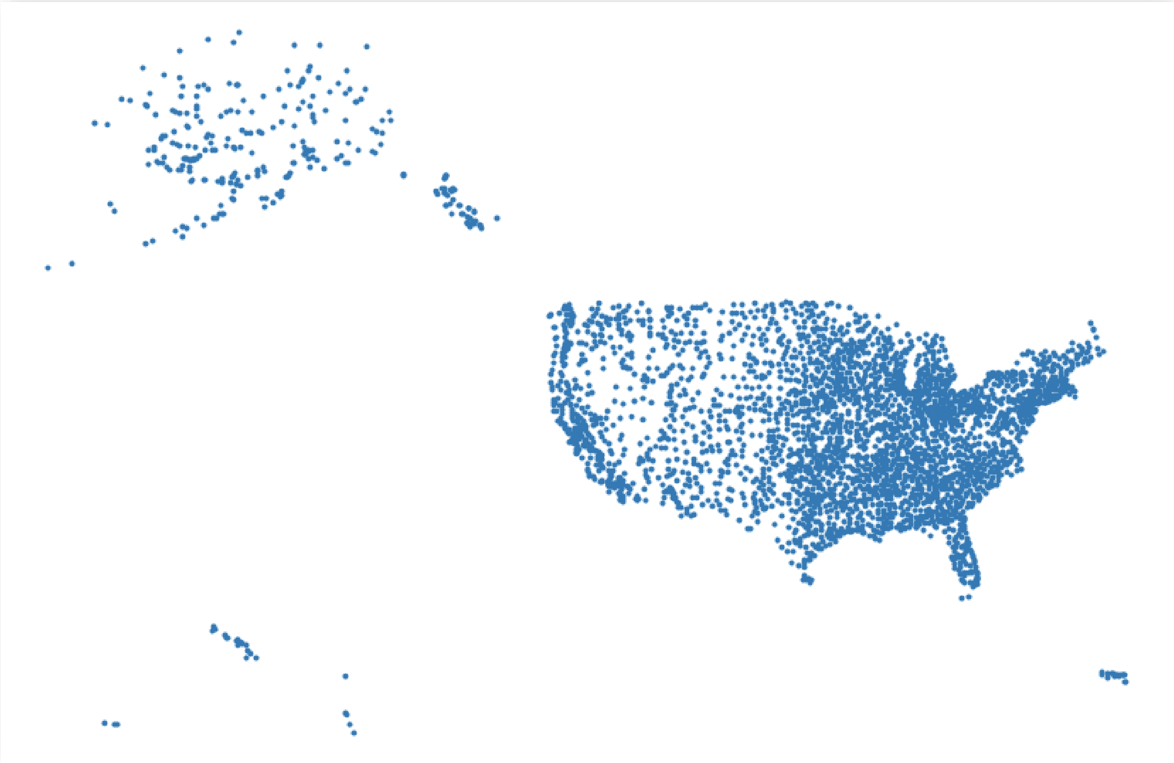
Aviation Analysis System (AAS)

Understanding USA flight activity in February 2020



Developed by Team 7:

- Clara Yuan (21803572)
- Georgia-Rose Collins (20763699)
- Kristy Gray (23123867)
- Shayne Bates (22563009)

Automating the exploration of USA Flight behaviours

There are over 3000 commercial airports in the USA, and in February of 2020, **574,268 flights** were scheduled to travel. It takes effective management and coordination to ensure that the maximum number of flights travel within intended timeframes. Passengers expect flights to be punctual and will be more likely to fly if the on-time arrival rate is high. Setting KPIs and achieving these expectations require specific goals focusing on increasing on-time arrivals. Appropriate parties require a deep understanding of current flight statistics, delays' causes, and potential location influences. This report details the scripts used to code and innovate the Aviation Analysis System (AAS) to automate this process.

Introducing the Aviation Analysis System (AAS)

The AAS is a combination of easy-to-use functions, which enables the end-user to identify operation flight issues. Examples of AAS functionality:

- A user can establish that in February 2020:
  - the **total number of delayed flights in the USA was 84, 616**
    - with a **total time delay of 5, 819, 054 minutes**
  - the airport with the largest number of delayed flights was **Hartsfield-Jackson Atlanta International with 4609 flights**
    - this airport had the **longest delay time of 352, 569 minutes**
    - this airport was located at **longitude -84.42694444, latitude 33.6404444**
  - a bottleneck occurred at **Dallas / Fort Worth International airport,** with the highest number of delayed flights in Texas **(3838 delayed flights)**
- Delays can be further investigated by **visualising the proportions of each delay cause,** using a pie-chart (fig 1. Flight Punctuality in the USA, February 2020)

