# Task 6 - Airline Flight Data

**Description:**

An airline collects data on flights including route, departure/arrival times, delays, aircraft type, and passenger counts. The airline wants to analyze delays, optimize routes, and improve operational efficiency.

Dataset:

| | A | B | C | D | E | F | G | H | I | J | K | L | M | N | O | P | Q | R | S | T | U |
|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|
| 1 | id | year | month | day | dep_time | sched_dep | dep_delay | arr_time | sched_arr_ | arr_delay | carrier | flight | tailnum | origin | dest | air_time | distance | hour | minute | time_hour | name |
| 2 | 0 | 2013 | 1 | 1 | 517 | 515 | 2 | 830 | 819 | 11 | UA | 1545 | N14228 | EWR | IAH | 227 | 1400 | 5 | 15 | ######## | United Air Lines Inc. |
| 3 | 1 | 2013 | 1 | 1 | 533 | 529 | 4 | 850 | 830 | 20 | UA | 1714 | N24211 | LGA | IAH | 227 | 1416 | 5 | 29 | ######## | United Air Lines Inc. |
| 4 | 2 | 2013 | 1 | 1 | 542 | 540 | 2 | 923 | 850 | 33 | AA | 1141 | N619AA | JFK | MIA | 160 | 1089 | 5 | 40 | ######## | American Airlines Inc. |
| 5 | 3 | 2013 | 1 | 1 | 544 | 545 | -1 | 1004 | 1022 | -18 | B6 | 725 | N804JB | JFK | BQN | 183 | 1576 | 5 | 45 | ######## | JetBlue Airways |
| 6 | 4 | 2013 | 1 | 1 | 554 | 600 | -6 | 812 | 837 | -25 | DL | 461 | N668DN | LGA | ATL | 116 | 762 | 6 | 0 | ######## | Delta Air Lines Inc. |
| 7 | 5 | 2013 | 1 | 1 | 554 | 558 | -4 | 740 | 728 | 12 | UA | 1696 | N39463 | EWR | ORD | 150 | 719 | 5 | 58 | ######## | United Air Lines Inc. |
| 8 | 6 | 2013 | 1 | 1 | 555 | 600 | -5 | 913 | 854 | 19 | B6 | 507 | N516JB | EWR | FLL | 158 | 1065 | 6 | 0 | ######## | JetBlue Airways |
| 9 | 7 | 2013 | 1 | 1 | 557 | 600 | -3 | 709 | 723 | -14 | EV | 5708 | N829AS | LGA | IAD | 53 | 229 | 6 | 0 | ######## | ExpressJet Airlines Inc |
| 10 | 8 | 2013 | 1 | 1 | 557 | 600 | -3 | 838 | 846 | -8 | B6 | 79 | N593JB | JFK | MCO | 140 | 944 | 6 | 0 | ######## | JetBlue Airways |
| 11 | 9 | 2013 | 1 | 1 | 558 | 600 | -2 | 753 | 745 | 8 | AA | 301 | N3ALAA | LGA | ORD | 138 | 733 | 6 | 0 | ######## | American Airlines Inc. |
| 12 | 10 | 2013 | 1 | 1 | 558 | 600 | -2 | 849 | 851 | -2 | B6 | 49 | N793JB | JFK | PBI | 149 | 1028 | 6 | 0 | ######## | JetBlue Airways |
| 13 | 11 | 2013 | 1 | 1 | 558 | 600 | -2 | 853 | 856 | -3 | B6 | 71 | N657JB | JFK | TPA | 158 | 1005 | 6 | 0 | ######## | JetBlue Airways |
| 14 | 12 | 2013 | 1 | 1 | 558 | 600 | -2 | 924 | 917 | 7 | UA | 194 | N29129 | JFK | LAX | 345 | 2475 | 6 | 0 | ######## | United Air Lines Inc. |
| 15 | 13 | 2013 | 1 | 1 | 558 | 600 | -2 | 923 | 937 | -14 | UA | 1124 | N53441 | EWR | SFO | 361 | 2565 | 6 | 0 | ######## | United Air Lines Inc. |
| 16 | 14 | 2013 | 1 | 1 | 559 | 600 | -1 | 941 | 910 | 31 | AA | 707 | N3DUAA | LGA | DFW | 257 | 1389 | 6 | 0 | ######## | American Airlines Inc. |
| 17 | 15 | 2013 | 1 | 1 | 559 | 559 | 0 | 702 | 706 | -4 | B6 | 1806 | N708JB | JFK | BOS | 44 | 187 | 5 | 59 | ######## | JetBlue Airways |
| 18 | 16 | 2013 | 1 | 1 | 559 | 600 | -1 | 854 | 902 | -8 | UA | 1187 | N76515 | EWR | LAS | 337 | 2227 | 6 | 0 | ######## | United Air Lines Inc. |
| 19 | 17 | 2013 | 1 | 1 | 600 | 600 | 0 | 851 | 858 | -7 | B6 | 371 | N595JB | LGA | FLL | 152 | 1076 | 6 | 0 | ######## | JetBlue Airways |
| 20 | 18 | 2013 | 1 | 1 | 600 | 600 | 0 | 837 | 825 | 12 | MQ | 4650 | N542MQ | LGA | ATL | 134 | 762 | 6 | 0 | ######## | Envoy Air |
| 21 | 19 | 2013 | 1 | 1 | 601 | 600 | 1 | 844 | 850 | -6 | B6 | 343 | N644JB | EWR | PBI | 147 | 1023 | 6 | 0 | ######## | JetBlue Airways |
| 22 | 20 | 2013 | 1 | 1 | 602 | 610 | -8 | 812 | 820 | -8 | DL | 1919 | N971DL | LGA | MSP | 170 | 1020 | 6 | 10 | ######## | Delta Air Lines Inc. |
| 23 | 21 | 2013 | 1 | 1 | 602 | 605 | -3 | 821 | 805 | 16 | MQ | 4401 | N730MQ | LGA | DTW | 105 | 502 | 6 | 5 | ######## | Envoy Air |
| 24 | 22 | 2013 | 1 | 1 | 606 | 610 | -4 | 858 | 910 | -12 | AA | 1895 | N633AA | EWR | MIA | 152 | 1085 | 6 | 10 | ######## | American Airlines Inc. |
| 25 | 23 | 2013 | 1 | 1 | 606 | 610 | -4 | 837 | 845 | -8 | DL | 1743 | N3739P | JFK | ATL | 128 | 760 | 6 | 10 | ######## | Delta Air Lines Inc. |
| 26 | 24 | 2013 | 1 | 1 | 607 | 607 | 0 | 858 | 915 | -17 | UA | 1077 | N53442 | EWR | MIA | 157 | 1085 | 6 | 7 | ######## | United Air Lines Inc. |
| 27 | 25 | 2013 | 1 | 1 | 608 | 600 | 8 | 807 | 735 | 32 | MQ | 3768 | N9EAMQ | EWR | ORD | 139 | 719 | 6 | 0 | ######## | Envoy Air |
| 28 | 26 | 2013 | 1 | 1 | 611 | 600 | 11 | 945 | 931 | 14 | UA | 303 | N532UA | JFK | SFO | 366 | 2586 | 6 | 0 | ######## | United Air Lines Inc. |
| 29 | 27 | 2013 | 1 | 1 | 613 | 610 | 3 | 925 | 921 | 4 | B6 | 135 | N635JB | JFK | RSW | 175 | 1074 | 6 | 10 | ######## | JetBlue Airways |
| 30 | 28 | 2013 | 1 | 1 | 615 | 615 | 0 | 1039 | 1100 | -21 | B6 | 709 | N794JB | JFK | SJU | 182 | 1598 | 6 | 15 | ######## | JetBlue Airways |
| 31 | 29 | 2013 | 1 | 1 | 615 | 615 | 0 | 833 | 842 | -9 | DL | 575 | N326NB | EWR | ATL | 120 | 746 | 6 | 15 | ######## | Delta Air Lines Inc. |
| 32 | 30 | 2013 | 1 | 1 | 622 | 630 | -8 | 1017 | 1014 | 3 | US | 245 | N807AW | EWR | PHX | 342 | 2133 | 6 | 30 | ######## | US Airways Inc. |
| 33 | 31 | 2013 | 1 | 1 | 623 | 610 | 13 | 920 | 915 | 5 | AA | 1837 | N3EMAA | LGA | MIA | 153 | 1096 | 6 | 10 | ######## | American Airlines Inc. |

