# Term Project Milestone 1 - Data selection and EDA

Airlines On-Time Performance, Delays, Cancellations and Diversions

Introduction: Airline cancellations or delays are one of the major causes of passenger inconvenience. With publicly available dataset, using data science, I am hoping to gain meaningful insights into the best-performing airlines and understand the causes of delays, diversions and cancellations across different airline carriers.

For the final project, I would like to analyze airline data to identify different factors and their effects on a carrier's performance. Using the available performance measures I would like to be able to predict the chances of a flight being on-time/delayed/cancelled.

Data Source: Excel files from BTS. The Excel data has airline performance factors such as cancelled, diverted, delayed and on-time data. The downloaded raw data has up to 34 columns. https://www.transtats.bts.gov/OT_Delay/OT_DelayCause1.asp?20=E (Download Raw Data link for data).

Problem statement addressed:

This study will benefit Customers as it will help predict a flights performance. Customers can lookup the chances of their flight reaching on-time during their booking or even before heading to the airport. Airlines can also benefit by comparing airline performances and predicting possibilities of delay based on aircraft/origin/destination and apply corrective measures to reduce cancellations and delays and improve on-time performance.

## Data Transformation

In the data transformation step, I will be modifying the following:

1. Cancellation reason in the flight dataset is represented as A, B, C and D. I will be updating the cancellation code as follows:

   A Carrier

   B Weather

   C National Air System

   D Security

1. I will be adding a new column 'Status' with the status of a flight such as, On-Time, Delayed, Cancelled, Diverted.

2. Diverted column is of binary value which can be modified to a Yes/No

Data Visualization:

In [1]:
```
#Load necessary libraries
import pandas as pd
import numpy as np
```

```
import plotly.express as px
import seaborn as sns
import matplotlib.pyplot as plt
```

In [2]:
```
#Read flight data from "https://www.transtats.bts.gov/OT_Delay/OT_DelayCause1.asp?20=E"

flight_data_df = pd.read_csv('T_ONTIME_MARKETING_May.csv')
flight_data_df.head()
```

Out[2]:

| | YEAR | QUARTER | MONTH | DAY_OF_MONTH | DAY_OF_WEEK | FL_DATE | MKT_UNIQUE_CARRIER | OP_UNIQUE_CAI |
|---|---|---|---|---|---|---|---|---|
| **0** | 2022 | 2 | 5 | 1 | 7 | 5/1/2022 12:00:00 AM | AA | |
| **1** | 2022 | 2 | 5 | 1 | 7 | 5/1/2022 12:00:00 AM | AA | |
| **2** | 2022 | 2 | 5 | 1 | 7 | 5/1/2022 12:00:00 AM | AA | |
| **3** | 2022 | 2 | 5 | 1 | 7 | 5/1/2022 12:00:00 AM | AA | |
| **4** | 2022 | 2 | 5 | 1 | 7 | 5/1/2022 12:00:00 AM | AA | |

5 rows × 39 columns

Duplicates cause inconsistent results when dealing with statistics. Hence dropping duplicate rows.

In [3]:
```
print('Dataframe before dropping duplicates :', flight_data_df.shape)
flight_data_df = flight_data_df.drop_duplicates() # 1,389 rows dropped
print('Dataframe after dropping duplicates :',flight_data_df.shape)
```

```
Dataframe before dropping duplicates : (602950, 39)
Dataframe after dropping duplicates : (601561, 39)
```

