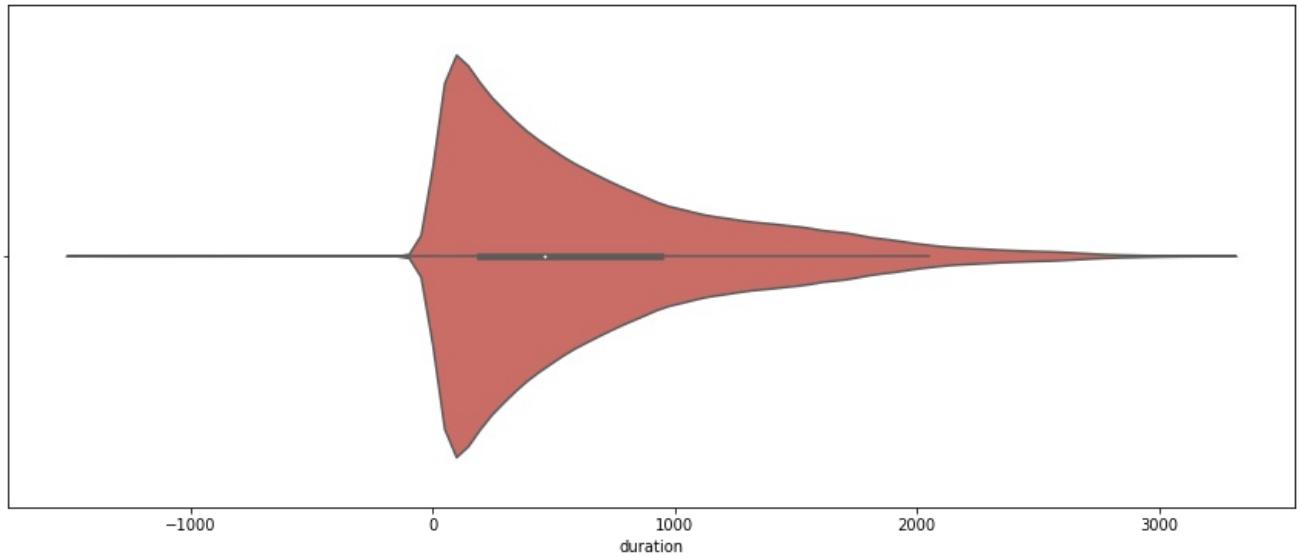
```
In [29]: for i in continuous:  
    plt.figure(figsize=(15,6))  
    sns.violinplot(i, data = df, palette='hls')  
    plt.show()
```



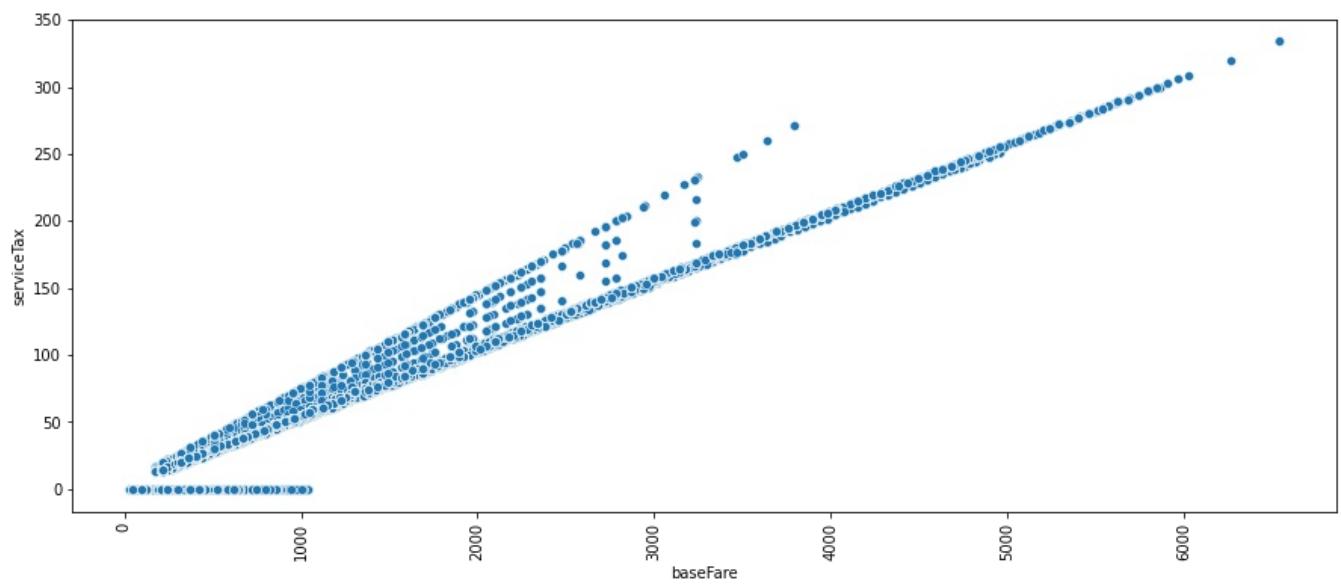


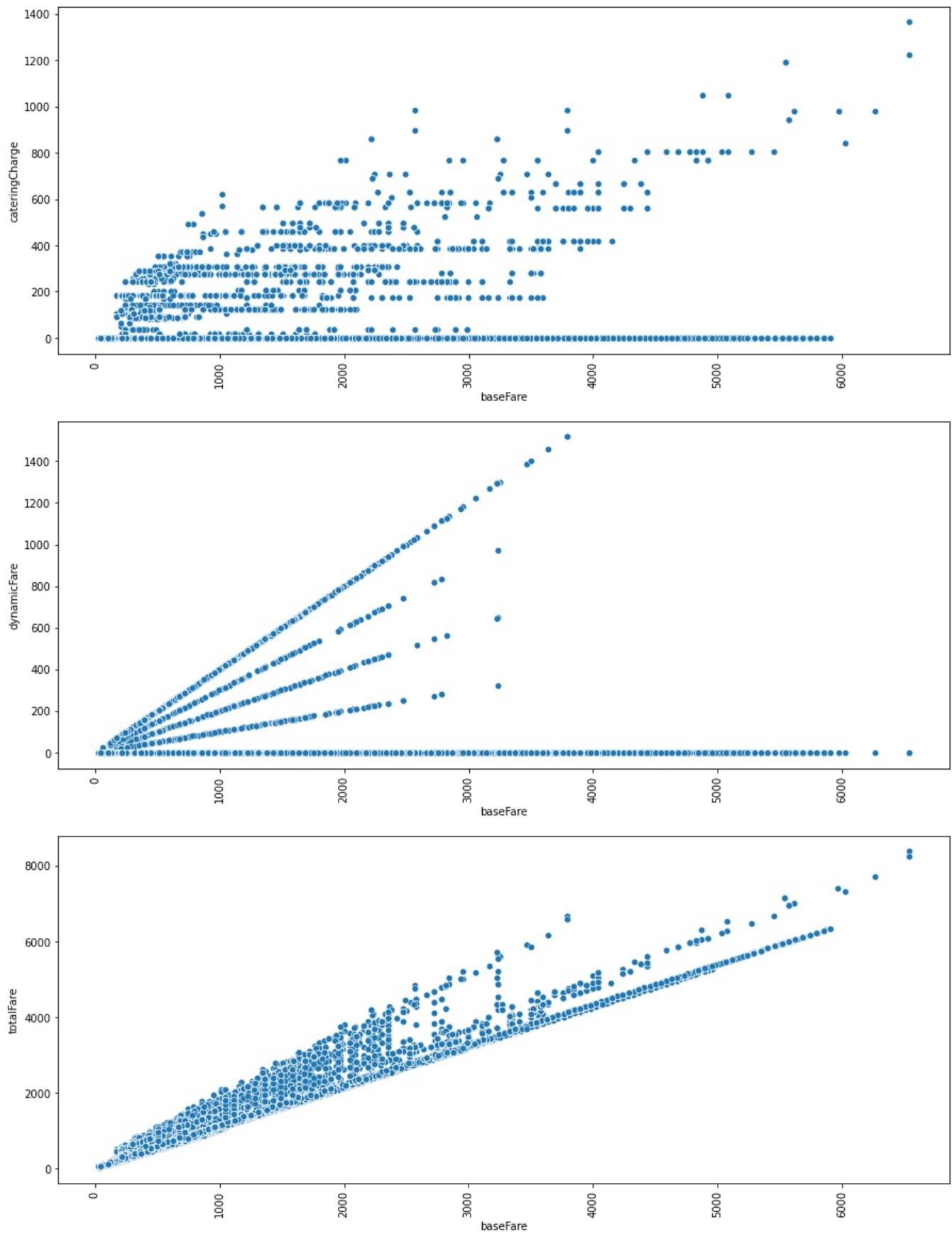


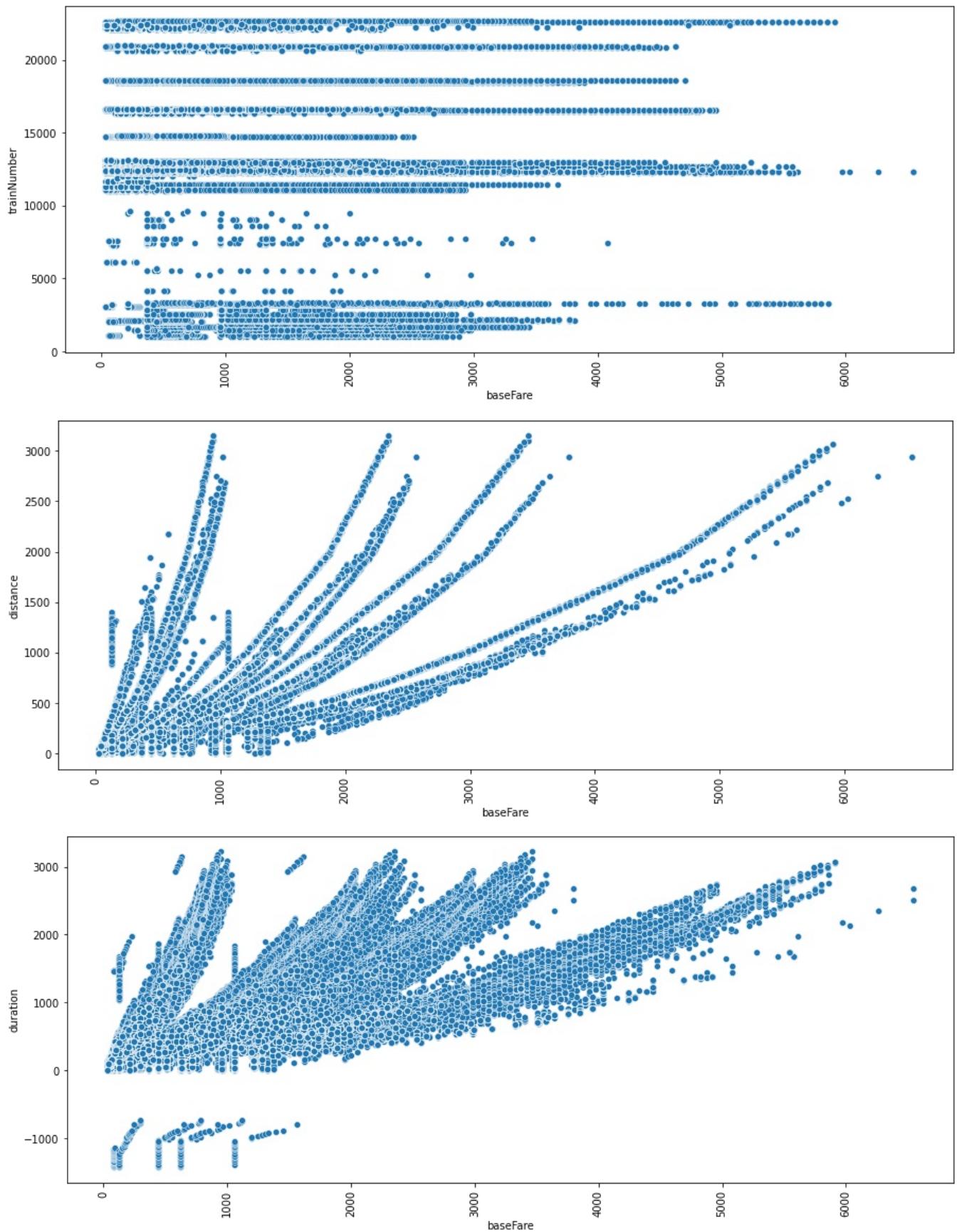


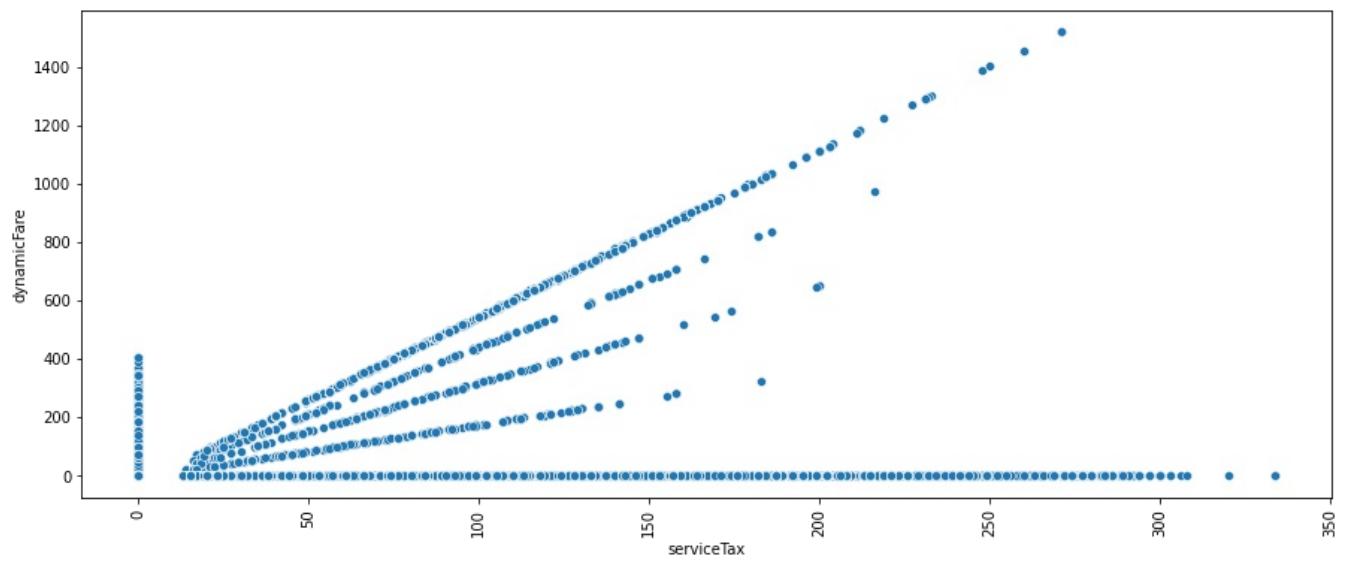
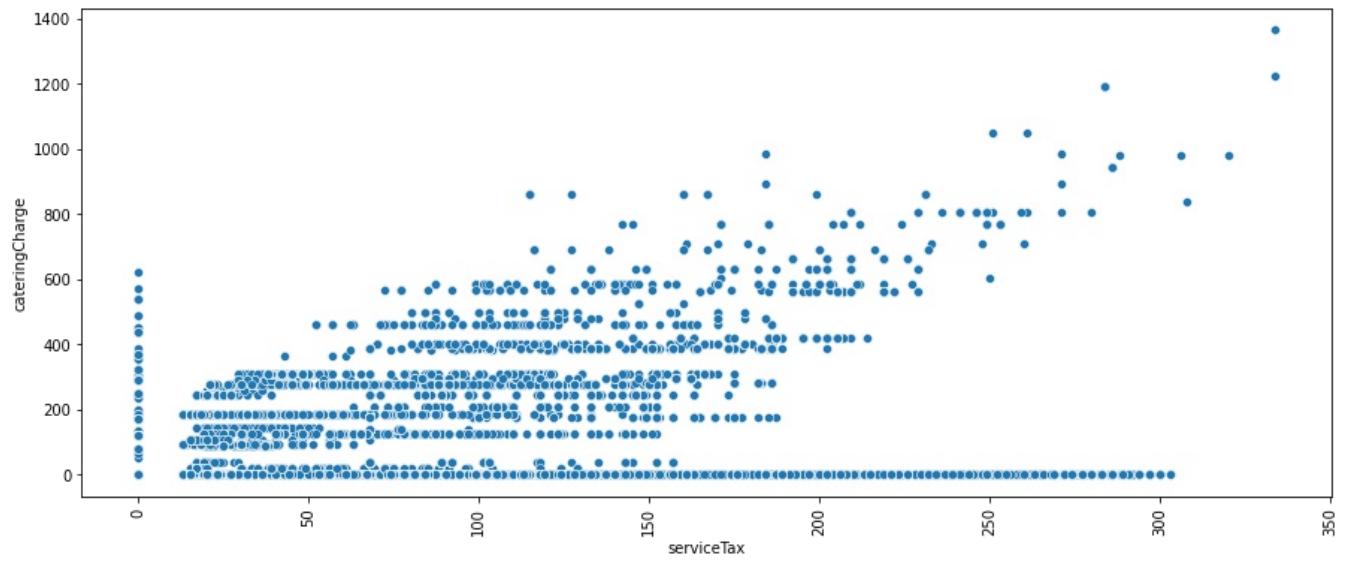
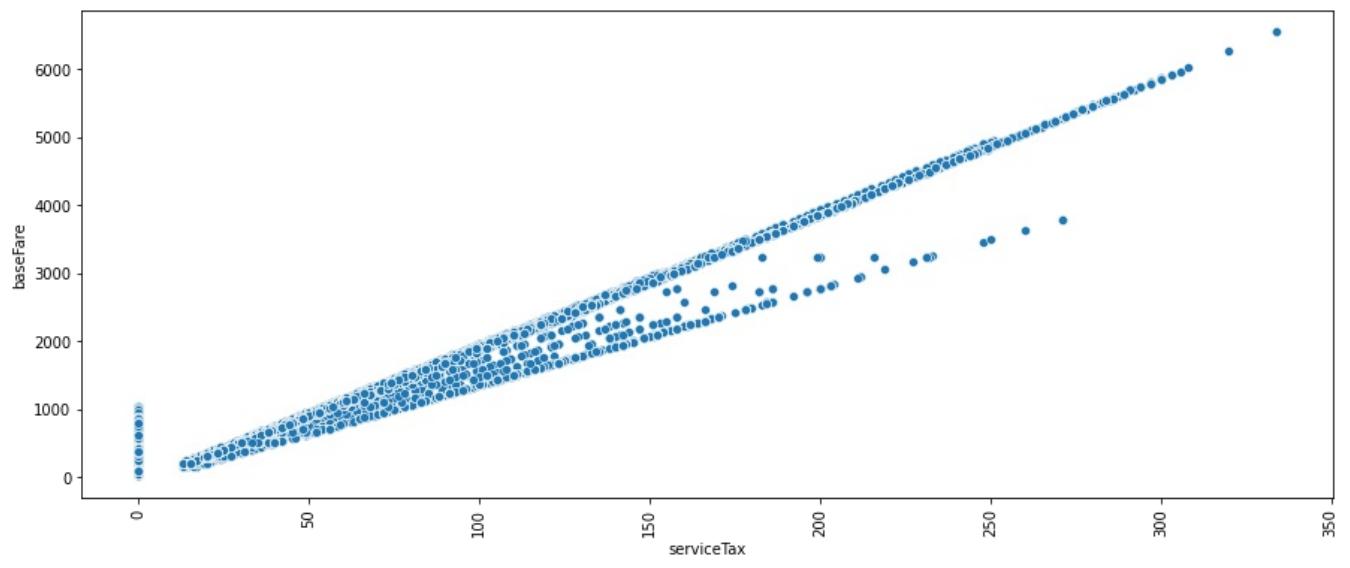


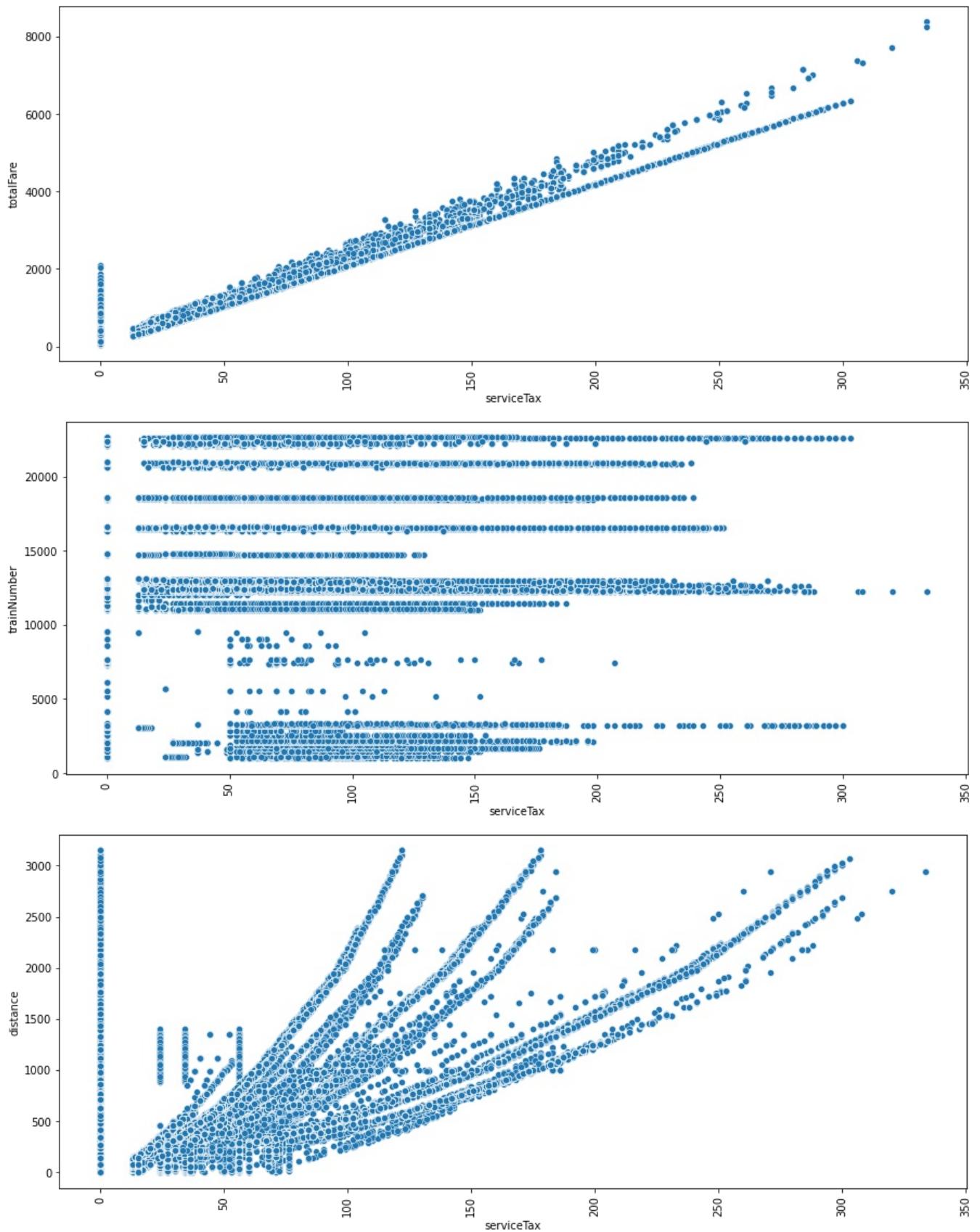
```
In [30]: for i in continuous:
    for j in continuous:
        if i != j:
            plt.figure(figsize=(15,6))
            sns.scatterplot(x = i, y = j, data = df, ci = None, palette='hls')
            plt.xticks(rotation = 90)
            plt.show()
```

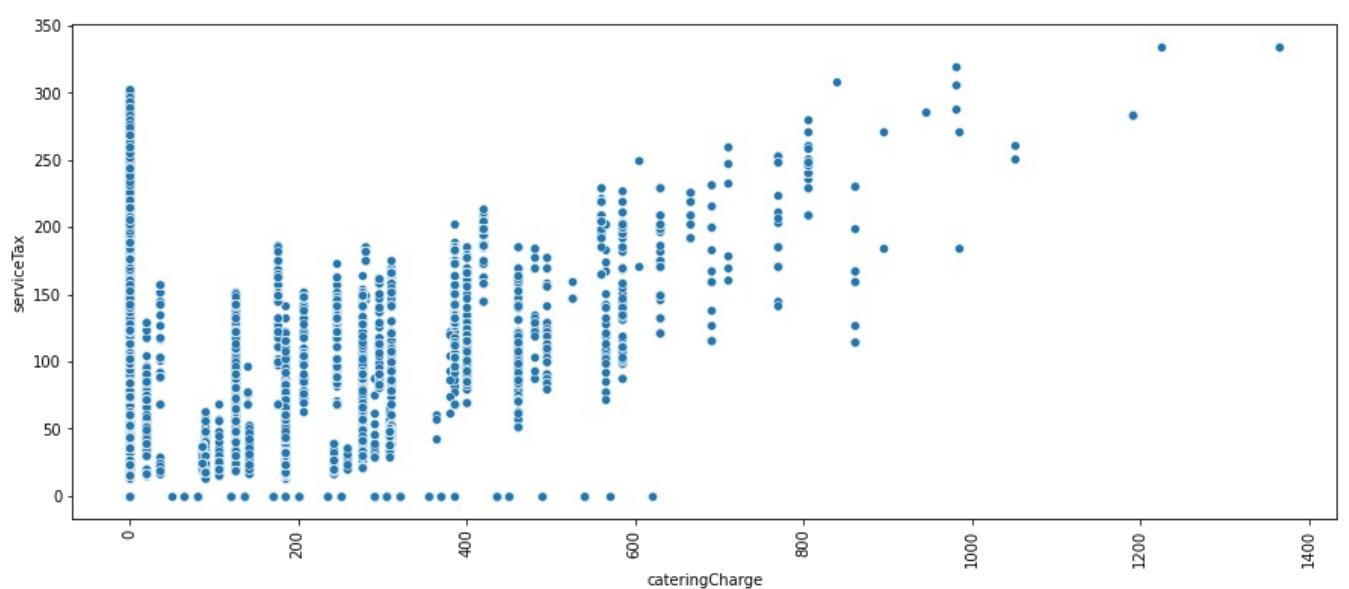
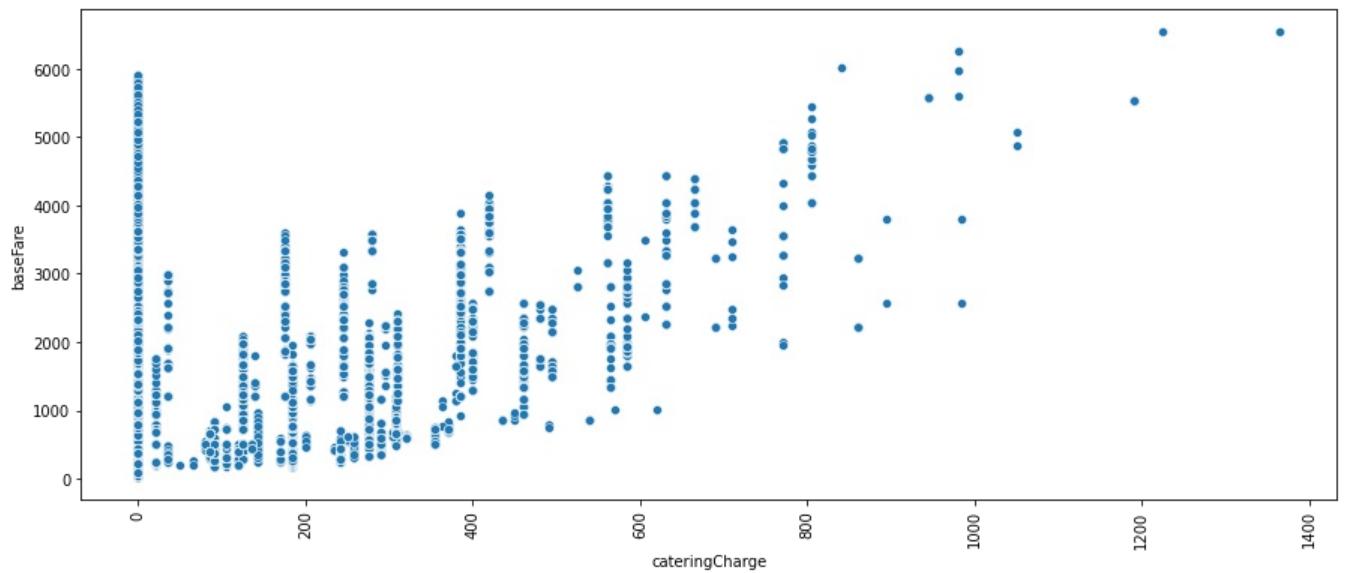
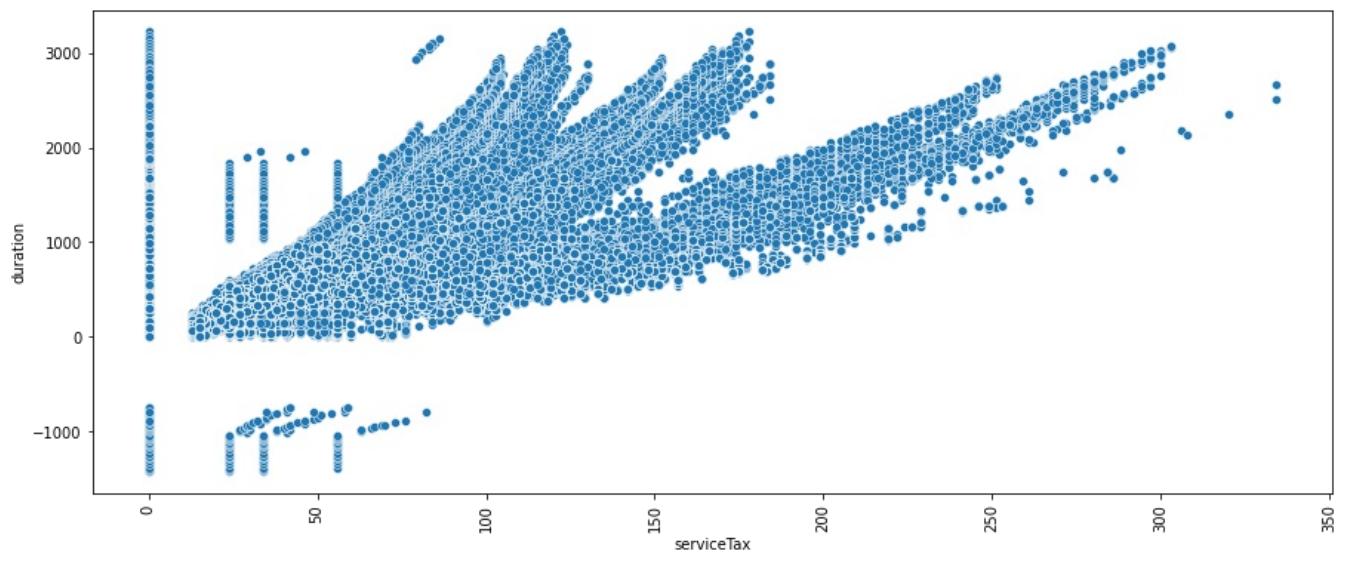


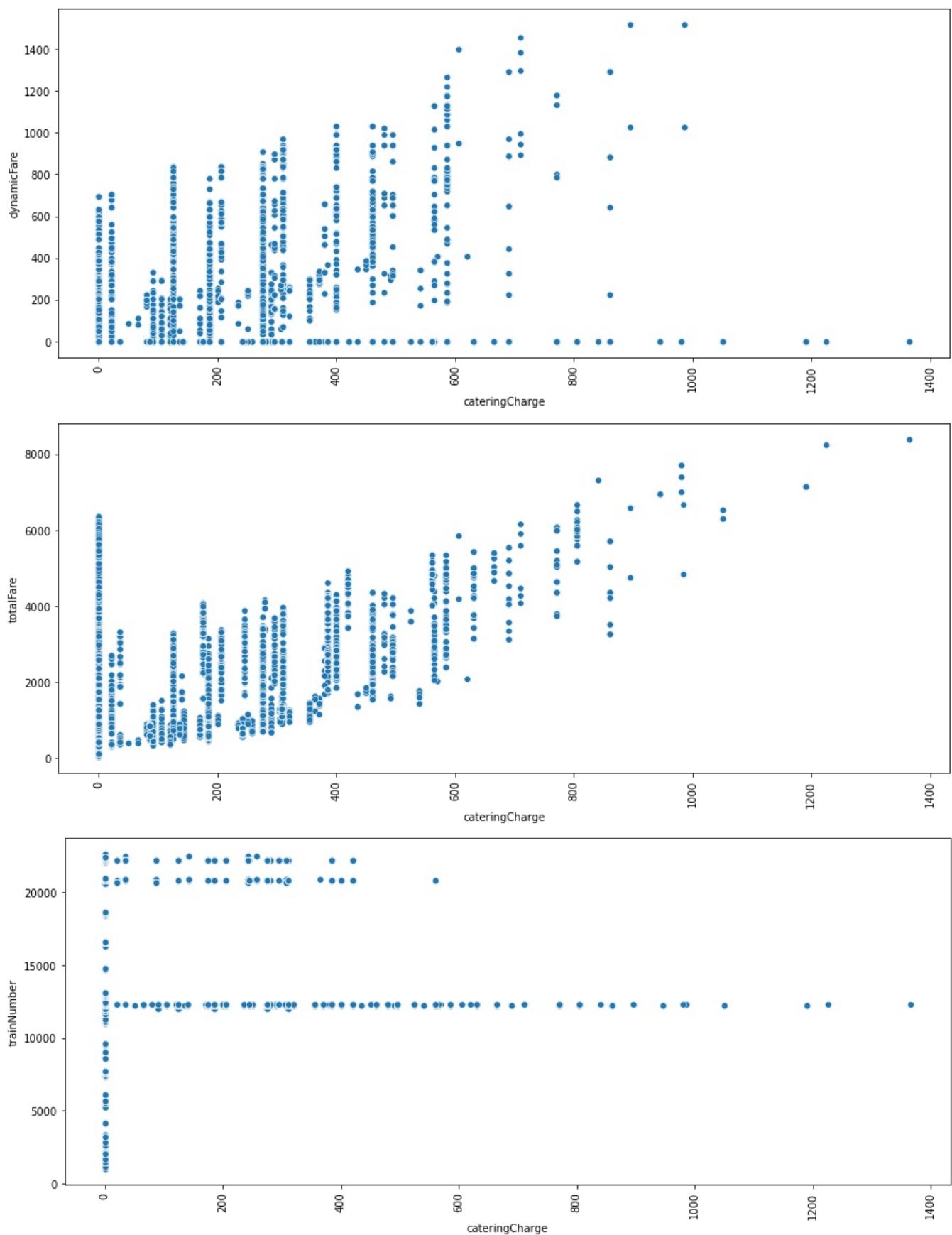


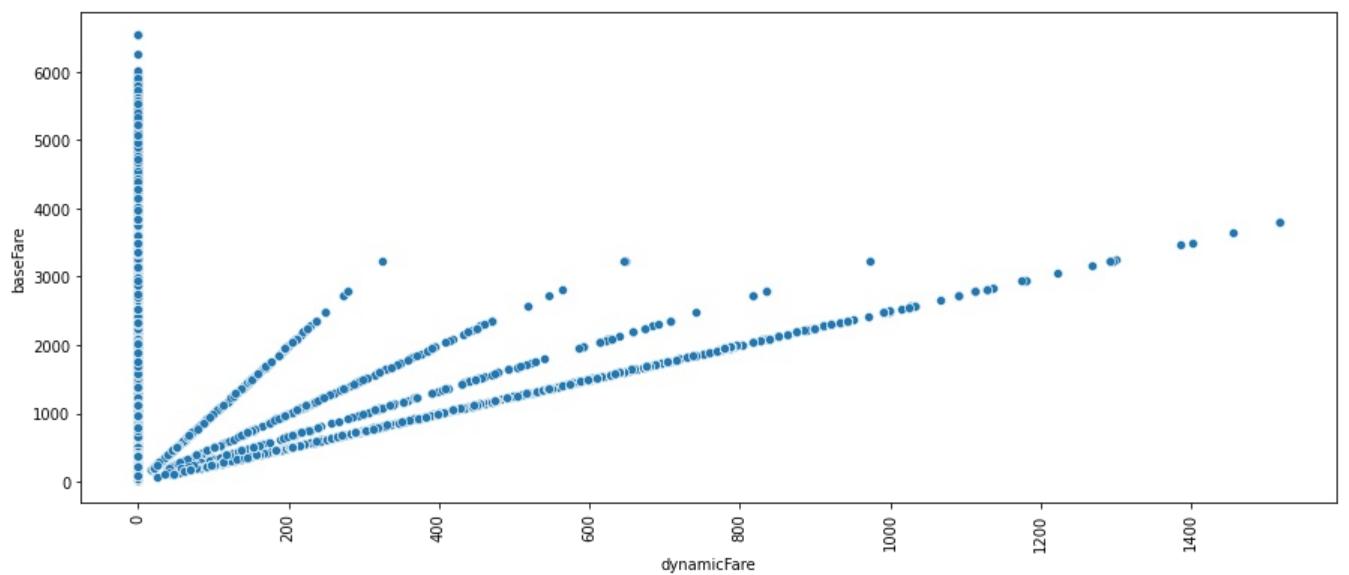
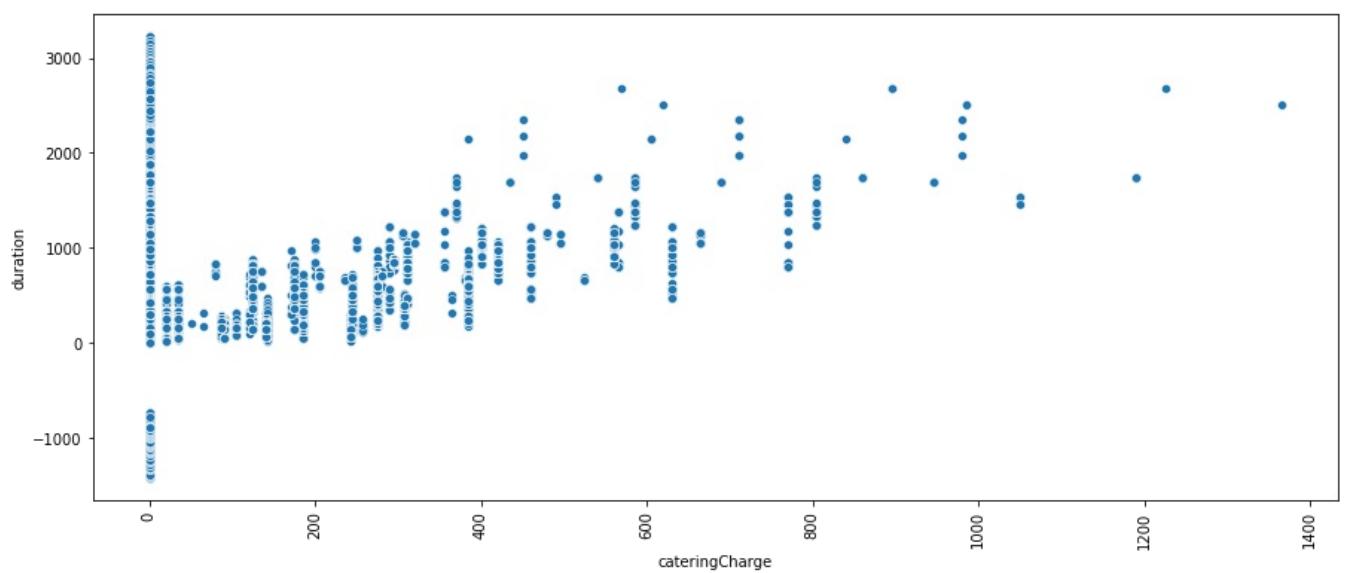
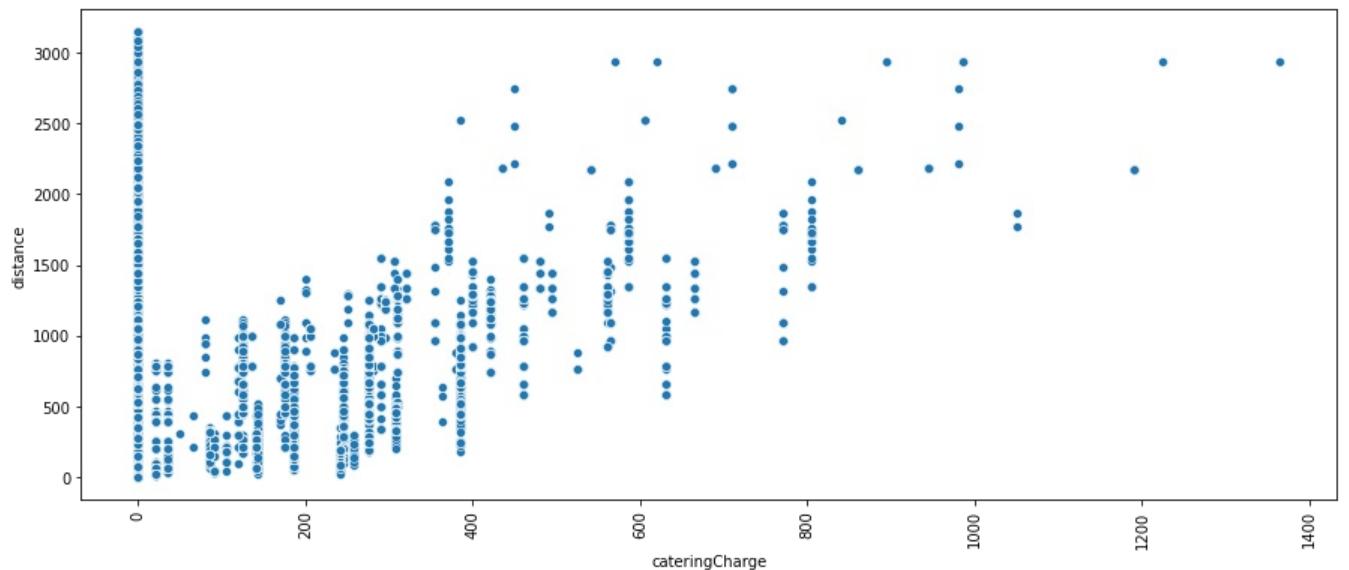


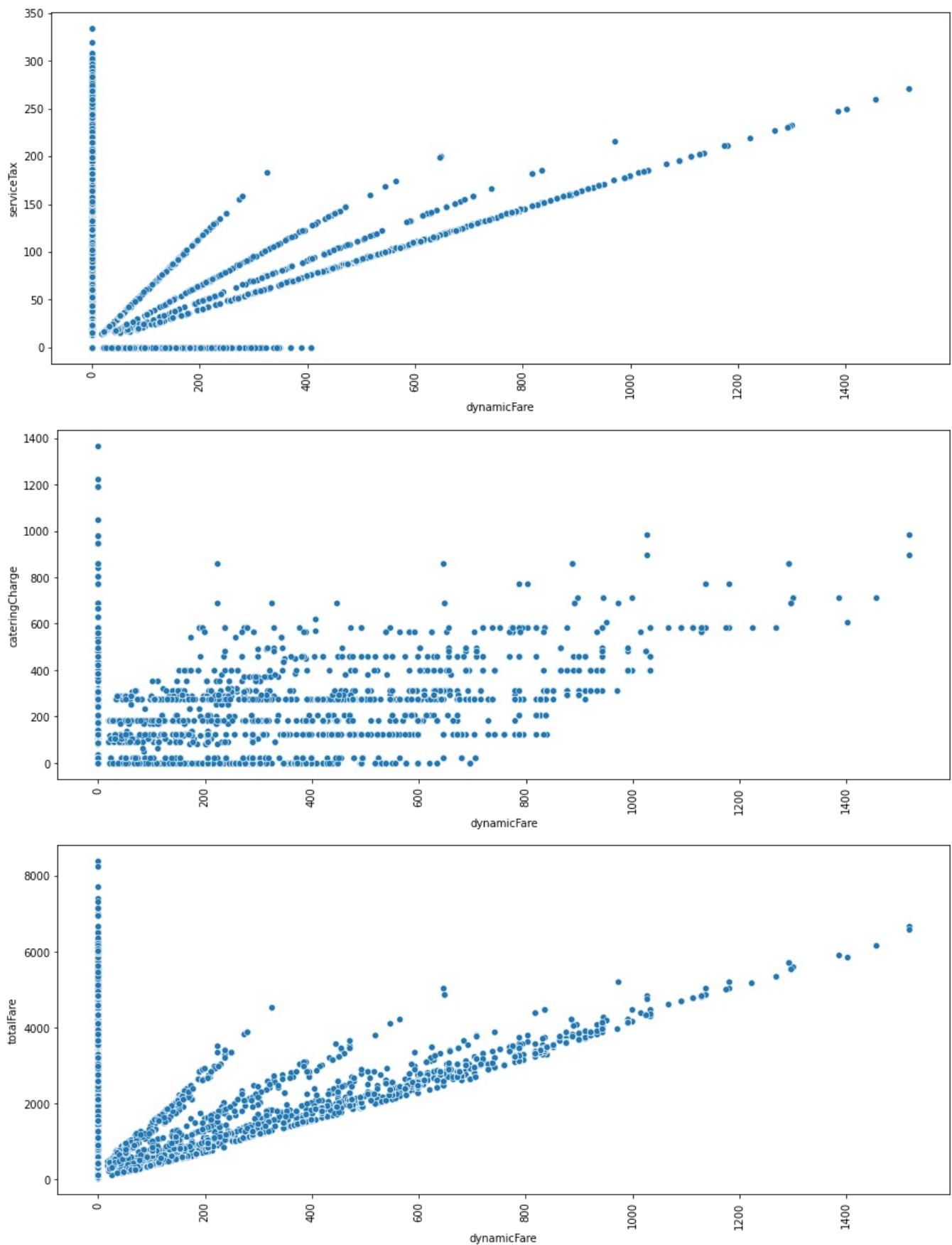


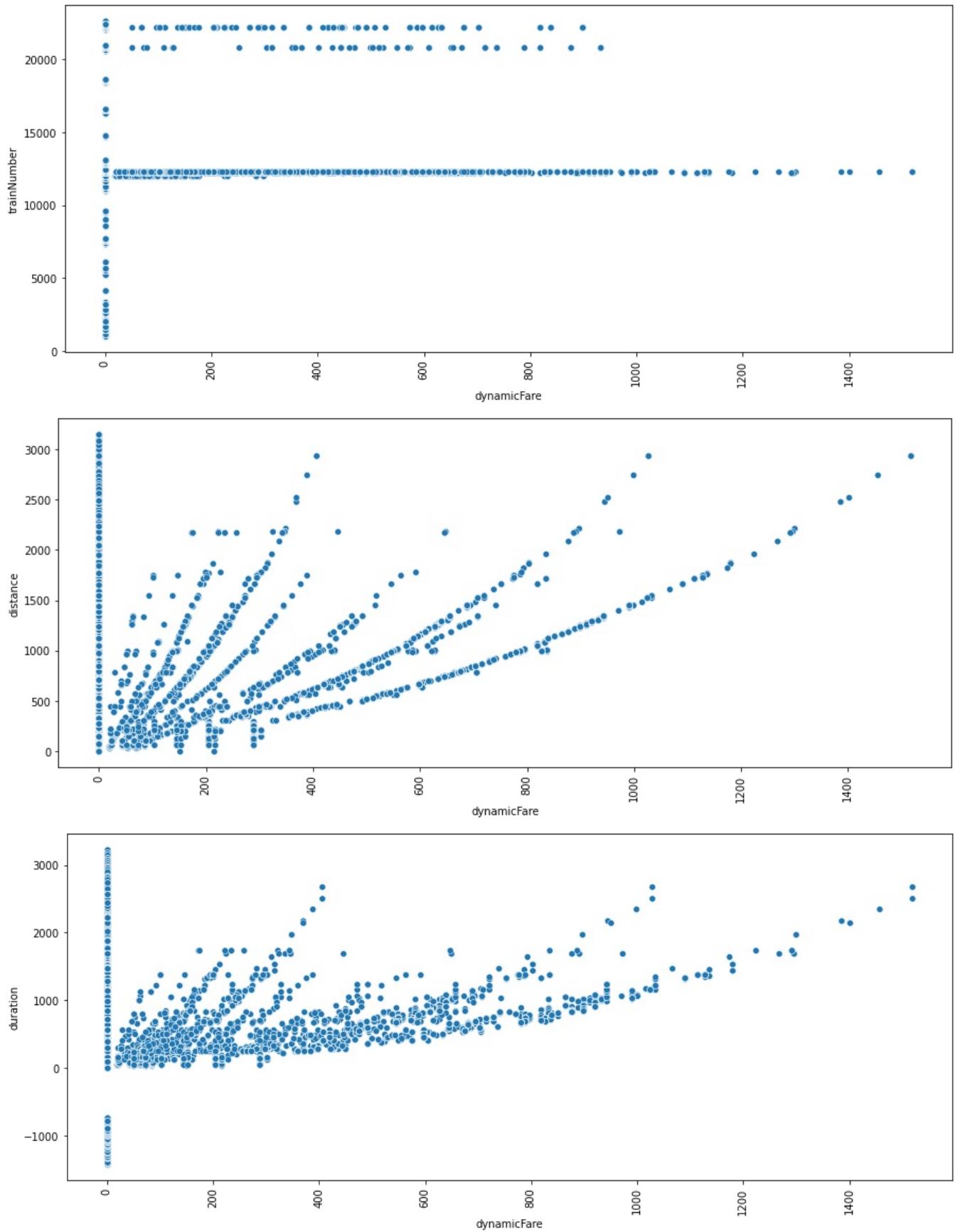


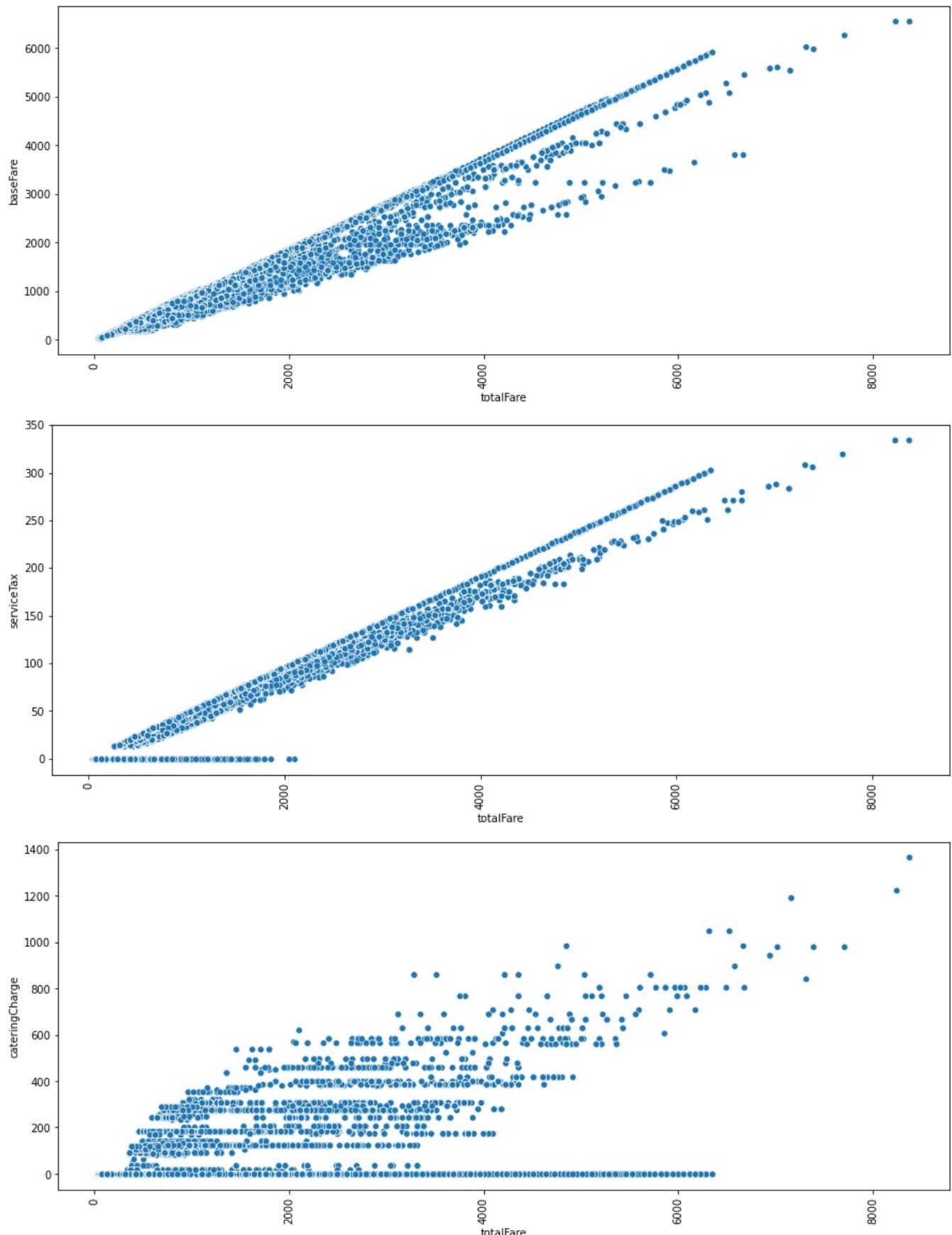


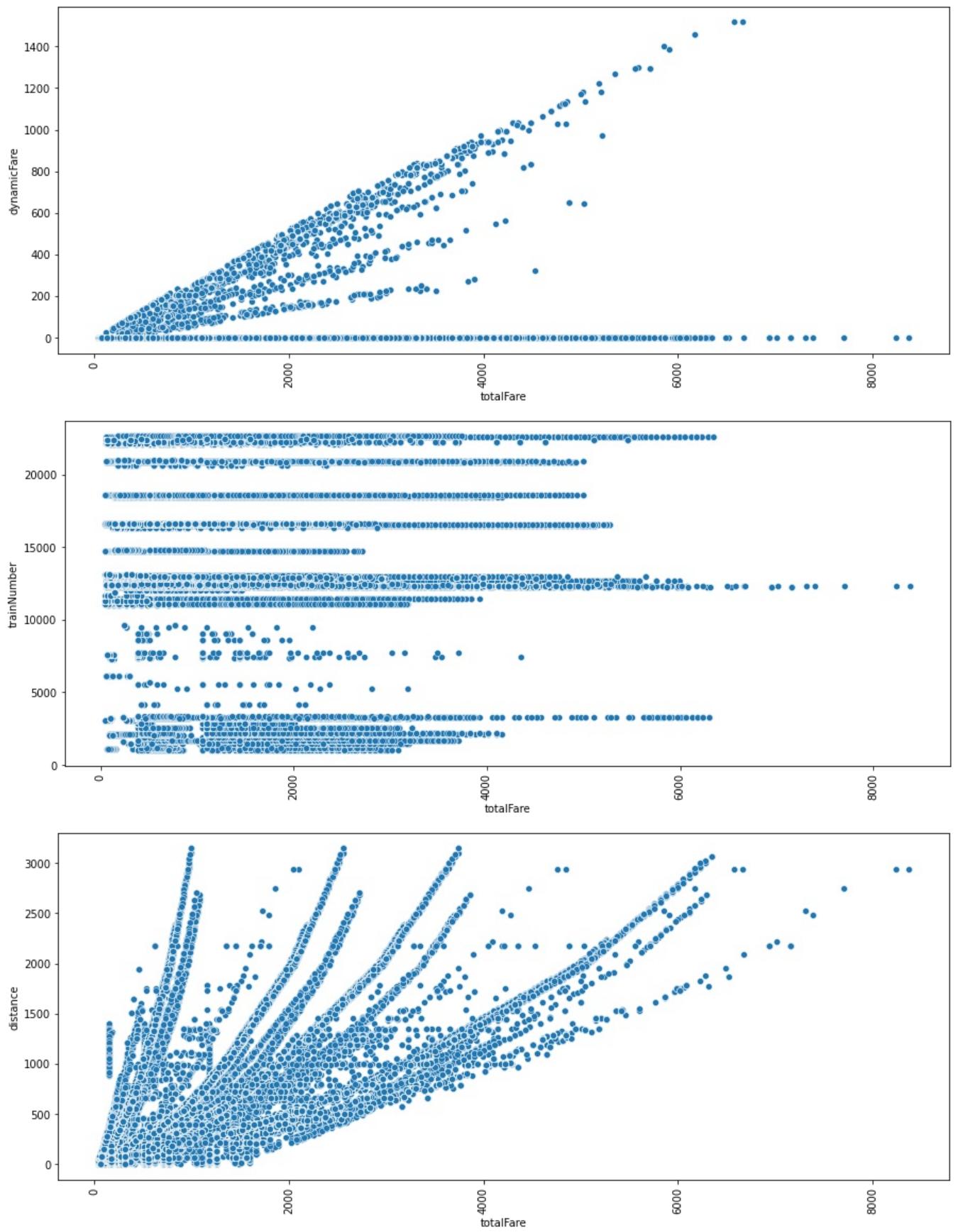


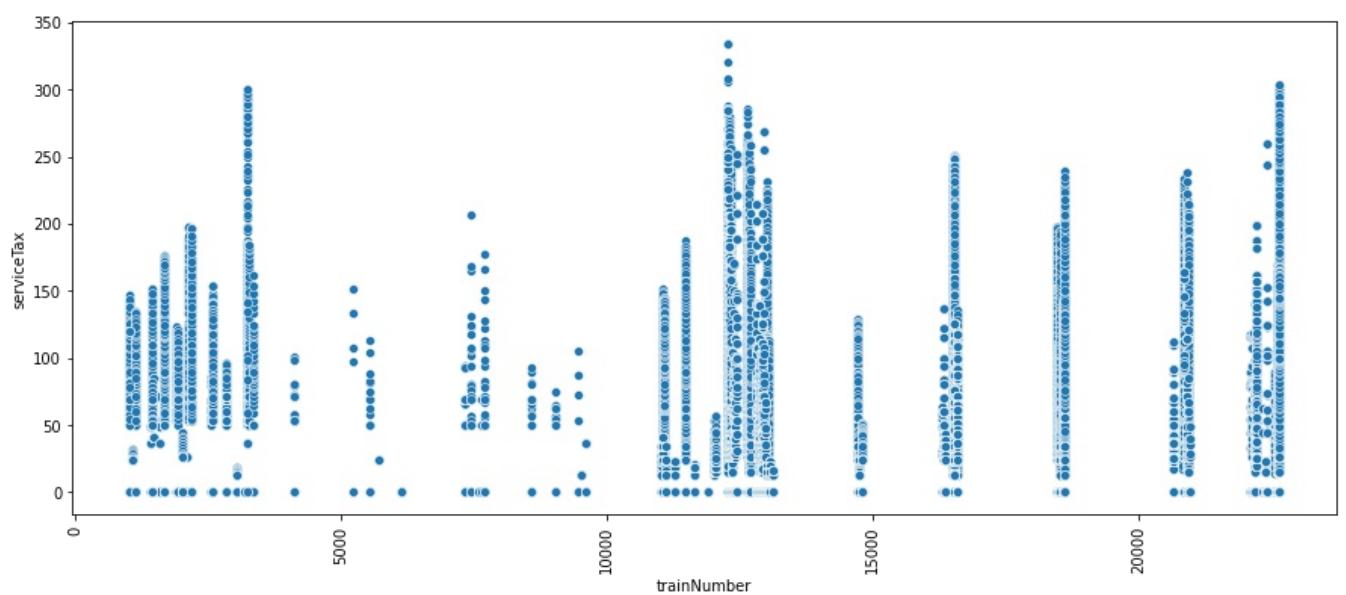
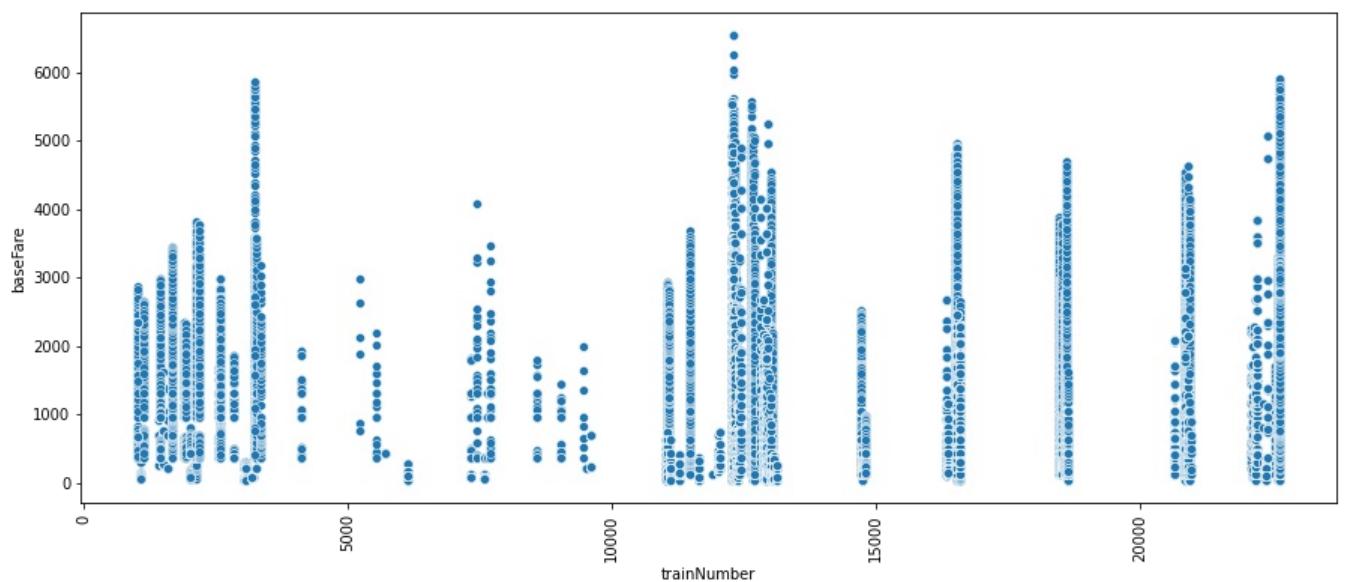
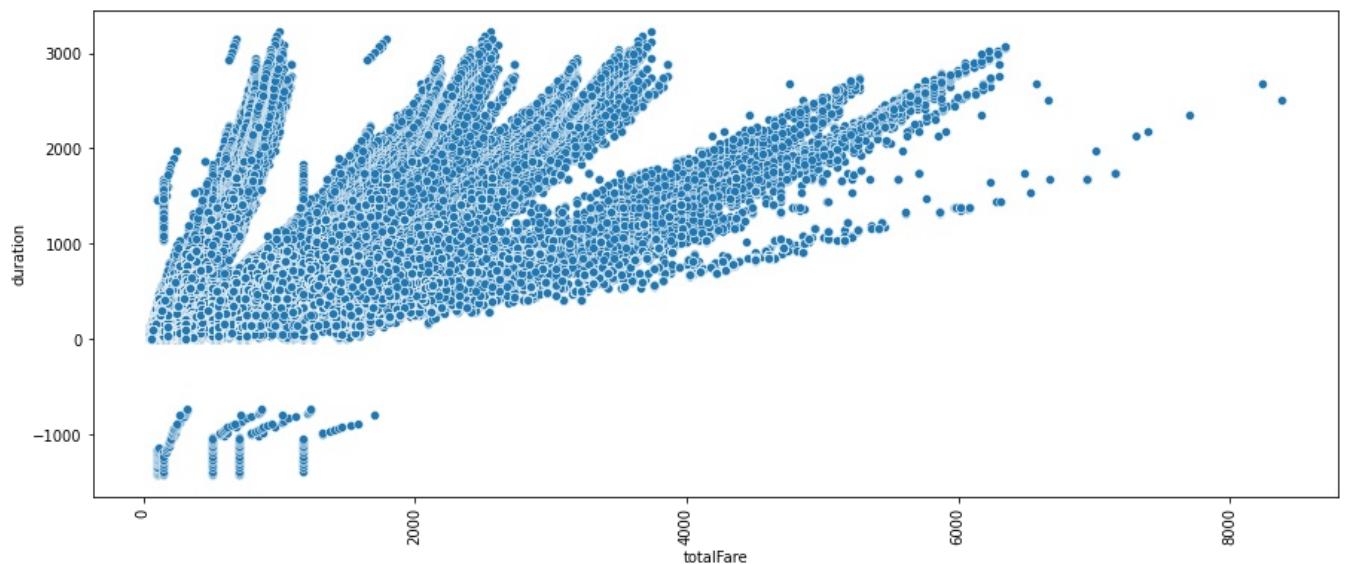


