Air_Traffic_Passenger

In [1]:
```python
import pandas as pd
import numpy as np
import seaborn as sns
import matplotlib.pyplot as plt
import warnings
warnings.filterwarnings("ignore")
```

load data set

In [2]:
```python
df=pd.read_csv("Air_Traffic_Passenger_Statistics.csv.xls")
```

In [3]:
```python
df.head()
```

Out[3]:

	Activity Period	Operating Airline	Operating Airline IATA Code	Published Airline	Published Airline IATA Code	GEO Summary	GEO Region	Activity Type Code	Price Category Code	Terminal	Boarding Area	Passenger Count	Adjustec Activity Type Code
0	200507	ATA Airlines	TZ	ATA Airlines	TZ	Domestic	US	Deplaned	Low Fare	Terminal 1	B	27271	Deplanec
1	200507	ATA Airlines	TZ	ATA Airlines	TZ	Domestic	US	Enplaned	Low Fare	Terminal 1	B	29131	Enplanec
2	200507	ATA Airlines	TZ	ATA Airlines	TZ	Domestic	US	Thru / Transit	Low Fare	Terminal 1	B	5415	Thru / Transit * 2
3	200507	Air Canada	AC	Air Canada	AC	International	Canada	Deplaned	Other	Terminal 1	B	35156	Deplanec
4	200507	Air Canada	AC	Air Canada	AC	International	Canada	Enplaned	Other	Terminal 1	B	34090	Enplanec

In [4]:
```python
df.info()
```

```
<class 'pandas.core.frame.DataFrame'>
RangeIndex: 15007 entries, 0 to 15006
Data columns (total 16 columns):
 #   Column                       Non-Null Count  Dtype 
---  ------                       --------------  ----- 
 0   Activity Period              15007 non-null  int64 
 1   Operating Airline            15007 non-null  object
 2   Operating Airline IATA Code  14953 non-null  object
 3   Published Airline            15007 non-null  object
 4   Published Airline IATA Code  14953 non-null  object
 5   GEO Summary                  15007 non-null  object
 6   GEO Region                   15007 non-null  object
 7   Activity Type Code           15007 non-null  object
 8   Price Category Code          15007 non-null  object
 9   Terminal                     15007 non-null  object
 10  Boarding Area                15007 non-null  object
 11  Passenger Count              15007 non-null  int64 
 12  Adjusted Activity Type Code  15007 non-null  object
 13  Adjusted Passenger Count     15007 non-null  int64 
 14  Year                         15007 non-null  int64 
 15  Month                        15007 non-null  object
dtypes: int64(4), object(12)
memory usage: 1.8+ MB
```

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

Out[5]:

	Activity Period	Passenger Count	Adjusted Passenger Count	Year
count	15007.000000	15007.000000	15007.000000	15007.000000
mean	201045.073366	29240.521090	29331.917105	2010.385220
std	313.336196	58319.509284	58284.182219	3.137589
min	200507.000000	1.000000	1.000000	2005.000000
25%	200803.000000	5373.500000	5495.500000	2008.000000
50%	201011.000000	9210.000000	9354.000000	2010.000000
75%	201308.000000	21158.500000	21182.000000	2013.000000
max	201603.000000	659837.000000	659837.000000	2016.000000

missing value handaling

In [6]:

```python
from sklearn.impute import SimpleImputer
si=SimpleImputer(missing_values=np.nan,strategy='most_frequent')
df["Operating Airline IATA Code"]=si.fit_transform(df[["Operating Airline IATA Code"]])
df["Published Airline IATA Code"]=si.fit_transform(df[["Published Airline IATA Code"]])
```

In [7]:

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

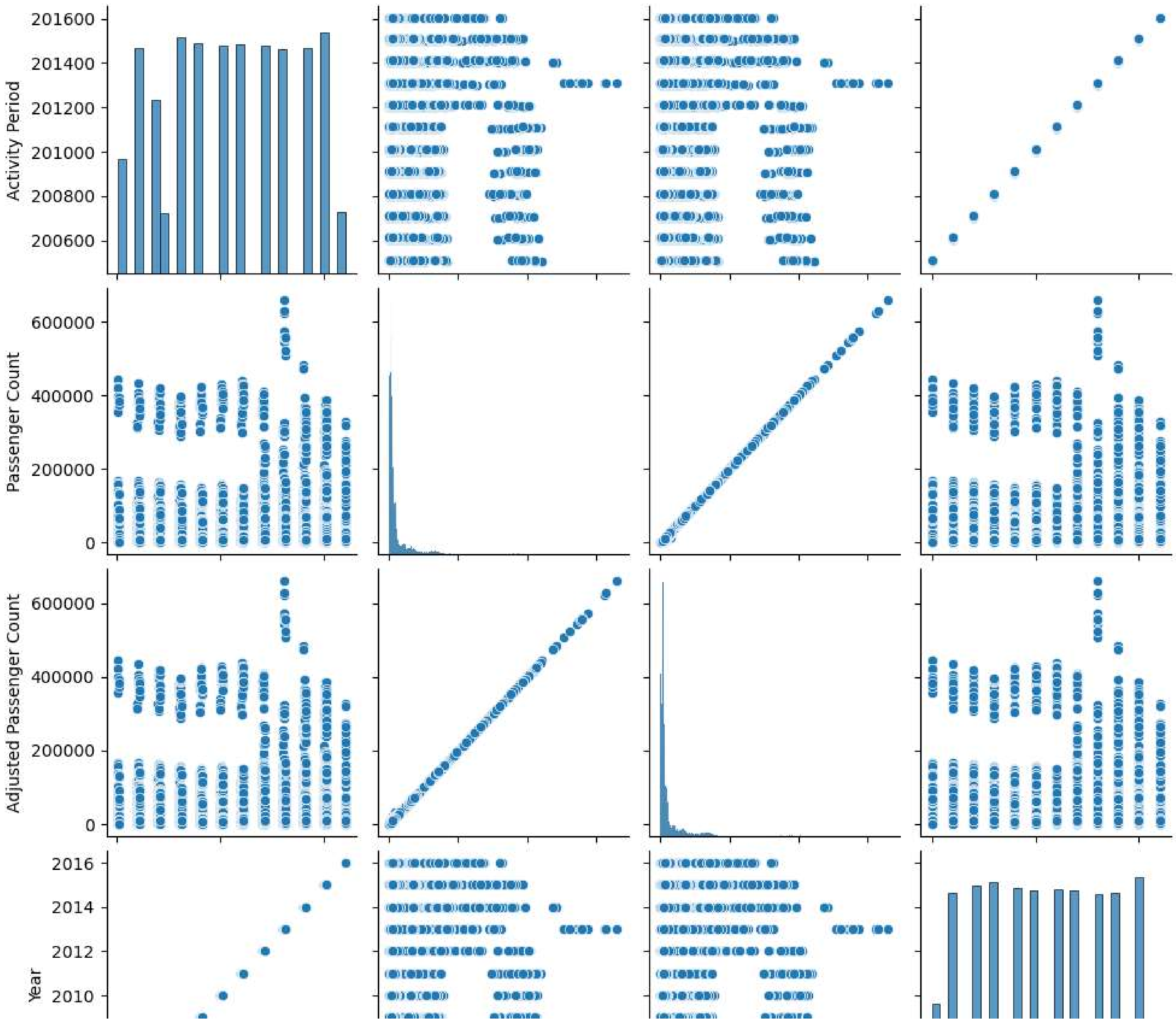
```
<class 'pandas.core.frame.DataFrame'>
RangeIndex: 15007 entries, 0 to 15006
Data columns (total 16 columns):
 #   Column                       Non-Null Count  Dtype 
---  ------                       --------------  ----- 
 0   Activity Period              15007 non-null  int64 
 1   Operating Airline            15007 non-null  object
 2   Operating Airline IATA Code  15007 non-null  object
 3   Published Airline            15007 non-null  object
 4   Published Airline IATA Code  15007 non-null  object
 5   GEO Summary                  15007 non-null  object
 6   GEO Region                   15007 non-null  object
 7   Activity Type Code           15007 non-null  object
 8   Price Category Code          15007 non-null  object
 9   Terminal                     15007 non-null  object
 10  Boarding Area                15007 non-null  object
 11  Passenger Count              15007 non-null  int64 
 12  Adjusted Activity Type Code  15007 non-null  object
 13  Adjusted Passenger Count     15007 non-null  int64 
 14  Year                         15007 non-null  int64 
 15  Month                        15007 non-null  object
dtypes: int64(4), object(12)
memory usage: 1.8+ MB
```

