airlines-on-time-performance

July 27, 2024

1 Airlines on-time Performance

Student Name: Naveen Kavitha Gunasekaran

1.1 The DS Problem

The Data Science problem involves analyzing the 2019 airline on-time performance data for flights originating from or departing to Arizona (AZ), Nevada (NV), and California (CA) to uncover patterns and factors influencing flight delays. The dataset includes metrics such as flight dates, carrier codes, flight numbers, origin and destination airports, departure and arrival times, delays, elapsed time, and distances. The objectives are to understand delay patterns, compare carrier performance, analyze the relationship between flight duration, distance, and delays, examine temporal variations, and assess airport-specific delays. The goal is to derive actionable insights to improve on-time performance through data cleaning, exploratory data analysis, comparative analysis, time series analysis, geospatial analysis, and predictive modeling.

1.2 Prepare the Data

Evaluate and Convert Data Types

[1]:
```
import pandas as pd
#importing libraries
import pandas as pd

# Load the data into a dataframe
file_path = 'C:/Users/navee/Downloads/2019_ONTIME_REPORTING_FSW.csv'
data = pd.read_csv(file_path)
data.tail(30)
```

[1]:
```
            FL_DATE CARRIER_CODE TAIL_NUM  FL_NUM ORIGIN ORIGIN_ST DEST  \
1897473  2019-01-31           UA   N77867     264    DEN        CO  SFO
1897474  2019-01-31           UA   N77542     264    SFO        CA  IAH
1897475  2019-01-31           UA   N37508     261    LAX        CA  MCO
1897476  2019-01-31           UA   N34455     258    SFO        CA  AUS
1897477  2019-01-31           UA   N771UA     257    DEN        CO  SFO
1897478  2019-01-31           UA   N76508     257    SFO        CA  LAX
1897479  2019-01-31           UA   N87512     256    DEN        CO  SFO
1897480  2019-01-31           UA   N471UA     256    SFO        CA  LAX
1897481  2019-01-31           UA   N422UA     254    PHX        AZ  DEN
```

1897482	2019-01-31	UA	N822UA	251	SAN	CA	IAH
1897483	2019-01-31	UA	N37504	250	IAH	TX	SFO
1897484	2019-01-31	UA	NaN	248	PHX	AZ	ORD
1897485	2019-01-31	UA	N596UA	247	LAX	CA	EWR
1897486	2019-01-31	UA	N17133	242	SFO	CA	BOS
1897487	2019-01-31	UA	N491UA	239	BUR	CA	SFO
1897488	2019-01-31	UA	N422UA	237	DEN	CO	PHX
1897489	2019-01-31	UA	N38403	235	PHX	AZ	IAH
1897490	2019-01-31	UA	N47505	234	MCO	FL	LAX
1897491	2019-01-31	UA	N56859	234	SFO	CA	MCO
1897492	2019-01-31	UA	N411UA	230	EWR	NJ	PHX
1897493	2019-01-31	UA	N37263	230	SNA	CA	ORD
1897494	2019-01-31	UA	N19141	229	IAD	VA	SAN
1897495	2019-01-31	UA	N69840	223	SFO	CA	DEN
1897496	2019-01-31	UA	NaN	222	ORD	IL	SFO
1897497	2019-01-31	UA	N481UA	214	SEA	WA	SFO
1897498	2019-01-31	UA	N73256	209	SNA	CA	SFO
1897499	2019-01-31	UA	N39416	208	IAD	VA	LAX
1897500	2019-01-31	UA	N17104	207	BOS	MA	SFO
1897501	2019-01-31	UA	N813UA	205	SFO	CA	PDX
1897502	2019-01-31	UA	N75861	204	ORD	IL	LAX

	DEST_ST	DEP_TIME	DEP_DELAY	ARR_TIME	ARR_DELAY	ELAPSED_TIME	\
1897473	CA	1227.0	37.0	1408.0	32.0	161.0	
1897474	TX	1536.0	24.0	2116.0	24.0	220.0	
1897475	FL	1111.0	15.0	1913.0	36.0	302.0	
1897476	TX	731.0	0.0	1255.0	0.0	204.0	
1897477	CA	1857.0	0.0	2017.0	0.0	140.0	
1897478	CA	2233.0	3.0	2357.0	0.0	84.0	
1897479	CA	553.0	0.0	714.0	0.0	141.0	
1897480	CA	1229.0	89.0	1420.0	102.0	111.0	
1897481	CO	1104.0	0.0	1244.0	0.0	100.0	
1897482	TX	1022.0	0.0	1520.0	0.0	178.0	
1897483	CA	1008.0	33.0	1211.0	6.0	243.0	
1897484	IL	NaN	NaN	NaN	NaN	NaN	
1897485	NJ	706.0	0.0	1533.0	3.0	327.0	
1897486	MA	1351.0	0.0	2223.0	0.0	332.0	
1897487	CA	710.0	5.0	832.0	0.0	82.0	
1897488	AZ	758.0	0.0	943.0	0.0	105.0	
1897489	TX	1520.0	0.0	1849.0	0.0	149.0	
1897490	CA	1906.0	0.0	2141.0	0.0	335.0	
1897491	FL	832.0	0.0	1637.0	0.0	305.0	
1897492	AZ	2029.0	59.0	2346.0	31.0	317.0	
1897493	IL	647.0	0.0	1257.0	3.0	250.0	
1897494	CA	836.0	0.0	1100.0	0.0	324.0	
1897495	CO	1039.0	0.0	1411.0	0.0	152.0	
1897496	CA	NaN	NaN	NaN	NaN	NaN	

```
1897497       CA    1942.0         0.0    2143.0         0.0          121.0
1897498       CA     750.0         0.0     911.0         0.0           81.0
1897499       CA    1855.0         0.0    2148.0         0.0          353.0
1897500       CA     802.0         2.0    1128.0         0.0          386.0
1897501       OR     604.0         0.0     802.0         0.0          118.0
1897502       CA     813.0        18.0    1028.0         0.0          255.0

         DISTANCE
1897473       967
1897474      1635
1897475      2218
1897476      1504
1897477       967
1897478       337
1897479       967
1897480       337
1897481       602
1897482      1303
1897483      1635
1897484      1440
1897485      2454
1897486      2704
1897487       326
1897488       602
1897489      1009
1897490      2218
1897491      2446
1897492      2133
1897493      1726
1897494      2253
1897495       967
1897496      1846
1897497       679
1897498       372
1897499      2288
1897500      2704
1897501       550
1897502      1744
```

[2]:
```
data.shape
```

[2]:
```
(1897503, 14)
```

[3]:
```
data.info()
```

```
<class 'pandas.core.frame.DataFrame'>
RangeIndex: 1897503 entries, 0 to 1897502
```

```
Data columns (total 14 columns):
 #   Column        Dtype
---  ------        -----
 0   FL_DATE       object
 1   CARRIER_CODE  object
 2   TAIL_NUM      object
 3   FL_NUM        int64
 4   ORIGIN        object
 5   ORIGIN_ST     object
 6   DEST          object
 7   DEST_ST       object
 8   DEP_TIME      float64
 9   DEP_DELAY     float64
 10  ARR_TIME      float64
 11  ARR_DELAY     float64
 12  ELAPSED_TIME  float64
 13  DISTANCE      int64
dtypes: float64(5), int64(2), object(7)
memory usage: 202.7+ MB
```

[4]:
```
# Convert FL_DATE to datetime
data['FL_DATE'] = pd.to_datetime(data['FL_DATE'], format='%Y-%m-%d')

# Replace NaN values in DEP_TIME and ARR_TIME with '0000'
data['DEP_TIME'].fillna(0, inplace=True)
data['ARR_TIME'].fillna(0, inplace=True)

# Convert DEP_TIME and ARR_TIME to strings and pad with zeros
data['DEP_TIME'] = data['DEP_TIME'].apply(lambda x: '{:04d}'.format(int(x)))
data['ARR_TIME'] = data['ARR_TIME'].apply(lambda x: '{:04d}'.format(int(x)))

# Handle invalid time values: Replace '2400' with '0000' and increment the date
def fix_times(df, time_col, date_col):
    df[time_col] = df[time_col].apply(lambda x: '0000' if x == '2400' else x)
    # Adjust date if time was '2400'
    df.loc[df[time_col] == '0000', date_col] += pd.Timedelta(days=1)
    return df

data = fix_times(data, 'DEP_TIME', 'FL_DATE')
data = fix_times(data, 'ARR_TIME', 'FL_DATE')

# Combine FL_DATE with DEP_TIME and ARR_TIME to create datetime objects
data['DEP_DATETIME'] = pd.to_datetime(data['FL_DATE'].astype(str) + ' ' +
  ↪data['DEP_TIME'].str[:2] + ':' + data['DEP_TIME'].str[2:], format='%Y-%m-%d
  ↪%H:%M')
```

```
data['ARR_DATETIME'] = pd.to_datetime(data['FL_DATE'].astype(str) + ' ' +␣
↪data['ARR_TIME'].str[:2] + ':' + data['ARR_TIME'].str[2:], format='%Y-%m-%d␣
↪%H:%M')

# Convert relevant columns to string
for col in ['CARRIER_CODE', 'FL_NUM', 'TAIL_NUM','ORIGIN', 'ORIGIN_ST', 'DEST',␣
↪'DEST_ST']:
    data[col] = data[col].astype(str)

# Convert delay and distance columns to numeric
for col in ['DEP_DELAY', 'ARR_DELAY', 'ELAPSED_TIME', 'DISTANCE']:
    data[col] = pd.to_numeric(data[col], errors='coerce')

# Display data types after conversion
print(data.dtypes)
```

```
FL_DATE         datetime64[ns]
CARRIER_CODE            object
TAIL_NUM                object
FL_NUM                  object
ORIGIN                  object
ORIGIN_ST               object
DEST                    object
DEST_ST                 object
DEP_TIME                object
DEP_DELAY              float64
ARR_TIME                object
ARR_DELAY              float64
ELAPSED_TIME           float64
DISTANCE                 int64
DEP_DATETIME    datetime64[ns]
ARR_DATETIME    datetime64[ns]
dtype: object
```