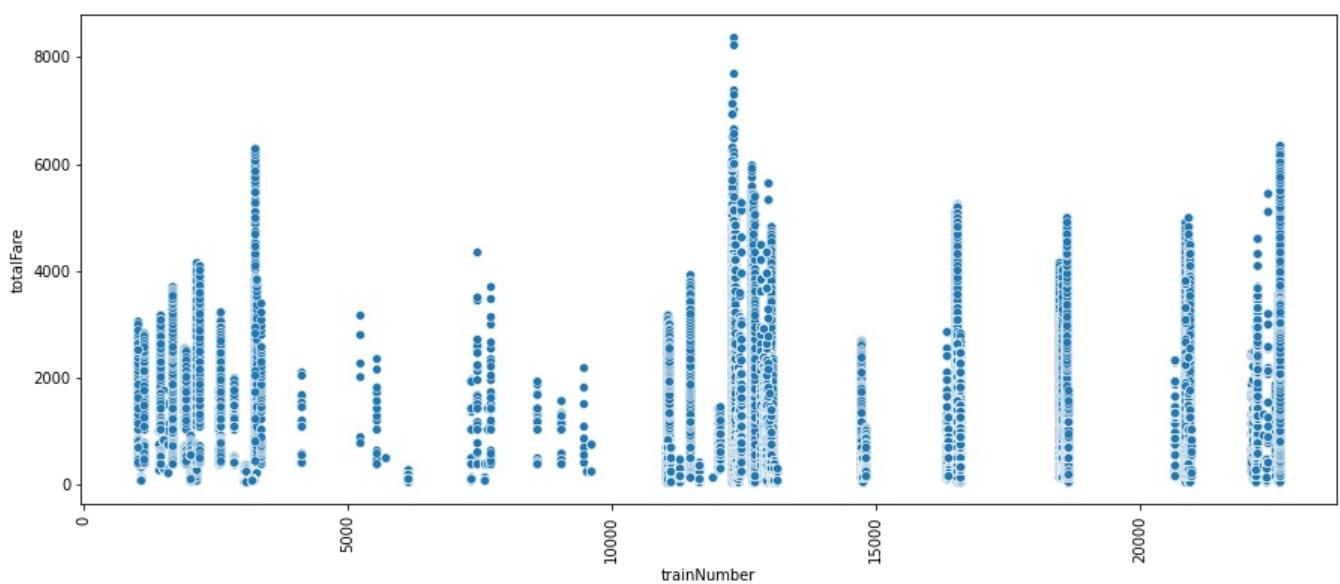
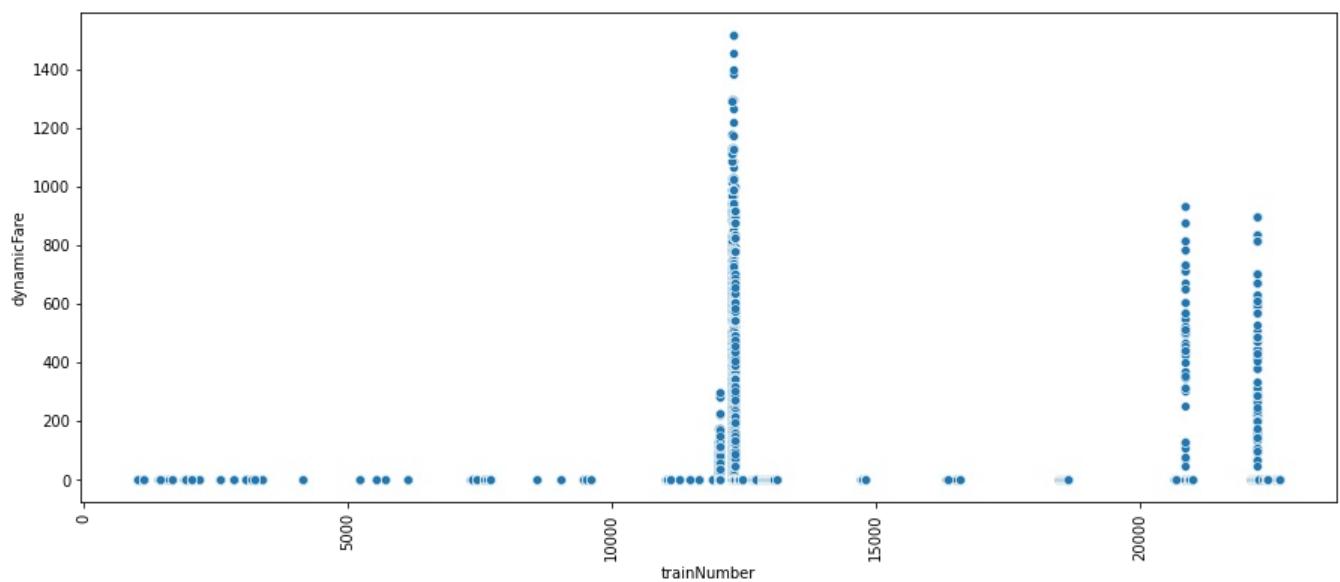
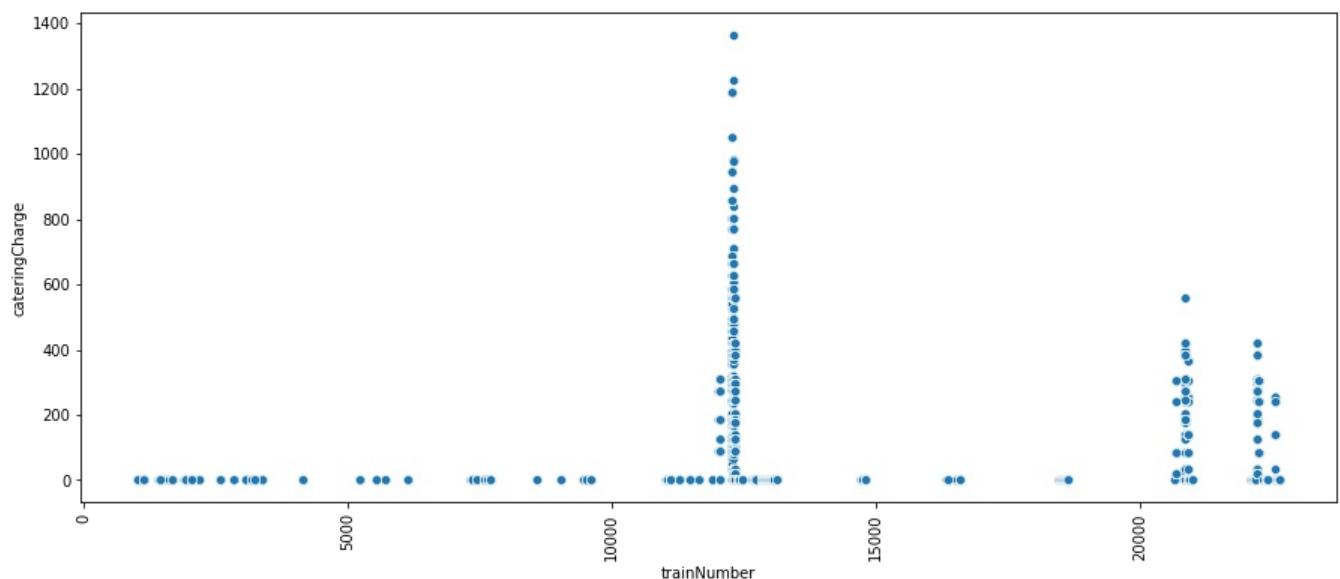


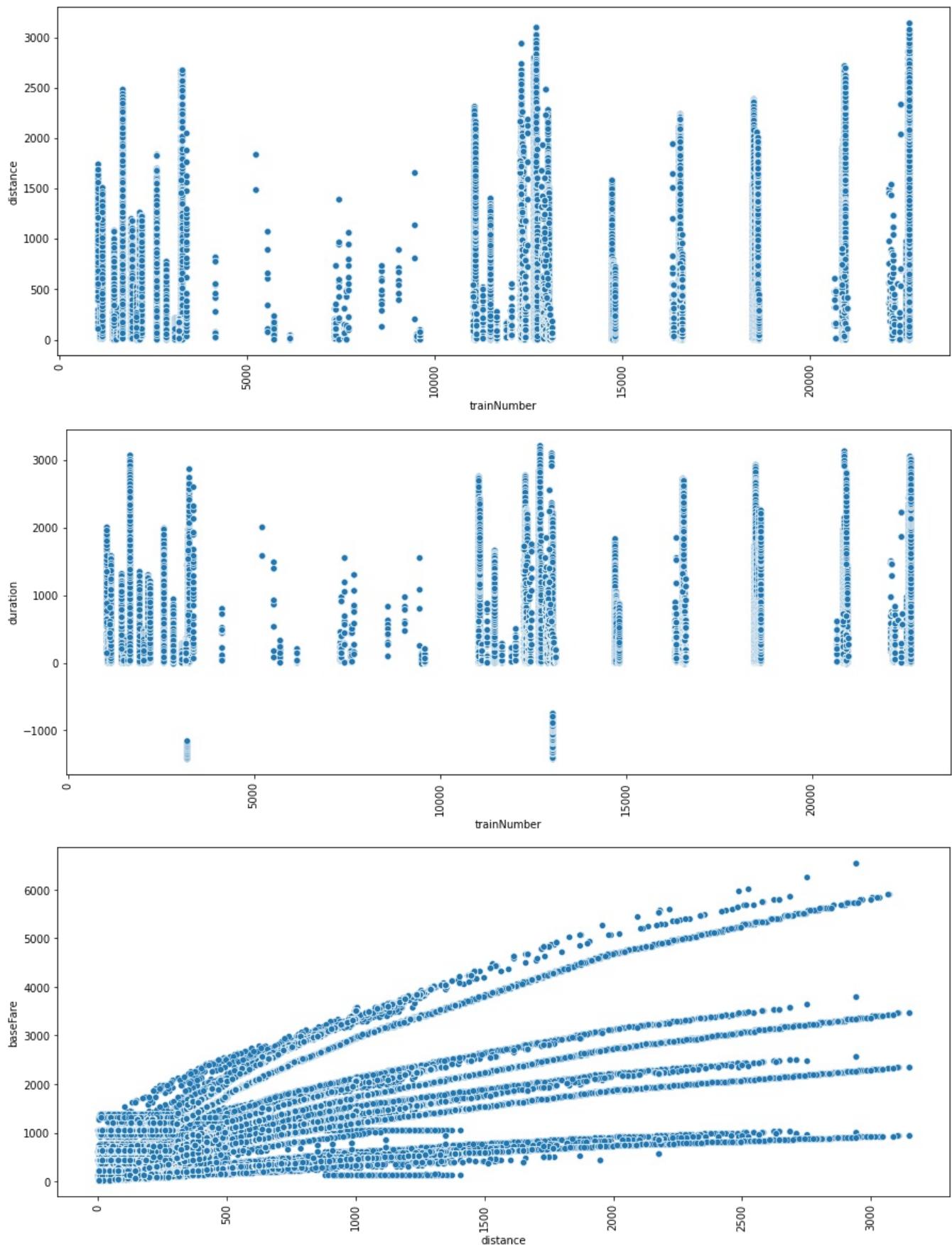


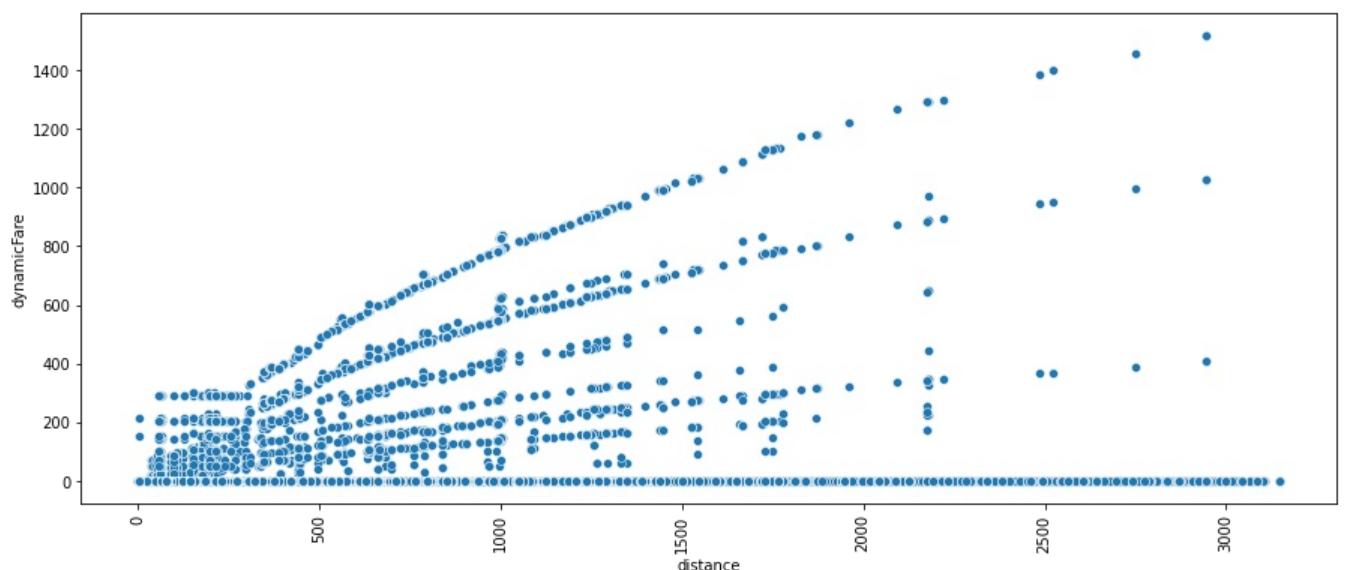
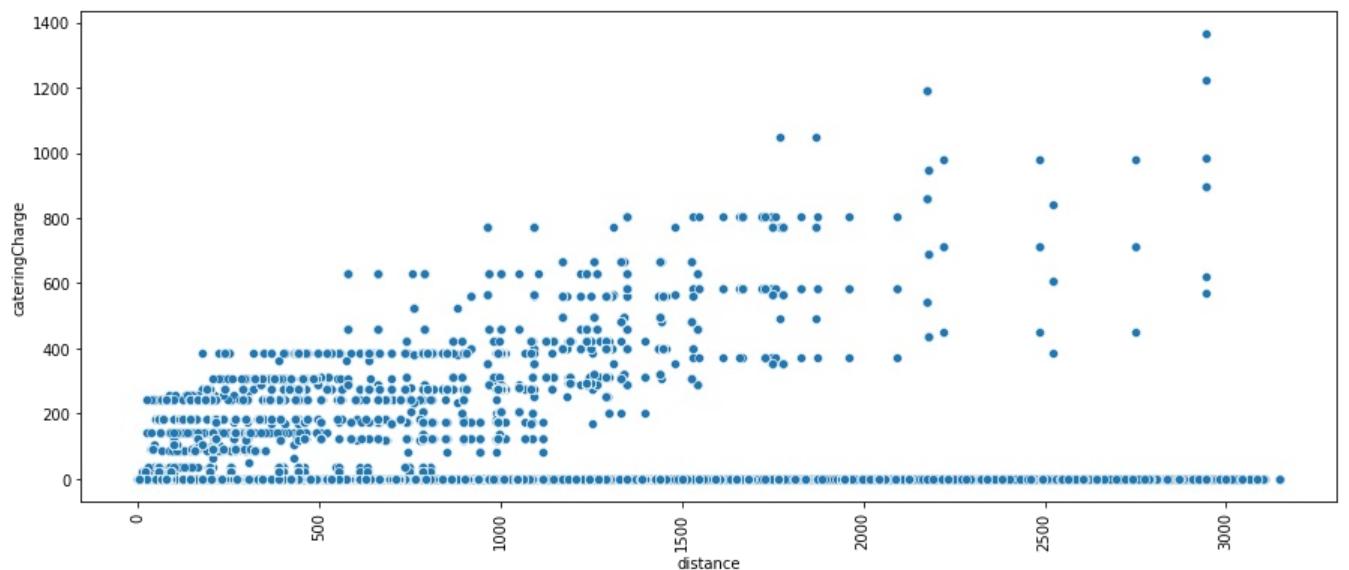
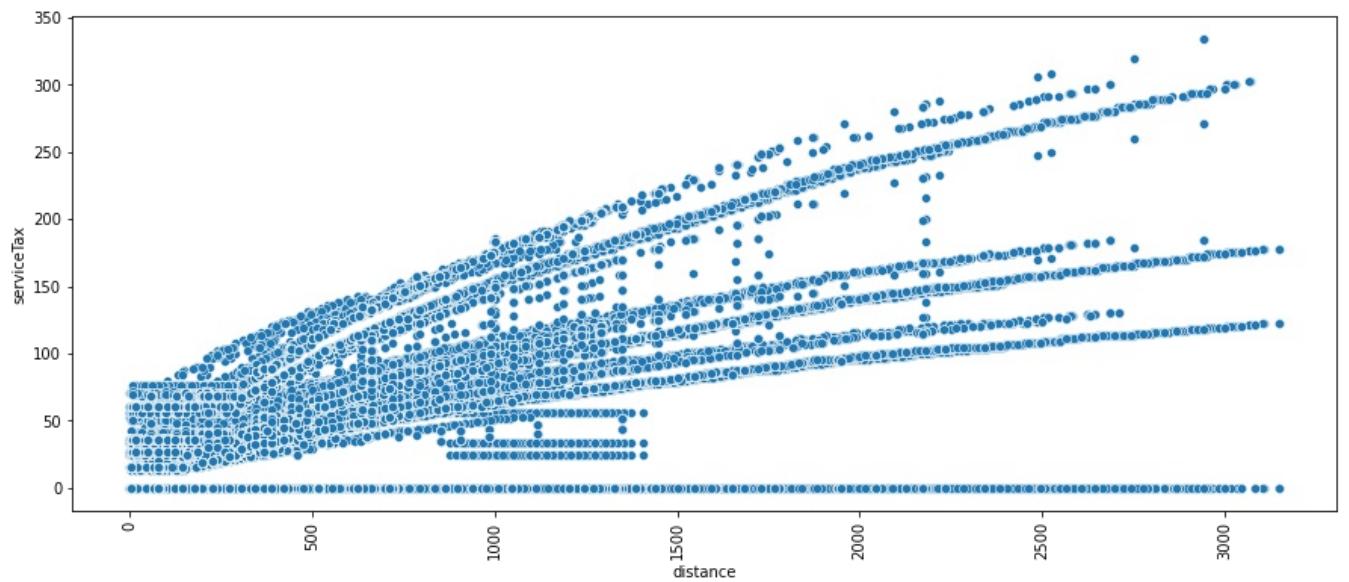


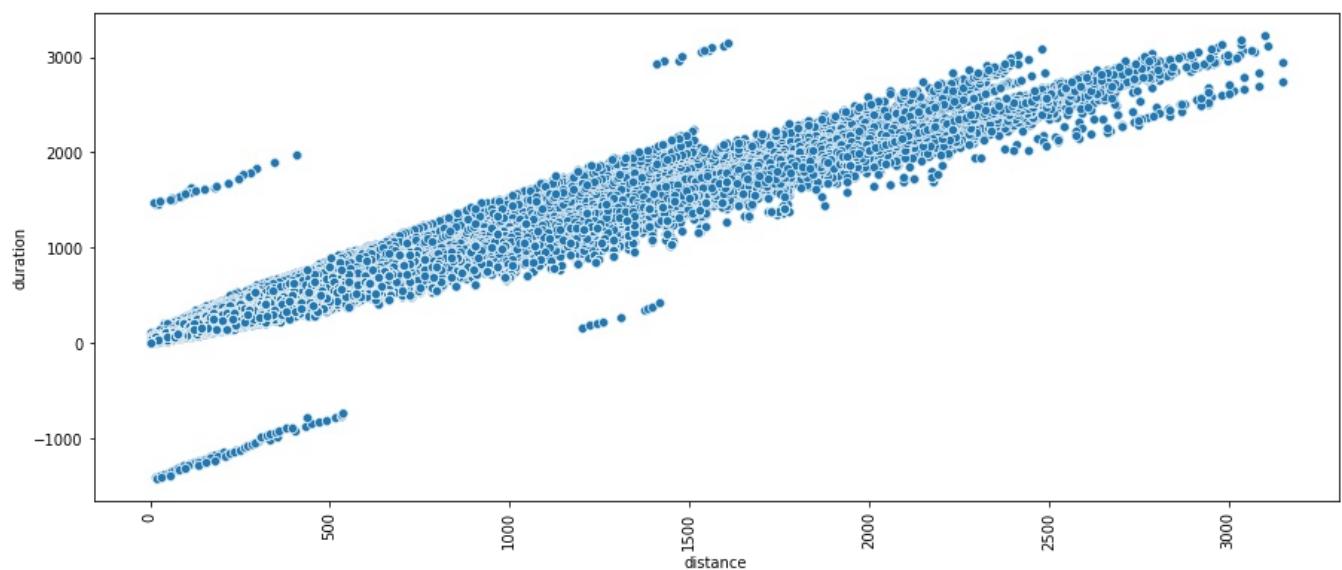
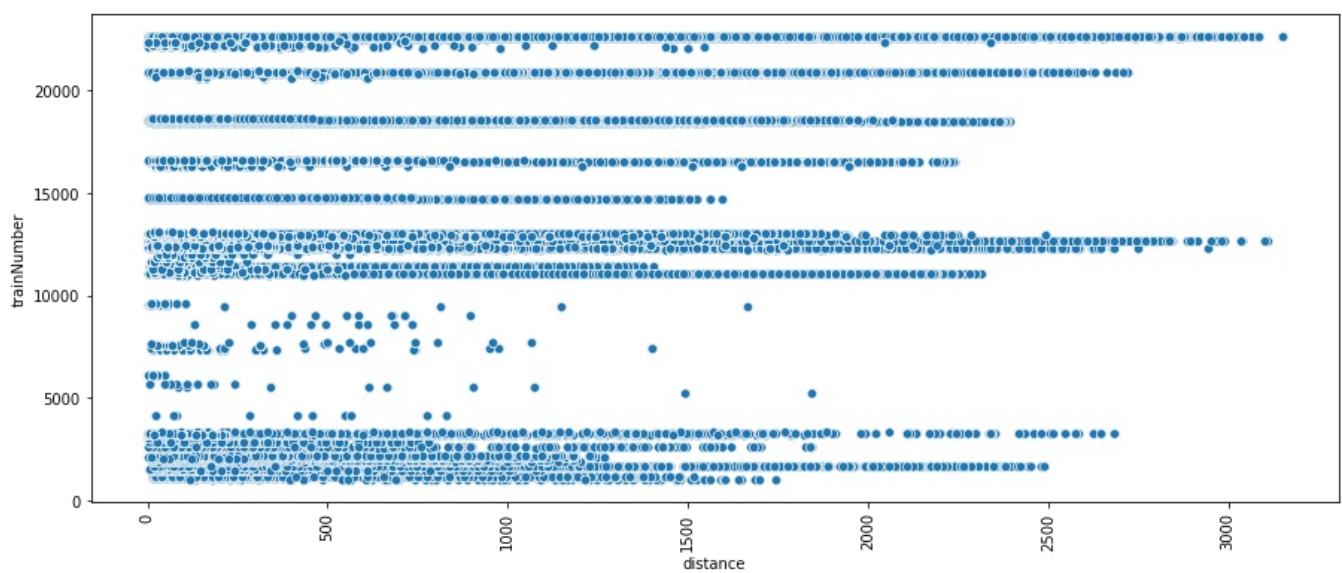
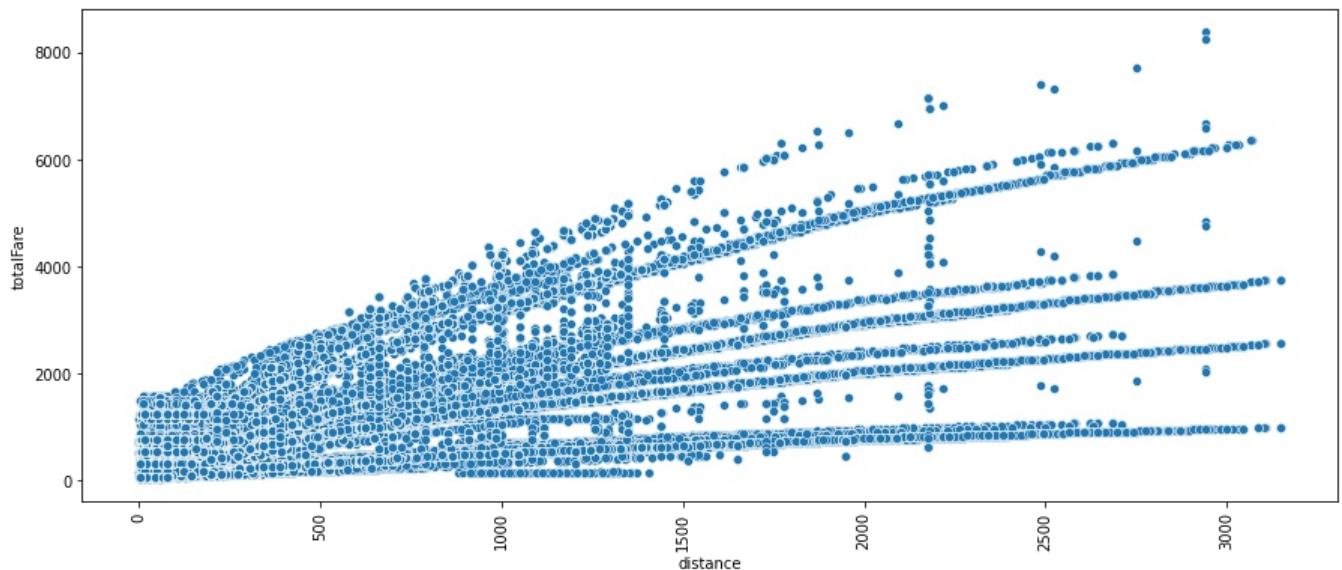


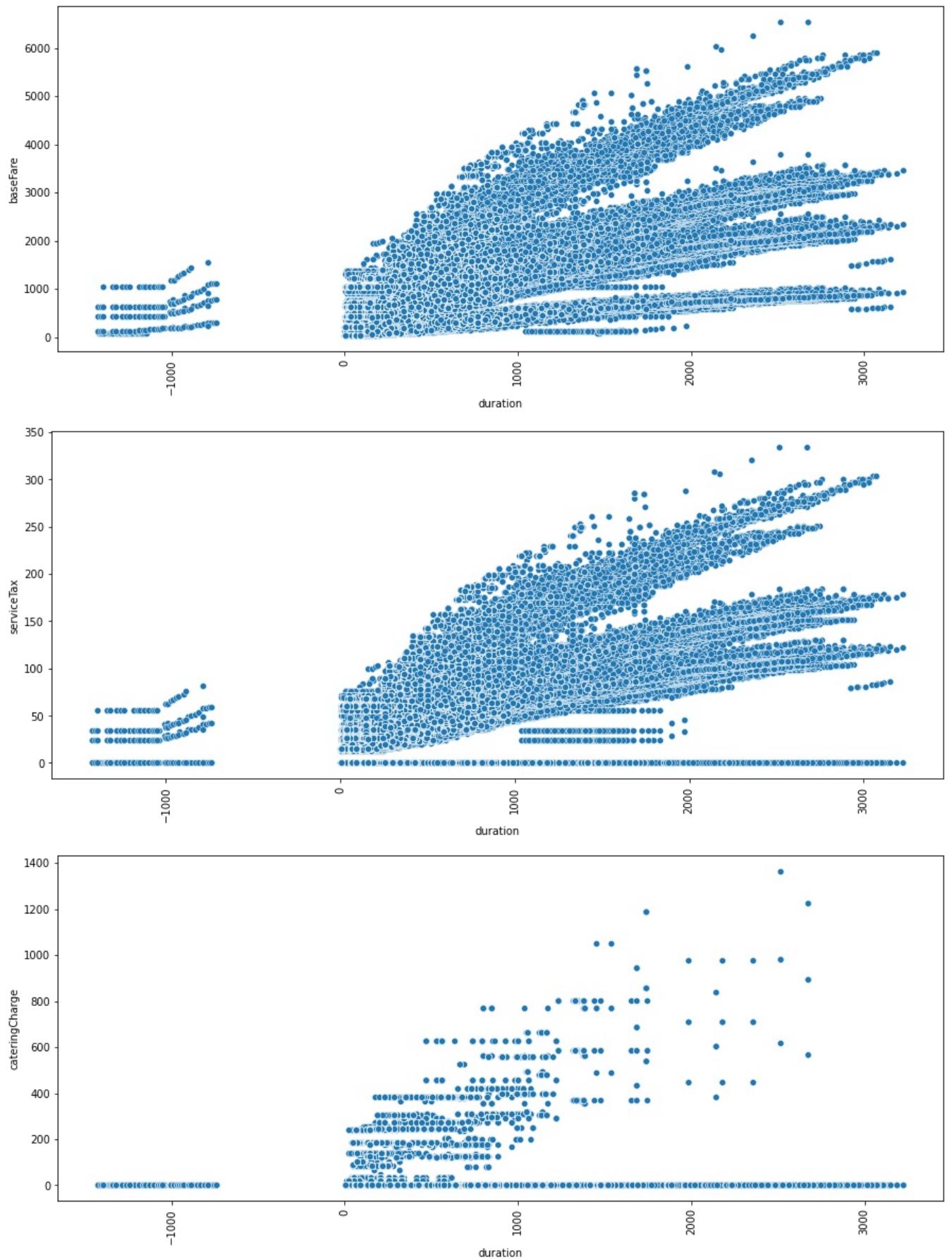


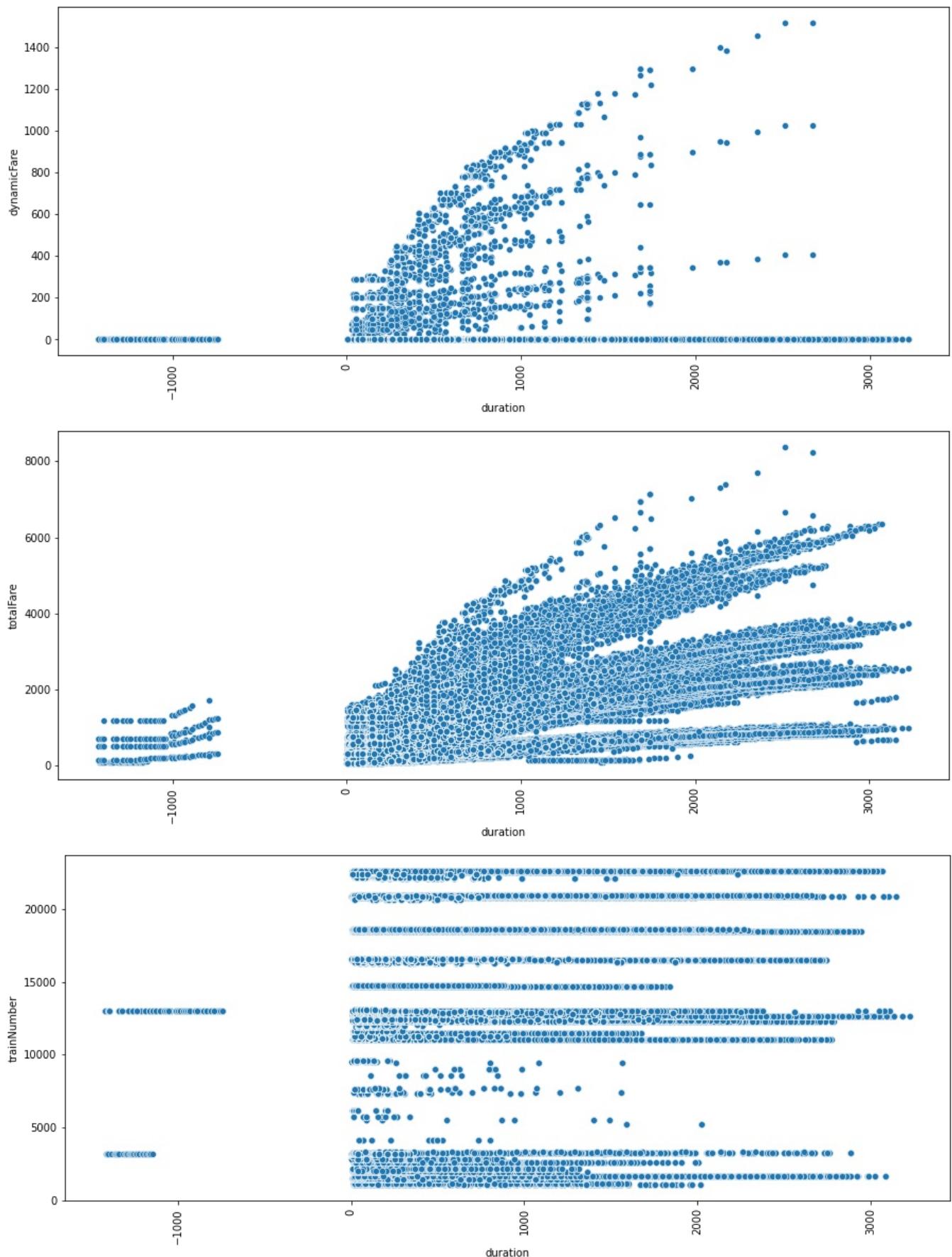


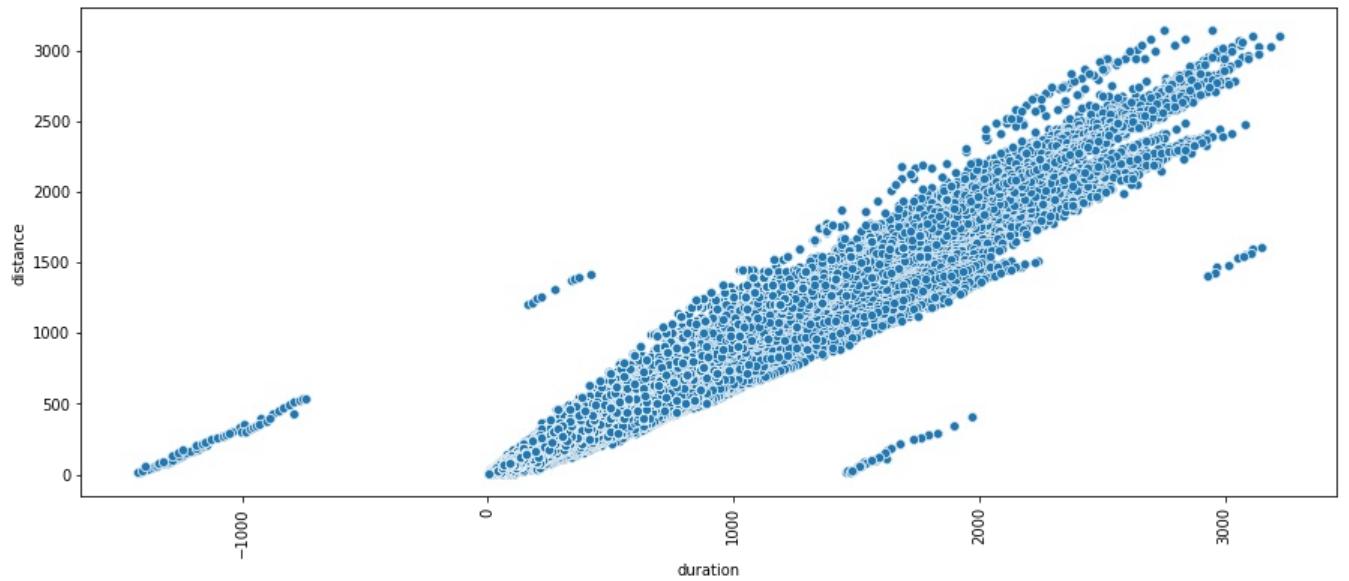




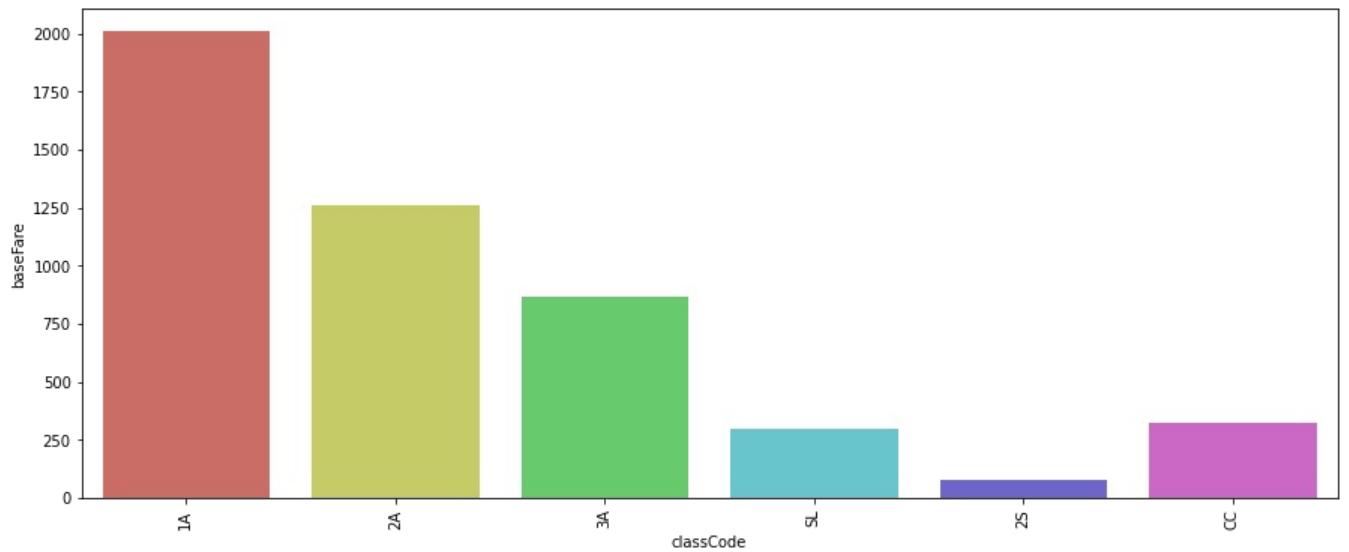


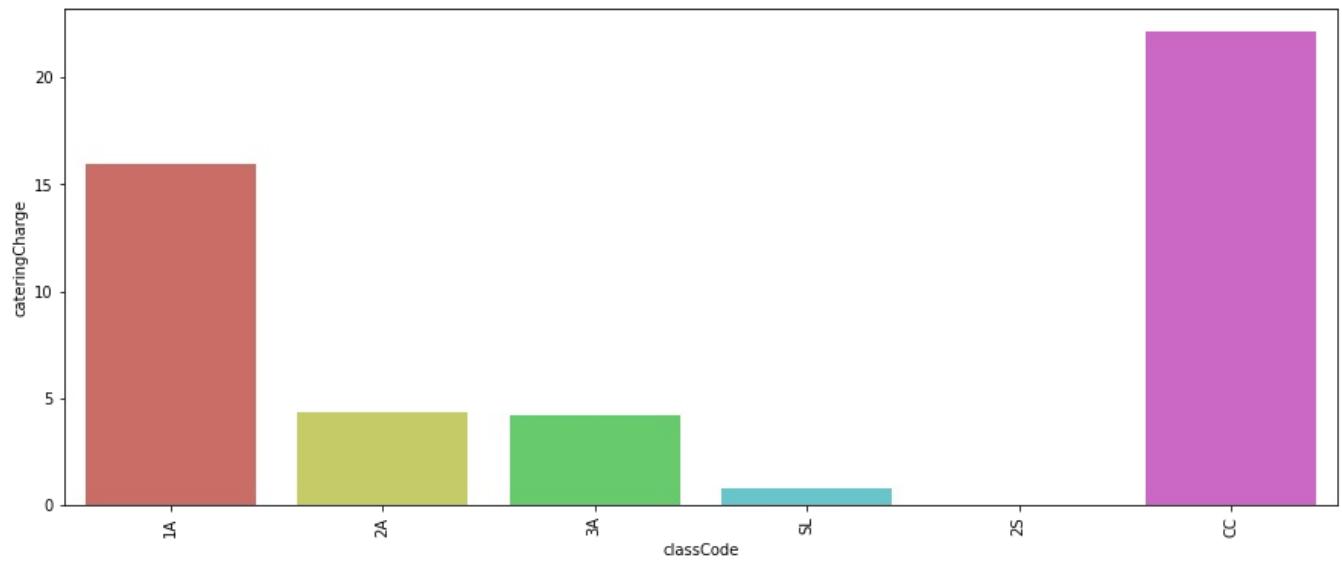
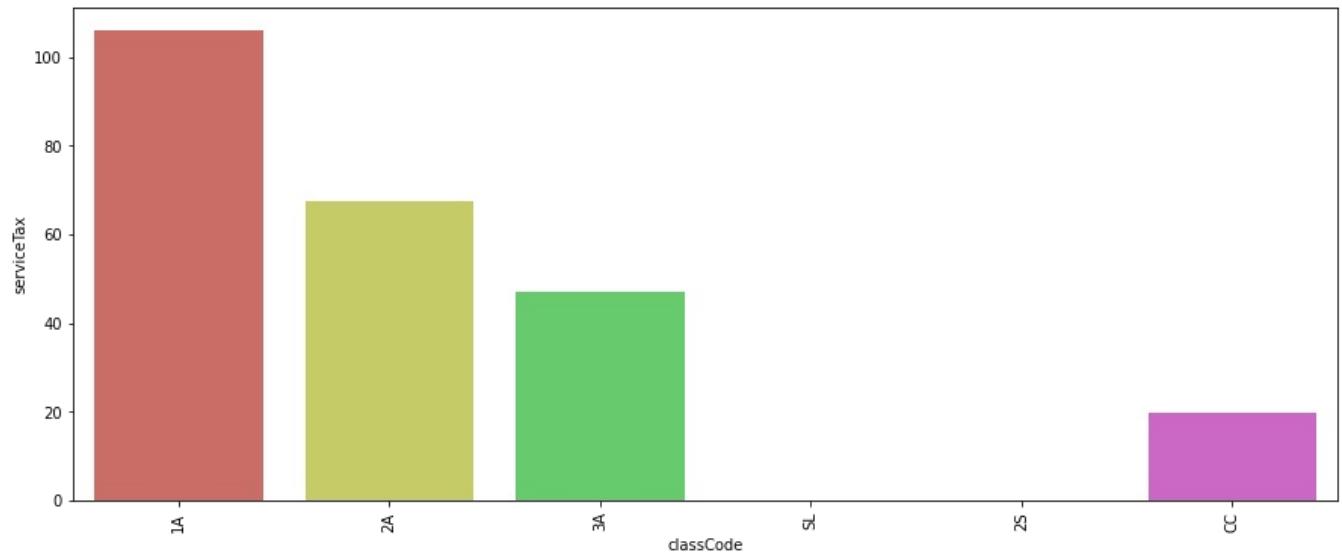


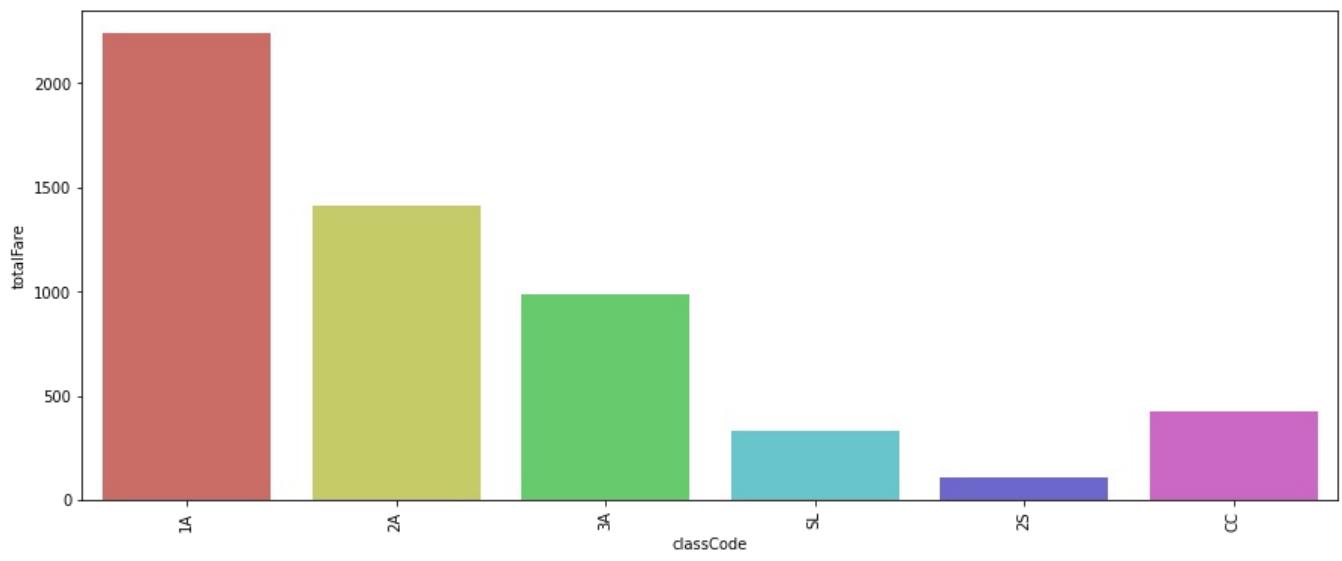
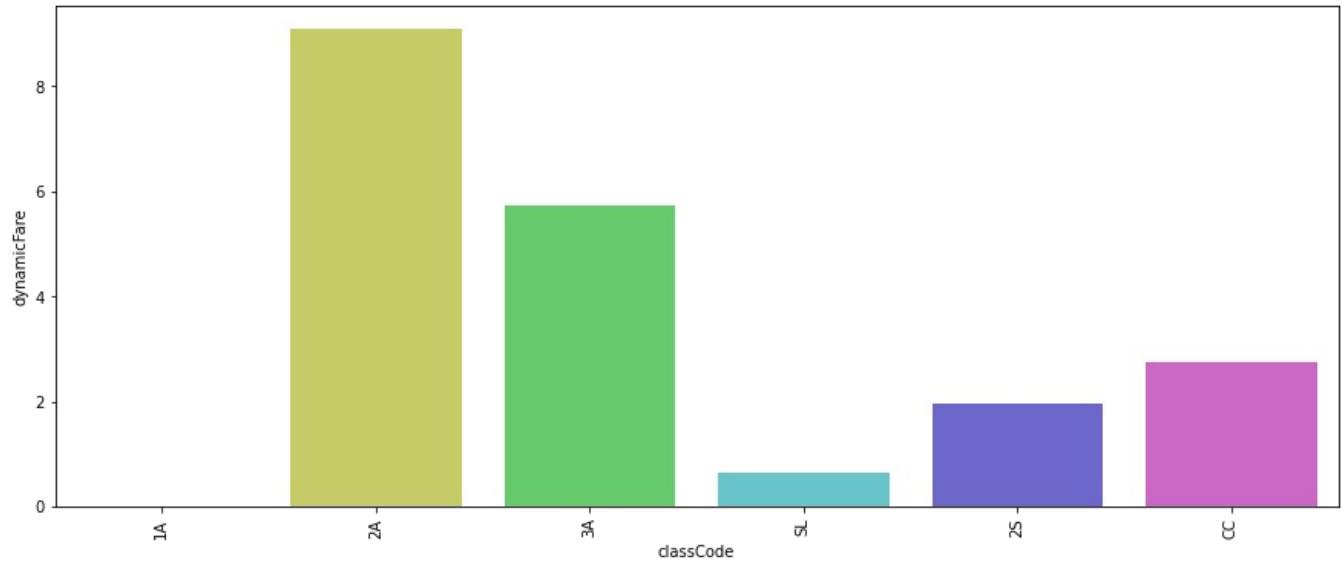


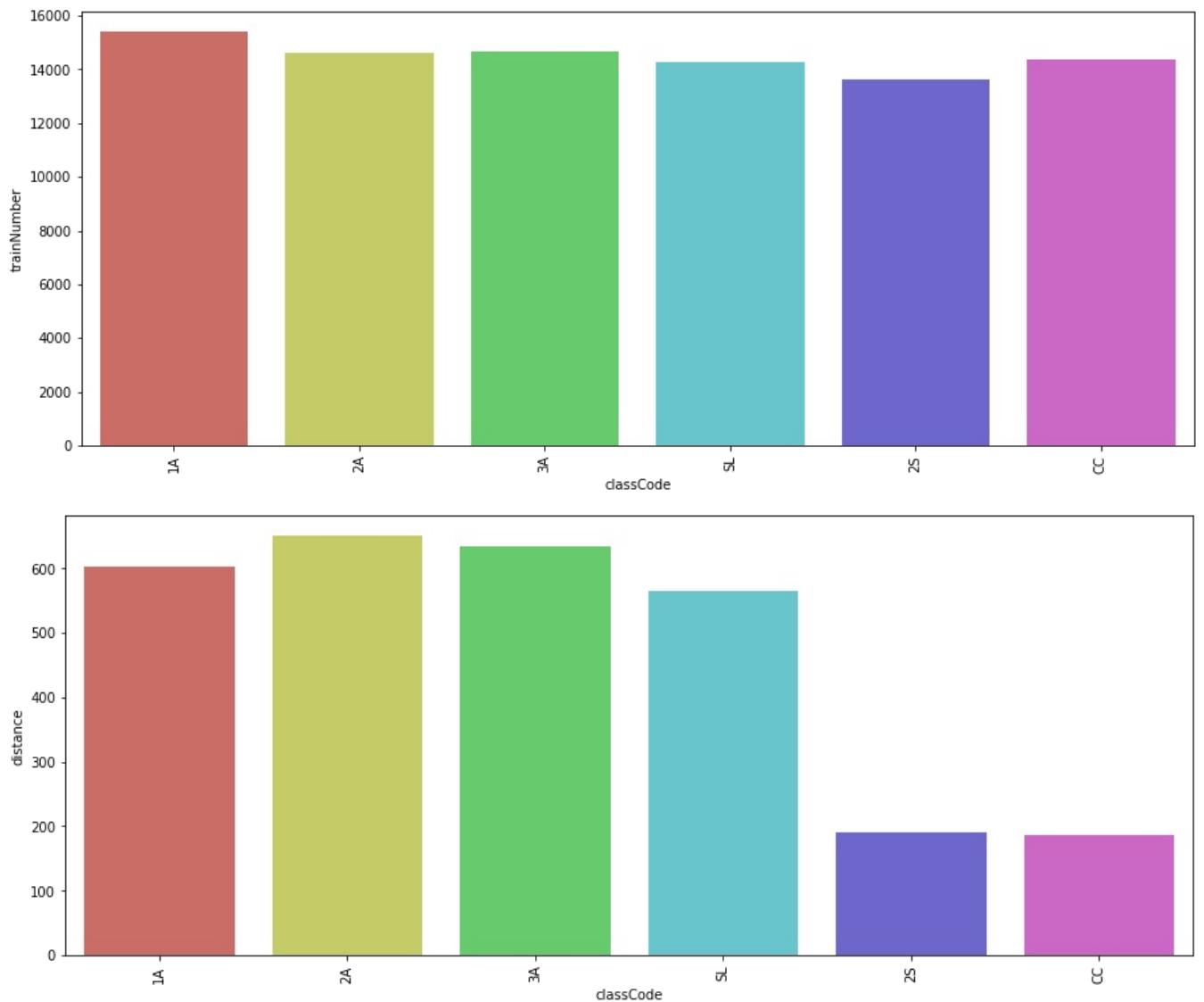


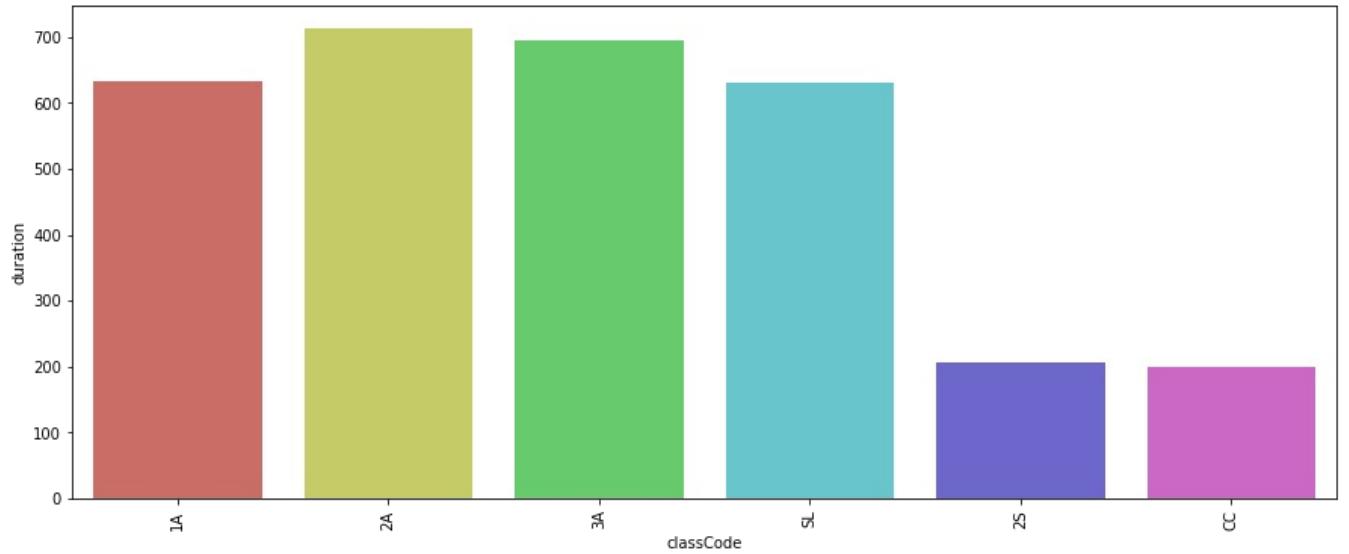
```
In [31]: for i in categorical:
    for j in continuous:
        plt.figure(figsize=(15,6))
        sns.barplot(x = df[i], y = df[j], data = df, ci = None, palette = 'hls')
        plt.xticks(rotation = 90)
        plt.show()
```