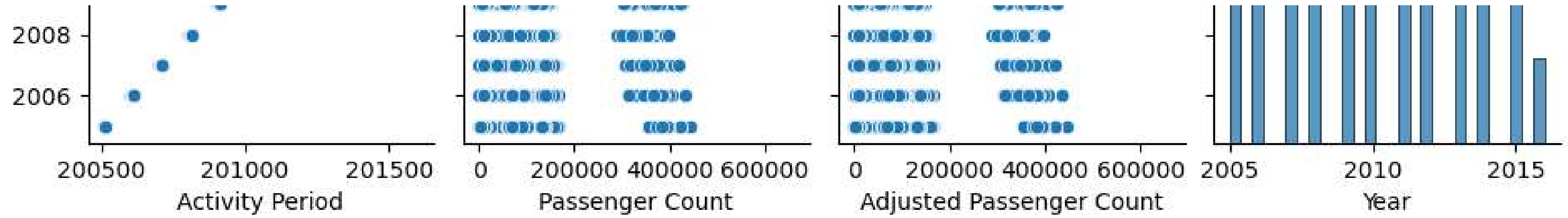
In [8]:

```python
sns.pairplot(df)
```

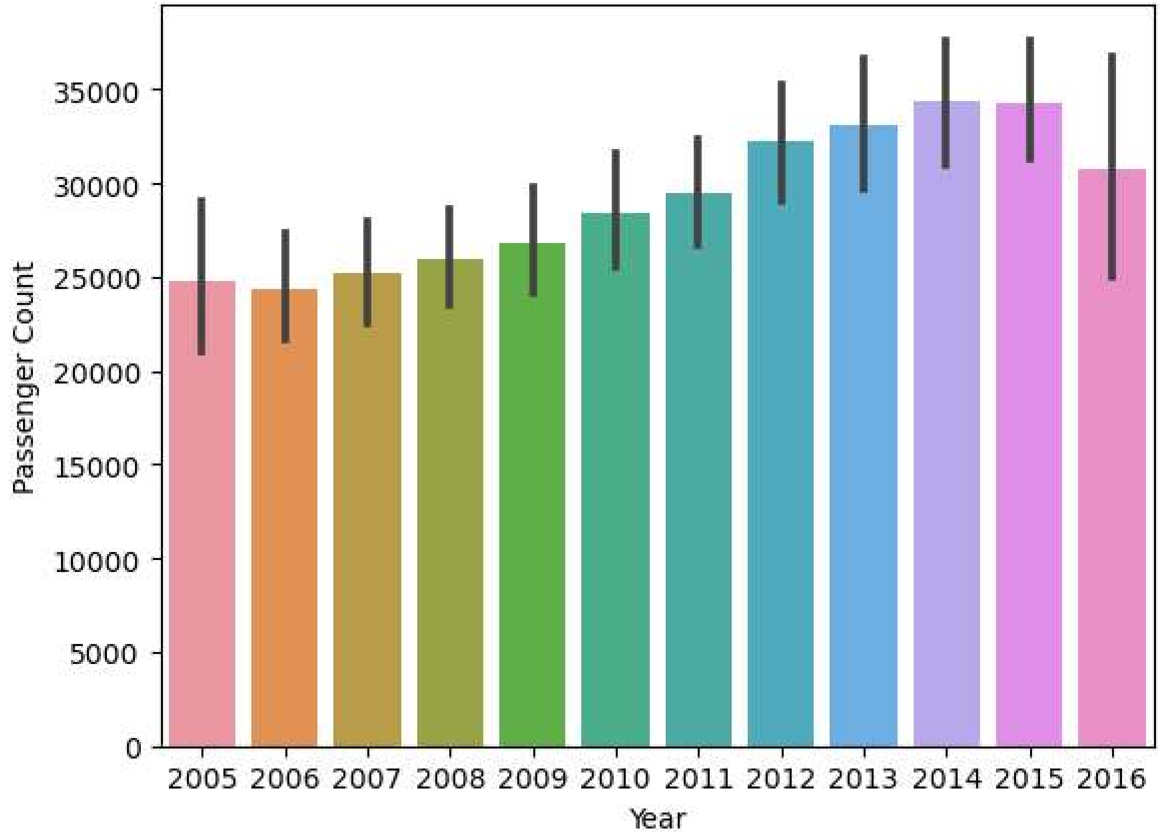
Out[8]:

```
<seaborn.axisgrid.PairGrid at 0x21d03ab63d0>
```


Activity Period
201600
201400
201200
201000
200800
200600
Passenger Count
600000
400000
200000
0
Adjusted Passenger Count
600000
400000
200000
0
Year
2016
2014
2012
2010