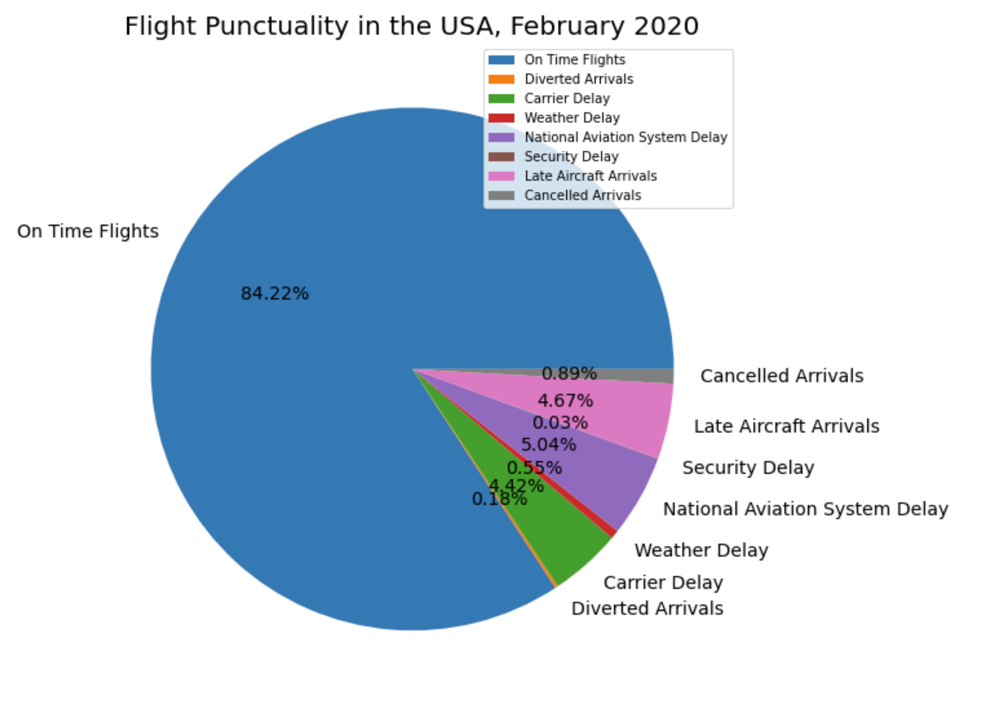


Figure 1 – AAS generated pie-chart: Flight Punctuality in the USA, February 2020

The Aviation Analysis System (ASS) gives the commercial aviation industry continuous access to user-friendly code, enabling them to assess issues and put steps into place to mitigate them.

Importing USA airport data from a CSV file

We commenced our study by flowcharting an algorithm (fig. 2) to establish how to code the extraction of data from a comma-separated (CSV) file. The algorithm extracts airports and location coordinates from the CSV, then and writes them to a text file.

