Fall 2022

Airline Search Engine Project

Arun Kailasa

AIRLINE SEARCH ENGINE PROJECT

By

Arun Kailasa
B. Tech, Vaagdevi College of Engineering, 2020

GRADUATE CAPSTONE SEMINAR PROJECT

Submitted in partial fulfillment of the requirements

For the Degree of Master of Science,

With a Major in Computer Science

Governors State University
University Park, IL 60484

2022

ABSTRACT

The Airline Search Engine Project is a tool that helps anyone to find the facts/data related to Airlines/Airports. For this project, the raw data set is available in the .dat format. We are going to use this data, which can be downloaded from [1].

The tool may also do some first cleaning of the data if needed for forming dimensional data, the cleaning process such as data value unification, data type and size unification, deduplication, dropping columns, and correcting some known errors.

The data will be processed with the help of languages like Python and Spark. By storing the data, we can distribute storage systems such as Hadoop and Amazon S3. The Integrated Development Environment (IDE) used in this project would be editors such as Google Colab and PyCharm.

This tool can be run as a job in different clusters such as EMR (Elastic MapReduce), HDInsight, Cloudera, and Databricks. It can solve/derive data by analyzing terra bytes of raw data into useful information. We can create reports out of it, which Data Analysts, Data Scientists, and businesspeople can use.

Table of Contents

1 Project Description

This tool is going to process various raw data sets which you can find in **Appendix A** and from this raw data we can derive some useful facts which you can find in **Appendix B**. The tool will process raw data and initially create various dimensional data models such as Airports, Airlines, Routes, Planes, and Countries tables. The schema of those tables can be found in **Appendix C**.

1.1 Appendix A:

The raw data sets are

1) Airport.dat – Which contains information related to Airports such as Airport id, Airport Name, etc.
2) Airlines.dat – Which contains information related to Airlines such as Airline id, Airline name, etc et al. [5].
3) Routes.dat – Which contains information related to routes such as Source Airport, Destination Airport.
4) Plane – Which contains information related to plane such as Plane name, etc.
5) Country – Which contains information related to Country name, iso_code et al. [5].

1.2 Appendix B:

a. Find list of Airports operating in the Country X.

b. Find the list of Airlines having X stops.

c. List of Airlines operating with code share.

d. Find the list of Active Airlines in the United States.

    i. Airline aggregation:

e. Which Country (or) Territory has the highest number of Airports.

f. The top K cities with most Incoming/Outgoing Airlines.

    i. Trip recommendation:

g. Define a trip as a sequence of connected routes. Find a trip that connects two cities X and Y (reachability).

h. Find a trip that connects X and Y with less than Z stops (constrained reachability).

i. Find all the cities reachable within d hops of a city (bounded reachability).

a. Fast Transitive closure/connected component implemented in parallel/distributed algorithms.

1.3 Appendix C:

Table name	**Airports**
airport_id	bigint
Name	string
city	string
country	String
iata	String
icao	String
latitude	Double
longitude	Double
altitude	Bigint
timezone	Double
dst	String
tz_database	String
type	String
source	String

Table name	Airlines
Airlineid	bigint
Name	string
Alias	String
Iata	String
Icao	String
Callsign	String
Country	String
active	String

Table Name	Routes
Airline	string
Airlineid	String
Source_airport	String
Source_airport_id	String
Destination_airport	string
Destination_airportid	string
Codeshare	string
Stops	Bigint
Equipment	string

Table Name	Planes
Name	String
Iata	String
Icao	string

Table Name	Countries
Name	String
Iso_code	String
Dafif_code	String

2 *Architecture and flow of the Data Pipeline*

The given data set will be uploaded to either the Amazon S3 bucket et al. [4,6] or can be uploaded to Hadoop attributed filesystem. The uploaded data will be processed with the help of Apache Spark engine et al. [3]. The Apache Spark engine mostly will be cluster like Amazon Elastic Map Reduce (EMR) service or locally installed Spark. Once the data is processed, we can store the data again in another Amazon S3 bucket or it can be stored in the HDFS also. The output data can be viewed with the help of various tools such as Apache Superset, Tableau, Presto query engine, Amazon Athena et al. [6] or it can be created as another Hive table et al. [3].

