ETL Project

Ana Saavedra | Analiza Asuncio | Avantika Singh | Eric Wetzel | Sahar Alaei | Shilpa Nagendra

Extraction–

Dates: 12/31/2019 to 5/31/2020

Obtained Data from:

1. https://opensky-network.org/community/blog/item/6-opensky-covid-19-flight-dataset
2. https://ourairports.com/data/
3. https://ourworldindata.org/coronavirus-source-data

We wanted to analyze flight airline data from before the closures of the pandemic and the effects while full quarantine was enforced. We found data that supported this theory not just here in the US but throughout the globe. The information was extensive and not necessarily legible as raw data. Which was great for the objectives of the project are. We layered in the COVID data for an additional angle of the data analysis.

Transformation–

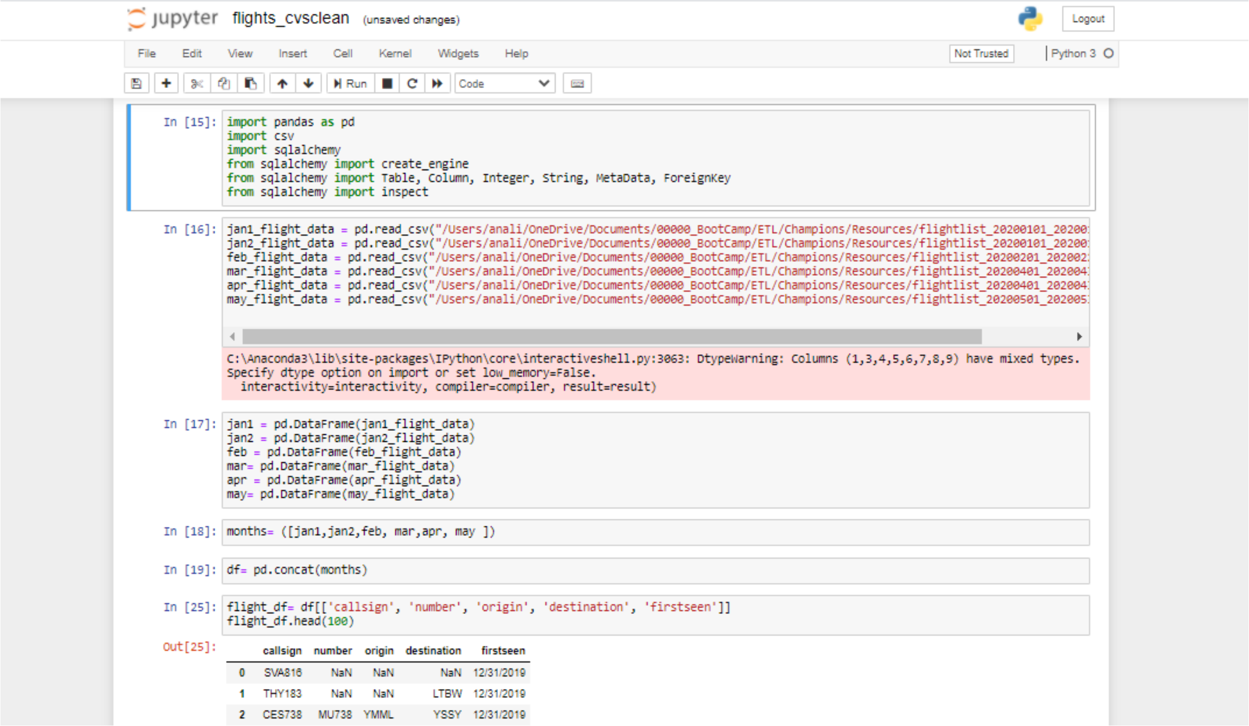
Data I

As a group, we decided to remove the following data using Python and Jupyter Notebook. We felt we would not need the information for future projects, or the information in the columns did not make sense for making tables in SQL.