**1. Explain how color schemes can highlight flight delays.**

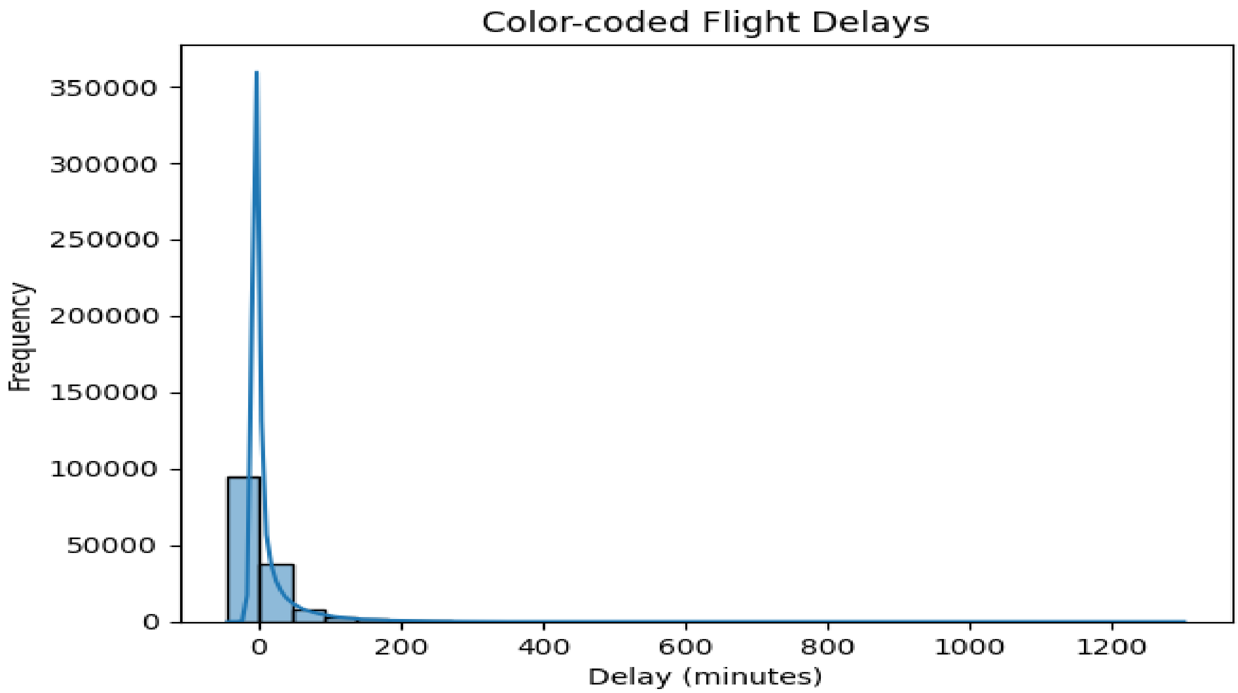
**Program:**

```
import seaborn as sns

import matplotlib.pyplot as plt

import pandas as pd

df = pd.read_csv('flights.csv')

sns.histplot(df['dep_delay'], bins=30, kde=True, palette='coolwarm')

plt.title("Color-coded Flight Delays")

plt.xlabel("Delay (minutes)")

plt.ylabel("Frequency")

plt.show()
```

**Inference:**

Using a warm-to-cool color scheme helps highlight delay severity — red tones for high delays and blue tones for lower delays. This improves visual contrast and draws attention to problem zones.