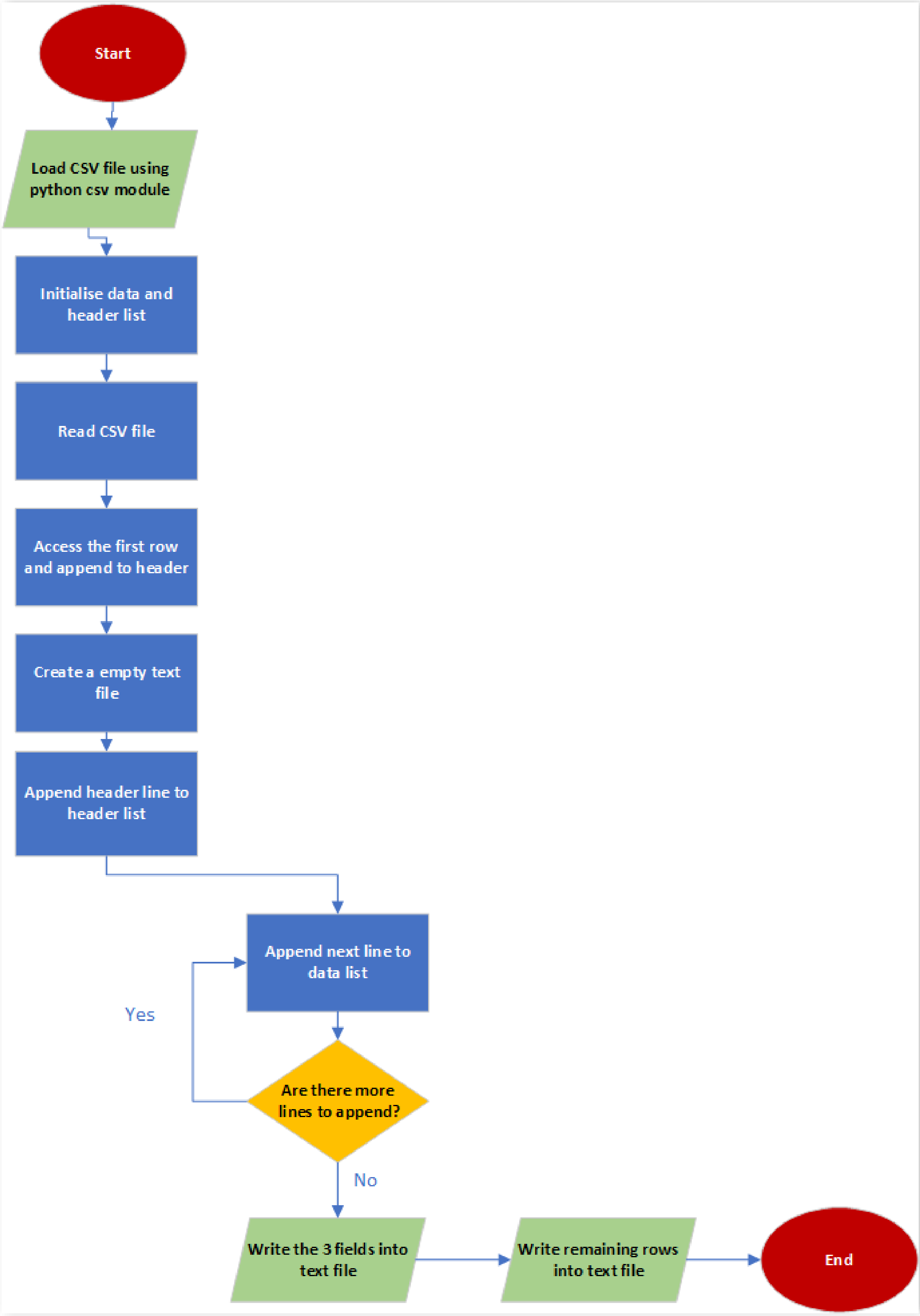


Figure 2 - Part A, Question 1 – Flowcharting an algorithm to extract coordinates of airports

Using the algorithm, the AAS script airportname_coordinates (fig. 3) was created to import/read a CSV file, export data, and write to text file.

```python
def airportname_coordinates (importcsvfile, newtxtfile):
    """
    This function takes a csv file as input and creates a text file with the name you pass in.

    step 1: update your current working directory to correct location
    step 2: ensure your datafile is within inthis current working directory

    This function has 2 arguments:
    1) importcsvfile = the name of your existing csv file to import. Naming convention = "filename.csv"
    2) newtxtfile = the name of the text file you would like to create. Naming convention = "txtfilename.txt"

    """
    import csv
    global data
    global header
    header, data = [ ],[ ]

    with open (importcsvfile, "r") as csv_file:
        csv_reader = csv.reader(csv_file)
        header.append(next(csv_reader))
        for line in csv_reader:
            data.append(line)

    with open(newtxtfile,"w") as airporttextfile:
        for elm in header:
            airporttextfile.write(f"{elm[1]} {elm[5]} {elm[6]}\n")
        for elm in data:
            airporttextfile.write(f"\n{elm[1]} {elm[5]} {elm[6]}\n")

# RUN the code:
airportname_coordinates("airports.csv","airporttextfiletest.txt")
```

```
airporttextfiletest - Notepad
File  Edit  Format  View  Help
airport lat long

Thigpen  31.95376472 -89.23450472

Livingston Municipal 30.68586111 -95.01792778

Meadow Lake 38.94574889 -104.5698933

Perry-Warsaw 42.74134667 -78.05208056

Hilliard Airpark 30.6880125 -81.90594389

Tishomingo County 34.49166667 -88.20111111

Gragg-Wade  32.85048667 -86.61145333

Capitol 43.08751 -88.17786917

Columbiana County 40.67331278 -80.64140639

Memphis Memorial 40.44725889 -92.22696056

Calhoun County 33.93011222 -89.34285194
```

Figure 3 - Part A - Question 2 - AAS Script airportname_coordinates to import, extract and write data to a textfile

Understanding the distribution of airports in the USA

The AAS script airport_scatterplot (fig. 4) transfers all commercial airport locations in the USA to a scatterplot (fig. 5), utilising the data extracted from the CSV in the airportname_coordinates (fig. 3) script.

```python
def airport_scatterplot (importcsvfile, newtxtfile):
    """
    This function uses airportname_coordinates function to import data and create a scatterplot
    airportname_coordinates function takes a csv file as input and creates a text file with the name you pass in.

    step 1: update your current working directory to correct location
    step 2: ensure your datafile is within inthis current working directory

    This function has 2 arguments:
    1) importcsvfile = the name of your existing csv file to import. Naming convention = "filename.csv"
    2) newtxtfile = the name of the text file you would like to create. Naming convention = "txtfilename.txt"

    """
    airportname_coordinates(importcsvfile, newtxtfile)
    import matplotlib.pyplot as plt

    latitude, longitude = [ ], [ ]
    for elm in data:
        if float(elm[-1])<0:
            #Latitude of Y-axis
            latitude.append(float(elm[-2]))
            #Longitude of X-axis
            longitude.append(float(elm[-1]))

    #plot the scatterport
    plt.figure(figsize=(15,10))
    plt.scatter(longitude,latitude,marker='.')
    plt.xlabel("Longitude")
    plt.ylabel("Latitude")
    plt.title("Distribution of Airports in the US", size = 20, color = 'steelblue')
    plt.show()

airport_scatterplot ("airports.csv","airporttextfiletest.txt")
```

Figure 4 - Part A, Question 3 – AAS Script airport_scatterplot create a Scatterplot for USA Airports