Drop null rows, if any and update null values to 0 for delays

In [4]:
```
#Drop null values
flight_data_df.dropna()
#Update null values to 0
flight_data_df.DISTANCE = flight_data_df.DISTANCE.fillna(0)
flight_data_df.DEP_DELAY = flight_data_df.DEP_DELAY.fillna(0)
flight_data_df.ARR_DELAY = flight_data_df.ARR_DELAY.fillna(0)
flight_data_df.CARRIER_DELAY = flight_data_df.CARRIER_DELAY.fillna(0)
flight_data_df.WEATHER_DELAY = flight_data_df.WEATHER_DELAY.fillna(0)
flight_data_df.NAS_DELAY = flight_data_df.NAS_DELAY.fillna(0)
flight_data_df.SECURITY_DELAY = flight_data_df.SECURITY_DELAY.fillna(0)
flight_data_df.LATE_AIRCRAFT_DELAY = flight_data_df.LATE_AIRCRAFT_DELAY.fillna(0)
```

Cancellation code is represented as A, B, C and D, which is not very informative. The BTS website provided details on this code as follows:

In [5]:
```
flight_data_df['CANCELLATION_REASON'] = ''
flight_data_df.CANCELLATION_REASON = np.where(flight_data_df.CANCELLATION_CODE=='A', 'Ca
                                     np.where(flight_data_df.CANCELLATION_CODE=='B', 'Weathe
                                              np.where(flight_data_df.CANCELLATION_CODE=='C'
```

```
                                          np.where(flight_data_df.CANCELLATION_

flight_data_df.groupby(['CANCELLATION_REASON'])['CANCELLATION_REASON'].count().sort_inde
```

Out[5]:

```
CANCELLATION_REASON
                           590957
Carrier                      4902
National Air System          1394
Security                        1
Weather                      4307
Name: CANCELLATION_REASON, dtype: int64
```

Adding a new column 'STATUS' that tells the status of a flight

In [6]:

```
flight_data_df['STATUS'] = ''

flight_data_df.STATUS = np.where(flight_data_df.CANCELLED==1, 'Cancelled',
                                 np.where(flight_data_df.DIVERTED==1, 'Diverted',
                                          np.where(flight_data_df.ARR_DELAY<=15, 'On-Tim
                                                   np.where(flight_data_df.ARR_DELAY>15,
flight_data_df.groupby(['STATUS'])['STATUS'].count().sort_index()
```

Out[6]:

```
STATUS
Cancelled     10604
Delayed      119624
Diverted       1581
On-Time      469752
Name: STATUS, dtype: int64
```

Creating a new column 'ARR_DELAYED'. A flag that represents if a flight was delayed. Similar to CANCELLED and DIVERTED As a step to data reduction, I will be considering flights departing or arriving 15 minutes or later as delayed

In [7]:

```
flight_data_df.loc[(flight_data_df['ARR_DELAY']>15), 'ARR_DELAYED'] = True
flight_data_df.loc[(flight_data_df['ARR_DELAY']<=15), 'ARR_DELAYED'] = False

flight_data_df.groupby(['ARR_DELAYED'])['ARR_DELAYED'].count().sort_index()
```

Out[7]:

```
ARR_DELAYED
False    481937
True     119624
Name: ARR_DELAYED, dtype: int64
```

Add a new column for DEP_DELAYED

In [8]:

```
flight_data_df.loc[(flight_data_df['DEP_DELAY']>15), 'DEP_DELAYED'] = True
flight_data_df.loc[(flight_data_df['DEP_DELAY']<=15), 'DEP_DELAYED'] = False

flight_data_df.groupby(['DEP_DELAYED'])['DEP_DELAYED'].count().sort_index()
```

Out[8]:

```
DEP_DELAYED
False    481039
True     120522
Name: DEP_DELAYED, dtype: int64
```

Adding a new column 'DELAY_REASON' that tells the reason for a flight getting delayed

In [9]:

```
flight_data_df['DELAY_REASON'] = np.where(flight_data_df.CARRIER_DELAY != 0, 'Carrier',
                                          np.where(flight_data_df.LATE_AIRCRAFT_DELAY !=
                                                   np.where(flight_data_df.WEATHER_DELAY
                                                            np.where(flight_data_df.NAS_
                                                                     np.where(flight_dat

flight_data_df.groupby(['DELAY_REASON'])['DELAY_REASON'].count().sort_index()
```

