Assignment 2 Cloud Computing CA675

Data Analytics Application

1. Group 14

2. Member details

Name	Student ID	Email ID
Sachin Patel	22265660	sachin.patel3@mail.dcu.ie
Rohit Shinde	22260213	rohit.shinde2@mail.dcu.ie
Sneha Yadav	22268001	sneha.yadav4@mail.dcu.ie
Yash Modi	22260873	yash.modi2@mail.dcu.ie
Chirag Shah	22261247	chirag.shah2@mail.dcu.ie

3. Important Links

Project link- Project Link

Gitlab link- GitLab Repository Link

Link for the Dataset- Link

4. Introduction and Motivation

Flights are the best way to travel in the world from one place to another in a minimal amount of time. The requirement arises when someone has to travel a long distance then the best way to travel this far would be via flights. If someone has to take an emergency flight for any reason, this app will help them figure out which particular flight to take in less time considering external factors such as weather situations, flight delays, airport situations, plane status, etc

Choice of Technologies

Cloud Technology: Google Cloud Platform (dataproc cluster, bucket)

Database : Hive

Languages used in Hadoop: Python, HiveQL, Pig Latin

Framework: Jupyter Notebook

Libraries: Pandas, NumPy, Seaborn, Matplotlib, Voila, PySpark, Folium, Ipywidgets

5. Related Work

1. AirHelp

This application helps user to see if they are eligible to claim the money if the flight gets delayed or canceled.

2. Your airline's app

App lets us monitor our flight status but the main thing it provides is if the flight gets delay or cancel it sends notification about that and we can rebook the next flight but this app is only for US airlines.

3. Flight-Delay-Web-App: Link

This is a simple web application which will give information about whether there will be departure delay based on the factors like Origin, Destination, Airline, Terminal, Days of Week, Departure HOUR, etc.

Our proposed application does exploratory data analysis for the user to analyse the different attributes related to flight. With the help of this user can get the insights of flight details as per their requirements in an emergency. We have used the flight delay data of 2019 to work upon this solution. Our application can provide the top 10 airports with maximum availability which have less delay time. Users can explore available options and use it according to the need. It also provides various visuals to the user which helps the user to analyse easily and helps in better decision making.

6. Description of the dataset

a. Source of the data and Process of extraction/collection

Kaggle link for Dataset- Link

Cloud Platform System Setup : Used GCP Dataproc to create a cluster named “cluster-e182” with 1 master node and 2 slave nodes. First, we uploaded all the raw data files into a google cloud bucket named “ccass2”. At the same time, the additional components required for the project namely Jupyter, Hive and a component gateway were also configured.

Dataset preliminary observations and analysis :

This categorization dataset contains specific information about the weather, airports, airlines, and employee jobs. The issue is a binary classification evaluating a delayed departure when using the included train/test pre-combined data. A multiclass problem can be created by adding cancellation, specific delay reasons, and/or arrival delays to the dataset, which can be done by incorporating all raw data files. Since all weather data was manually downloaded, raw files for weather only contain the top 90% of airports for passenger volume.

In this flight delay dataset, we used the raw data files to explore in detail and arrange the data according to our needs and requirements.

Possible questions:

- Analyze the data set using exploratory data analysis. Which airlines are most and least dependable for leaving on time?
- Which are the best and the worst airports for timely departures?
- What are the aspects of the dataset which have the strongest correlations to departure delays?

b. Data pre-processing – preparation and cleaning

The dataset raw files were first uploaded to the google bucket and

From there we loaded them to the Hadoop file system with the below sample command:

-- for copying a file from the bucket to the Hadoop file system

hadoop fs -cp 'gs://ccass2/data/airportweather2019.csv' /data

-- for copying a file from the Hadoop file system to the local directory

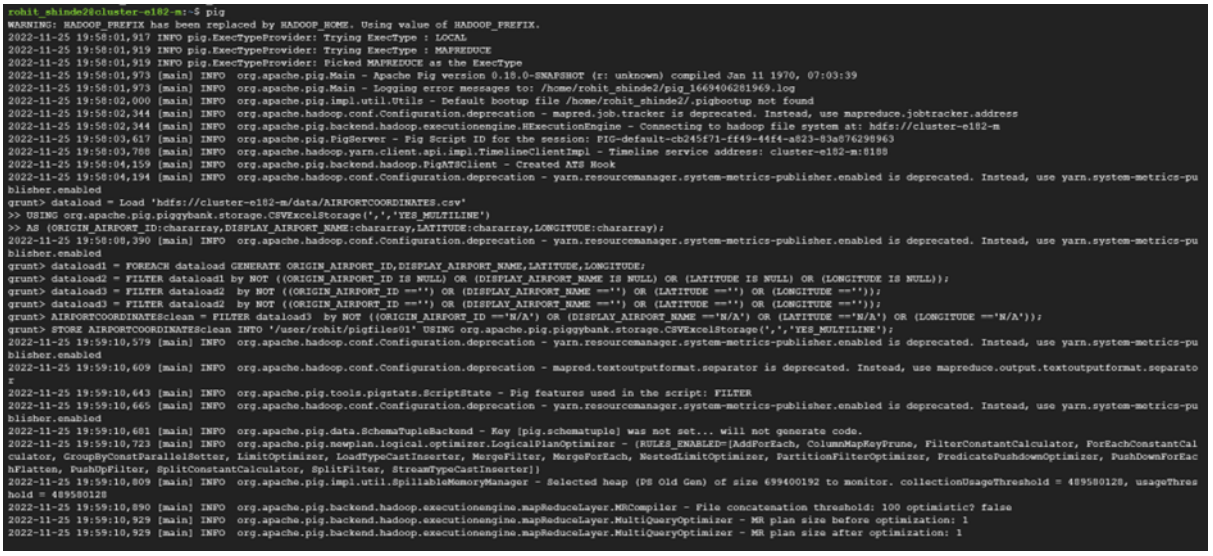
hadoop fs -get /data/airportweather2019.csv /home/rohit_shinde2/data/

For data loading and basic cleaning purposes we used pig:

Once the file is cleaned, loaded that file to local file directory folder name “cleandata”.

--for copying the final clean file from the Hadoop file system to a local directory

hadoop fs -getmerge /user/rohit/pigfiles04/
/home/rohit_shinde2/cleandata/airportweather2019.csv



```
rohit_shinde2@cluster-e182-m:~$ hadoop fs -ls /user/rohit/pigfiles01/
Found 2 items
-rw-r--r--   2 rohit_shinde2 hadoop          0 2022-11-25 19:59 /user/rohit/pigfiles01/_SUCCESS
-rw-r--r--   2 rohit_shinde2 hadoop     868902 2022-11-25 19:59 /user/rohit/pigfiles01/part-m-00000
rohit_shinde2@cluster-e182-m:~$ hadoop fs -rm /user/rohit/pigfiles01/_SUCCESS
Deleted /user/rohit/pigfiles01/_SUCCESS
rohit_shinde2@cluster-e182-m:~$
```

7. Description of data processing

Relevant data Information:-

As there is tons of data generated by the jet engine per minute which is approximately 2TB per minute and this streaming data usually are stored in the Big Data platforms. The air carrier sends weather conditions data and projectile data and real time data analysis is being done to avoid delays or failures which may arise. In one of the Data Expo challenges the flight delay related data was given for analysis.

The similar flight delay data of the airports in the states of the USA was publicly available on Kaggle for analysis. We fetched the same data for our application of flight delay analysis.

Dataset Information:-

The 2019 Airline Delays w/Weather and Airport Detail dataset downloaded from Kaggle for the analysis. This dataset contains following raw CSV files:

- airports_list
- B43_AIRCRAFT_INVENTORY
- CARRIER_DECODE
- P10_EMPLOYEES
- T3_AIR_CARRIER_SUMMARY_AIRPORT_ACTIVITY
- AIRPORT_COORDINATES
- airport_weather
- Ontime Reporting (Multiple file as per months)