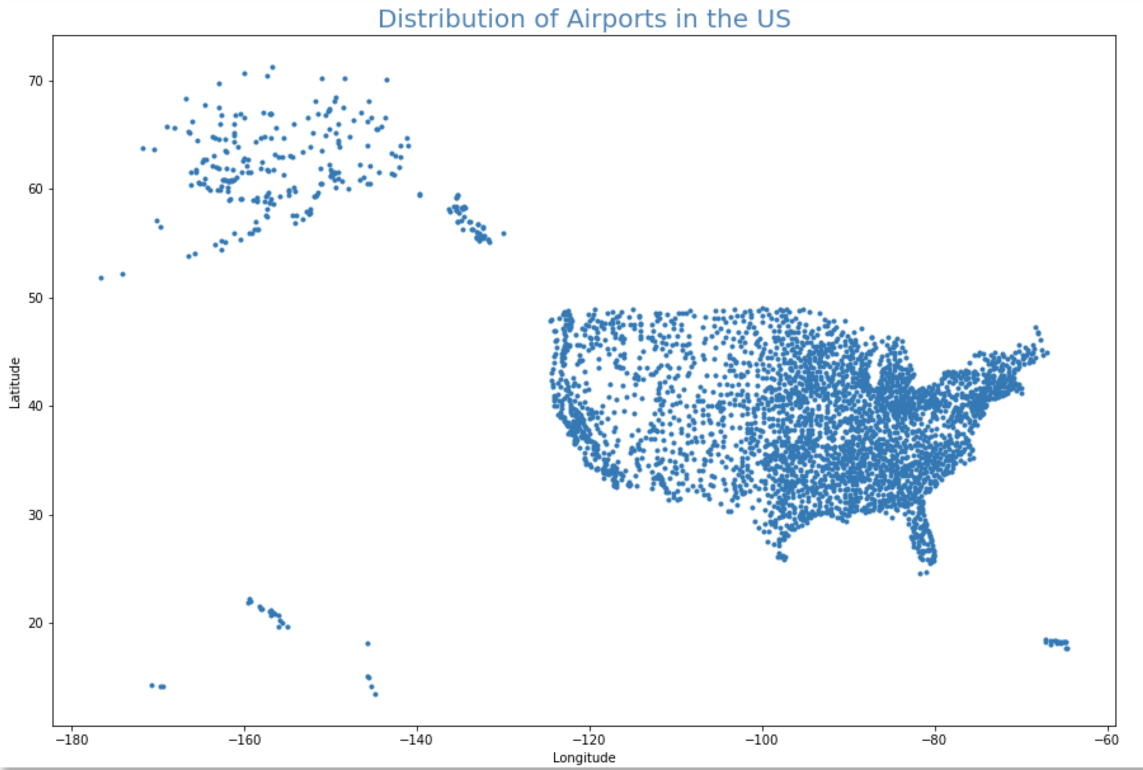


Figure 5 - Part A, Question 3 – Visualisation of data from AAS script airport_scatterplot

Preparing the airport data *from two datasets* for analysis

USA flight data was analysed from February 2020 by importing and processing two data sets (fig. 6) into the AAS. Within the system, for efficiency the commonly used **'airline_delay_causes' CSV file was defined as df_1** and the generic '**airports' CSV was defined as df_2.**

```
In [2]:
import os
import pandas as pd
import numpy as np
df_1 = pd.read_csv('airports.csv')
df_2 = pd.read_csv('airline_delay_causes_Feb2020.csv', delimiter = ",")
```

Figure 6 - Part B - Importing the datasets and constructing dataframes into the AAS

A rigorous data integrity check was completed to explore the structure and to understand the organisation of df_1 in the AAS (fig. 7)

```
df_1.info() ### understand how the data is formatted.
```

```
<class 'pandas.core.frame.DataFrame'>
RangeIndex: 3376 entries, 0 to 3375
Data columns (total 7 columns):
 #   Column    Non-Null Count  Dtype
---  ------    --------------  -----
 0   iata      3376 non-null   object
 1   airport   3376 non-null   object
 2   city      3364 non-null   object
 3   state     3364 non-null   object
 4   country   3376 non-null   object
 5   lat       3376 non-null   float64
 6   long      3376 non-null   float64
dtypes: float64(2), object(5)
memory usage: 184.8+ KB
```

```
df_1.describe() ### further understand data
```

	lat	long
count	3376.000000	3376.000000
mean	40.036524	-98.621205
std	8.329559	22.869458
min	7.367222	-176.646031
25%	34.688427	-108.761121
50%	39.434449	-93.599425
75%	43.372612	-84.137519
max	71.285448	145.621384

Figure 7 - Part B - Exploring df_1 and understanding how data is structured within the AAS

The same approach was used for df_2 (fig. 8).

```
df_2.info() ### understand how the data is formatted.
```

```
<class 'pandas.core.frame.DataFrame'>
RangeIndex: 1769 entries, 0 to 1768
Data columns (total 21 columns):
 #   Column             Non-Null Count  Dtype
---  ------             --------------  -----
 0   year               1769 non-null   int64
 1   month              1769 non-null   int64
 2   carrier            1769 non-null   object
 3   carrier_name       1769 non-null   object
 4   airport            1769 non-null   object
 5   airport_name       1769 non-null   object
 6    arr_flights       1769 non-null   int64
 7   arr_del15          1767 non-null   float64
 8   carrier_ct         1769 non-null   float64
 9     weather_ct       1769 non-null   float64
 10  nas_ct             1769 non-null   float64
 11  security_ct        1769 non-null   float64
 12   late_aircraft_ct  1769 non-null   float64
 13  arr_cancelled      1769 non-null   int64
 14  arr_diverted       1769 non-null   int64
 15  arr_delay          1769 non-null   int64
 16  carrier_delay      1769 non-null   int64
 17  weather_delay      1769 non-null   int64
 18  nas_delay          1769 non-null   int64
 19  security_delay     1769 non-null   int64
 20  late_aircraft_delay  1769 non-null   int64
dtypes: float64(6), int64(11), object(4)
memory usage: 290.4+ KB
```

```
df_2.describe() ### further understand data
```

	year	month	arr_flights	arr_del15	carrier_ct
count	1769.0	1769.0	1769.000000	1767.000000	1769.000000
mean	2020.0	2.0	324.628604	47.886814	14.389158
std	0.0	0.0	896.920719	131.605956	34.388350
min	2020.0	2.0	1.000000	0.000000	0.000000
25%	2020.0	2.0	42.000000	6.000000	1.960000
50%	2020.0	2.0	87.000000	15.000000	5.160000
75%	2020.0	2.0	219.000000	35.000000	12.530000
max	2020.0	2.0	18334.000000	2605.000000	487.650000