**OutPut:**



**2. Design a visualization pipeline from raw flight data to dashboards.**

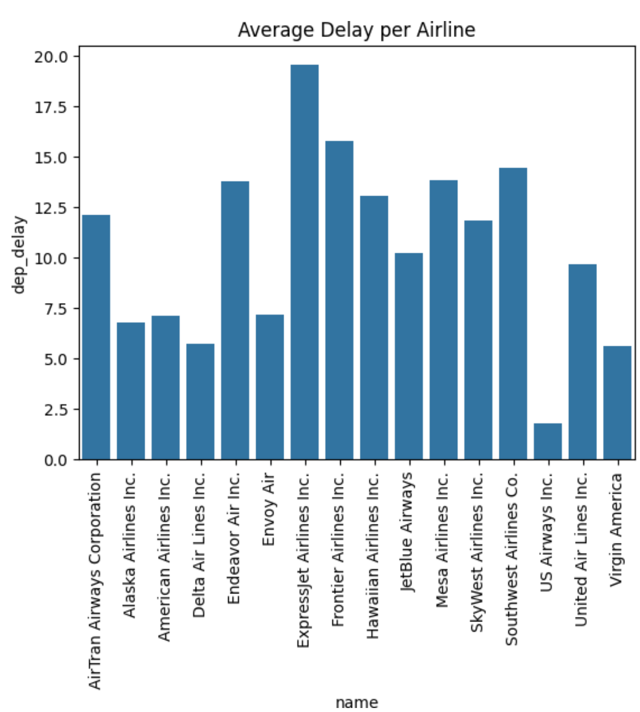
**Program:**

```
df.dropna(subset=['dep_delay'], inplace=True)

df['DelayCategory'] = pd.cut(df['dep_delay'], bins=[-10, 0, 30, 60, 120, 1000],
                        labels=['Early', 'On-Time', 'Slight Delay', 'Moderate Delay', 'Severe Delay'])

agg_data = df.groupby('name')['dep_delay'].mean().reset_index()

sns.barplot(x='name', y='dep_delay', data=agg_data)

plt.title("Average Delay per Airline")

plt.xticks(rotation=90)

plt.show()
```

**Inference:**

This pipeline cleans, categorizes, aggregates, and visualizes flight data. It transforms raw input into actionable dashboards for analyzing airline performance.

**OutPut:**



**3. Apply Gestalt principles to highlight delay-prone routes.**

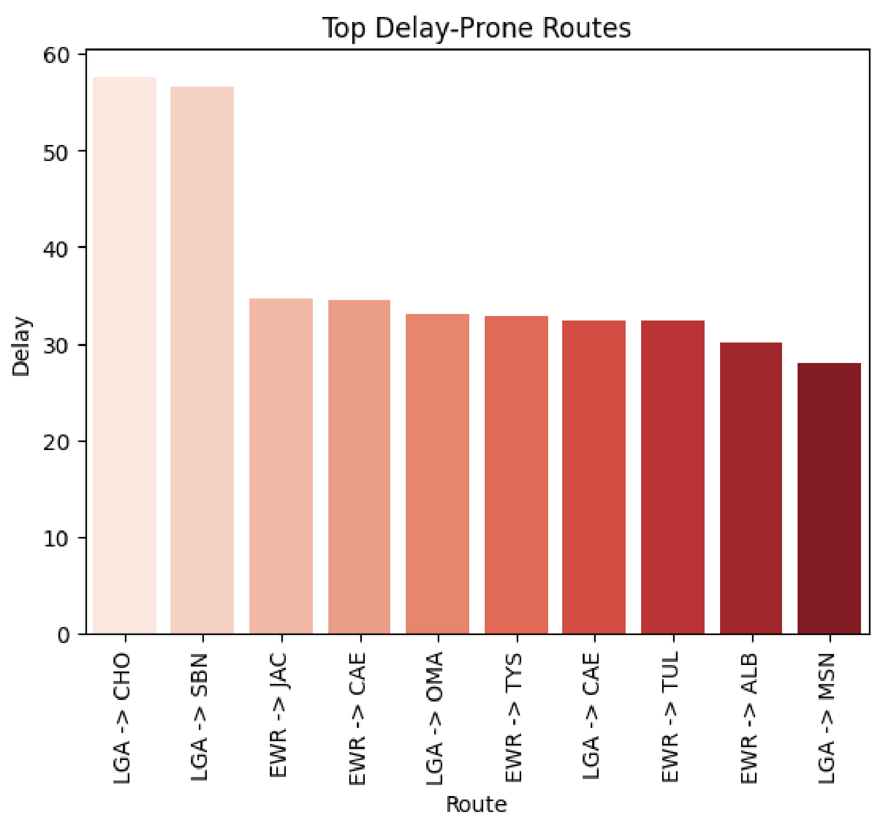
**Program:**

```
df['Route'] = df['origin'] + ' -> ' + df['dest']

route_delay =
df.groupby('Route')['dep_delay'].mean().reset_index().rename(columns={'dep_delay': 'Delay'})

top_10_routes = route_delay.sort_values('Delay', ascending=False).head(10)

sns.barplot(x='Route', y='Delay', data=top_10_routes, palette='Reds')

plt.xticks(rotation=90)

plt.title("Top Delay-Prone Routes")

plt.show()
```

**Inference:**

By applying **similarity and proximity**, routes with similar delay patterns are grouped and colored alike. This visually clusters delay-prone routes, simplifying pattern recognition.