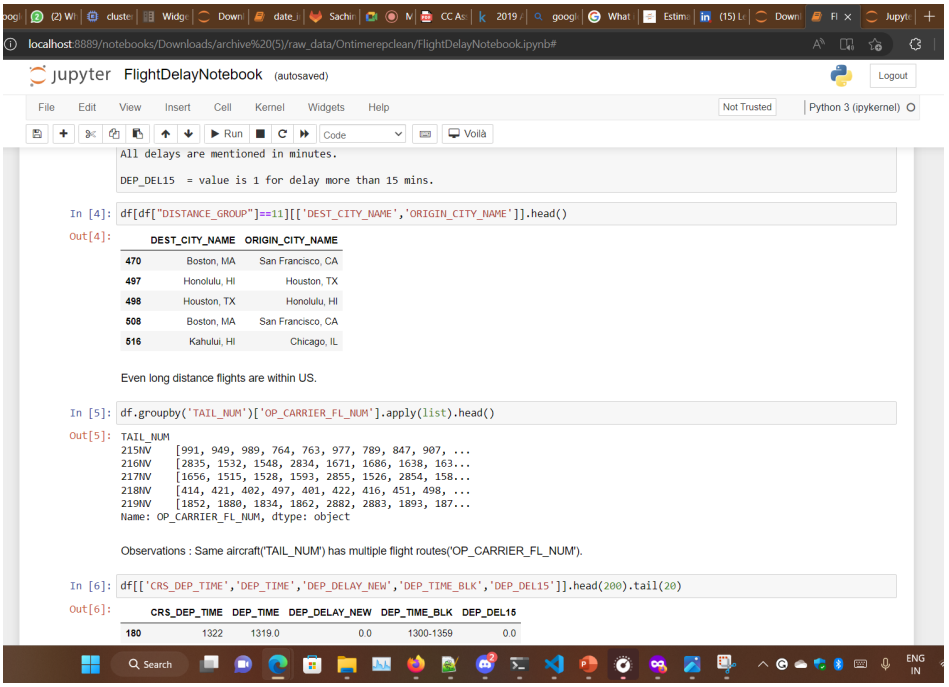
Cloud System Environment Setup:-

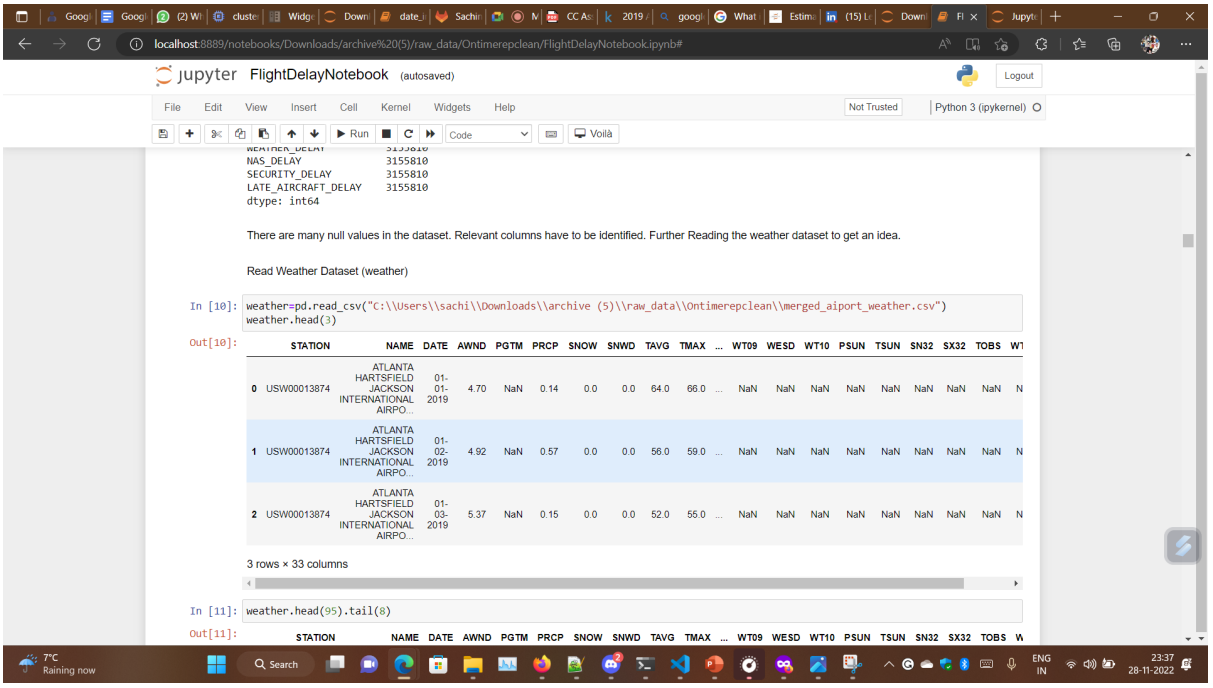
For the cloud infrastructure, we provisioned a dataproc cluster which is provided by Google Cloud.

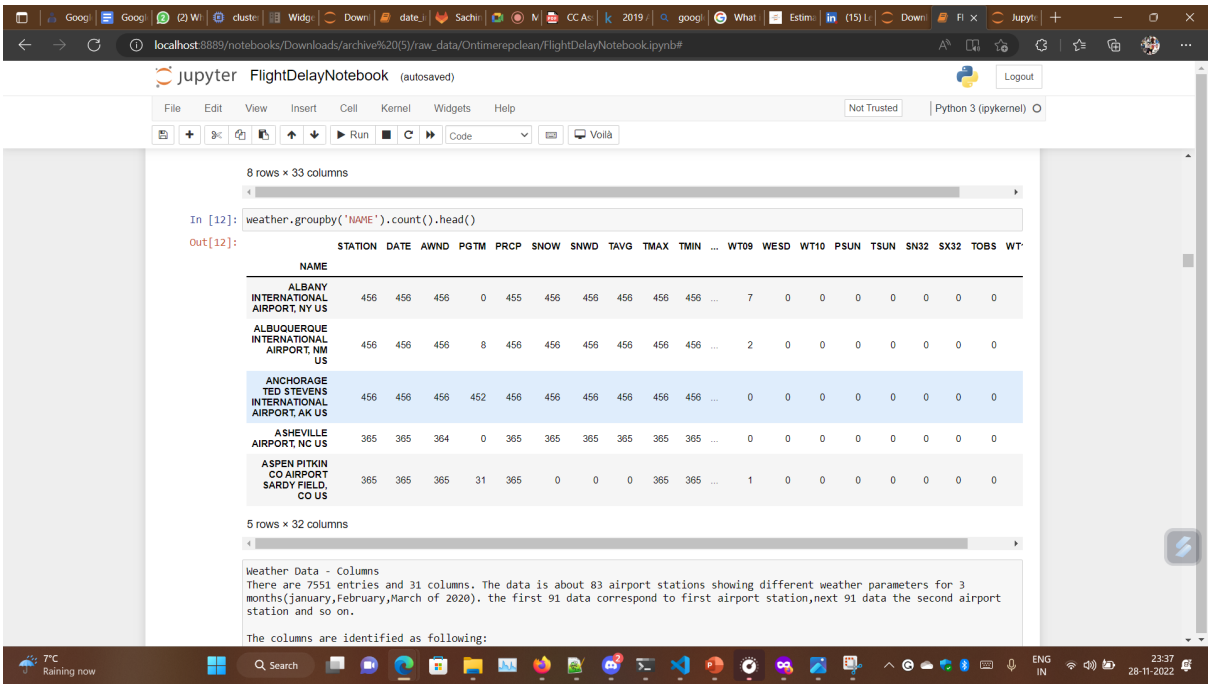
In the Assignment2 project on Google Cloud Platform, the dataproc cluster was provisioned with 1 Master Node and 2 Worker Nodes and Google cloud bucket linked with the cluster.

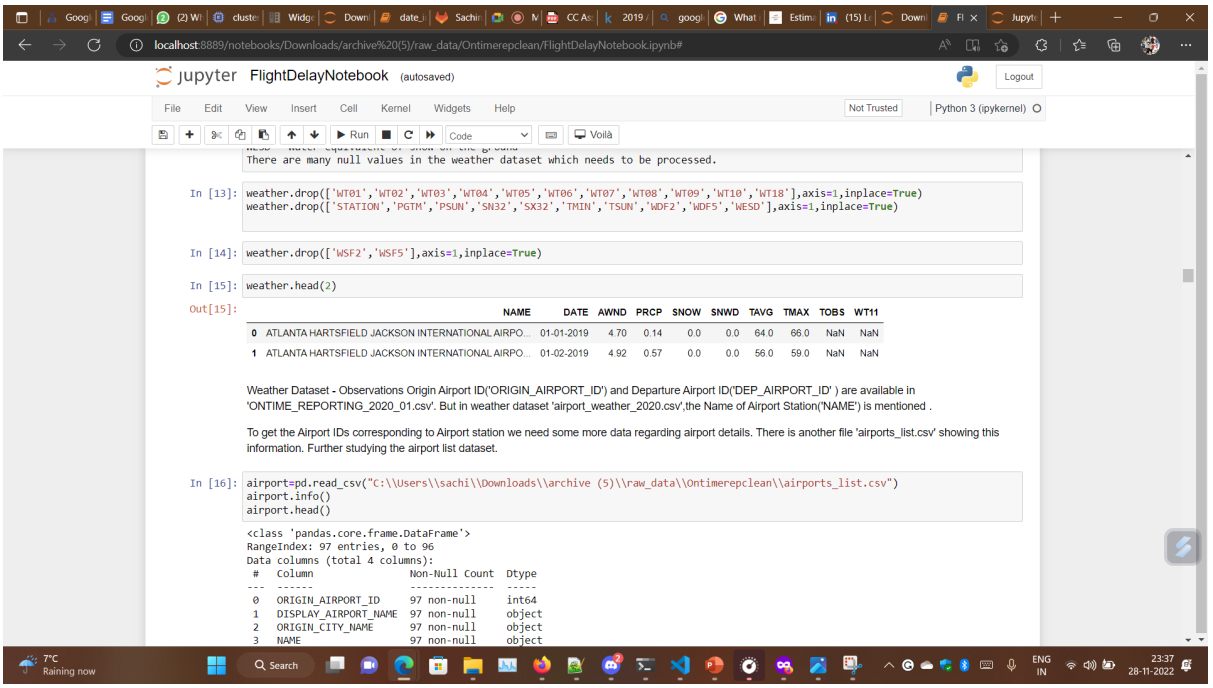
Data Preprocessing stage:-

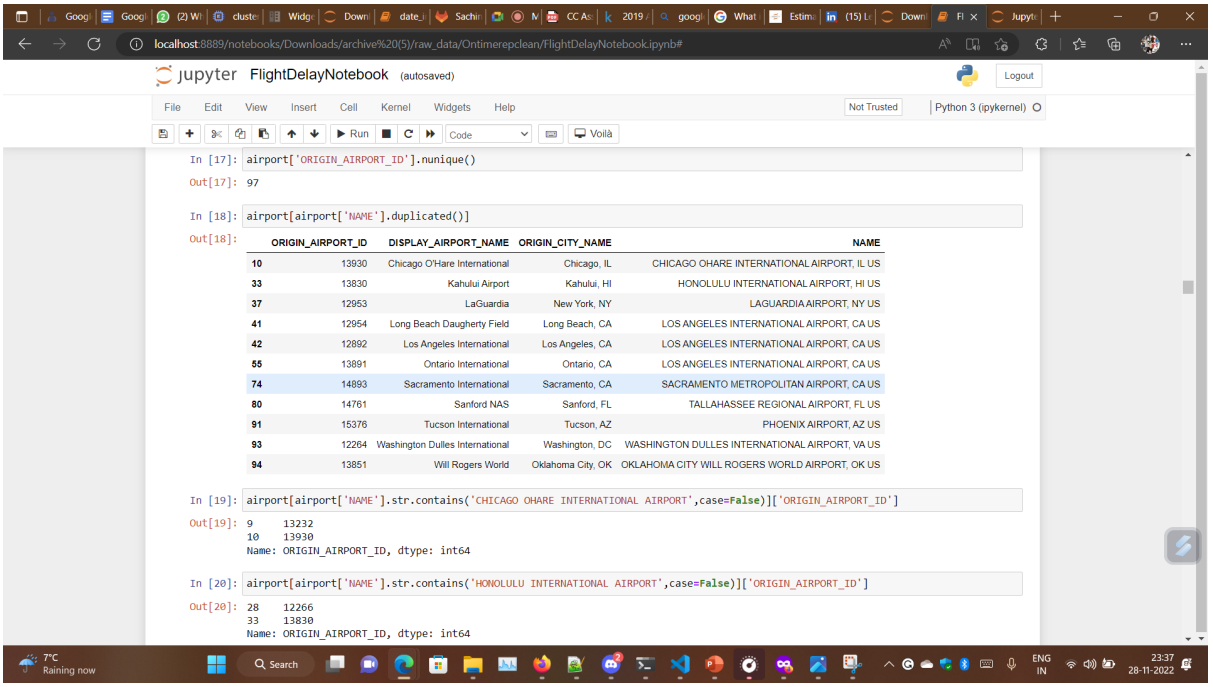
Each and every raw file in this dataset contains nulls, missing values, and these files need to be cleaned and pre-processed.