Analysis and preprocessing

1. Checking for Missing values as invalid data was handled above only

[5]:
```
# Displaying the number of missing values
data.isnull().sum()
```

[5]:
```
FL_DATE             0
CARRIER_CODE        0
TAIL_NUM            0
FL_NUM              0
ORIGIN              0
ORIGIN_ST           0
DEST                0
```

```
DEST_ST             0
DEP_TIME            0
DEP_DELAY       26715
ARR_TIME            0
ARR_DELAY       31884
ELAPSED_TIME    31884
DISTANCE            0
DEP_DATETIME        0
ARR_DATETIME        0
dtype: int64
```

[6]:
```
# Displaying the number of missing values as percentages
missing_percentages = data.isnull().sum() * 100 / len(data)
print(missing_percentages.round(2))
```

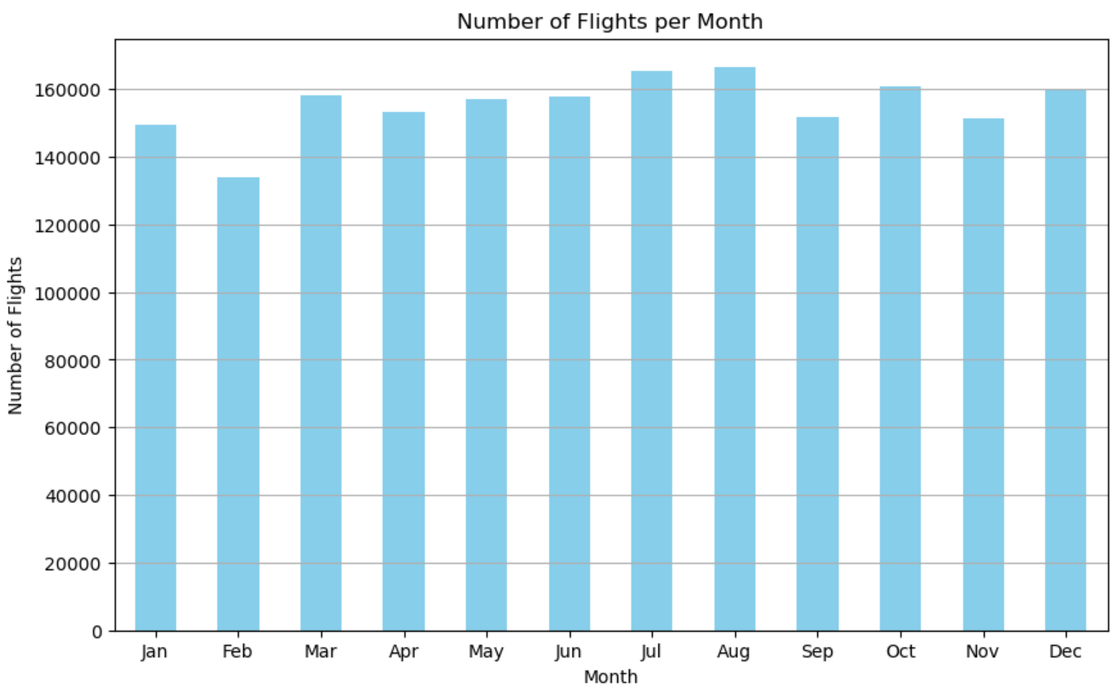
```
FL_DATE         0.00
CARRIER_CODE    0.00
TAIL_NUM        0.00
FL_NUM          0.00
ORIGIN          0.00
ORIGIN_ST       0.00
DEST            0.00
DEST_ST         0.00
DEP_TIME        0.00
DEP_DELAY       1.41
ARR_TIME        0.00
ARR_DELAY       1.68
ELAPSED_TIME    1.68
DISTANCE        0.00
DEP_DATETIME    0.00
ARR_DATETIME    0.00
dtype: float64
```

[7]:
```
# Handle missing values by deleting the values
data.dropna(subset=['DEP_DELAY', 'ARR_DELAY', 'ELAPSED_TIME'], inplace=True)
```

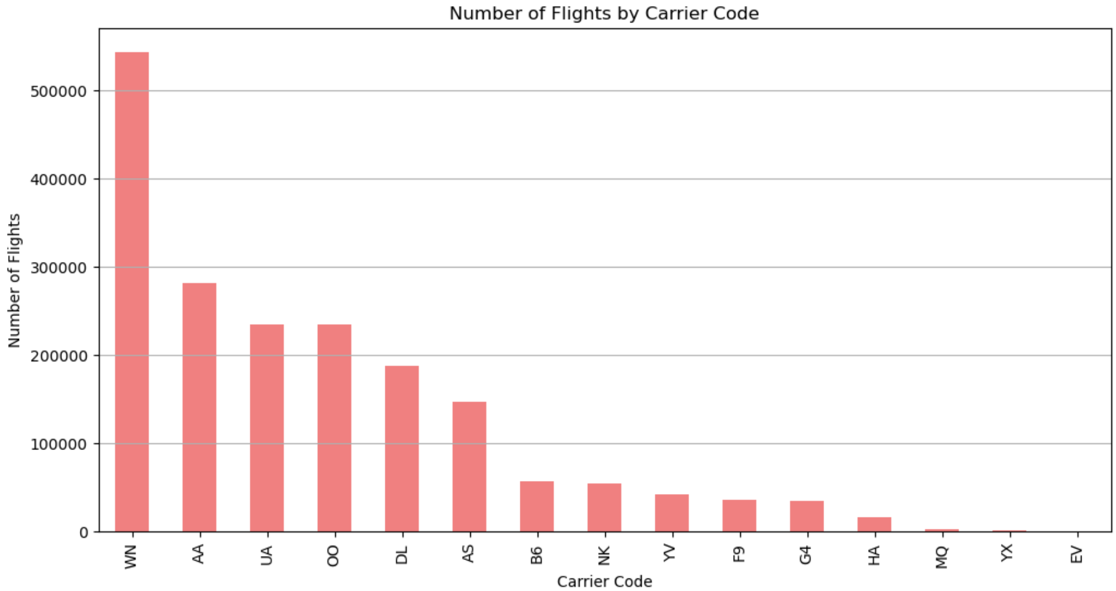
Dataset Overview and visualizations The dataset contains flight details for 2019, including dates, carrier codes, flight numbers, aircraft identifiers, origin and destination airports, departure and arrival times, delays, elapsed times, and distances for flights in Arizona, Nevada, and California.

[8]:
```
import matplotlib.pyplot as plt

# Number of Flights per Month
data['month'] = data['FL_DATE'].dt.month
monthly_flights = data['month'].value_counts().sort_index()

plt.figure(figsize=(10, 6))
```