Removed columns:

- registration – number - type code – day

We then merged all of the clean (5) CSV files into one to then import into PostgreSQL. This information will take the database information to make the appropriate tables for the specific information that we are seeking.


jupyter flights_cvsclean (unsaved changes)
Logout
File Edit View Insert Cell Kernel Widgets Help
Not Trusted
Python 3 ○
Run
Code

```
In [15]: import pandas as pd
         import csv
         import sqlalchemy
         from sqlalchemy import create_engine
         from sqlalchemy import Table, Column, Integer, String, MetaData, ForeignKey
         from sqlalchemy import inspect

In [16]: jan1_flight_data = pd.read_csv("/Users/anali/OneDrive/Documents/00000_BootCamp/ETL/Champions/Resources/flightlist_20200101_20200
         jan2_flight_data = pd.read_csv("/Users/anali/OneDrive/Documents/00000_BootCamp/ETL/Champions/Resources/flightlist_20200101_20200
         feb_flight_data = pd.read_csv("/Users/anali/OneDrive/Documents/00000_BootCamp/ETL/Champions/Resources/flightlist_20200201_202002
         mar_flight_data = pd.read_csv("/Users/anali/OneDrive/Documents/00000_BootCamp/ETL/Champions/Resources/flightlist_20200401_202004
         apr_flight_data = pd.read_csv("/Users/anali/OneDrive/Documents/00000_BootCamp/ETL/Champions/Resources/flightlist_20200401_202004
         may_flight_data = pd.read_csv("/Users/anali/OneDrive/Documents/00000_BootCamp/ETL/Champions/Resources/flightlist_20200501_202005

         C:\Anaconda3\lib\site-packages\IPython\core\interactiveshell.py:3063: DtypeWarning: Columns (1,3,4,5,6,7,8,9) have mixed types.
         Specify dtype option on import or set low_memory=False.
           interactivity=interactivity, compiler=compiler, result=result)

In [17]: jan1 = pd.DataFrame(jan1_flight_data)
         jan2 = pd.DataFrame(jan2_flight_data)
         feb = pd.DataFrame(feb_flight_data)
         mar= pd.DataFrame(mar_flight_data)
         apr = pd.DataFrame(apr_flight_data)
         may= pd.DataFrame(may_flight_data)

In [18]: months= ([jan1,jan2,feb, mar,apr, may ])

In [19]: df= pd.concat(months)

In [25]: flight_df= df[['callsign', 'number', 'origin', 'destination', 'firstseen']]
         flight_df.head(100)

Out[25]:
```

	callsign	number	origin	destination	firstseen
0	SVA816	NaN	NaN	NaN	12/31/2019
1	THY183	NaN	NaN	LTBW	12/31/2019
2	CES738	MU738	YMML	YSSY	12/31/2019

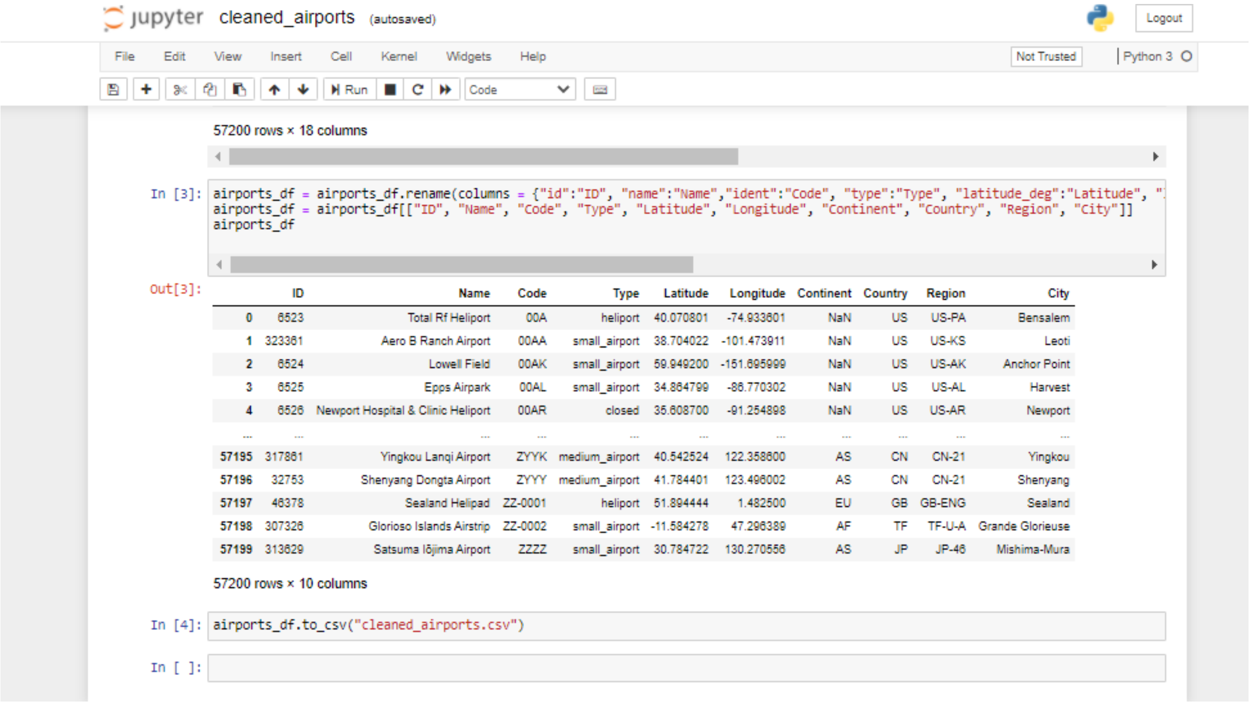
Figure 1. Jupiter notebook of flight data after cleaning using Python.