Out[9]:
```
DELAY_REASON
                   477611
Carrier             74794
LateAircraft        26097
NAS                 18695
Security              142
Weather              4222
Name: DELAY_REASON, dtype: int64
```

## Implementing arithmetic functions for statistical analysis

Creating a new dataframe with total number of flights per operating carrier to calculate the %

In [10]:
```
flight_totals = flight_data_df.value_counts(subset=['OP_UNIQUE_CARRIER']).reset_index()
flight_totals_df = pd.DataFrame(flight_totals)
flight_totals_df.columns = ['OP_UNIQUE_CARRIER','TOTAL']
flight_totals_df['PERCENTAGE'] = round(flight_totals_df.TOTAL/flight_totals_df.TOTAL.sum

flight_totals_df = flight_totals_df.sort_values('PERCENTAGE',ascending=False)
flight_totals_df.head(5)
```

Out[10]:

| | OP_UNIQUE_CARRIER | TOTAL | PERCENTAGE |
|---|---|---|---|
| **0** | WN | 107950 | 17.94 |
| **1** | DL | 76021 | 12.64 |
| **2** | AA | 71471 | 11.88 |
| **3** | OO | 66615 | 11.07 |
| **4** | UA | 53535 | 8.90 |

Calculate percentage by carrier and flight status

In [11]:
```
flight_status = flight_data_df.value_counts(subset=['OP_UNIQUE_CARRIER','STATUS']).reset
flight_status_df = pd.DataFrame(flight_status) #create a dataframe
flight_status_df.columns = ['OP_UNIQUE_CARRIER','STATUS', 'COUNT'] #Add column names
flight_status_df = flight_status_df.sort_values('OP_UNIQUE_CARRIER') #Sort by operating

flight_status_df['PERCENTAGE'] = ''

for index, row in flight_status_df.iterrows():
    tot = flight_totals.loc[flight_totals.OP_UNIQUE_CARRIER==row.OP_UNIQUE_CARRIER].TOTA
    val = (row.COUNT/tot * 100)
    flight_status_df.at[index,'PERCENTAGE'] = round(val[0].astype(float),2) #Calculate t

flight_status_df.head(10)
```

Out[11]:

| | OP_UNIQUE_CARRIER | STATUS | COUNT | PERCENTAGE |
|---|---|---|---|---|
| **33** | 9E | Delayed | 3113 | 15.33 |
| **48** | 9E | Cancelled | 542 | 2.67 |
| **74** | 9E | Diverted | 35 | 0.17 |
| **8** | 9E | On-Time | 16613 | 81.83 |
| **41** | AA | Cancelled | 973 | 1.36 |
| **56** | AA | Diverted | 215 | 0.3 |
| **3** | AA | On-Time | 55403 | 77.52 |

| | | | | |
|---|---|---|---|---|
| **11** | AA | Delayed | 14880 | 20.82 |
| **47** | AS | Cancelled | 608 | 3.12 |
| **10** | AS | On-Time | 15502 | 79.49 |

## Bar chart for carier performance in May 2022

In [28]:

```
fig = px.bar(flight_status_df, x="OP_UNIQUE_CARRIER", y="PERCENTAGE", title="Carrier Per
             color="STATUS", text="STATUS",
             labels={"OP_UNIQUE_CARRIER": "Operating Carrier",
                     "PERCENTAGE": "Percentage (%)"})
fig.update_layout(autosize=False,width=900, height=600)
fig.show()
```

Hawaian airlines had the best on-time performance in May'22 followed by Air Wisconsin(ZW). Frontier airlines(F9) had the most number of delays at 32.9% GoJet had the most cancellations at 7%

## Pie chart for Overall Carrier performance in May'22

In [13]:

```
fig = px.pie(flight_totals_df, values='PERCENTAGE', names='OP_UNIQUE_CARRIER',
             title='Overall Operating Carrier Performance (May''22)')
```

```
fig.update_traces(textposition='inside', textinfo='percent+label')
fig.show()
```

We can see southwest carrier (WN) had the most number of flights in May 2022.