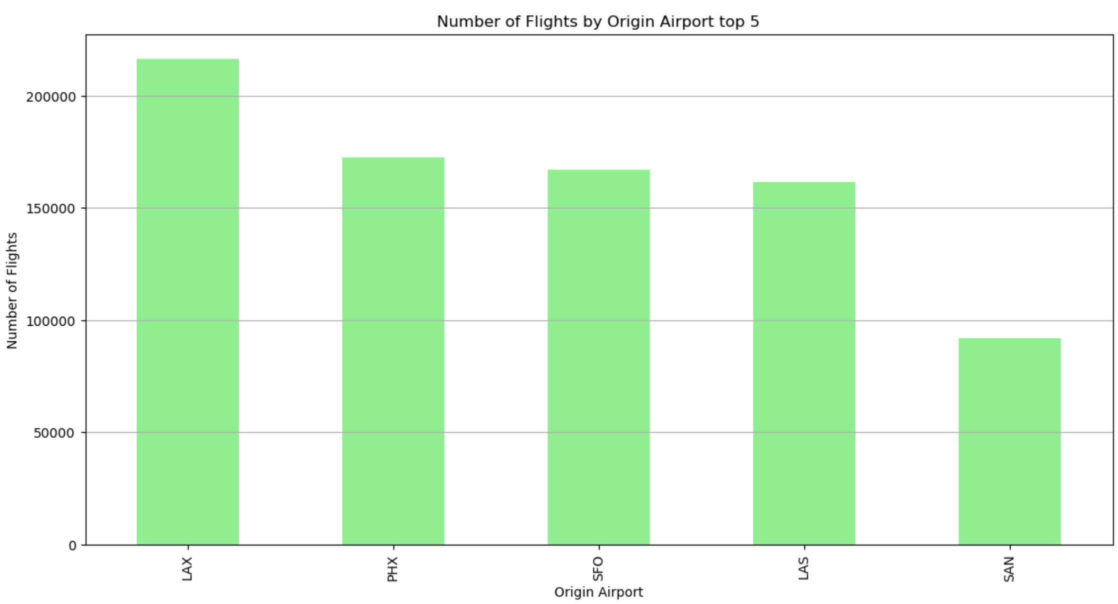
```
monthly_flights.plot(kind='bar', color='skyblue')
plt.title('Number of Flights per Month')
plt.xlabel('Month')
plt.ylabel('Number of Flights')
plt.xticks(range(12), ['Jan', 'Feb', 'Mar', 'Apr', 'May', 'Jun', 'Jul', 'Aug',␣
↪'Sep', 'Oct', 'Nov', 'Dec'], rotation=0)
plt.grid(axis='y')
plt.show()
```



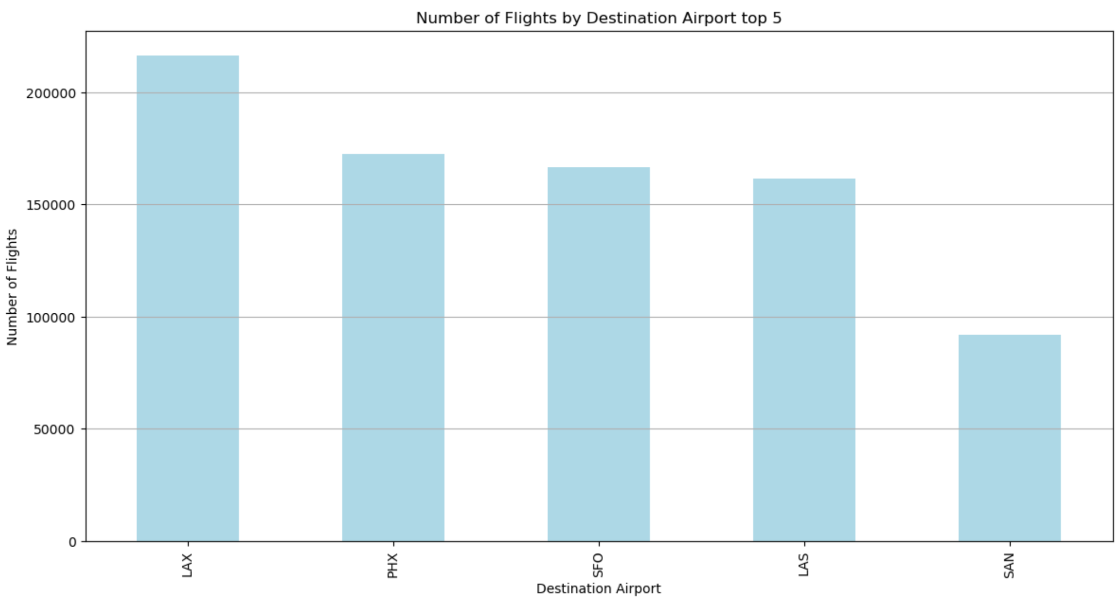
[9]:
```
# Flights by Carrier Code
carrier_flights = data['CARRIER_CODE'].value_counts()

plt.figure(figsize=(12, 6))
carrier_flights.plot(kind='bar', color='lightcoral')
plt.title('Number of Flights by Carrier Code')
plt.xlabel('Carrier Code')
plt.ylabel('Number of Flights')
plt.grid(axis='y')
plt.show()
```



[10]:
```python
# Flights by Origin Airport
origin_airports = data['ORIGIN'].value_counts().head(5)

plt.figure(figsize=(14, 7))
origin_airports.plot(kind='bar', color='lightgreen')
plt.title('Number of Flights by Origin Airport top 5')
plt.xlabel('Origin Airport')
plt.ylabel('Number of Flights')
plt.grid(axis='y')
plt.show()
```



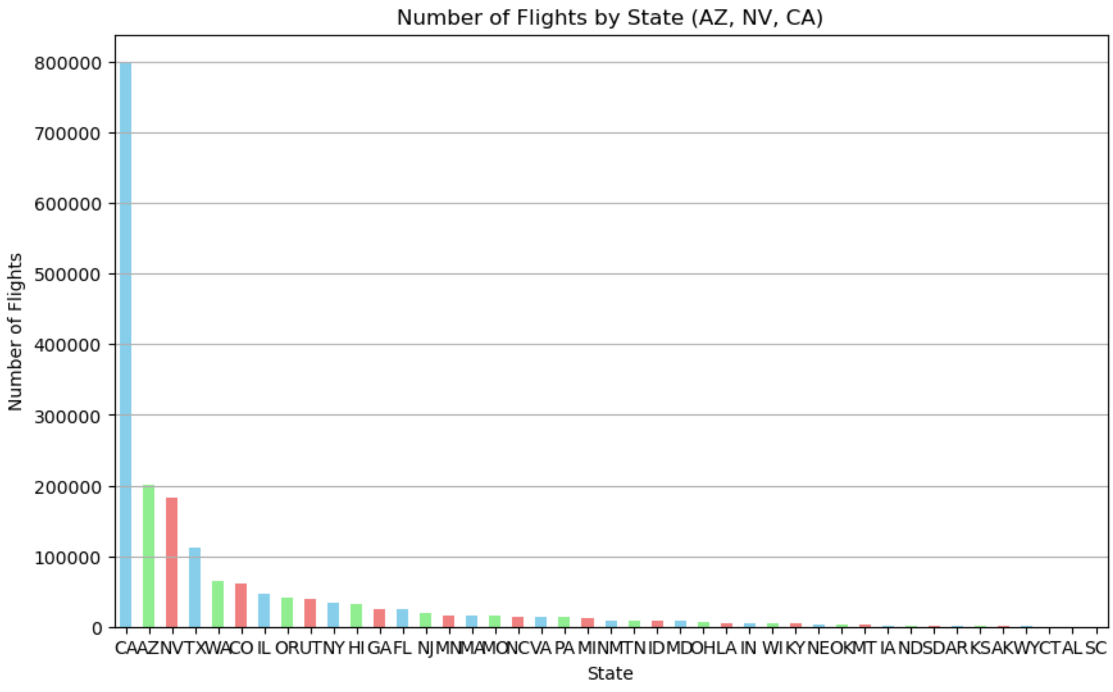
[11]:
```python
# Flights by Destination Airport
destination_airports = data['DEST'].value_counts().head(5)

plt.figure(figsize=(14, 7))
destination_airports.plot(kind='bar', color='lightblue')
plt.title('Number of Flights by Destination Airport top 5')
plt.xlabel('Destination Airport')
plt.ylabel('Number of Flights')
plt.grid(axis='y')
plt.show()
```



Air Traffic by Region (AZ, NV, CA) To determine which region has the most air traffic, we will calculate the number of flights originating from each state (AZ, NV, CA) and visualize the results.

[12]:
```python
import matplotlib.pyplot as plt

# Extract state information from the ORIGIN_ST column
state_traffic = data['ORIGIN_ST'].value_counts()

# Plot the number of flights for each state
plt.figure(figsize=(10, 6))
state_traffic.plot(kind='bar', color=['skyblue', 'lightgreen', 'lightcoral'])
plt.title('Number of Flights by State (AZ, NV, CA)')
plt.xlabel('State')
```

```
plt.ylabel('Number of Flights')
plt.xticks(rotation=0)
plt.grid(axis='y')
plt.show()

# Comment on findings
print(state_traffic)
```




```
ORIGIN_ST
CA    798690
AZ    201804
NV    181985
TX    112837
WA     65009
CO     62033
IL     47322
OR     41807
UT     40132
NY     33213
HI     31918
GA     25355
FL     24416
NJ     19097
MN     16510
MA     16153
```