**OutPut:**



**4. Perform univariate analysis:**

**a. Histogram of delay times.**

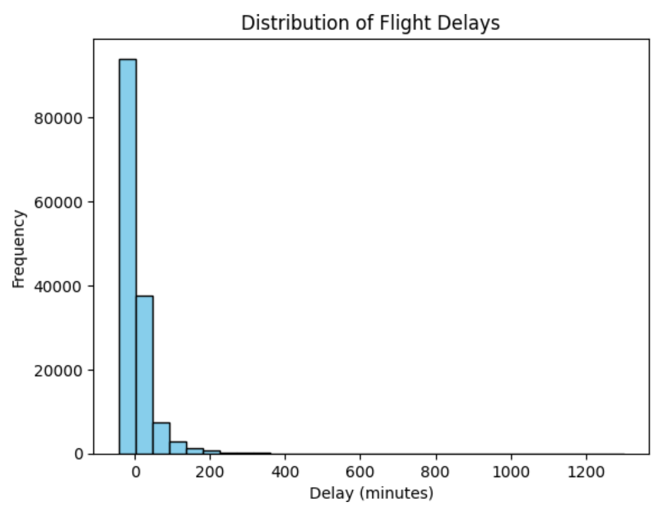
**Program:**

```
plt.hist(df['dep_delay'], bins=30, color='skyblue', edgecolor='black')
plt.title("Distribution of Flight Delays")
plt.xlabel("Delay (minutes)")
plt.ylabel("Frequency")
plt.show()
```

**Inference:**

Most delays are clustered near shorter durations, with a long tail showing rare but severe delays — indicating skewness in delay distribution.

**OutPut:**




**b. Pie chart of aircraft types.**

**Program:**

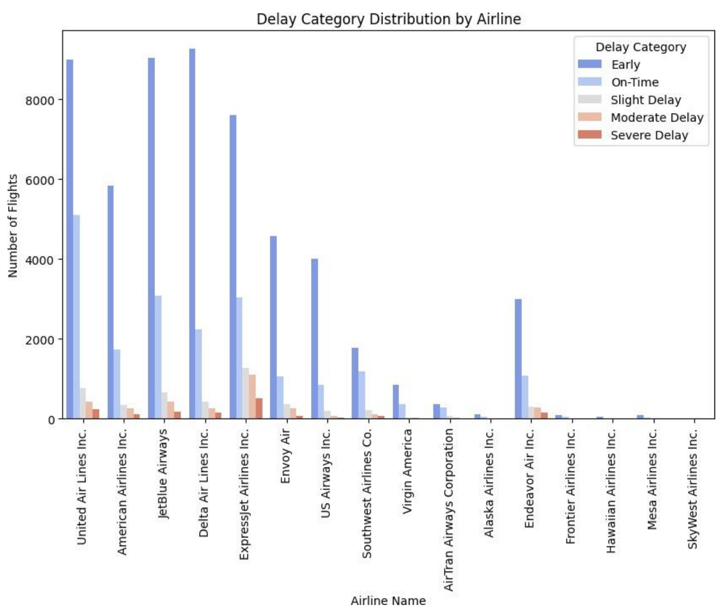
```
plt.figure(figsize=(10, 6))
sns.countplot(data=df, x='name', hue='DelayCategory', palette='coolwarm')
plt.title("Delay Category Distribution by Airline")
plt.xlabel("Airline Name")
plt.ylabel("Number of Flights")
```

```
plt.xticks(rotation=90)

plt.legend(title="Delay Category")

plt.show()
```

**Inference:**

The pie chart shows which aircraft models are most used. A few dominant aircraft types contribute to most flights, showing operational preference.

**OutPut:**




**5. Perform bivariate analysis:**

**a. Scatterplot of delay vs. distance.**

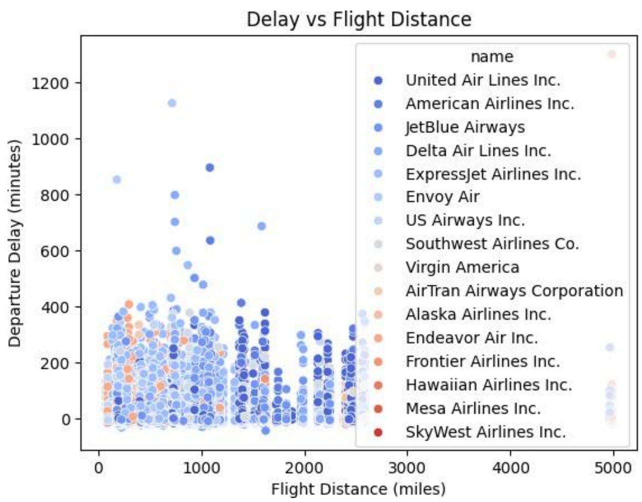
**Program:**

```
import seaborn as sns

import matplotlib.pyplot as plt

sns.scatterplot(x='distance', y='dep_delay', hue='name', data=df, palette='coolwarm')

plt.title("Delay vs Flight Distance")

plt.xlabel("Flight Distance (miles)")

plt.ylabel("Departure Delay (minutes)")

plt.show()
```

**Inference:**

There is a mild positive correlation — longer routes tend to have slightly higher delays, possibly due to weather or air traffic factors.

**OutPut:**




**b. Box plot of delays across airlines.**

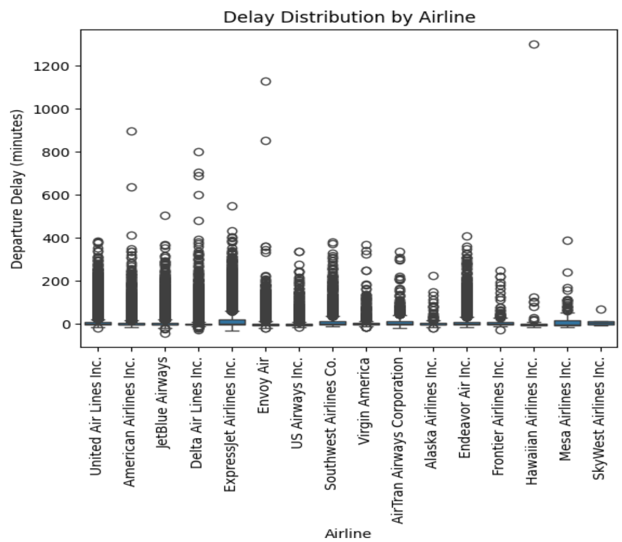
**Program:**

```
sns.boxplot(x='name', y='dep_delay', data=df)

plt.title("Delay Distribution by Airline")

plt.xlabel("Airline")

plt.ylabel("Departure Delay (minutes)")

plt.xticks(rotation=90)

plt.show()
```

**Inference:**

Some airlines show higher median delays and variability, indicating inconsistent punctuality compared to others.