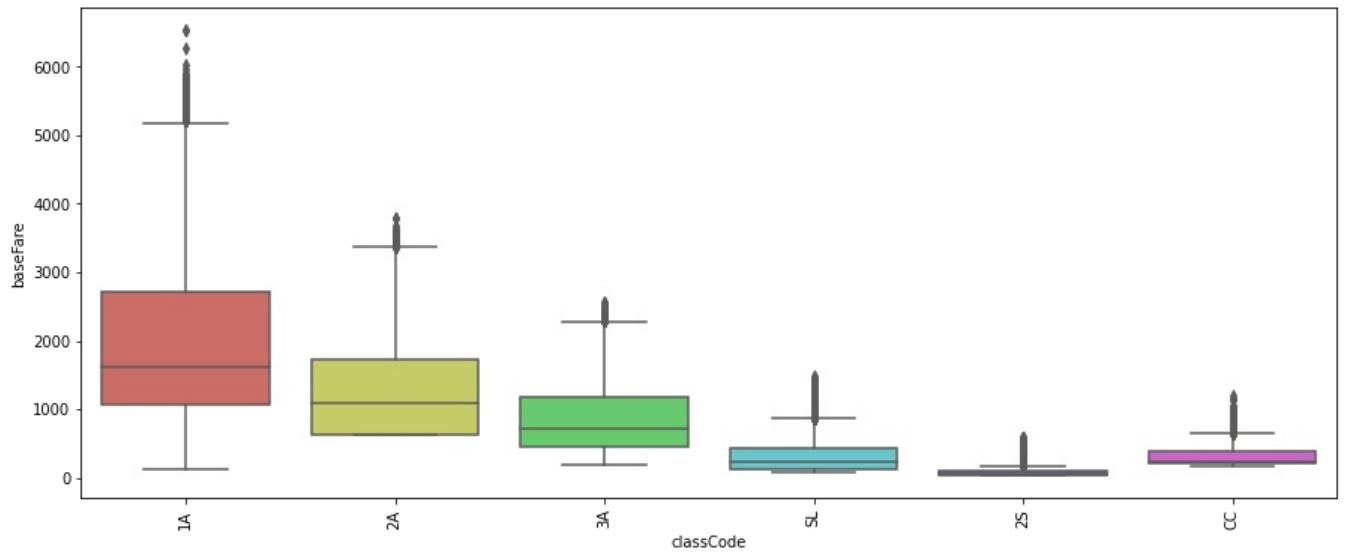


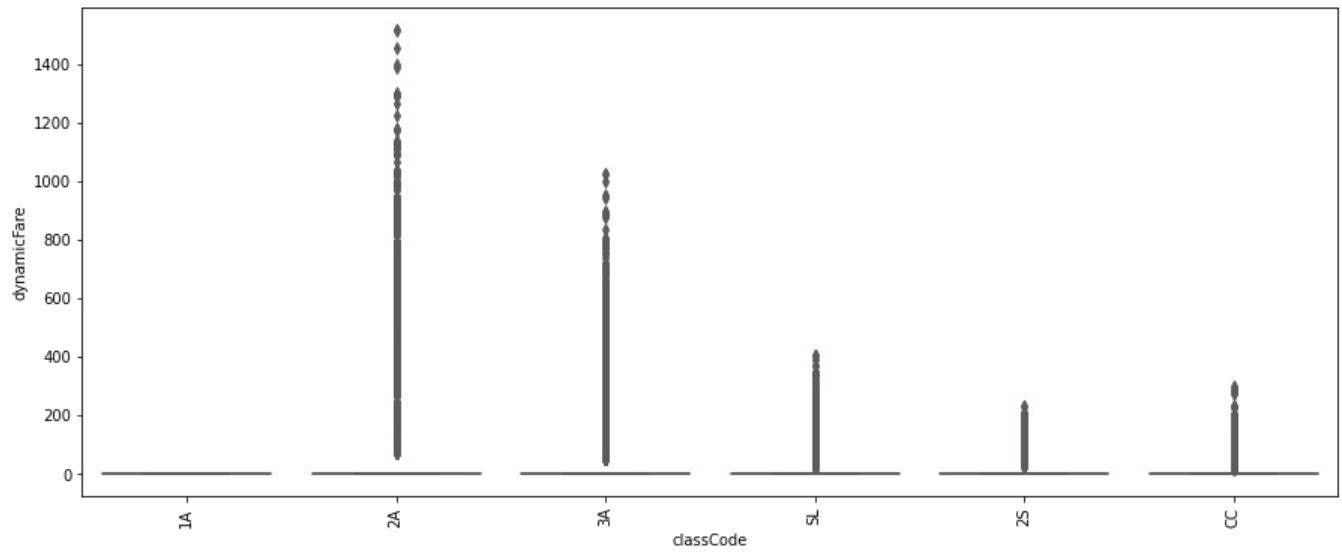
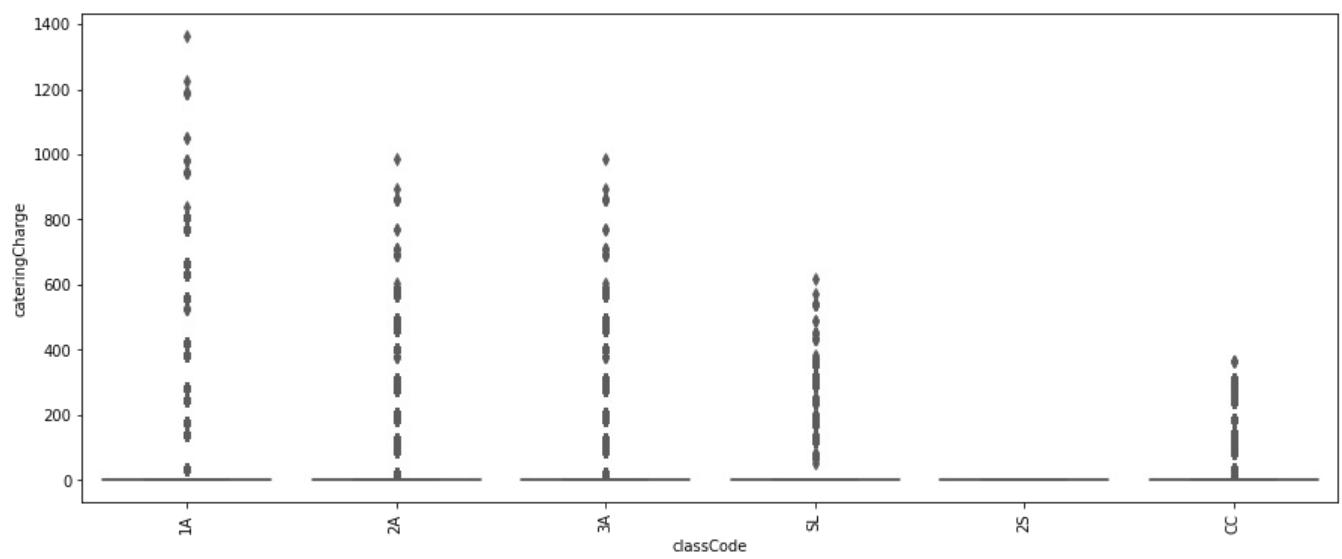
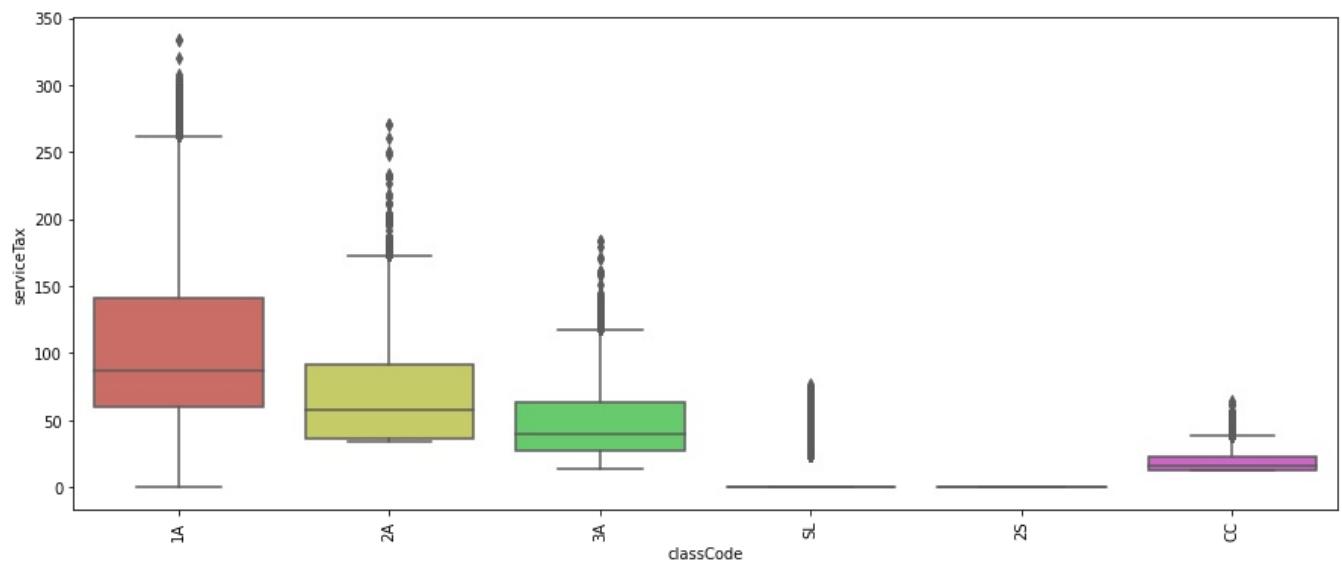


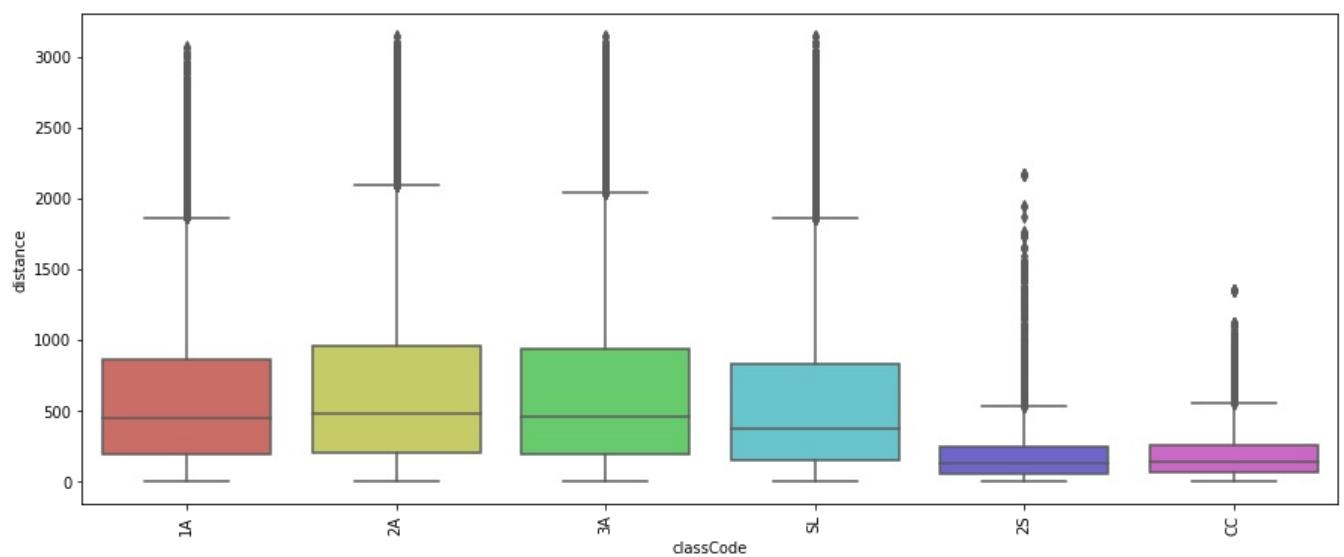
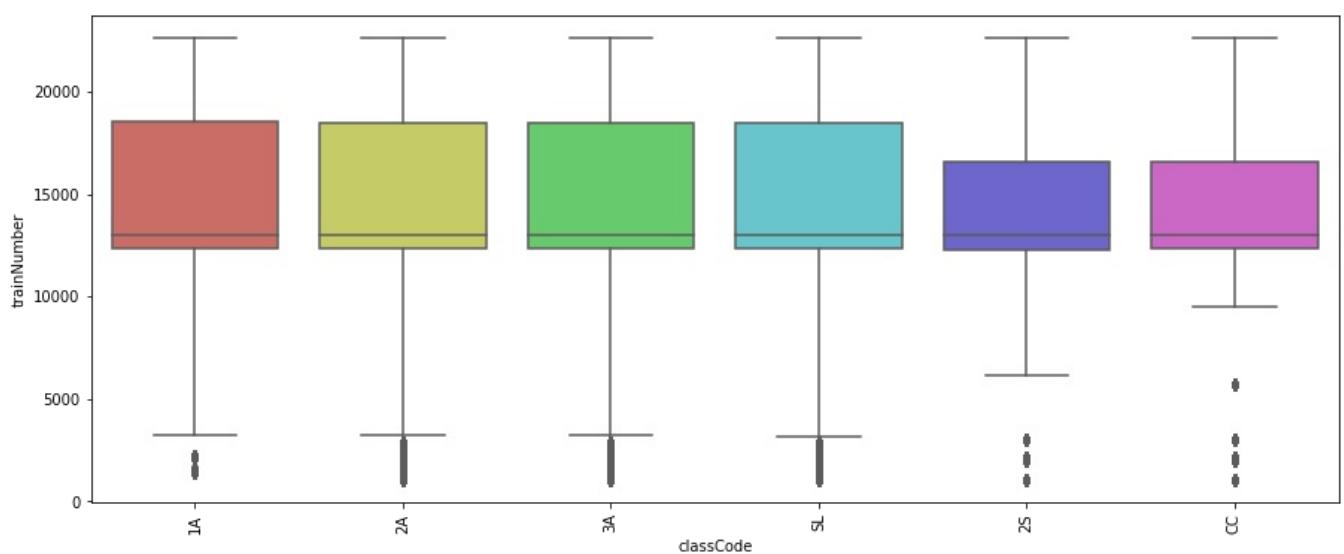
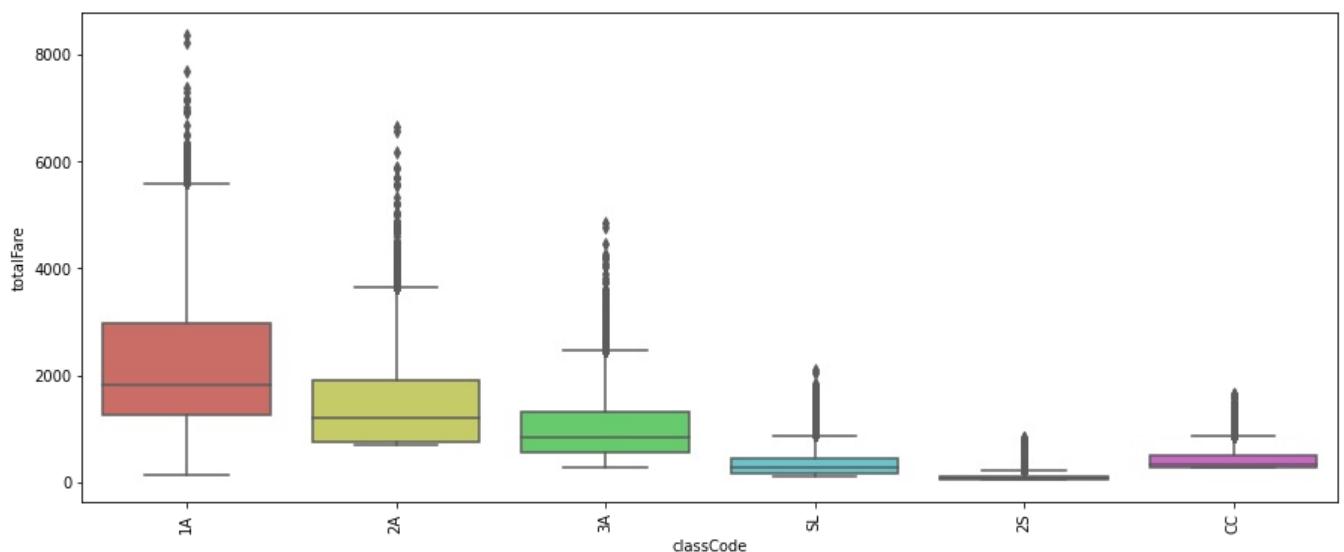


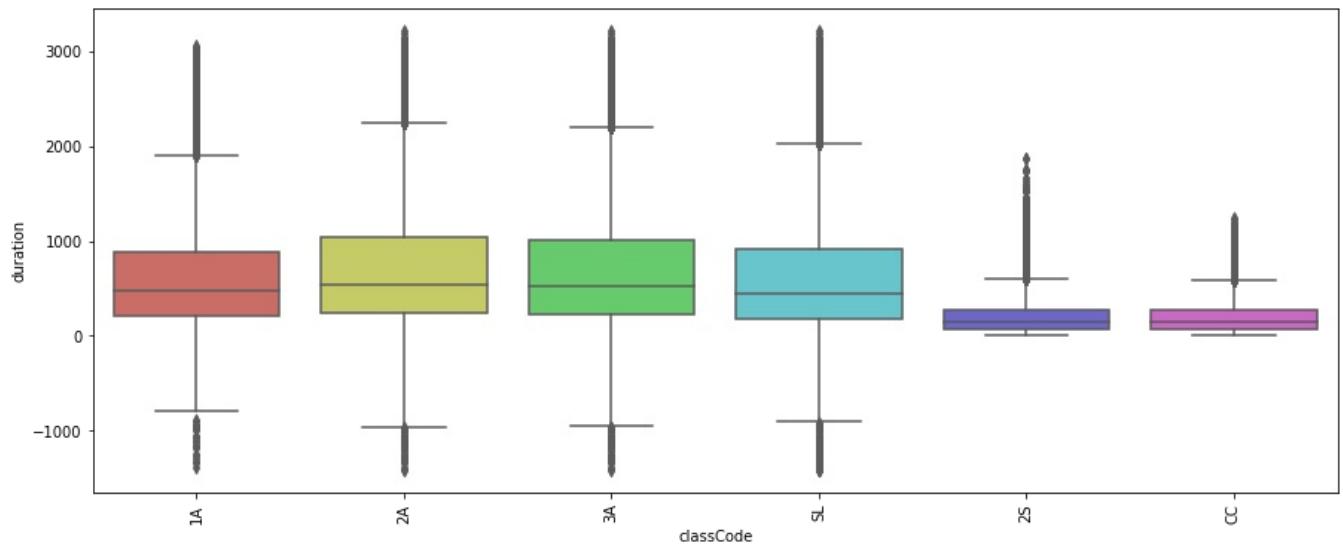


```
In [32]: for i in categorical:  
    for j in continuous:  
        plt.figure(figsize=(15,6))  
        sns.boxplot(x = df[i], y = df[j], data = df, palette = 'hls')  
        plt.xticks(rotation = 90)  
        plt.show()
```

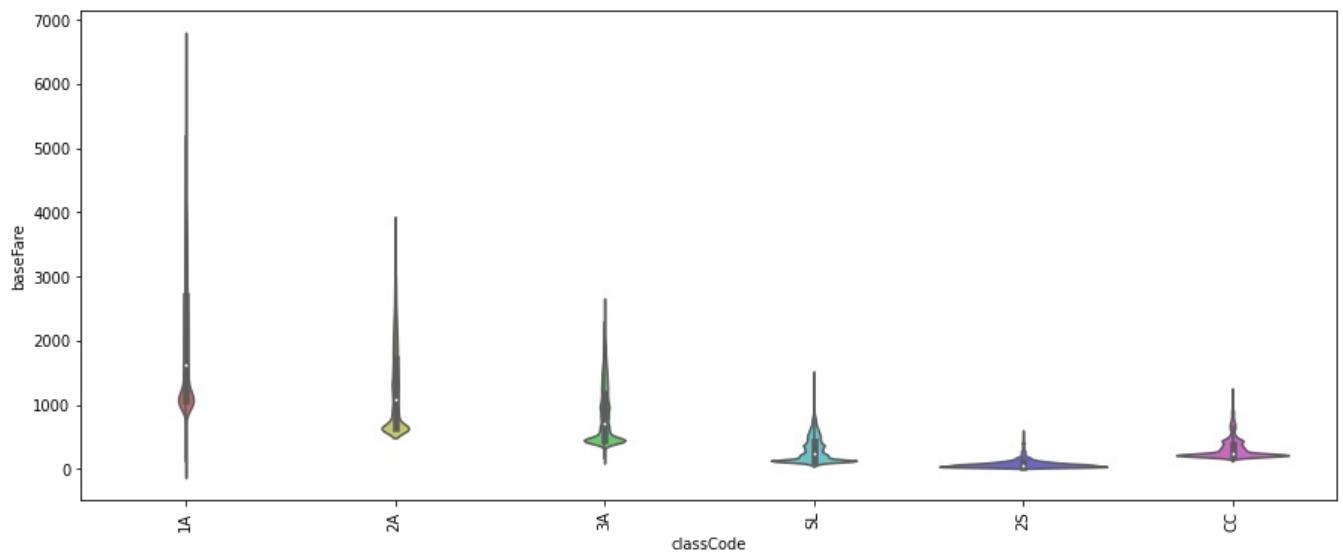


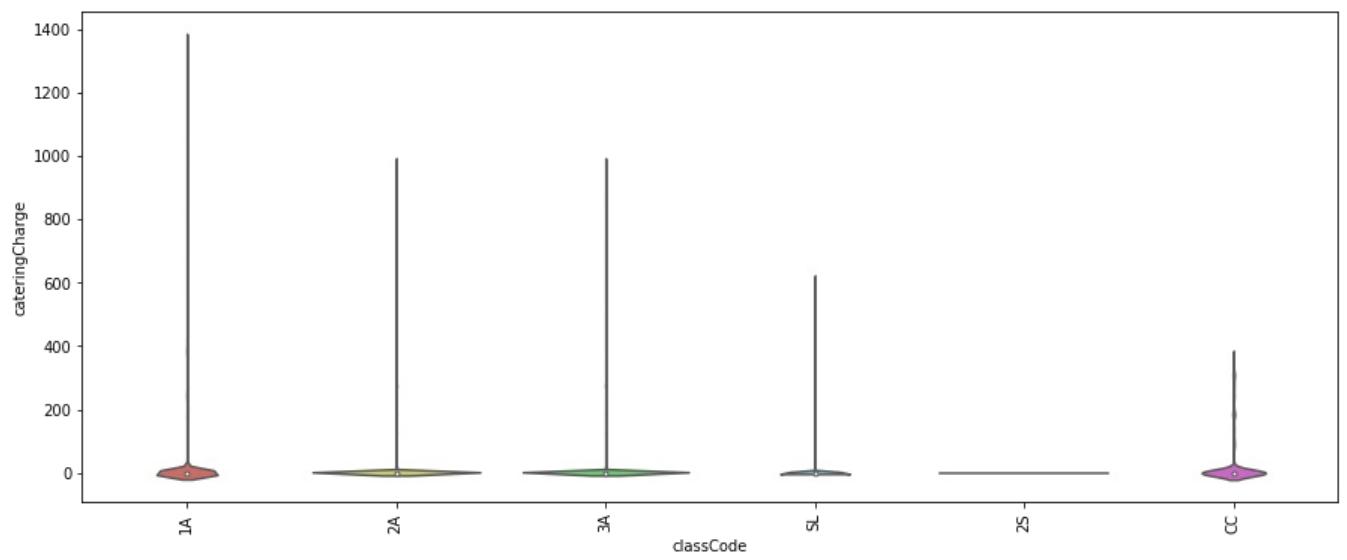
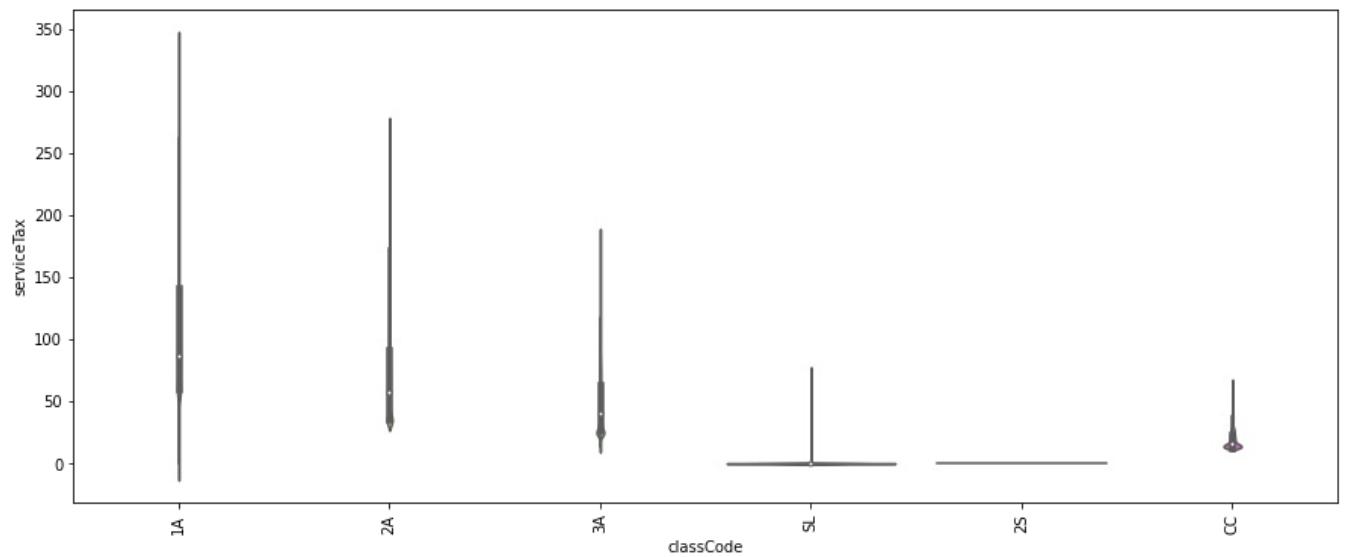


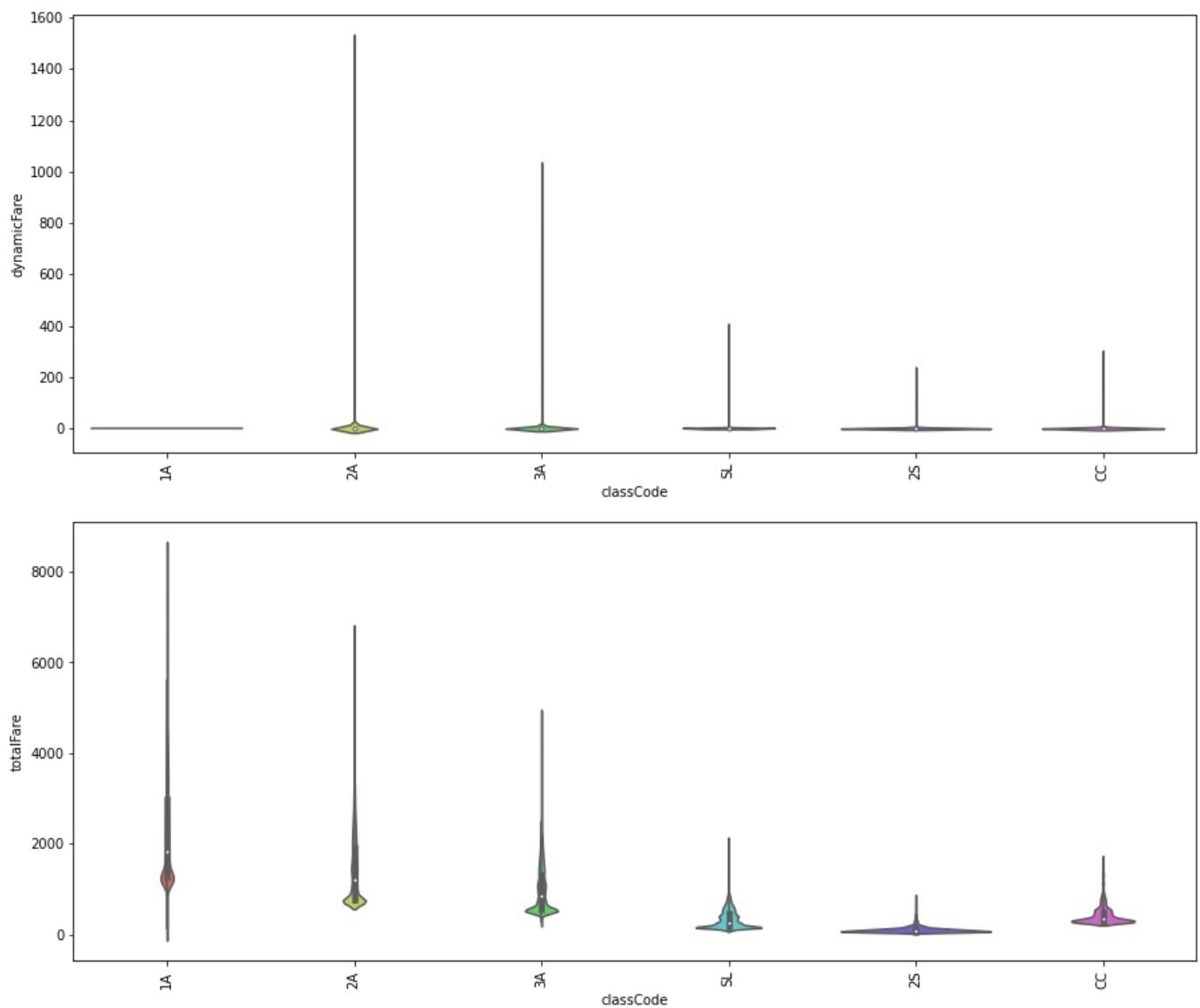


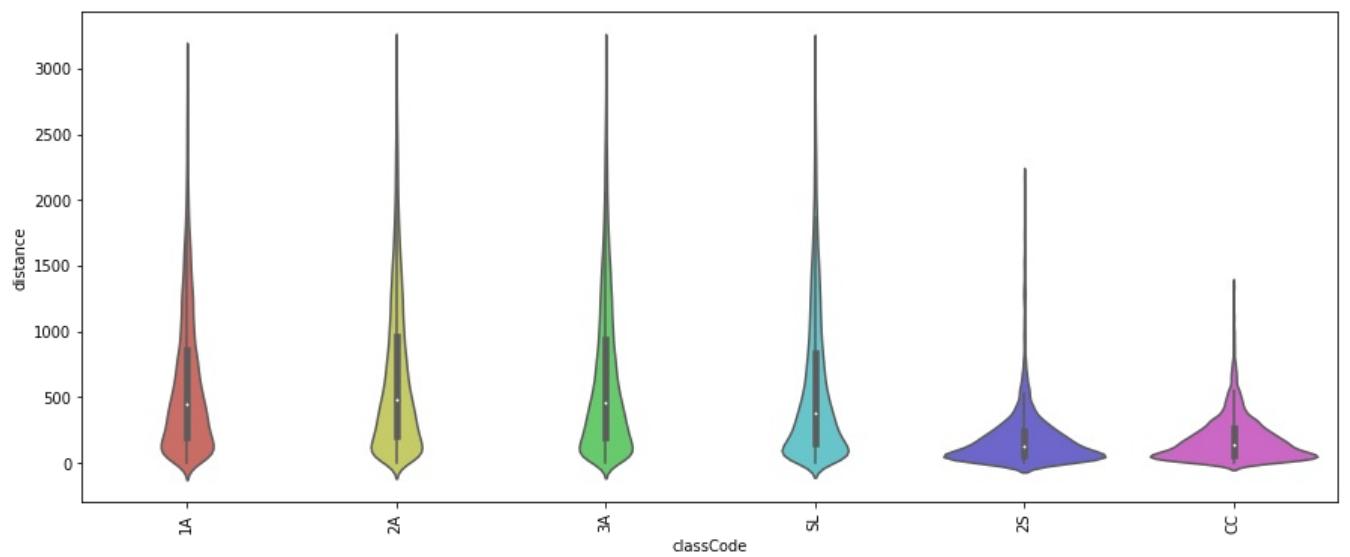
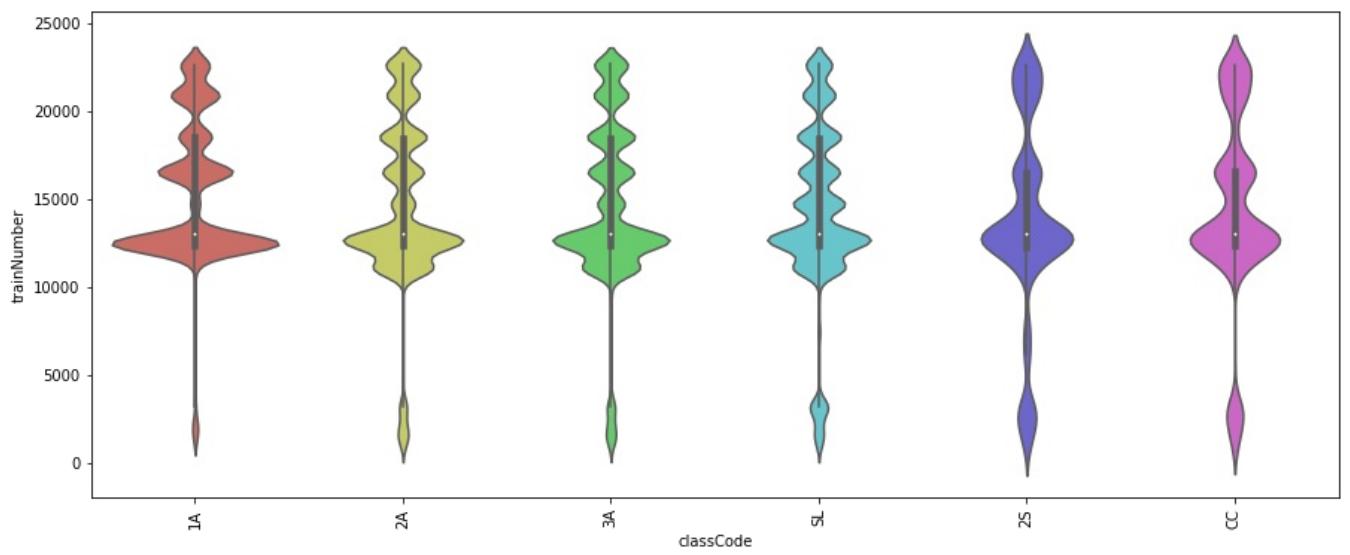


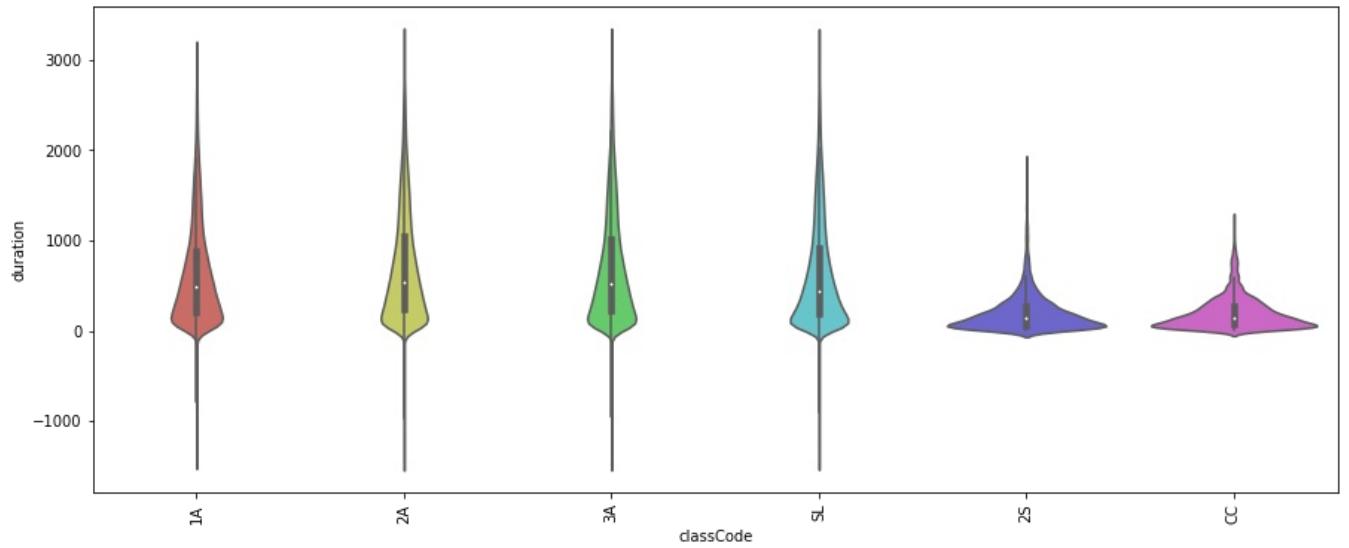
```
In [33]: for i in categorical:
    for j in continuous:
        plt.figure(figsize=(15,6))
        sns.violinplot(x = df[i], y = df[j], data = df, palette = 'hls')
        plt.xticks(rotation = 90)
        plt.show()
```



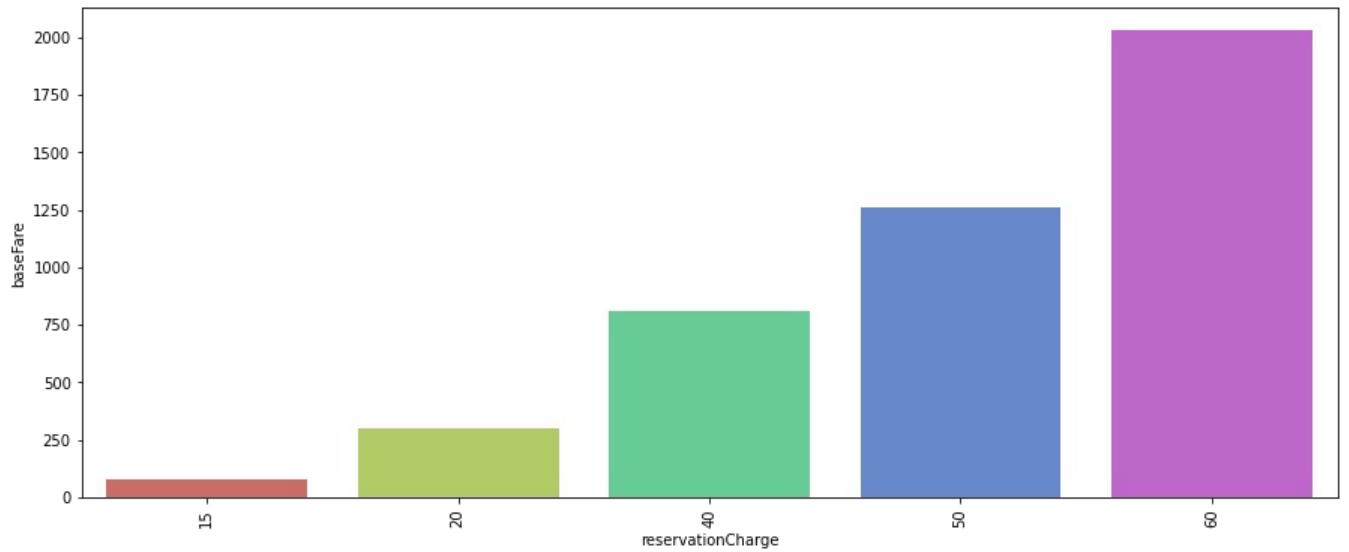


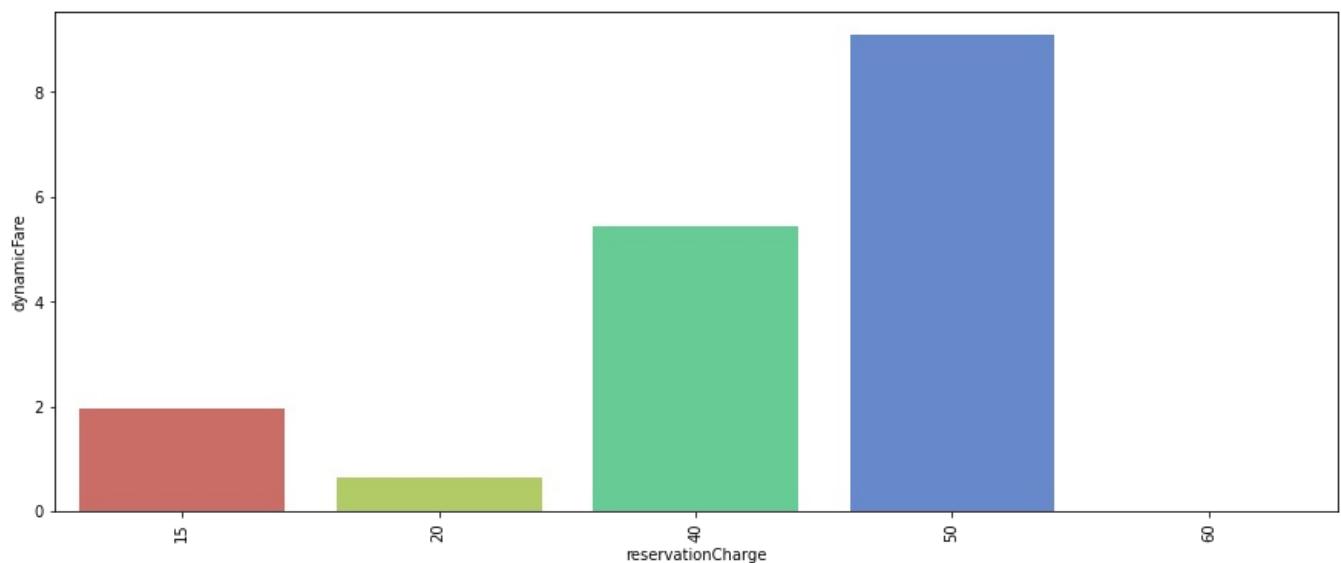
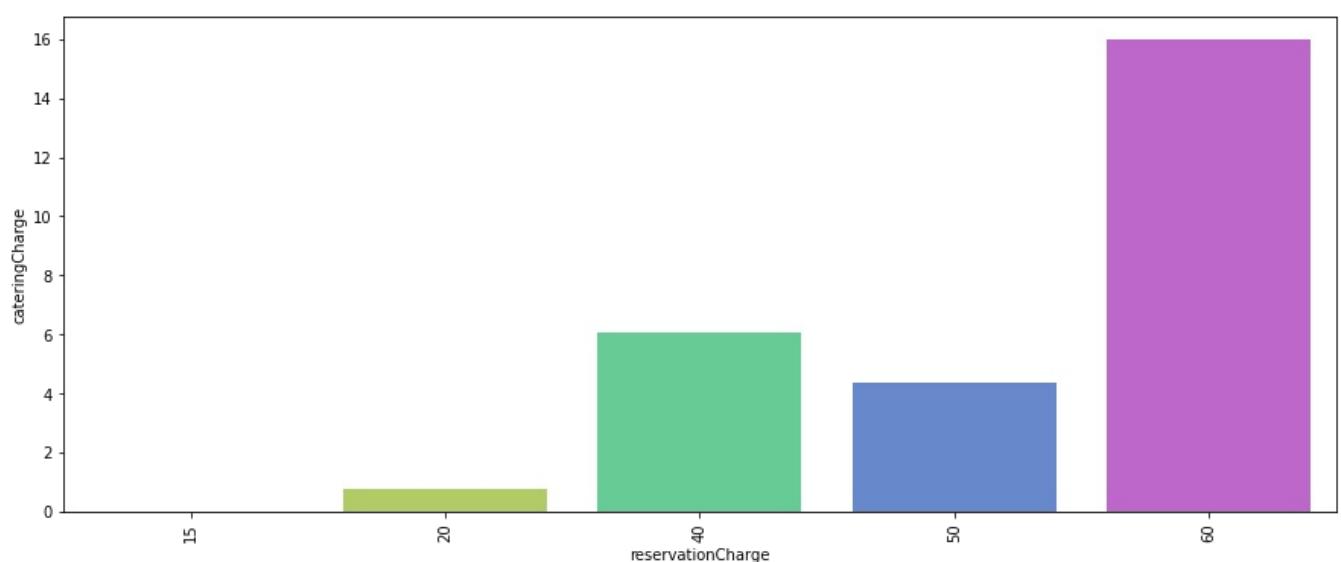
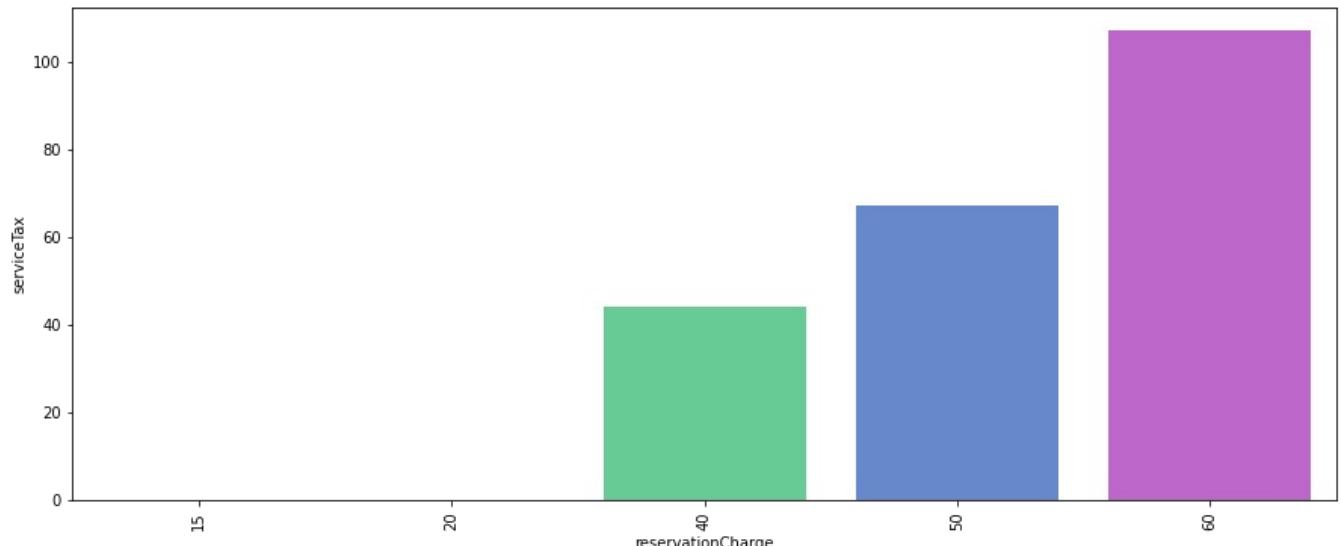


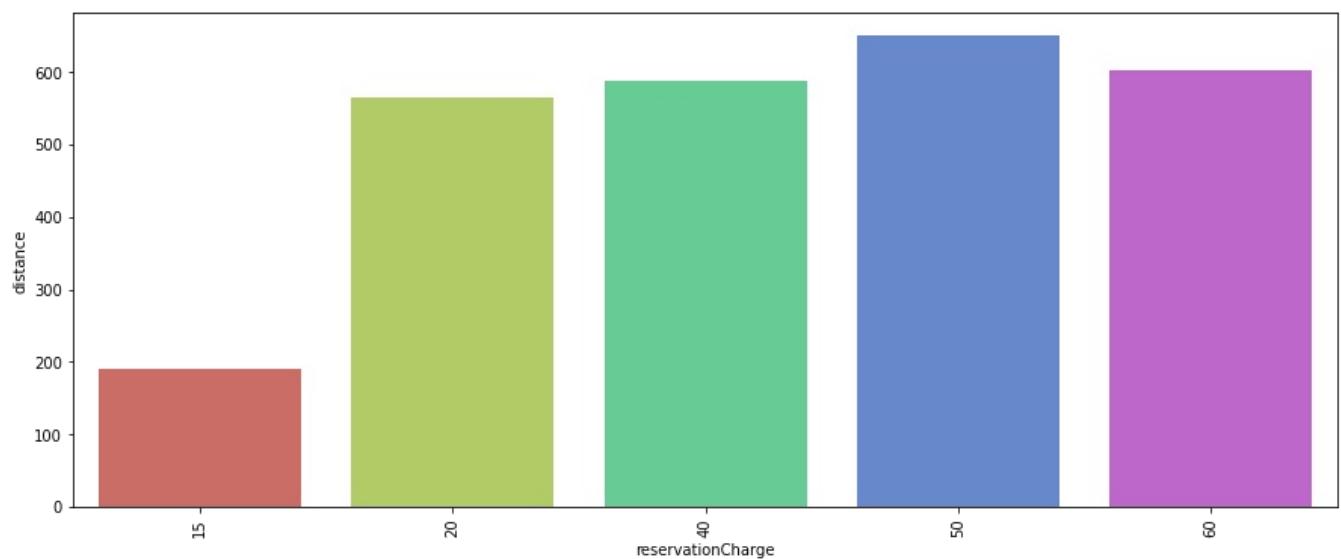
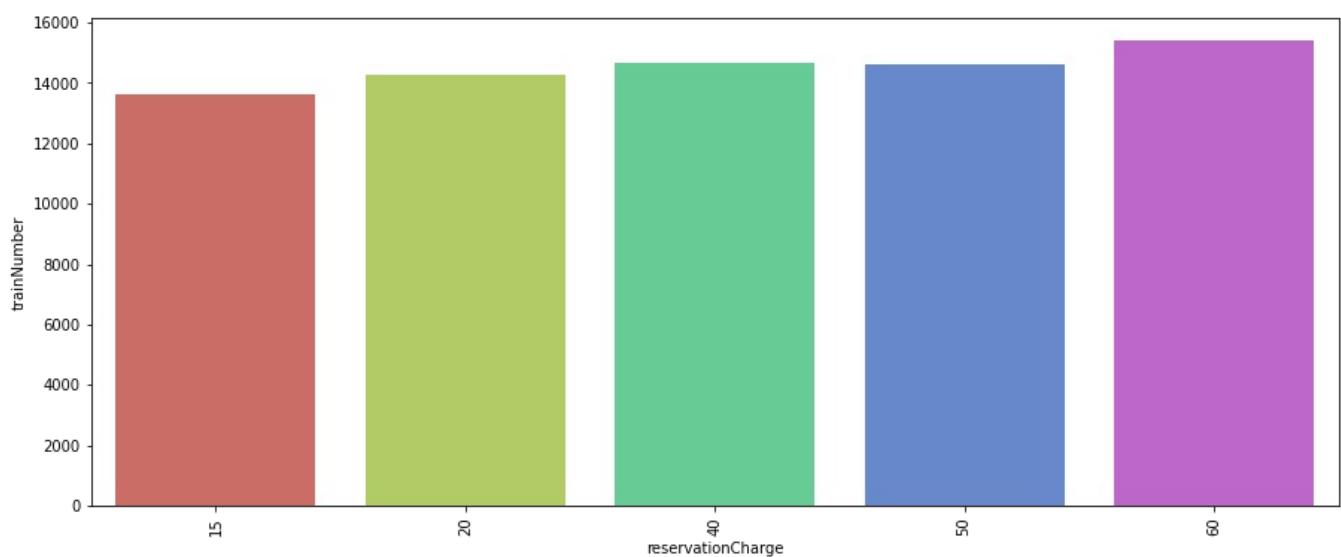
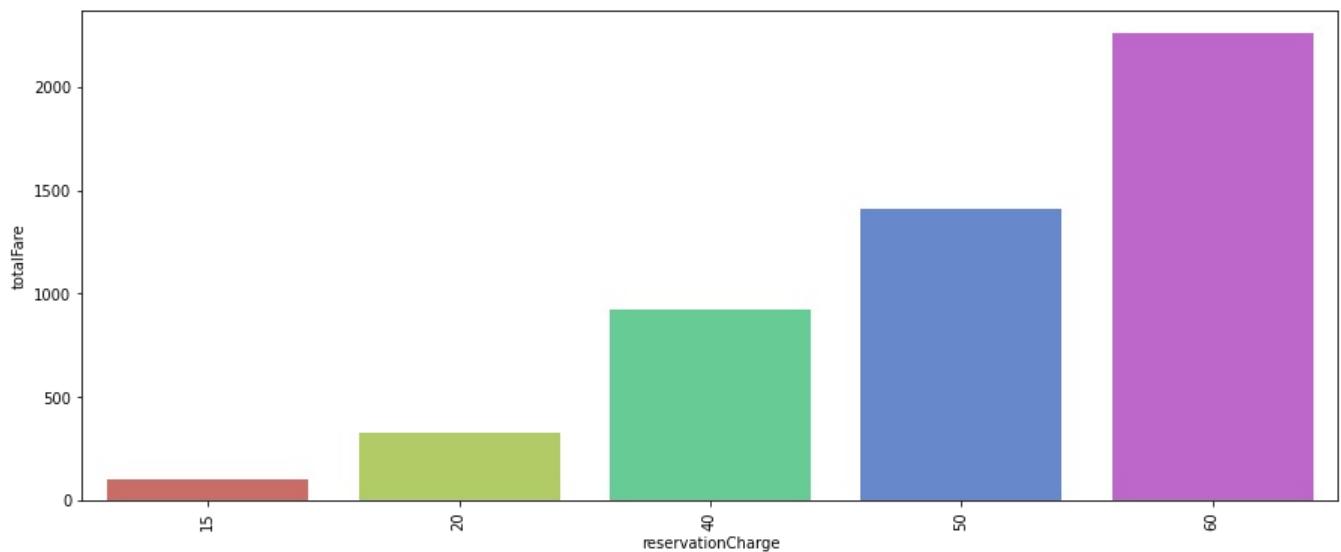


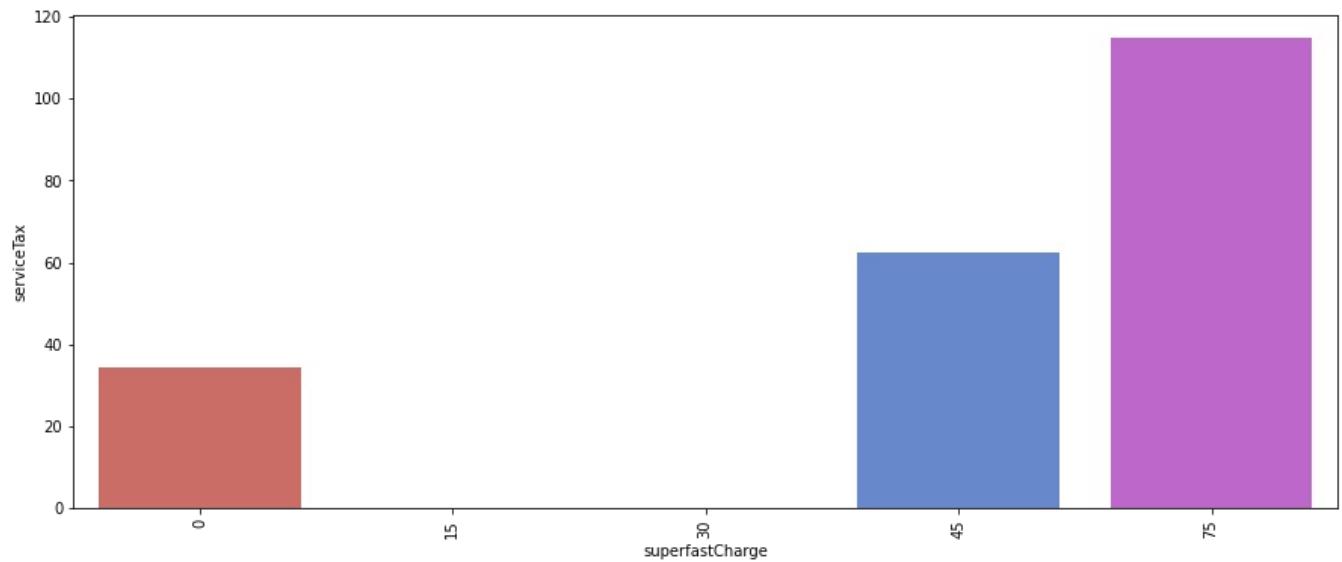
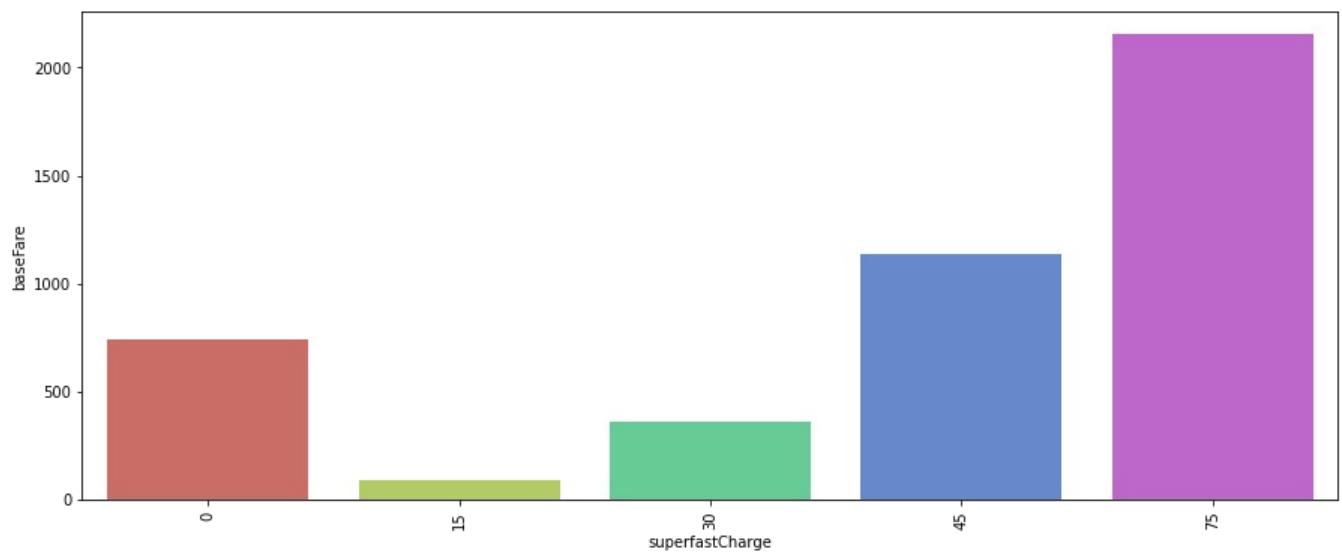
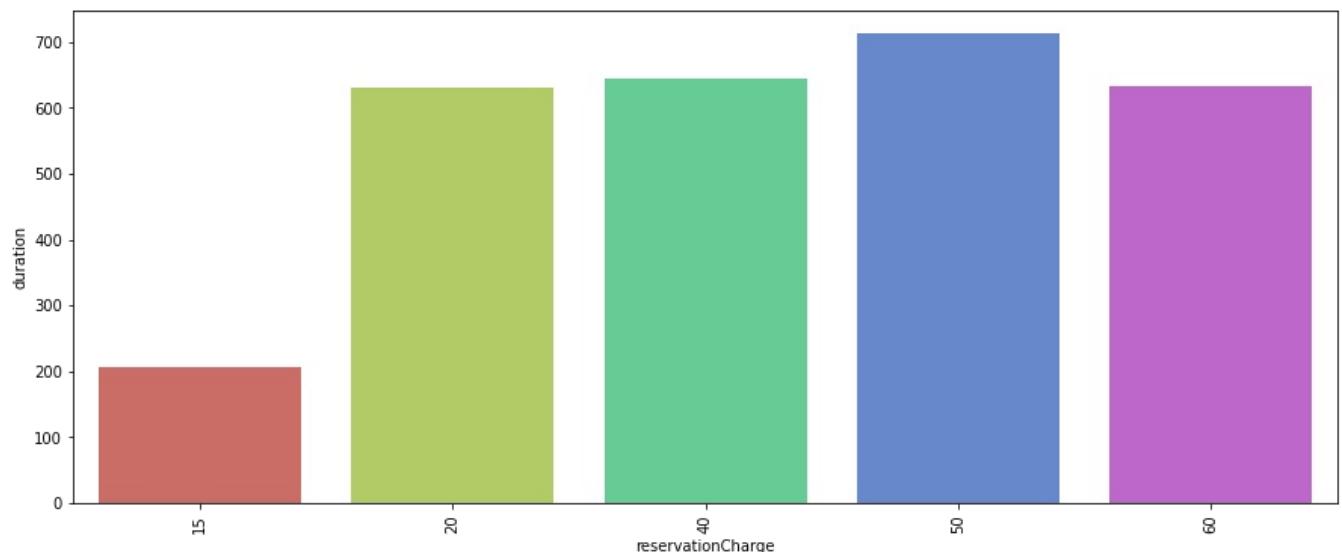