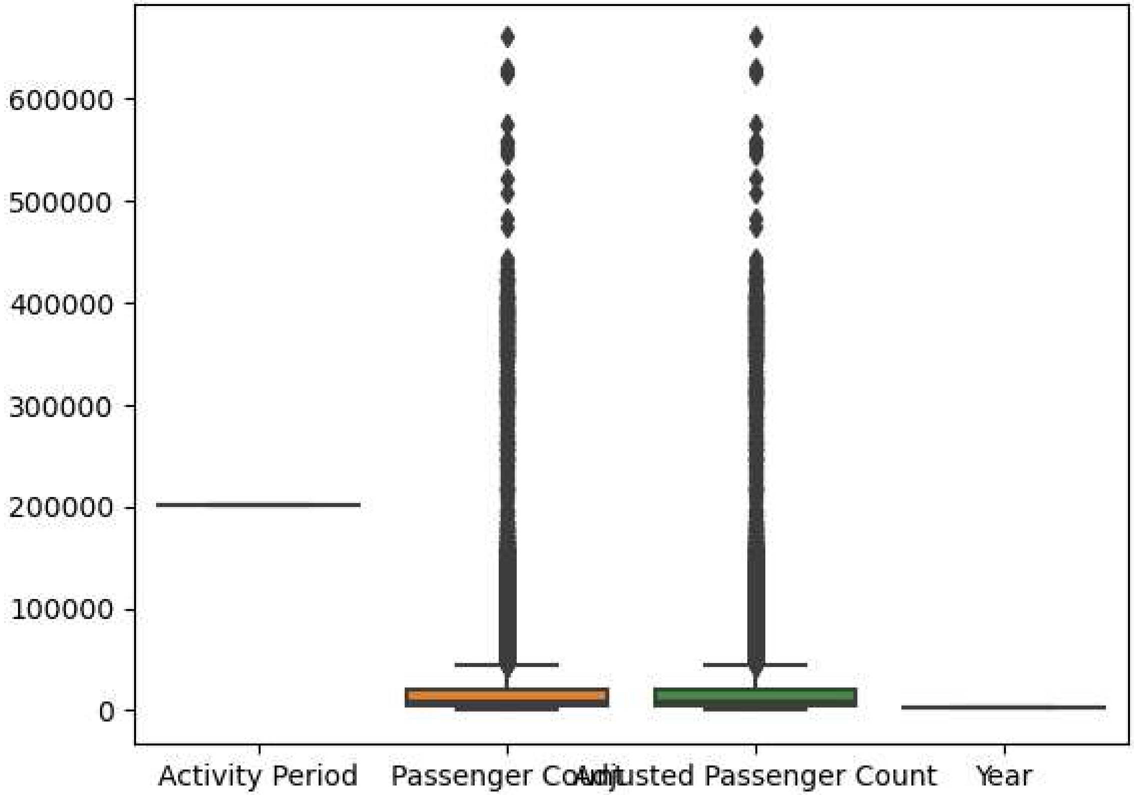


In [9]:
```
sns.barplot(x=df["Year"],y=df["Passenger Count"])
plt.show()
```



outliers detection

In [10]:
```
sns.boxplot(data=df)
plt.show()
```



checking corelation

In [11]:
```
df.corr().style.background_gradient()
```

Out[11]:

	Activity Period	Passenger Count	Adjusted Passenger Count	Year
Activity Period	1.000000	0.060311	0.059336	0.999940
Passenger Count	0.060311	1.000000	0.999941	0.060069
Adjusted Passenger Count	0.059336	0.999941	1.000000	0.059096
Year	0.999940	0.060069	0.059096	1.000000

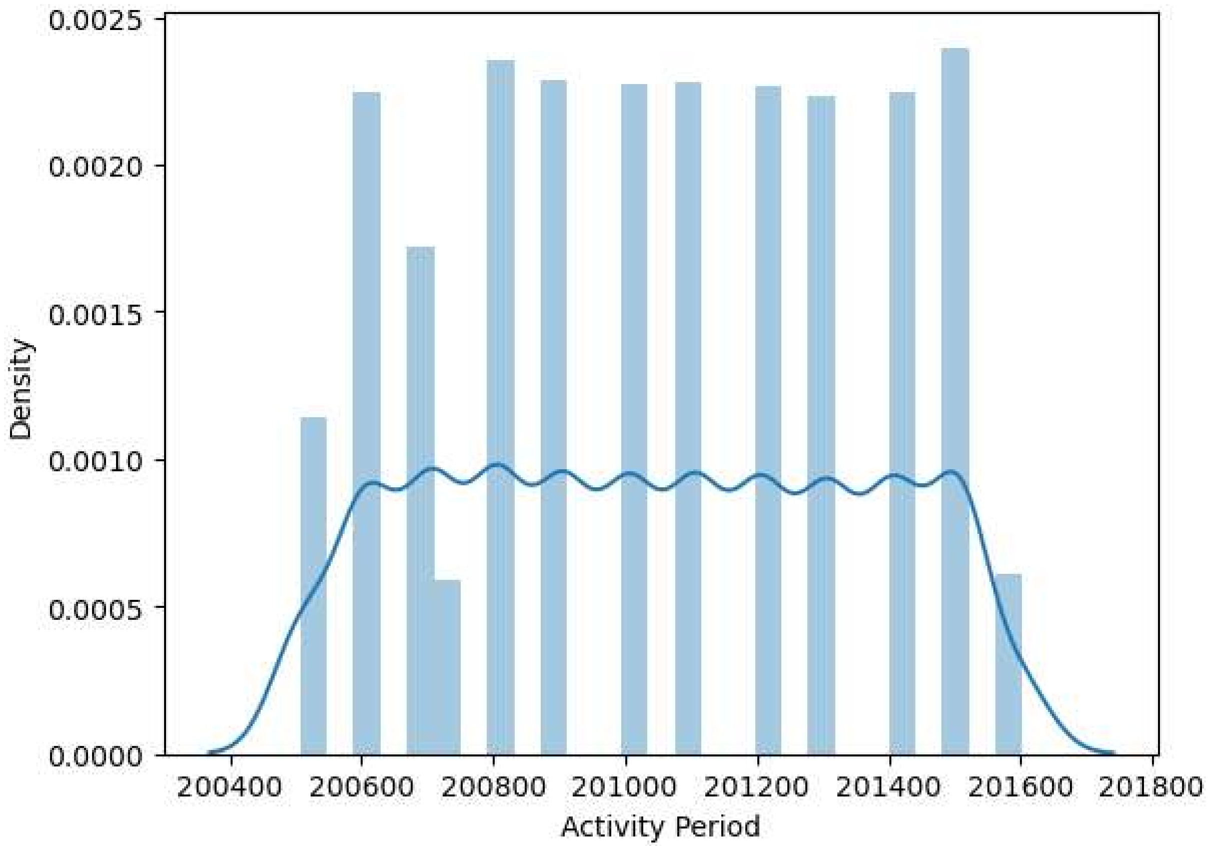
skew

In [18]:
```
col=df.select_dtypes("int64","float64").columns
x=col=df.select_dtypes("int64","float64")
```