Figure 8 – Part B – Exploring df_2 and understanding how data is structured within the AAS

Once understood, the missing values within each of the datasets were identified (fig. 9).

```
print(df_1[df_1.isnull().any(1)])

### displays the NAN values for df_1.
### decision: do not fill in gaps as the city/state for these datapoints are not relevant.
### Do not drop these rows as they are still required for analysis
```

```
     iata                      airport city state  \
1136  CLD   MC Clellan-Palomar Airport  NaN   NaN
1715  HHH                  Hilton Head  NaN   NaN
2251  MIB                    Minot AFB  NaN   NaN
2312  MQT      Marquette County Airport  NaN   NaN
2752  RCA                Ellsworth AFB  NaN   NaN
2759  RDR              Grand Forks AFB  NaN   NaN
2794  ROP                  Prachinburi  NaN   NaN
2795  ROR             Babelthoup/Koror  NaN   NaN
2900  SCE              University Park  NaN   NaN
2964  SKA                Fairchild AFB  NaN   NaN
3001  SPN  Tinian International Airport  NaN   NaN
3355  YAP            Yap International  NaN   NaN

                            country        lat        long
1136                            USA  33.127231 -117.278727
1715                            USA  32.224384  -80.697629
2251                            USA  48.415769 -101.358039
2312                            USA  46.353639  -87.395361
2752                            USA  44.145094 -103.103567
2759                            USA  47.961167  -97.401167
2794                       Thailand  14.078333  101.378334
2795                          Palau   7.367222  134.544167
2900                            USA  40.851206  -77.846302
2964                            USA  47.615058 -117.655803
3001                N Mariana Islands  14.996111  145.621384
3355  Federated States of Micronesia   9.516700  138.100000
```

```
print(df_2[df_2.isnull().any(1)])

### displays the NAN values for df_2.
### decision: fill in gaps with 0 as it is a logical value
```

```
      year  month carrier               carrier_name airport  \
568   2020      2      EV  ExpressJet Airlines LLC     PIA
1709  2020      2      YX         Republic Airline     GRB

                                          airport_name  arr_flights  \
568   Peoria, IL: General Downing - Peoria Internati...            1
1709  Green Bay, WI: Green Bay Austin Straubel Inter...            1

      arr_del15  carrier_ct   weather_ct  ...  security_ct  \
568         NaN         0.0          0.0  ...          0.0
1709        NaN         0.0          0.0  ...          0.0
```

Figure 9 - Part B - Identifying missing values – Excerpts of output

The specific values missing in df_1 do not impact this study, so these were not adjusted or removed. Analysis of the data missing in df_2 revealed that these values logically should be = 0 (due to the breakdown of the data). These missing values were filled with 0 (fig. 10).

```
df_2 = df_2.fillna(0) ### fills null with 0 for data in arr_del15
# explored NAN output above. Understood that these should be 0 values (above)
```

Figure 10 - Part B - Cleaning data by filling NaN values

For effective utilisation, the datasets were merged to consolidate information, defined as df_both (fig. 11).

```
df_2 = df_2.rename(columns={'airport': 'iata'}) ### renames columns so data set can be merged
df_both = df_2.merge(df_1, how = "left", on= 'iata')### merges data sets
```

Figure 11 - Part B - Merging dataframes df_1 and df_2

The merged dataset df_both was explored (fig. 12) and analysis of this dataset commenced.

```
df_both.info()
```

```
<class 'pandas.core.frame.DataFrame'>
Int64Index: 1769 entries, 0 to 1768
Data columns (total 27 columns):
 #   Column             Non-Null Count  Dtype
---  ------             --------------  -----
 0   year               1769 non-null   int64
 1   month              1769 non-null   int64
 2   carrier            1769 non-null   object
 3   carrier_name       1769 non-null   object
 4   iata               1769 non-null   object
 5   airport_name       1769 non-null   object
 6    arr_flights       1769 non-null   int64
 7   arr_del15          1769 non-null   float64
 8   carrier_ct         1769 non-null   float64
 9     weather_ct       1769 non-null   float64
 10  nas_ct             1769 non-null   float64
 11  security_ct        1769 non-null   float64
 12   late_aircraft_ct  1769 non-null   float64
 13  arr_cancelled      1769 non-null   int64
 14  arr_diverted       1769 non-null   int64
 15  arr_delay          1769 non-null   int64
 16  carrier_delay      1769 non-null   int64
 17  weather_delay      1769 non-null   int64
 18  nas_delay          1769 non-null   int64
 19  security_delay     1769 non-null   int64
 20  late_aircraft_delay 1769 non-null  int64
 21  airport            1758 non-null   object
 22  city               1751 non-null   object
 23  state              1751 non-null   object
 24  country            1758 non-null   object
 25  lat                1758 non-null   float64
 26  long               1758 non-null   float64
dtypes: float64(8), int64(11), object(8)
memory usage: 387.0+ KB
```

```
df_both.describe()
```