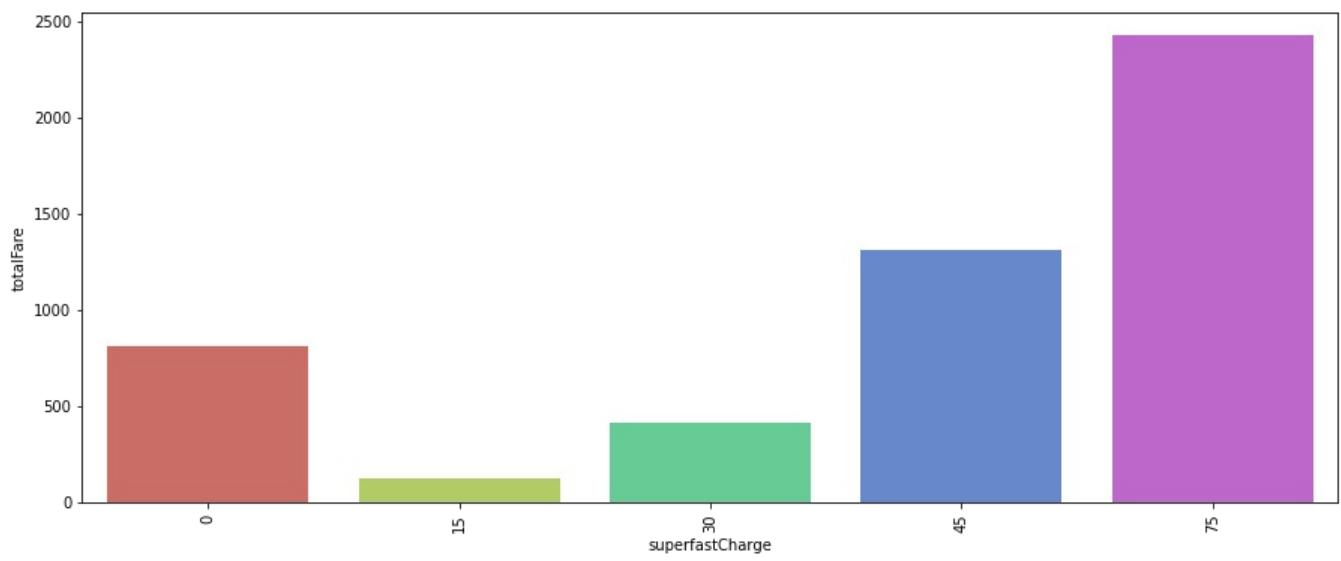
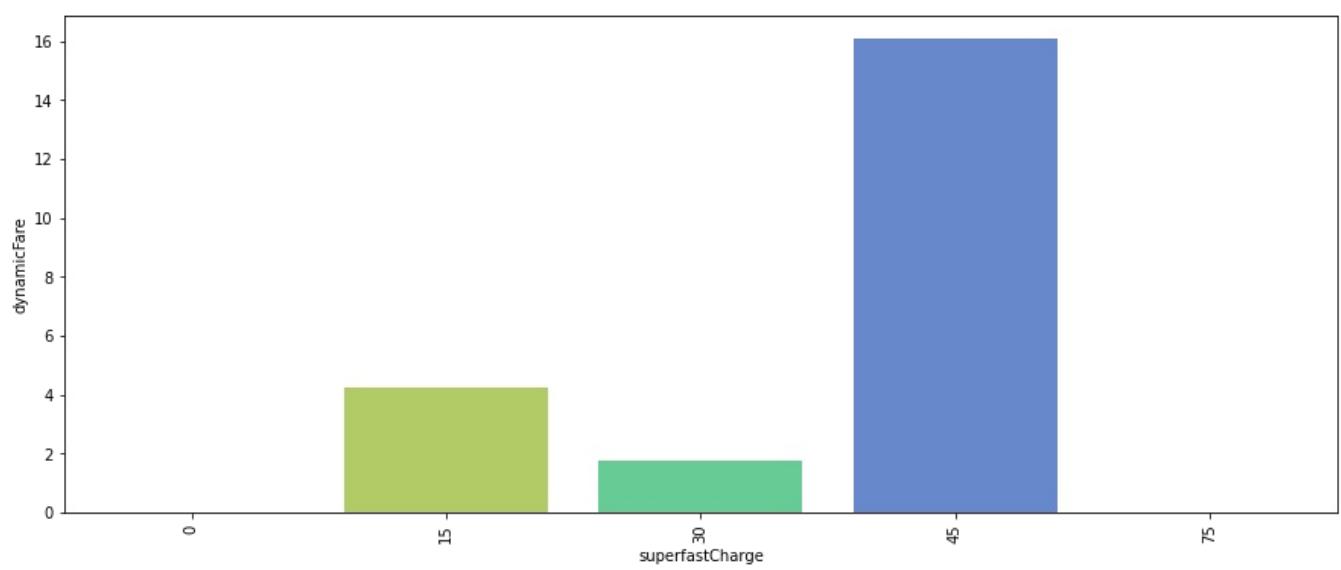
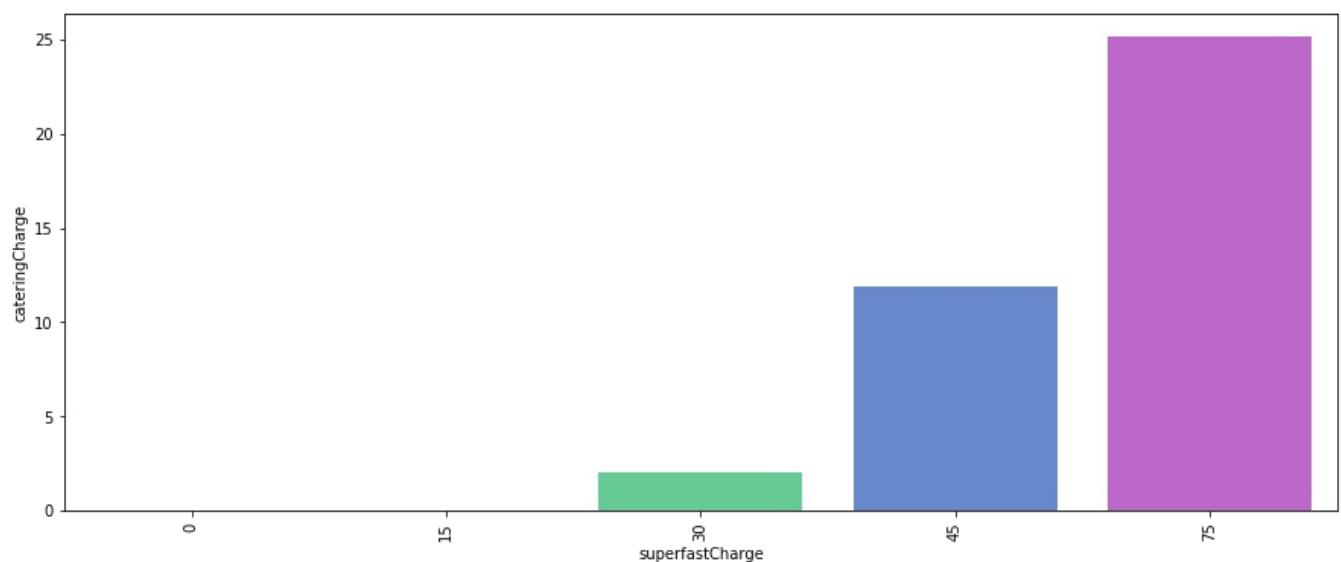
```
In [34]: for i in discrete:
    for j in continuous:
        plt.figure(figsize=(15,6))
        sns.barplot(x = df[i], y = df[j], data = df, ci = None, palette = 'hls')
        plt.xticks(rotation = 90)
        plt.show()
```

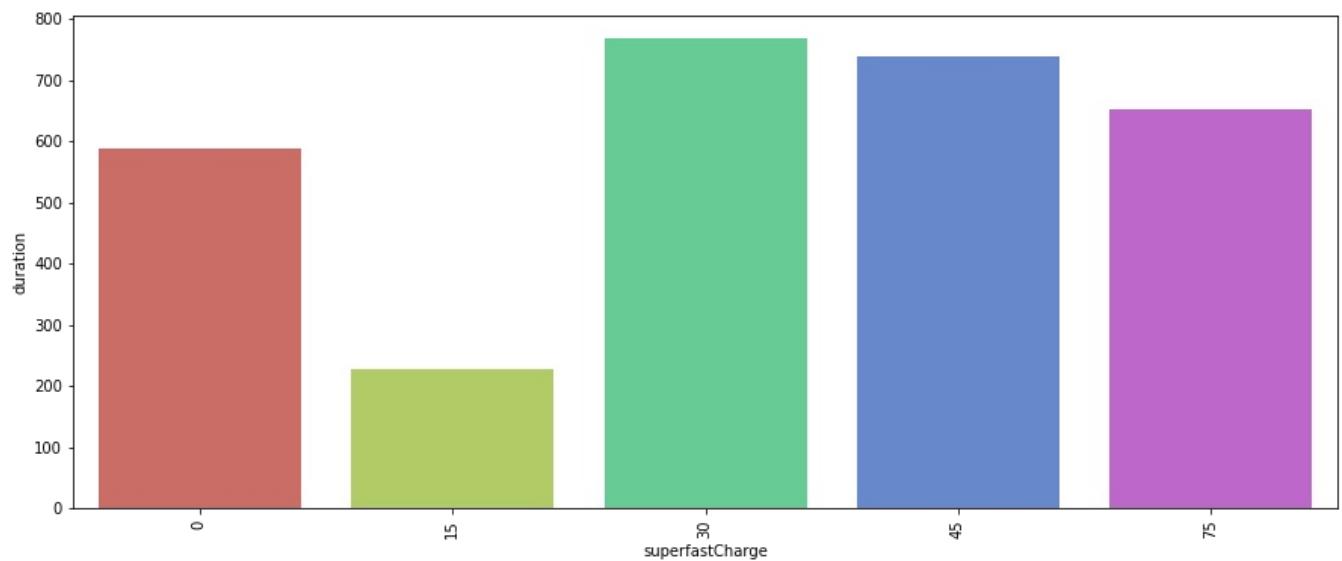
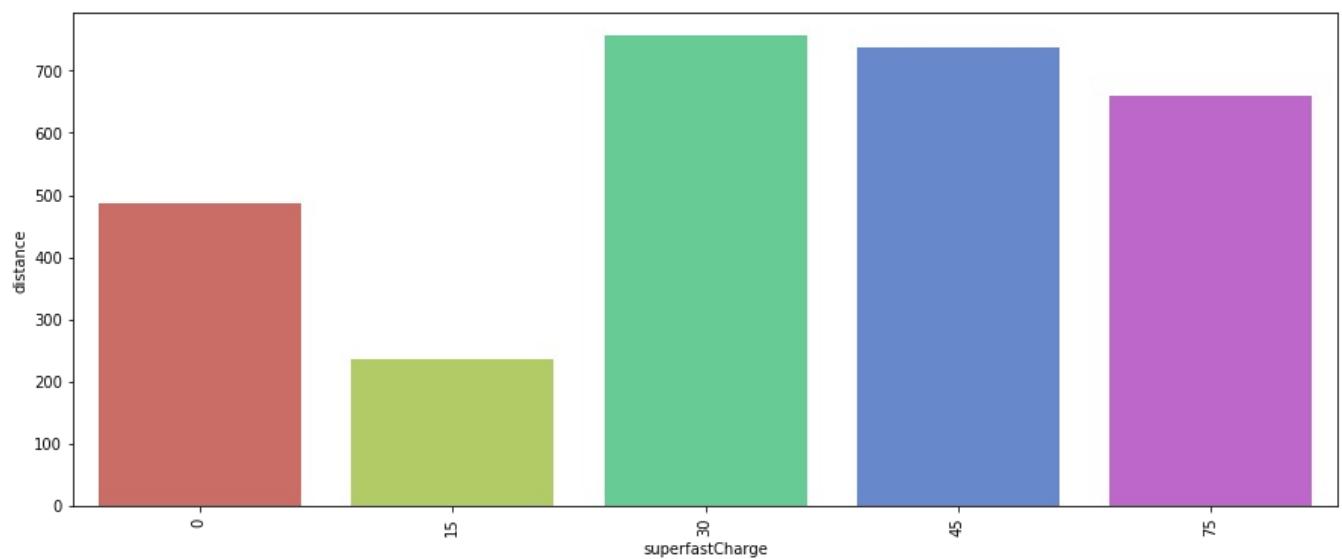
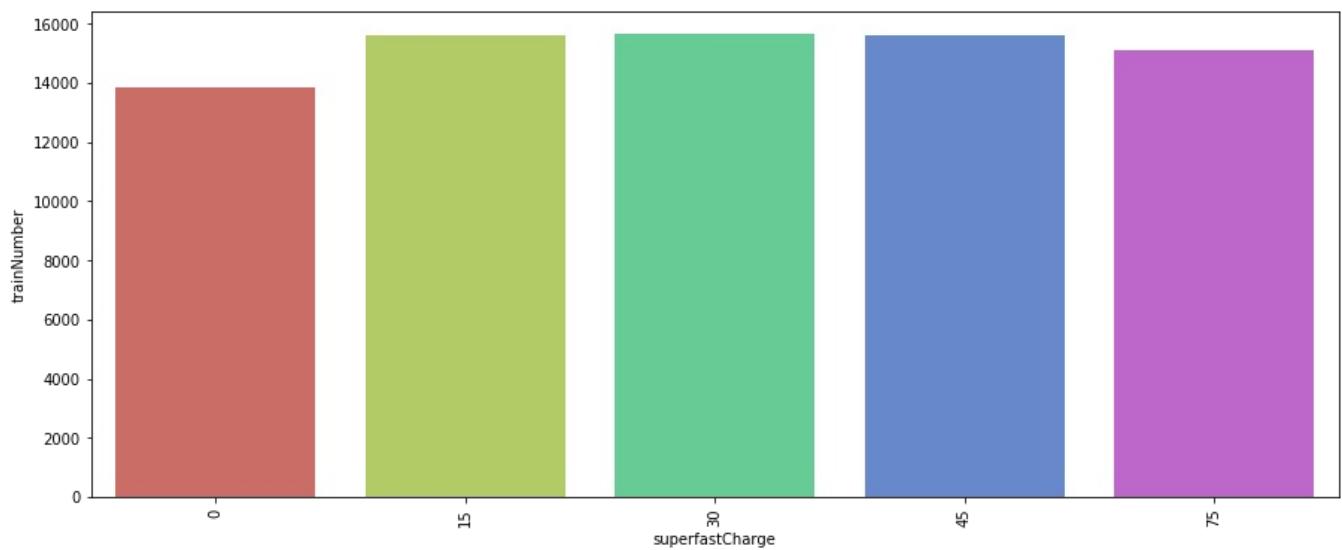


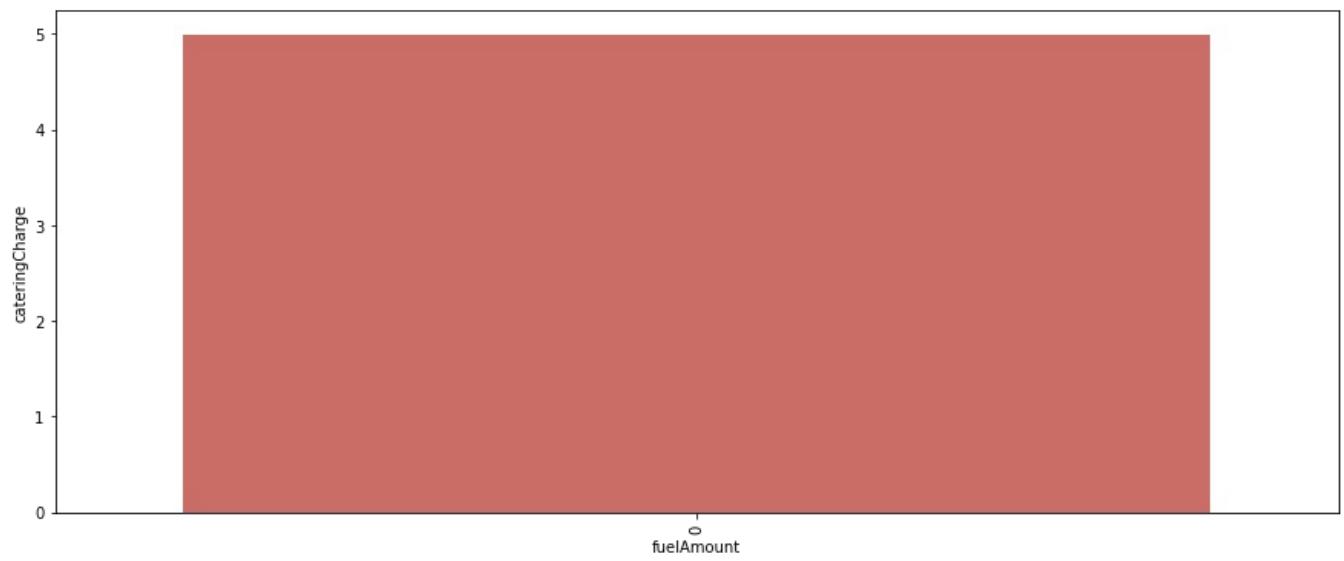
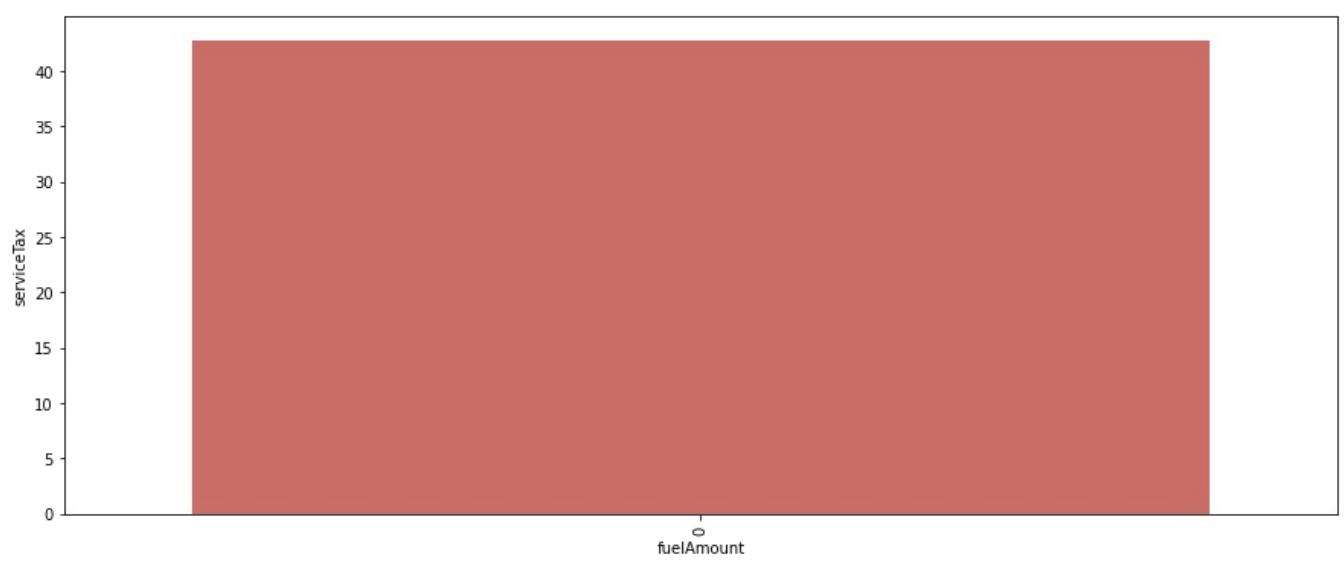
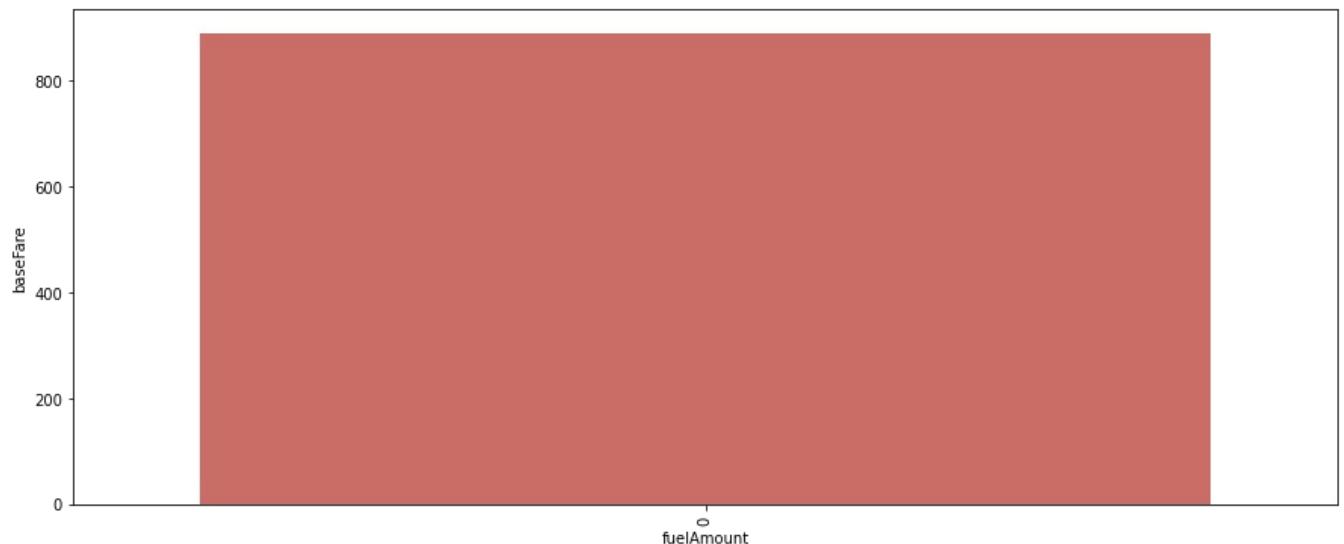


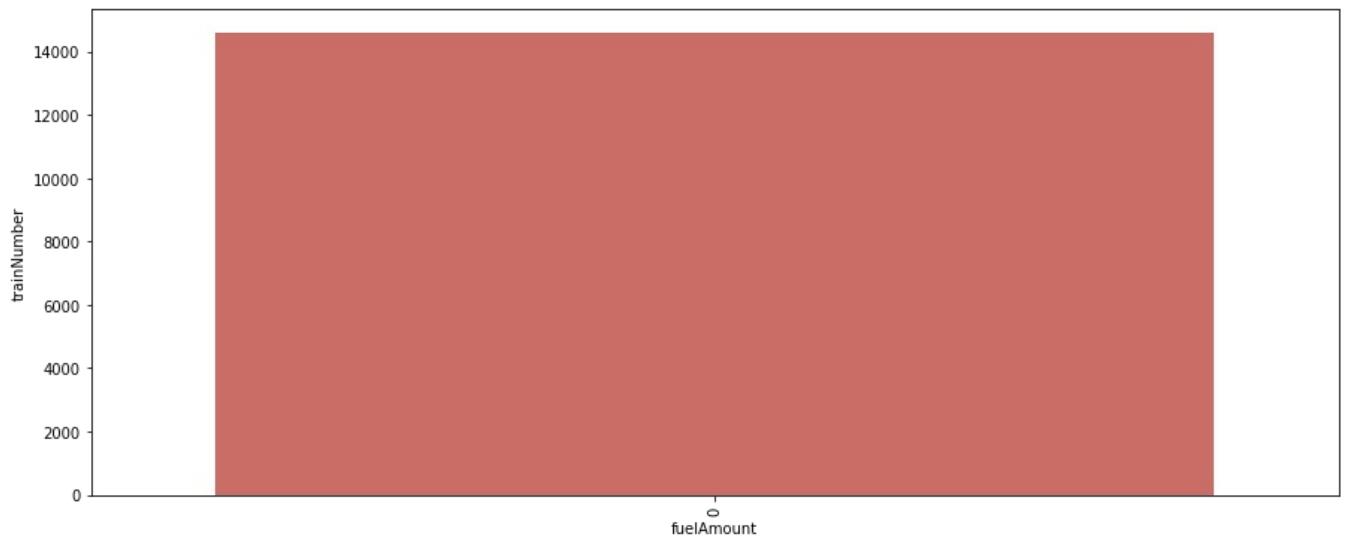
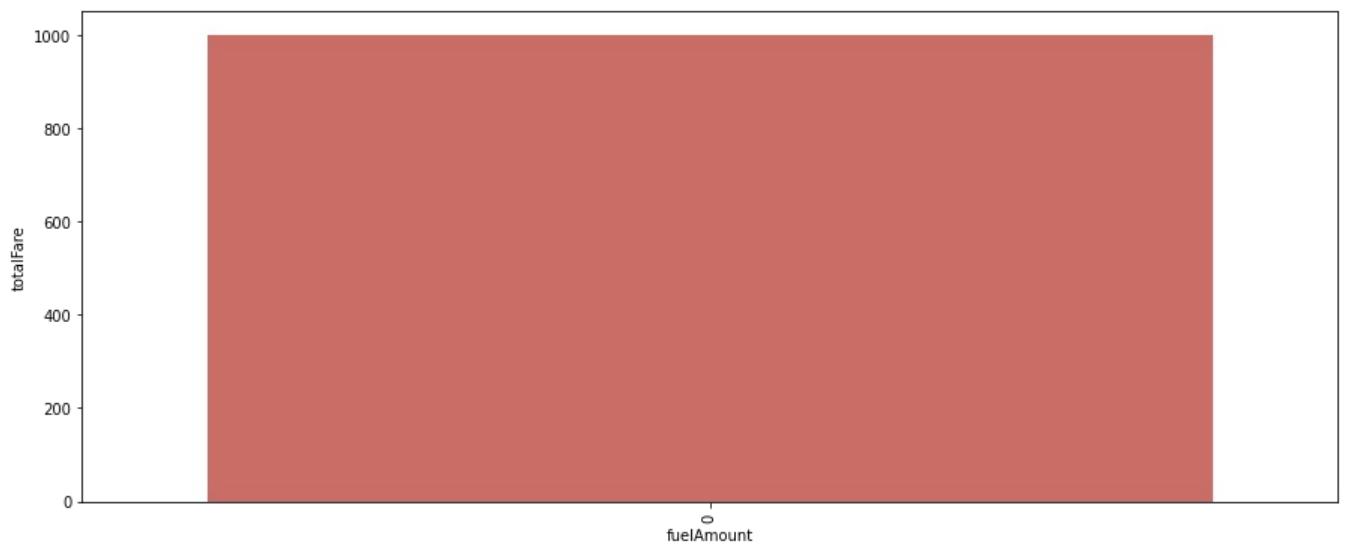
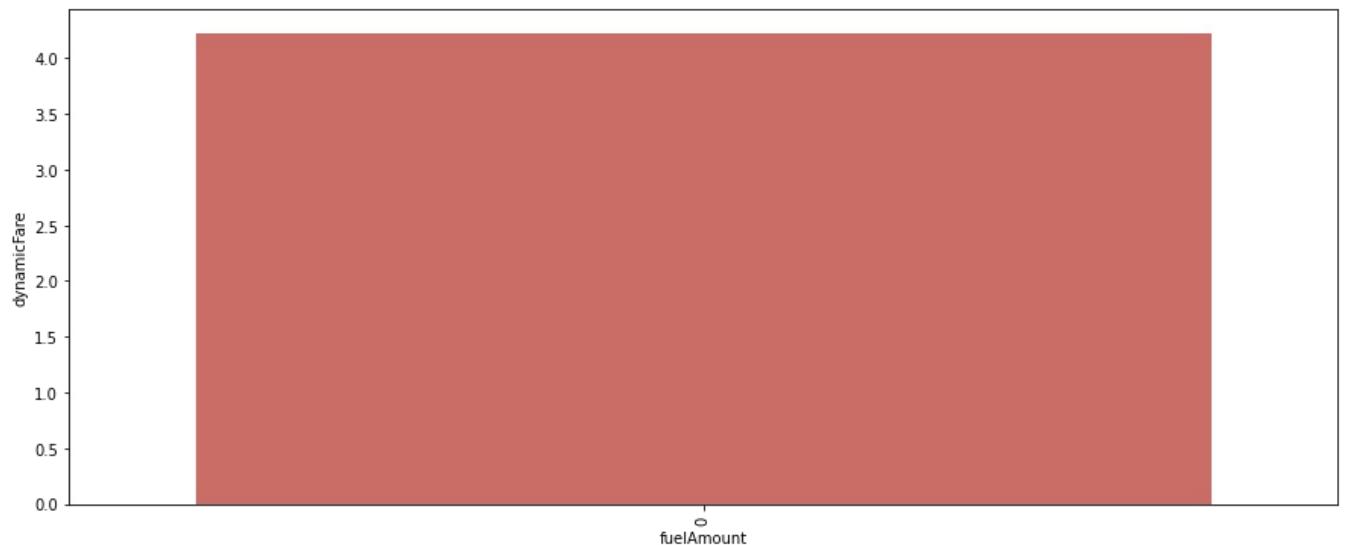


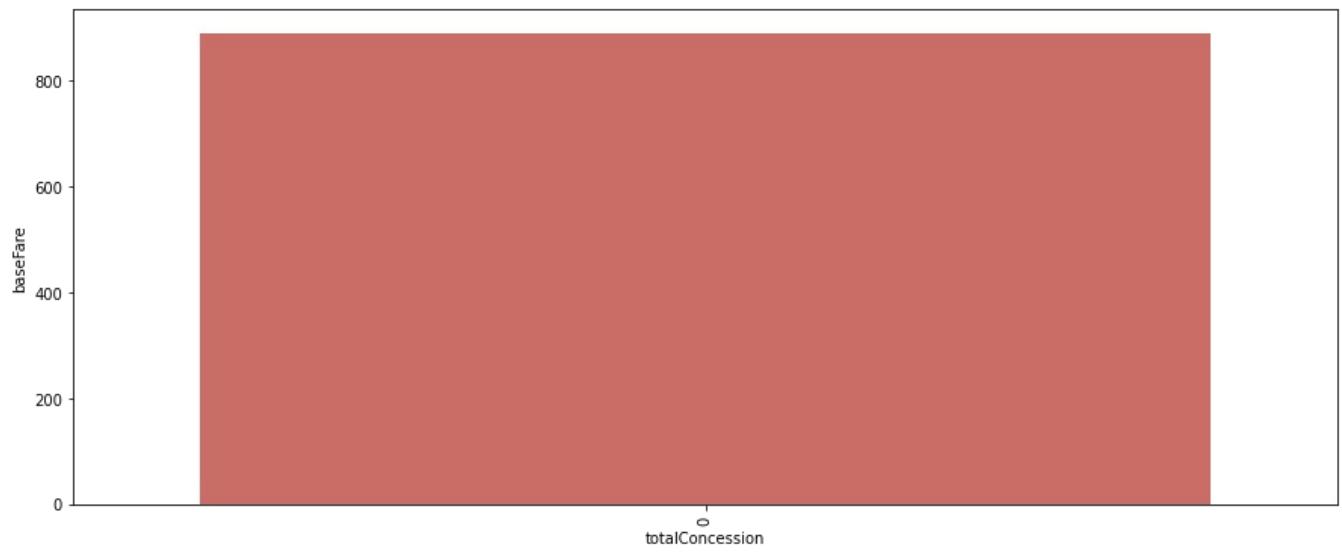
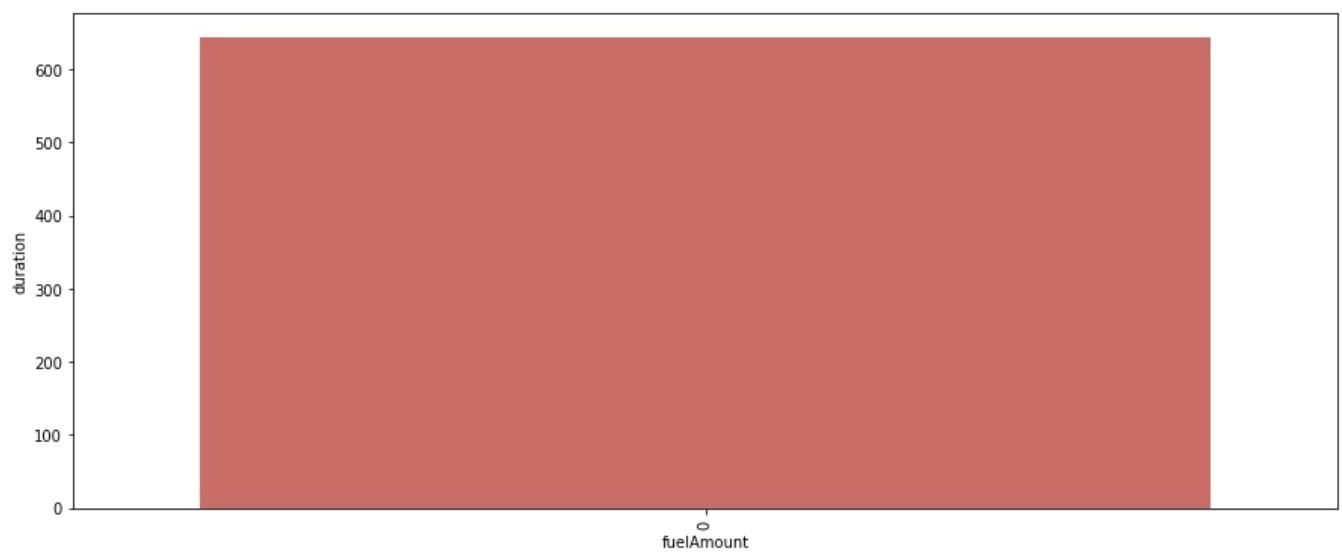
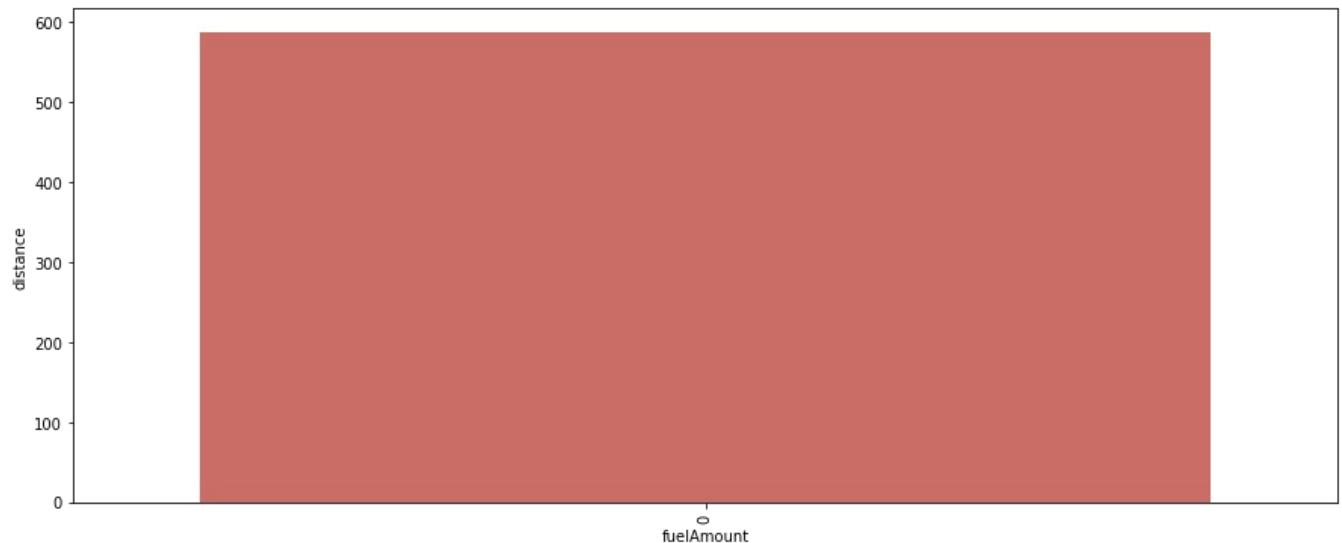


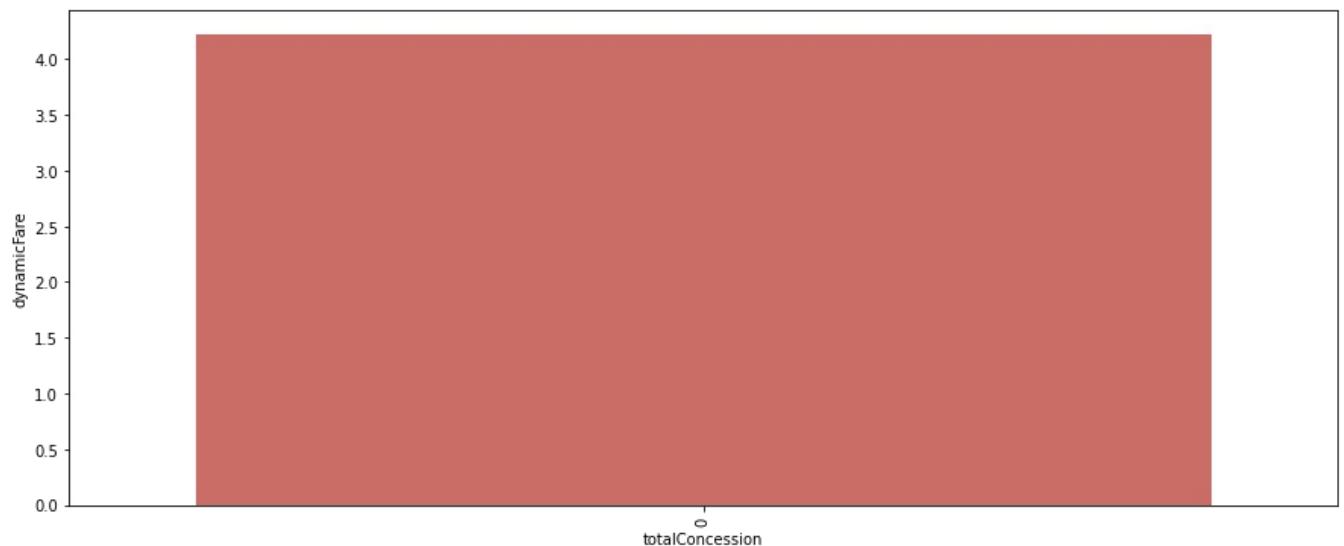
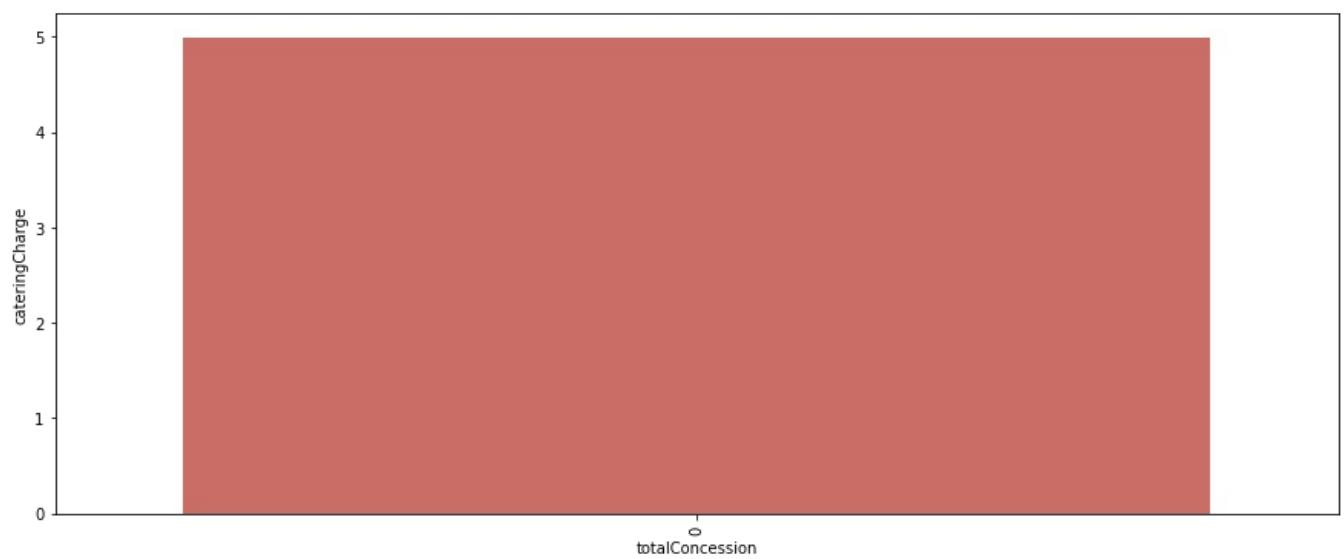
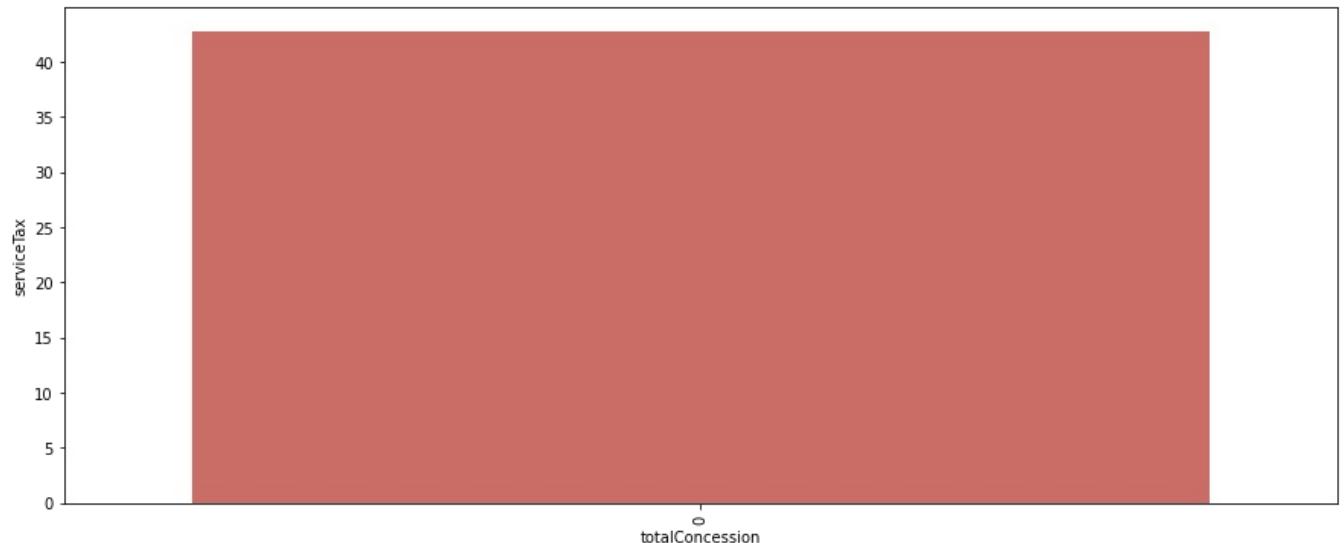


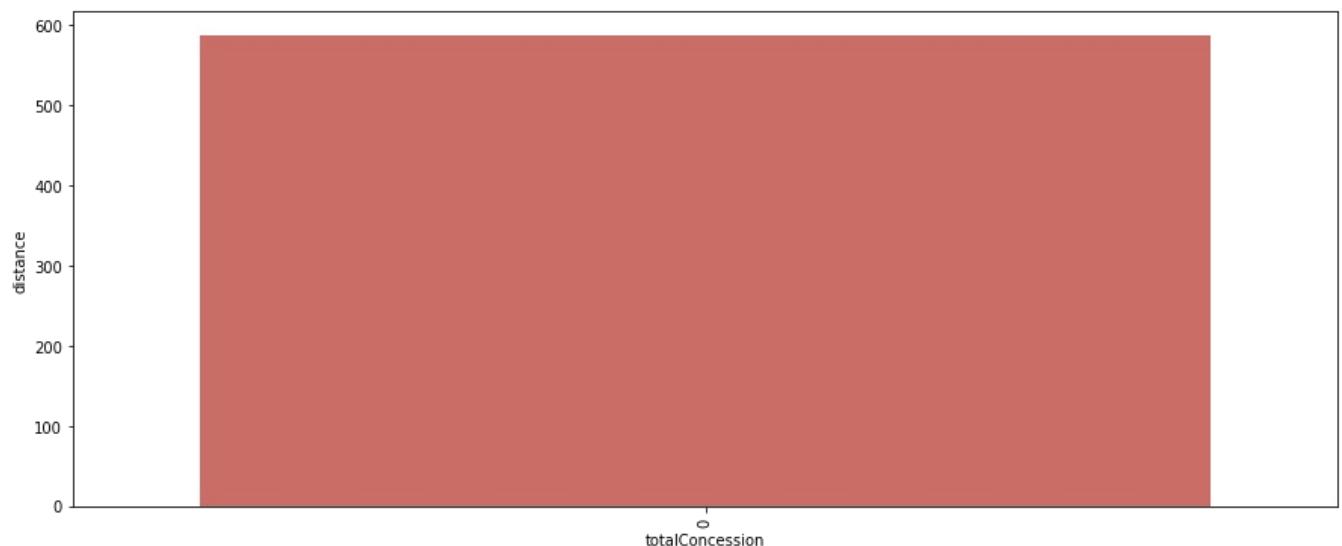
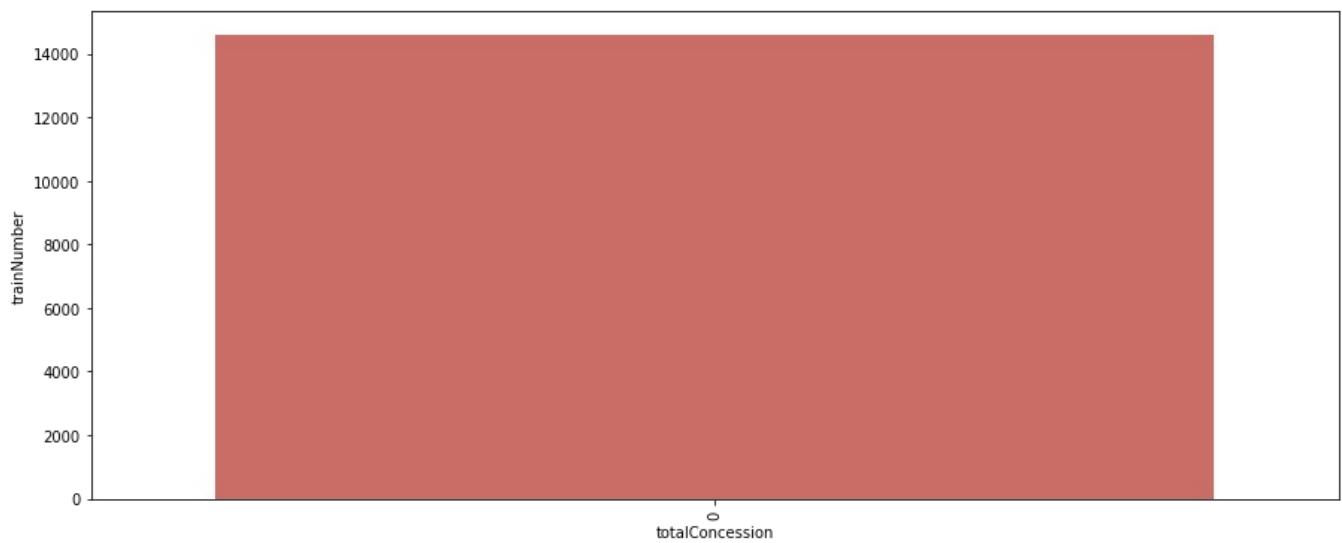
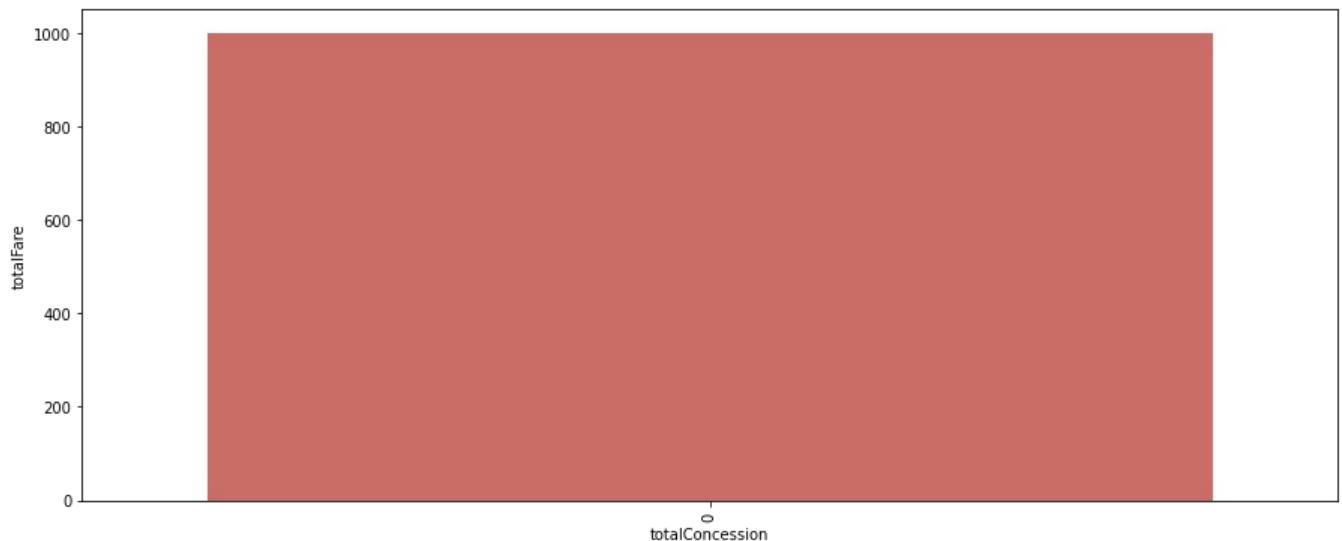


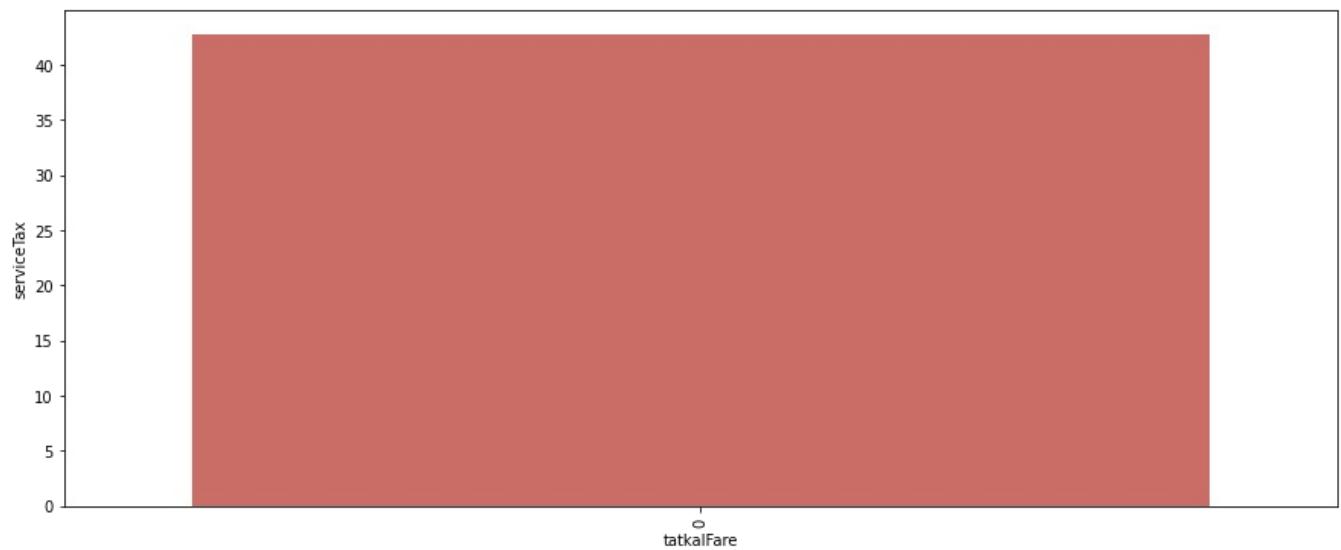
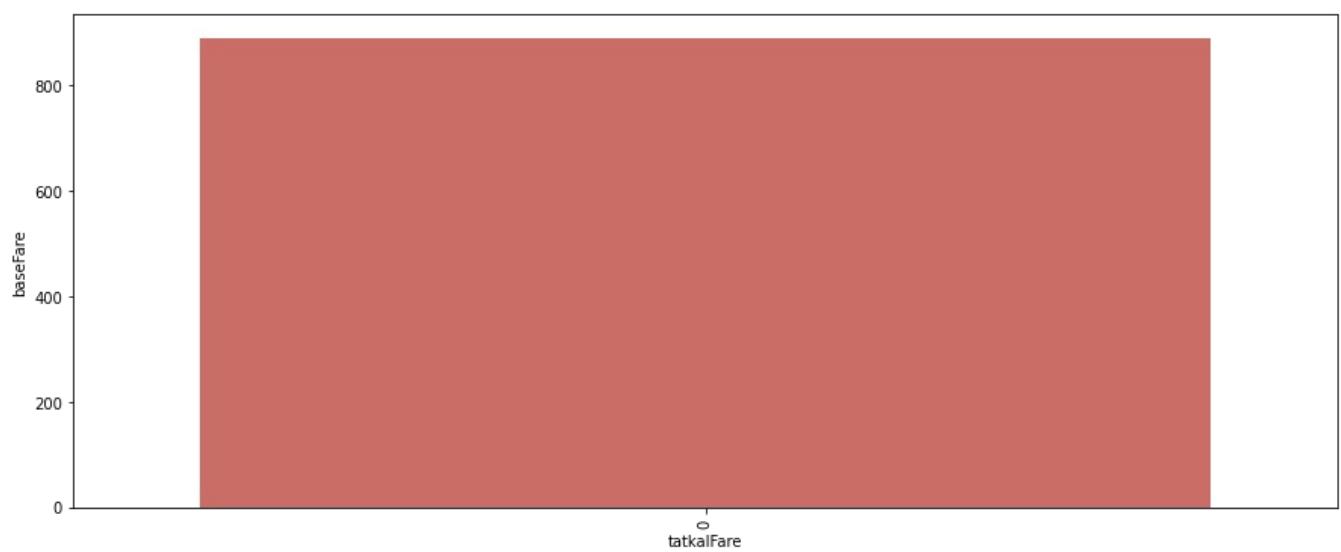
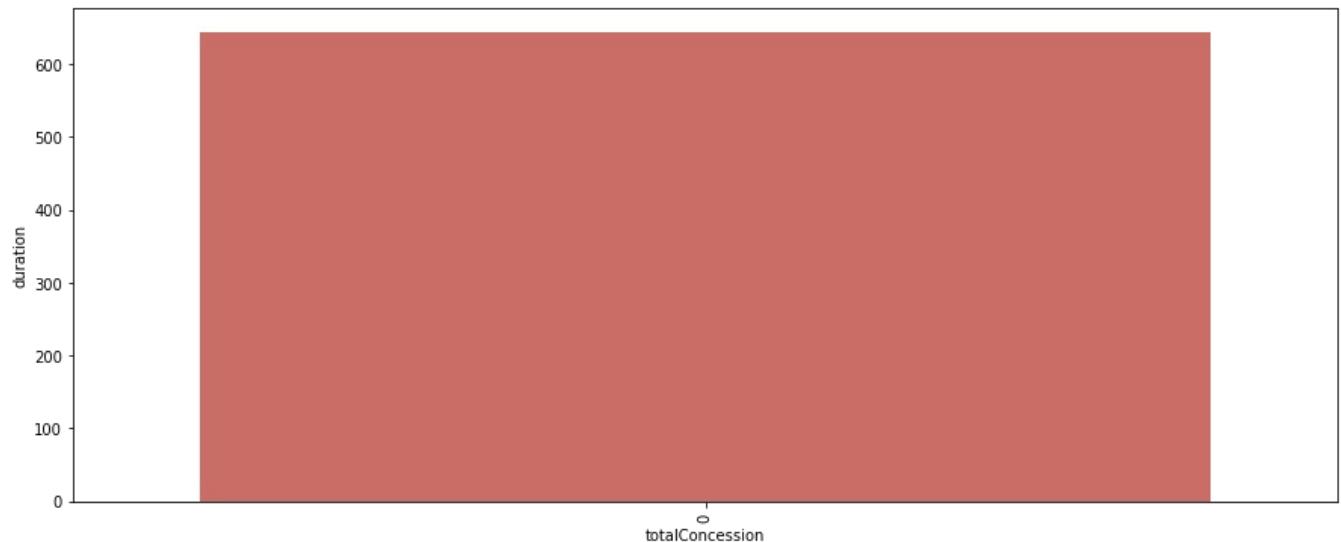