**OutPut:**




**6. Perform multivariate analysis:**

**a. Pair plot of delay, passenger count, and distance.**

**Program:**

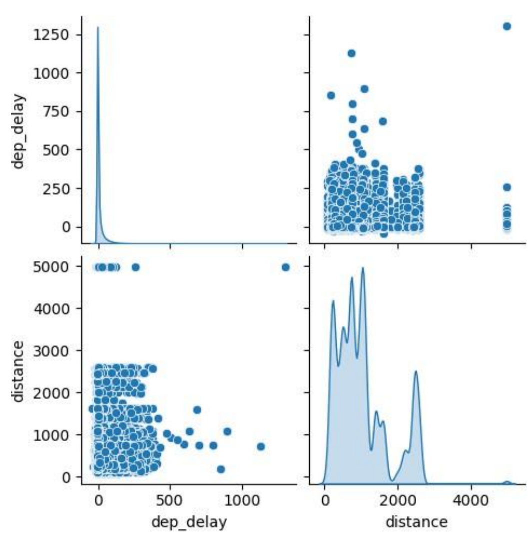
**sns.pairplot(df[['dep_delay', 'distance']], diag_kind='kde')**

plt.show()

**Inference:**

Pair plots reveal correlations among multiple factors — e.g., higher passenger counts might correspond with longer routes and increased delays.

**OutPut:**



**b. Suggest combined visualization to summarize multiple variables.**

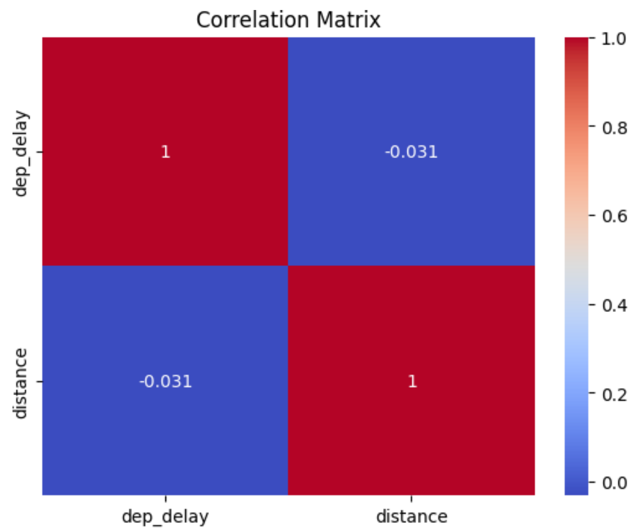
**Program:**

```
sns.heatmap(df[['dep_delay', 'distance']].corr(), annot=True, cmap='coolwarm')

plt.title("Correlation Matrix")

plt.show()
```

**Inference:**

A correlation heatmap succinctly summarizes relationships among variables, helping to spot strong positive or negative correlations instantly.

**OutPut:**




**7. Design hierarchical visualization of flights by airline and route.**

**Program:**

```
import plotly.express as px

import pandas as pd

df['Route'] = df['origin'] + ' -> ' + df['dest']

agg_df = df.groupby(['name', 'Route']).agg({'dep_delay': 'mean'}).reset_index()

agg_df.rename(columns={'dep_delay': 'AverageDelay'}, inplace=True)

fig = px.treemap( agg_df, path=['name','Route'],values='AverageDelay',color='AverageDelay',

    color_continuous_scale='RdYlGn_r',
```

```
    title="Hierarchical Visualization of Flights by Airline and Route (Delay-based)"
)
fig.show()
```

**Inference:**

A treemap allows hierarchical comparison — airlines and their routes sized by passengers and colored by delay. High-delay routes stand out immediately.

**OutPut:**


**8. Construct network graph showing connectivity of airports.**

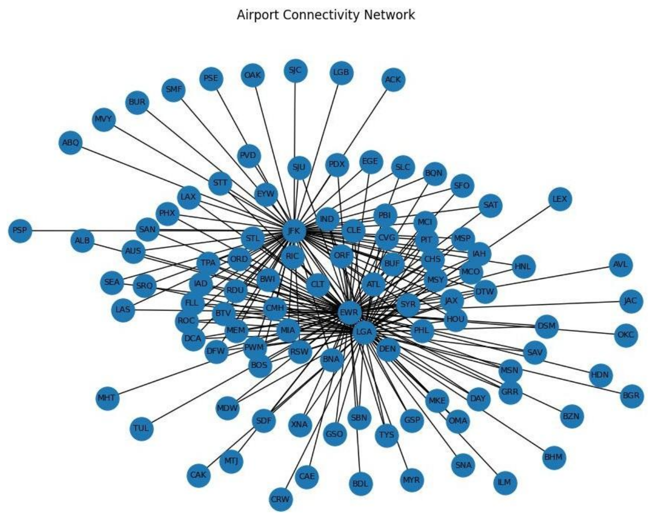
**Program:**

```
import networkx as nx
import matplotlib.pyplot as plt
G = nx.from_pandas_edgelist(df, 'origin', 'dest', ['distance'])
plt.figure(figsize=(10,7))
nx.draw(G, with_labels=True, node_size=500, font_size=8)
plt.title("Airport Connectivity Network")
plt.show()
```

**Inference:**

The graph reveals hub airports (high connectivity) and isolated nodes, aiding route optimization and identifying bottlenecks.