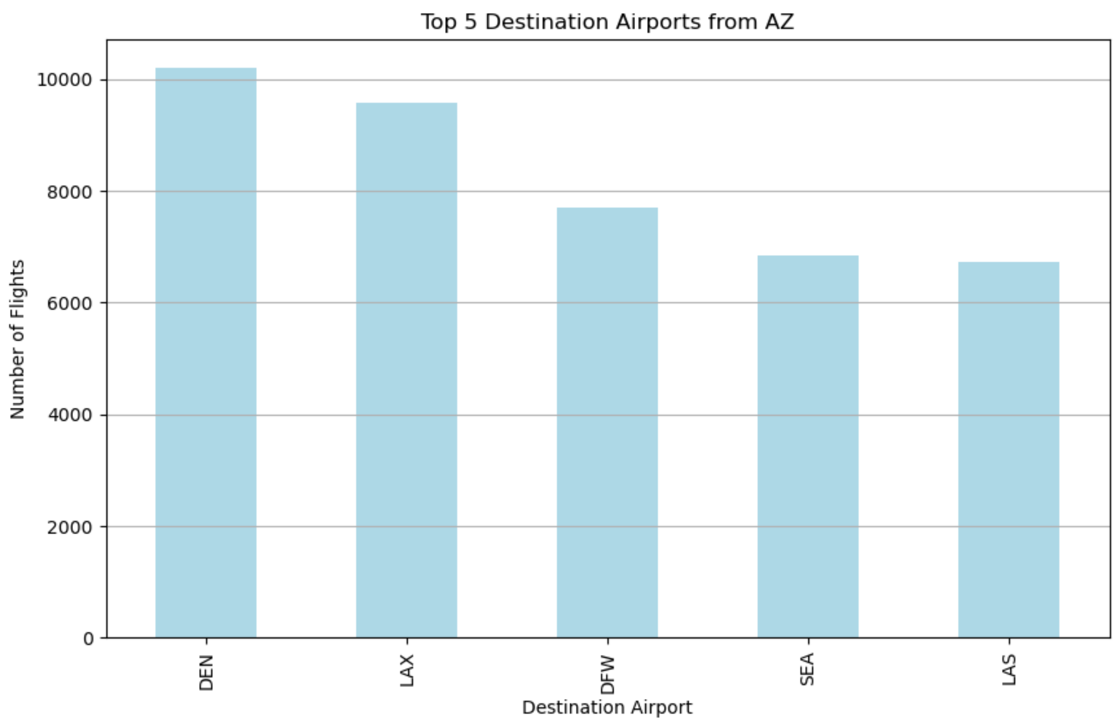
```
MO     15725
NC     13646
VA     13217
PA     13156
MI     12718
NM      9435
TN      8568
ID      8439
MD      8230
OH      6547
LA      5566
IN      4723
WI      4280
KY      4045
NE      4011
OK      3751
MT      2558
IA      2003
ND      1408
SD      1272
AR      1178
KS       958
AK       780
WY       644
CT       248
AL       173
SC        59
Name: count, dtype: int64
```

1. Findings we can visualize which state has the most air traffic based on the number of flights originating from airports in Arizona (AZ), Nevada (NV), and California (CA). From the data it seems that CA had the maximum number of flights.

Popular Outbound/Destination Airports for Each Region We will analyze the top 5 destination airports for flights originating from each state (AZ, NV, CA).

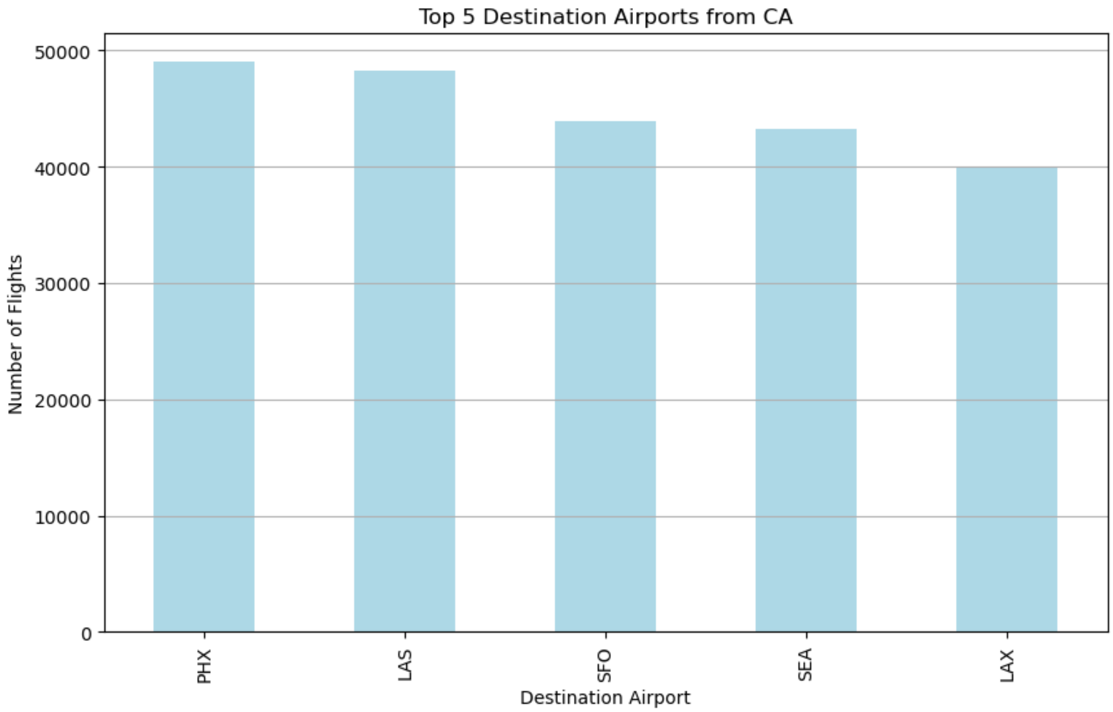
[13]:
```python
# Function to plot top 5 destination airports for a given state
def plot_top_destinations(state):
    state_data = data[data['ORIGIN_ST'] == state]
    top_destinations = state_data['DEST'].value_counts().head(5)

    plt.figure(figsize=(10, 6))
    top_destinations.plot(kind='bar', color='lightblue')
    plt.title(f'Top 5 Destination Airports from {state}')
    plt.xlabel('Destination Airport')
    plt.ylabel('Number of Flights')
    plt.grid(axis='y')
    plt.show()
```

```
    print(f"Top 5 destinations for {state}:\n{top_destinations}")

# Plot and comment on top destinations for AZ
plot_top_destinations('AZ')

# Plot and comment on top destinations for NV
plot_top_destinations('NV')

# Plot and comment on top destinations for CA
plot_top_destinations('CA')
```




```
Top 5 destinations for AZ:
DEST
DEN    10197
LAX     9585
DFW     7692
SEA     6836
LAS     6730
Name: count, dtype: int64
```


```
Top 5 destinations for NV:
DEST
LAX    13834
SFO     9233
DEN     8605
PHX     7755
SEA     7164
Name: count, dtype: int64
```




```
Top 5 destinations for CA:
DEST
PHX    48997
LAS    48239
SFO    43958
SEA    43233
LAX    40025
Name: count, dtype: int64
```

From the above visuals we can see that top destination airports for AZ, NV, CA are DEN, LAX, PHX respectivily

Explore the carriers. Calculate the proportion of flights for each airline/operator. Visualize the top 10 results. Explain the results.Analyze the flight delays for each Airline/Carrier and prepare summary statistics to explain the patterns in the delays. Visualize the results. Explain the patterns and demonstrate which carriers are more prone to flight delays. Note: you will need to analyze the airlines/carriers across multiple airports in order to conclude that they have a pattern of being late.

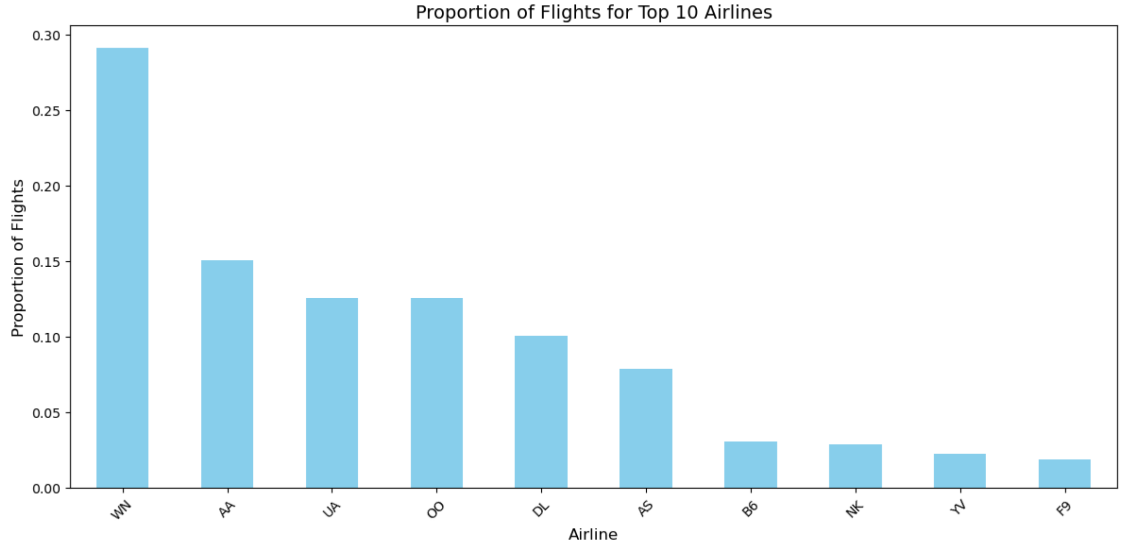
[14]:
```
# Calculate the number of flights for each airline
grouped = data.groupby(['CARRIER_CODE', 'ORIGIN'])

flight_counts = data['CARRIER_CODE'].value_counts()

# Calculate the total number of flights
```

```
total_flights = flight_counts.sum()

# Calculate the proportion of flights for each airline
flight_proportions = flight_counts / total_flights

# Get the top 10 airlines by number of flights
top_10_airlines = flight_proportions.head(50)
```