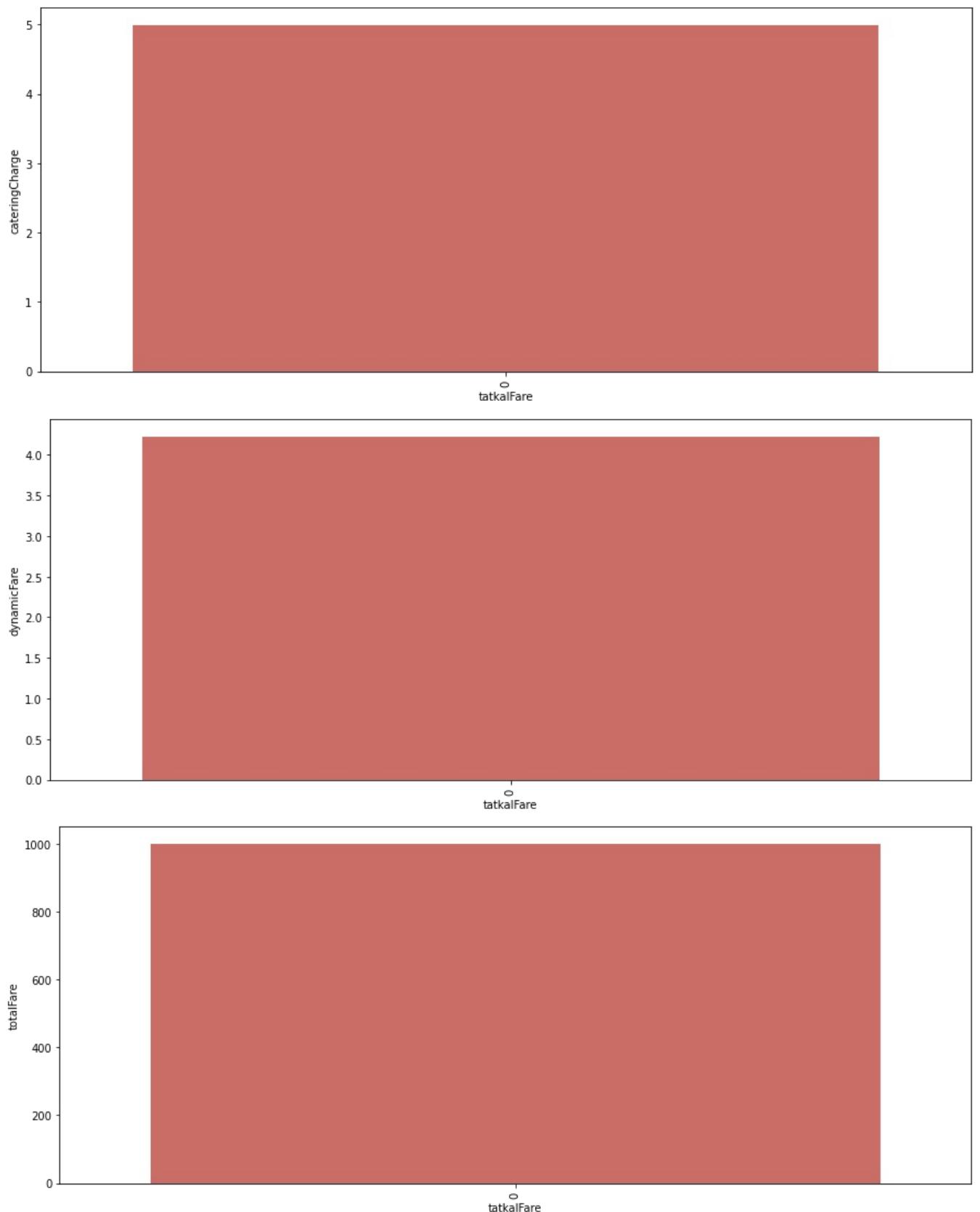


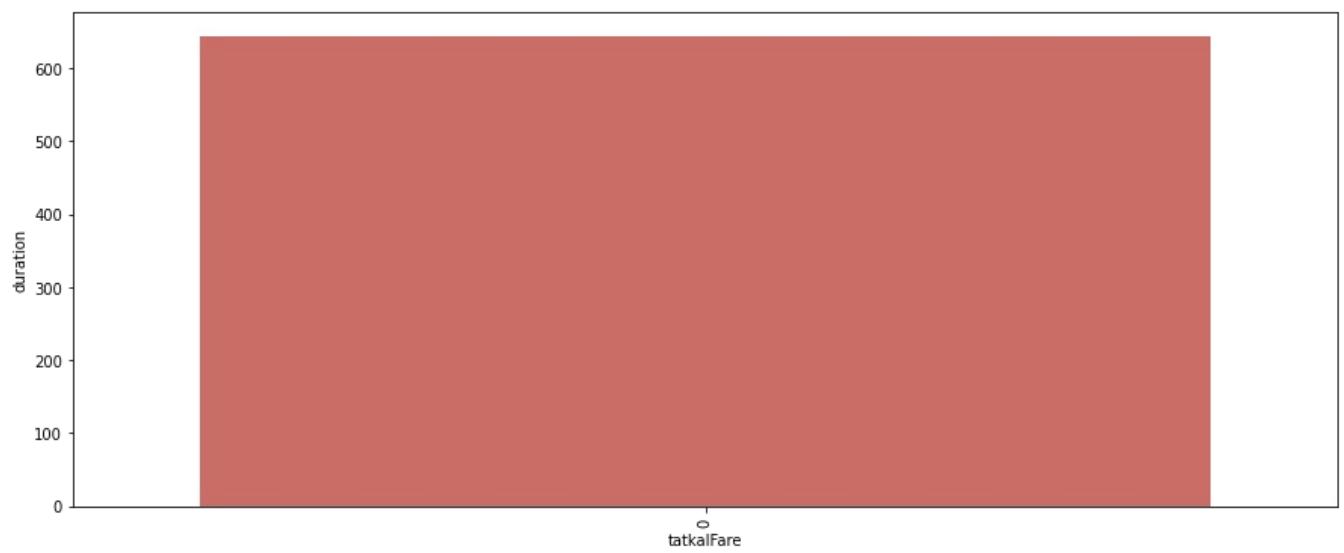
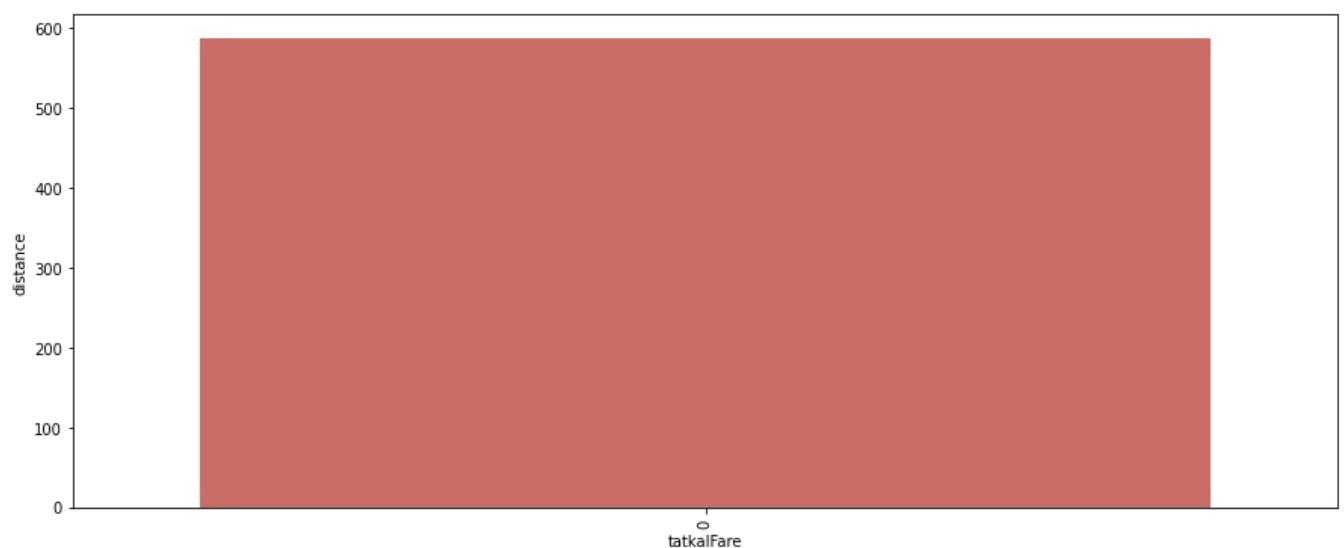
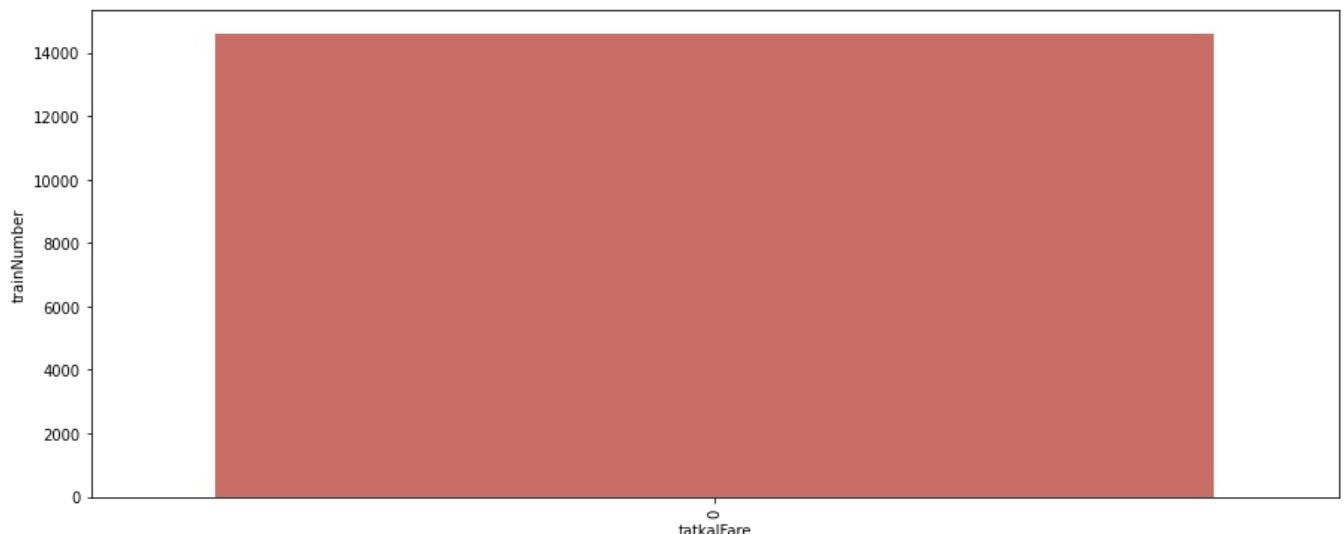


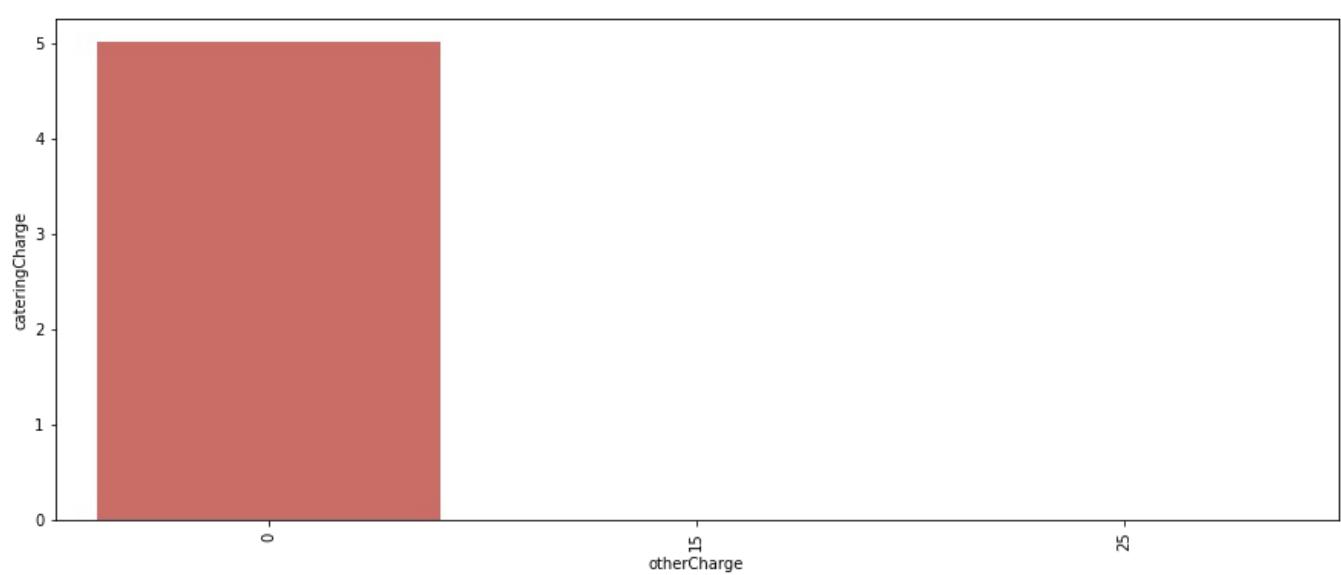
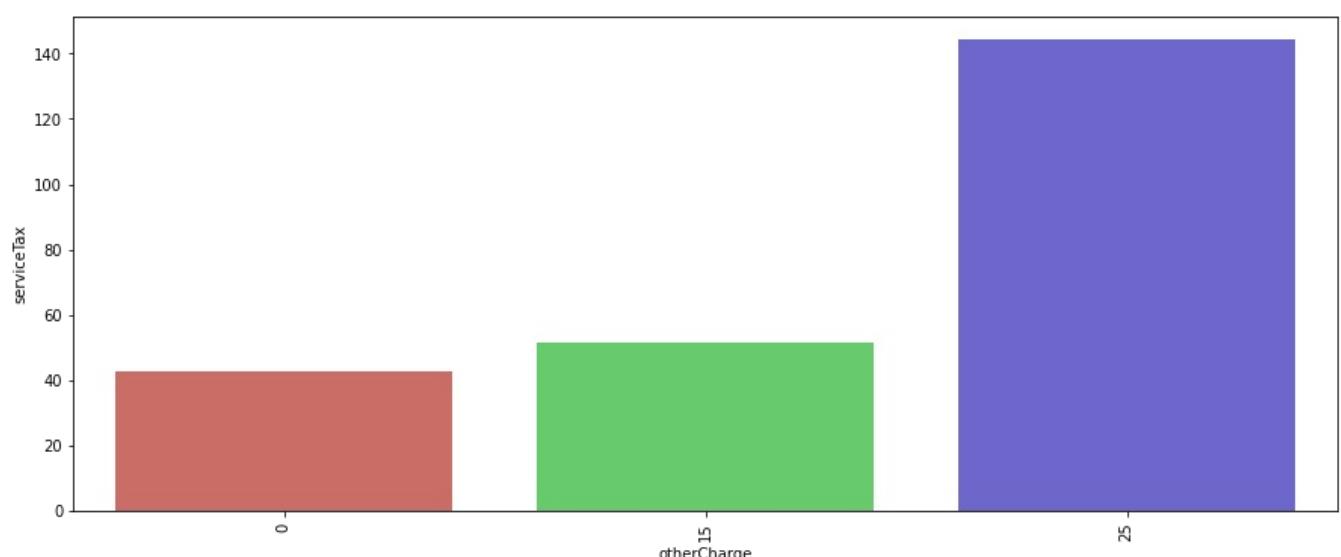
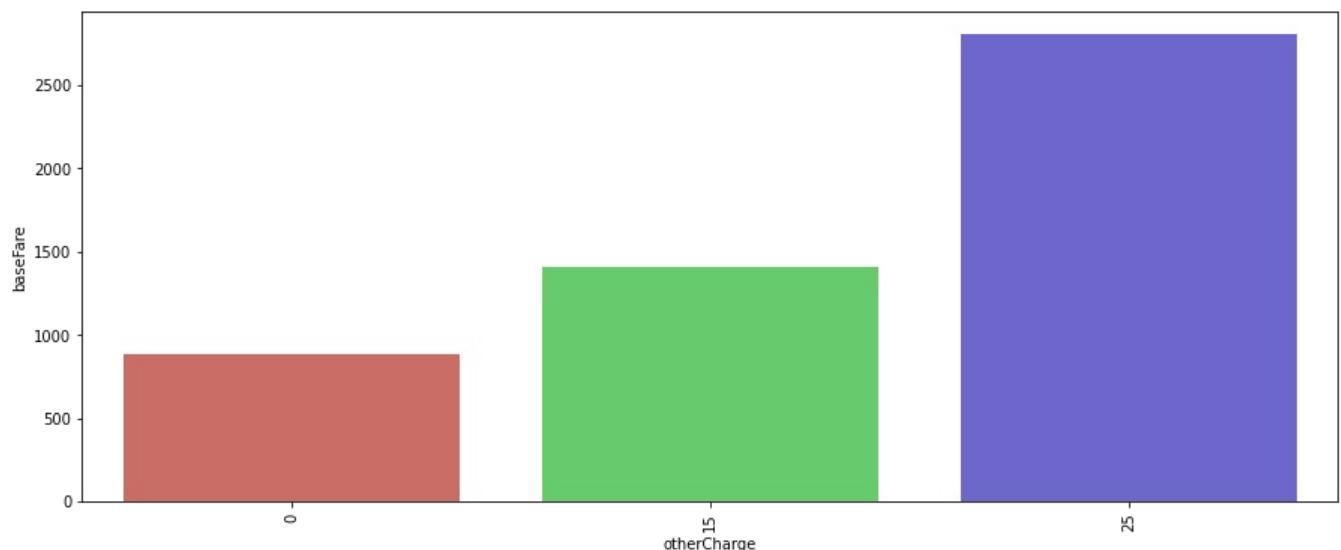


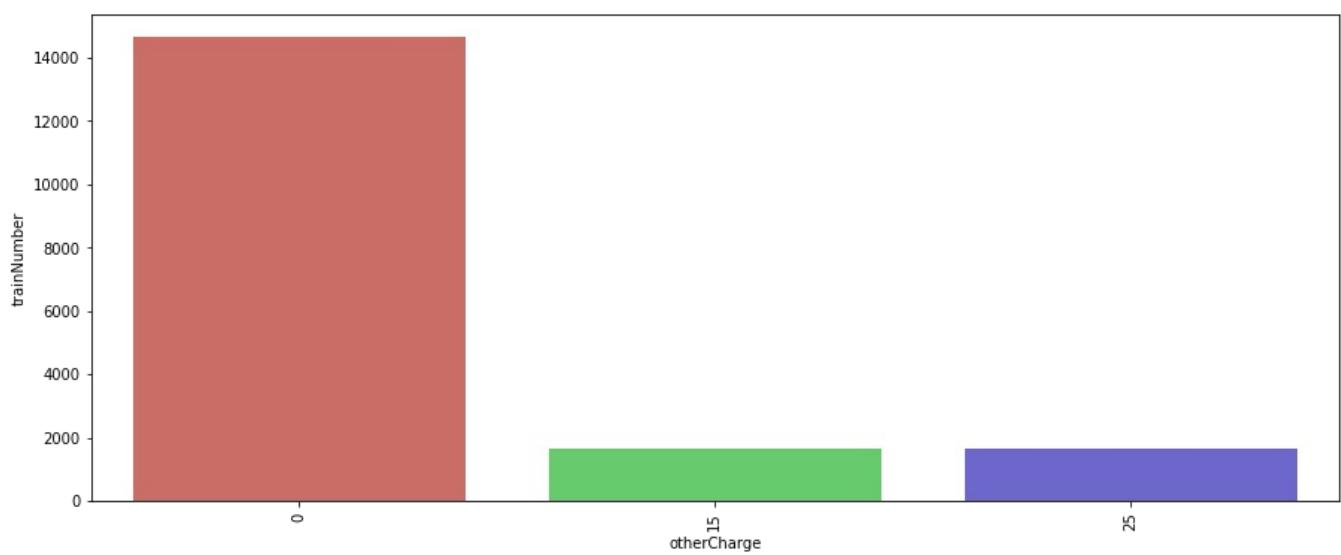
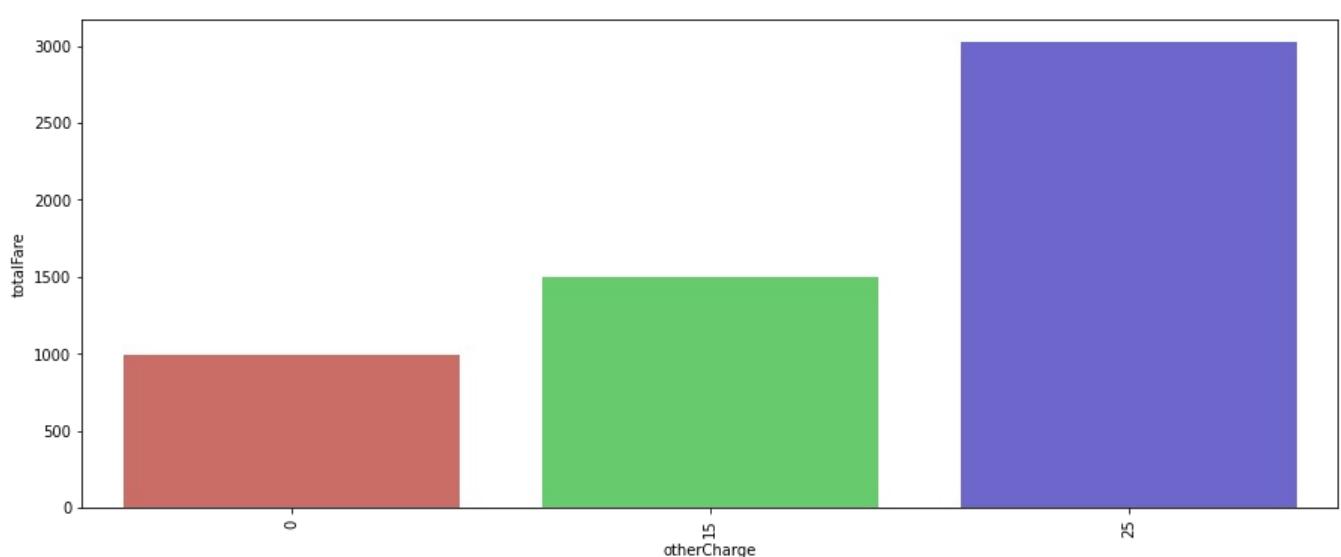
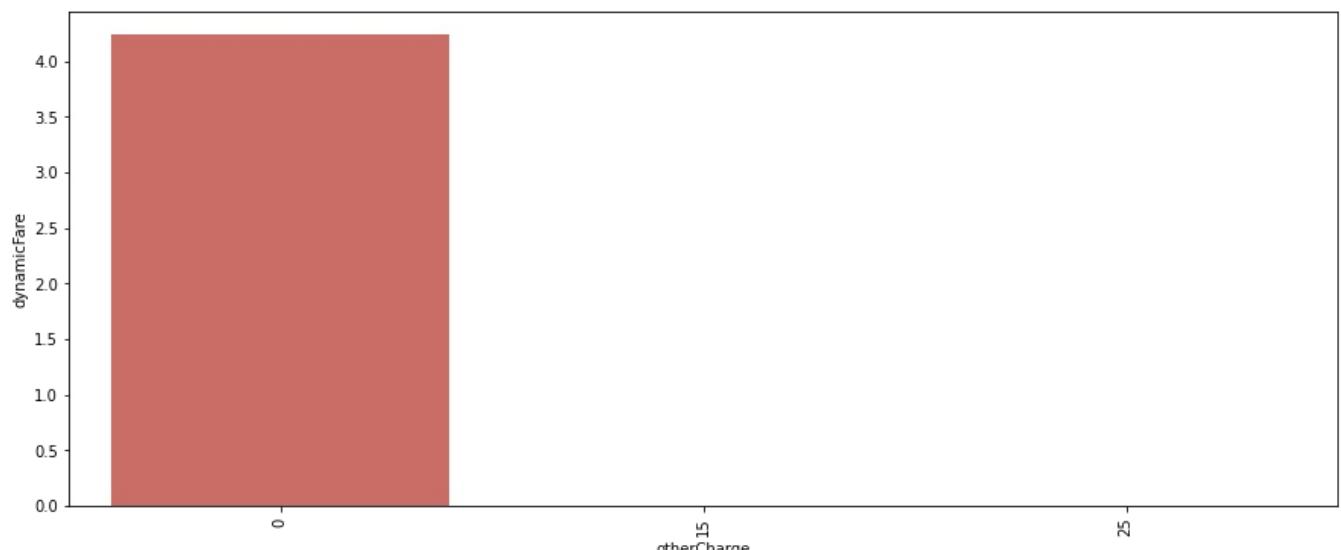


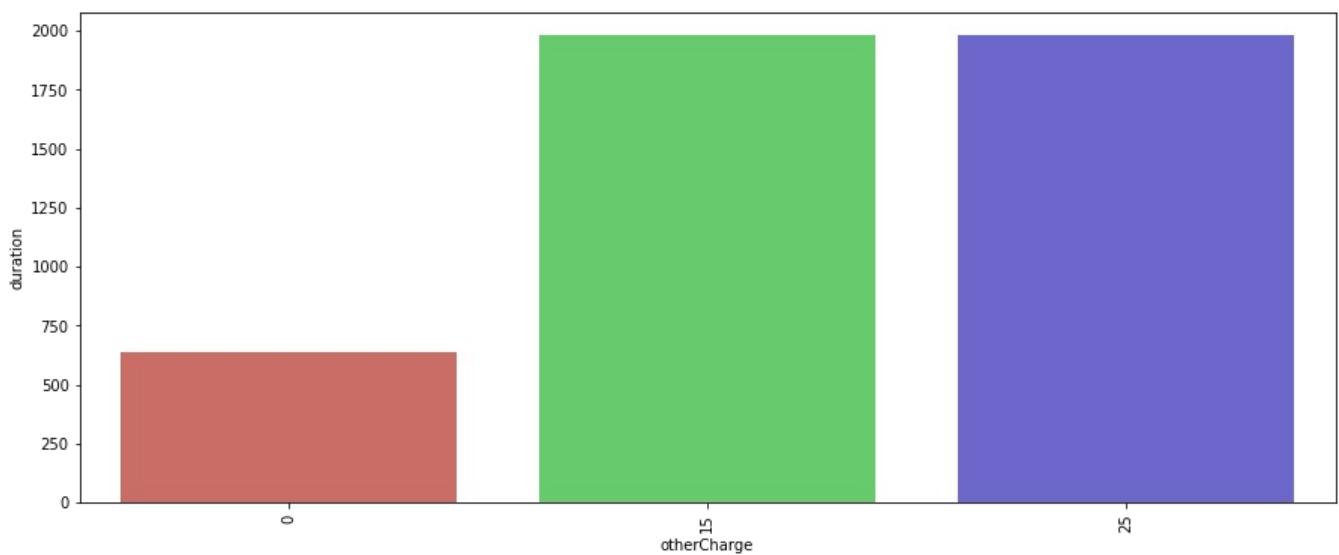
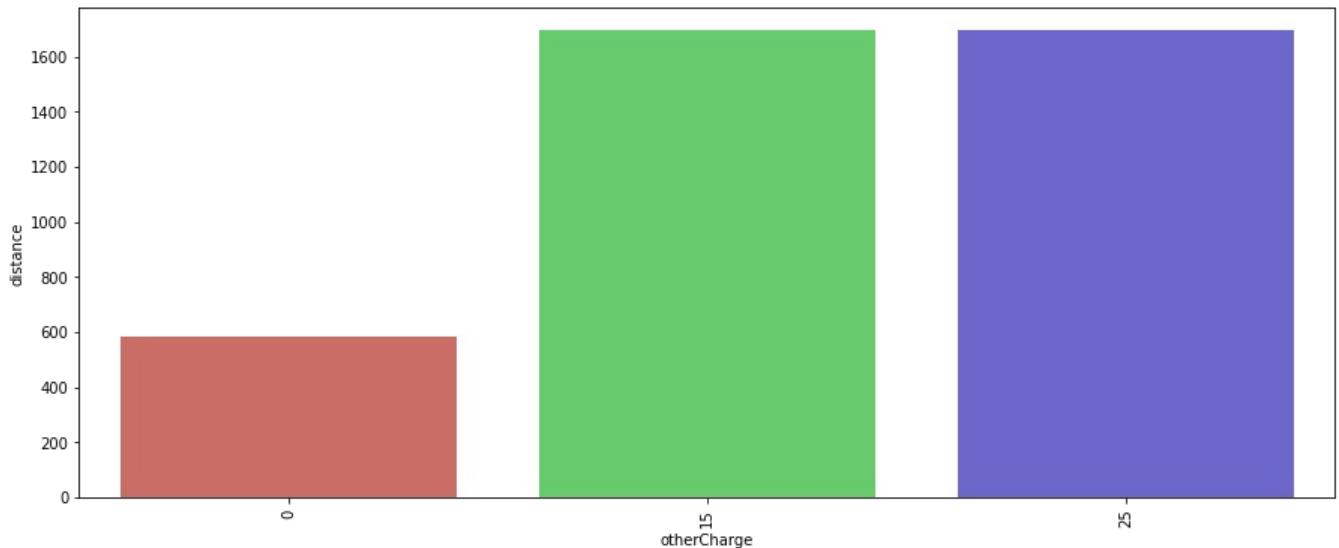




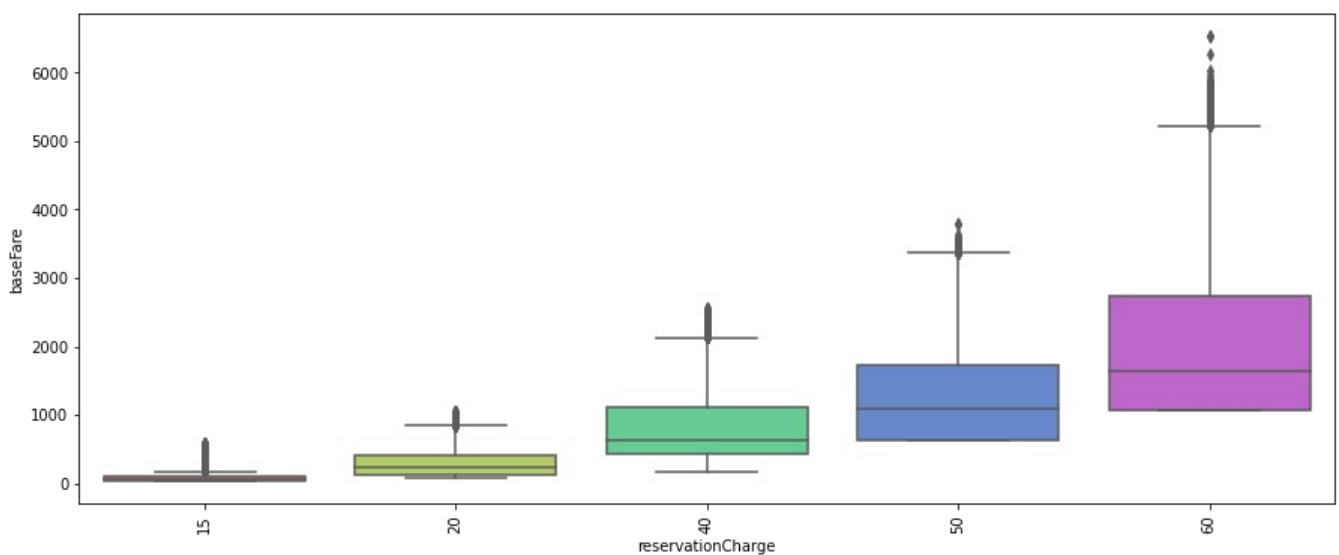


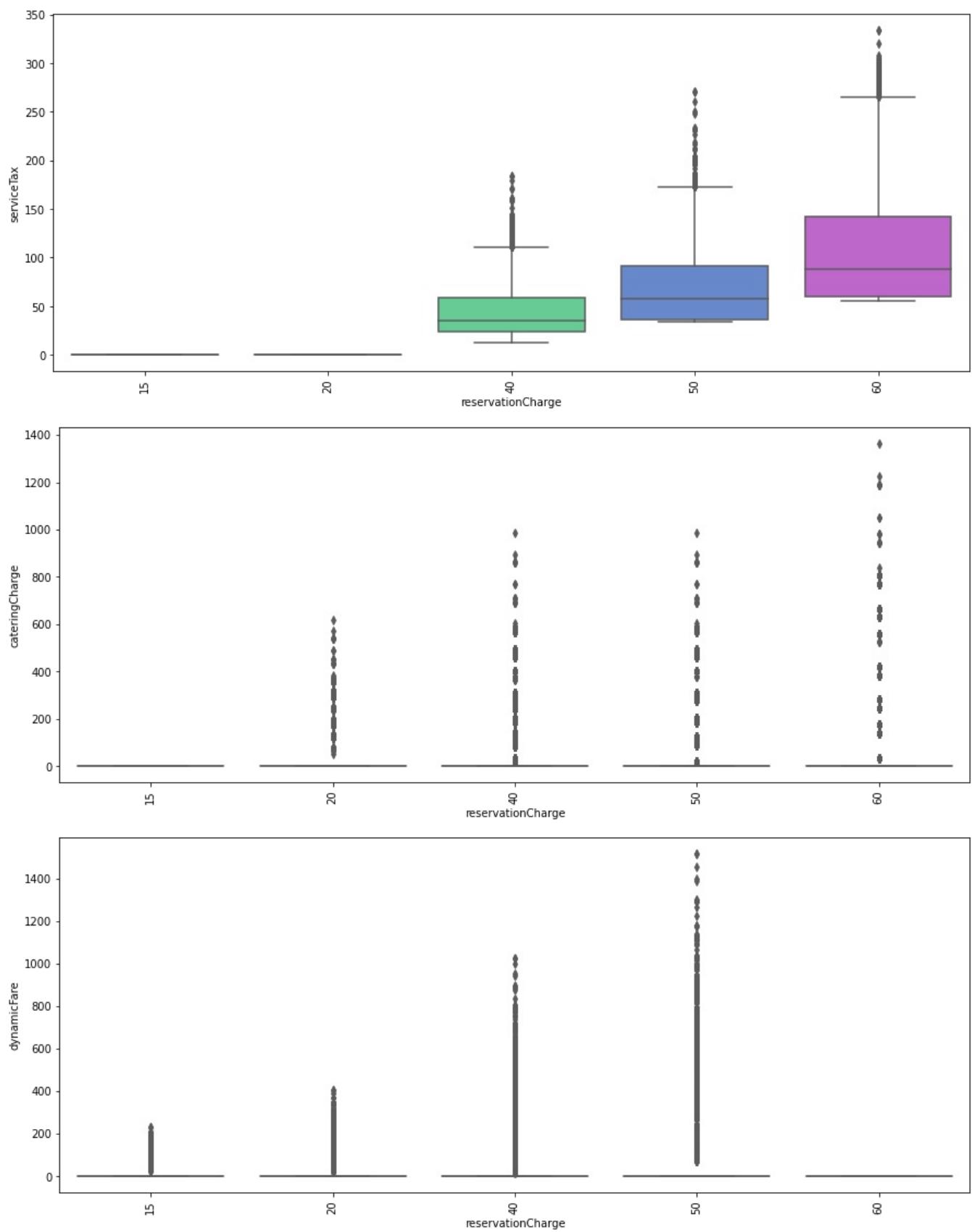


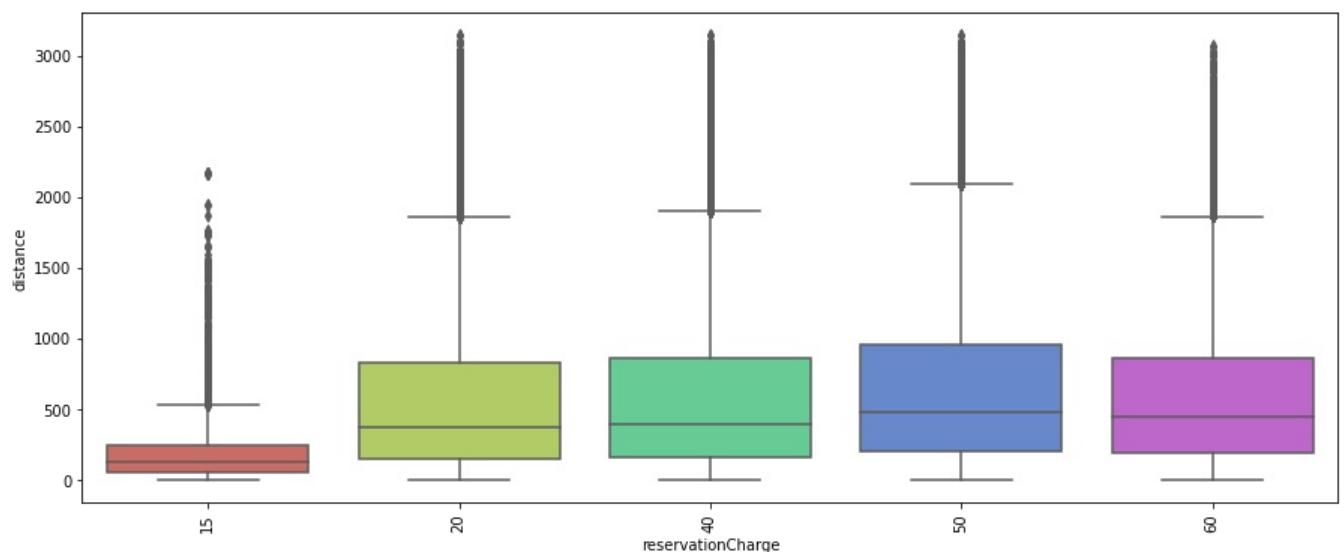
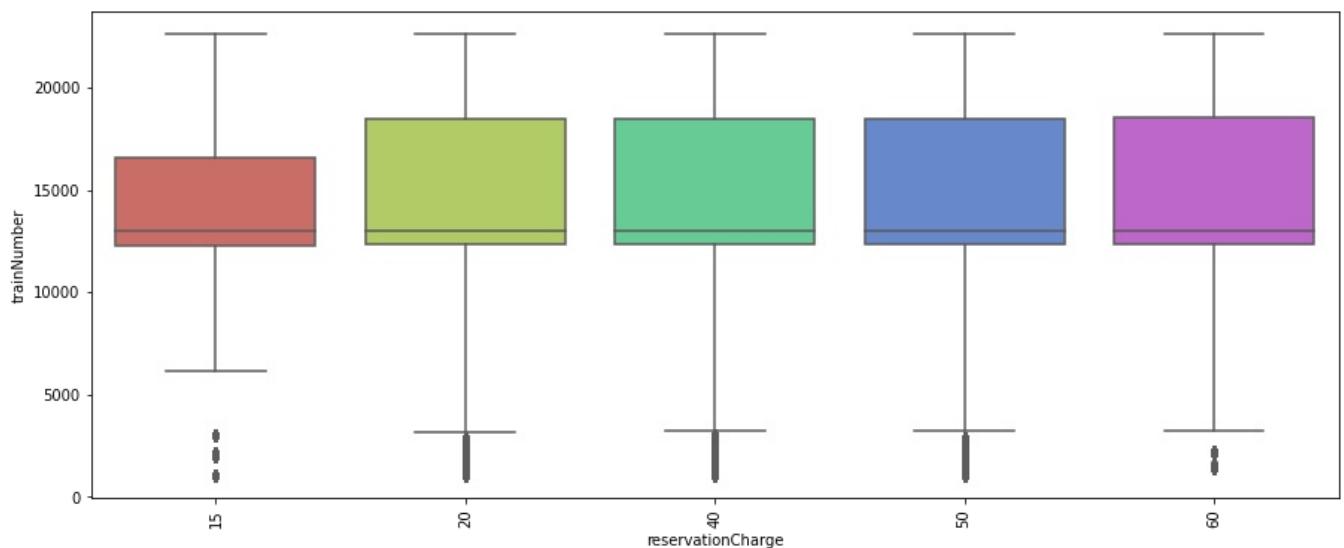
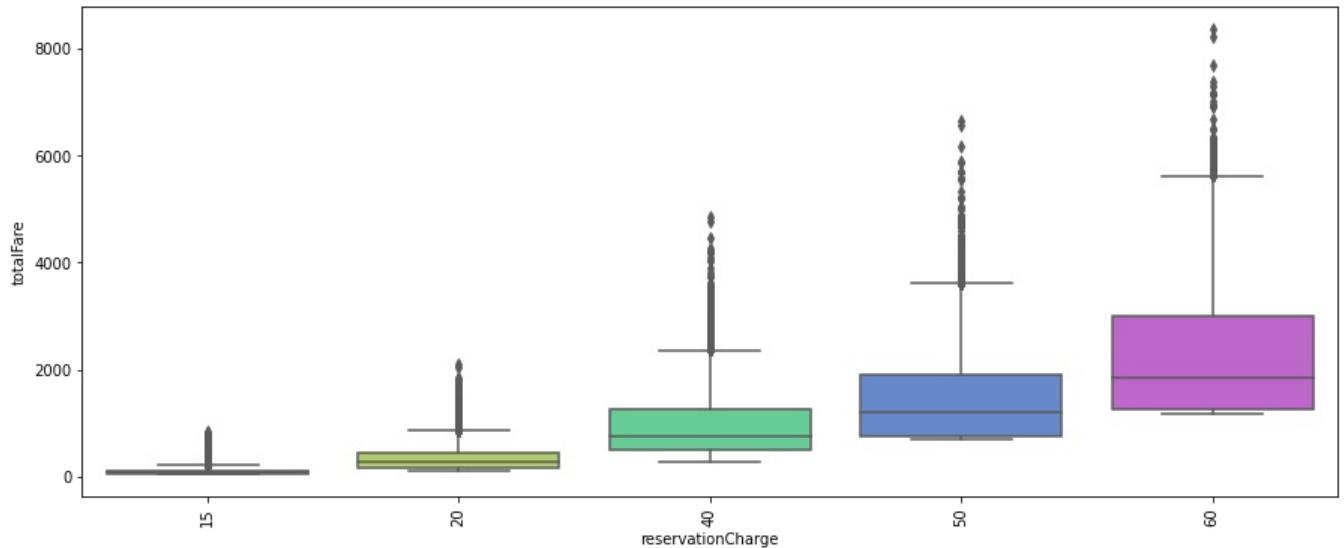


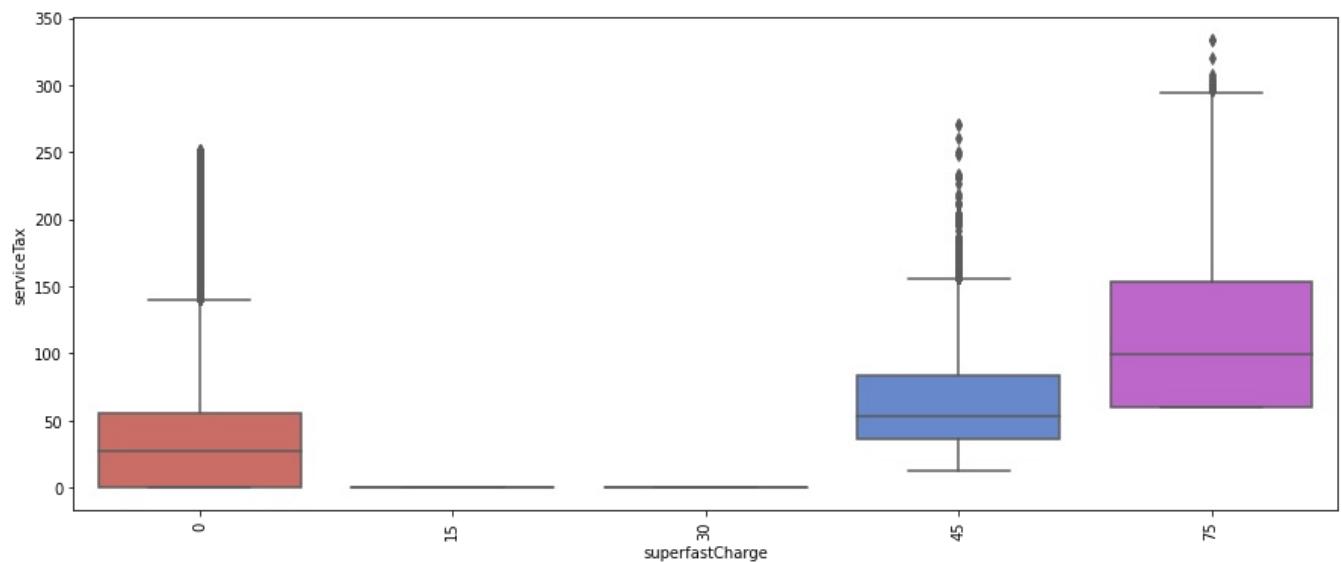
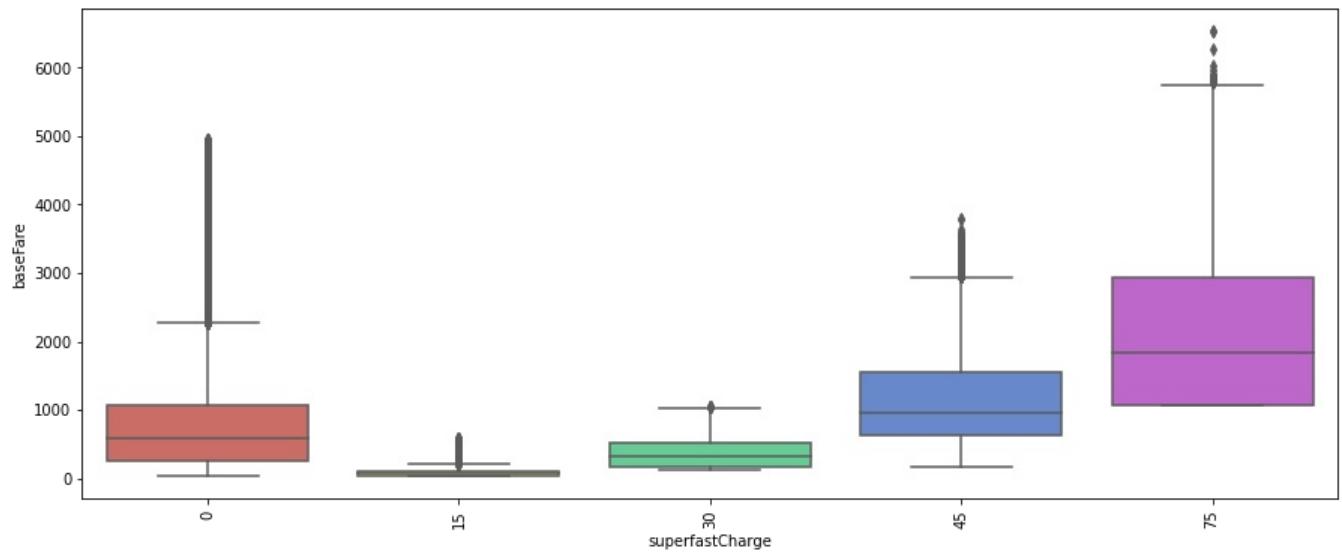
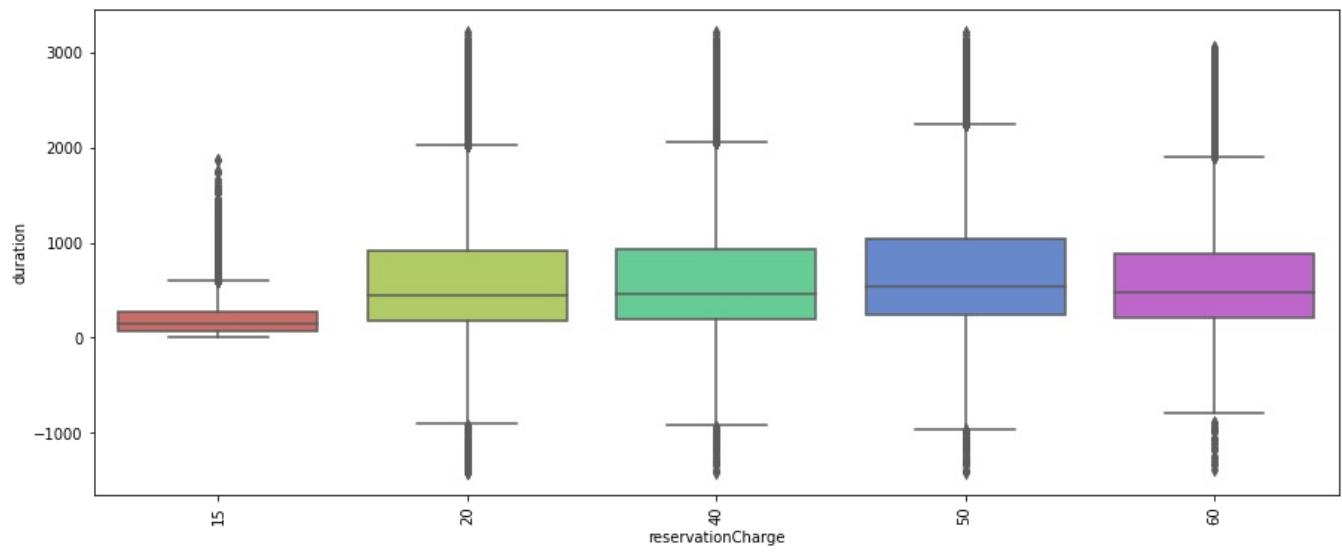


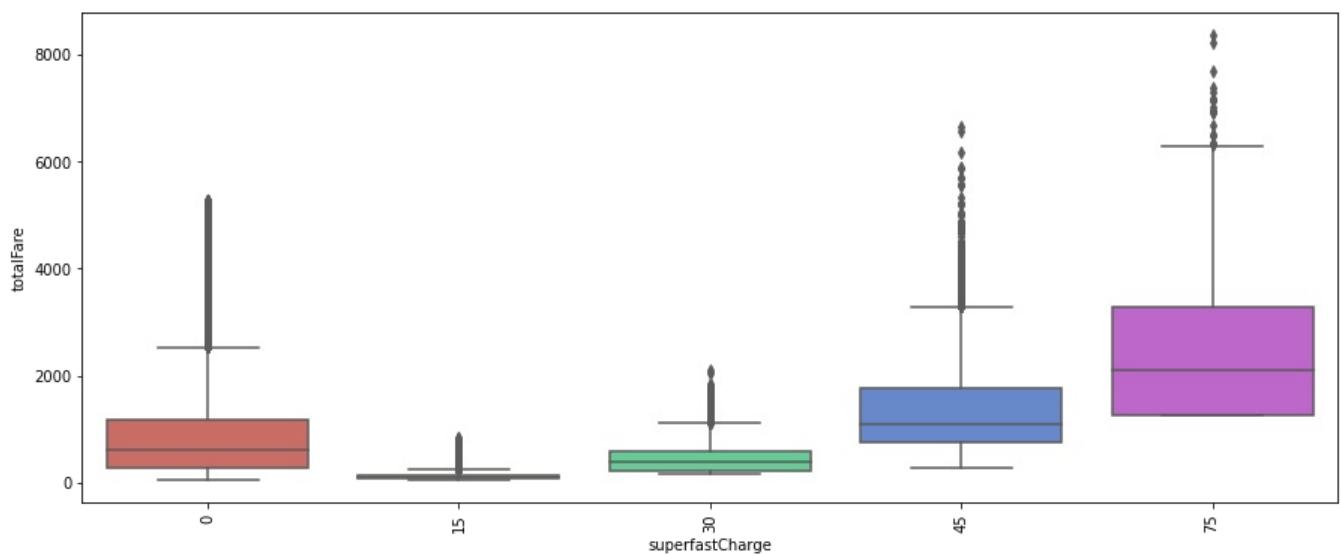
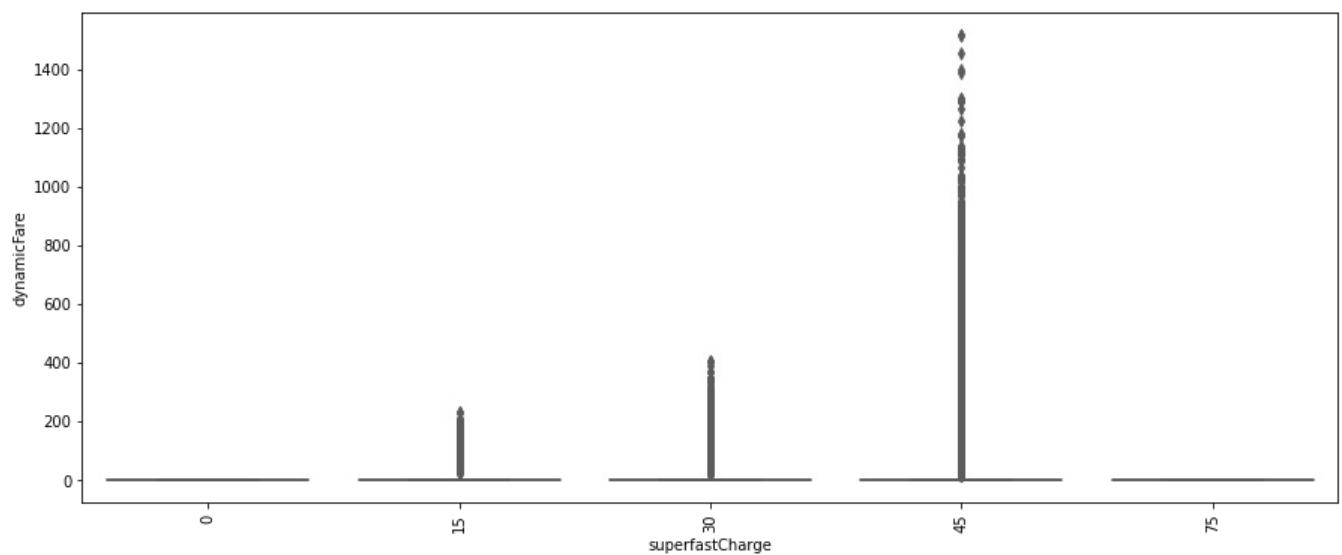
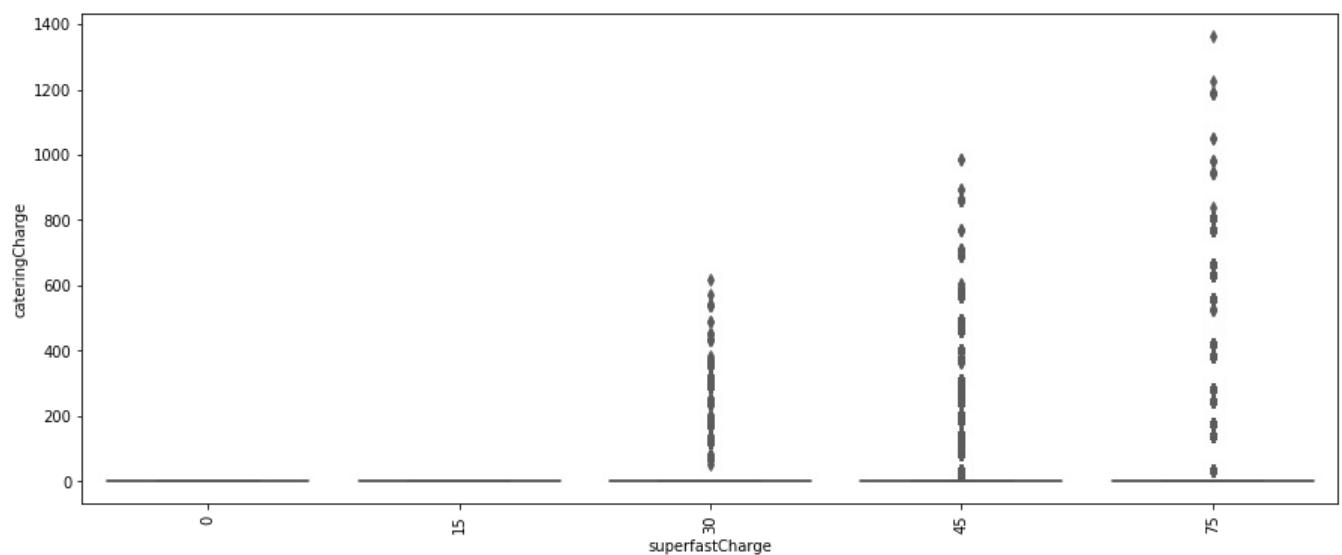
```
In [35]: for i in discrete:  
    for j in continuous:  
        plt.figure(figsize=(15,6))  
        sns.boxplot(x = df[i], y = df[j], data = df, palette = 'hls')  
        plt.xticks(rotation = 90)  
        plt.show()
```

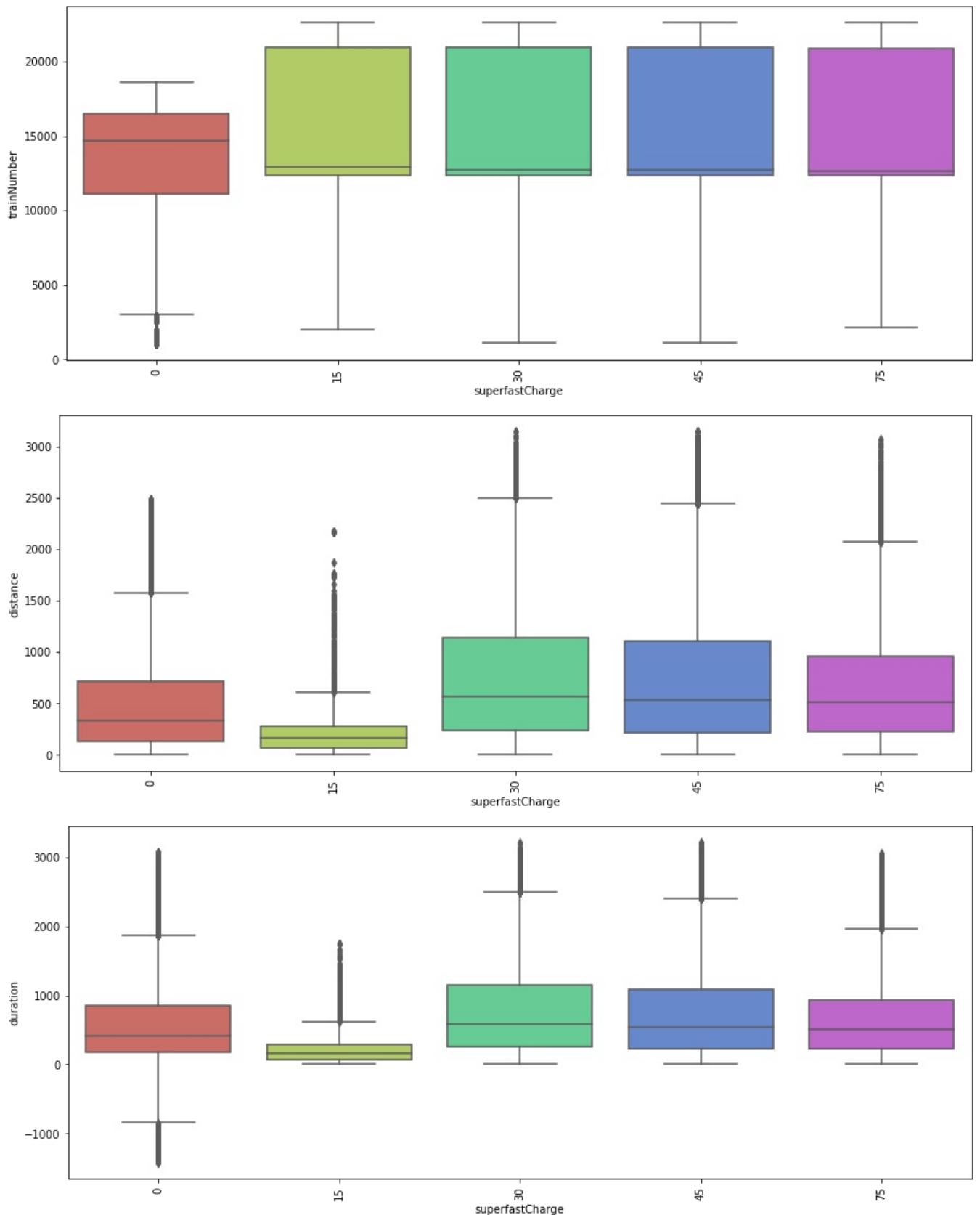


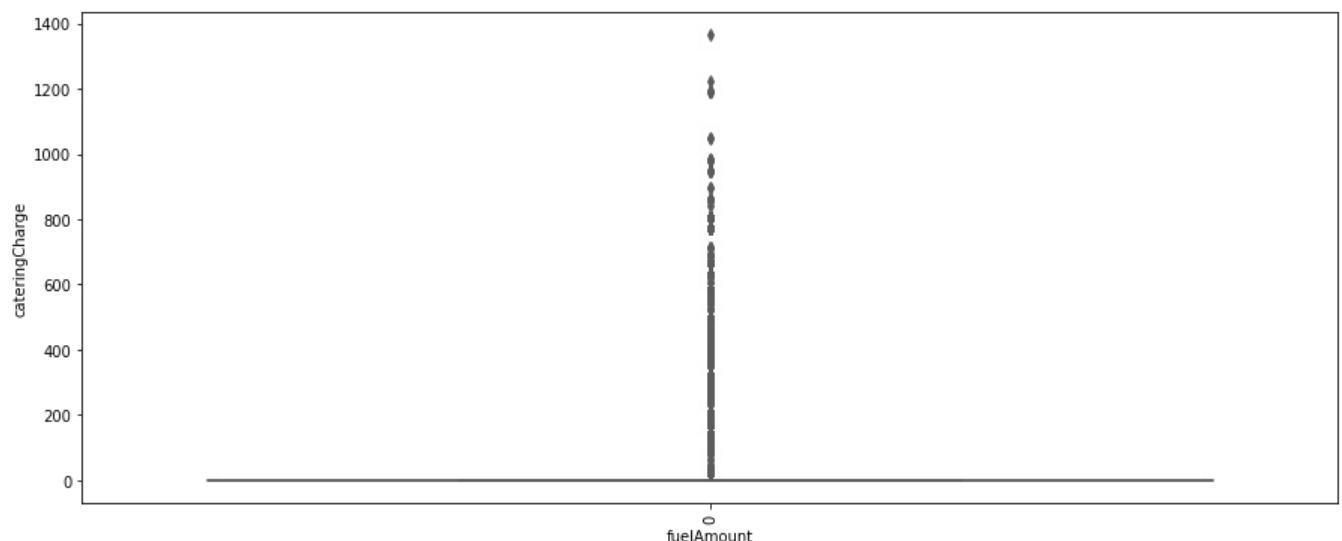
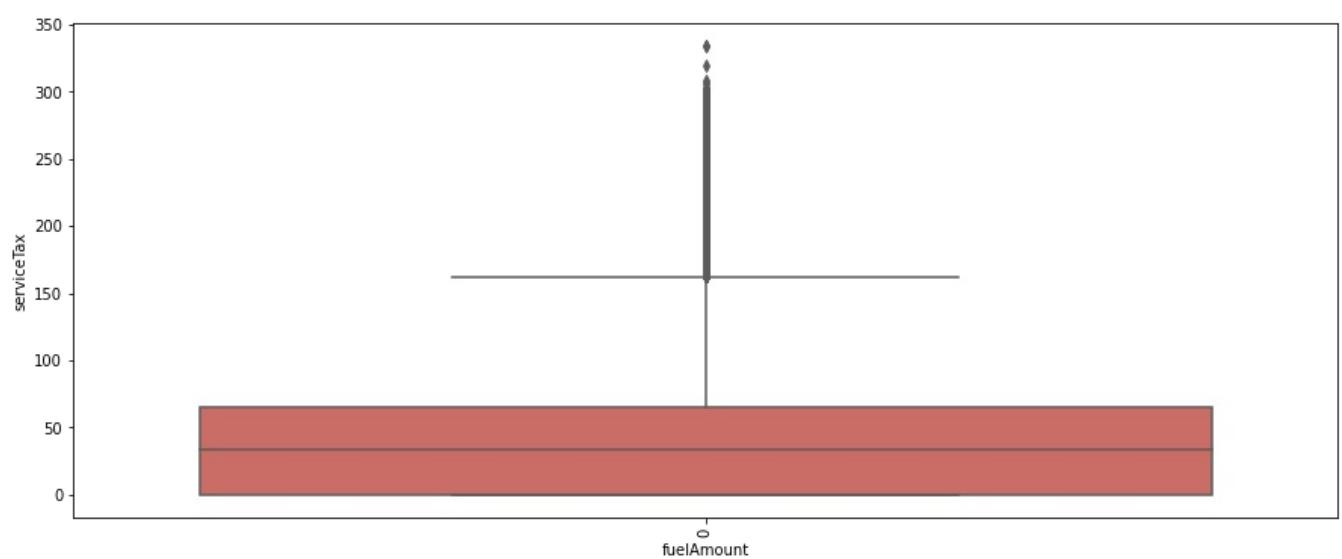
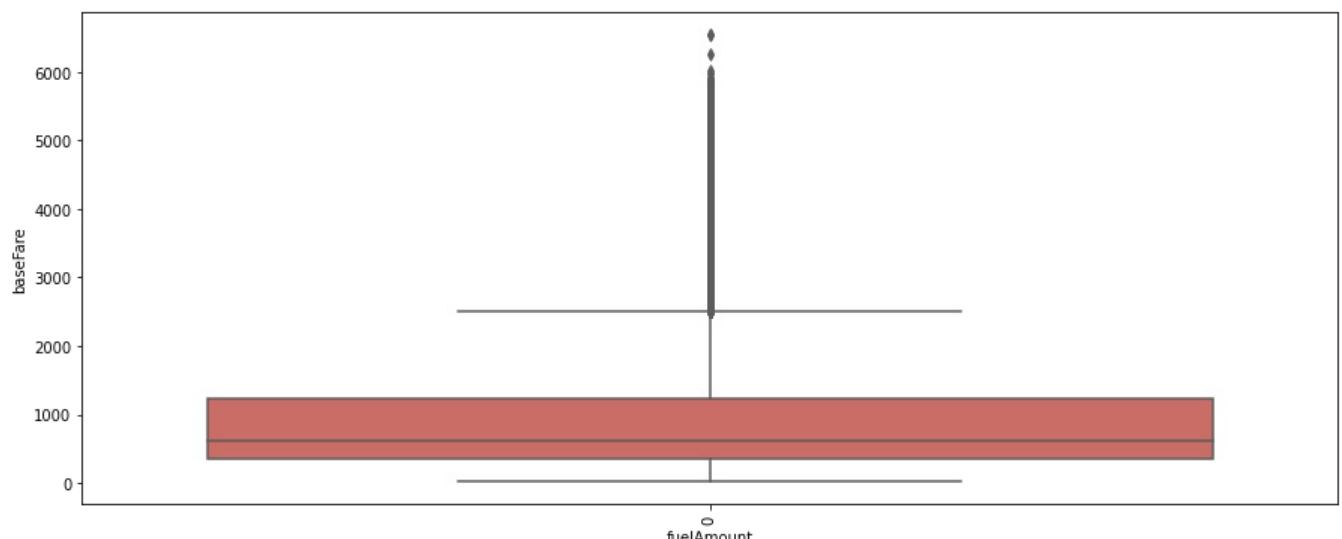