## Bar plot for Airline with best on-time performance

In [34]:
```
airline_on_time_performance = flight_status_df[flight_status_df.STATUS == 'On-Time'].sor

fig=px.bar(airline_on_time_performance,
           x=airline_on_time_performance.OP_UNIQUE_CARRIER,
           y=airline_on_time_performance.PERCENTAGE, title="Airline On-Time Performance"
           text=airline_on_time_performance.PERCENTAGE.apply(lambda x: '{0:1.2f}%'.forma
           labels=dict(OP_UNIQUE_CARRIER="Airline Carrier", PERCENTAGE="Percentage (%)")
fig.update_xaxes(tickangle=45)
fig.update_layout(autosize=False,width=900, height=700)
```

Hawaiian airline was the best performing airline in May'22 with 87.33% on time performance and Go-Jet is the least performing airline with 64.6% on-time performance.

In [14]:
```
#Load csv file with airport names for origin and destination
airport_data_df = pd.read_csv('L_AIRPORT.csv')
airport_data_df.head()
```

Out[14]:

| | Code | Description |
|---|---|---|
| **0** | 01A | Afognak Lake, AK: Afognak Lake Airport |
| **1** | 03A | Granite Mountain, AK: Bear Creek Mining Strip |
| **2** | 04A | Lik, AK: Lik Mining Camp |
| **3** | 05A | Little Squaw, AK: Little Squaw Airport |
| **4** | 06A | Kizhuyak, AK: Kizhuyak Bay |

In [15]:
```
#Create a new dataframe with the percentage by origin airport and status
flight_origin_totals = flight_data_df.value_counts(subset=['ORIGIN']).reset_index()   #ge
flight_origin_totals_df = pd.DataFrame(flight_origin_totals)#create a dataframe
flight_origin_totals_df.columns = ['ORIGIN','TOTAL']#Add column names
#Calculate the percentage by origin airport
flight_origin_totals_df['PERCENTAGE'] = round(flight_origin_totals_df.TOTAL/flight_origi


origin_airport_delays = flight_data_df.value_counts(subset=['ORIGIN','STATUS']).reset_in
origin_airport_df = pd.DataFrame(origin_airport_delays) #create a dataframe
origin_airport_df.columns = ['ORIGIN','STATUS', 'COUNT'] #add column names
origin_airport_df = origin_airport_df.sort_values('ORIGIN') #sort by origin
origin_airport_df['PERCENTAGE'] = ''
```

```
for index, row in origin_airport_df.iterrows():
    tot = flight_origin_totals.loc[flight_origin_totals.ORIGIN==row.ORIGIN].TOTAL.values
    val = (row.COUNT/tot * 100)
    origin_airport_df.at[index,'PERCENTAGE'] = round(val[0].astype(float),2)  #calulate

origin_airport_df.head(10)
origin_airport_df = origin_airport_df.sort_values('PERCENTAGE',ascending=False) #sort by

#Add the airport name from the airport_data_df and add as a new column to the origin_air
origin_airport_df=pd.merge(origin_airport_df, airport_data_df, how='left', left_on='ORIG
origin_airport_df.rename(columns={'Description':'ORIGIN_AIRPORT_NAME'}, inplace=True)
del origin_airport_df['Code']

new = origin_airport_df.ORIGIN_AIRPORT_NAME.str.split(":", n = 1, expand = True)
origin_airport_df["ORIGIN_AIRPORT_NAME"] =  new[1]
origin_airport_df.head()
```

Out[15]:

| | ORIGIN | STATUS | COUNT | PERCENTAGE | ORIGIN_AIRPORT_NAME |
|---|---|---|---|---|---|
| **0** | GST | On-Time | 12 | 100.0 | Gustavus Airport |
| **1** | STC | On-Time | 1 | 100.0 | St. Cloud Regional |
| **2** | LWS | On-Time | 95 | 96.94 | Lewiston Nez Perce County |
| **3** | BGM | On-Time | 30 | 96.77 | Greater Binghamton/Edwin A. Link Field |
| **4** | DRT | On-Time | 60 | 96.77 | Del Rio International |