Data II

The second data set provides more information for the airports. This gives us additional information that the first set of data did not contain. We can later use this data for layering visuals like heat or weather maps. The data includes Lat, Long, City, and State and Country.

Removed columns:

- elevation - schedule - gps code - iata code - local_code - home_link - wikipedia_link – keyword


jupyter cleaned_airports (autosaved)
Logout
File Edit View Insert Cell Kernel Widgets Help
Not Trusted
Python 3 ○
Run
Code

57200 rows × 18 columns

```
In [3]: airports_df = airports_df.rename(columns = {"id":"ID", "name":"Name","ident":"Code", "type":"Type", "latitude_deg":"Latitude", "
        airports_df = airports_df[["ID", "Name", "Code", "Type", "Latitude", "Longitude", "Continent", "Country", "Region", "City"]]
        airports_df

Out[3]:
```

	ID	Name	Code	Type	Latitude	Longitude	Continent	Country	Region	City
0	6523	Total Rf Heliport	00A	heliport	40.070801	-74.933601	NaN	US	US-PA	Bensalem
1	323361	Aero B Ranch Airport	00AA	small_airport	38.704022	-101.473911	NaN	US	US-KS	Leoti
2	6524	Lowell Field	00AK	small_airport	59.949200	-151.695999	NaN	US	US-AK	Anchor Point
3	6525	Epps Airpark	00AL	small_airport	34.864799	-86.770302	NaN	US	US-AL	Harvest
4	6526	Newport Hospital & Clinic Heliport	00AR	closed	35.608700	-91.254898	NaN	US	US-AR	Newport
...	...	...	...	...	...	...	...	...	...	...
57195	317861	Yingkou Lanqi Airport	ZYYK	medium_airport	40.542524	122.358600	AS	CN	CN-21	Yingkou
57196	32753	Shenyang Dongta Airport	ZYYY	medium_airport	41.784401	123.496002	AS	CN	CN-21	Shenyang
57197	46378	Sealand Helipad	ZZ-0001	heliport	51.894444	1.482500	EU	GB	GB-ENG	Sealand
57198	307326	Glorioso Islands Airstrip	ZZ-0002	small_airport	-11.584278	47.296389	AF	TF	TF-U-A	Grande Glorieuse
57199	313629	Satsuma Iōjima Airport	ZZZZ	small_airport	30.784722	130.270556	AS	JP	JP-46	Mishima-Mura