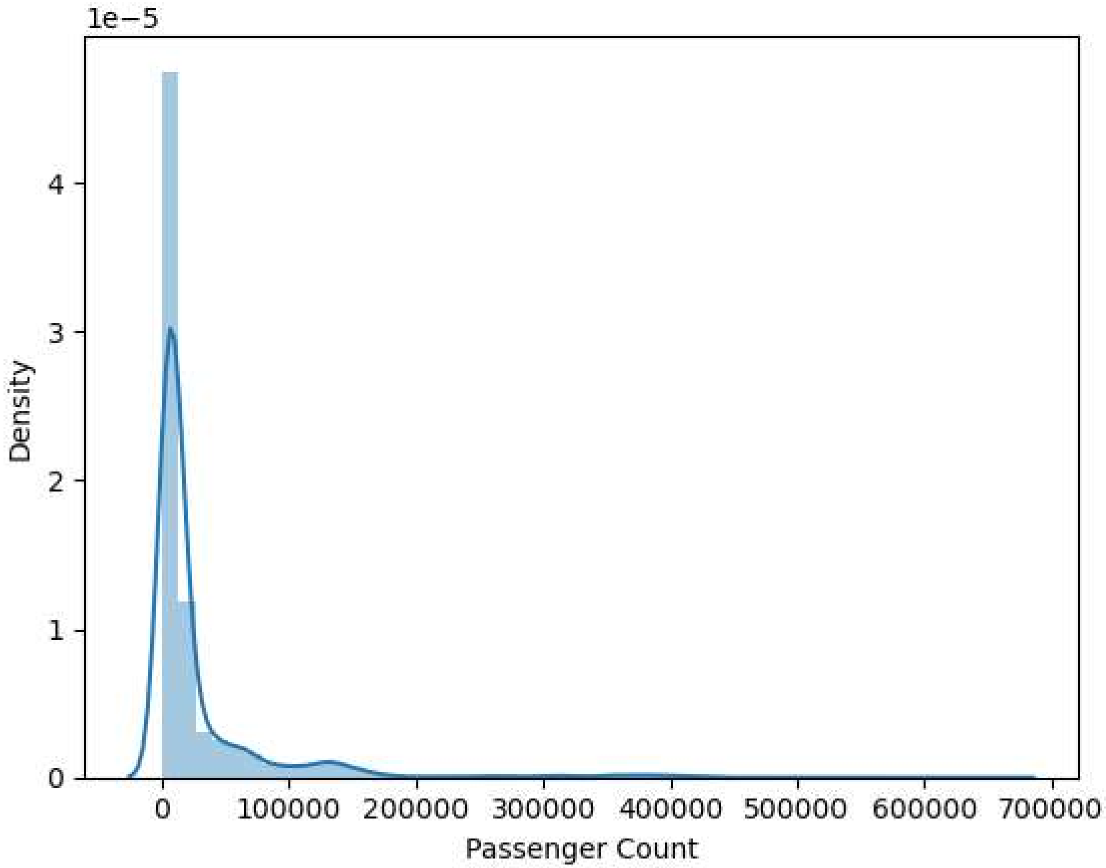
In [19]:
```
from scipy.stats import skew
```

In [20]:
```
for i in col:
    print(i)
    print(skew(df[i]))
    plt.figure()
    sns.distplot(df[i])
    plt.show()
```

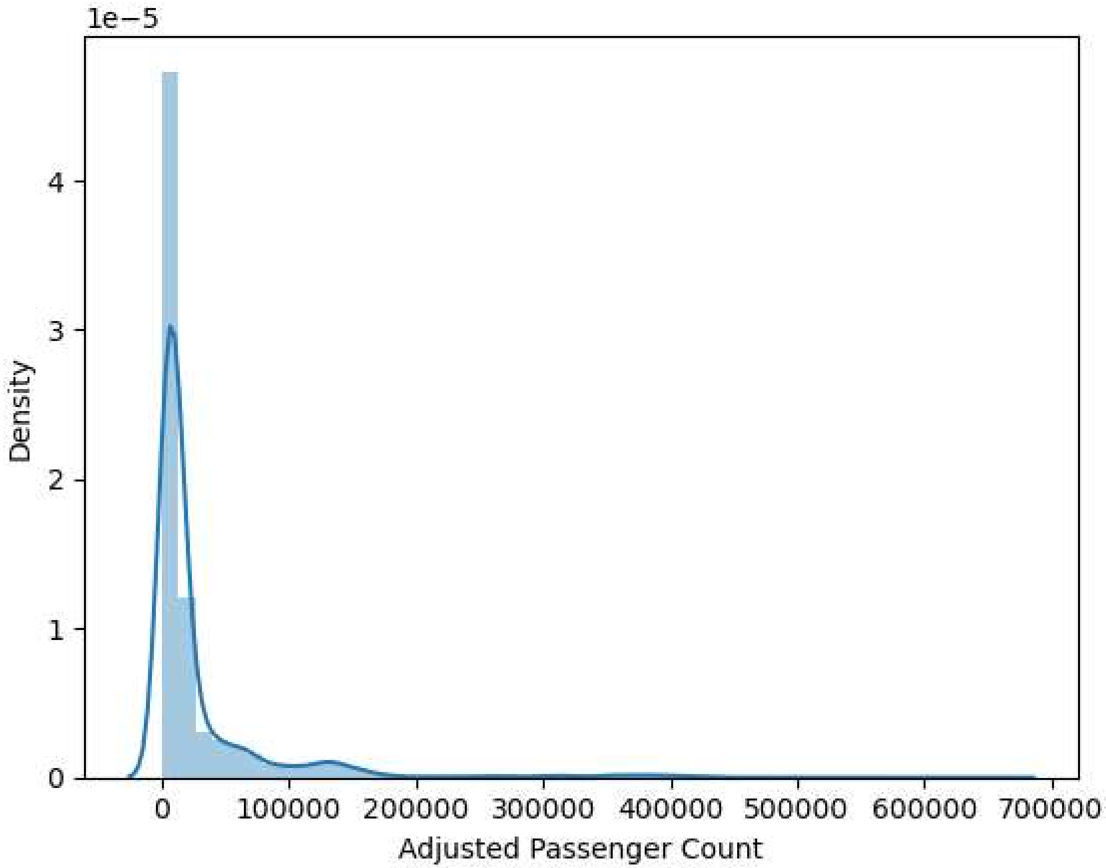
```
Activity Period
0.005277425987164216
```