	year	month	arr_flights	arr_del15	carrier_ct	weather_ct
count	1769.0	1769.0	1769.000000	1769.000000	1769.000000	1769.000000
mean	2020.0	2.0	324.628604	47.832674	14.389158	1.774483
std	0.0	0.0	896.920719	131.541346	34.388350	6.470108
min	2020.0	2.0	1.000000	0.000000	0.000000	0.000000
25%	2020.0	2.0	42.000000	6.000000	1.960000	0.000000
50%	2020.0	2.0	87.000000	15.000000	5.160000	0.230000
75%	2020.0	2.0	219.000000	35.000000	12.530000	1.340000
max	2020.0	2.0	18334.000000	2605.000000	487.650000	110.080000

Figure 12 - Part B – Exploring merged dataframe df_both

Processing the merged data

Understanding and exploring the behaviours of flights in the USA required establishing a specific set of questions to create efficient scripts to automate this process. The merged dataset df_both was used to test these scripts.

Questions presented of the data and AAS scripts created to process these questions:

1. What is the total number of flights in the USA?

Including cancelled, diverted, delayed and on-time.

```
def flight_stats_totals (df, columnname, newcolumnname):
    """ pass in all as a string:
    (dataframe, index columnname, newcolumnname)

    df = the dataframe created from imported csv file
    columnname = the name of the column you want to sum  eg: ' arr_flights'
    newcolumnname = a reader-friendly name for the columnname

    This returns the total of the specified column in a dataframe.
    """
    columnsum = int(df[columnname].sum())
    return f"The total of {newcolumnname} = {columnsum}"
```

```
flight_stats_totals (df_both, " arr_flights", "all_flights")
```

```
'The total of all_flights = 574268'
```

2. What is the total number of delayed flights in the USA?

```
flight_stats_totals (df_both, "arr_del15", "delayed flights")
```

```
'The total of delayed flights = 84616'
```

3. What is the total delayed time (in minutes) of flights in the USA?

```
flight_stats_totals (df_both, "arr_delay", "delayed flights in minutes")
```

```
'The total of delayed flights in minutes = 5819054'
```

4. What is the airport with the largest number of delayed flights?

```
def airport_most_delayed_count (dataframe):
    """
    Pass in the desired dataframe.
    This function takes that dataframe and uses it to:
    1. find the airport with the highest number of delayed flights
    2. return the name of the airport and the number of flights delayed
    """

#group column airport_name
    df_largestdelay = dataframe.groupby("airport_name")["arr_del15"].sum()
    largestdelayed_airport = df_largestdelay.idxmax()
    largestdelayed_flights = int(df_largestdelay.max())
    return f"Airport with the largest number of delayed flights: {largestdelayed_airport}" + \
    f"{largestdelayed_flights} flights"

airport_most_delayed_count(df_both)
```

```
'Airport with the largest number of delayed flights: Atlanta, GA: Hartsfield-Jackson Atlanta International4609 flights'
```

5. What are the coordinates of the airport with the highest delayed time?

```
def airport_most_delayed_minutes (dataframe):
    """
    Pass in the desired dataframe.
    This function takes that dataframe and uses it to:
    1. find the airport with the highest delay time
    2. find the corresponding latitude and longitude of that airport
    3. return the name of the airport, the total delay in minutes, the longitude and latitude

    """

    #find airport with highest total delayed time
    df_highestdelayedtime = dataframe.groupby(["airport_name", "long", "lat"], as_index = False)["arr_delay"].sum()

    highestdelayed_index = df_highestdelayedtime['arr_delay'].idxmax()
    highestdelayed_airport = df_highestdelayedtime['airport_name'][highestdelayed_index]
    highest_long = df_highestdelayedtime['long'][highestdelayed_index]
    highest_lat = df_highestdelayedtime['lat'][highestdelayed_index]
    highestdelayed_time = df_highestdelayedtime['arr_delay'][highestdelayed_index]

    return f"The coordinates of the airport with highest delayed time is: longitude {highest_long}"+ \
    f" latitude {highest_lat} which is {highestdelayed_airport} with delayed time of {highestdelayed_time} minutes"

airport_most_delayed_minutes (df_both)
```

```
'The coordinates of the airport with highest delayed time is: longitude -84.42694444 latitude 33.64044444 which is Atlanta,
GA: Hartsfield-Jackson Atlanta International with delayed time of 352569 minutes'
```

6. What is the airport in Texas that has the highest number of delayed flights?

```
def Max_airport_delay_by_state(dataframe, state):
    """
    Pass in the desired dataframe.
    state = string acronym of state ID. eg: "tx" = texas
    This function takes the dataframe required to find the airport in Texas that
    has the largest number of delayed flights.
    """
    state = state.upper()
    df_state = dataframe[dataframe["airport_name"].str.contains(state)]

    airport_delay_state = df_state.groupby("airport_name")["arr_del15"].sum()
    max_airport = airport_delay_state.idxmax()

    largestno_delayed_flight = int(airport_delay_state.max())

    return f"The Airport in {state} that has the largest number of delayed flights is: {max_airport}"+ \
    f" with {largestno_delayed_flight} delayed flights"

Max_airport_delay_by_state(df_both, "TX")
```

```
'The Airport in TX that has the largest number of delayed flights is: Dallas/Fort Worth, TX: Dallas/Fort Worth International
with 3838 delayed flights'
```