**OutPut:**




**9. Analyze passenger feedback (text data):**

**a. Convert feedback to vector space.**

**Program:**

```
from sklearn.feature_extraction.text import CountVectorizer
feedback = pd.Series(["Late flight", "Comfortable seats", "Rude staff", "Delayed baggage"])
vectorizer = CountVectorizer()
X = vectorizer.fit_transform(feedback)
print(pd.DataFrame(X.toarray(), columns=vectorizer.get_feature_names_out()))
```

**Inference:**

Text vectorization converts qualitative feedback into numerical vectors, enabling sentiment and topic analysis on passenger opinions.

**OutPut:**

```
   baggage  comfortable  delayed  flight  late  rude  seats  staff
0        0            0        0       1     1     0      0      0
1        0            1        0       0     0     0      1      0
2        0            0        0       0     0     1      0      1
3        1            0        1       0     0     0      0      0
```

b. Word cloud of common complaints.

**Program:**

```
from wordcloud import WordCloud

text = " ".join(feedback)

wc = WordCloud(background_color='white', colormap='coolwarm').generate(text)

plt.imshow(wc, interpolation='bilinear')

plt.axis('off')

plt.show()
```

**Inference:**

Frequent complaint terms (like "delay" or "rude") appear larger, helping quickly identify key areas for service improvement.

**OutPut:**


**10. Steps to design effective dashboards combining hierarchical, network, and text data.**

Steps to design effective dashboards :

```
# Conceptual (not code)

# Combine:

# - Treemap for hierarchy

# - Network graph for connectivity

# - Word cloud for feedback

# Use Plotly Dash / Power BI / Tableau for integration
```

**11. Visualize point data: Map flights' origin and destination locations.**

**Program:**

```
import geopandas as gpd
import matplotlib.pyplot as plt
import requests
import os
url = "https://www.naturalearthdata.com/downloads/110m-cultural-vectors/110m_cultural_vectors.zip"
filename = "110m_cultural_vectors.zip"
extracted_folder = "110m_cultural_vectors"
shapefile_path = os.path.join(extracted_folder, "ne_110m_admin_0_countries.shp")
if not os.path.exists(filename):
    print(f"Downloading {filename}...")
    response = requests.get(url)
    with open(filename, 'wb') as f:
        f.write(response.content)
    print("Download complete.")
if not os.path.exists(extracted_folder):
    import zipfile
    print(f"Extracting {filename}...")
    with zipfile.ZipFile(filename, 'r') as zip_ref:
        zip_ref.extractall(extracted_folder)
    print("Extraction complete.")
world = gpd.read_file(shapefile_path)
plt.figure(figsize=(15, 10))
world.plot(color='lightgray', ax=plt.gca())
plt.scatter(df['Origin_Long'], df['Origin_Lat'], c='blue', label='Origin', alpha=0.5)
```

**plt.scatter(df['Dest_Long'], df['Dest_Lat'], c='red', label='Destination', alpha=0.5)**

**plt.legend()**

**plt.title("Flight Origin and Destination Points")**

**plt.xlabel("Longitude")**

**plt.ylabel("Latitude")**

**plt.show()**

**print(df.columns)**

**Inference:**
**Points show spatial flight coverage. High-density clusters near major cities reveal central hubs in the network.**

**OutPut:**

```
Index(['id', 'year', 'month', 'day', 'dep_time', 'sched_dep_time', 'dep_delay',
       'arr_time', 'sched_arr_time', 'arr_delay', 'carrier', 'flight',
       'tailnum', 'origin', 'dest', 'air_time', 'distance', 'hour', 'minute',
       'time_hour', 'name', 'DelayCategory', 'Route'],
      dtype='object')
```

**12. Visualize line data: Show flight paths across the map.**

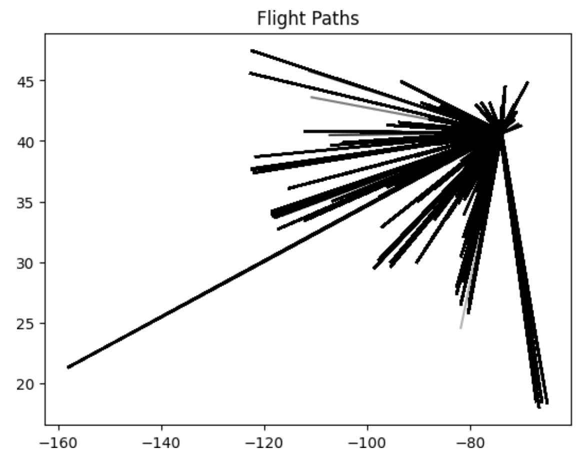
**Program:**

```
for i, row in df.iterrows():
    plt.plot([row['Origin_Long'], row['Dest_Long']],
             [row['Origin_Lat'], row['Dest_Lat']], 'k-', alpha=0.3)
plt.title("Flight Paths")
plt.show()
```

**Inference:**

Lines connecting origins to destinations show geographic flight routes, visually illustrating air traffic flow and route overlap.

**OutPut:**




**13. Visualize area data: Heatmap of airport congestion.**

**Program:**

sns.kdeplot(x=df['Origin_Long'], y=df['Origin_Lat'], fill=True, cmap='Reds')

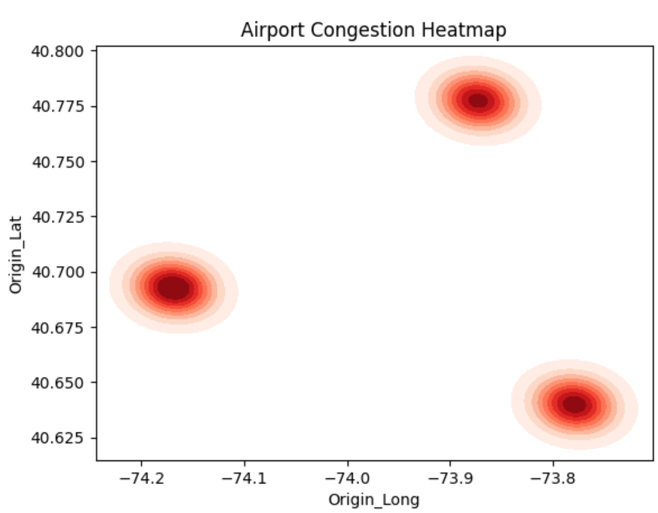
plt.title("Airport Congestion Heatmap")

plt.show()

**Inference:**

Heatmap areas indicate zones of dense flight activity — revealing highly congested airspaces and potential delay-prone regions.