## Bar chart for Origin airport with most delays

In [16]:

```
fig = px.bar(origin_airport_df[origin_airport_df.STATUS=="Delayed"], x="ORIGIN_AIRPORT_N
             title="Origin Airport with most Delays",
             text=origin_airport_df[origin_airport_df.STATUS=="Delayed"].PERCENTAGE.appl
             labels=dict(ORIGIN_AIRPORT_NAME="Origin Airport", PERCENTAGE="Percentage (%
fig.update_xaxes(tickangle=80)
fig.update_layout(autosize=False,width=900, height=700)
fig.show()
```

It appears Tri Cities has multiple entries for different origin airports. Identify and update the airport name.

In [17]:
```
origin_airport_df[origin_airport_df.ORIGIN_AIRPORT_NAME.str.contains('Tri Cities')]
```

Out[17]:

| | ORIGIN | STATUS | COUNT | PERCENTAGE | ORIGIN_AIRPORT_NAME |
|---|---|---|---|---|---|
| **29** | PSC | On-Time | 451 | 90.56 | Tri Cities |
| **207** | TRI | On-Time | 302 | 81.4 | Tri Cities |
| **506** | TRI | Delayed | 66 | 17.79 | Tri Cities |
| **708** | PSC | Delayed | 44 | 8.84 | Tri Cities |
| **1018** | TRI | Cancelled | 3 | 0.81 | Tri Cities |
| **1093** | PSC | Diverted | 2 | 0.4 | Tri Cities |
| **1178** | PSC | Cancelled | 1 | 0.2 | Tri Cities |

Updating the airport name for PSC

In [18]:
```
origin_airport_df.loc[origin_airport_df["ORIGIN"] == "PSC", "ORIGIN_AIRPORT_NAME"] = 'Tr
```

In [19]:
```
origin_airport_df[origin_airport_df.ORIGIN_AIRPORT_NAME.str.contains('Tri Cities')]
```

Out[19]:

| | ORIGIN | STATUS | COUNT | PERCENTAGE | ORIGIN_AIRPORT_NAME |
|---|---|---|---|---|---|
| **29** | PSC | On-Time | 451 | 90.56 | Tri Cities(PSC) |
| **207** | TRI | On-Time | 302 | 81.4 | Tri Cities |
| **506** | TRI | Delayed | 66 | 17.79 | Tri Cities |
| **708** | PSC | Delayed | 44 | 8.84 | Tri Cities(PSC) |
| **1018** | TRI | Cancelled | 3 | 0.81 | Tri Cities |
| **1093** | PSC | Diverted | 2 | 0.4 | Tri Cities(PSC) |
| **1178** | PSC | Cancelled | 1 | 0.2 | Tri Cities(PSC) |

Since the chart has many airports to fit, filtering the list to get the top 10 origin airports with most delays

In [20]:
```
top_10_origin_delay_airports = origin_airport_df[origin_airport_df.STATUS=='Delayed'].he
top_10_origin_delay_airports
```

Out[20]:

| | ORIGIN | STATUS | COUNT | PERCENTAGE | ORIGIN_AIRPORT_NAME |
|---|---|---|---|---|---|
| **344** | PPG | Delayed | 2 | 66.67 | Pago Pago International |
| **361** | IAG | Delayed | 22 | 57.89 | Niagara Falls International |
| **368** | SCK | Delayed | 22 | 46.81 | Stockton Metro |
| **370** | HGR | Delayed | 9 | 45.0 | Hagerstown Regional-Richard A. Henson Field |
| **371** | RFD | Delayed | 22 | 42.31 | Chicago/Rockford International |
| **373** | BLV | Delayed | 45 | 42.06 | Scott AFB MidAmerica St Louis |
| **374** | PSE | Delayed | 39 | 41.94 | Mercedita |
| **375** | PQI | Delayed | 22 | 41.51 | Presque Isle International |
| **376** | USA | Delayed | 30 | 40.54 | Concord Padgett Regional |
| **378** | RIW | Delayed | 14 | 40.0 | Central Wyoming Regional |