[15]:
```
flight_proportions.head(10)
```

[15]:
```
CARRIER_CODE
WN    0.291471
AA    0.150694
UA    0.125390
OO    0.125301
DL    0.100451
AS    0.078885
B6    0.030390
NK    0.028785
YV    0.022195
F9    0.018746
Name: count, dtype: float64
```

[16]:
```
# Visualization of the top 10 airlines by proportion of flights
plt.figure(figsize=(12, 6))
top_10_airlines.head(10).plot(kind='bar', color='skyblue')
plt.title('Proportion of Flights for Top 10 Airlines', fontsize=14)
plt.xlabel('Airline', fontsize=12)
plt.ylabel('Proportion of Flights', fontsize=12)
plt.xticks(rotation=45)
plt.tight_layout()
plt.show()
```



Here from the results we can see that airline WN has the highest proportion of flights whereas F9 has the least proportion of flights among Top 10 Airlines

[17]:
```
grouped = data.groupby(['CARRIER_CODE', 'ORIGIN'])

# Calculate summary statistics for delays
delay_stats = data.groupby(['CARRIER_CODE', 'ORIGIN']).agg({
    'DEP_DELAY': ['mean', 'median', 'std', 'min', 'max', 'count'],
    'ARR_DELAY': ['mean', 'median', 'std', 'min', 'max', 'count']
}).reset_index()

# Rename columns for clarity
delay_stats.columns = ['Airline', 'Origin_Airport', 'Mean_DEP_DELAY',␣
 ↪'Median_DEP_DELAY', 'STD_DEP_DELAY', 'Min_DEP_DELAY', 'Max_DEP_DELAY',␣
 ↪'Count_DEP',
                       'Mean_ARR_DELAY', 'Median_ARR_DELAY', 'STD_ARR_DELAY',␣
 ↪'Min_ARR_DELAY', 'Max_ARR_DELAY', 'Count_ARR']

# Filter for top 10 airlines by number of flights
top_10_delay_stats = delay_stats[delay_stats['Airline'].isin(top_10_airlines.
 ↪index)]
```

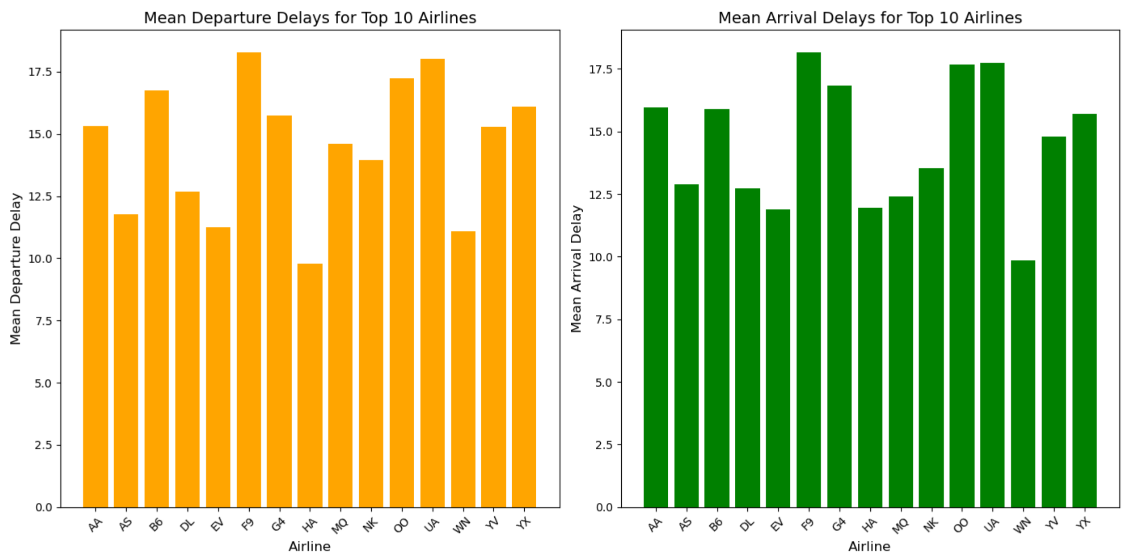
[18]:
```
top_10_delay_stats
```

[18]:
```
    Airline Origin_Airport  Mean_DEP_DELAY  Median_DEP_DELAY  STD_DEP_DELAY  \
0        AA            ABQ        5.771812               0.0      19.939815
1        AA            ANC       28.620155               5.0     102.131517
2        AA            ATL       10.349646               0.0      43.421143
3        AA            AUS       13.536156               0.0      55.497511
```

```
4        AA           BDL      10.649194             0.0      43.143508
..       …            …          …          …         …
588      YV           YUM      14.679887             0.0      52.029407
589      YX           DEN      23.000000             0.0      71.277021
590      YX           ORD       5.127660             0.0      27.032683
591      YX           SMF       8.888889             0.0      21.103581
592      YX           TUS      27.323529             0.0      80.378478

     Min_DEP_DELAY  Max_DEP_DELAY  Count_DEP  Mean_ARR_DELAY  \
0              0.0          152.0        149        7.281879
1              0.0          924.0        129       23.651163
2              0.0         1102.0       1696       10.553656
3              0.0         1183.0       2669       13.802922
4              0.0          430.0        248       11.407258
..             …              …      …             …
588            0.0          526.0        353       14.889518
589            0.0          383.0         30       21.433333
590            0.0          183.0         47        7.787234
591            0.0           64.0          9        7.888889
592            0.0          500.0         68       25.705882

     Median_ARR_DELAY  STD_ARR_DELAY  Min_ARR_DELAY  Max_ARR_DELAY  Count_ARR
0                 0.0      24.668637            0.0          196.0        149
1                 0.0     100.847166            0.0          919.0        129
2                 0.0      42.988941            0.0         1110.0       1696
3                 0.0      55.395012            0.0         1227.0       2669
4                 0.0      40.524490            0.0          417.0        248
..                …             …            …             …         …
588               0.0      51.831641            0.0          529.0        353
589               0.0      69.030070            0.0          370.0         30
590               0.0      27.536487            0.0          170.0         47
591               0.0      23.666667            0.0           71.0          9
592               0.0      77.687291            0.0          481.0         68

[593 rows x 14 columns]
```

[19]:
```python
# Visualization of mean departure and arrival delays for top 10 airlines
plt.figure(figsize=(14, 7))
grouped_df = top_10_delay_stats.groupby('Airline').
 ↪agg(Overall_Mean_Dep=('Mean_DEP_DELAY', 'mean'),
                                        Overall_Mean_Arr=('Mean_ARR_DELAY',␣
 ↪'mean')).reset_index()

# Departure Delays
plt.subplot(1, 2, 1)
plt.bar(grouped_df['Airline'], grouped_df['Overall_Mean_Dep'], color='orange')
plt.title('Mean Departure Delays for Top 10 Airlines', fontsize=14)
```

```
plt.xlabel('Airline', fontsize=12)
plt.ylabel('Mean Departure Delay', fontsize=12)
plt.xticks(rotation=45)
# Arrival Delays
plt.subplot(1, 2, 2)
plt.bar(grouped_df['Airline'], grouped_df['Overall_Mean_Arr'], color='green')
plt.title('Mean Arrival Delays for Top 10 Airlines', fontsize=14)
plt.xlabel('Airline', fontsize=12)
plt.ylabel('Mean Arrival Delay', fontsize=12)
plt.xticks(rotation=45)
plt.tight_layout()
plt.show()
```