57200 rows × 10 columns

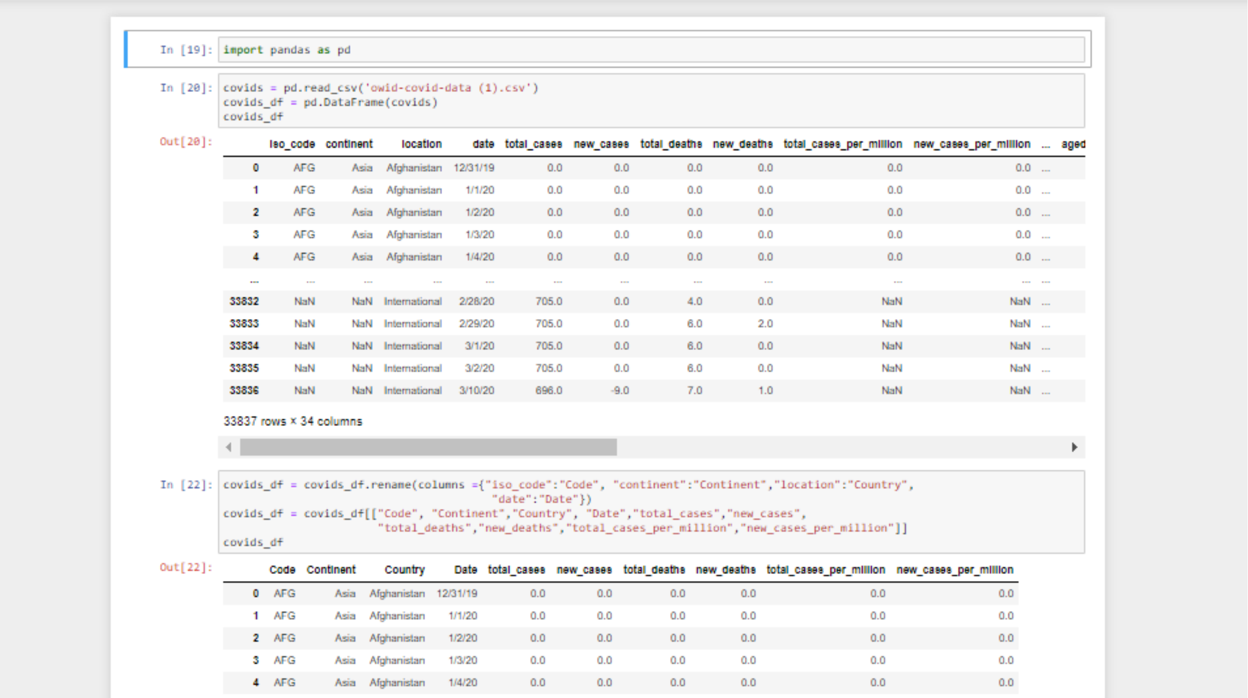
```
In [4]: airports_df.to_csv("cleaned_airports.csv")

In [ ]:
```

Figure 2. Jupiter notebook of airport data after cleaning using Python.

Data III

This is for our COVID data. Due to the large number of columns the raw data, we just coded what information we wanted and extracted it to new and clean CSV to import into PostgreSQL. This includes a renaming function to those columns to be able to merge in with other data in PostgreSQL effectively



In [19]:
```
import pandas as pd
```

In [20]:
```
covids = pd.read_csv('owid-covid-data (1).csv')
covids_df = pd.DataFrame(covids)
covids_df
```

Out[20]:

	iso_code	continent	location	date	total_cases	new_cases	total_deaths	new_deaths	total_cases_per_million	new_cases_per_million	...	aged
0	AFG	Asia	Afghanistan	12/31/19	0.0	0.0	0.0	0.0	0.0	0.0	...	
1	AFG	Asia	Afghanistan	1/1/20	0.0	0.0	0.0	0.0	0.0	0.0	...	
2	AFG	Asia	Afghanistan	1/2/20	0.0	0.0	0.0	0.0	0.0	0.0	...	
3	AFG	Asia	Afghanistan	1/3/20	0.0	0.0	0.0	0.0	0.0	0.0	...	
4	AFG	Asia	Afghanistan	1/4/20	0.0	0.0	0.0	0.0	0.0	0.0	...	
...	...	...	...	...	...	...	...	...	...	...	...	
33832	NaN	NaN	International	2/28/20	705.0	0.0	4.0	0.0	NaN	NaN	...	
33833	NaN	NaN	International	2/29/20	705.0	0.0	6.0	2.0	NaN	NaN	...	
33834	NaN	NaN	International	3/1/20	705.0	0.0	6.0	0.0	NaN	NaN	...	
33835	NaN	NaN	International	3/2/20	705.0	0.0	6.0	0.0	NaN	NaN	...	
33836	NaN	NaN	International	3/10/20	696.0	-9.0	7.0	1.0	NaN	NaN	...	

33837 rows × 34 columns

In [22]:
```
covids_df = covids_df.rename(columns ={"iso_code":"Code", "continent":"Continent","location":"Country",
                                       "date":"Date"})
covids_df = covids_df[["Code", "Continent","Country", "Date","total_cases","new_cases",
                      "total_deaths","new_deaths","total_cases_per_million","new_cases_per_million"]]
covids_df
```