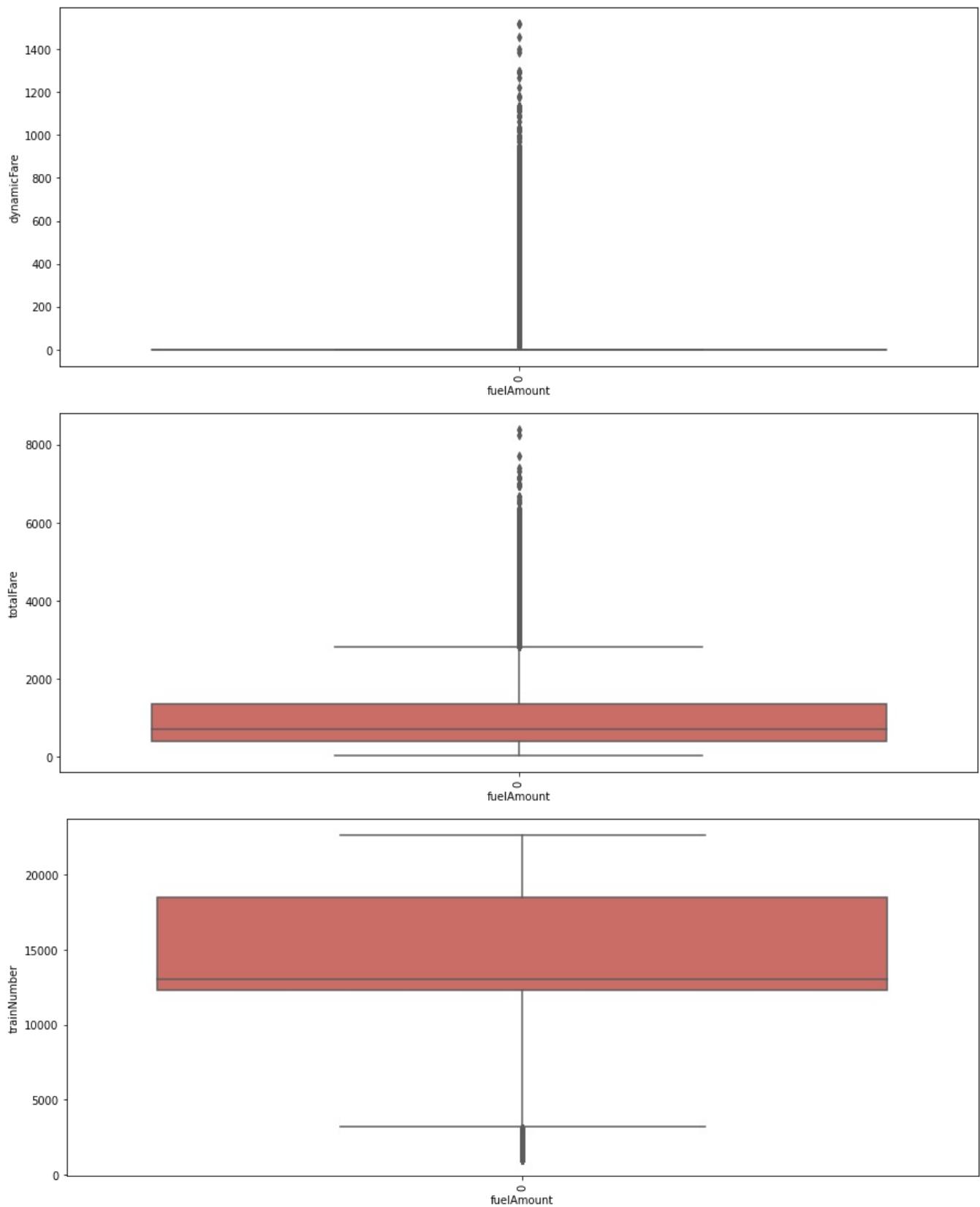


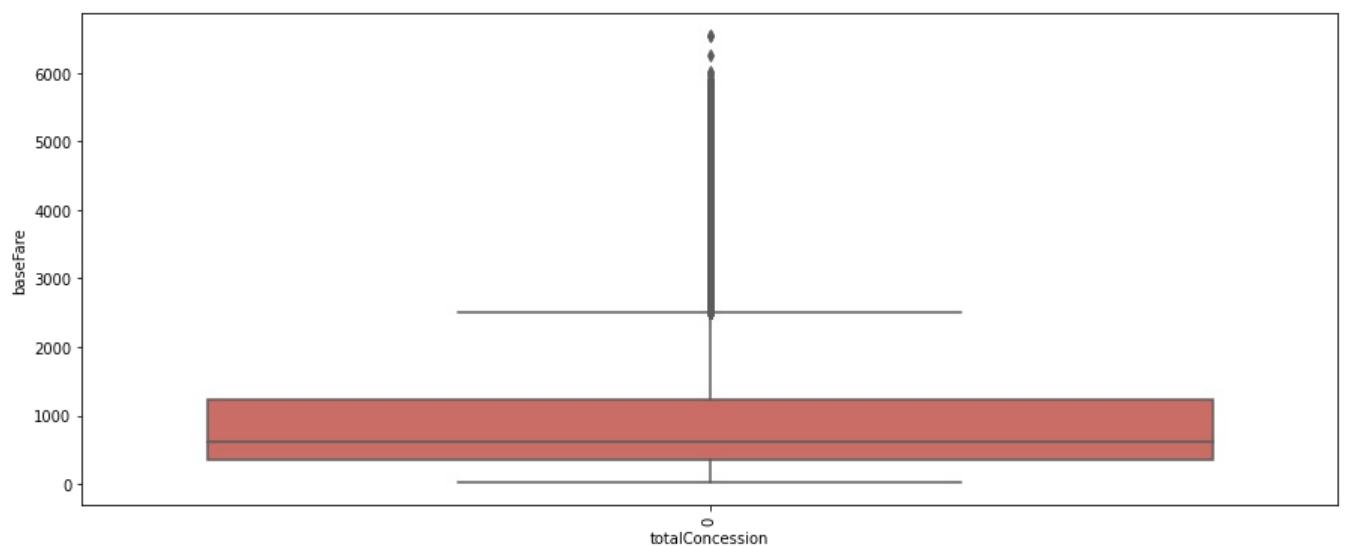
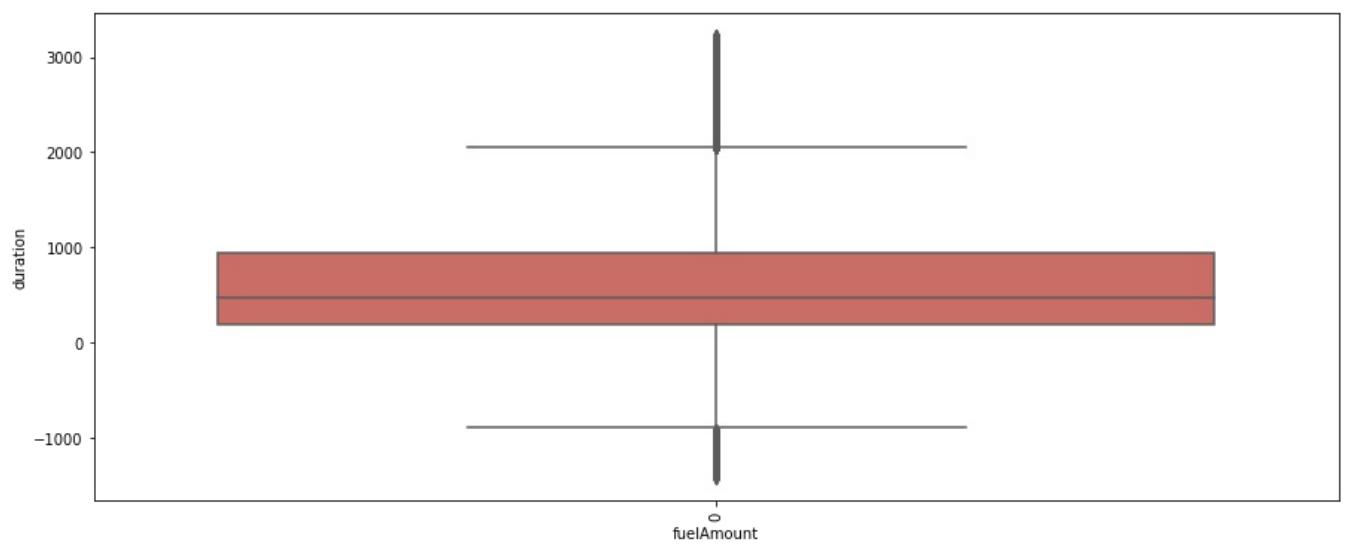
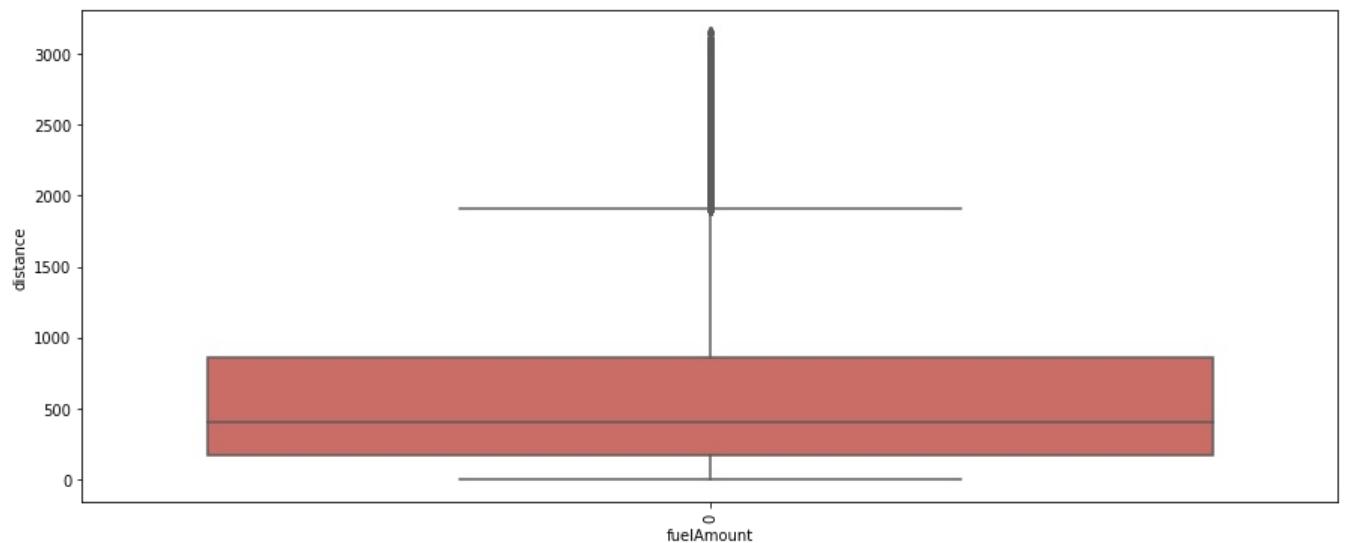


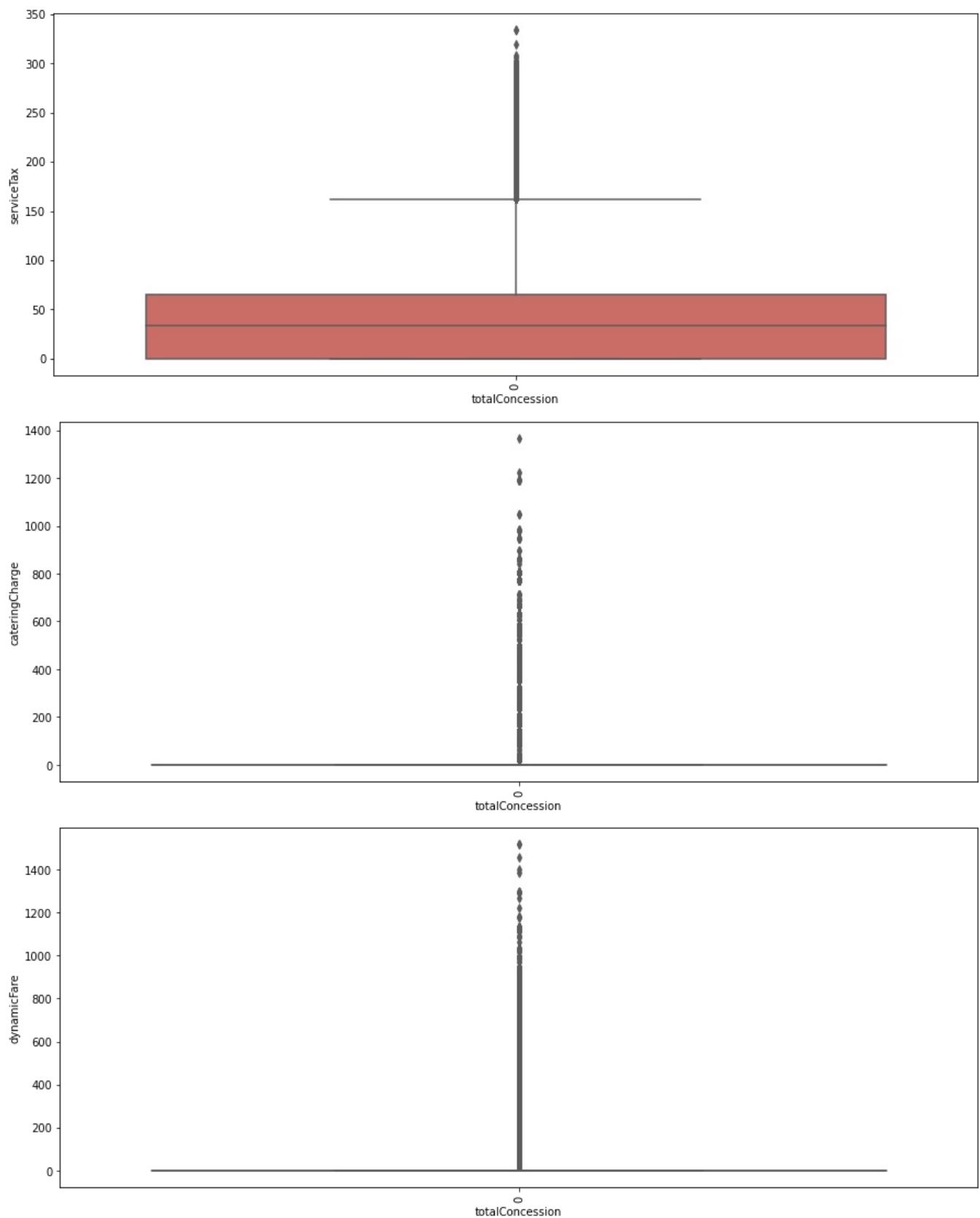


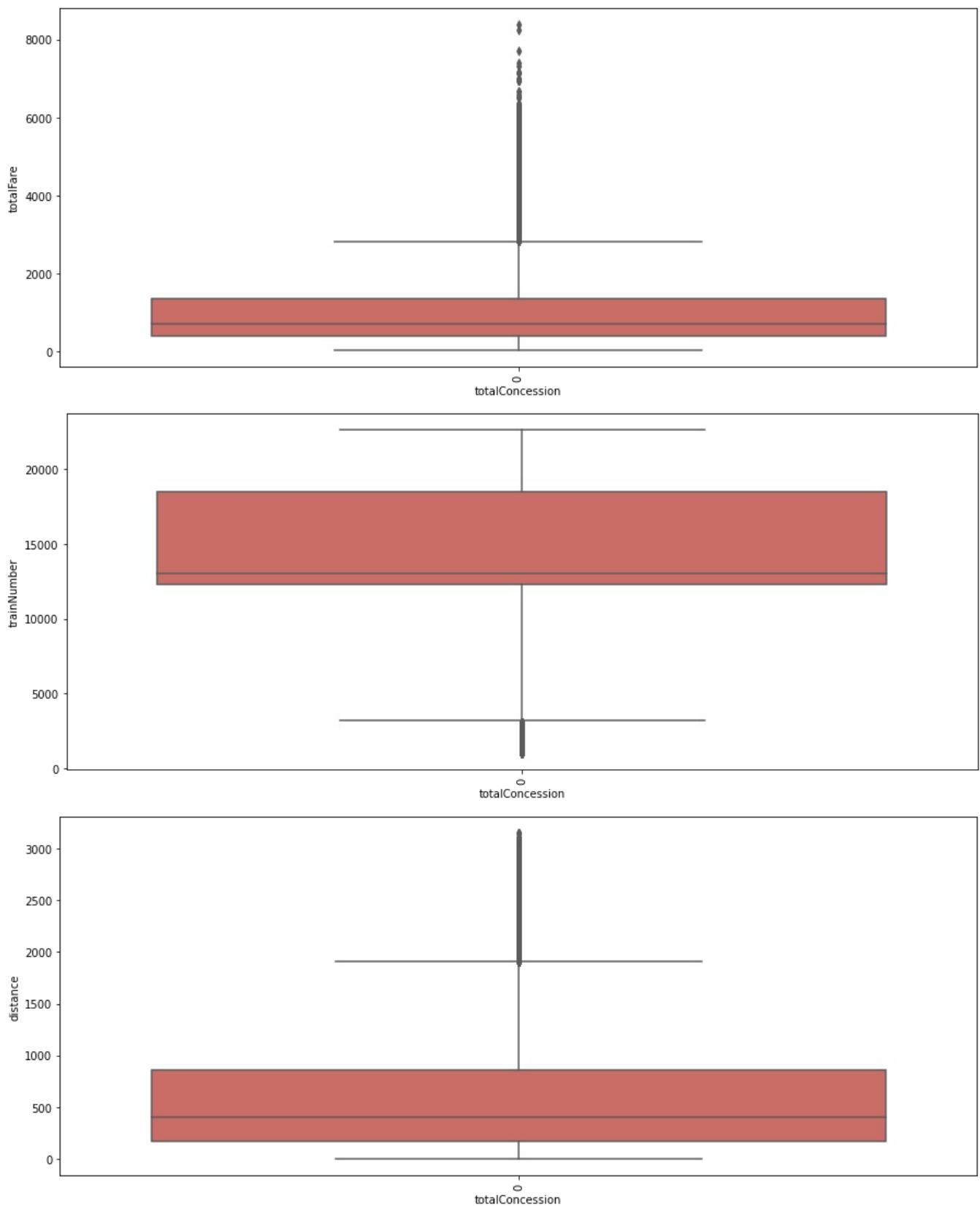


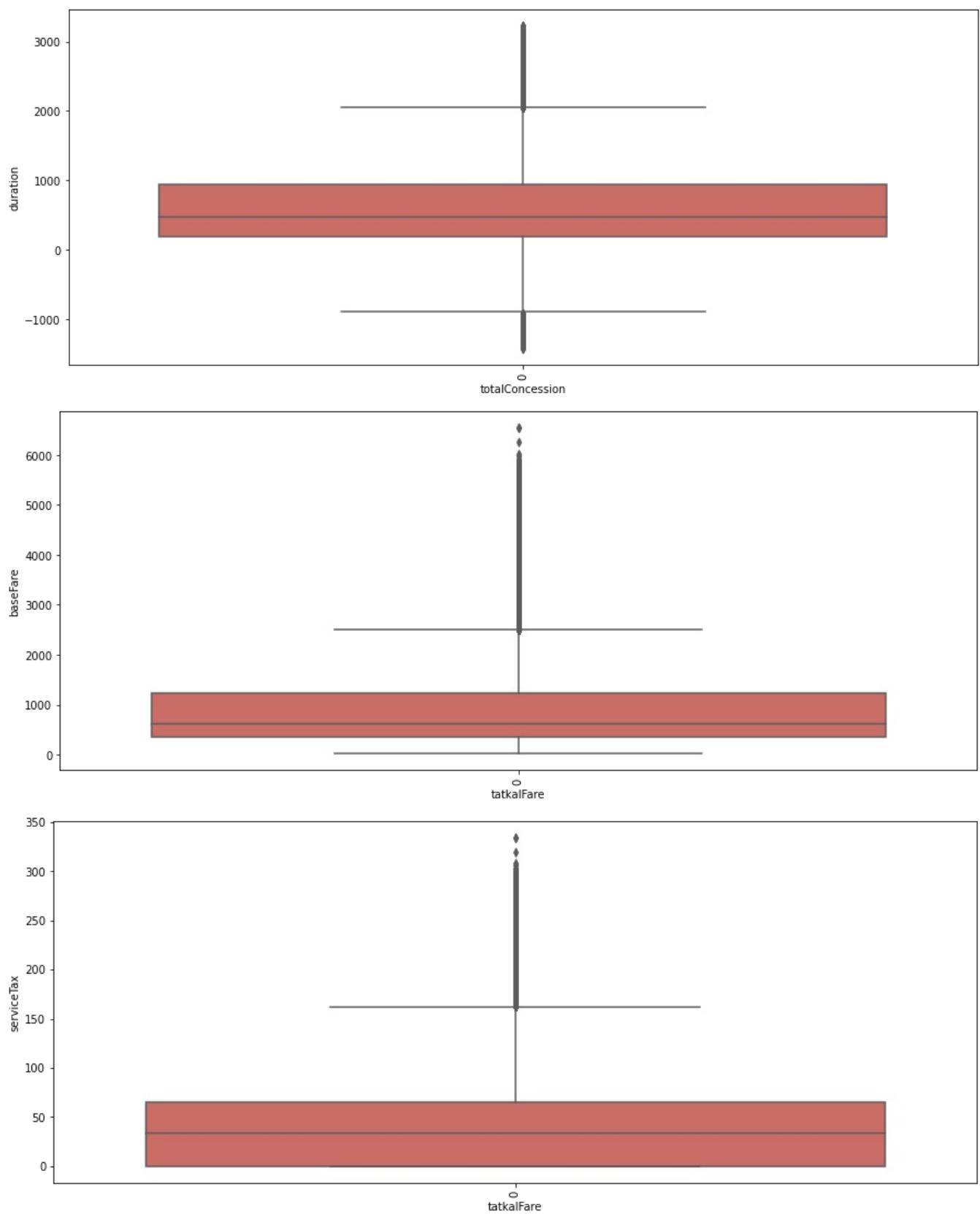


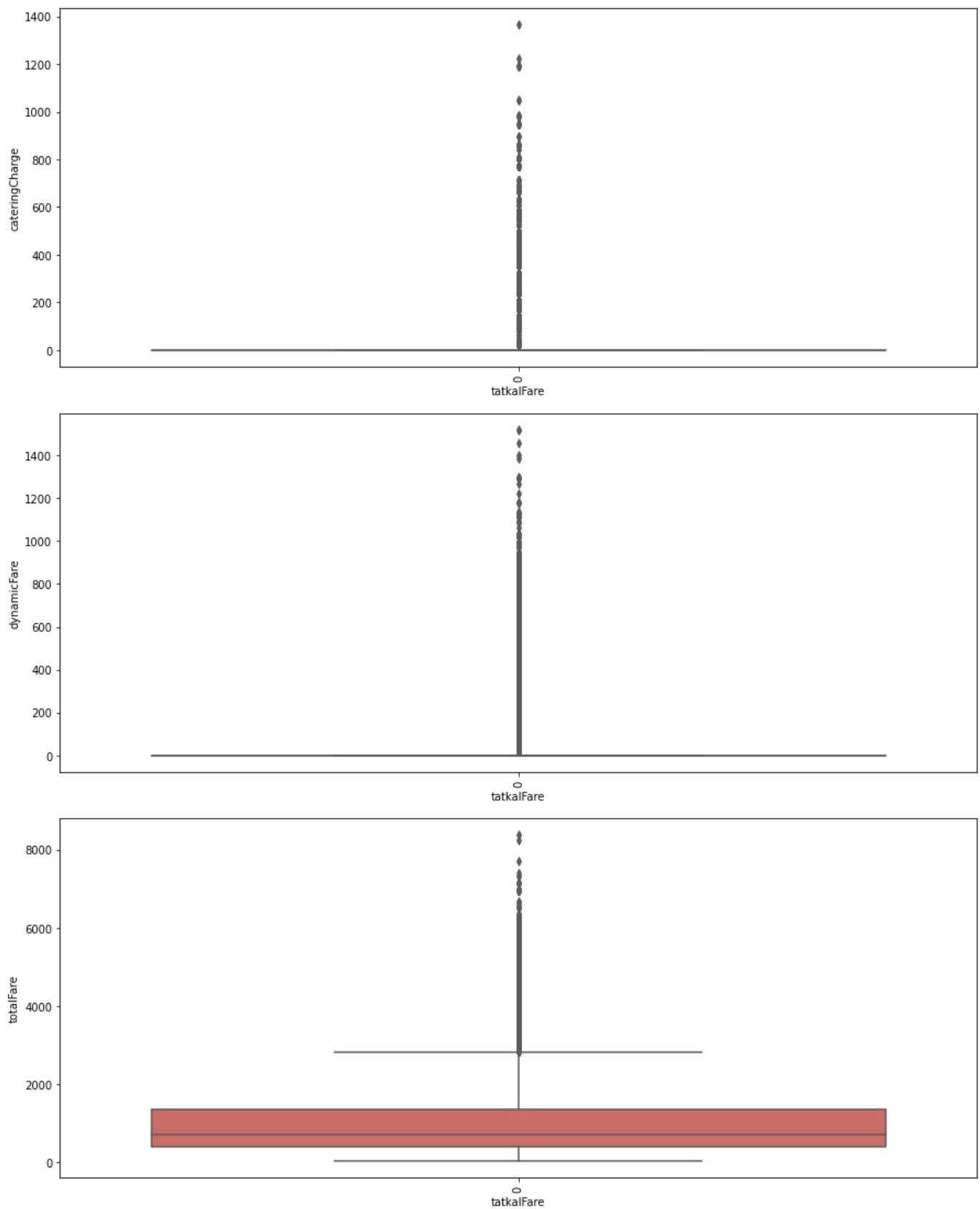


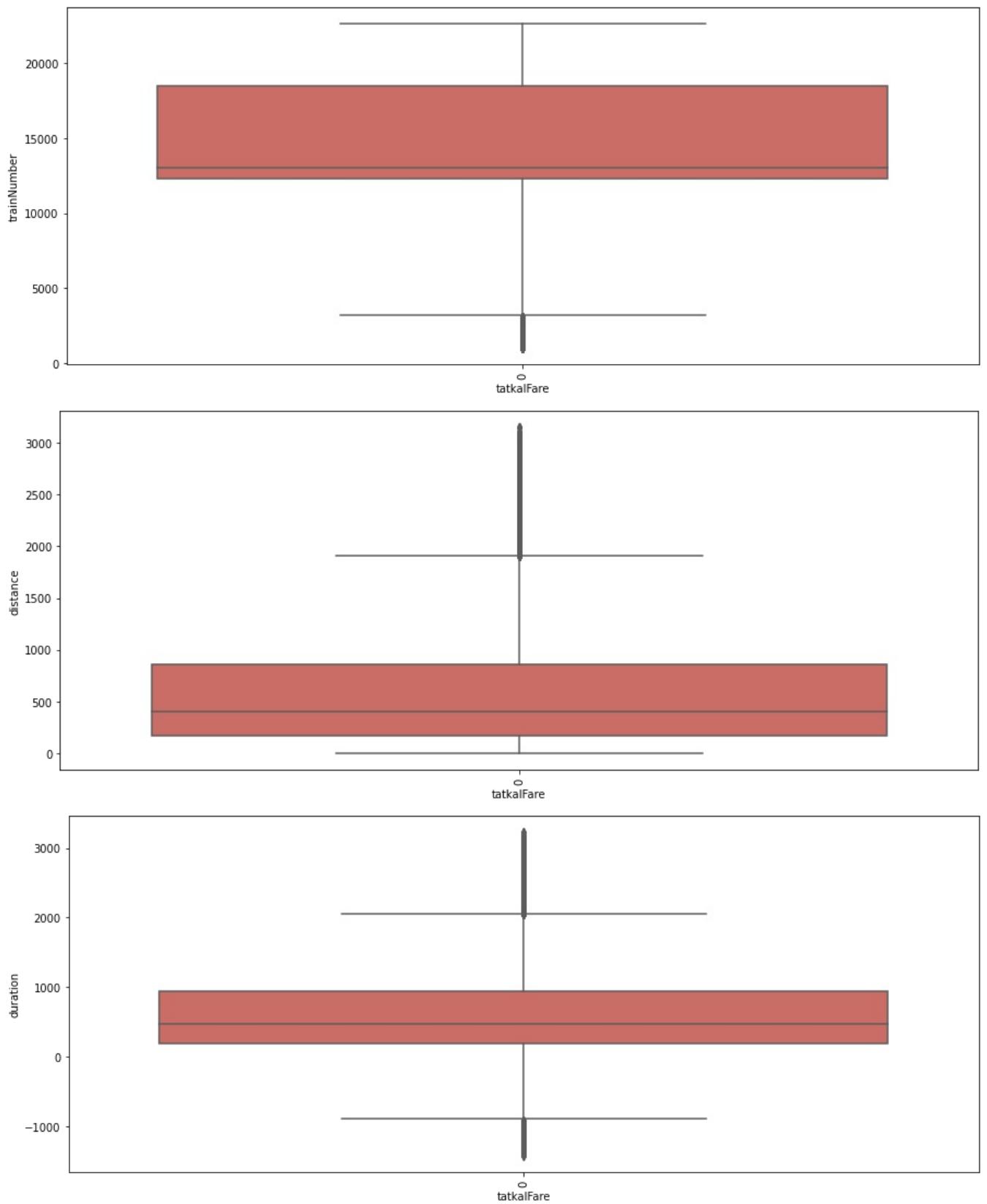


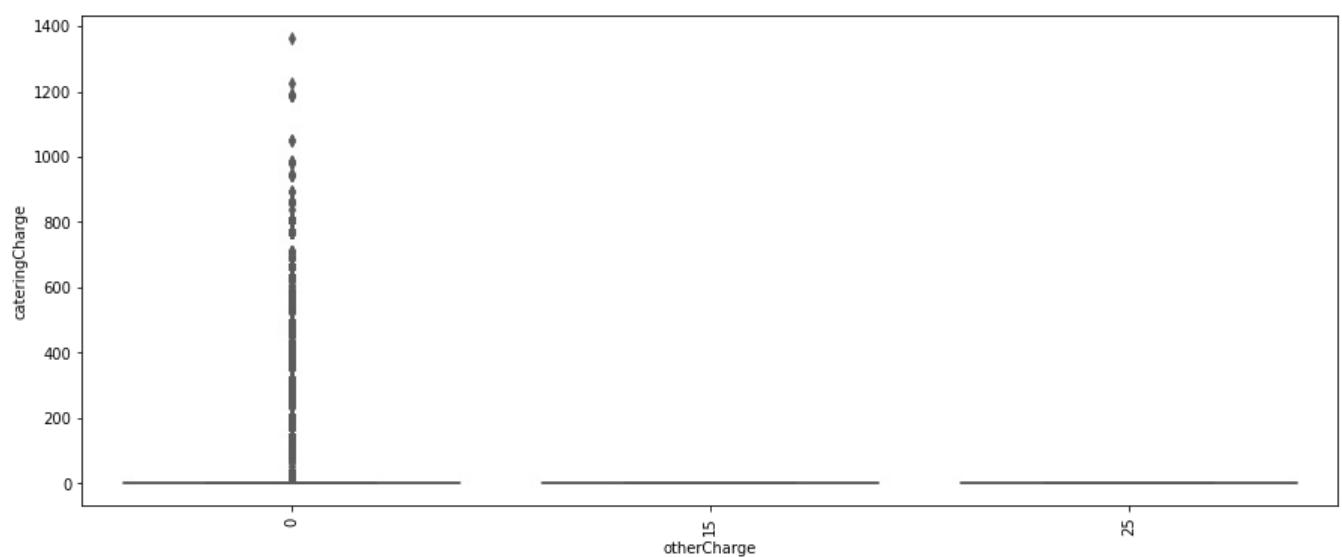
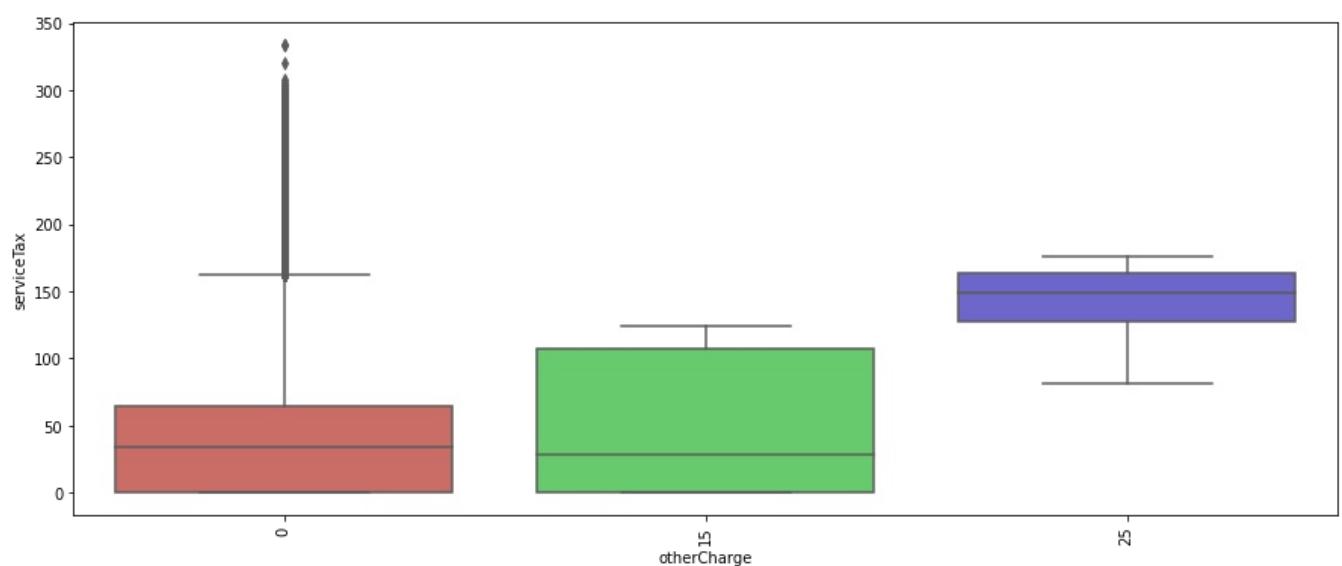
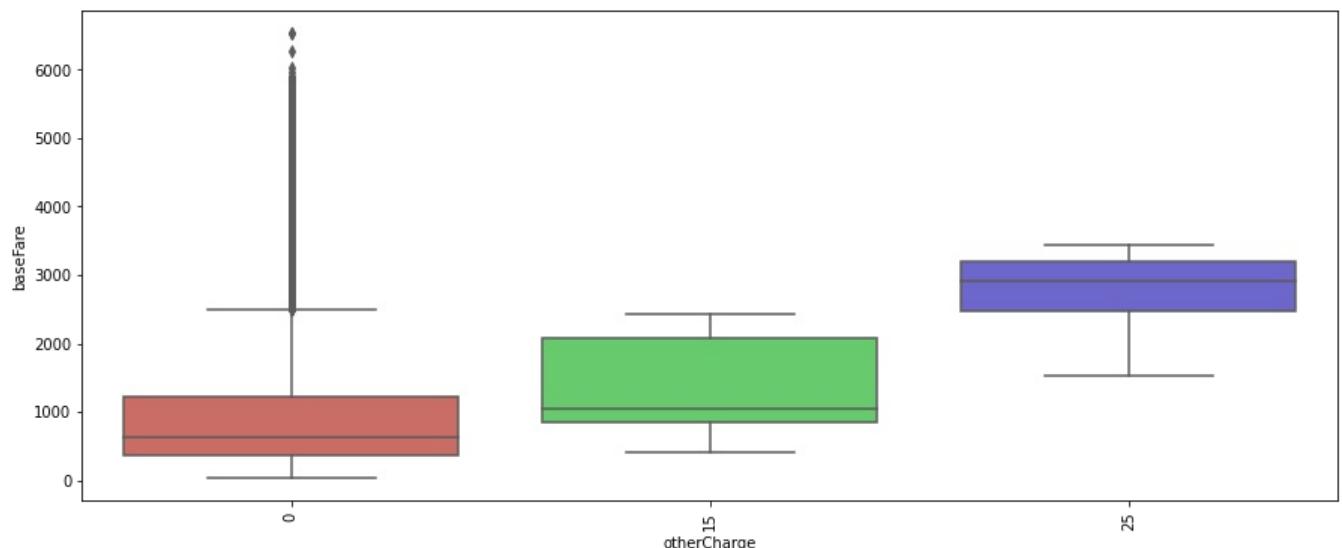


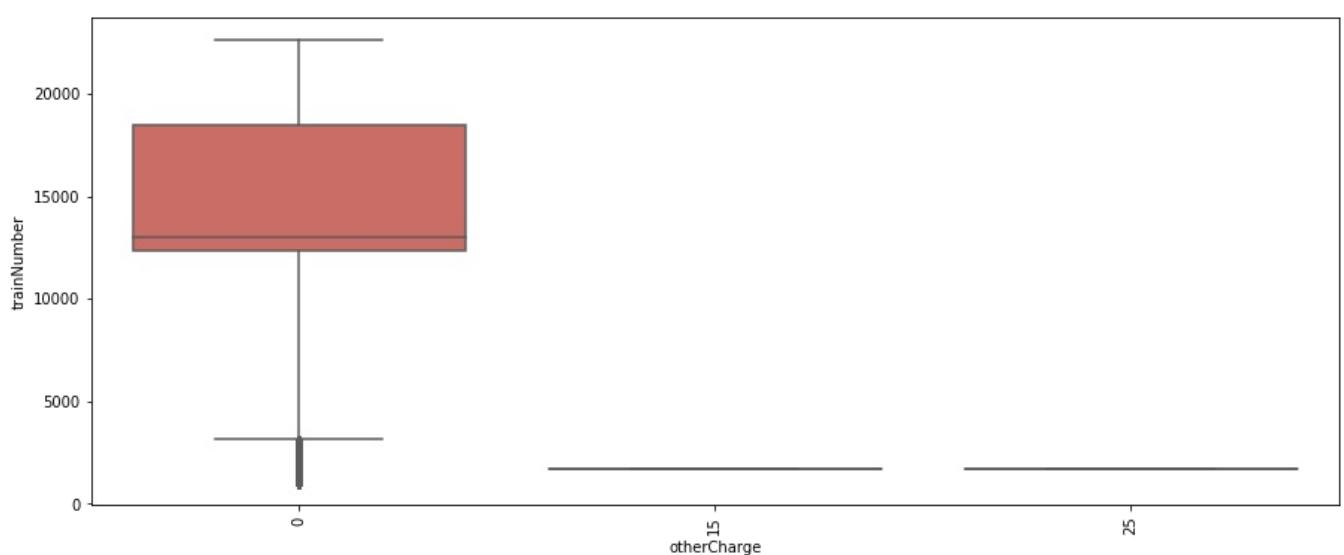
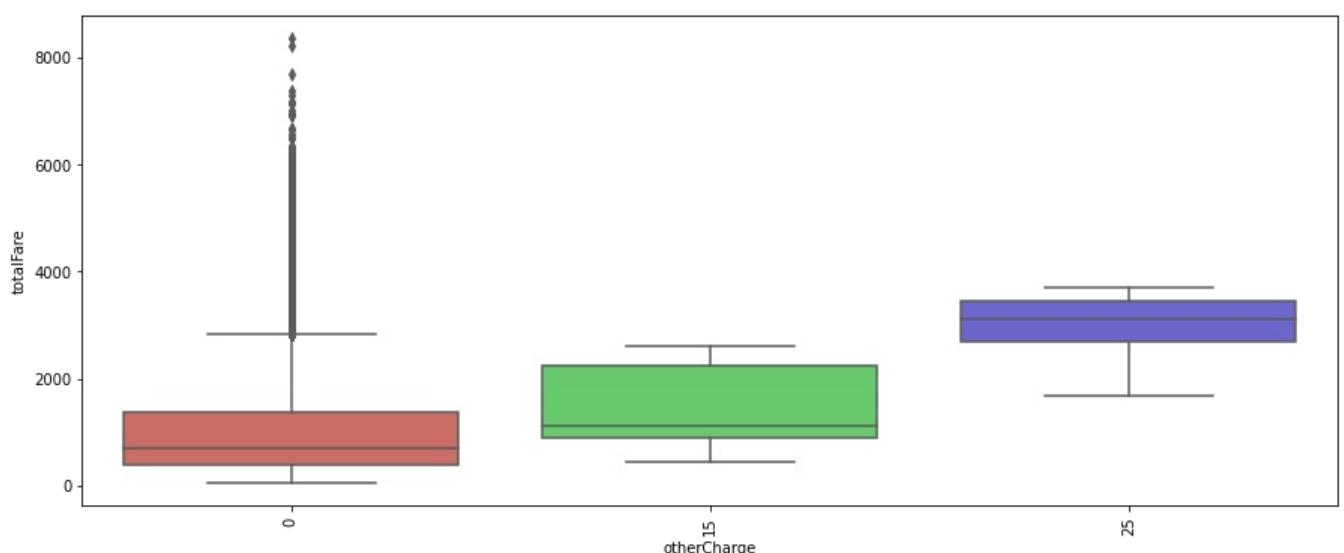
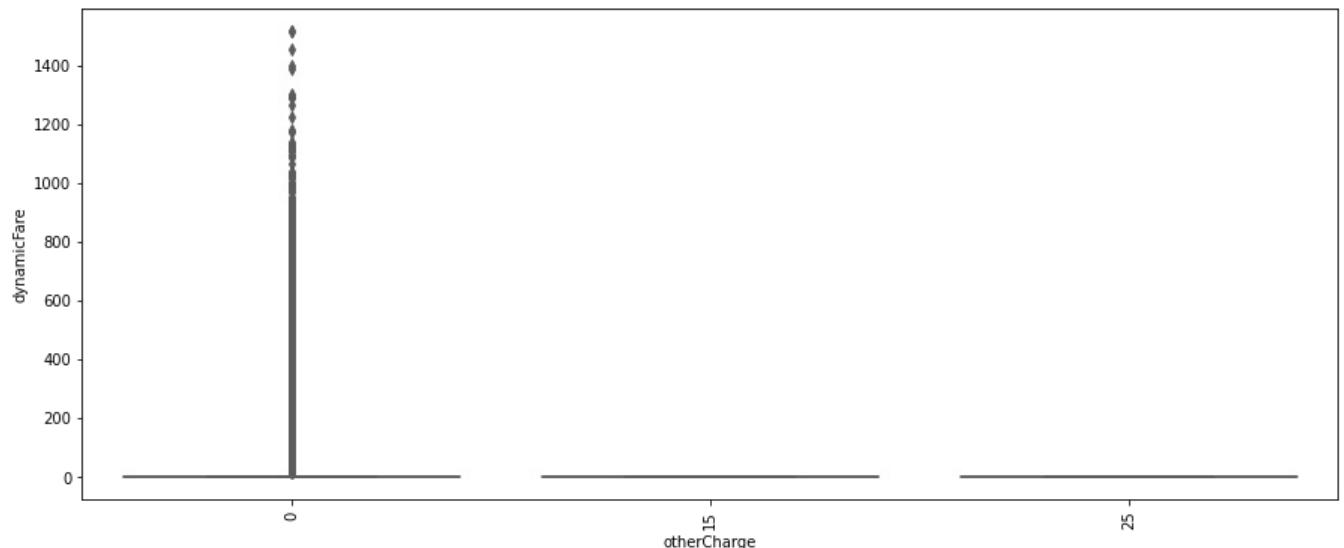


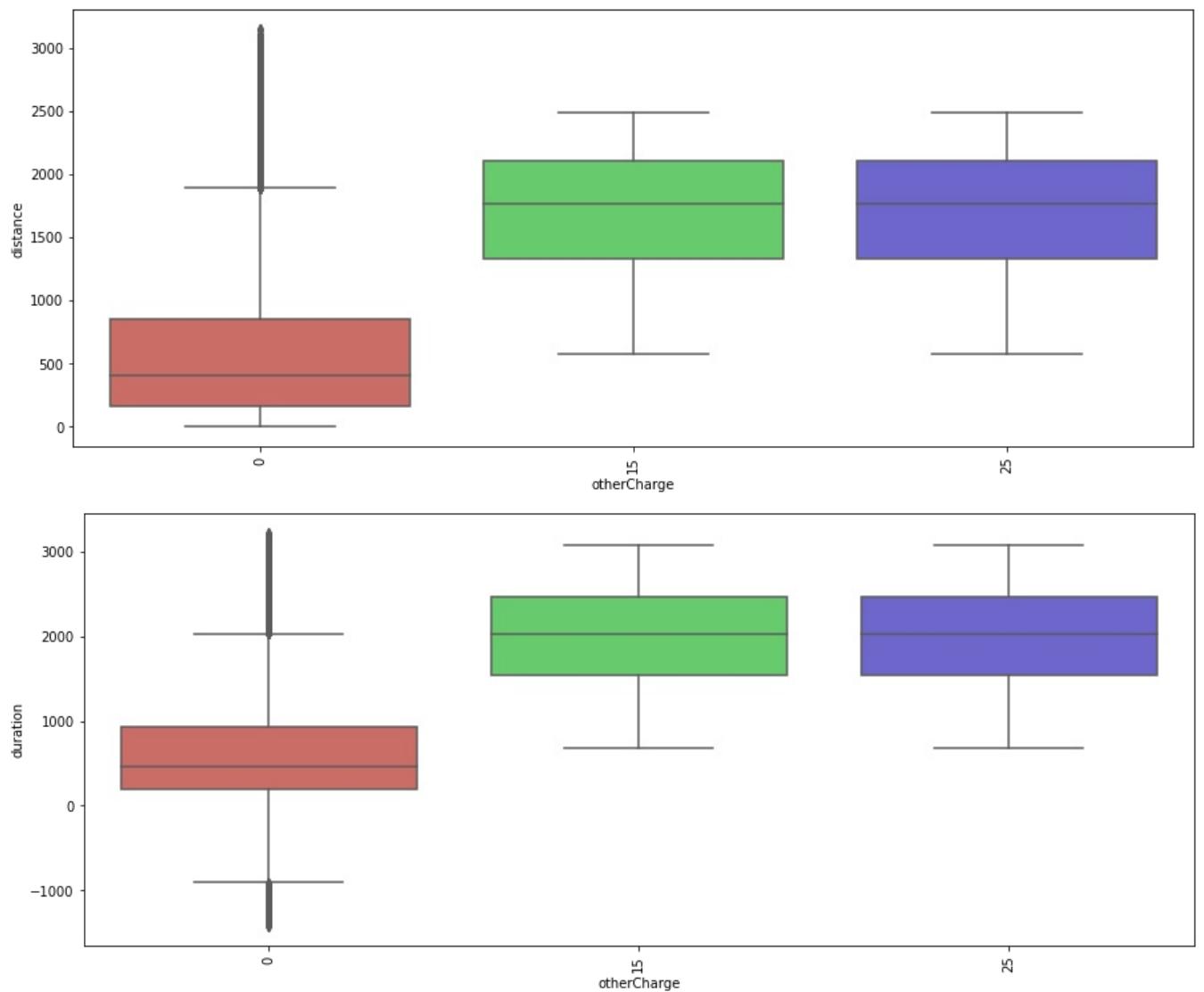




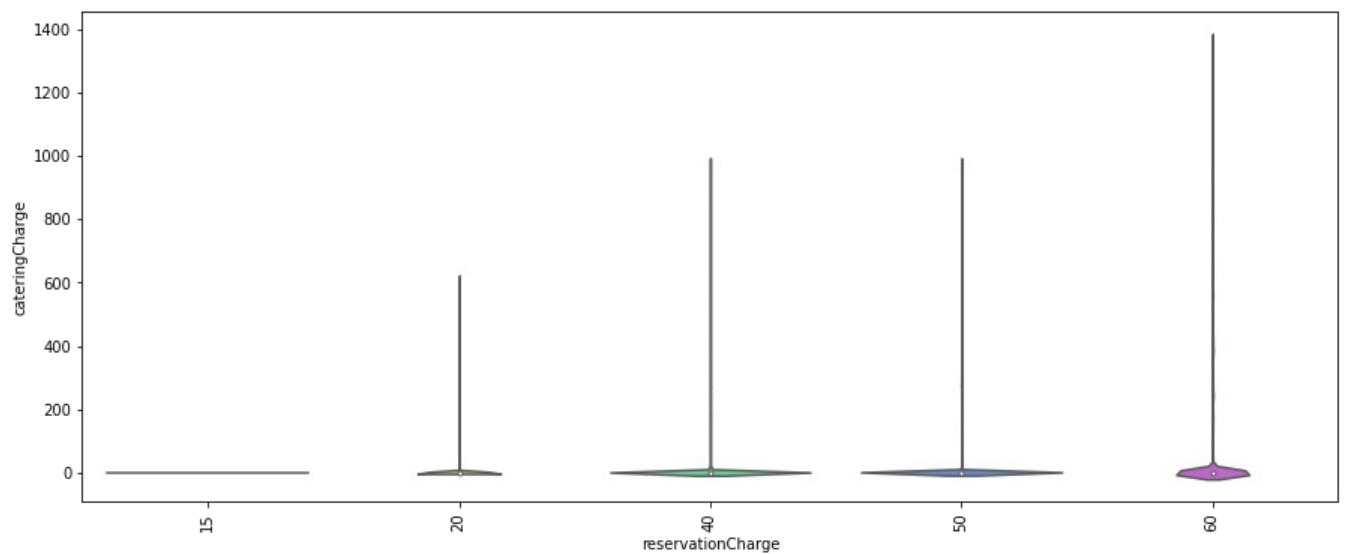
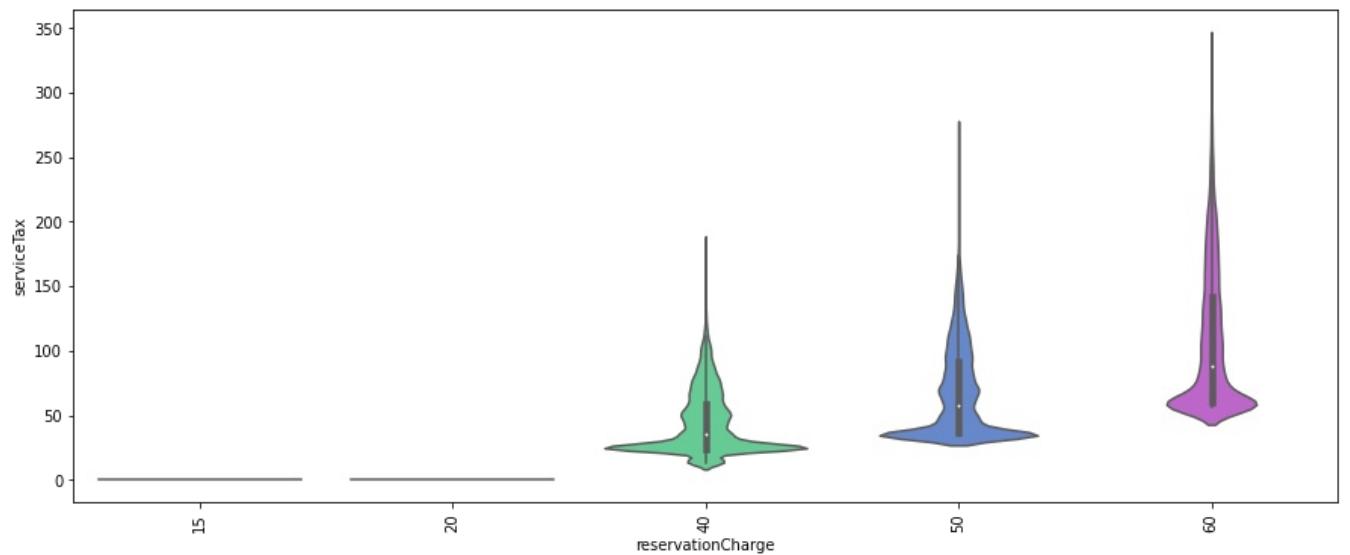
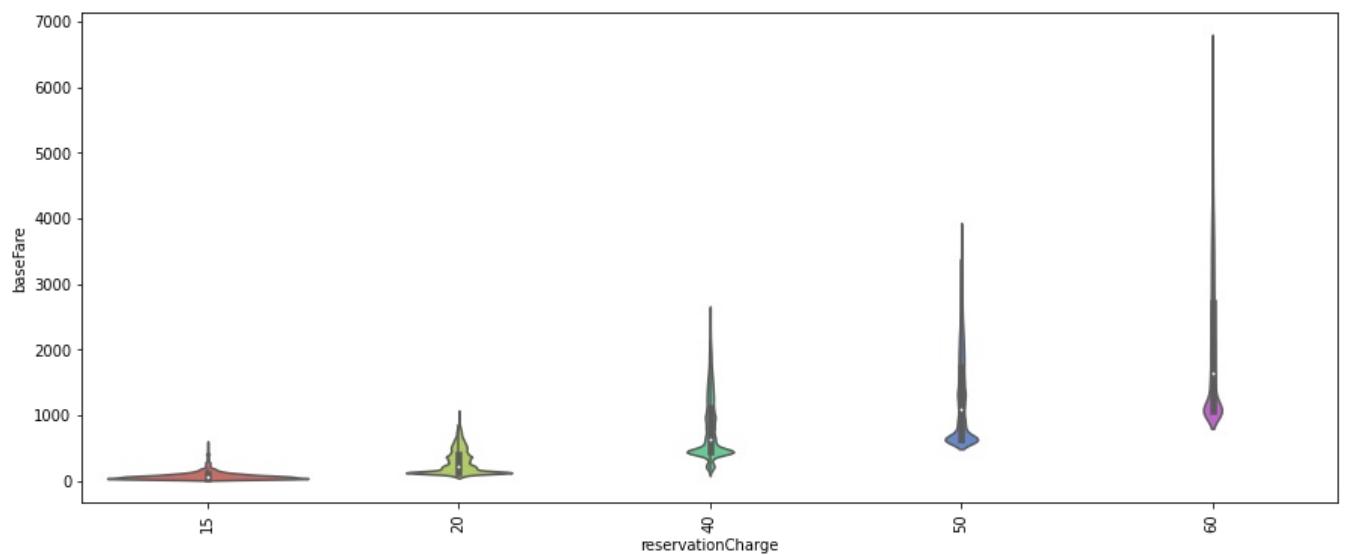


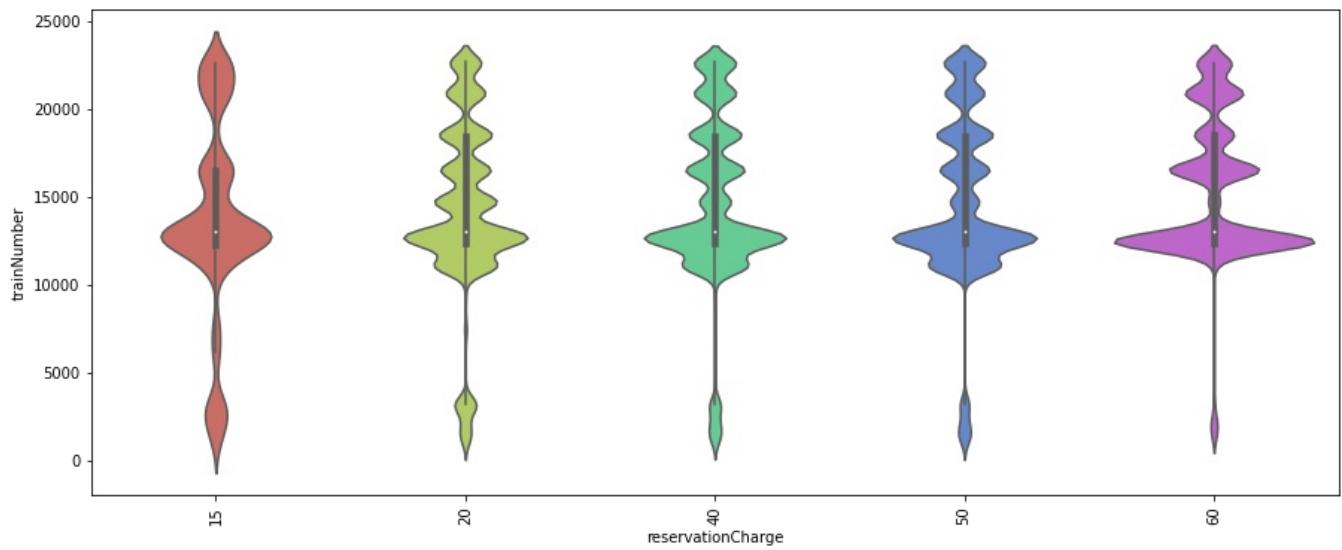
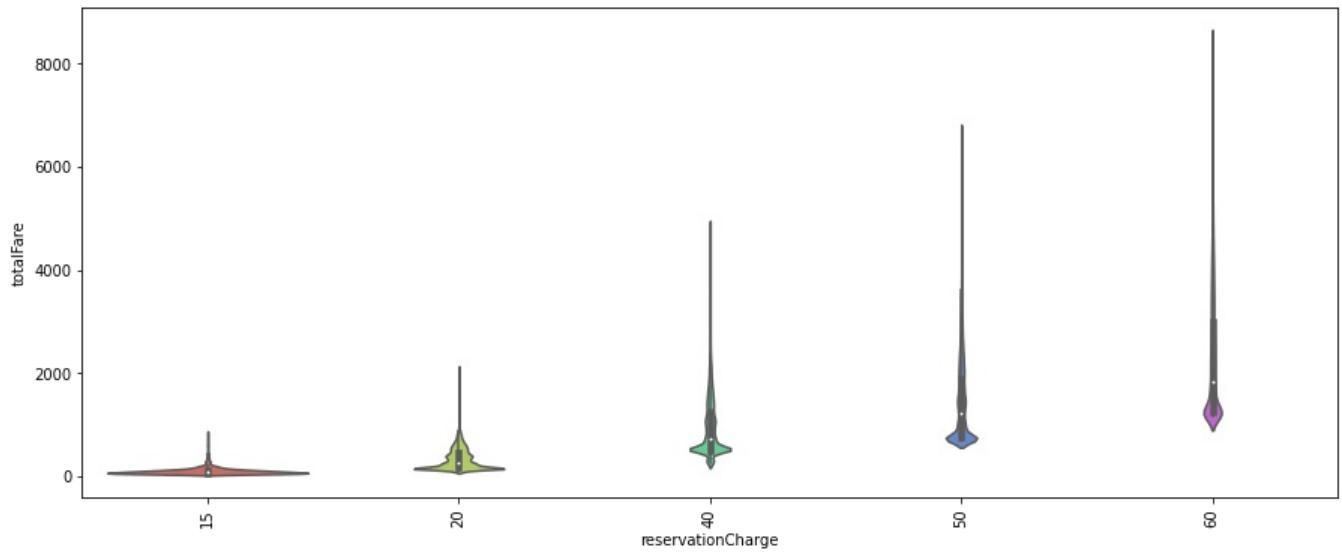
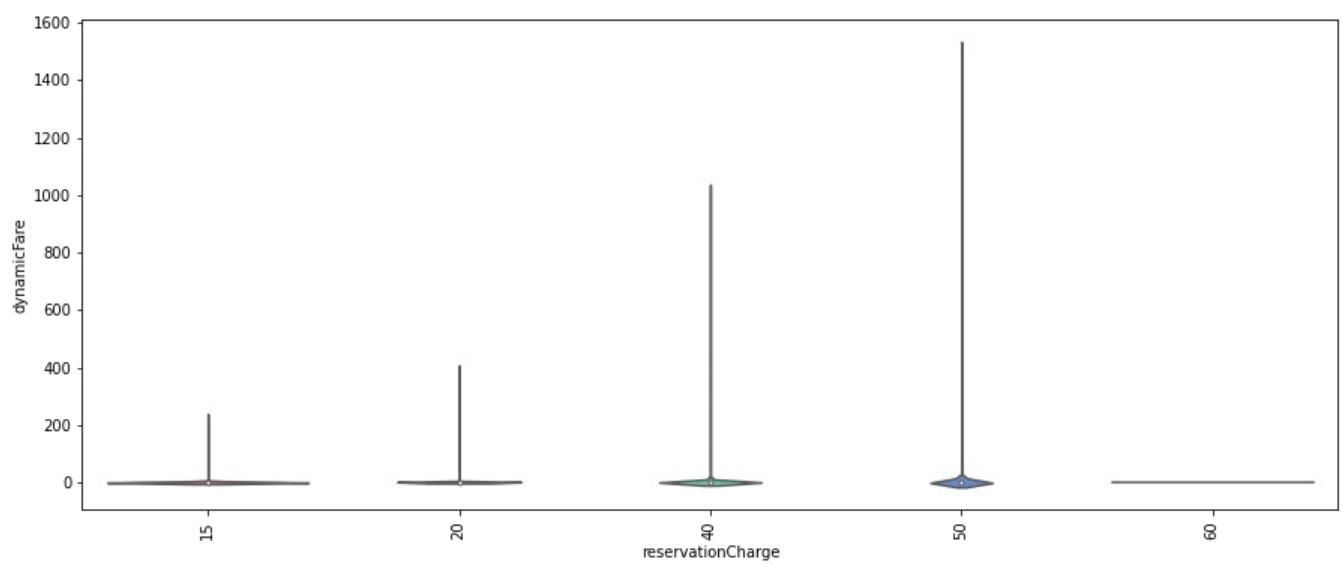