In [35]:

```
fig = px.bar(top_10_origin_delay_airports[top_10_origin_delay_airports.STATUS=="Delayed"
             title="Top 10 Origin Airport with most Delays",
             text=top_10_origin_delay_airports[top_10_origin_delay_airports.STATUS=="Del
             labels=dict(ORIGIN_AIRPORT_NAME="Origin Airport", PERCENTAGE="Percentage (%
fig.update_xaxes(tickangle=80)
fig.update_layout(autosize=False,width=900, height=700)
fig.show()
```

Flights originating from Pago Pago International are delayed 66.67%

DESTINATION

In [22]:

```
#Create a new dataframe with the percentage by origin airport and status
flight_dest_totals = flight_data_df.value_counts(subset=['DEST']).reset_index() #get the
flight_dest_totals_df = pd.DataFrame(flight_dest_totals)#create a dataframe
flight_dest_totals_df.columns = ['DEST','TOTAL']#Add column names
#Calculate the percentage by destination airport
flight_dest_totals_df['PERCENTAGE'] = round(flight_dest_totals_df.TOTAL/flight_dest_tota


dest_airport_delays = flight_data_df.value_counts(subset=['DEST','STATUS']).reset_index(
dest_airport_df = pd.DataFrame(dest_airport_delays) #create a dataframe
dest_airport_df.columns = ['DEST','STATUS', 'COUNT']#add column names
dest_airport_df = dest_airport_df.sort_values('DEST')#sort by destination
dest_airport_df['PERCENTAGE'] = ''

for index, row in dest_airport_df.iterrows():
    tot = flight_dest_totals.loc[flight_dest_totals.DEST==row.DEST].TOTAL.values  #get t
    val = (row.COUNT/tot * 100)
    dest_airport_df.at[index,'PERCENTAGE'] = round(val[0].astype(float),2) #calulate the

dest_airport_df.head(10)
dest_airport_df = dest_airport_df.sort_values('PERCENTAGE',ascending=False)#sort by perc

#Add the airport name from the airport_data_df and add as a new column to the dest_airpo
dest_airport_df=pd.merge(dest_airport_df, airport_data_df, how='left', left_on='DEST', r
dest_airport_df.rename(columns={'Description':'DEST_AIRPORT_NAME'}, inplace=True)
del dest_airport_df['Code']

new = dest_airport_df.DEST_AIRPORT_NAME.str.split(":", n = 1, expand = True)
dest_airport_df["DEST_AIRPORT_NAME"] =  new[1]
dest_airport_df.head()
```

Out[22]:

| | DEST | STATUS | COUNT | PERCENTAGE | DEST_AIRPORT_NAME |
|---|---|---|---|---|---|
| **0** | GST | On-Time | 12 | 100.0 | Gustavus Airport |
| **1** | STC | On-Time | 1 | 100.0 | St. Cloud Regional |
| **2** | PPG | Delayed | 3 | 100.0 | Pago Pago International |
| **3** | TWF | On-Time | 31 | 96.88 | Joslin Field - Magic Valley Regional |
| **4** | PIH | On-Time | 30 | 96.77 | Pocatello Regional |

## Bar chart for Destination Airports with most delays

In [23]:

```
fig = px.bar(dest_airport_df[dest_airport_df.STATUS=="Delayed"], x="DEST_AIRPORT_NAME",
             title="Destination Airport with most Delays",
             text=dest_airport_df[dest_airport_df.STATUS=="Delayed"].PERCENTAGE.apply(la
             labels=dict(DEST_AIRPORT_NAME="Destination Airport", PERCENTAGE="Percentage
fig.update_xaxes(tickangle=80)
```

```
fig.update_layout(autosize=False,width=900, height=700)
fig.show()
```