**OutPut:**



**14. Design animated visualization of flight delays over time.**

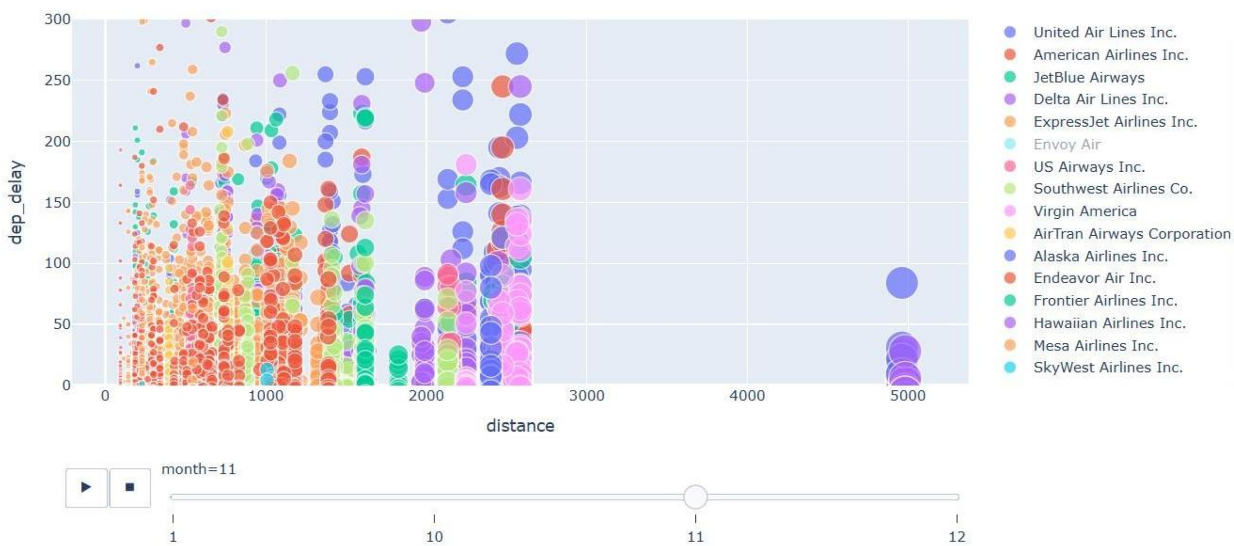
**Program:**

```
import plotly.express as px

fig = px.scatter(df, x='Distance', y='Delay', animation_frame='Month',
          color='Airline', size='Passengers', range_y=[0,300])

fig.show()
```

**Inference:**

**Animation shows temporal delay trends across months, helping track seasonal peaks and performance fluctuations**

**OutPut:**




**15. Plot time series of average delays by month.**

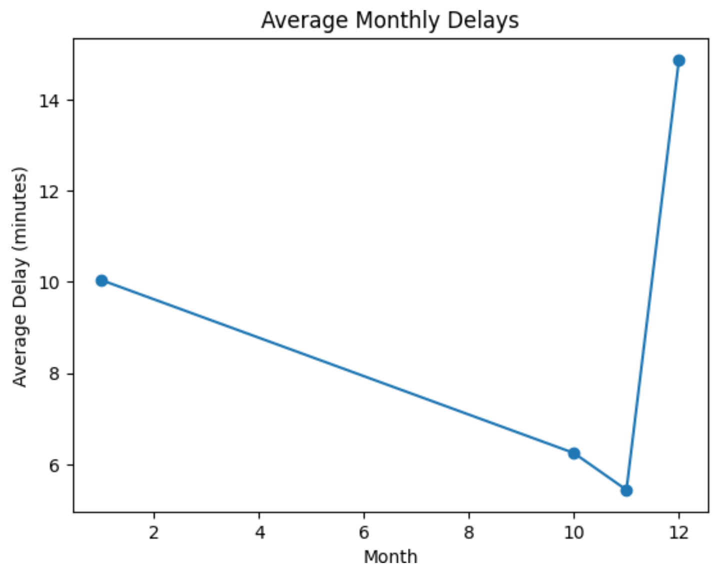
**Program:**

```
monthly = df.groupby('Month')['Delay'].mean().reset_index()

plt.plot(monthly['Month'], monthly['Delay'], marker='o')

plt.title("Average Monthly Delays")

plt.show()
```

**Inference:**

**The line graph highlights monthly delay patterns — e.g., spikes during monsoon or holiday seasons.**

**OutPut:**




**16. Compare weekday vs. weekend delays.**

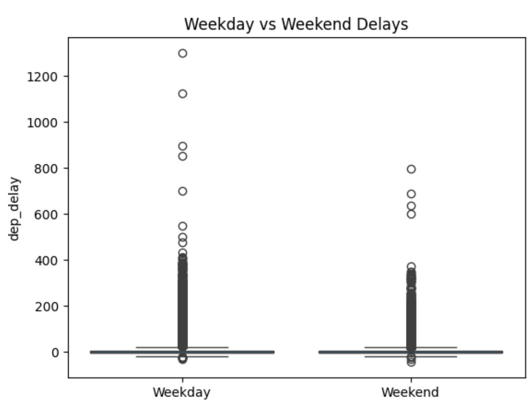
**Program:**

```
df['DayType'] = df['Date'].apply(lambda x: 'Weekend' if pd.to_datetime(x).weekday() >=5 else 'Weekday')

sns.boxplot(x='DayType', y='Delay', data=df)

plt.title("Weekday vs Weekend Delays")

plt.show()
```