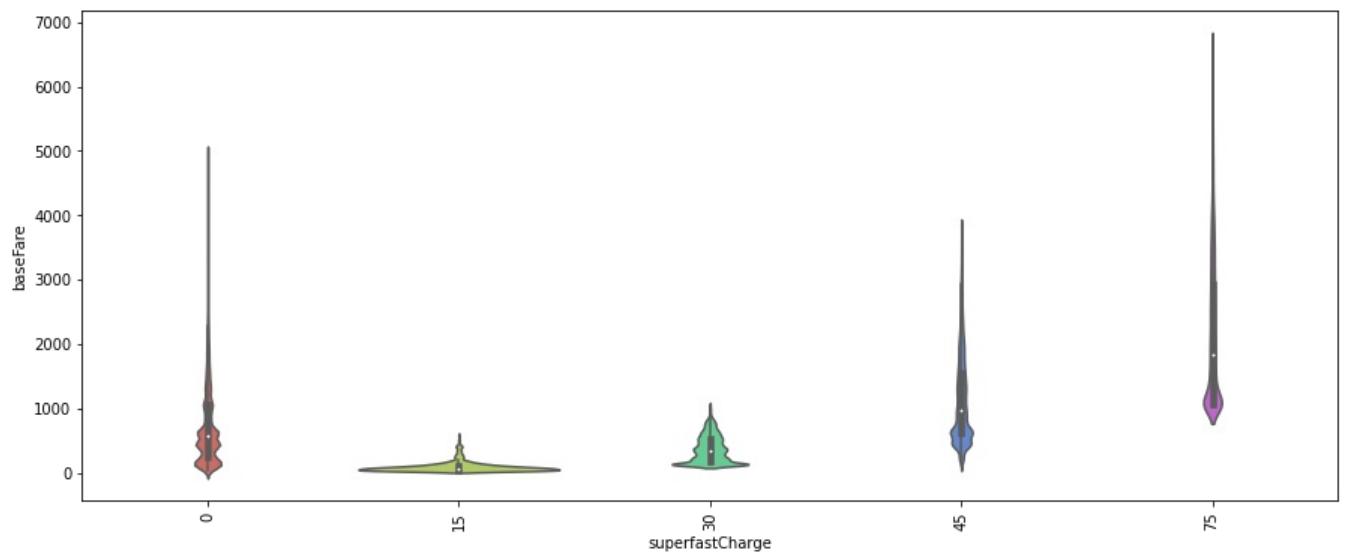
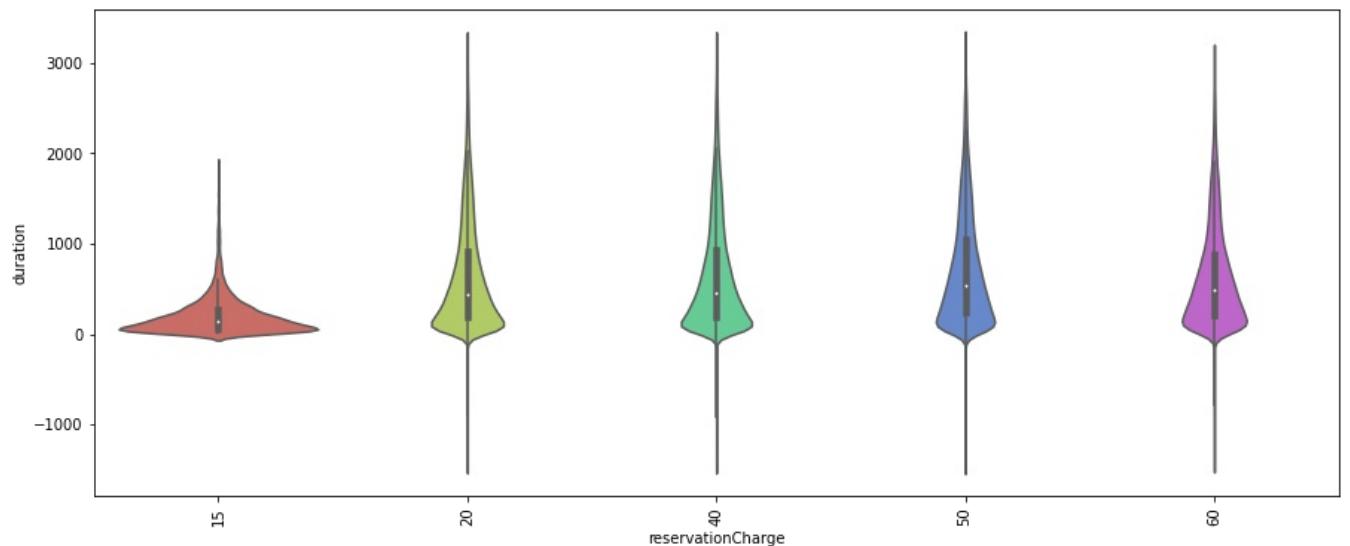
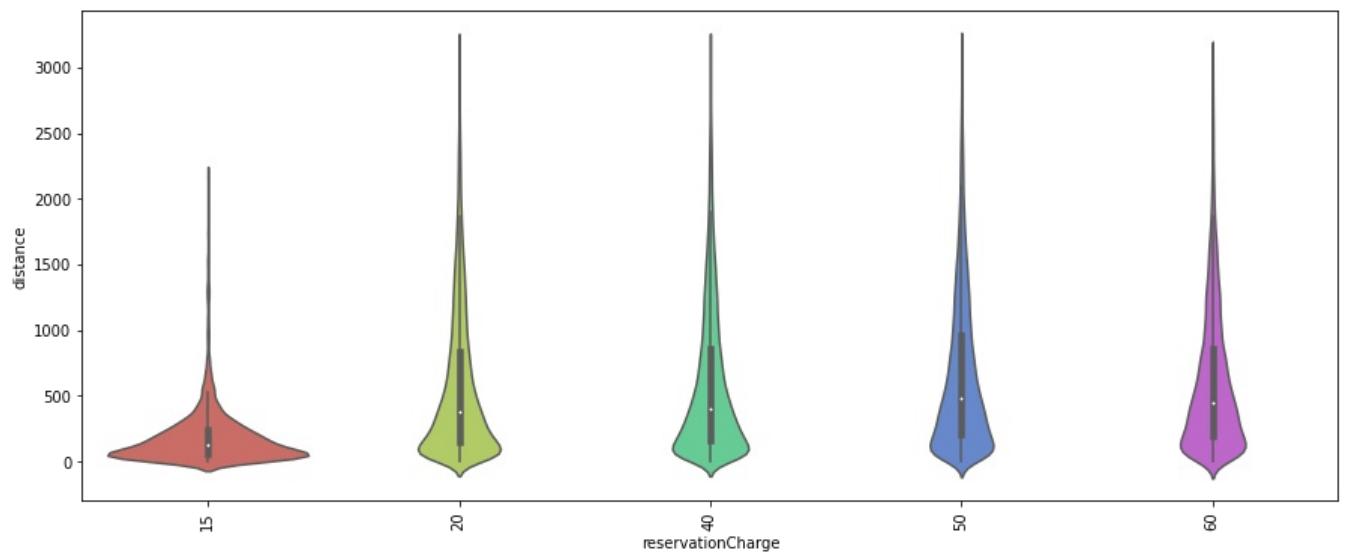


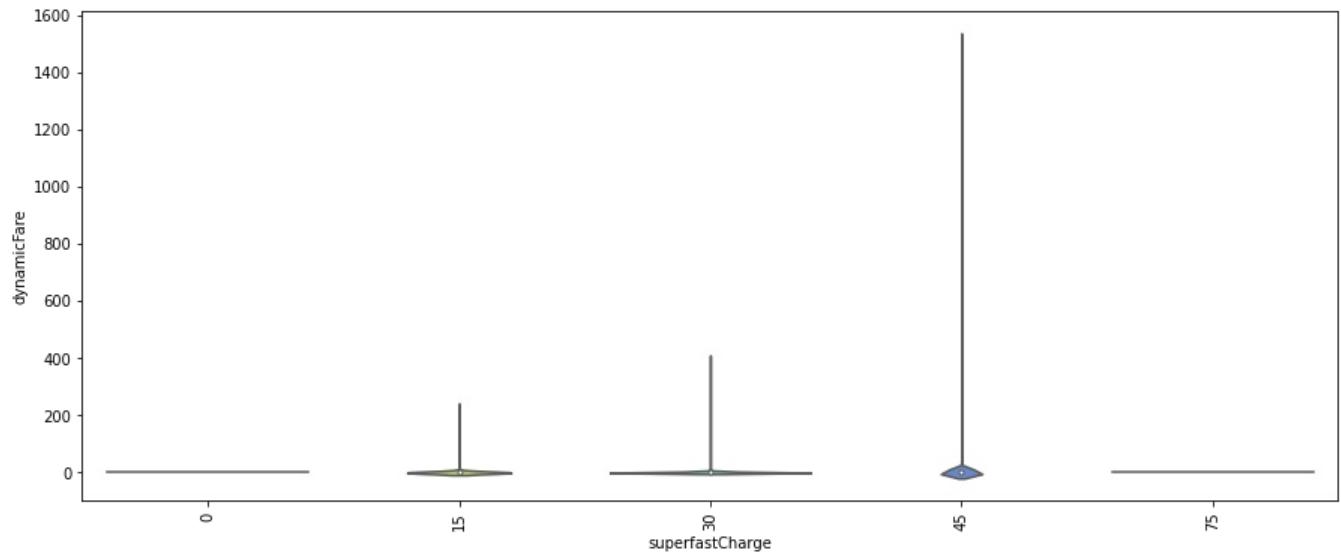
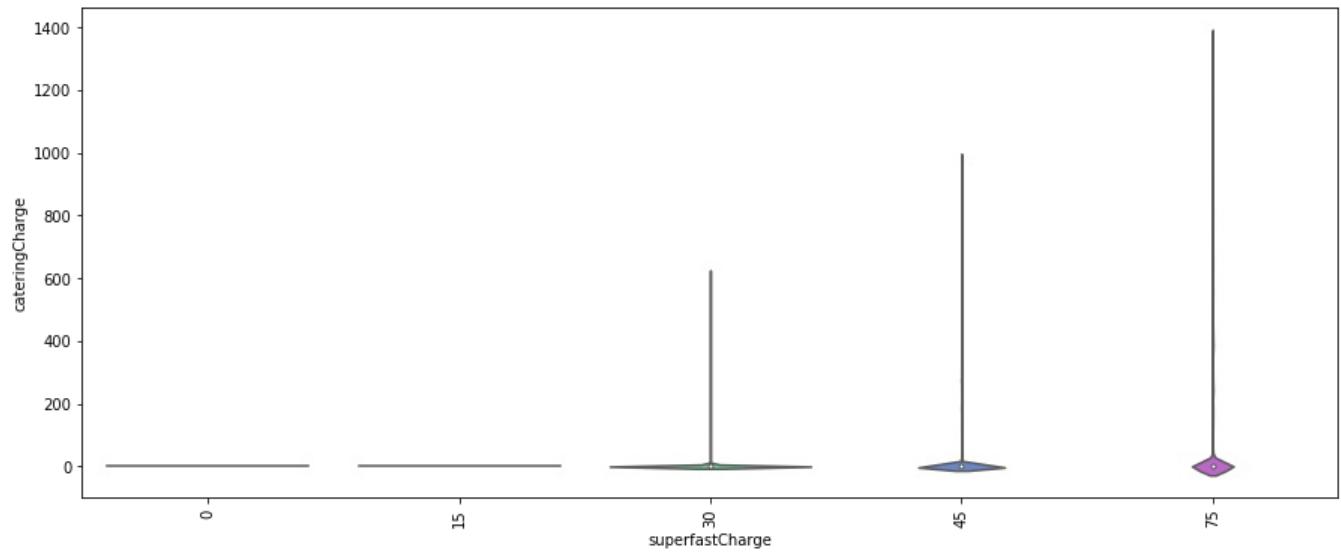
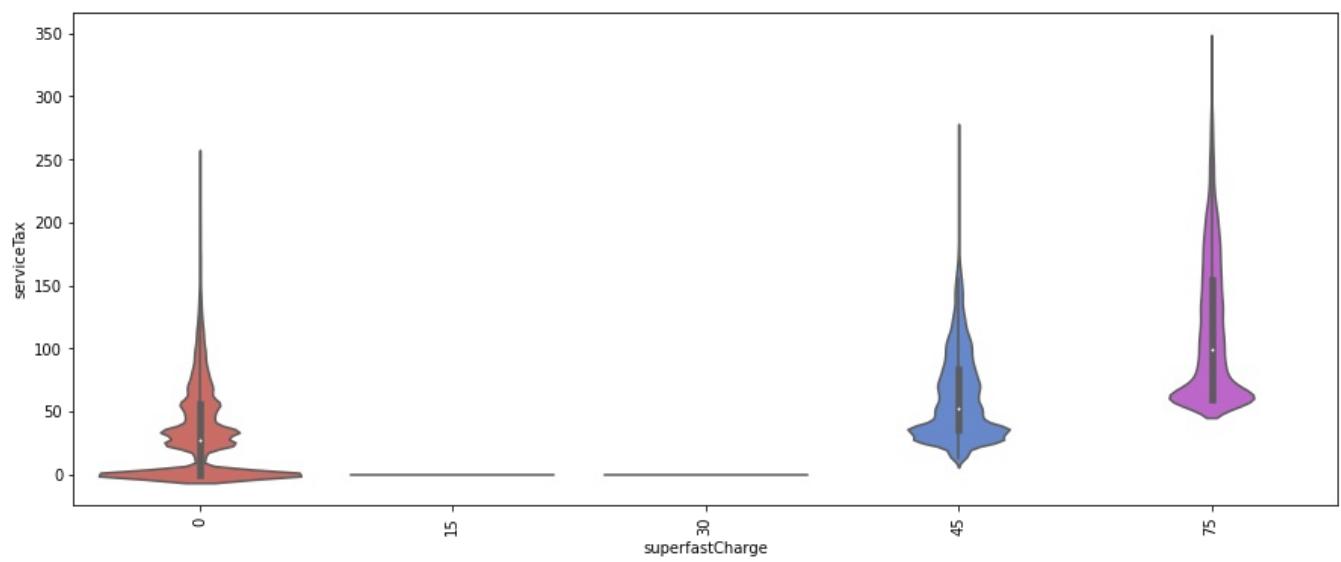


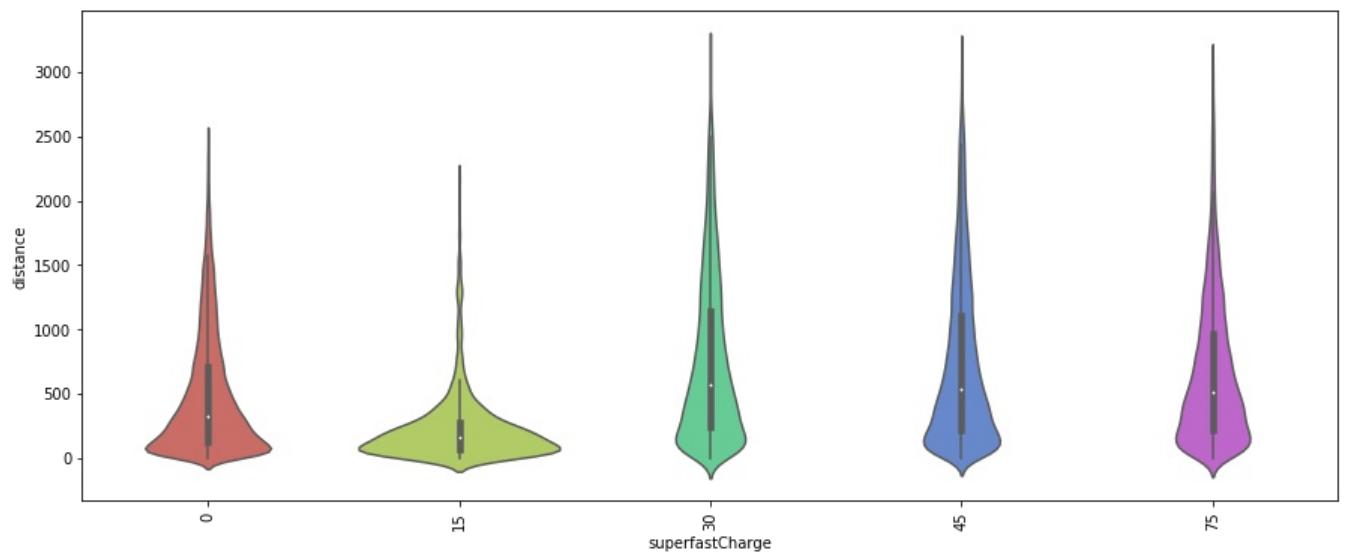
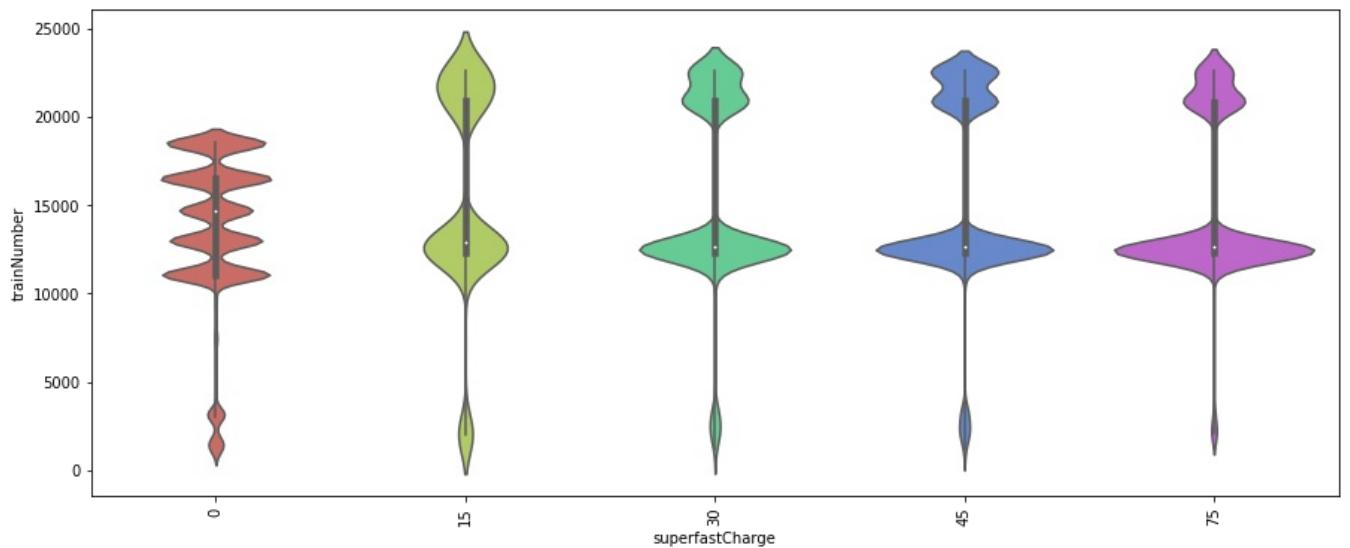
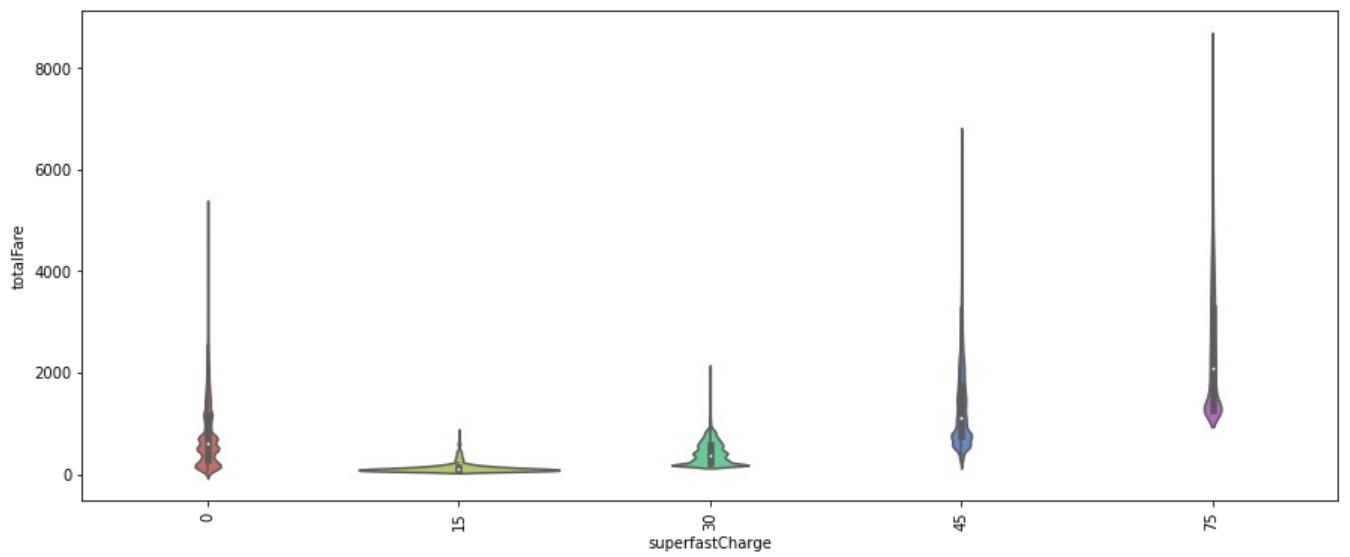
```
In [36]: for i in discrete:
    for j in continuous:
        plt.figure(figsize=(15,6))
        sns.violinplot(x = df[i], y = df[j], data = df, palette = 'hls')
        plt.xticks(rotation = 90)
        plt.show()
```

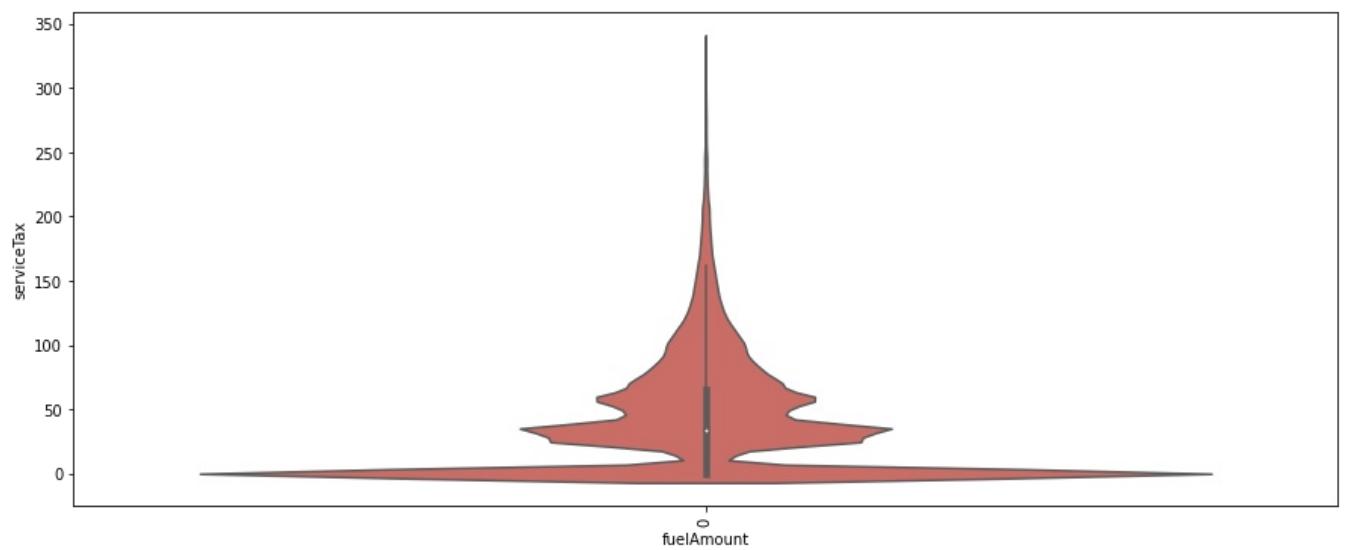
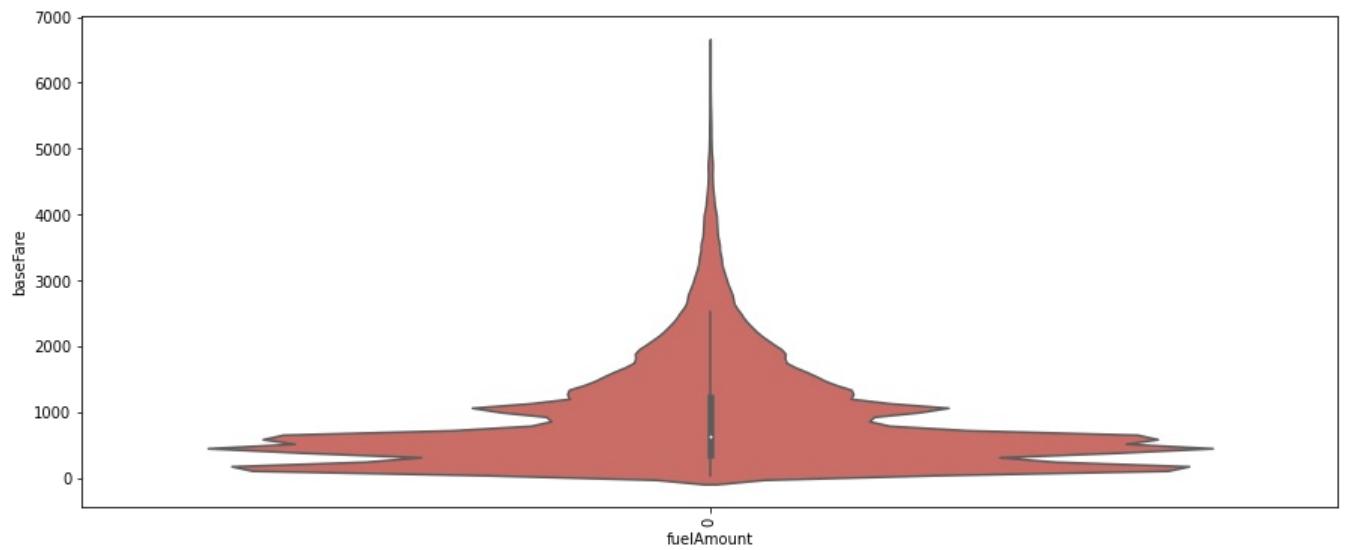
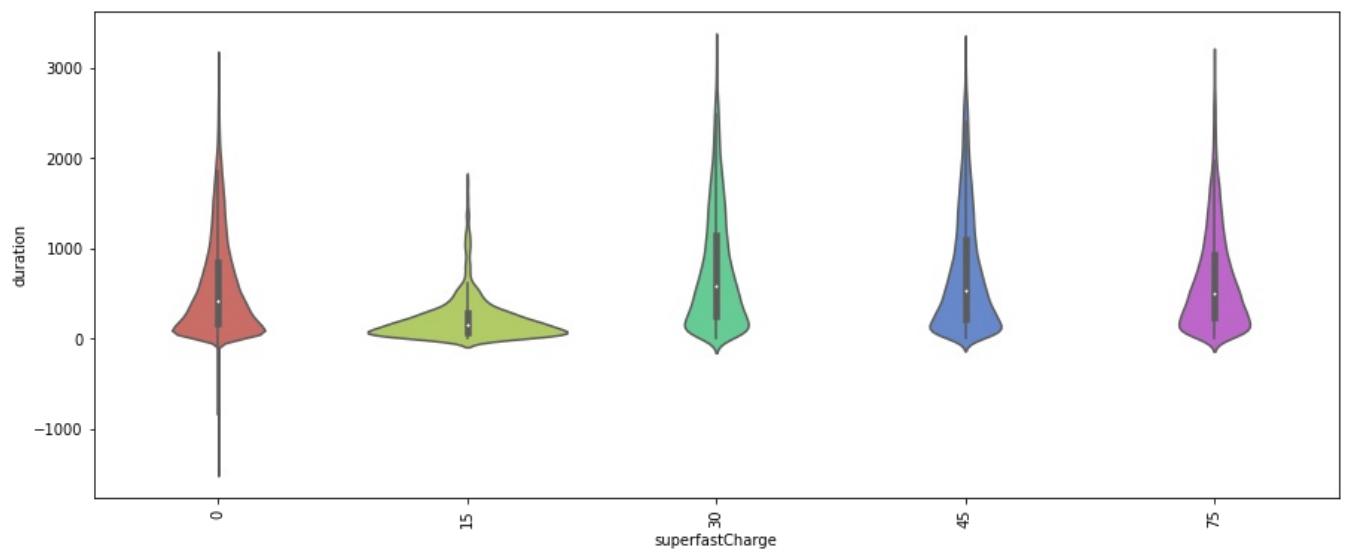


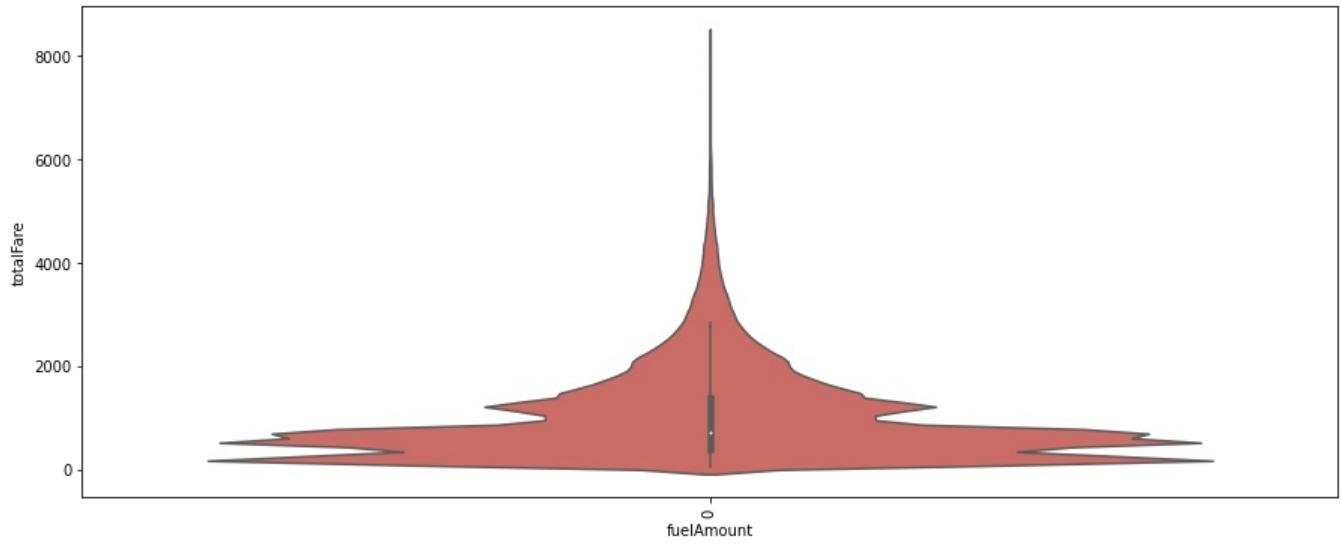
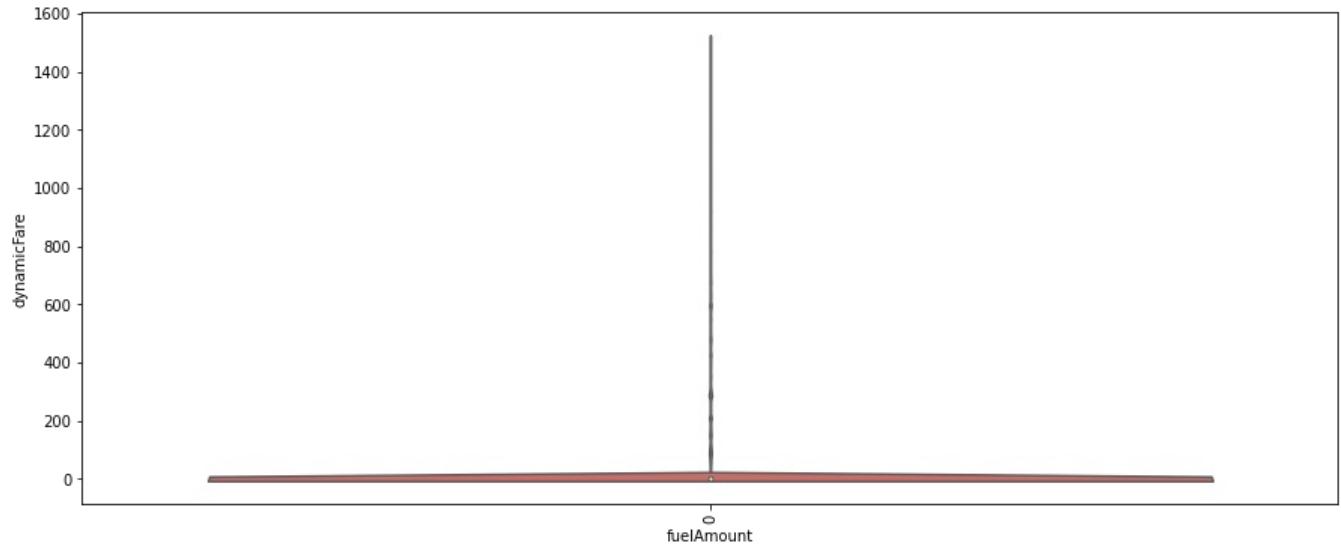
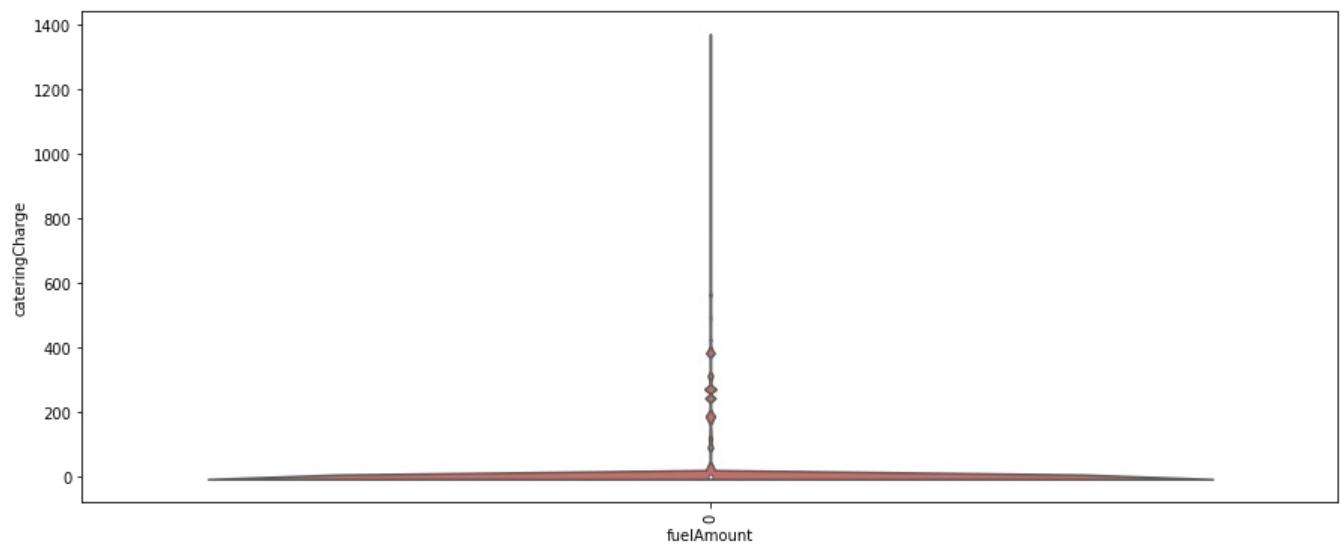


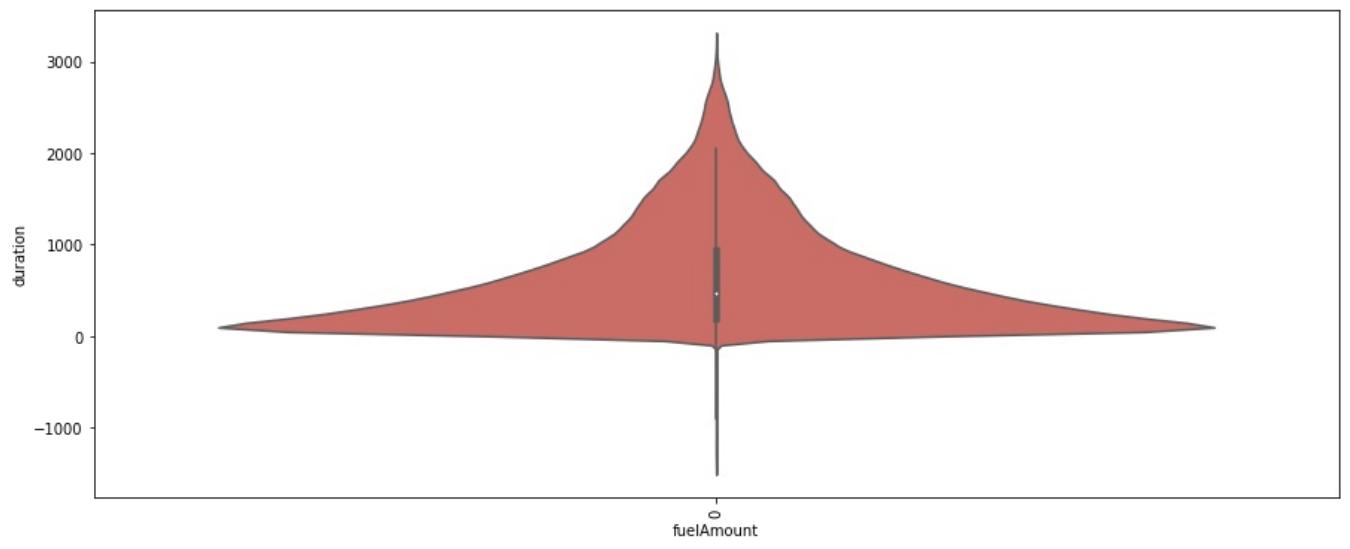
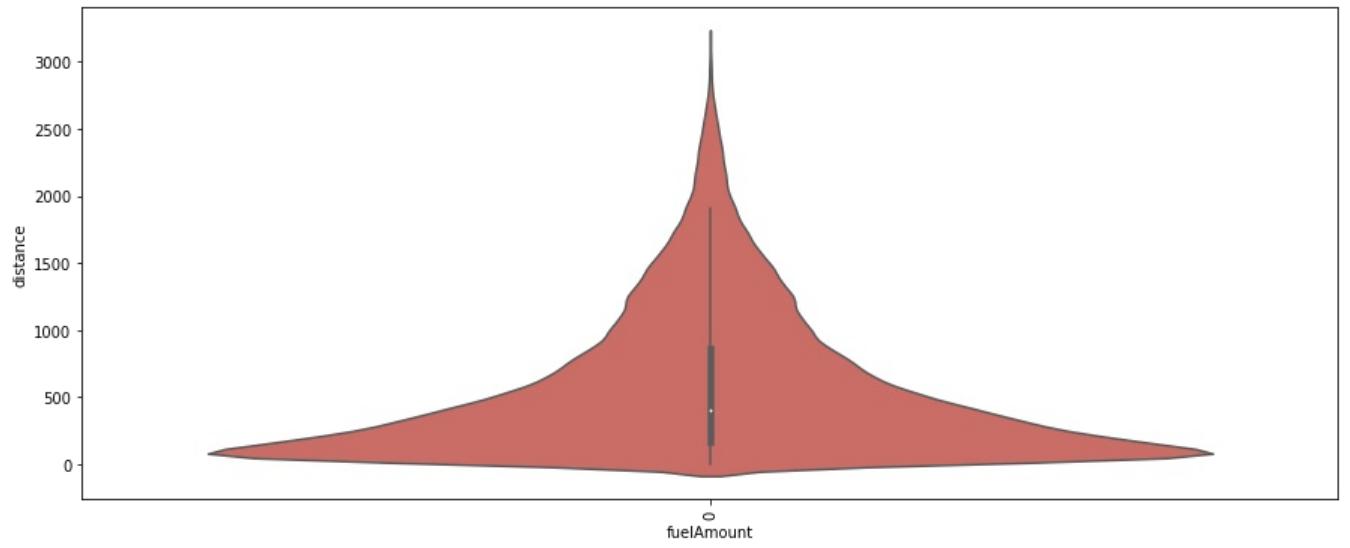
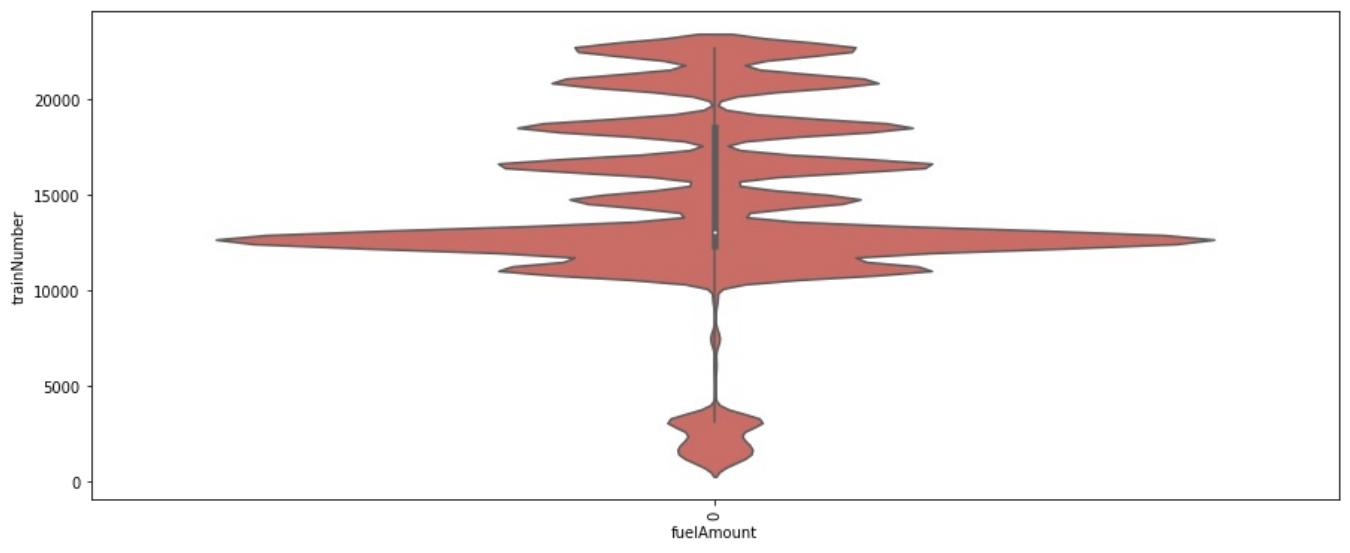


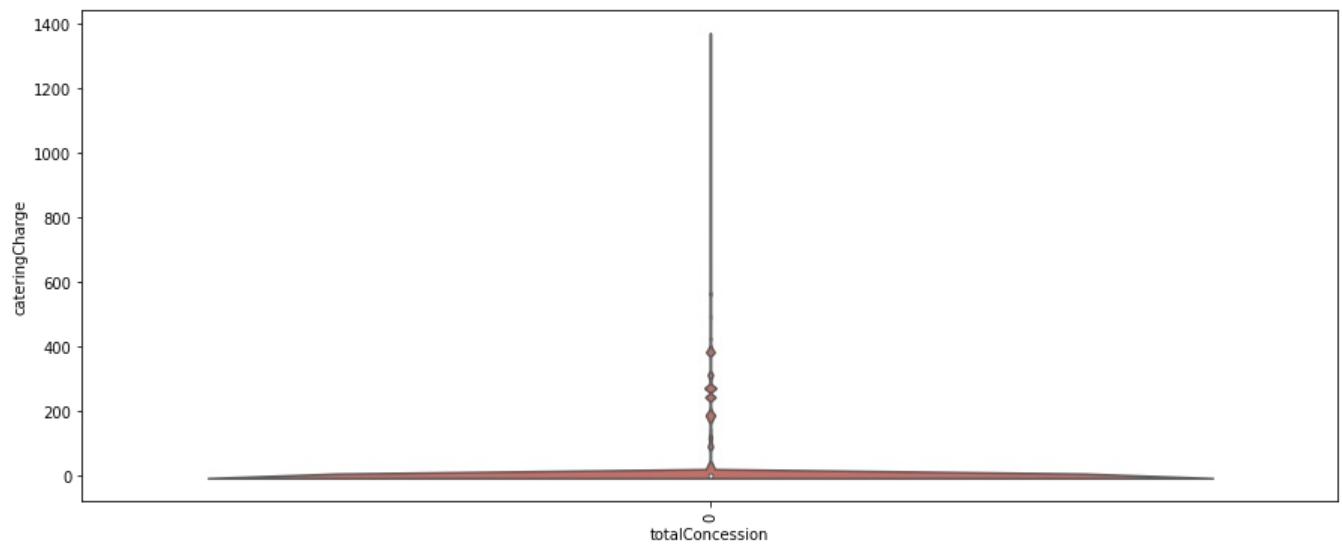
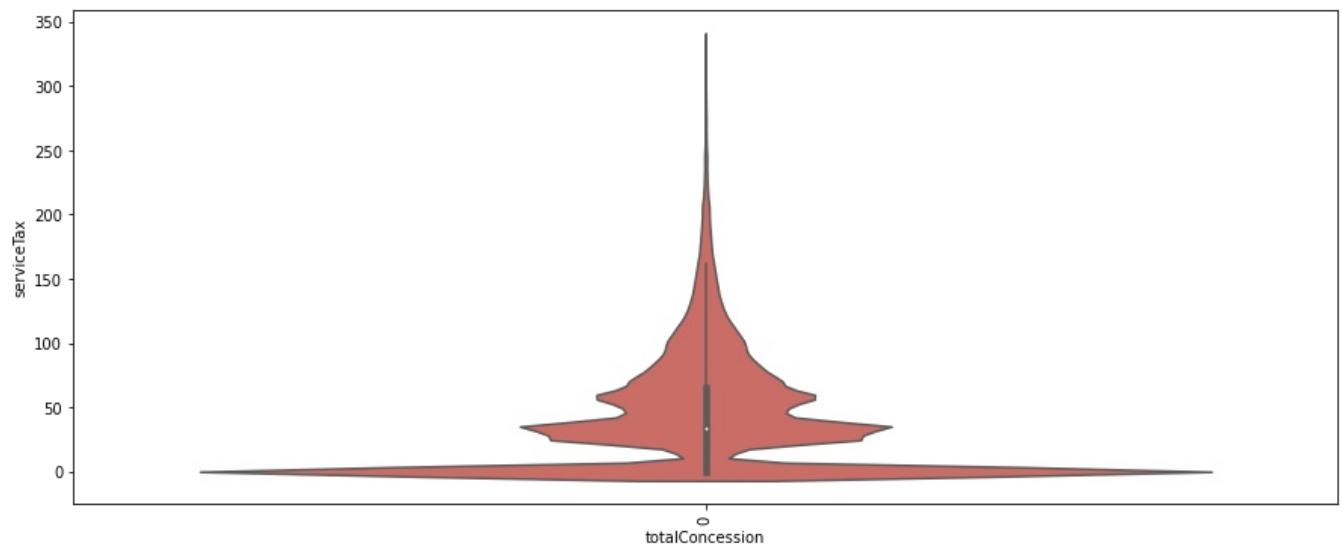
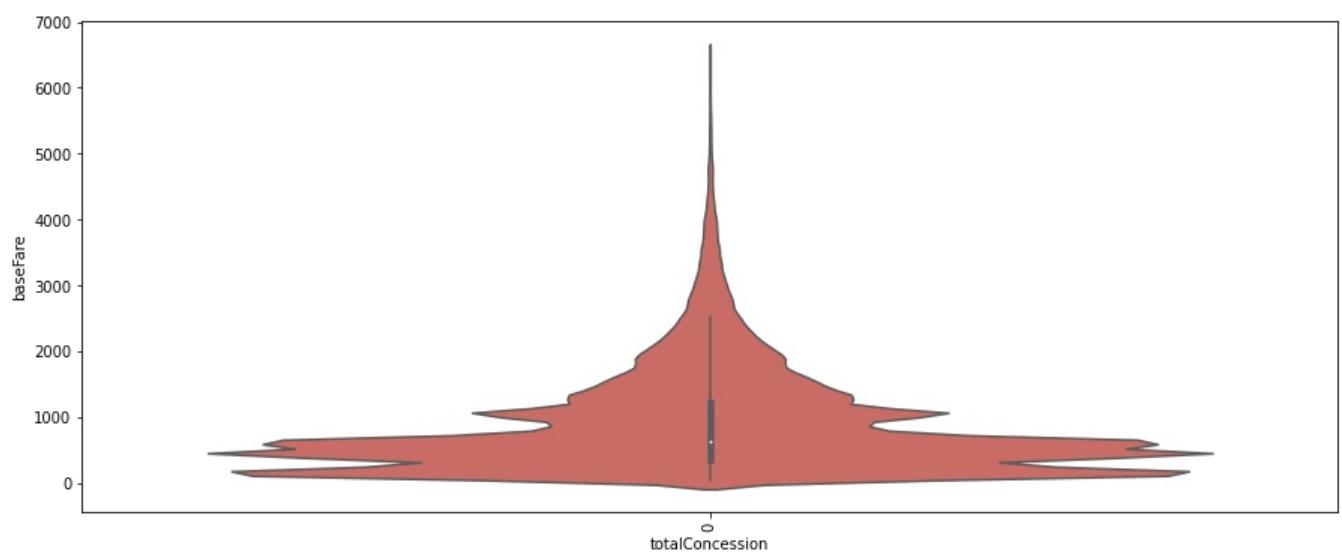


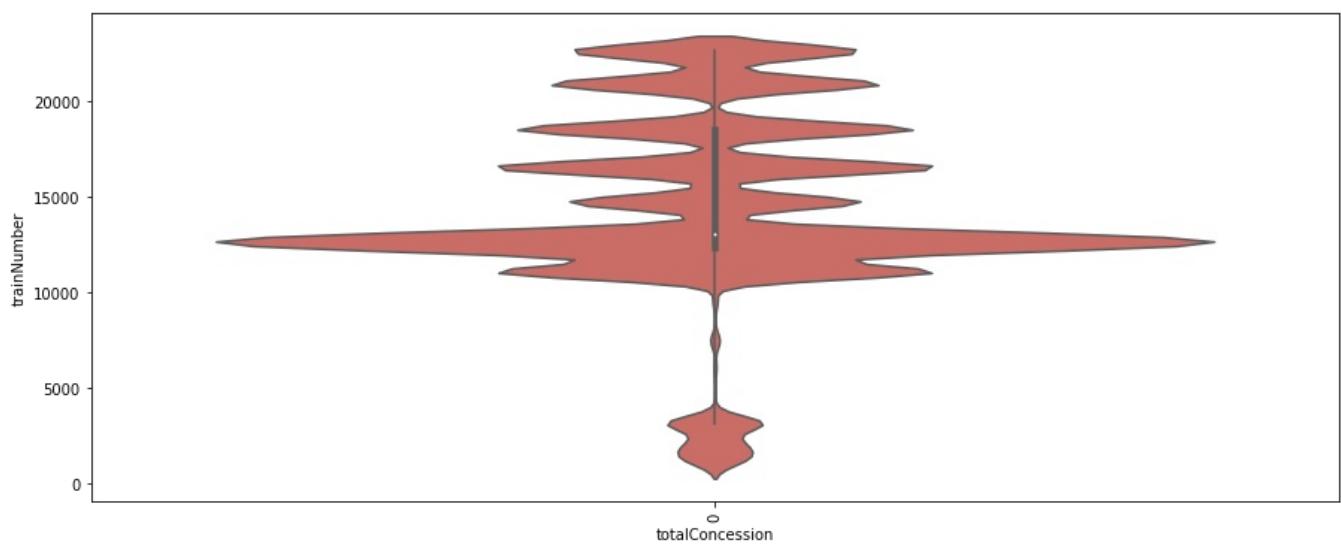
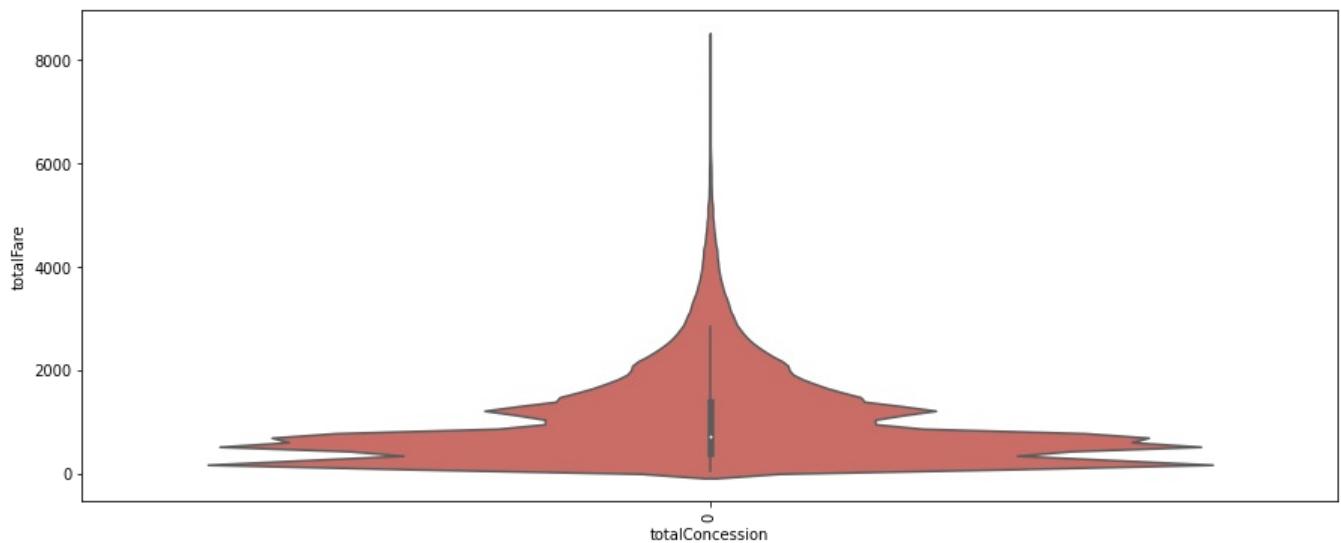
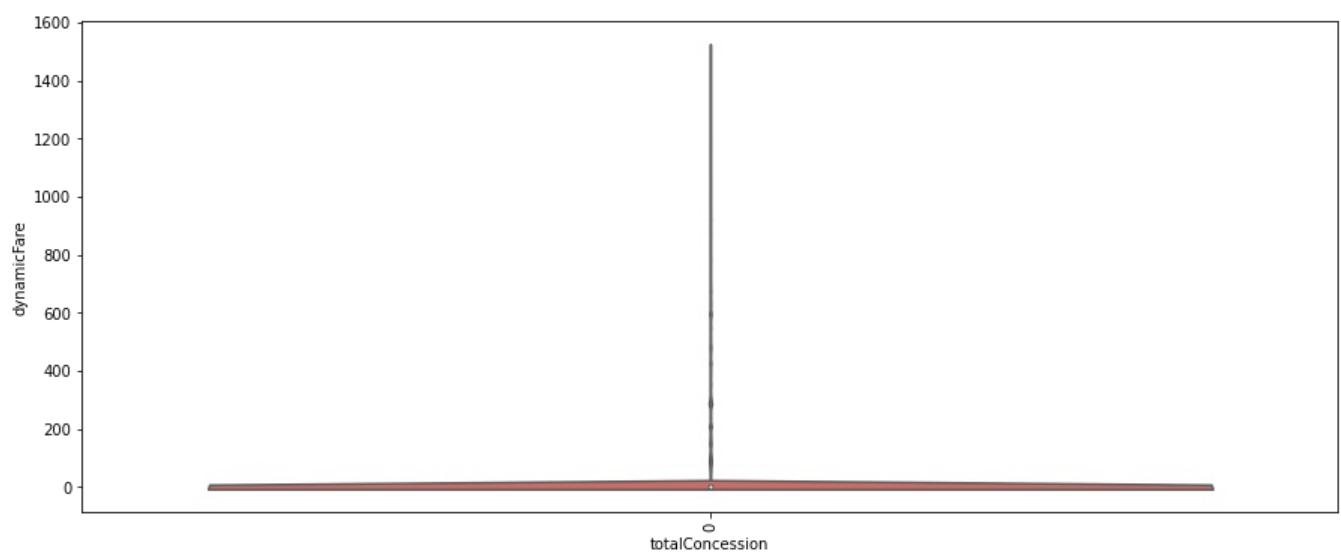