[20]:
```
# Analysis of patterns in delays
print("Summary Statistics ")
print(top_10_delay_stats)

# Determine airlines more prone to delays
prone_to_delays = top_10_delay_stats[['Airline', 'Mean_DEP_DELAY',␣
↪'Mean_ARR_DELAY']].sort_values(by=['Mean_ARR_DELAY', 'Mean_DEP_DELAY'],␣
↪ascending=False)
print("\nAirlines more prone to delays (sorted by arrival delays):")
print(prone_to_delays)
```

```
Summary Statistics
    Airline Origin_Airport  Mean_DEP_DELAY  Median_DEP_DELAY  STD_DEP_DELAY  \
0        AA            ABQ        5.771812               0.0      19.939815
1        AA            ANC       28.620155               5.0     102.131517
2        AA            ATL       10.349646               0.0      43.421143
```

```
3        AA          AUS      13.536156           0.0      55.497511
4        AA          BDL      10.649194           0.0      43.143508
..       …           …        …                   …        …
588      YV          YUM      14.679887           0.0      52.029407
589      YX          DEN      23.000000           0.0      71.277021
590      YX          ORD       5.127660           0.0      27.032683
591      YX          SMF       8.888889           0.0      21.103581
592      YX          TUS      27.323529           0.0      80.378478

     Min_DEP_DELAY  Max_DEP_DELAY  Count_DEP  Mean_ARR_DELAY  \
0              0.0          152.0        149        7.281879
1              0.0          924.0        129       23.651163
2              0.0         1102.0       1696       10.553656
3              0.0         1183.0       2669       13.802922
4              0.0          430.0        248       11.407258
..             …            …          …          …
588            0.0          526.0        353       14.889518
589            0.0          383.0         30       21.433333
590            0.0          183.0         47        7.787234
591            0.0           64.0          9        7.888889
592            0.0          500.0         68       25.705882

     Median_ARR_DELAY  STD_ARR_DELAY  Min_ARR_DELAY  Max_ARR_DELAY  Count_ARR
0                 0.0      24.668637            0.0          196.0        149
1                 0.0     100.847166            0.0          919.0        129
2                 0.0      42.988941            0.0         1110.0       1696
3                 0.0      55.395012            0.0         1227.0       2669
4                 0.0      40.524490            0.0          417.0        248
..                …            …              …              …             …
588               0.0      51.831641            0.0          529.0        353
589               0.0      69.030070            0.0          370.0         30
590               0.0      27.536487            0.0          170.0         47
591               0.0      23.666667            0.0           71.0          9
592               0.0      77.687291            0.0          481.0         68

[593 rows x 14 columns]

Airlines more prone to delays (sorted by arrival delays):
    Airline  Mean_DEP_DELAY  Mean_ARR_DELAY
429      UA      140.500000      148.000000
145      DL      141.500000      129.500000
22       AA       89.384615       83.307692
20       AA       67.948454       68.680412
422      OO       64.377049       58.573770
..       …            …               …
425      UA        1.571429        0.000000
42       AA        0.000000        0.000000
157      DL        0.000000        0.000000
```

```
193       F9         0.000000          0.000000
280       G4         0.000000          0.000000

[593 rows x 3 columns]
```

Here form the visuals we can see that airlines F9, UA, B6 and OO are more prone to delays.

Patterns in Flight Delays:

Airlines like F9, OO, UA and B6 exhibit higher average delays, indicating a potential pattern of being late.

Carriers such as WN and EV show lower average delays, suggesting better on-time performance.

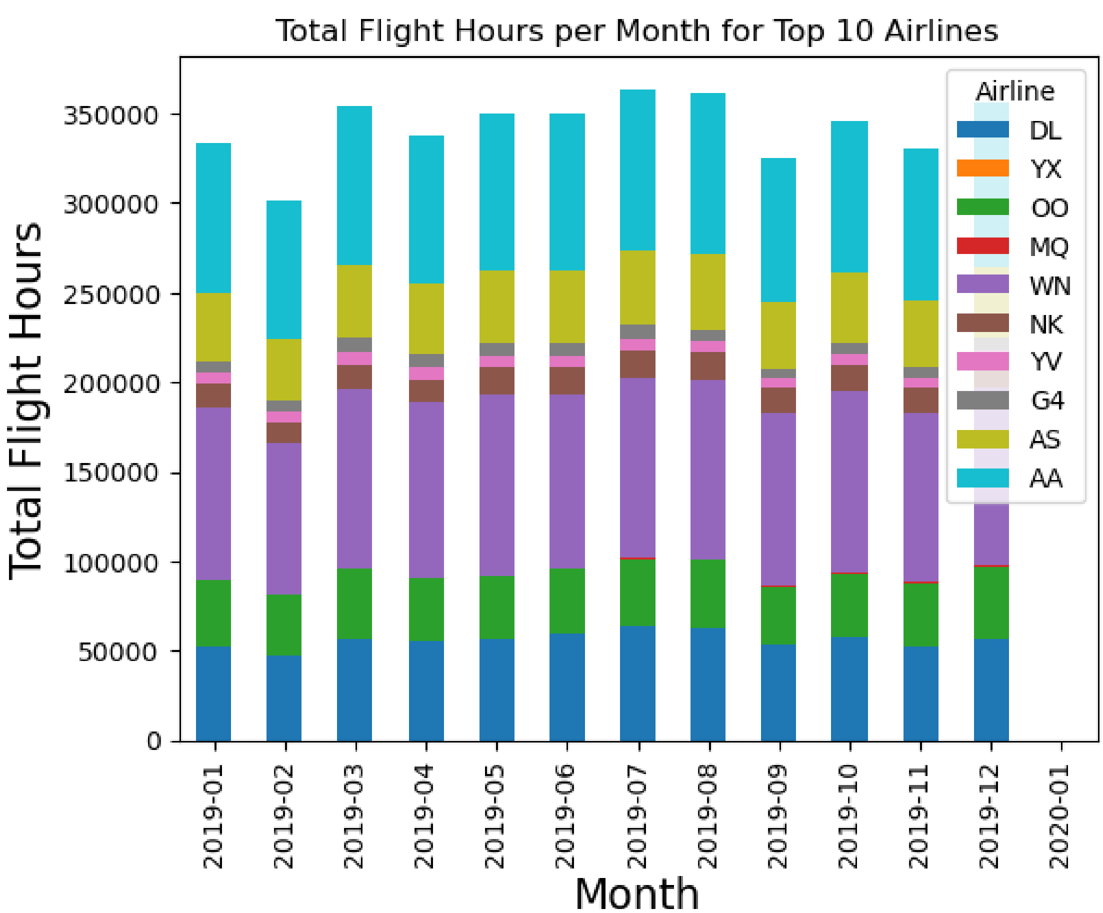
Evaluate which airlines have the best record.Display the top 10. #### calculate their total flight hours for each month.

[21]:
```python
import matplotlib.pyplot as plt

# Define on-time flights (arrival delay <= 15 minutes)
data['ON_TIME'] = data['ARR_DELAY'] <= 15

# Calculate proportion of on-time flights for each airline
on_time_proportion = data.groupby('CARRIER_CODE')['ON_TIME'].mean().
 ↪sort_values(ascending=False)

# Display the top 10 airlines with the best on-time records
top_10_airlines = on_time_proportion.head(10)
print(top_10_airlines)

# Convert elapsed time from minutes to hours
data['ELAPSED_HOURS'] = data['ELAPSED_TIME'] / 60

# Extract month and year from FL_DATE
data['YEAR_MONTH'] = data['FL_DATE'].dt.to_period('M')

# Calculate total flight hours for each airline per month
flight_hours_per_month = data.groupby(['CARRIER_CODE',␣
 ↪'YEAR_MONTH'])['ELAPSED_HOURS'].sum().unstack(fill_value=0)

# Filter to only include the top 10 airlines
top_10_flight_hours = flight_hours_per_month.loc[top_10_airlines.index]
plt.figure(figsize=(20, 12))

# Plot the total flight hours for the top 10 airlines
top_10_flight_hours.T.plot(kind='bar', stacked = 'True')
plt.title('Total Flight Hours per Month for Top 10 Airlines')
plt.xlabel('Month', fontsize=16)
plt.ylabel('Total Flight Hours', fontsize=16)
```

```
plt.legend(title='Airline')
plt.show()

plt.figure(figsize=(20, 12))
```

```
CARRIER_CODE
DL     0.852522
YX     0.837662
OO     0.837190
MQ     0.836723
WN     0.834652
NK     0.832964
YV     0.814987
G4     0.811495
AS     0.809241
AA     0.803719
Name: ON_TIME, dtype: float64
```

```
<Figure size 2000x1200 with 0 Axes>
```



[21]: <Figure size 2000x1200 with 0 Axes>

<Figure size 2000x1200 with 0 Axes>

From the above visuals we can see the total flight hours appear to be relatively stable month-to-month with some fluctuations. There are slight peaks observed in March, May,June, July , Aug and October 2019, indicating higher flight activity during these months. There might be seasonal patterns, such as higher flight hours during the summer months May, June, July and lower flight hours in January and February. Airlines like AA, DL and WN seem to have larger segments, indicating they have more flight hours compared to others like G4, NK, OO and YV.

Q5. Select any (3) aircraft, and explore the data to determine where it often travels. Calculate its average arrival and departure delays at the airports. After which analyze all the results to identify any patterns that are evident and also indicate which airline operates that aircraft. Explain your findings and visualize the results. Note: the TAIL_NUM can help you to identify each unique aircraft.

Selecting any (3) aircraft, and explore the data to determine where it often travels. Calculate its average arrival and departure delays at the airports. After which analyze all the results to identify any patterns that are evident and also indicate which airline operates that aircraft.