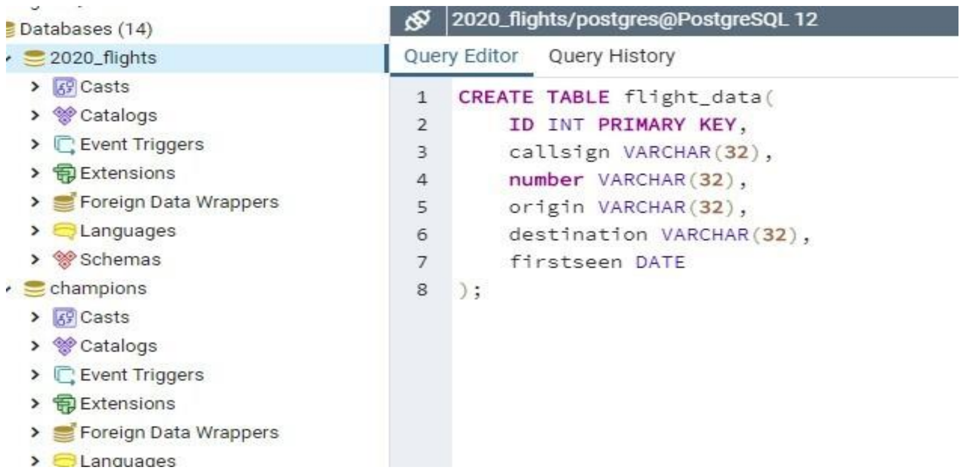
Out[22]:

	Code	Continent	Country	Date	total_cases	new_cases	total_deaths	new_deaths	total_cases_per_million	new_cases_per_million
0	AFG	Asia	Afghanistan	12/31/19	0.0	0.0	0.0	0.0	0.0	0.0
1	AFG	Asia	Afghanistan	1/1/20	0.0	0.0	0.0	0.0	0.0	0.0
2	AFG	Asia	Afghanistan	1/2/20	0.0	0.0	0.0	0.0	0.0	0.0
3	AFG	Asia	Afghanistan	1/3/20	0.0	0.0	0.0	0.0	0.0	0.0
4	AFG	Asia	Afghanistan	1/4/20	0.0	0.0	0.0	0.0	0.0	0.0

Figure 3. Jupiter notebook of COVID data after cleaning using Python.

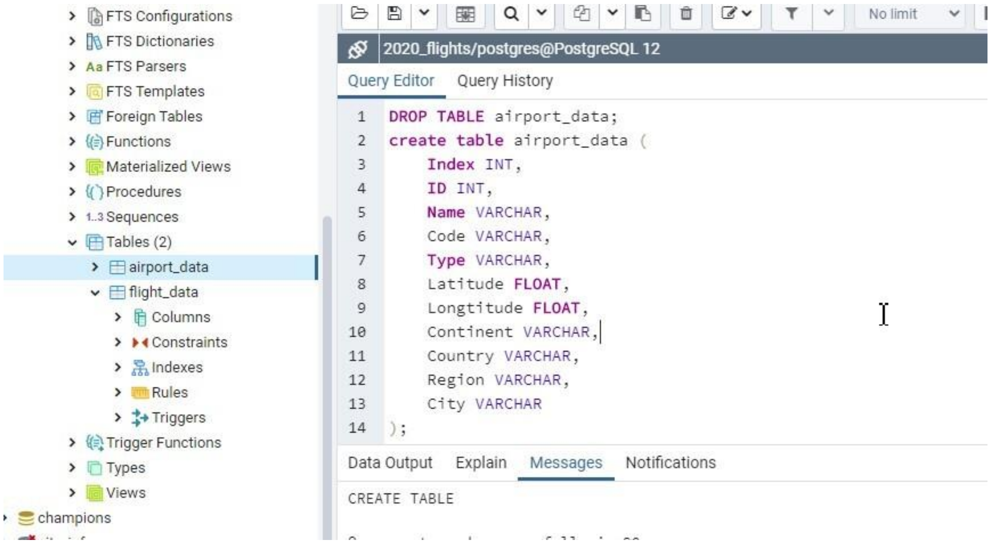
Load –

As a group, we made tables for each of the CVS that we acquired for our project. Later, we can query or join for what specifics we are looking for in the data.



```
CREATE TABLE flight_data(
    ID INT PRIMARY KEY,
    callsign VARCHAR(32),
    number VARCHAR(32),
    origin VARCHAR(32),
    destination VARCHAR(32),
    firstseen DATE
);
```