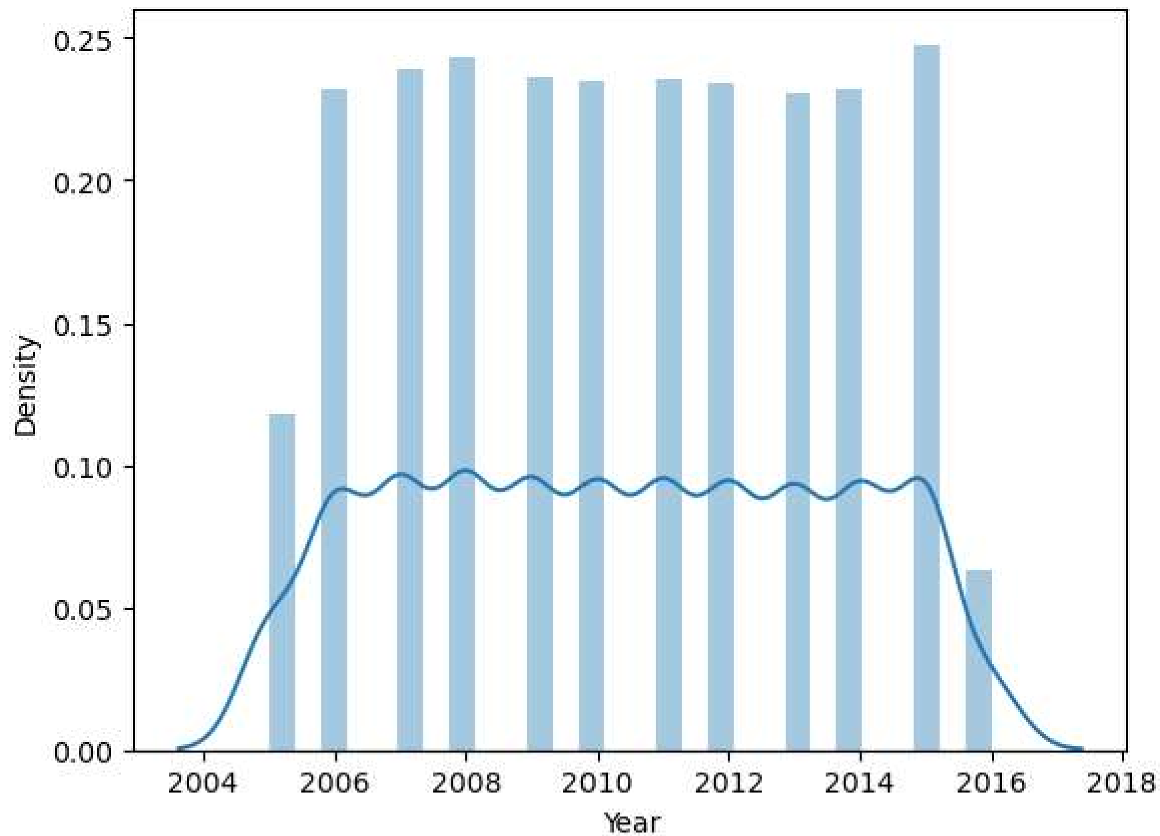

```
Passenger Count
4.3603199034139735
```




```
Adjusted Passenger Count
4.364279143574455
```




```
Year
0.004957949587778114
```



encoding

In [21]:
```
catcol=df.select_dtypes("object").columns
catcol
```

Out[21]:
```
Index([], dtype='object')
```

In [22]:
```
from sklearn.preprocessing import OrdinalEncoder
oe=OrdinalEncoder()
df[catcol]=oe.fit_transform(df[catcol])
df.head()
```

Out[22]:

	Activity Period	Operating Airline	Operating Airline IATA Code	Published Airline	Published Airline IATA Code	GEO Summary	GEO Region	Activity Type Code	Price Category Code	Terminal	Boarding Area	Passenger Count	Adjusted Activity Type Code	F
0	200507	0.0	60.0	0.0	54.0	0.0	8.0	0.0	0.0	2.0	1.0	27271	0.0	
1	200507	0.0	60.0	0.0	54.0	0.0	8.0	1.0	0.0	2.0	1.0	29131	1.0	
2	200507	0.0	60.0	0.0	54.0	0.0	8.0	2.0	0.0	2.0	1.0	5415	2.0	
3	200507	4.0	6.0	4.0	6.0	1.0	2.0	0.0	1.0	2.0	1.0	35156	0.0	
4	200507	4.0	6.0	4.0	6.0	1.0	2.0	1.0	1.0	2.0	1.0	34090	1.0	

scaler

In [23]:

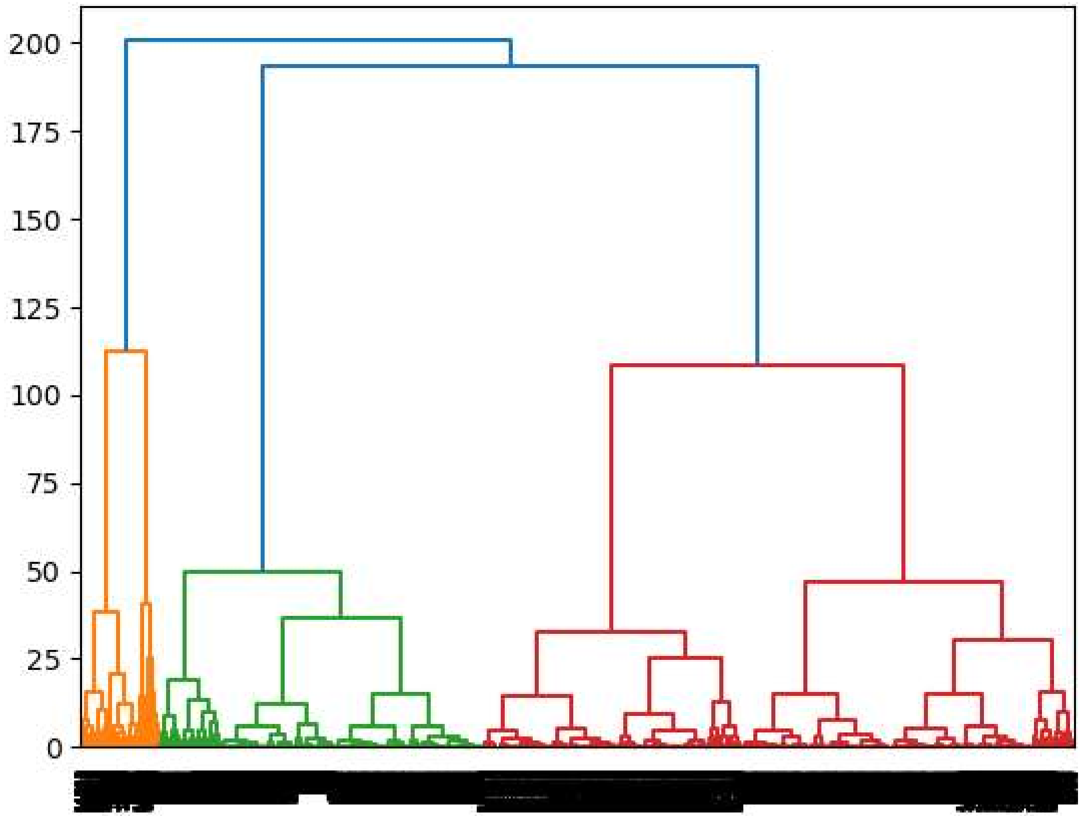
```
from sklearn.preprocessing import StandardScaler
sc = StandardScaler()
x = sc.fit_transform(x)
```

clustering

hierarchy clustering

In [24]:

```
from scipy.cluster import hierarchy as hi
lk = hi.linkage(x, method="ward")
ddg = hi.dendrogram(lk)
plt.show()
```



In [27]:
```
from sklearn.cluster import AgglomerativeClustering
hc = AgglomerativeClustering(n_clusters=5)
ylabel = hc.fit_predict(x)
```

In [28]:
```
df["label"]=ylabel
```

In [29]:
```
df.head()
```

Out[29]:

	Activity Period	Operating Airline	Operating Airline IATA Code	Published Airline	Published Airline IATA Code	GEO Summary	GEO Region	Activity Type Code	Price Category Code	Terminal	Boarding Area	Passenger Count	Adjusted Activity Type Code	F
0	200507	0.0	60.0	0.0	54.0	0.0	8.0	0.0	0.0	2.0	1.0	27271	0.0	
1	200507	0.0	60.0	0.0	54.0	0.0	8.0	1.0	0.0	2.0	1.0	29131	1.0	
2	200507	0.0	60.0	0.0	54.0	0.0	8.0	2.0	0.0	2.0	1.0	5415	2.0	
3	200507	4.0	6.0	4.0	6.0	1.0	2.0	0.0	1.0	2.0	1.0	35156	0.0	
4	200507	4.0	6.0	4.0	6.0	1.0	2.0	1.0	1.0	2.0	1.0	34090	1.0	

In [30]:

```
df.groupby("label")[["Published Airline", "Passenger Count"]].mean()
```

Out[30]:

label	Published Airline	Passenger Count
0	37.356411	18576.278370
1	60.617363	348929.652733
2	37.540610	13955.409469
3	40.827119	133642.480226
4	35.475651	13111.436485

In [31]:

```
df1=df.copy()
```

CLASIFICATION

In [32]:

```
df1.head(5)
```

Out[32]:

	Activity Period	Operating Airline	Operating Airline IATA Code	Published Airline	Published Airline IATA Code	GEO Summary	GEO Region	Activity Type Code	Price Category Code	Terminal	Boarding Area	Passenger Count	Adjusted Activity Type Code	F
0	200507	0.0	60.0	0.0	54.0	0.0	8.0	0.0	0.0	2.0	1.0	27271	0.0	
1	200507	0.0	60.0	0.0	54.0	0.0	8.0	1.0	0.0	2.0	1.0	29131	1.0	
2	200507	0.0	60.0	0.0	54.0	0.0	8.0	2.0	0.0	2.0	1.0	5415	2.0	
3	200507	4.0	6.0	4.0	6.0	1.0	2.0	0.0	1.0	2.0	1.0	35156	0.0	
4	200507	4.0	6.0	4.0	6.0	1.0	2.0	1.0	1.0	2.0	1.0	34090	1.0	

In [33]:

```
x=df1.iloc[:,:-1]
y=df1["label"]
```

MODEL TRAINING

In [34]:

```
from sklearn.model_selection import train_test_split
xtrain,xtest,ytrain,ytest=train_test_split(x,y,test_size=0.2,random_state=1)
```

In [35]:

```
def algorithm(algo):
    algo.fit(xtrain,ytrain)
    ypred=algo.predict(xtest)

    train=algo.score(xtrain,ytrain)
    test=algo.score(xtest,ytest)
    print(f"TrainingAccuracy: {train}\n Testin Accuracy: {test}\n\n")
    print(classification_report(ytest,ypred))
    return algo
from sklearn.metrics import classification_report
```

In [36]:

```
from sklearn.tree import DecisionTreeClassifier
from sklearn.linear_model import LogisticRegression
from sklearn.neighbors import KNeighborsClassifier
from sklearn.naive_bayes import BernoulliNB,MultinomialNB,GaussianNB
```

In [37]:

```
dt=algorithm(DecisionTreeClassifier())
```

```
TrainingAccuracy: 1.0
 Testin Accuracy: 0.9983344437041972


              precision    recall  f1-score   support

           0       1.00      1.00      1.00       976
           1       1.00      1.00      1.00        79
           2       1.00      1.00      1.00       990
           3       0.97      1.00      0.98       154
           4       1.00      1.00      1.00       803

    accuracy                           1.00      3002
   macro avg       0.99      1.00      1.00      3002
weighted avg       1.00      1.00      1.00      3002
```

In [38]:

```
lr=algorithm(LogisticRegression())
```

```
TrainingAccuracy: 0.43598500624739694
 Testin Accuracy: 0.41872085276482346


              precision    recall  f1-score   support

           0       0.42      0.25      0.31       976
           1       1.00      0.97      0.99        79
           2       0.36      0.80      0.50       990
           3       0.86      0.93      0.89       154
           4       0.00      0.00      0.00       803

    accuracy                           0.42      3002
   macro avg       0.53      0.59      0.54      3002
weighted avg       0.33      0.42      0.34      3002
```

In [39]:

```
knn=algorithm(KNeighborsClassifier())
```

```
TrainingAccuracy: 0.975177009579342
 Testin Accuracy: 0.9513657561625583


              precision    recall  f1-score   support

           0       0.96      0.98      0.97       976
           1       1.00      0.97      0.99        79
           2       0.92      0.96      0.94       990
           3       0.91      0.95      0.93       154
           4       0.98      0.92      0.95       803

    accuracy                           0.95      3002
   macro avg       0.96      0.95      0.95      3002
weighted avg       0.95      0.95      0.95      3002
```

In [40]:

```
bnb=algorithm(BernoulliNB())
```

```
TrainingAccuracy: 0.3271137026239067
 Testin Accuracy: 0.3104596935376416


              precision    recall  f1-score   support

           0       0.35      0.25      0.29       976
           1       0.00      0.00      0.00        79
           2       0.35      0.30      0.33       990
           3       0.17      0.61      0.26       154
           4       0.33      0.37      0.35       803

    accuracy                           0.31      3002
   macro avg       0.24      0.31      0.25      3002
weighted avg       0.33      0.31      0.31      3002
```

In [41]:

```
mnb=algorithm(MultinomialNB())
```

```
TrainingAccuracy: 0.34435651811745105
 Testin Accuracy: 0.34043970686209196


              precision    recall  f1-score   support

           0       0.34      0.19      0.24       976
           1       0.96      1.00      0.98        79
           2       0.29      0.04      0.07       990
           3       0.46      0.98      0.62       154
           4       0.30      0.71      0.42       803

    accuracy                           0.34      3002
   macro avg       0.47      0.58      0.47      3002
weighted avg       0.33      0.34      0.27      3002
```

In [42]:
```python
gnb=algorithm(GaussianNB())
```

```
TrainingAccuracy: 0.9645980841316119
 Testin Accuracy: 0.9643570952698202


              precision    recall  f1-score   support

           0       0.99      0.97      0.98       976
           1       0.84      1.00      0.91        79
           2       1.00      0.95      0.98       990
           3       0.70      0.91      0.79       154
           4       0.97      0.98      0.97       803

    accuracy                           0.96      3002
   macro avg       0.90      0.96      0.93      3002
weighted avg       0.97      0.96      0.97      3002
```