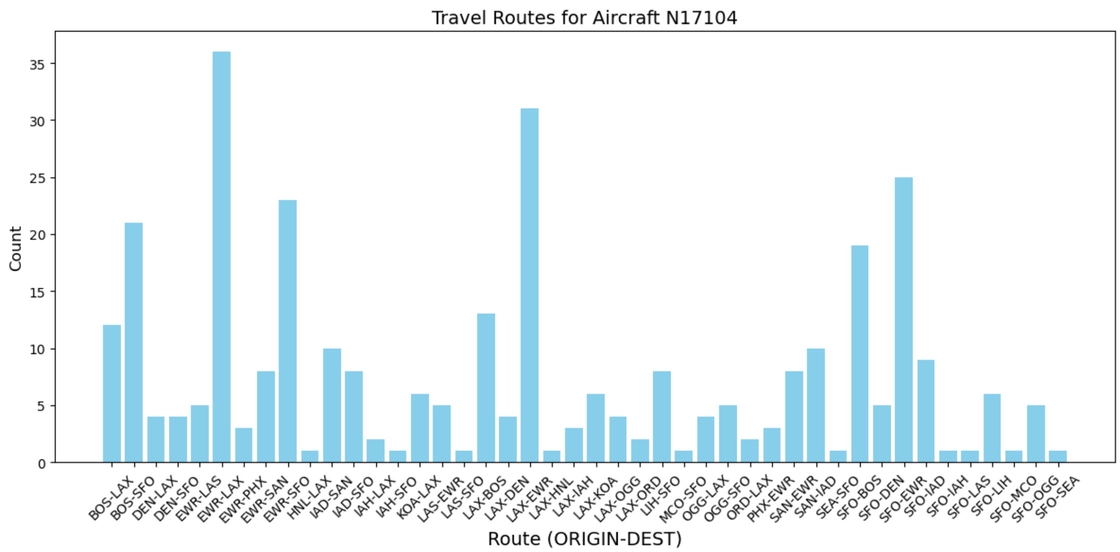
[22]:
```python
selected_tail_nums = ['N17104', 'N38403', 'N37508']
df_selected = data[data['TAIL_NUM'].isin(selected_tail_nums)]

# Calculate most frequent routes for each aircraft
frequent_routes = df_selected.groupby(['TAIL_NUM', 'ORIGIN', 'DEST']).size().
↪reset_index(name='count')
print(frequent_routes)

# Calculate average arrival and departure delays for each aircraft at various␣
↪airports
average_delays = df_selected.groupby(['TAIL_NUM', 'ORIGIN', 'DEST',␣
↪'CARRIER_CODE']).agg({
    'DEP_DELAY': 'mean',
    'ARR_DELAY': 'mean'
}).reset_index()
print(average_delays)
```

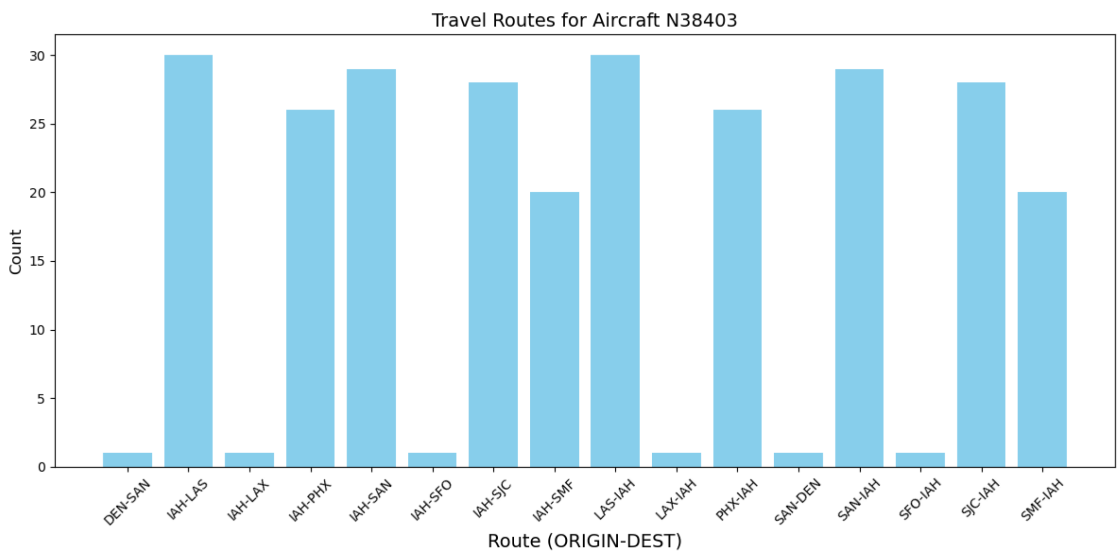
```
   TAIL_NUM ORIGIN DEST  count
0    N17104    BOS  LAX     12
1    N17104    BOS  SFO     21
2    N17104    DEN  LAX      4
3    N17104    DEN  SFO      4
4    N17104    EWR  LAS      5
..      …     …    …       …
71   N38403    SAN  DEN      1
72   N38403    SAN  IAH     29
73   N38403    SFO  IAH      1
```

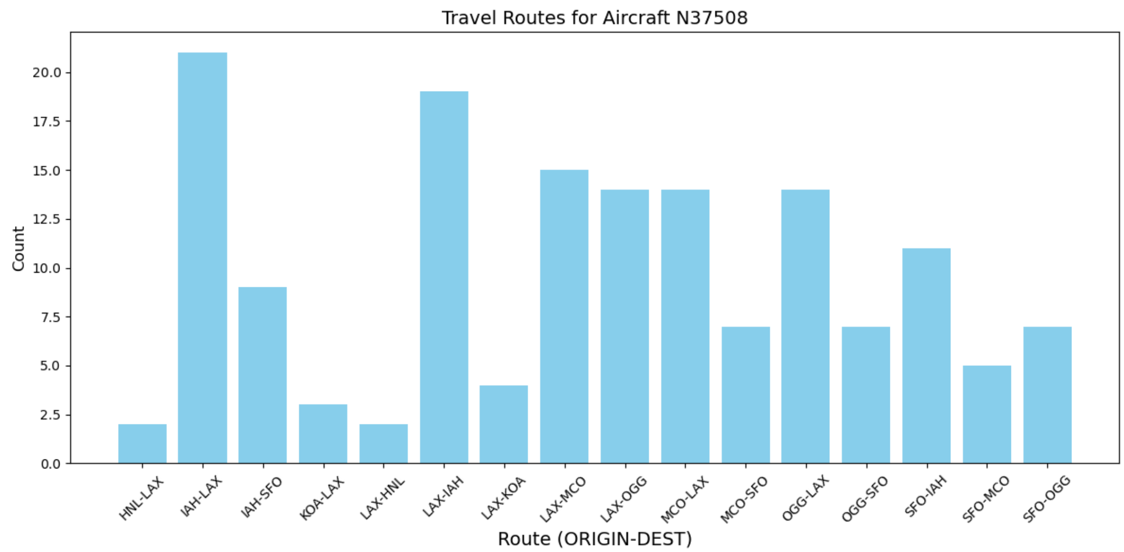
```
74   N38403    SJC  IAH     28
75   N38403    SMF  IAH     20

[76 rows x 4 columns]
   TAIL_NUM ORIGIN DEST CARRIER_CODE  DEP_DELAY  ARR_DELAY
0    N17104    BOS  LAX           UA   9.583333  11.916667
1    N17104    BOS  SFO           UA  44.904762  39.190476
2    N17104    DEN  LAX           UA  14.250000  12.750000
3    N17104    DEN  SFO           UA  45.000000  45.250000
4    N17104    EWR  LAS           UA  38.000000  27.600000
..      …    …   …          …        …         …
71   N38403    SAN  DEN           UA   0.000000   0.000000
72   N38403    SAN  IAH           UA   2.344828   1.965517
73   N38403    SFO  IAH           UA  64.000000  48.000000
74   N38403    SJC  IAH           UA  20.178571  19.678571
75   N38403    SMF  IAH           UA   6.200000   6.200000

[76 rows x 6 columns]
```

[23]:

```
# Identify patterns and visualize the results

# Visualization of travel patterns
for tail_num in selected_tail_nums:
    routes = df_selected[df_selected['TAIL_NUM'] == tail_num].
 ↪groupby(['ORIGIN', 'DEST', 'CARRIER_CODE']).size().reset_index(name='count')
    plt.figure(figsize=(12, 6))
    plt.bar(routes['ORIGIN'] + '-' + routes['DEST'], routes['count'],␣
 ↪color='skyblue')
    plt.title(f'Travel Routes for Aircraft {tail_num}', fontsize=14)
    plt.xlabel('Route (ORIGIN-DEST)', fontsize=14)
    plt.ylabel('Count', fontsize=12)
    plt.xticks(rotation=45)
    plt.tight_layout()
    plt.show()
```