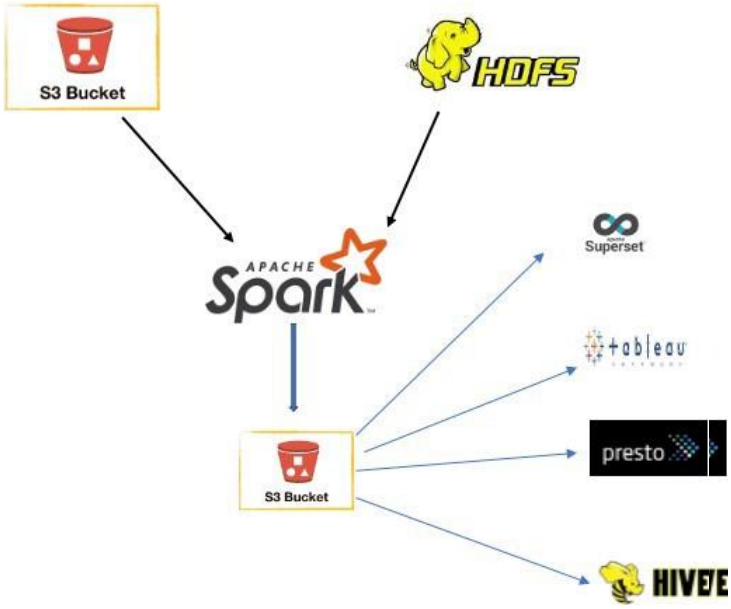


Figure 1: Architecture and flow of the Data Pipeline [2].

3 *Tools and Technologies*

Google Colab, Spark, Python, AWS, PyCharm, HDFS, AWS Resources such as S3 bucket, Identity Access Management (IAM), AWS Glue Data Catalog, AWS Glue Crawler, AWS Athena, SQL.

4 Project Structure

The Airline Search Engine Project is developed with Integrated Development Environment (IDE) such as PyCharm et al. [8] and by installing necessary language binaries like PySpark and Spark et al. [3,11].