localhost:8889/notebooks/Downloads/archive%20(5)/raw_data/Ontimerepclean/FlightDelayNotebook.ipynb#
jupyter FlightDelayNotebook (autosaved)
Logout
File Edit View Insert Cell Kernel Widgets Help
Not Trusted
Python 3 (ipykernel)
Run
Code
Voilà
All delays are mentioned in minutes.
DEP_DEL15 = value is 1 for delay more than 15 mins.
In [4]: df[df["DISTANCE_GROUP"]==11][['DEST_CITY_NAME','ORIGIN_CITY_NAME']].head()
Out[4]:
DEST_CITY_NAME ORIGIN_CITY_NAME
470 Boston, MA San Francisco, CA
497 Honolulu, HI Houston, TX
498 Houston, TX Honolulu, HI
508 Boston, MA San Francisco, CA
516 Kahului, HI Chicago, IL
Even long distance flights are within US.
In [5]: df.groupby('TAIL_NUM')['OP_CARRIER_FL_NUM'].apply(list).head()
Out[5]:
TAIL_NUM
215NV [991, 949, 989, 764, 763, 977, 789, 847, 907, ...
216NV [2835, 1532, 1548, 2834, 1671, 1686, 1638, 163...
217NV [1656, 1515, 1528, 1593, 2855, 1526, 2854, 158...
218NV [414, 421, 402, 497, 401, 422, 416, 451, 498, ...
219NV [1852, 1880, 1834, 1862, 2882, 2883, 1893, 187...
Name: OP_CARRIER_FL_NUM, dtype: object
Observations : Same aircraft('TAIL_NUM') has multiple flight routes('OP_CARRIER_FL_NUM').
In [6]: df[['CRS_DEP_TIME','DEP_TIME','DEP_DELAY_NEW','DEP_TIME_BLK','DEP_DEL15']].head(200).tail(20)
Out[6]:
CRS_DEP_TIME DEP_TIME DEP_DELAY_NEW DEP_TIME_BLK DEP_DEL15
180 1322 1319.0 0.0 1300-1359 0.0
Search
ENG
IN


localhost:8889/notebooks/Downloads/archive%20(5)/raw_data/Ontimerepclean/FlightDelayNotebook.ipynb#
jupyter FlightDelayNotebook (autosaved)
Logout
File Edit View Insert Cell Kernel Widgets Help
Not Trusted
Python 3 (ipykernel)
Run
Code
Voilà
NAS_DELAY 3155810
SECURITY_DELAY 3155810
LATE_AIRCRAFT_DELAY 3155810
dtype: int64
There are many null values in the dataset. Relevant columns have to be identified. Further Reading the weather dataset to get an idea.
Read Weather Dataset (weather)
In [10]: weather=pd.read_csv("C:\\Users\\sachi\\Downloads\\archive (5)\\raw_data\\Ontimerepclean\\merged_aiport_weather.csv")
weather.head(3)
Out[10]:
STATION NAME DATE AWND PGTM PRCP SNOW SNWD TAVG TMAX ... WT09 WESD WT10 PSUN TSUN SN32 SX32 TOBS W
0 USW00013874 ATLANTA HARTSFIELD JACKSON INTERNATIONAL AIRPO... 01-01-2019 4.70 NaN 0.14 0.0 0.0 64.0 66.0 ... NaN NaN NaN NaN NaN NaN NaN NaN N
1 USW00013874 ATLANTA HARTSFIELD JACKSON INTERNATIONAL AIRPO... 01-02-2019 4.92 NaN 0.57 0.0 0.0 56.0 59.0 ... NaN NaN NaN NaN NaN NaN NaN NaN N
2 USW00013874 ATLANTA HARTSFIELD JACKSON INTERNATIONAL AIRPO... 01-03-2019 5.37 NaN 0.15 0.0 0.0 52.0 55.0 ... NaN NaN NaN NaN NaN NaN NaN NaN N
3 rows × 33 columns
In [11]: weather.head(95).tail(8)
Out[11]:
STATION NAME DATE AWND PGTM PRCP SNOW SNWD TAVG TMAX ... WT09 WESD WT10 PSUN TSUN SN32 SX32 TOBS W
7°C
Raining now
Search
ENG
IN
23:37
28-11-2022


localhost:8889/notebooks/Downloads/archive%20(5)/raw_data/OntimerepcleanFlightDelayNotebook.ipynb#
jupyter FlightDelayNotebook (autosaved)
Logout
File Edit View Insert Cell Kernel Widgets Help
Not Trusted
Python 3 (ipykernel)
Run
Code
Voilà
8 rows × 33 columns
In [12]:
weather.groupby('NAME').count().head()
Out[12]:
NAME
STATION DATE AWND PGTM PRCP SNOW SNWD TAVG TMAX TMIN ... WT09 WESD WT10 PSUN TSUN SN32 SX32 TOBS WT
ALBANY INTERNATIONAL AIRPORT, NY US 456 456 456 0 455 456 456 456 456 456 ... 7 0 0 0 0 0 0 0
ALBUQUERQUE INTERNATIONAL AIRPORT, NM US 456 456 456 8 456 456 456 456 456 456 ... 2 0 0 0 0 0 0 0
ANCHORAGE TED STEVENS INTERNATIONAL AIRPORT, AK US 456 456 456 452 456 456 456 456 456 456 ... 0 0 0 0 0 0 0 0
ASHEVILLE AIRPORT, NC US 365 365 364 0 365 365 365 365 365 365 ... 0 0 0 0 0 0 0 0
ASPEN PITKIN CO AIRPORT SARDY FIELD, CO US 365 365 365 31 365 0 0 0 365 365 ... 1 0 0 0 0 0 0 0
5 rows × 32 columns
Weather Data - Columns
There are 7551 entries and 31 columns. The data is about 83 airport stations showing different weather parameters for 3 months(january,February,March of 2020). the first 91 data correspond to first airport station,next 91 data the second airport station and so on.
The columns are identified as following:
7°C Raining now
Search
ENG IN
23:37 28-11-2022


localhost:8889/notebooks/Downloads/archive%20(5)/raw_data/OntimerepcleanFlightDelayNotebook.ipynb#
jupyter FlightDelayNotebook (autosaved)
Logout
File Edit View Insert Cell Kernel Widgets Help
Not Trusted
Python 3 (ipykernel)
Run
Code
Voilà
There are many null values in the weather dataset which needs to be processed.
In [13]:
weather.drop(['WT01','WT02','WT03','WT04','WT05','WT06','WT07','WT08','WT09','WT10','WT18'],axis=1,inplace=True)
weather.drop(['STATION','PGTM','PSUN','SN32','SX32','TMIN','TSUN','WDF2','WDF5','WESD'],axis=1,inplace=True)
In [14]:
weather.drop(['WSF2','WSF5'],axis=1,inplace=True)
In [15]:
weather.head(2)
Out[15]:
NAME DATE AWND PRCP SNOW SNWD TAVG TMAX TOBS WT11
0 ATLANTA HARTSFIELD JACKSON INTERNATIONAL AIRPO... 01-01-2019 4.70 0.14 0.0 0.0 64.0 66.0 NaN NaN
1 ATLANTA HARTSFIELD JACKSON INTERNATIONAL AIRPO... 01-02-2019 4.92 0.57 0.0 0.0 56.0 59.0 NaN NaN
Weather Dataset - Observations Origin Airport ID('ORIGIN_AIRPORT_ID') and Departure Airport ID('DEP_AIRPORT_ID') are available in 'ONTIME_REPORTING_2020_01.csv'. But in weather dataset 'airport_weather_2020.csv',the Name of Airport Station('NAME') is mentioned .
To get the Airport IDs corresponding to Airport station we need some more data regarding airport details. There is another file 'airports_list.csv' showing this information. Further studying the airport list dataset.
In [16]:
airport=pd.read_csv("C:\\Users\\sachi\\Downloads\\archive (5)\\raw_data\\Ontimerepclean\\airports_list.csv")
airport.info()
airport.head()
<class 'pandas.core.frame.DataFrame'>
RangeIndex: 97 entries, 0 to 96
Data columns (total 4 columns):
Column Non-Null Count Dtype
--- ------ -------------- -----
0 ORIGIN_AIRPORT_ID 97 non-null int64
1 DISPLAY_AIRPORT_NAME 97 non-null object
2 ORIGIN_CITY_NAME 97 non-null object
3 NAME 97 non-null object
7°C Raining now
Search
ENG IN
23:37 28-11-2022



jupyter FlightDelayNotebook (autosaved)

File Edit View Insert Cell Kernel Widgets Help

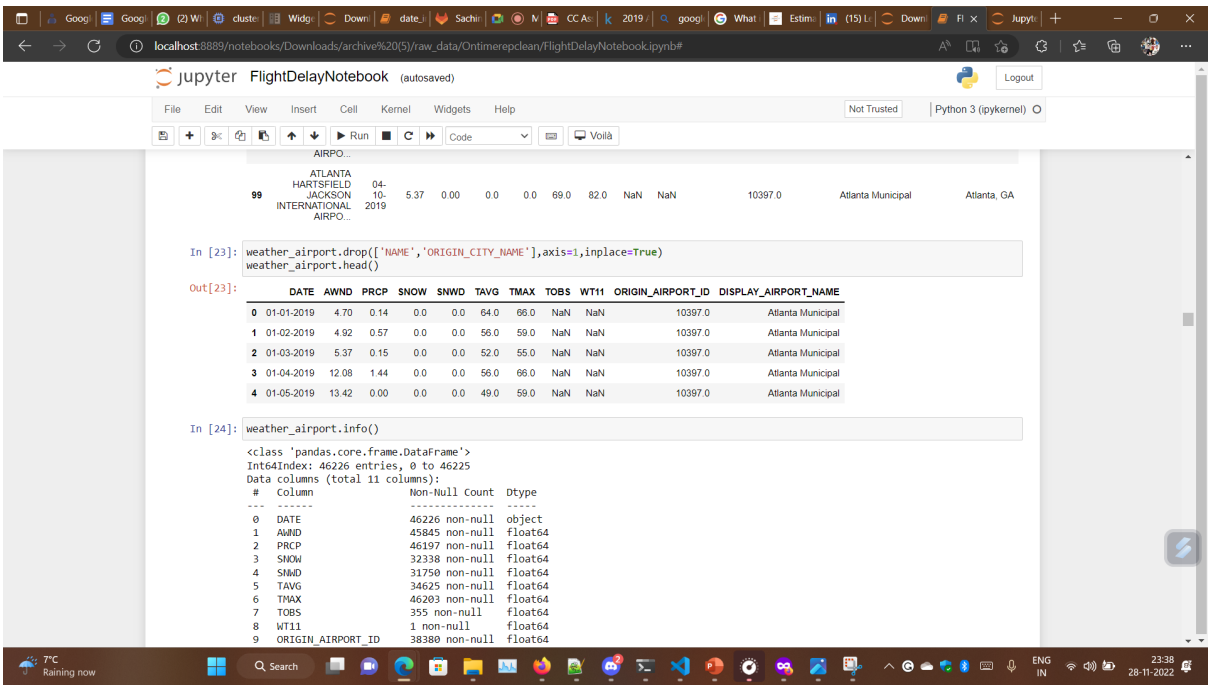
Not Trusted | Python 3 (ipykernel)

```
In [17]: airport['ORIGIN_AIRPORT_ID'].nunique()
Out[17]: 97

In [18]: airport[airport['NAME'].duplicated()]
```