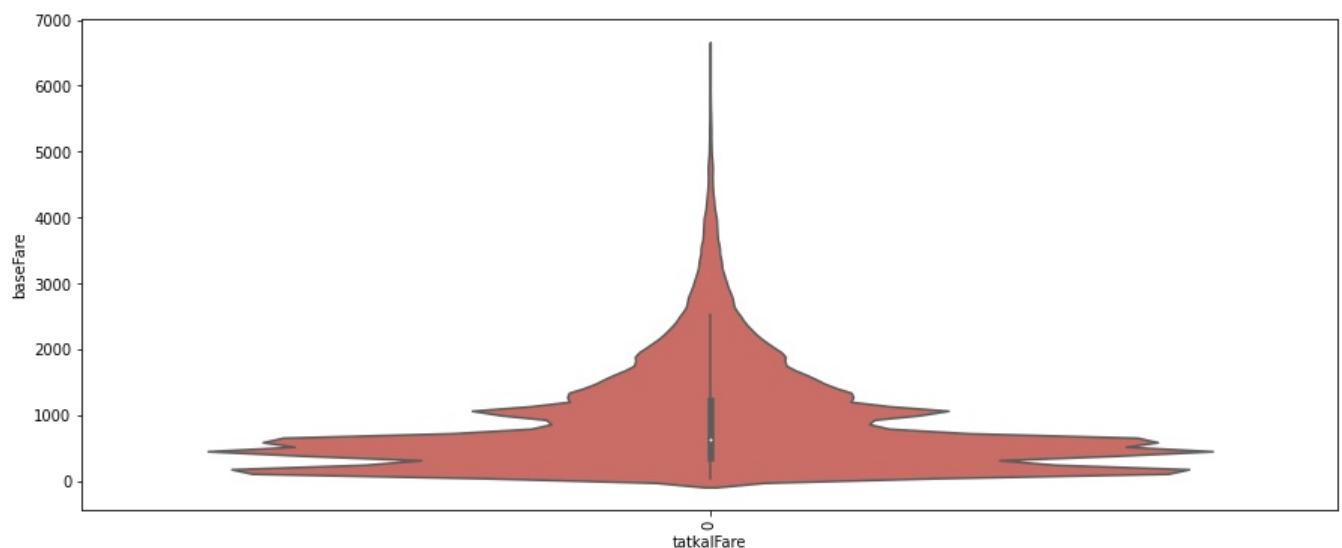
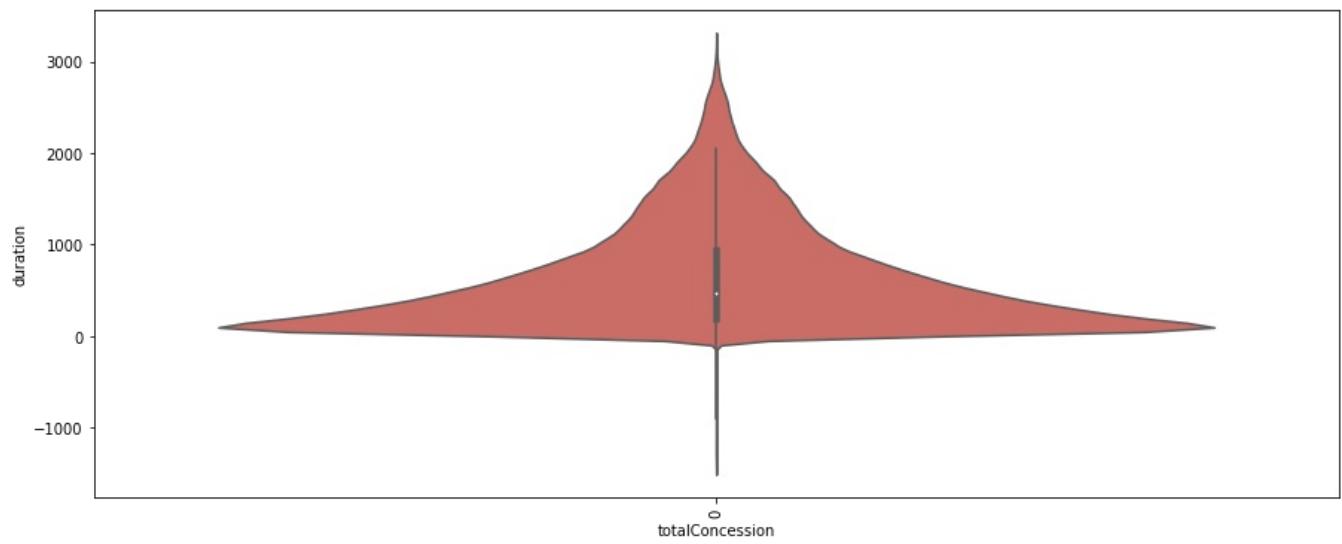
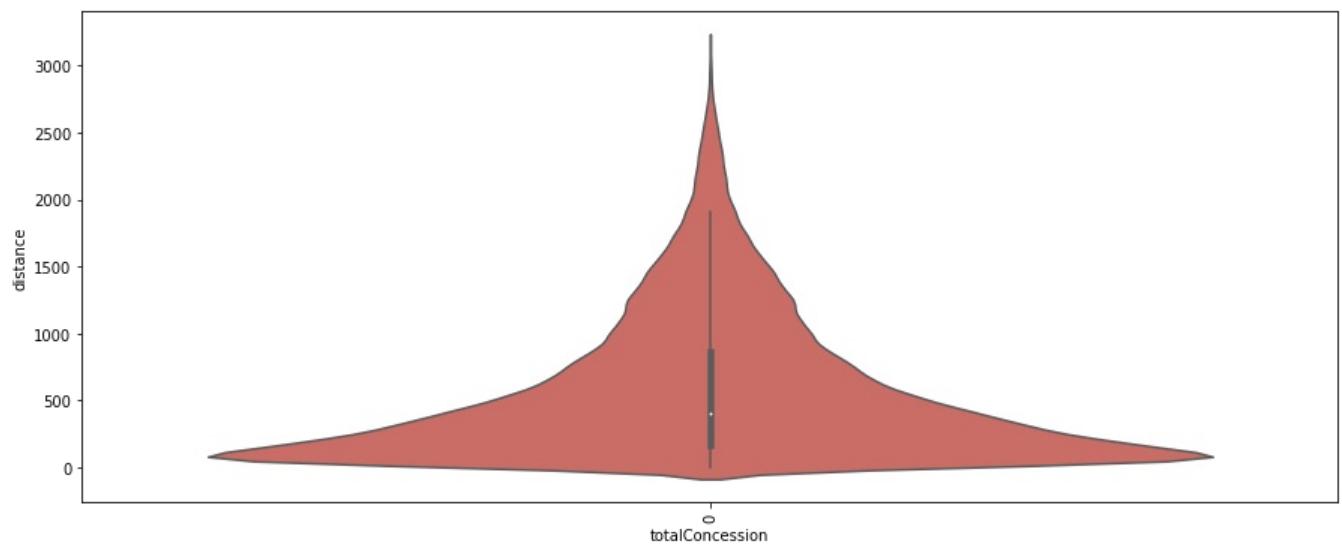


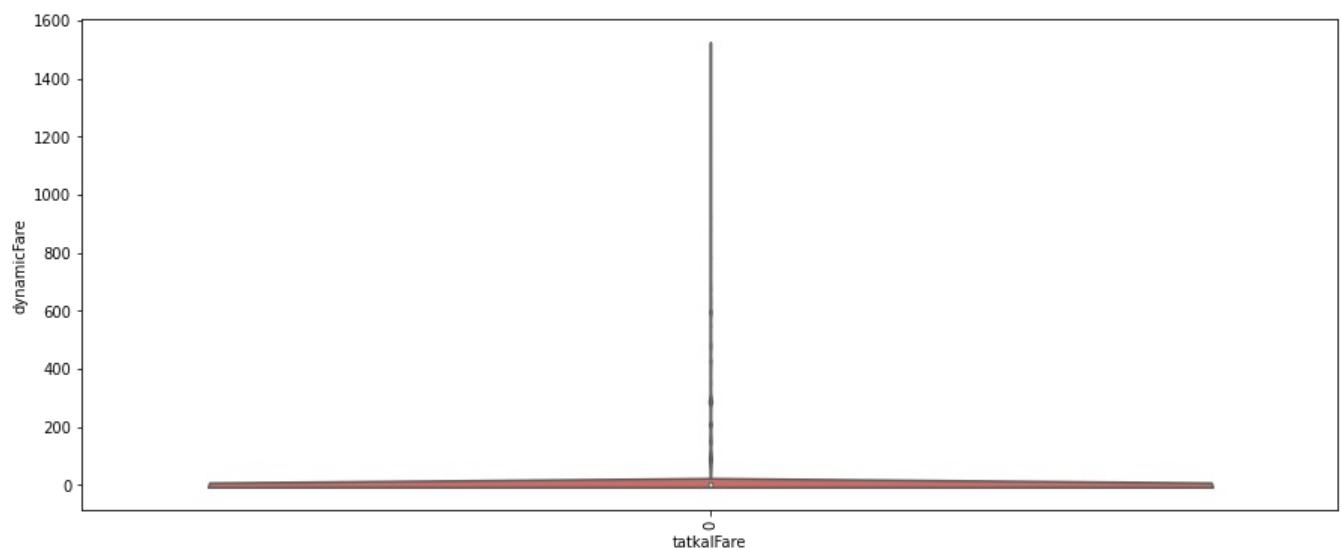
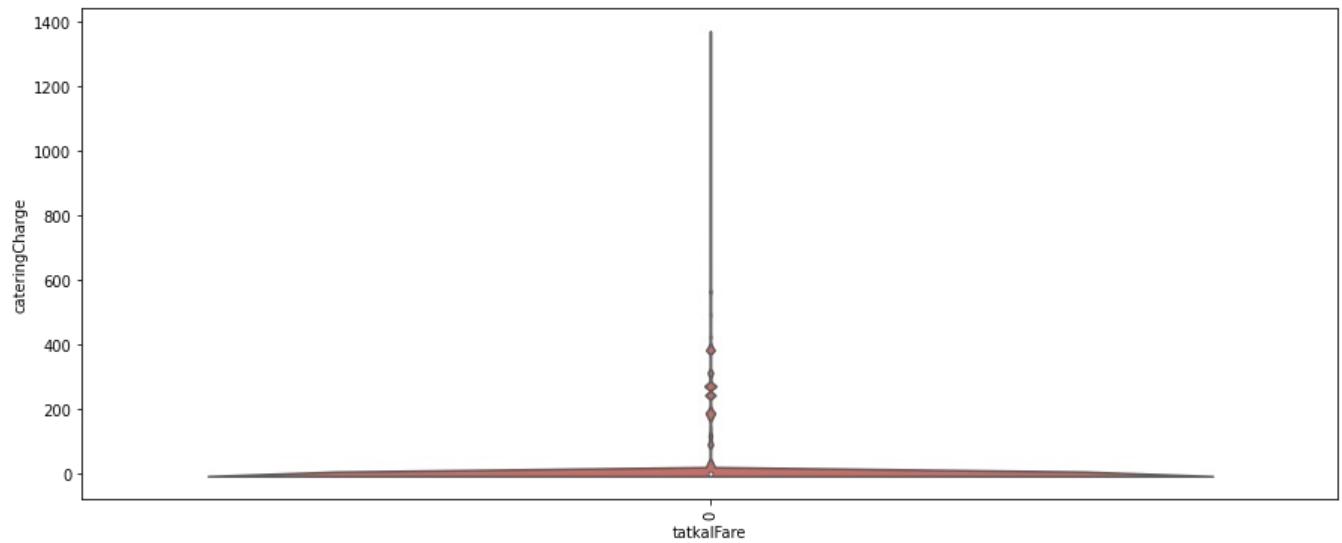
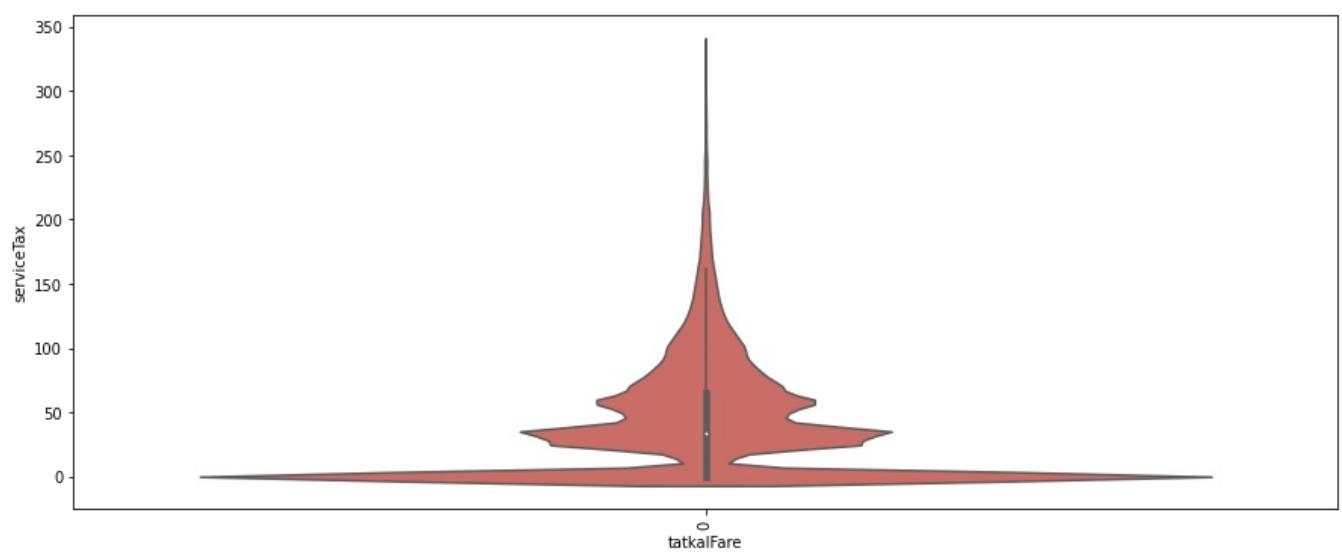


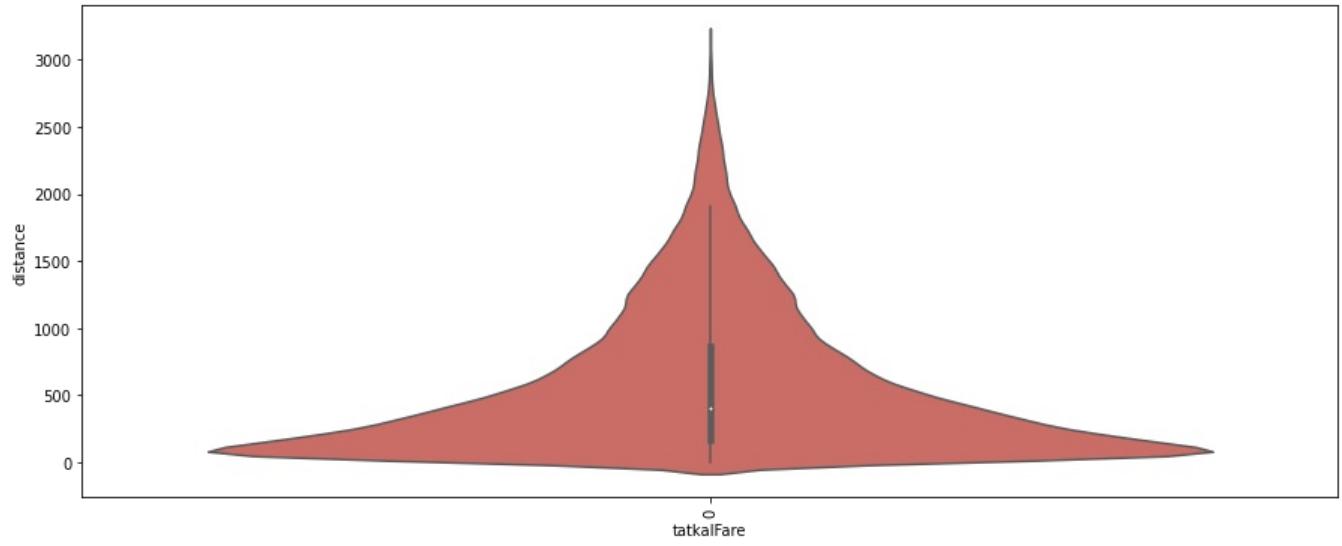
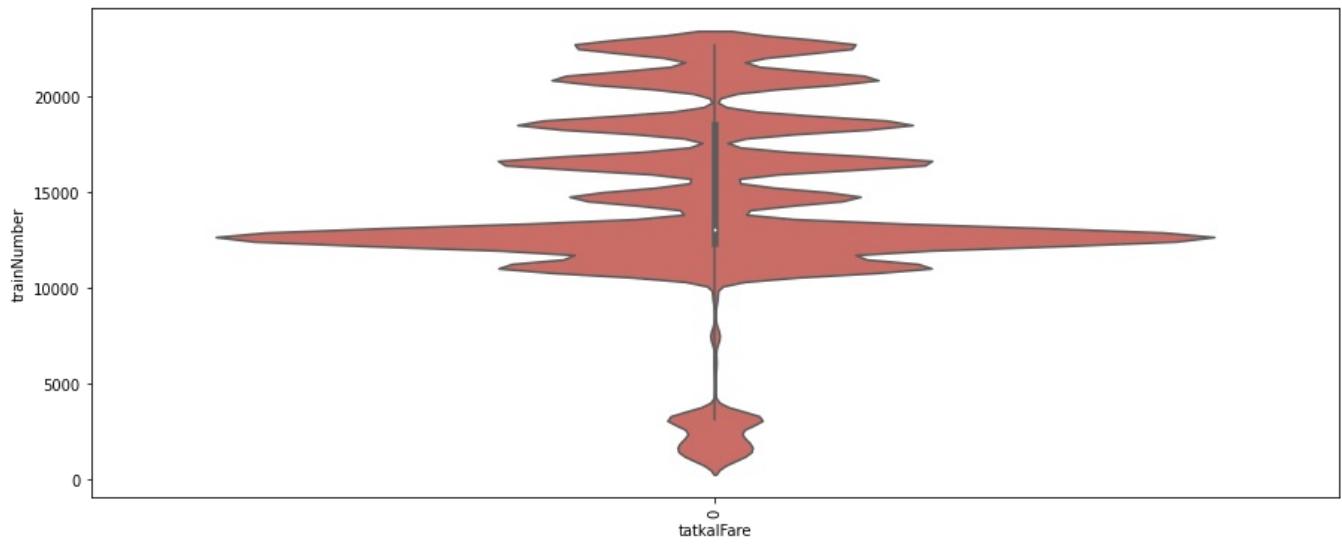
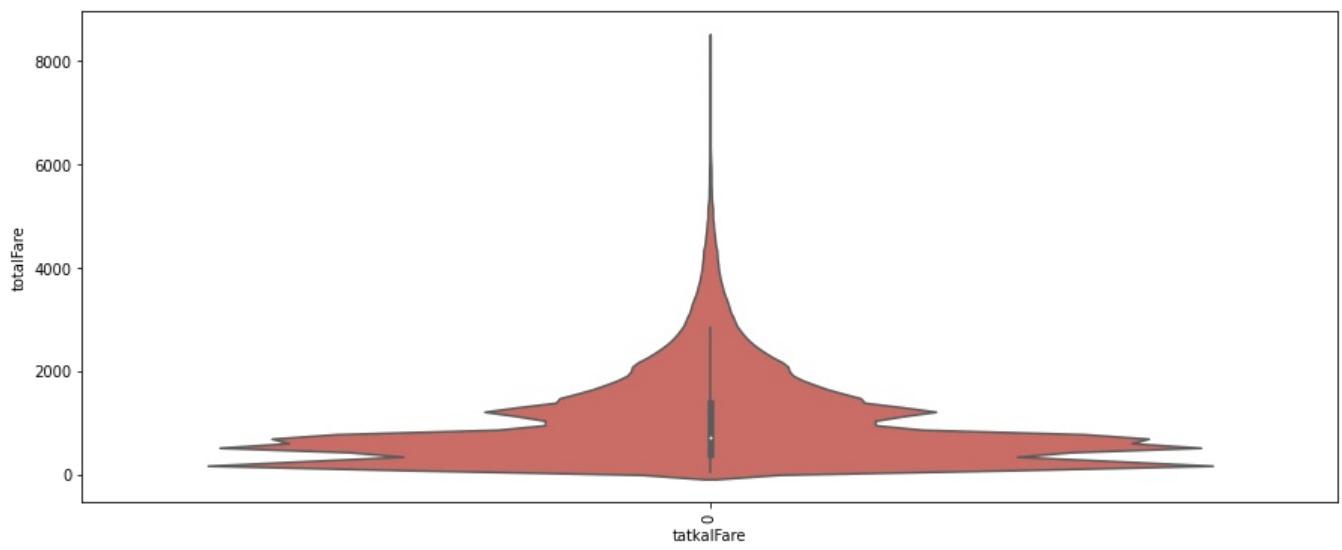


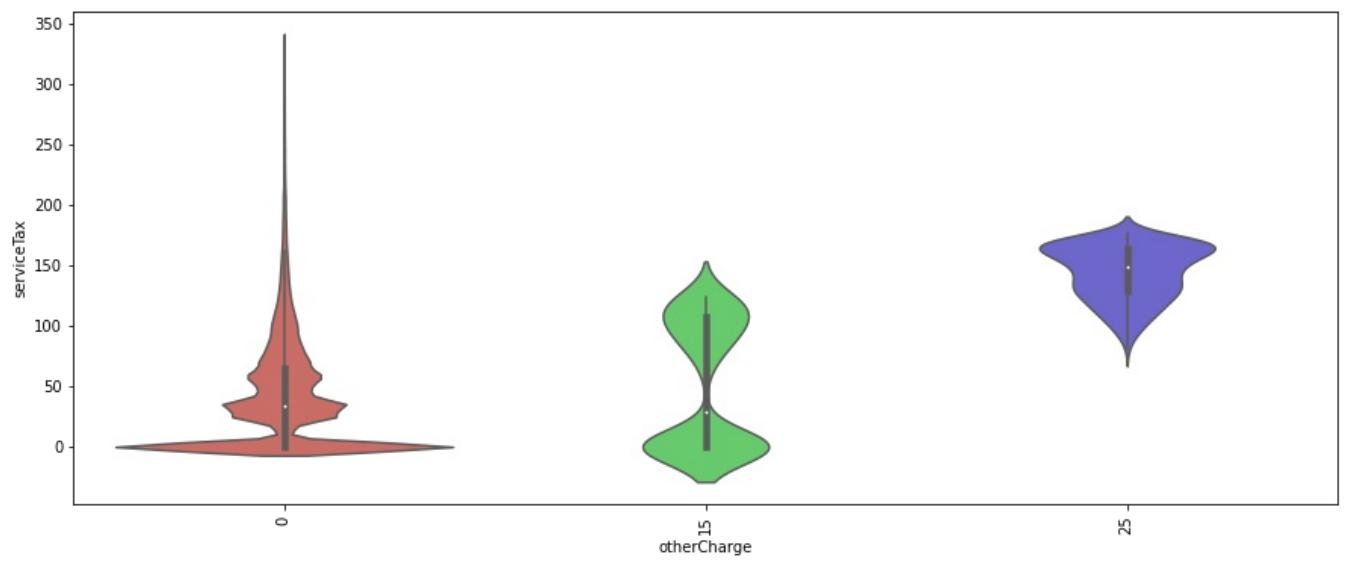
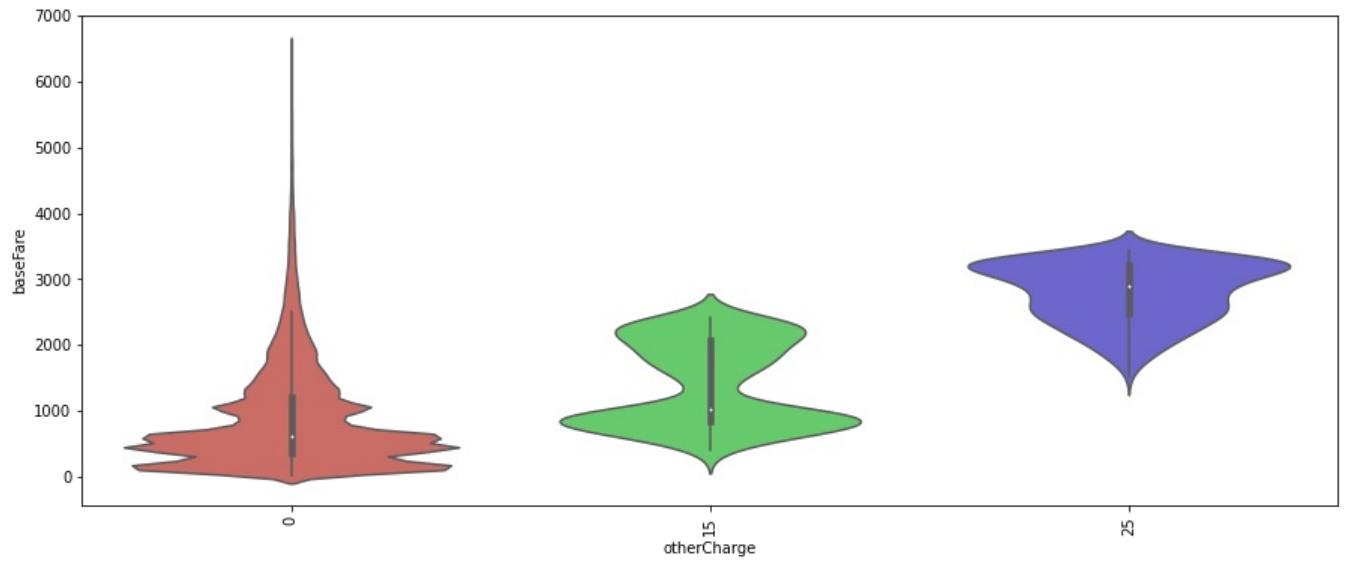
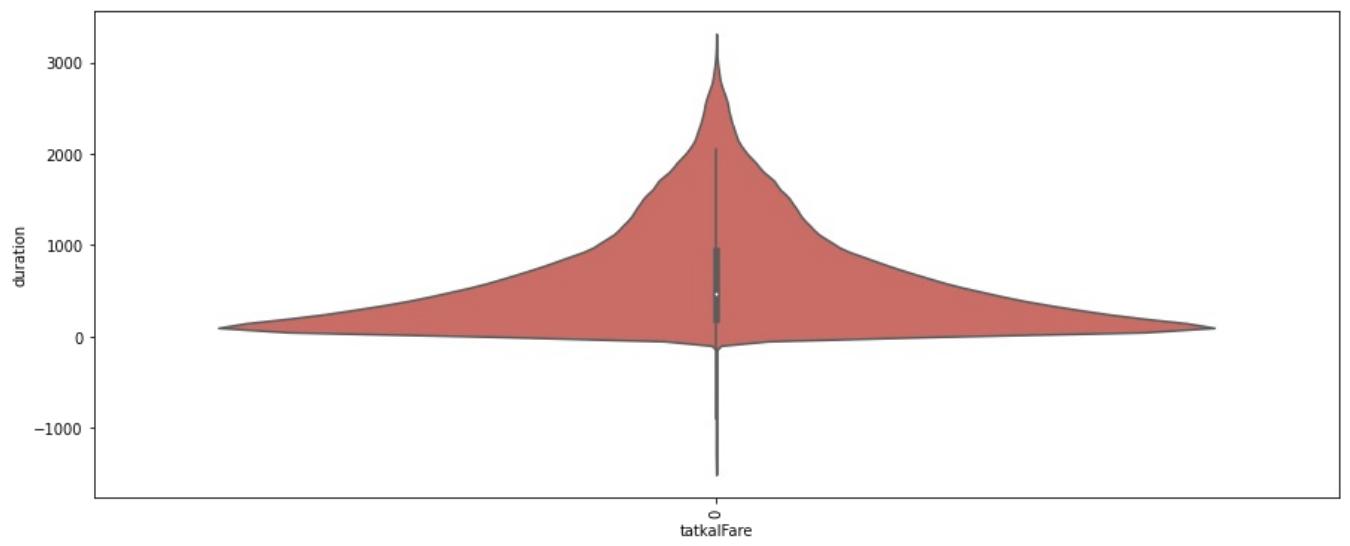


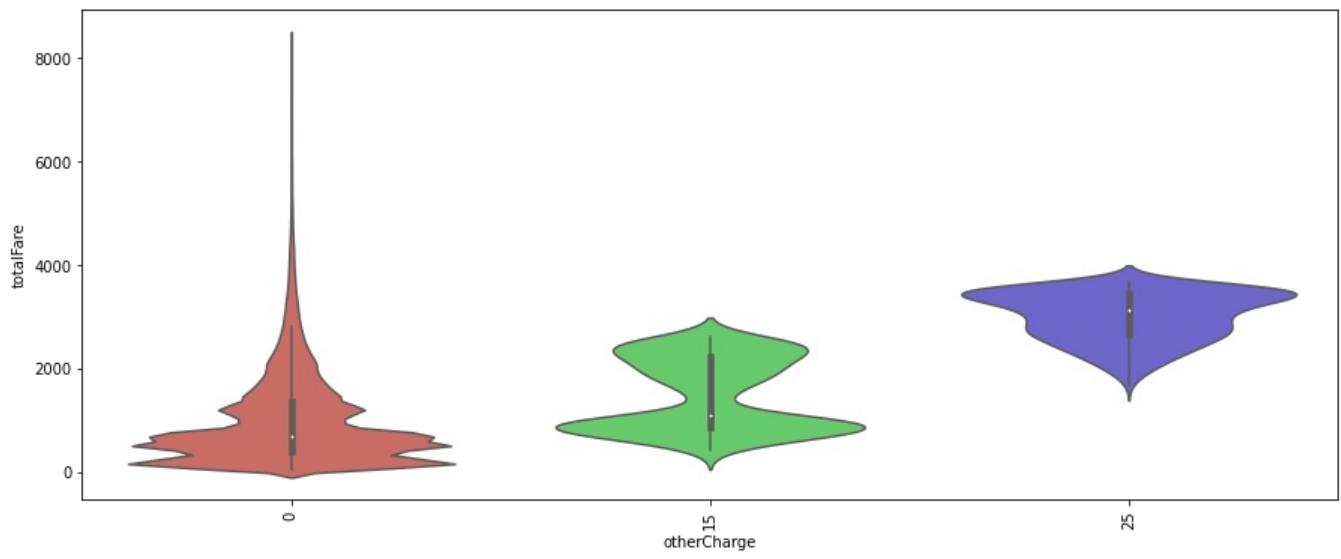
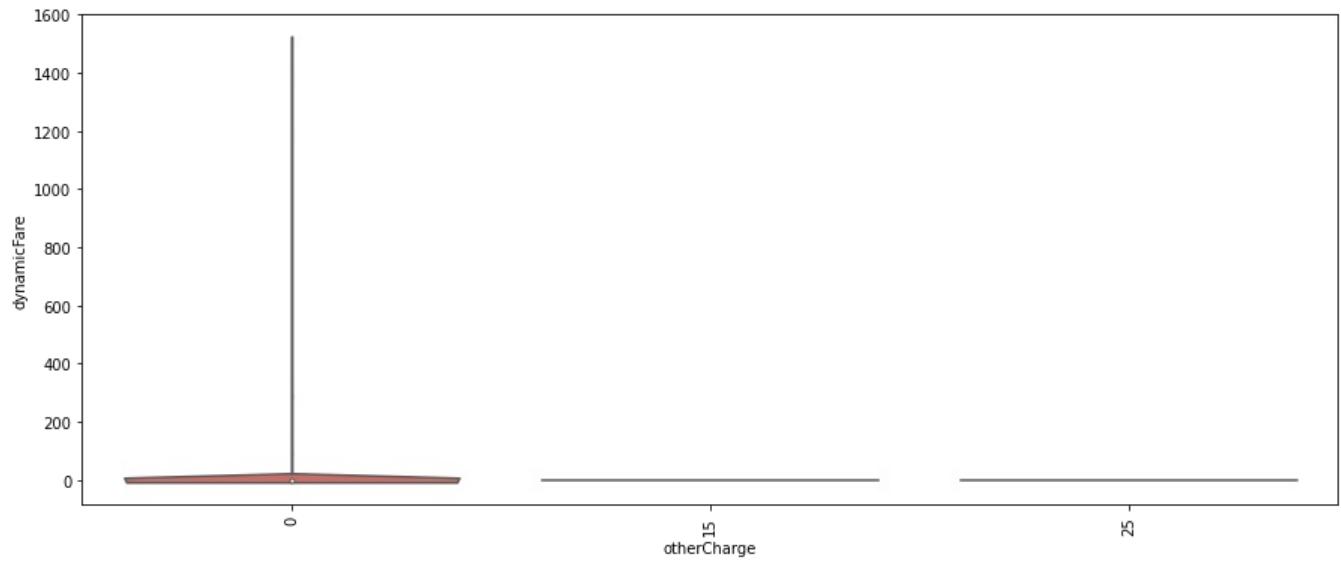
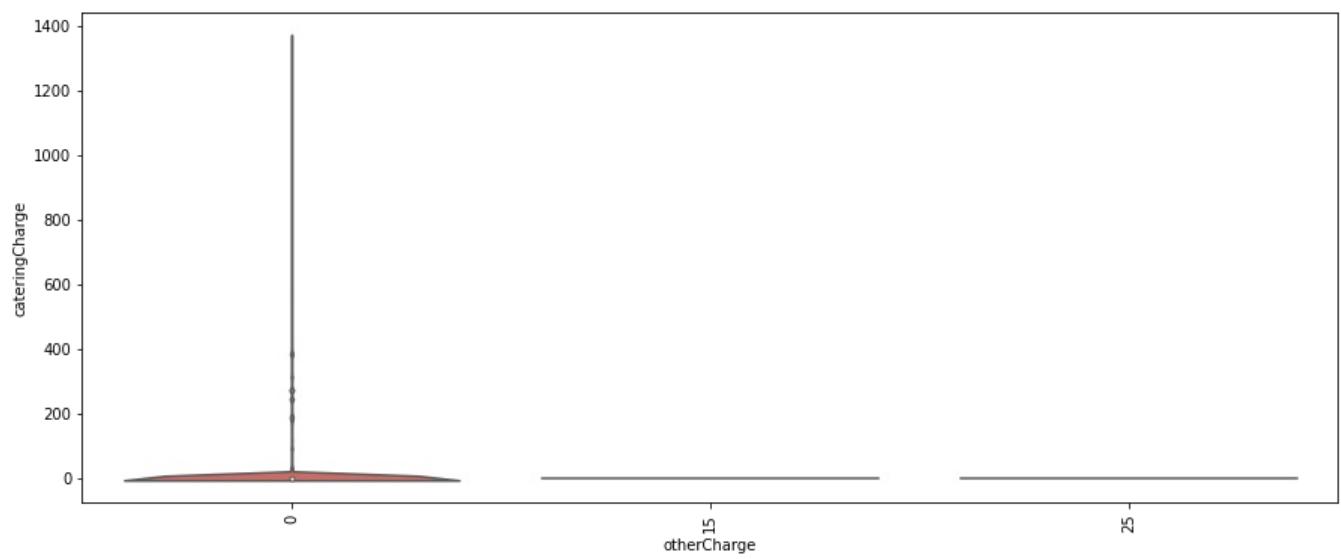


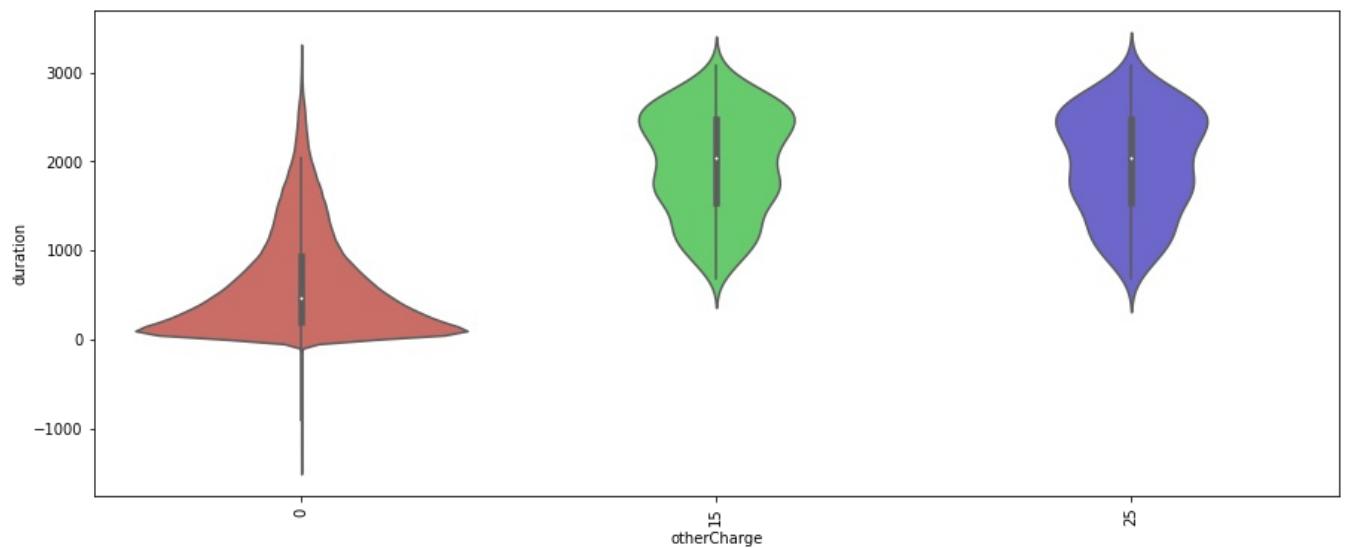
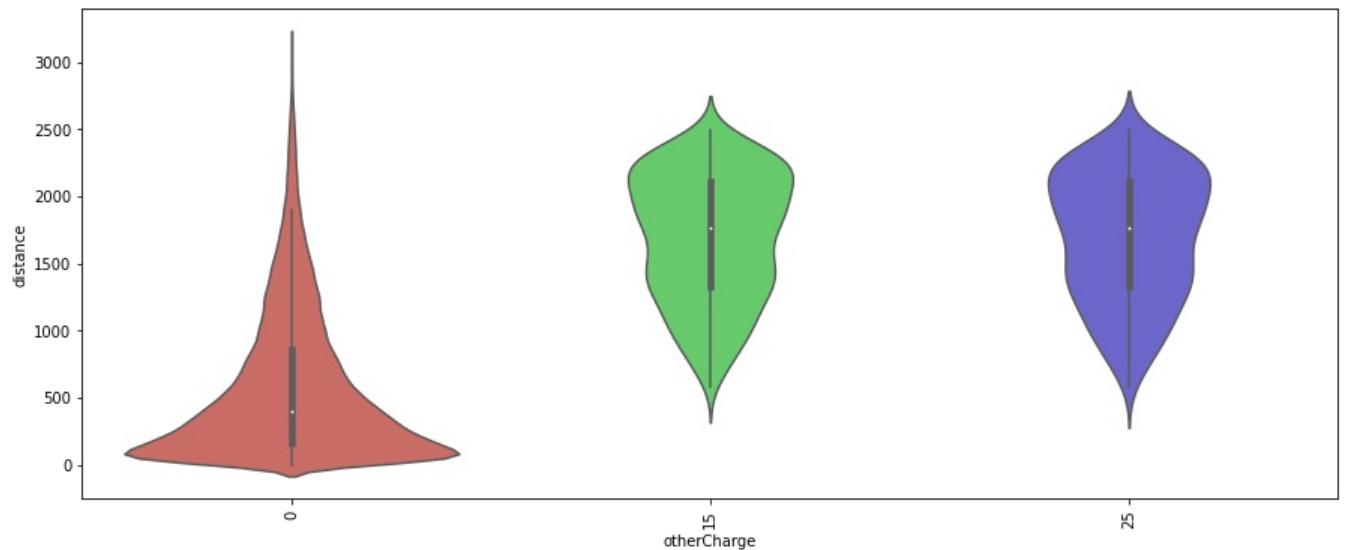
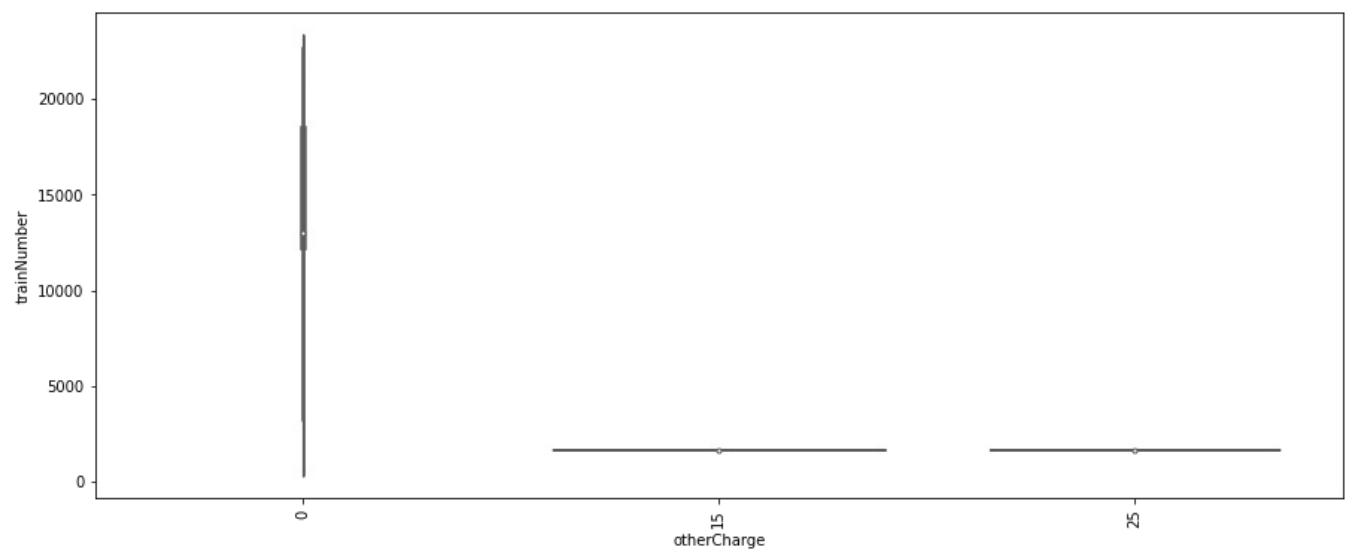










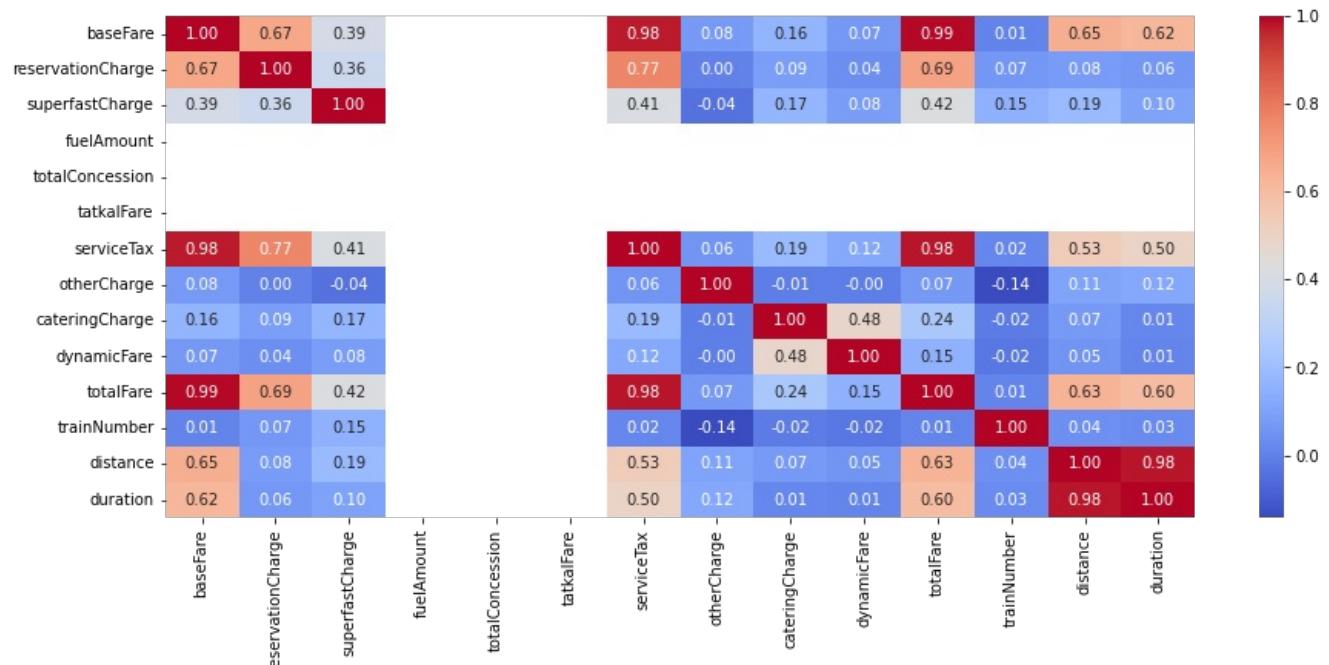


```
In [37]: correlation_matrix = df.corr()
```

```
In [38]: correlation_matrix
```

	baseFare	reservationCharge	superfastCharge	fuelAmount	totalConcession	tatkalFare	serviceTax	otherCharge	cateringC
baseFare	1.000000	0.673303	0.390930	NaN	NaN	NaN	0.977310	0.075733	0.1
reservationCharge	0.673303	1.000000	0.356367	NaN	NaN	NaN	0.766696	0.003403	0.0
superfastCharge	0.390930	0.356367	1.000000	NaN	NaN	NaN	0.411748	-0.042521	0.1
fuelAmount	NaN	NaN	NaN	NaN	NaN	NaN	NaN	NaN	NaN
totalConcession	NaN	NaN	NaN	NaN	NaN	NaN	NaN	NaN	NaN
tatkalFare	NaN	NaN	NaN	NaN	NaN	NaN	NaN	NaN	NaN
serviceTax	0.977310	0.766696	0.411748	NaN	NaN	NaN	1.000000	0.058940	0.1
otherCharge	0.075733	0.003403	-0.042521	NaN	NaN	NaN	0.058940	1.000000	-0.0
cateringCharge	0.164796	0.085607	0.167257	NaN	NaN	NaN	0.186007	-0.006123	1.0
dynamicFare	0.067645	0.036815	0.083361	NaN	NaN	NaN	0.122862	-0.004500	0.4
totalFare	0.994863	0.685292	0.424102	NaN	NaN	NaN	0.981640	0.071453	0.2
trainNumber	0.005852	0.065096	0.150937	NaN	NaN	NaN	0.017397	-0.142707	-0.0
distance	0.650017	0.077533	0.186578	NaN	NaN	NaN	0.526907	0.105292	0.0
duration	0.624673	0.064718	0.098778	NaN	NaN	NaN	0.499302	0.119933	0.0

```
In [39]: plt.figure(figsize=(15, 6))
sns.heatmap(correlation_matrix, annot=True, cmap='coolwarm', fmt=".2f")
plt.show()
```



```
In [40]: from sklearn.model_selection import train_test_split
from sklearn.linear_model import LinearRegression
from sklearn.preprocessing import StandardScaler, OneHotEncoder
from sklearn.compose import ColumnTransformer
from sklearn.pipeline import Pipeline
from sklearn.metrics import mean_squared_error, r2_score
```

```
In [41]: X = df[['classCode', 'availability', 'timeStamp', 'fromStnCode', 'toStnCode',
           'reservationCharge', 'superfastCharge', 'fuelAmount', 'totalConcession',
           'tatkalFare', 'otherCharge', 'baseFare', 'serviceTax', 'cateringCharge',
           'dynamicFare', 'trainNumber', 'distance', 'duration']]
```

```
In [42]: y = df['totalFare']
```

```
In [43]: X_train, X_test, y_train, y_test = train_test_split(X, y, test_size=0.2, random_state=42)
```

```
In [44]: numeric_features = ['reservationCharge', 'superfastCharge', 'fuelAmount', 'totalConcession',
                           'tatkalFare', 'otherCharge', 'baseFare', 'serviceTax', 'cateringCharge',
                           'dynamicFare', 'trainNumber', 'distance', 'duration']

categorical_features = ['classCode', 'availability', 'fromStnCode', 'toStnCode']
```

```
In [45]: numeric_transformer = Pipeline(steps=[
    ('scaler', StandardScaler())
])

categorical_transformer = Pipeline(steps=[
```

```
('onehot', OneHotEncoder(handle_unknown='ignore'))
])
```

```
In [46]: preprocessor = ColumnTransformer(
    transformers=[
        ('num', numeric_transformer, numeric_features),
        ('cat', categorical_transformer, categorical_features)
    ])
])
```

```
In [47]: model_lr = Pipeline(steps=[
    ('preprocessor', preprocessor),
    ('regressor', LinearRegression())
])
])
```

```
In [48]: model_lr.fit(X_train, y_train)
```

```
Out[48]: Pipeline
  preprocessor: ColumnTransformer
    num      cat
    StandardScaler OneHotEncoder
      |
      |
      LinearRegression
```

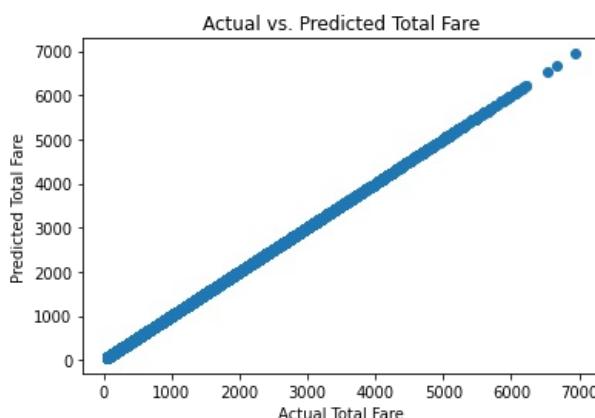
```
In [49]: y_pred = model_lr.predict(X_test)
```

```
In [50]: mse = mean_squared_error(y_test, y_pred)
r2 = r2_score(y_test, y_pred)

print(f'Mean Squared Error: {mse}')
print(f'R-squared: {r2}')

Mean Squared Error: 0.0636917967886374
R-squared: 0.9999999160165995
```

```
In [51]: plt.scatter(y_test, y_pred)
plt.xlabel('Actual Total Fare')
plt.ylabel('Predicted Total Fare')
plt.title('Actual vs. Predicted Total Fare')
plt.show()
```



```
In [52]: correlation_matrix = X.corr().abs()
```

```
In [53]: upper_triangle = correlation_matrix.where(np.triu(np.ones(correlation_matrix.shape), k=1).astype(np.bool))
```

```
In [54]: correlated_features = [column for column in upper_triangle.columns if any(upper_triangle[column] > 0.75)]
```

```
In [55]: X_filtered = X.drop(correlated_features, axis=1)
```

```
In [56]: X_train, X_test, y_train, y_test = train_test_split(X_filtered, y, test_size=0.2, random_state=42)
```

```
In [57]: numeric_features = list(X_filtered.select_dtypes(include=['float64', 'int64']).columns)
categorical_features = list(X_filtered.select_dtypes(include=['object']).columns)
```

```
In [58]: numeric_transformer = Pipeline(steps=[
    ('scaler', StandardScaler())
])
])
```

```
categorical_transformer = Pipeline(steps=[
    ('onehot', OneHotEncoder(handle_unknown='ignore'))
])
```

In [59]:

```
preprocessor = ColumnTransformer(
    transformers=[
        ('num', numeric_transformer, numeric_features),
        ('cat', categorical_transformer, categorical_features)
    ])
```

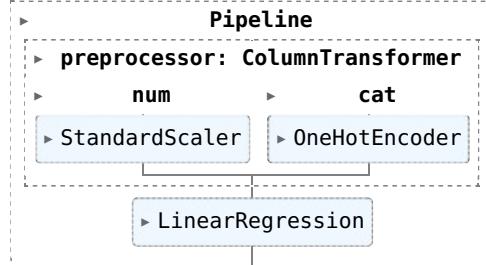
In [60]:

```
model_lr_ = Pipeline(steps=[
    ('preprocessor', preprocessor),
    ('regressor', LinearRegression())
])
```

In [61]:

```
model_lr_.fit(X_train, y_train)
```

Out[61]:



In [62]:

```
y_pred = model_lr_.predict(X_test)
```

In [63]:

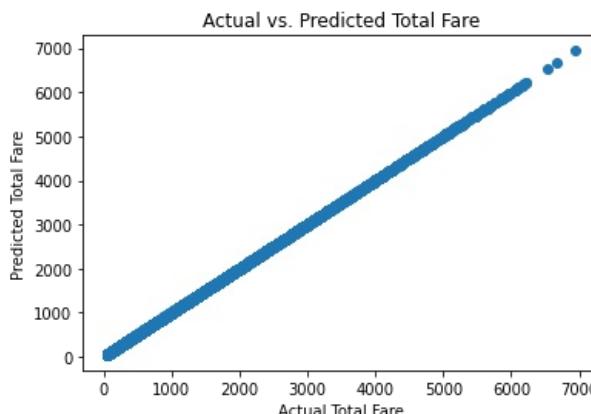
```
mse = mean_squared_error(y_test, y_pred)
r2 = r2_score(y_test, y_pred)

print(f'Mean Squared Error: {mse}')
print(f'R-squared: {r2}' )
```

Mean Squared Error: 3.7379905707774577
R-squared: 0.9999950711210074

In [64]:

```
plt.scatter(y_test, y_pred)
plt.xlabel('Actual Total Fare')
plt.ylabel('Predicted Total Fare')
plt.title('Actual vs. Predicted Total Fare')
plt.show()
```



In [65]:

```
df_new = df.copy()
```

In [68]:

```
import ast
```

In [69]:

```
df_new['availability'] = df_new['availability'].apply(ast.literal_eval)
```

In [70]:

```
df_new['availability_date'] = df_new['availability'].apply(lambda x: [entry['date'] for entry in x])
```

In [71]:

```
df_new['availability_date']
```

Out[71]:

```
0      [2-12-2023, 3-12-2023, 5-12-2023, 6-12-2023, 7...
1      [2-12-2023, 3-12-2023, 5-12-2023, 6-12-2023, 7...
2      [2-12-2023, 3-12-2023, 5-12-2023, 6-12-2023, 7...
3      [2-12-2023, 3-12-2023, 5-12-2023, 6-12-2023, 7...
4      [2-12-2023, 3-12-2023, 5-12-2023, 6-12-2023, 7...
...
326638  [26-11-2023, 28-11-2023, 30-11-2023, 2-12-2023...
326639  [7-12-2023, 9-12-2023, 10-12-2023, 12-12-2023...
326640  [17-12-2023, 19-12-2023, 21-12-2023, 23-12-202...
326641  [28-12-2023, 30-12-2023, 31-12-2023, 2-1-2024, ...
326642  [7-1-2024, 9-1-2024, 11-1-2024, 13-1-2024, 14-...
Name: availability_date, Length: 326643, dtype: object
```

In [72]:

```
df_new['availability_status'] = df_new['availability'].apply(lambda x: [entry['status'] for entry in x])
```

Out[72]:

```
df_new['availability_status']
```

```
In [73]: df_new['availability_status']
```

```
Out[73]: 0      [AVAILABLE-0008, AVAILABLE-0014, AVAILABLE-001...  
1      [AVAILABLE-0020, AVAILABLE-0026, AVAILABLE-002...  
2      [AVAILABLE-0136, AVAILABLE-0192, AVAILABLE-018...  
3      [AVAILABLE-0090, AVAILABLE-0101, AVAILABLE-010...  
4      [AVAILABLE-0008, AVAILABLE-0014, AVAILABLE-001...  
     ...  
326638  [RLWL9/WL9, AVAILABLE-0005, RLWL2/WL2, RLWL3/W...  
326639  [RLWL3/WL3, AVAILABLE-0006, AVAILABLE-0003, AV...  
326640  [AVAILABLE-0006, AVAILABLE-0006, AVAILABLE-000...  
326641  [AVAILABLE-0006, AVAILABLE-0005, AVAILABLE-000...  
326642  [AVAILABLE-0006, AVAILABLE-0006, AVAILABLE-000...  
Name: availability_status, Length: 326643, dtype: object
```

```
In [74]: df_new['is_overnight'] = df_new['duration'] > 23
```

```
In [76]: import re
```

```
In [77]: waiting_pattern = re.compile(r'RLWL(\d+)/WL(\d+)')
```

```
In [78]: df_new['has_waiting_ticket'] = df_new['availability_status'].apply(lambda x: any(waiting_pattern.match(status)
```

```
In [79]: df_new['has_waiting_ticket']
```

```
Out[79]: 0      False  
1      False  
2      False  
3      False  
4      False  
     ...  
326638  True  
326639  True  
326640  False  
326641  False  
326642  False  
Name: has_waiting_ticket, Length: 326643, dtype: bool
```

```
In [80]: df_new['priority'] = (df_new['is_overnight'] & df_new['has_waiting_ticket']).astype(int)
```

```
In [81]: df_new['priority']
```

```
Out[81]: 0      0  
1      0  
2      0  
3      0  
4      0  
     ..  
326638  1  
326639  1  
326640  0  
326641  0  
326642  0  
Name: priority, Length: 326643, dtype: int32
```

```
In [82]: df_priority = df_new[['availability', 'availability_date', 'availability_status', 'duration', 'is_overnight', 'priority']]
```

```
In [83]: df_priority
```

Out[83]:

	availability	availability_date	availability_status	duration	is_overnight	has_waiting_ticket	priority
0	[{'date': '2-12-2023', 'status': 'AVAILABLE-00...']	[2-12-2023, 3-12-2023, 5-12-2023, 6-12-2023, 7...]	[AVAILABLE-0008, AVAILABLE-0014, AVAILABLE-001...	33	True	False	0
1	[{'date': '2-12-2023', 'status': 'AVAILABLE-00...']	[2-12-2023, 3-12-2023, 5-12-2023, 6-12-2023, 7...]	[AVAILABLE-0020, AVAILABLE-0026, AVAILABLE-002...	33	True	False	0
2	[{'date': '2-12-2023', 'status': 'AVAILABLE-01...']	[2-12-2023, 3-12-2023, 5-12-2023, 6-12-2023, 7...]	[AVAILABLE-0136, AVAILABLE-0192, AVAILABLE-018...	33	True	False	0
3	[{'date': '2-12-2023', 'status': 'AVAILABLE-00...']	[2-12-2023, 3-12-2023, 5-12-2023, 6-12-2023, 7...]	[AVAILABLE-0090, AVAILABLE-0101, AVAILABLE-010...	33	True	False	0
4	[{'date': '2-12-2023', 'status': 'AVAILABLE-00...']	[2-12-2023, 3-12-2023, 5-12-2023, 6-12-2023, 7...]	[AVAILABLE-0008, AVAILABLE-0014, AVAILABLE-001...	49	True	False	0
...
326638	[{'date': '26-11-2023', 'status': 'RLWL9/WL9'}...]	[26-11-2023, 28-11-2023, 30-11-2023, 2-12-2023...]	[RLWL9/WL9, AVAILABLE-0005, RLWL2/WL2, RLWL3/W...	570	True	True	1
326639	[{'date': '7-12-2023', 'status': 'RLWL3/WL3'}...]	[7-12-2023, 9-12-2023, 10-12-2023, 12-12-2023,...]	[RLWL3/WL3, AVAILABLE-0006, AVAILABLE-0003, AV...	570	True	True	1
326640	[{'date': '17-12-2023', 'status': 'AVAILABLE-0...']	[17-12-2023, 19-12-2023, 21-12-2023, 23-12-202...]	[AVAILABLE-0006, AVAILABLE-0006, AVAILABLE-000...	570	True	False	0
326641	[{'date': '28-12-2023', 'status': 'AVAILABLE-0...']	[28-12-2023, 30-12-2023, 31-12-2023, 2-1-2024,...]	[AVAILABLE-0006, AVAILABLE-0005, AVAILABLE-000...	570	True	False	0
326642	[{'date': '7-1-2024', 'status': 'AVAILABLE-000...']	[7-1-2024, 9-1-2024, 11-1-2024, 13-1-2024, 14-...	[AVAILABLE-0006, AVAILABLE-0006, AVAILABLE-000...	570	True	False	0

326643 rows × 7 columns

Thanks !!!

Loading [MathJax]/extensions/Safe.js