7. What is the percentage breakdown of?

- On-time flights

```python
def difference_calculator(basevariable, differencelist, differencevariablenane):
    """
    This function takes a base variable, sums the values of a list and returns the difference of these values.

    Pass in:
    basevariable = the number to deduct the values from
    differencelist = define a list with a list of values to sum
    differencevariablename = string of the name for this new variable
    """

    global differencevariable
    differencevariable = basevariable - sum(differencelist)
    return differencevariable




#instantiate list
differencelist = [df_both["arr_del15"],df_both["arr_cancelled"],df_both["arr_diverted"]]
#run code
difference_calculator(df_both[" arr_flights"], differencelist, "on_time_flights")
```

- Delayed flights (over 15 minutes late)
    - air-carrier delays
    - weather delays
    - National Aviation System (NAS) delays
    - security delays
    - aircraft arriving late
- Cancelled flights
- Diverted flights

```python
def Piechart_creator(dataframe, x_datalist, x_datalabels, x_datacolours, pietitle, piesize):
    """
    This function creates a pie chart from the below passed in attributes:

    Pass in:
    The desired dataframe
    A list of elements for x axis
    A list of label names for x axis
    A list of colours for x axis
    A string for the title of the pie chart
    A number for the pie size

    """

    import matplotlib.pyplot as plt

    x = x_datalist
    fig=plt.figure(figsize=(10,10))
    ax=fig.subplots()

    label = x_datalabels
    my_colours = x_datacolours
    ax.pie(x, labels = label, autopct ='%0.2f%%', textprops={'fontsize': 14})
    ax.legend()

    plt.title(pietitle, size = piesize)
    plt.tight_layout()
    return plt.show()




#instantiate lists
x_datalist = [on_time_flights.sum(),df_both["arr_diverted"].sum(),df_both["carrier_ct"].sum(),
              df_both['  weather_ct'].sum(),df_both["nas_ct"].sum(),df_both["security_ct"].sum(),
              df_both[" late_aircraft_ct"].sum(),df_both["arr_cancelled"].sum()]

x_datalabels = ["On Time Flights","Diverted Arrivals", "Carrier Delay", "Weather Delay",
                "National Aviation System Delay", "Security Delay","Late Aircraft Arrivals",
                "Cancelled Arrivals"]

x_datacolours = ['grey','lightsteelblue','red','pink','yellow','orange','blue','lightblue']




#run code
Piechart_creator(df_both, x_datalist, x_datalabels, x_datacolours,
                             "Flight Punctuality in the USA, February 2020", 20)
```

Analysing the results

Establishing benchmarks based on historical data, coupled with collecting data via the ASS, will assist the aviation industry in establishing acceptable parameters. Users will be able to analyse data regularly to determine if airports and states are performing according to the benchmarks. Utilising the AAS, February 2020 data reveals:

Questions 1, 2, 3

Figure 13 - Bar Chart/Log Scale of total Flights in the USA

In February 2020, the USA had a total of 574,268 flights, which included on-time, cancelled, diverted, and delayed flights (fig. 13)

14.73% of these flights (84,616) were delayed, which were caused by carrier, weather, NAS, Security, and late arrival of aircraft.

Total flight delay time was 5,819,054 minutes, which equates to over 4,000 days' worth of delayed flights.

Questions 4, 5

The airport with the largest number of delayed flights was Hartsfield-Jackson International in Atlanta, Georgia. This airport had a total of 4609 flights, accounting for a little over 5% of all delayed flights in the USA. Hartsfield-Jackson Airport (fig. 14) also had the highest delayed time at 352,569 minutes (5,876.15 hours), representing 6% of the total delayed time in the USA.