Out[18]:

	ORIGIN_AIRPORT_ID	DISPLAY_AIRPORT_NAME	ORIGIN_CITY_NAME	NAME
10	13930	Chicago O'Hare International	Chicago, IL	CHICAGO OHARE INTERNATIONAL AIRPORT, IL US
33	13830	Kahului Airport	Kahului, HI	HONOLULU INTERNATIONAL AIRPORT, HI US
37	12953	LaGuardia	New York, NY	LAGUARDIA AIRPORT, NY US
41	12954	Long Beach Daugherty Field	Long Beach, CA	LOS ANGELES INTERNATIONAL AIRPORT, CA US
42	12892	Los Angeles International	Los Angeles, CA	LOS ANGELES INTERNATIONAL AIRPORT, CA US
55	13891	Ontario International	Ontario, CA	LOS ANGELES INTERNATIONAL AIRPORT, CA US
74	14893	Sacramento International	Sacramento, CA	SACRAMENTO METROPOLITAN AIRPORT, CA US
80	14761	Sanford NAS	Sanford, FL	TALLAHASSEE REGIONAL AIRPORT, FL US
91	15376	Tucson International	Tucson, AZ	PHOENIX AIRPORT, AZ US
93	12264	Washington Dulles International	Washington, DC	WASHINGTON DULLES INTERNATIONAL AIRPORT, VA US
94	13851	Will Rogers World	Oklahoma City, OK	OKLAHOMA CITY WILL ROGERS WORLD AIRPORT, OK US

```
In [19]: airport[airport['NAME'].str.contains('CHICAGO OHARE INTERNATIONAL AIRPORT',case=False)]['ORIGIN_AIRPORT_ID']
Out[19]: 9     13232
         10    13930
         Name: ORIGIN_AIRPORT_ID, dtype: int64

In [20]: airport[airport['NAME'].str.contains('HONOLULU INTERNATIONAL AIRPORT',case=False)]['ORIGIN_AIRPORT_ID']
Out[20]: 28    12266
         33    13830
         Name: ORIGIN_AIRPORT_ID, dtype: int64
```



jupyter FlightDelayNotebook (autosaved)

File Edit View Insert Cell Kernel Widgets Help

Not Trusted | Python 3 (ipykernel)

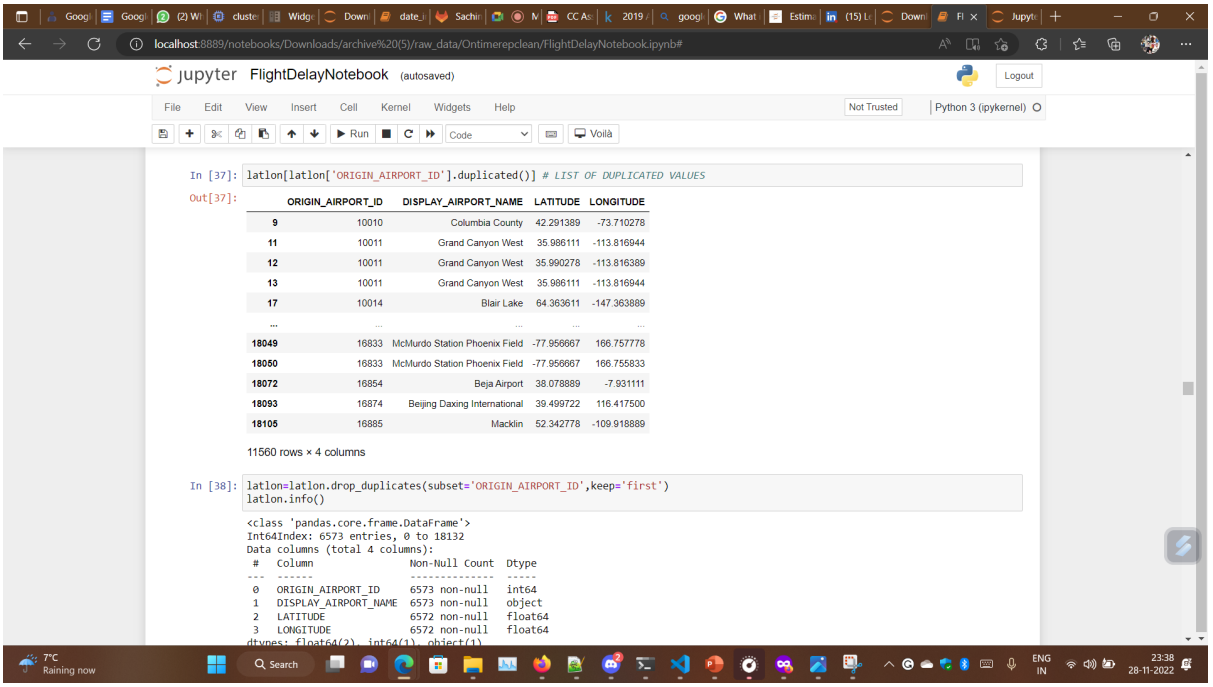
99	ATLANTA HARTSFIELD JACKSON INTERNATIONAL AIRPO...	04-10-2019	5.37	0.00	0.0	0.0	69.0	82.0	NaN	NaN	10397.0	Atlanta Municipal	Atlanta, GA

```
In [23]: weather_airport.drop(['NAME','ORIGIN_CITY_NAME'],axis=1,inplace=True)
         weather_airport.head()
```

Out[23]:

	DATE	AWND	PRCP	SNOW	SNWD	TAVG	TMAX	TOBS	WT11	ORIGIN_AIRPORT_ID	DISPLAY_AIRPORT_NAME
0	01-01-2019	4.70	0.14	0.0	0.0	64.0	66.0	NaN	NaN	10397.0	Atlanta Municipal
1	01-02-2019	4.92	0.57	0.0	0.0	56.0	59.0	NaN	NaN	10397.0	Atlanta Municipal
2	01-03-2019	5.37	0.15	0.0	0.0	52.0	55.0	NaN	NaN	10397.0	Atlanta Municipal
3	01-04-2019	12.08	1.44	0.0	0.0	56.0	66.0	NaN	NaN	10397.0	Atlanta Municipal
4	01-05-2019	13.42	0.00	0.0	0.0	49.0	59.0	NaN	NaN	10397.0	Atlanta Municipal

```
In [24]: weather_airport.info()

<class 'pandas.core.frame.DataFrame'>
Int64Index: 46226 entries, 0 to 46225
Data columns (total 11 columns):
 #   Column                Non-Null Count  Dtype
---  ------                --------------  -----
 0   DATE                  46226 non-null  object
 1   AWND                  45845 non-null  float64
 2   PRCP                  46197 non-null  float64
 3   SNOW                  32338 non-null  float64
 4   SNWD                  31750 non-null  float64
 5   TAVG                  34625 non-null  float64
 6   TMAX                  46203 non-null  float64
 7   TOBS                  355 non-null    float64
 8   WT11                  1 non-null      float64
 9   ORIGIN_AIRPORT_ID     38380 non-null  float64
```



jupyter FlightDelayNotebook (autosaved)

File Edit View Insert Cell Kernel Widgets Help

Not Trusted | Python 3 (ipykernel)

In [37]:
```
latlon[latlon['ORIGIN_AIRPORT_ID'].duplicated()] # LIST OF DUPLICATED VALUES
```

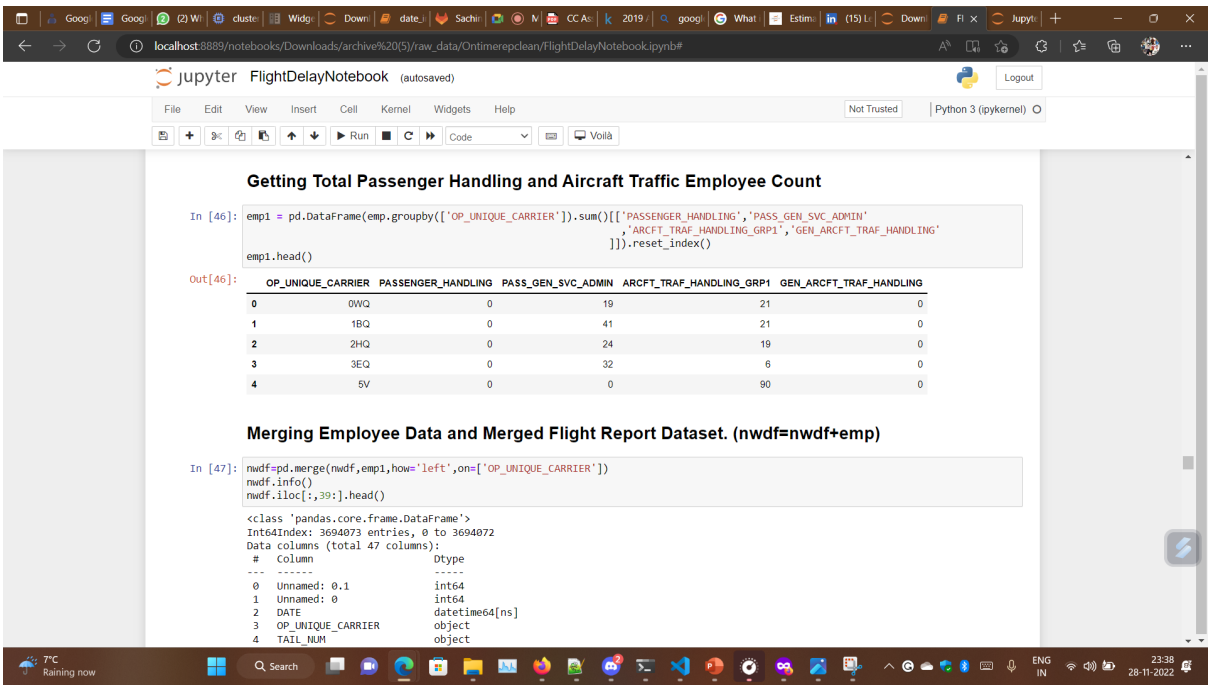
Out[37]:

	ORIGIN_AIRPORT_ID	DISPLAY_AIRPORT_NAME	LATITUDE	LONGITUDE
9	10010	Columbia County	42.291389	-73.710278
11	10011	Grand Canyon West	35.986111	-113.816944
12	10011	Grand Canyon West	35.990278	-113.816389
13	10011	Grand Canyon West	35.986111	-113.816944
17	10014	Blair Lake	64.363611	-147.363889
...	...	...	...	...
18049	16833	McMurdo Station Phoenix Field	-77.956667	166.757778
18050	16833	McMurdo Station Phoenix Field	-77.956667	166.755833
18072	16854	Beja Airport	38.078889	-7.931111
18093	16874	Beijing Daxing International	39.499722	116.417500
18105	16885	Macklin	52.342778	-109.918889

11560 rows × 4 columns

In [38]:
```
latlon=latlon.drop_duplicates(subset='ORIGIN_AIRPORT_ID',keep='first')
latlon.info()
```

```
<class 'pandas.core.frame.DataFrame'>
Int64Index: 6573 entries, 0 to 18132
Data columns (total 4 columns):
 #   Column               Non-Null Count  Dtype  
---  ------               --------------  -----  
 0   ORIGIN_AIRPORT_ID    6573 non-null   int64  
 1   DISPLAY_AIRPORT_NAME  6573 non-null   object 
 2   LATITUDE             6572 non-null   float64
 3   LONGITUDE            6572 non-null   float64
dtypes: float64(2), int64(1), object(1)
```



Getting Total Passenger Handling and Aircraft Traffic Employee Count

In [46]:
```
emp1 = pd.DataFrame(emp.groupby(['OP_UNIQUE_CARRIER']).sum()[['PASSENGER_HANDLING','PASS_GEN_SVC_ADMIN'
                                                              ,'ARCFT_TRAF_HANDLING_GRP1','GEN_ARCFT_TRAF_HANDLING'
                                                             ]]).reset_index()
emp1.head()
```

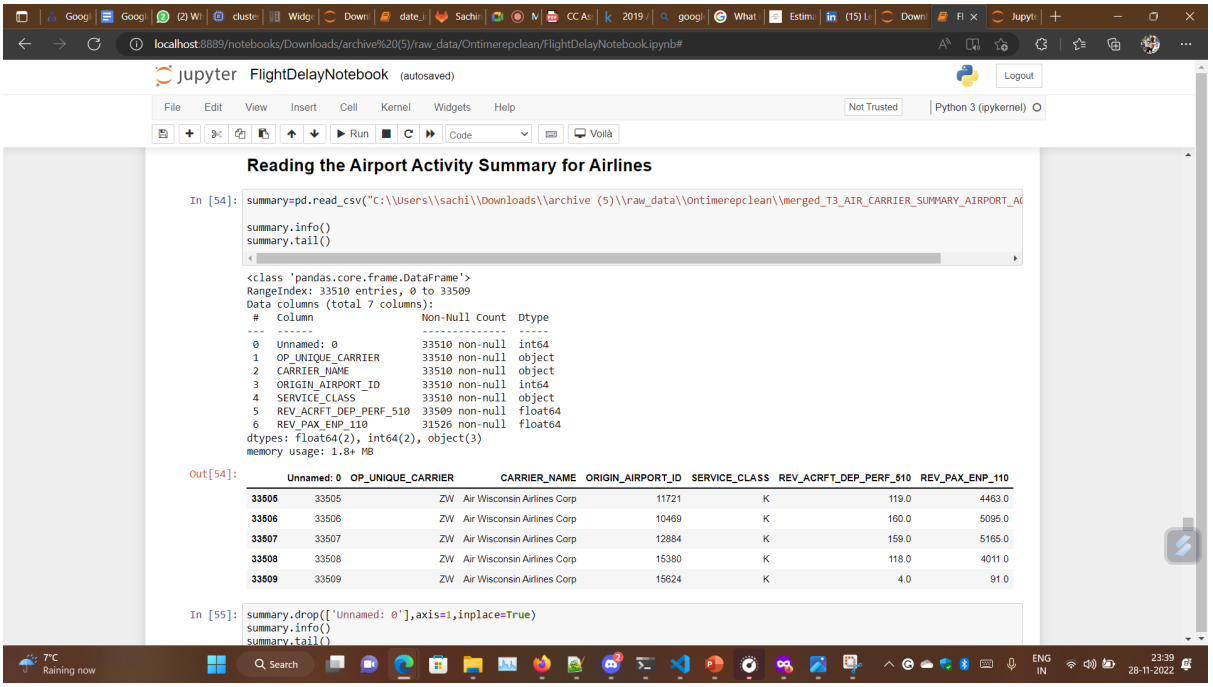
Out[46]:

	OP_UNIQUE_CARRIER	PASSENGER_HANDLING	PASS_GEN_SVC_ADMIN	ARCFT_TRAF_HANDLING_GRP1	GEN_ARCFT_TRAF_HANDLING
0	0WQ	0	19	21	0
1	1BQ	0	41	21	0
2	2HQ	0	24	19	0
3	3EQ	0	32	6	0
4	5V	0	0	90	0

Merging Employee Data and Merged Flight Report Dataset. (nwdf=nwdf+emp)

In [47]:
```
nwdf=pd.merge(nwdf,emp1,how='left',on=['OP_UNIQUE_CARRIER'])
nwdf.info()
nwdf.iloc[:,39:].head()
```

```
<class 'pandas.core.frame.DataFrame'>
Int64Index: 3694073 entries, 0 to 3694072
Data columns (total 47 columns):
 #   Column                Dtype         
---  ------                -----         
 0   Unnamed: 0.1          int64         
 1   Unnamed: 0            int64         
 2   DATE                  datetime64[ns]
 3   OP_UNIQUE_CARRIER     object        
 4   TAIL_NUM              object        
```



Reading the Airport Activity Summary for Airlines

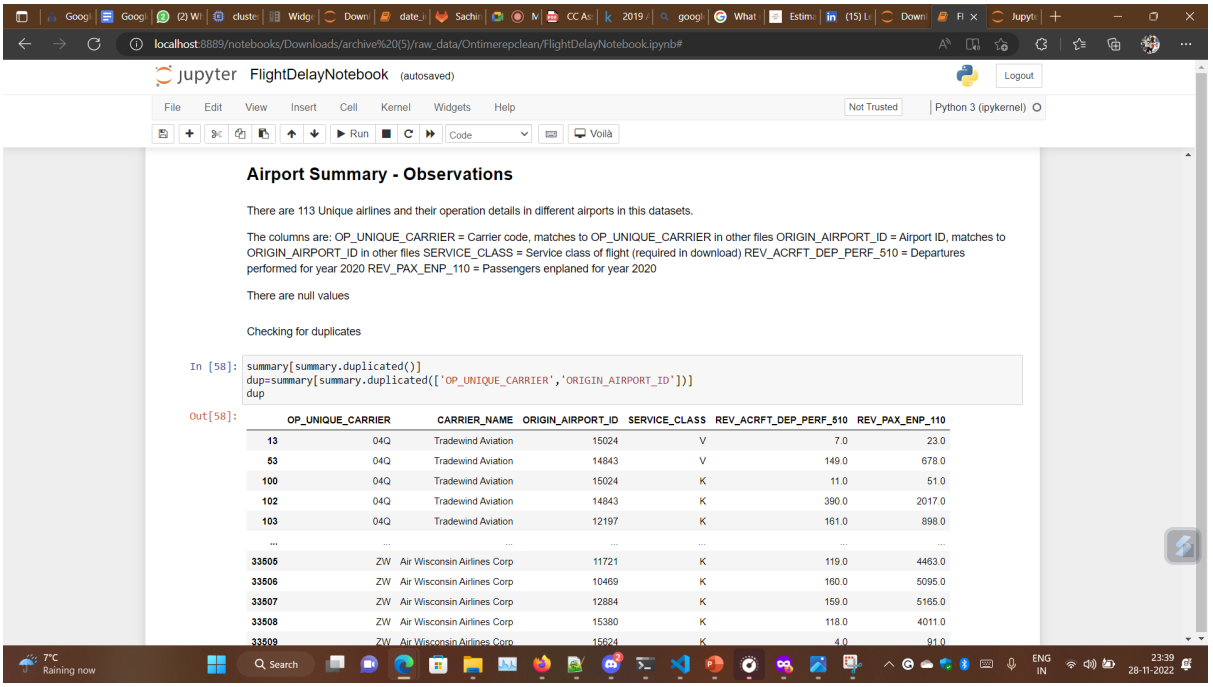
In [54]:
```
summary=pd.read_csv("C:\\Users\\sachi\\Downloads\\archive (5)\\raw_data\\Ontimerepclean\\merged_T3_AIR_CARRIER_SUMMARY_AIRPORT_A(

summary.info()
summary.tail()
```

```
<class 'pandas.core.frame.DataFrame'>
RangeIndex: 33510 entries, 0 to 33509
Data columns (total 7 columns):
 #   Column                 Non-Null Count  Dtype
---  ------                 --------------  -----
 0   Unnamed: 0             33510 non-null  int64
 1   OP_UNIQUE_CARRIER      33510 non-null  object
 2   CARRIER_NAME           33510 non-null  object
 3   ORIGIN_AIRPORT_ID      33510 non-null  int64
 4   SERVICE_CLASS          33510 non-null  object
 5   REV_ACRFT_DEP_PERF_510 33509 non-null  float64
 6   REV_PAX_ENP_110        31526 non-null  float64
dtypes: float64(2), int64(2), object(3)
memory usage: 1.8+ MB
```

Out[54]:

	Unnamed: 0	OP_UNIQUE_CARRIER	CARRIER_NAME	ORIGIN_AIRPORT_ID	SERVICE_CLASS	REV_ACRFT_DEP_PERF_510	REV_PAX_ENP_110
33505	33505	ZW	Air Wisconsin Airlines Corp	11721	K	119.0	4463.0
33506	33506	ZW	Air Wisconsin Airlines Corp	10469	K	160.0	5095.0
33507	33507	ZW	Air Wisconsin Airlines Corp	12884	K	159.0	5165.0
33508	33508	ZW	Air Wisconsin Airlines Corp	15380	K	118.0	4011.0
33509	33509	ZW	Air Wisconsin Airlines Corp	15624	K	4.0	91.0

In [55]:
```
summary.drop(['Unnamed: 0'],axis=1,inplace=True)
summary.info()
summary.tail()
```



Airport Summary - Observations

There are 113 Unique airlines and their operation details in different airports in this datasets.