Updating Destination name for PSC

In [24]:
```
dest_airport_df[dest_airport_df.DEST_AIRPORT_NAME.str.contains('Tri Cities')]
dest_airport_df.loc[dest_airport_df["DEST"] == "PSC", "DEST_AIRPORT_NAME"] = 'Tri Cities
dest_airport_df[dest_airport_df.DEST_AIRPORT_NAME.str.contains('Tri Cities')]
```

Out[24]:

| | DEST | STATUS | COUNT | PERCENTAGE | DEST_AIRPORT_NAME |
|---|---|---|---|---|---|
| **40** | PSC | On-Time | 439 | 88.15 | Tri Cities(PSC) |
| **125** | TRI | On-Time | 310 | 83.56 | Tri Cities |
| **610** | TRI | Delayed | 57 | 15.36 | Tri Cities |
| **680** | PSC | Delayed | 59 | 11.85 | Tri Cities(PSC) |
| **985** | TRI | Cancelled | 4 | 1.08 | Tri Cities |

Since the chart has many airports to fit, filtering the list to get the top 10 destination airports with most delays

In [25]:
```
top_10_dest_delay_airports = dest_airport_df[dest_airport_df.STATUS=='Delayed'].head(10)
top_10_dest_delay_airports
```

Out[25]:

| | DEST | STATUS | COUNT | PERCENTAGE | DEST_AIRPORT_NAME |
|---|---|---|---|---|---|
| **2** | PPG | Delayed | 3 | 100.0 | Pago Pago International |
| **368** | PGD | Delayed | 212 | 46.9 | Punta Gorda Airport |
| **369** | PSE | Delayed | 42 | 45.16 | Mercedita |
| **372** | SMX | Delayed | 4 | 40.0 | Santa Maria Public/Capt. G. Allan Hancock Field |
| **373** | PQI | Delayed | 20 | 40.0 | Presque Isle International |
| **374** | PSM | Delayed | 10 | 38.46 | Portsmouth International at Pease |
| **375** | EWR | Delayed | 5097 | 37.31 | Newark Liberty International |
| **376** | BKG | Delayed | 3 | 33.33 | Branson Airport |
| **377** | SCK | Delayed | 15 | 32.61 | Stockton Metro |
| **378** | USA | Delayed | 24 | 32.43 | Concord Padgett Regional |

## Bar chart for Destination Airprot with most delays

In [36]:
```
fig = px.bar(top_10_dest_delay_airports[top_10_dest_delay_airports.STATUS=="Delayed"], x
             title="Top 10 Destination Airport with most Delays",
             text=top_10_dest_delay_airports[top_10_dest_delay_airports.STATUS=="Delayed
             labels=dict(DEST_AIRPORT_NAME="Destination Airport", PERCENTAGE="Percentage
fig.update_xaxes(tickangle=80)
fig.update_layout(autosize=False,width=900, height=700)
fig.show()
```

All flights flying into Pago Pago Internation airport hasare delayed.

## Histogram for Overall cancellations by cancellation reason

In [27]:
```
#Using CANCELLED==1 to filter the dataframe for only cancelled rows.
fig = px.histogram(flight_data_df[flight_data_df.CANCELLED==1], x="CANCELLATION_REASON",
                   title="Number of Cancellation by Reasons",
                   labels=dict(CANCELLATION_REASON="Cancellation Reason"))
fig.show()
```

From the chart, we can see that most cancellations in May'22 were due to carriers followed by weather

## Conclusion

It appears we have enough information at this time to apply different models on the dataframe to be able to predict a flights performance. From the different charts, we can also see the best performing carriers, airports with most delays for arrivals and departures and reasons for cancellations.