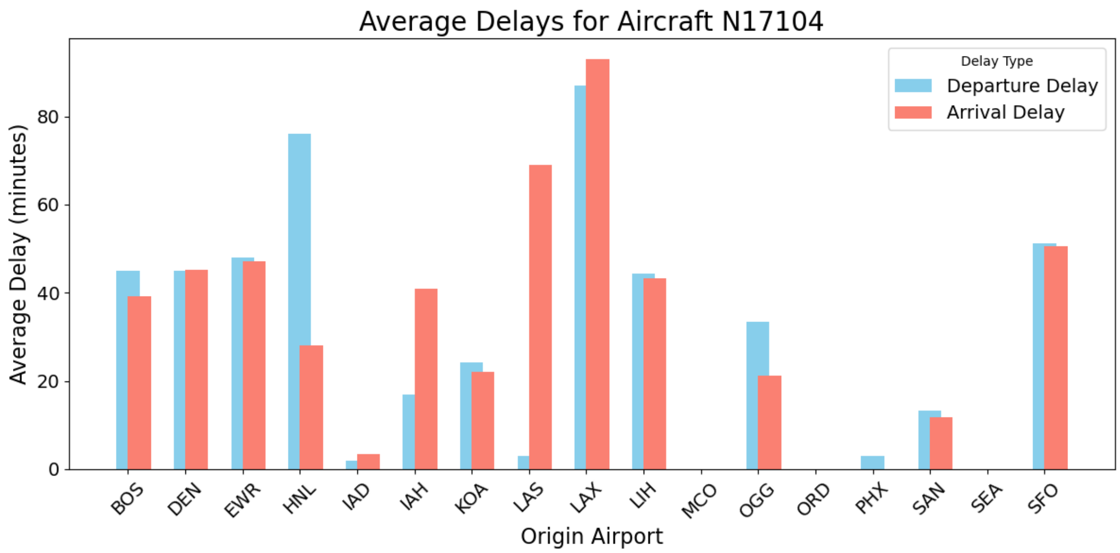

Travel Routes for Aircraft N17104
Count
Route (ORIGIN-DEST)
BOS-LAX
BOS-SFO
DEN-LAX
DEN-SFO
EWR-LAS
EWR-LAX
EWR-PHX
EWR-SAN
EWR-SFO
HNL-LAX
IAD-SAN
IAD-SFO
IAH-LAX
IAH-SFO
KOA-LAX
LAS-EWR
LAS-SFO
LAX-BOS
LAX-DEN
LAX-EWR
LAX-HNL
LAX-IAH
LAX-KOA
LAX-OGG
LAX-ORD
LIH-SFO
MCO-SFO
OGG-LAX
OGG-SFO
ORD-LAX
PHX-EWR
SAN-EWR
SAN-IAD
SEA-SFO
SFO-BOS
SFO-DEN
SFO-EWR
SFO-IAD
SFO-IAH
SFO-LAS
SFO-LIH
SFO-MCO
SFO-OGG
SFO-SEA

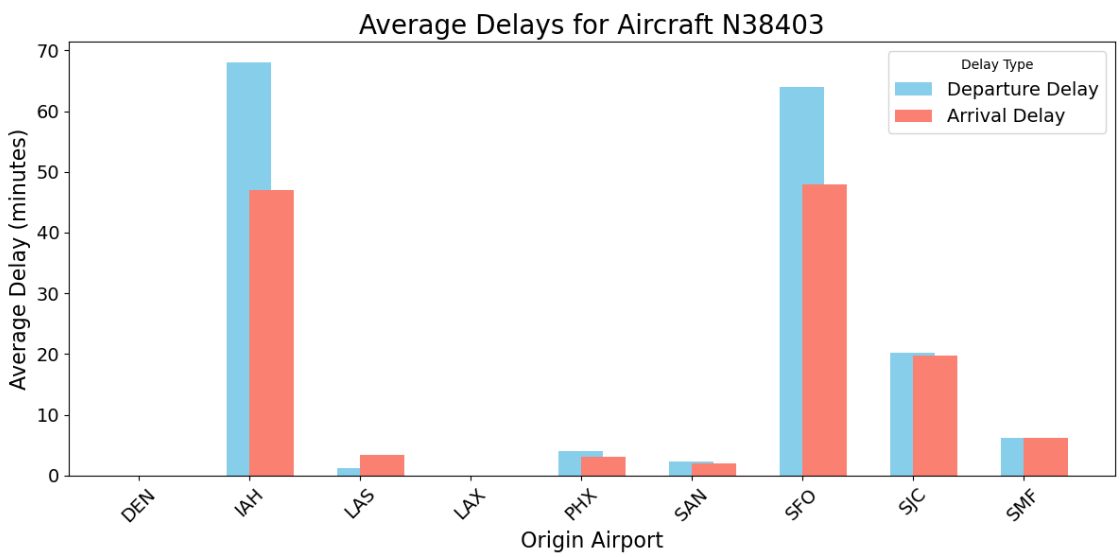

Travel Routes for Aircraft N38403
Count
Route (ORIGIN-DEST)
DEN-SAN
IAH-LAS
IAH-LAX
IAH-PHX
IAH-SAN
IAH-SFO
IAH-SJC
IAH-SMF
LAS-IAH
LAX-IAH
PHX-IAH
SAN-DEN
SAN-IAH
SFO-IAH
SJC-IAH
SMF-IAH

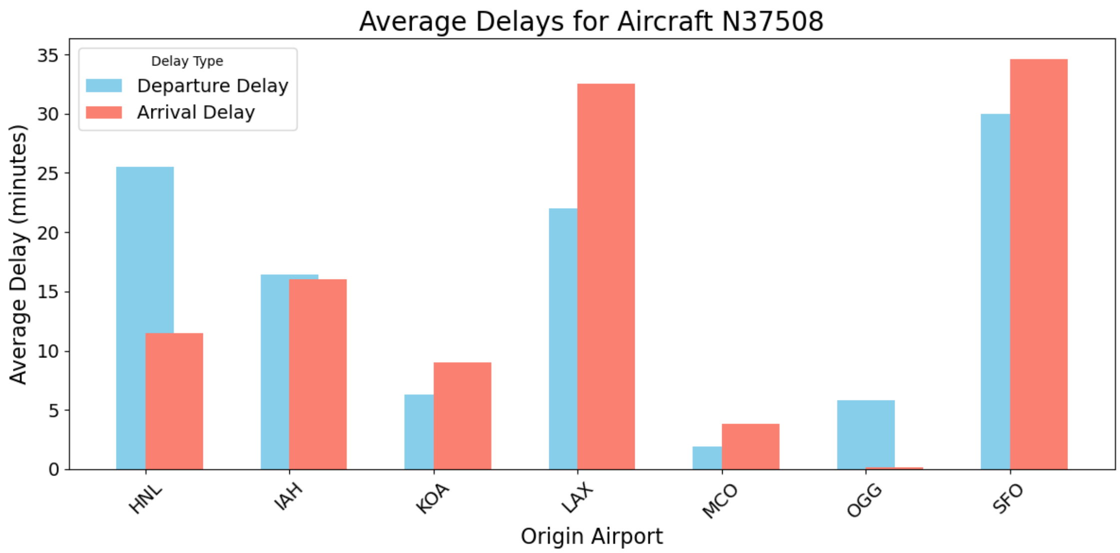


Here from the visuals we can see that the most travelled route is EWR-IAS for the first aircraft, IAH-LAS for the second aircraft and IAH-LAX for the third aircraft.

[24]:
```
for tail_num in selected_tail_nums:
    delays = average_delays[average_delays['TAIL_NUM'] == tail_num]
    plt.figure(figsize=(12, 6))

    # Plotting the bar chart for average delays
    plt.bar(delays['ORIGIN'], delays['DEP_DELAY'], width=0.4, align='center',␣
 ↪label='Departure Delay', color='skyblue')
    plt.bar(delays['ORIGIN'], delays['ARR_DELAY'], width=0.4, align='edge',␣
 ↪label='Arrival Delay', color='salmon')

    # Customizing the plot
    plt.title(f'Average Delays for Aircraft {tail_num}', fontsize=20)
    plt.xlabel('Origin Airport', fontsize=16)
    plt.ylabel('Average Delay (minutes)', fontsize=16)
    plt.xticks(rotation=45, fontsize=14)
    plt.yticks(fontsize=14)
    plt.legend(title='Delay Type', fontsize=14)

    # Adjusting the layout
    plt.tight_layout()

    # Showing the plot
    plt.show()
```


Average Delays for Aircraft N17104
Delay Type
Departure Delay
Arrival Delay
Average Delay (minutes)
0
20
40
60
80
BOS
DEN
EWR
HNL
IAD
IAH
KOA
LAS
LAX
LIH
MCO
OGG
ORD
PHX
SAN
SEA
SFO
Origin Airport


Average Delays for Aircraft N38403
Delay Type
Departure Delay
Arrival Delay
Average Delay (minutes)
0
10
20
30
40
50
60
70
DEN
IAH
LAS
LAX
PHX
SAN
SFO
SJC
SMF
Origin Airport



Here from the visuals we can see that the origin airport with the highest average departure and arrival delays for ths aircraft N17104 appears to be LAX whereas for aircraft N38403 it appears to be IAH and for aircraft N37508 it appears to be SFO. We can aslo see that airports like IAH and SFO seem to have relatively higher average delays, both for departures and arrivals acroos all three aircrafts. Airports like KOA and PHX generally had lower average delays for both departures and arrivals across all three aircrafts.