The columns are: OP_UNIQUE_CARRIER = Carrier code, matches to OP_UNIQUE_CARRIER in other files ORIGIN_AIRPORT_ID = Airport ID, matches to ORIGIN_AIRPORT_ID in other files SERVICE_CLASS = Service class of flight (required in download) REV_ACRFT_DEP_PERF_510 = Departures performed for year 2020 REV_PAX_ENP_110 = Passengers enplaned for year 2020

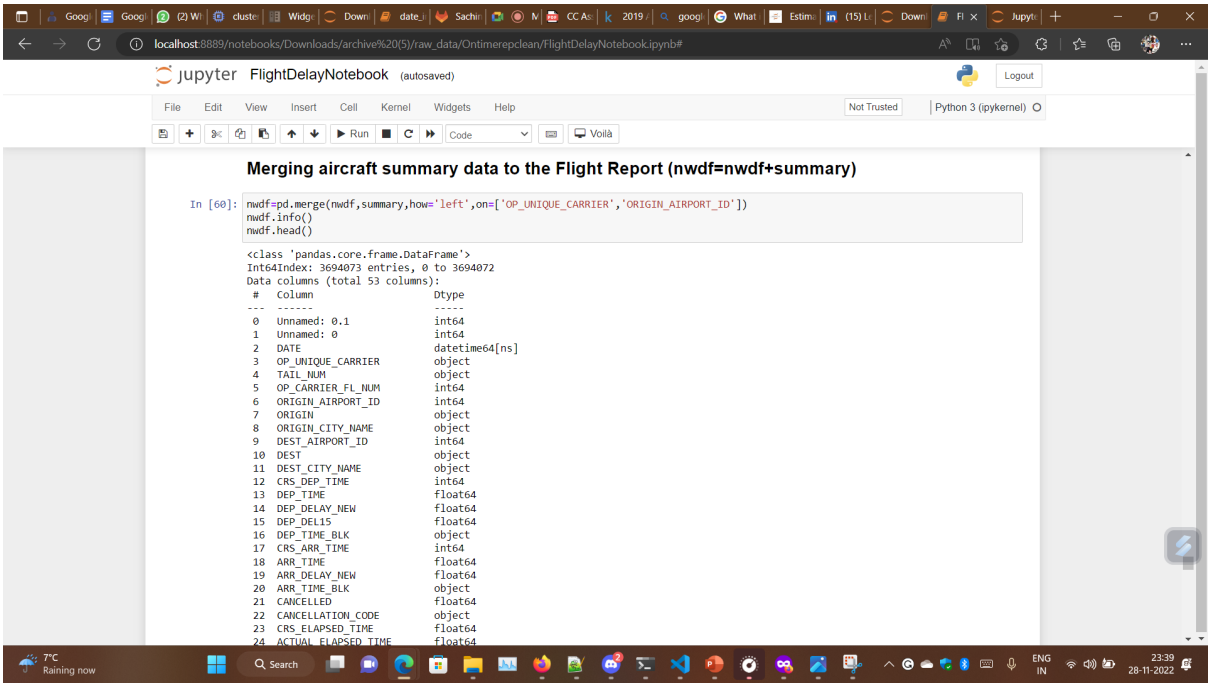
There are null values

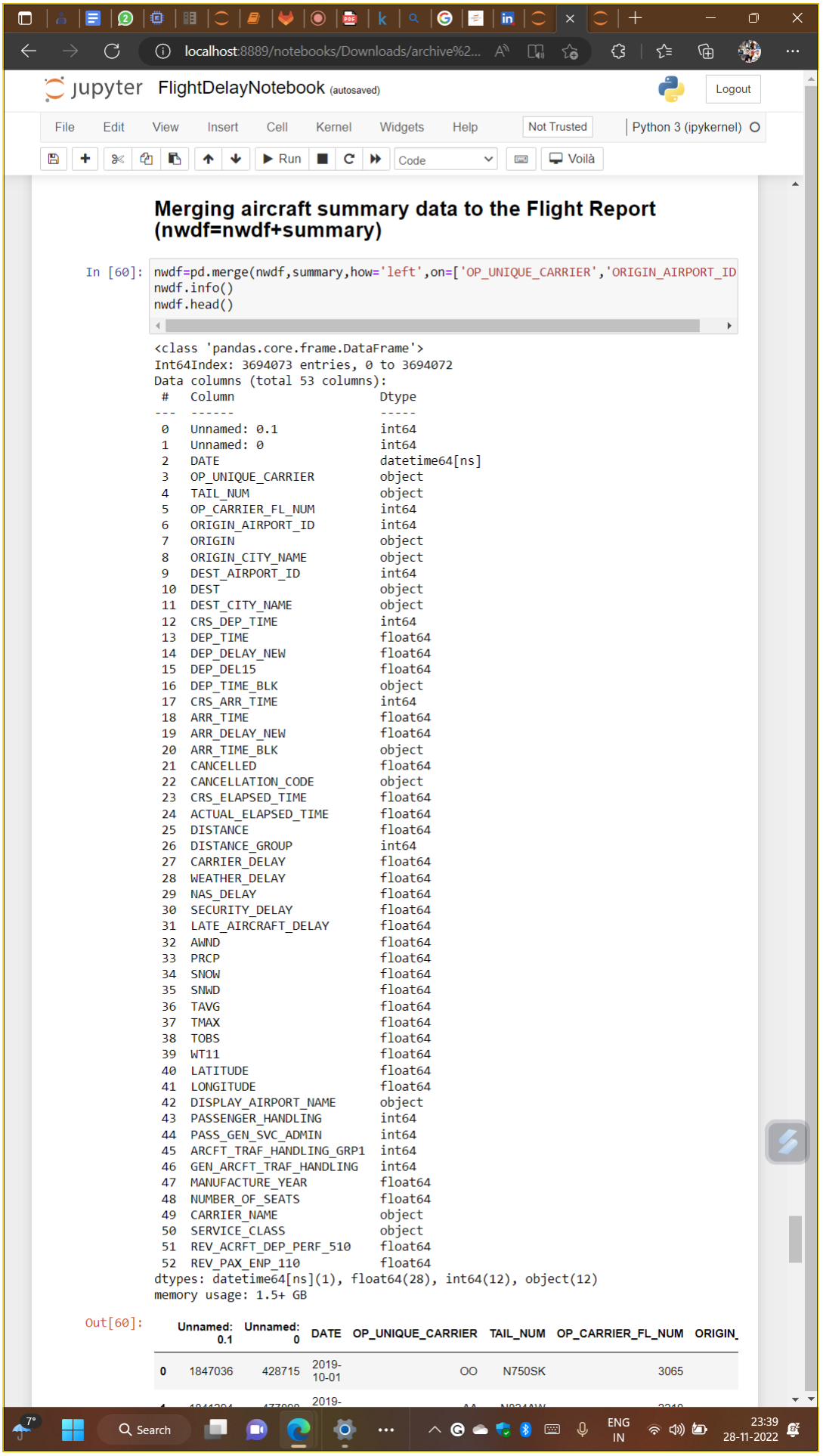
Checking for duplicates

In [58]:
```
summary[summary.duplicated()]
dup=summary[summary.duplicated(['OP_UNIQUE_CARRIER','ORIGIN_AIRPORT_ID'])]
dup
```

Out[58]:

	OP_UNIQUE_CARRIER	CARRIER_NAME	ORIGIN_AIRPORT_ID	SERVICE_CLASS	REV_ACRFT_DEP_PERF_510	REV_PAX_ENP_110
13	04Q	Tradewind Aviation	15024	V	7.0	23.0
53	04Q	Tradewind Aviation	14843	V	149.0	678.0
100	04Q	Tradewind Aviation	15024	K	11.0	51.0
102	04Q	Tradewind Aviation	14843	K	390.0	2017.0
103	04Q	Tradewind Aviation	12197	K	161.0	898.0
...	...	...	...	...	...	...
33505	ZW	Air Wisconsin Airlines Corp	11721	K	119.0	4463.0
33506	ZW	Air Wisconsin Airlines Corp	10469	K	160.0	5095.0
33507	ZW	Air Wisconsin Airlines Corp	12884	K	159.0	5165.0
33508	ZW	Air Wisconsin Airlines Corp	15380	K	118.0	4011.0
33509	ZW	Air Wisconsin Airlines Corp	15624	K	4.0	91.0


jupyter FlightDelayNotebook (autosaved)
Logout
File Edit View Insert Cell Kernel Widgets Help
Not Trusted
Python 3 (ipykernel)
Run
Code
Voilà
Merging aircraft summary data to the Flight Report (nwdf=nwdf+summary)
In [60]:
nwdf=pd.merge(nwdf,summary,how='left',on=['OP_UNIQUE_CARRIER','ORIGIN_AIRPORT_ID'])
nwdf.info()
nwdf.head()
<class 'pandas.core.frame.DataFrame'>
Int64Index: 3694073 entries, 0 to 3694072
Data columns (total 53 columns):
Column Dtype
--- ------ -----
0 Unnamed: 0.1 int64
1 Unnamed: 0 int64
2 DATE datetime64[ns]
3 OP_UNIQUE_CARRIER object
4 TAIL_NUM object
5 OP_CARRIER_FL_NUM int64
6 ORIGIN_AIRPORT_ID int64
7 ORIGIN object
8 ORIGIN_CITY_NAME object
9 DEST_AIRPORT_ID int64
10 DEST object
11 DEST_CITY_NAME object
12 CRS_DEP_TIME int64
13 DEP_TIME float64
14 DEP_DELAY_NEW float64
15 DEP_DEL15 float64
16 DEP_TIME_BLK object
17 CRS_ARR_TIME int64
18 ARR_TIME float64
19 ARR_DELAY_NEW float64
20 ARR_TIME_BLK object
21 CANCELLED float64
22 CANCELLATION_CODE object
23 CRS_ELAPSED_TIME float64
24 ACTUAL_ELAPSED_TIME float64
7°C
Raining now
Search
ENG
IN
23:39
28-11-2022



Merging aircraft summary data to the Flight Report (nwdf=nwdf+summary)