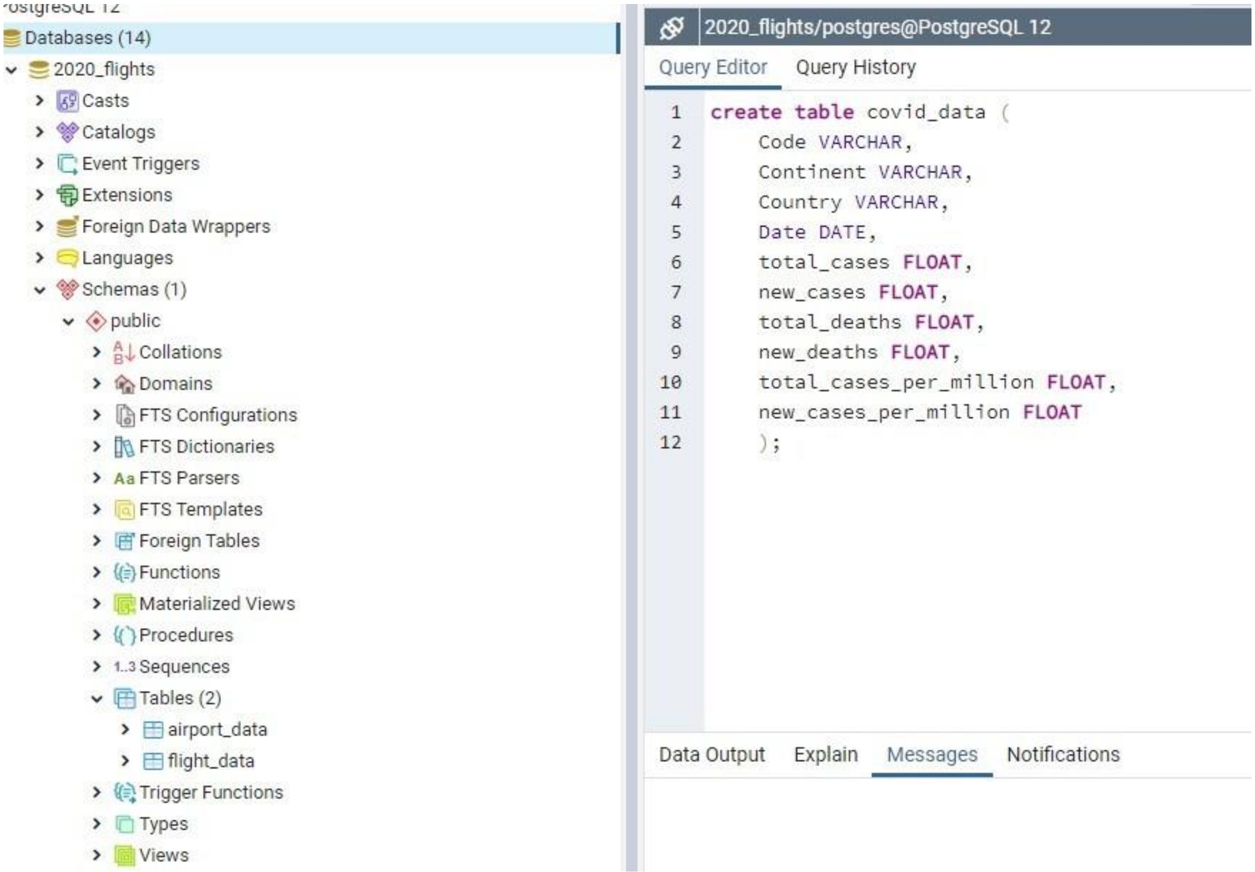
Figure 4. Flight Data Table



```sql
DROP TABLE airport_data;
create table airport_data (
    Index INT,
    ID INT,
    Name VARCHAR,
    Code VARCHAR,
    Type VARCHAR,
    Latitude FLOAT,
    Longtitude FLOAT,
    Continent VARCHAR,
    Country VARCHAR,
    Region VARCHAR,
    City VARCHAR
);
```

```
CREATE TABLE
```

Figure 5. Airport Data Table



```sql
create table covid_data (
    Code VARCHAR,
    Continent VARCHAR,
    Country VARCHAR,
    Date DATE,
    total_cases FLOAT,
    new_cases FLOAT,
    total_deaths FLOAT,
    new_deaths FLOAT,
    total_cases_per_million FLOAT,
    new_cases_per_million FLOAT
    );
```

Figure 6. COVID Data Table

Summary –

With the airport data and the flight data, we plan on joining them on the column origin. These are the queries we decided as a group to extract from the tables using PostgreSQL. By keeping a large portion of the original raw data, we will be able to obtain more precise information in the next part of the project.

1. Which airlines are part of the dataset?
2. What was the busiest day?
3. What was the slowest day?
4. What was the most popular route?
5. What was the least popular route?
6. What were the top 10 busiest airports?

We had several challenges while working on this group project. We had a hard time sharing the large files thru our GitHub repositories. There are limits to how large the data can be.