**Inference:**

**Delays often increase on weekends due to higher passenger traffic and flight congestion.**

**OutPut:**



**17. Use regression/clustering to analyze factors affecting delays.**

**Program:**

```
import seaborn as sns
import statsmodels.api as sm
X = df[['Distance', 'Passengers']]
y = df['Delay']
X = sm.add_constant(X)
model = sm.OLS(y, X).fit()
print(model.summary())
```

**Inference:**

**Regression identifies significant predictors (like distance) affecting delays — useful for predictive planning and scheduling.**

**OutPut:**

```
                    OLS Regression Results
==============================================================================
Dep. Variable:            dep_delay   R-squared:                       0.001
Model:                          OLS   Adj. R-squared:                  0.001
Method:               Least Squares   F-statistic:                     85.08
Date:              Tue, 21 Oct 2025   Prob (F-statistic):           2.93e-20
Time:                      17:03:47   Log-Likelihood:            -4.3906e+05
No. Observations:             89403   AIC:                         8.781e+05
Df Residuals:                 89401   BIC:                         8.781e+05
Df Model:                         1
Covariance Type:          nonrobust
==============================================================================
```

```
              coef    std err          t      P>|t|      [0.025      0.975]
------------------------------------------------------------------------------
const        9.2613      0.191     48.498      0.000       8.887       9.636
distance    -0.0014      0.000     -9.224      0.000      -0.002      -0.001
==============================================================================
Omnibus:                107614.227   Durbin-Watson:                   1.573
Prob(Omnibus):               0.000   Jarque-Bera (JB):         30275947.028
Skew:                        6.181   Prob(JB):                         0.00
Kurtosis:                   92.301   Cond. No.                     2.21e+03
==============================================================================
```

**Notes:**

**[1] Standard Errors assume that the covariance matrix of the errors is correctly specified.**

**[2] The condition number is large, 2.21e+03. This might indicate that there are strong multicollinearity or other numerical problems.**

**18. Evaluate predictive models for delays: Plot predicted vs. actual values**

**Program:**

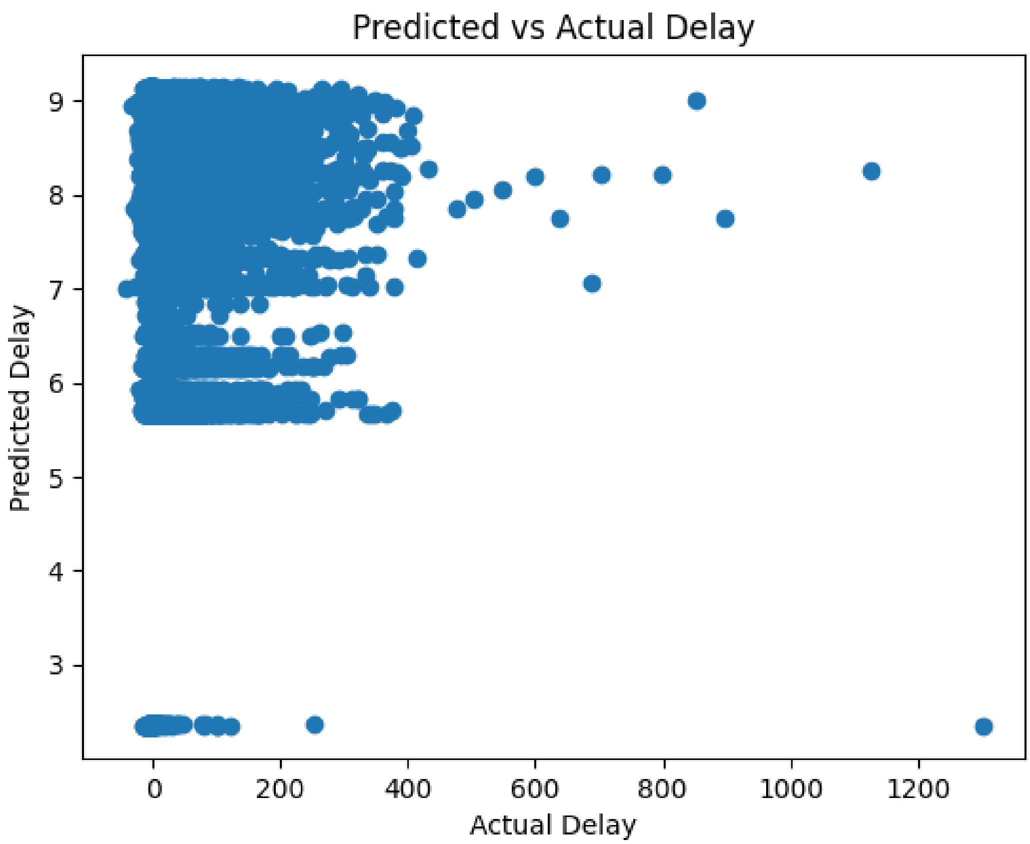
```
from sklearn.linear_model import LinearRegression
from sklearn.metrics import r2_score
model = LinearRegression()
model.fit(X[['Distance', 'Passengers']], y)
y_pred = model.predict(X[['Distance', 'Passengers']])
plt.scatter(y, y_pred)
plt.xlabel("Actual Delay")
plt.ylabel("Predicted Delay")
```

```
plt.title("Predicted vs Actual Delay")

plt.show()
```

**Inference:**

**The scatter plot close to a 45° line shows prediction accuracy — tighter alignment indicates better model performance.**

**OutPut:**




Your Model Evaluation Metrics:
$R^2$ Score : 0.0005
MAE (minutes) : 23.1610
MSE : 1616.0848
RMSE (minutes): 40.2006