In [60]:
```
nwdf=pd.merge(nwdf,summary,how='left',on=['OP_UNIQUE_CARRIER','ORIGIN_AIRPORT_ID
nwdf.info()
nwdf.head()
```

```
<class 'pandas.core.frame.DataFrame'>
Int64Index: 3694073 entries, 0 to 3694072
Data columns (total 53 columns):
 #   Column                    Dtype
---  ------                    -----
 0   Unnamed: 0.1              int64
 1   Unnamed: 0                int64
 2   DATE                      datetime64[ns]
 3   OP_UNIQUE_CARRIER         object
 4   TAIL_NUM                  object
 5   OP_CARRIER_FL_NUM         int64
 6   ORIGIN_AIRPORT_ID         int64
 7   ORIGIN                    object
 8   ORIGIN_CITY_NAME          object
 9   DEST_AIRPORT_ID           int64
 10  DEST                      object
 11  DEST_CITY_NAME            object
 12  CRS_DEP_TIME              int64
 13  DEP_TIME                  float64
 14  DEP_DELAY_NEW             float64
 15  DEP_DEL15                 float64
 16  DEP_TIME_BLK              object
 17  CRS_ARR_TIME              int64
 18  ARR_TIME                  float64
 19  ARR_DELAY_NEW             float64
 20  ARR_TIME_BLK              object
 21  CANCELLED                 float64
 22  CANCELLATION_CODE         object
 23  CRS_ELAPSED_TIME          float64
 24  ACTUAL_ELAPSED_TIME       float64
 25  DISTANCE                  float64
 26  DISTANCE_GROUP            int64
 27  CARRIER_DELAY             float64
 28  WEATHER_DELAY             float64
 29  NAS_DELAY                 float64
 30  SECURITY_DELAY            float64
 31  LATE_AIRCRAFT_DELAY       float64
 32  AWND                      float64
 33  PRCP                      float64
 34  SNOW                      float64
 35  SNWD                      float64
 36  TAVG                      float64
 37  TMAX                      float64
 38  TOBS                      float64
 39  WT11                      float64
 40  LATITUDE                  float64
 41  LONGITUDE                 float64
 42  DISPLAY_AIRPORT_NAME      object
 43  PASSENGER_HANDLING        int64
 44  PASS_GEN_SVC_ADMIN        int64
 45  ARCFT_TRAF_HANDLING_GRP1  int64
 46  GEN_ARCFT_TRAF_HANDLING   int64
 47  MANUFACTURE_YEAR          float64
 48  NUMBER_OF_SEATS           float64
 49  CARRIER_NAME              object
 50  SERVICE_CLASS             object
 51  REV_ACRFT_DEP_PERF_510    float64
 52  REV_PAX_ENP_110           float64
dtypes: datetime64[ns](1), float64(28), int64(12), object(12)
memory usage: 1.5+ GB
```

Out[60]:

	Unnamed: 0.1	Unnamed: 0	DATE	OP_UNIQUE_CARRIER	TAIL_NUM	OP_CARRIER_FL_NUM	ORIGIN_
0	1847036	428715	2019-10-01	OO	N750SK	3065	
1	1941284	477209	2019-	AA	N934AW	2210	

One of the best data cleaning tools is python which is fast and efficient for cleaning large datasets. This is faster than hive for basic cleaning and preprocessing the raw files.

Next step is to store these tables/files in hive for further analysis on the dataset.

```
Hive Session ID = 06cbdc37-8c0f-4aec-8168-29faf151a74b
hive> show databases;
OK
default
f_db
Time taken: 0.71 seconds, Fetched: 2 row(s)
hive> use f_db;
OK
Time taken: 0.043 seconds
hive> show tables;
OK
aircraft_inventory
airport_coordinates
airport_weather
airports_list
carrier_summary_airport_activity
flight_analysis2
flight_tbl
ontimerep
p10_employees
top10airports
Time taken: 0.041 seconds, Fetched: 10 row(s)
hive>
```

8. Development of the Application

For the development of the proposed Data analytics application, there were two options available one with the nice looking user interface with interactive controls and the other one is to follow a command line approach. As the data analysis was the main objective so the command line approach was the option to go for.

This application is developed using Jupyter Notebook as the python libraries are quite helpful for data analysis. This Jupyter Notebook is hosted on the dataproc's Master Node and the same is accessed via connection gateway.

The Airline and weather conditions data available on hive fetched using PySpark engine in the jupyter notebook using PySpark python module.

Once the PySpark data frame is created then using toPandas() method we manage to convert the PySpark dataframe into Pandas dataframe to utilise the full potential of the pandas library.

To make the Webapp with visuals we also make use of Ipywidgets which adds a layer of interactiveness on the visuals using javascripts/html and CSS.

As jupyter notebook has code/text cell and result box, so to hide the code from the user or the person who interacts we added another python module ''Voila'' which allows to convert a Jupyter Notebook into an interactive dashboard.

The final Webpage showcases the analysis over the Airlines Dataset giving some control to the user to interact with it.

9. Challenges and lessons learned

Challenges Faced:

There were many challenges faced by us while developing this Data Analytics Application:

1. Java Gateway log4J error : One of the issues we faced while running multiple sessions of jupyter notebook on the same dataproc cluster. We investigated the root cause of this and found that there was a bug which did not let the user open multiple sessions of the jupyter notebook.
2. Jupyter Notebook Buffer memory and iopub_data_rate_limit : The jupyter notebook default configurations don't allow it to fetch large data as it uses buffer memory for temporary storage area for data processing.

Lessons Learned:

While creating the Flight Report Dataset we observed that there are multiple factors affecting the aircraft arrival time like weather delay, security delay, carrier delay and late aircraft delay affects the departure delays but there are other factors like National Aviation System Delays that affects the aircraft arrival time. However there is an exception that the departure time delay should not be more than 15 minutes and it is independent of arrival time delays.

Depending on the region and city, the weather in the United States (USA) in January might be excellent or very poor. To anticipate a flight delay based on the cities and associated weather conditions, weather data must be gathered.

There were problems when combining the Flight Report dataset because the weather dataset only contains 97 unique ORIGIN AIRPORT IDs whereas airport ids like origin and destination have 350 unique values. We discovered that the airport names may be substituted by latitude and longitude to make it easier while implementing anything further to acquire the result, which was another problem we encountered while merging. Both the dates and airport id had to be taken into account.

There was additional information provided by the Employees in charge of passenger Handling and Aircraft Traffic Control for each Airline that can give an idea of how busy and vast an airline operation is and can be included for research. Many fields were found irrelevant in the cleaning process and were disregarded ,however the AirTraffic and Passenger Handling fields can be considered as it might have important information related to aircraft and its emergency protocol related guidelines.

There are no null values found in the cleaning process but there was data for the same airlines for different entities given as separate entries. Duplicate values were removed to get more accurate and refined data while predicting on the same.

10. Responsibility section

Tasks

1. Sachin Patel

Data Cleaning and Preprocessing : The Flight delay dataset contains “ontimereporting” files which are the on time reporting done by the aircraft carrier and there are total 15 files for each month ranging from Jan-2019 to Mar-2020. My task was to clean (nulls,special characters, NA values) these files using PigLatin and merge these files and perform data preparation like generating date column from the day and month column from the ontimerep files and merging them to a single file for further analysis which is to be done in hive.

Once the f_db database is created in hive and all the required dimension tables and aggregate tables loaded on it, then using Hive SQL queries I have performed the analysis on the flight delay data :-

- Finding the top 10 carriers with average delay minutes
- Finding the top 10 aircraft carriers with shortest travel distance/shortest flight time , longest distance / longest flight time

Further for making interesting Data visualisation we have used Jupyter Notebook for making visuals on the following two analysis:

- Finding the Correlation between the flight delay time with the factors affecting the delay such as weather delay, navigation delay, late aircraft delay, carrier delay
- Finding the top 50 Carriers with respective carrier delay percentage

The Jupyter Notebook was failing to load the large data so we changed the Jupyter Notebook’s buffer memory to a certain limit that it can load the data in its buffer and the kernel should stay alive and not die due to large data.

2. Rohit Shinde

Finding the dataset was the first important thing which I had to do so I found the flight delay dataset of year 2019 from kaggle. This dataset of volume 4.67 GB includes files like full_flightdelay, test and train data but decided to go with the raw data files of volume 1.67 GB which includes files like airport_coordinates, airportweather2019, carrier_decode, ontime_reporting, airport_list, etc. These raw files were semi structured. The reason to go with raw files is that it gives us a variety of options and columns so that we can do proper exploratory analysis from this dataset.

The second task was to analyse all files and clean it. For cleaning purposes Pig was used as it is faster than hive in terms of processing. Loaded all the files into google bucket “ccass2” from there loaded these files using Pig and did the standard cleaning like removing spaces, N/A, blanks, special characters, etc. one by one then final clean file is stored into hadoop file system path, removed the “_SUCCESS” file and final cleaned file “part-m-00000” is put into local file directory path. Cleaned all the raw files in the same manner. Also, created tables in hive and loaded clean data into them along with Sneha and Yash as there were 9 tables so we divided this task amongst each other.

Report structure outlining and preparation.

The screenshots for all the tasks and files are in our Git Repository.

3. Sneha Yadav

Data Cleaning and Preprocessing :- As the flight delay dataset contains 8 dimension files. Once the basic cleaning of all the dimension files was completed using PigLatin, then using the component gateway I connected to the Jupyter Notebook on the Master Node and accessed all the part-m files of each and every dimension and cleaned these files using the pandas module.

Data Preparation:- Removing duplicates records and restructuring the columns.

For aggregate tables, multiple tables were merged based on the joining criteria.

Created the f_db database and loaded the cleaned files into hive tables.

Loading the file into Jupyter notebooks for preprocessing and cleaning as there is a delay observed while fetching the queries from multiple tables hence we have merged the tables to form one table to reduce the time taken while processing. There were two tables of airport weather for the whole year 2019 and only three months (Jan, Feb, Mar) data for the year 2020, which was then merged to form one table to process further. The collection contains duplicate entries for 10 airport names ('NAME') so simply we have 86 airport names. The names of multiple airports are the same. The dataset airport coordinates contains latitude and longitude coordinates that can be used for Airport markers on the map.

- AIRPORT_COORDINATES
- T3_AIR_CARRIER_SUMMARY_AIRPORT_ACTIVITY
- airport_weather

4. Yash Modi

Project Configuration and Setup: Deciding which public cloud to use was based on the use case, for this project the suitable option was to use a dataproc cluster on GCP as it has all the necessary tools and softwares installed and setup. My task was to provision a dataproc cluster with 1 Master Node

and 2 Worker Node and once the cluster was up and running, we changed the default configuration Configure Spark on YARN so that the YARN can handle large amounts of data.

spark.executor.memory=7920m
spark.driver.memory=10000m
spark.yarn.am.memory=7640m
spark.driver.maxResultSize=7920m

Data Cleaning and Preprocessing:- My task was to clean the data in the following three dimension files using a Python Jupyter notebook running on Master Node using component gateway.