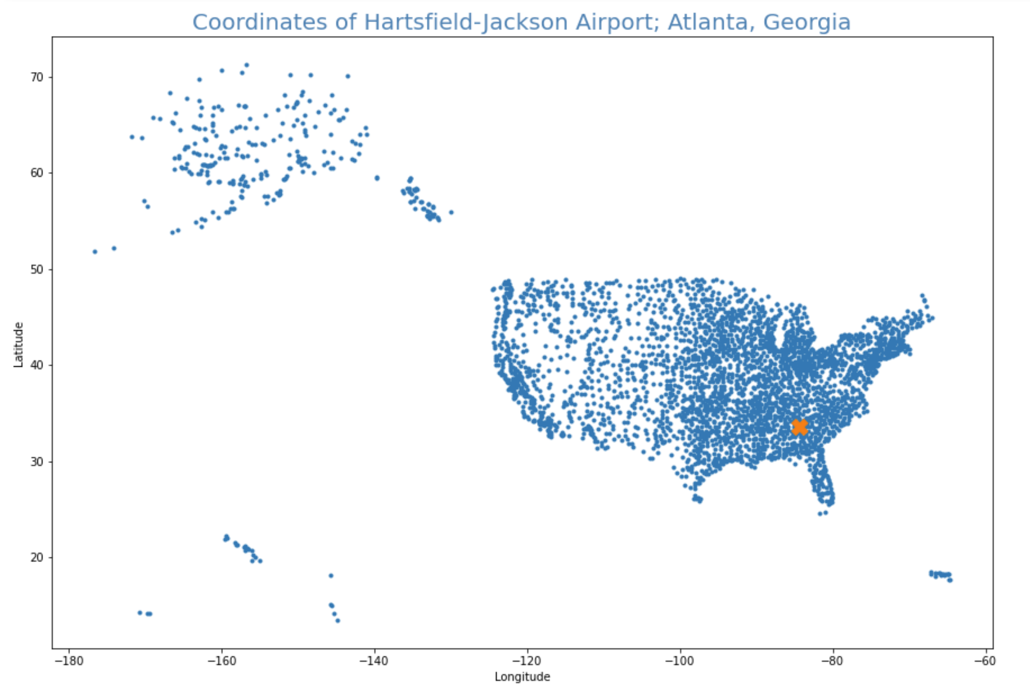


Figure 14 - Part B - Question 5 -
Coordinates of the airport with the highest delay time = X
Hartsfield-Jackson International (coordinates: -84.4269444 longitude x 33.6404444 latitude).

Question 6

Dallas / Fort Worth International Airport in Texas has a total of 3,838 delayed flights, which is 4.5% of the total delayed flights and 0.6% of all flights in the USA. Late aircraft was the primary cause of delay at this airport (fig. 15).

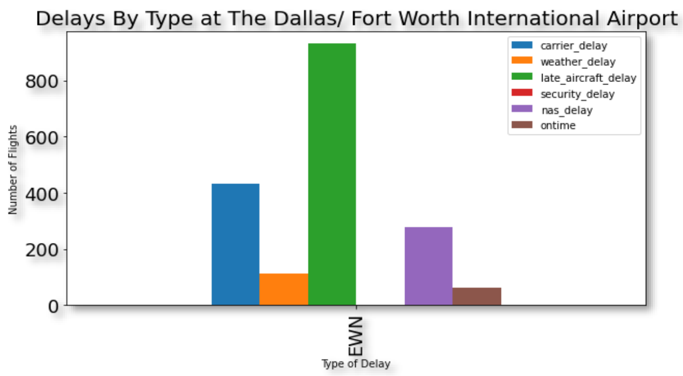


Figure 15 - Part B - Question 6 - Dallas/Fort Worth International Airport in Texas breakdown of delayed flights

Question 7

84.22% of flights were on time, 14.71% of all flights in the USA were delayed, 0. 89% were cancelled, and 0.18% were diverted (fig. 16). Delays in flights were caused by several factors (fig. 16), with National Aviation System Delay making up 5.04% of flights, late aircraft delays at 4.67% and carrier delays were 4.42%. The analysis suggests this is where improvements can be made. Weather and Security only had a negligible impact on delays, with 0.55% and 0.03%, respectively

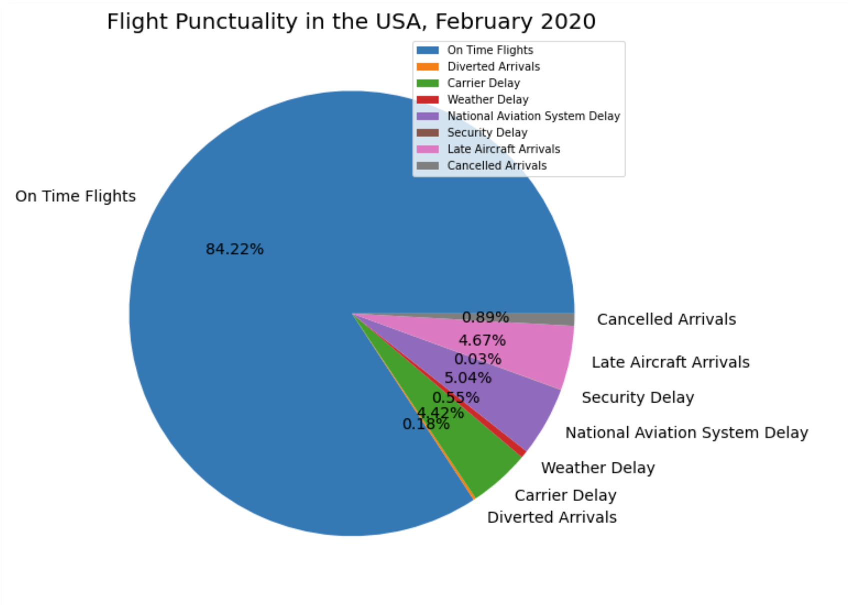


Figure 16 - Part B - Question 7 – Pie chart of flight punctuality in the USA

Leveraging the Aviation Analysis System

Passenger satisfaction is crucial in ensuring success in the airline industry. Flight delays, cancellations, and diversions form negative experiences for passengers, deterring them from travelling by aircraft. In February 2020, the on-time arrival rate for USA flights was approximately 21 in 25 (around 84%). In that month 84,616 flights were delayed, mostly caused by Late Aircraft Arrivals, National Aviation System (NAS) Delay and Carrier Delay. To improve passenger satisfaction (which increases patronage), on-time arrival rate should be increased. We recommend these Carriers, Aircrafts and the NAS utilise our Aviation Analysis System (AAS) to monitor performance, set benchmarks to compare to and incentivise their teams accordingly, to improve the national on-time arrival rate.