- airports_list
- B43_AIRCRAFT_INVENTORY
- CARRIER_DECODE

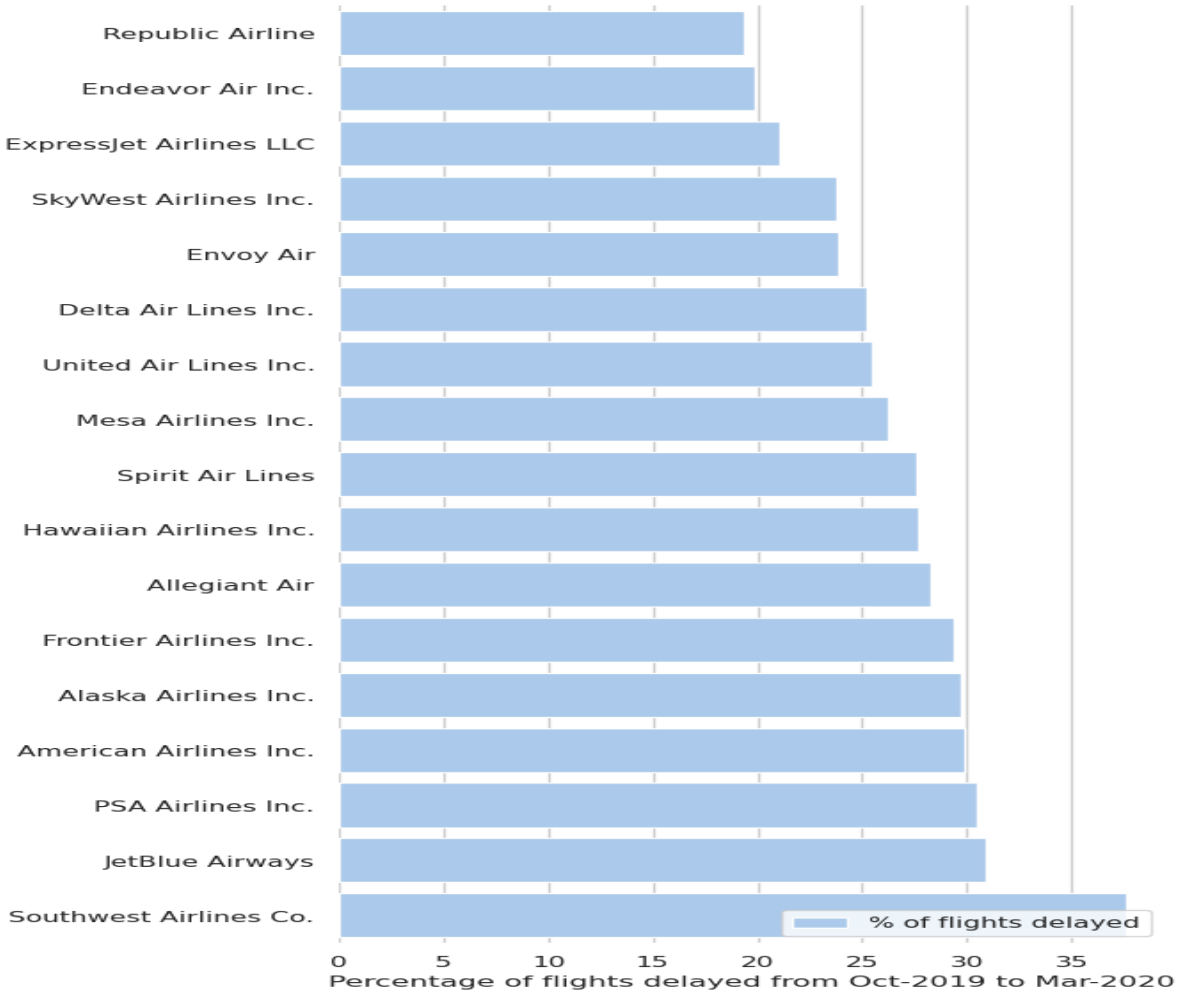
The Airport List file contains the list of airports with the respective airports name, there were duplicates in all the three files so removed the duplicates and prepared these files to load the data in the hive f_db database.
For generating the visuals it is necessary to know the data in the tables so as to create interesting visuals. I took part in designing the questions for visuals to present on the dashboard.

5. Chirag Shah

After the data is cleaned and transformed, exploration is needed to gain knowledge from it . We performed our analysis using Python. Firstly, we use the PySpark library to fetch the hive data. Then with the help of Pandas and Numpy libraries we identified various reasons for flight delays, which carriers and airports have the most delayed flights. Lastly, we plot this data using libraries seaborn, matplotlib and folium to visualise.

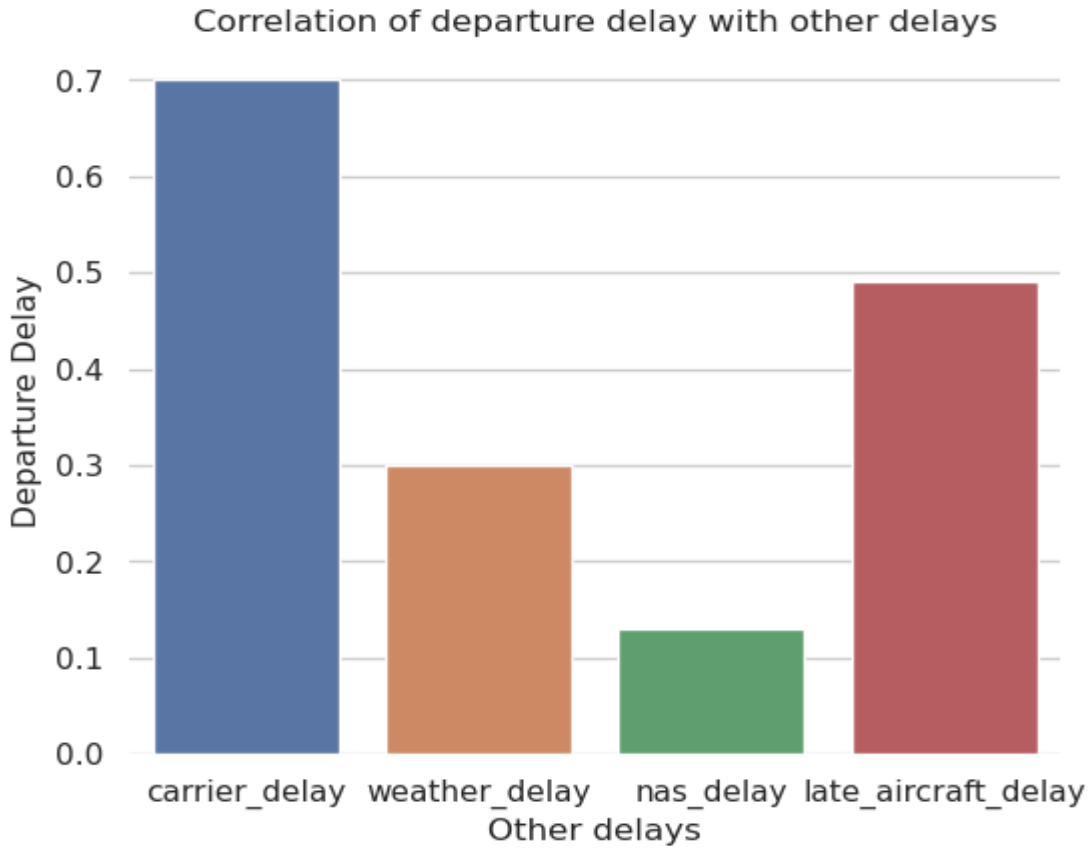
11. Relevant screenshots

The below screenshot represents the percentage of flights delayed by each carrier between October, 2019 and March, 2020.

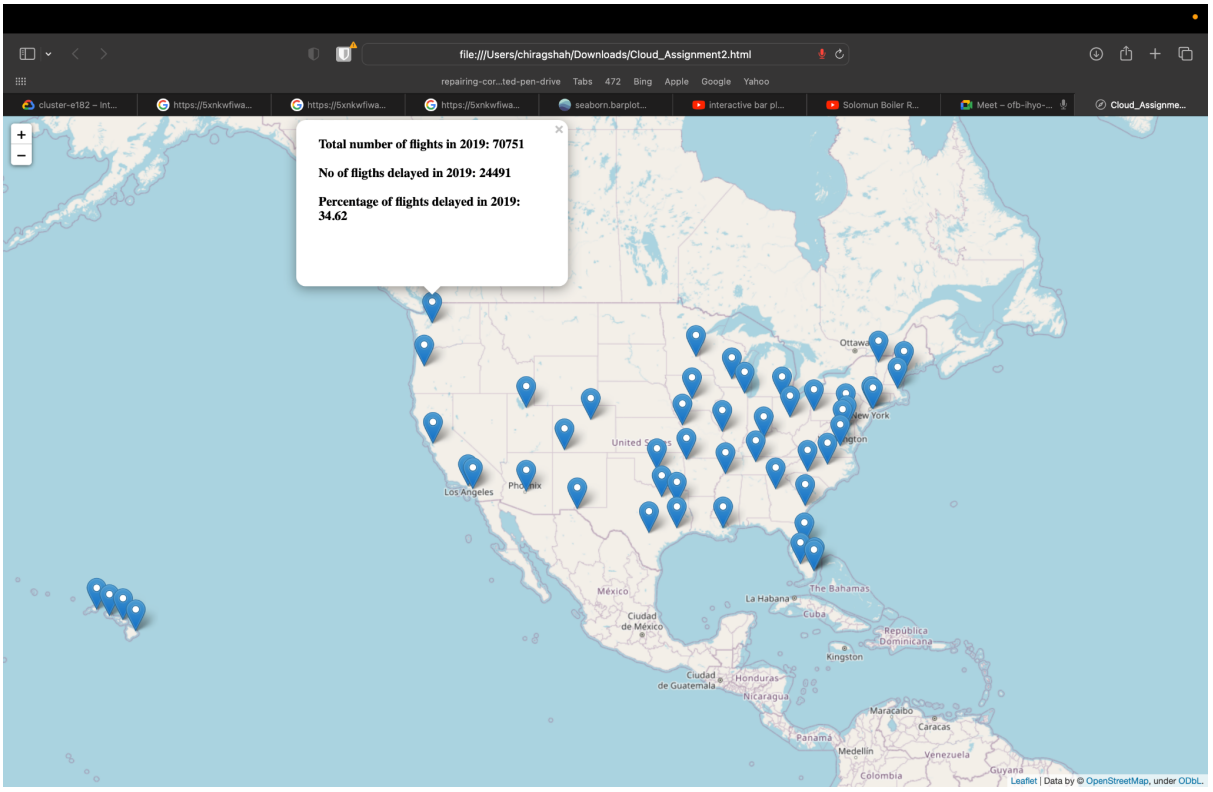


The below screenshot shows a bar plot of various correlations between Departure delay and other delays. The plot depicts which features contributed in making a scheduled

departure delay



The below map projected 50 airports with maximum delays in flight departure.



References:

For resolving issues of challenges.

[1] https://tljh.jupyter.org/en/latest/howto/admin/resource-estimation.html

[2] https://www.linkedin.com/pulse/leveraging-power-jupyter-notebooks-chinmay-wyawahare/