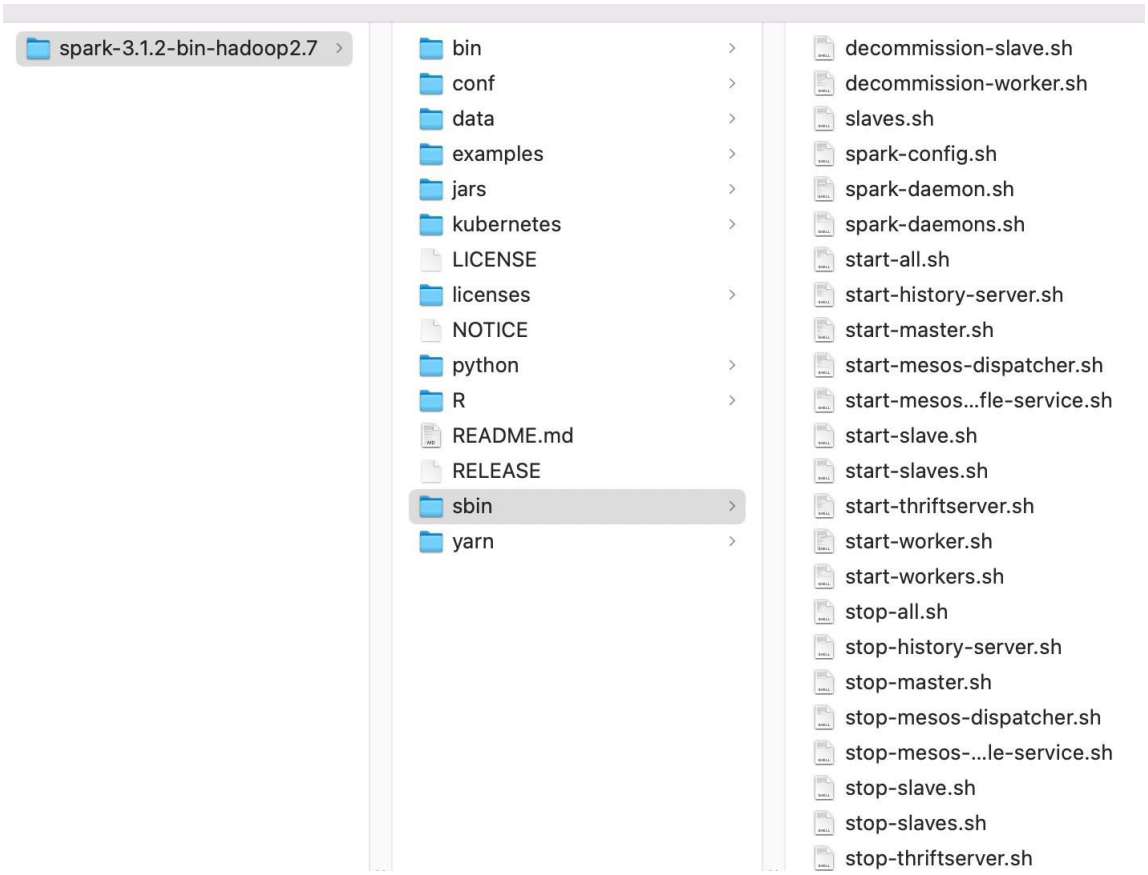


Figure 2: PySpark version 3.1.2 and Spark version 3.1.2.

The pip list command shows the PySpark version used in this project. PySpark version 3.1.2 and Spark version 3.1.2.

```
Package    Version
---------- -------
pip        21.1.2
py4j       0.10.9
pyspark    3.1.2
setuptools 57.0.0
wheel      0.36.2
WARNING: You are using pip version 21.1.2; howe
You should consider upgrading via the '/Users/p
```

Figure 3: pip list command showing PySpark Version.

5 Project folder Hierarchy

A separate project is created for this, and it includes a separate virtual environment to install the necessary project dependency modules like Pandas et al. [10], NumPy, etc. The folder structure includes a separate folder for data loading/reading and some util Spark code will be developed and developed folder like the util folder.

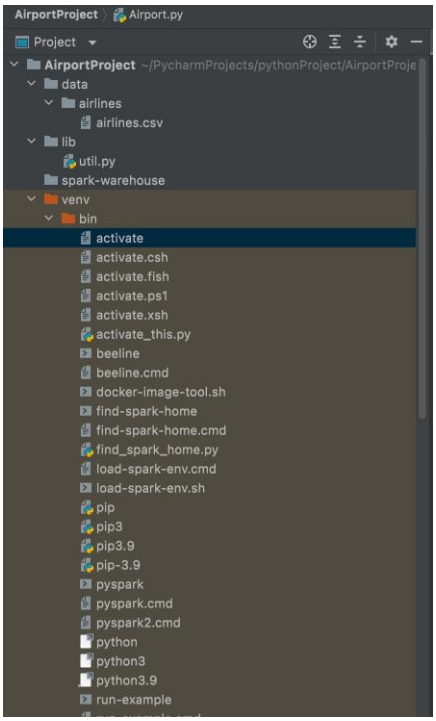


Figure 4: Project folder Hierarchy

6 *Utility Code*

Utility code was developed to read the Spark session configuration and to set the Spark configuration at run time as well. The load_df utility was developed to read the data. You can find the code in the belowscreenshot.

```python
import configparser
from pyspark import SparkConf
def get_spark_app_config():
    conf = SparkConf()
    config = configparser.ConfigParser()
    config.read("spark.conf")
    for (key, val) in config.items("SPARK_APP_CONFIGS"):
        conf.set(key, val)
    return conf

def load_df(spark, data_file):
    # You are telling header is there in this file
    # You assume the data type by yourself by specifying inferSchema
    return spark.read.option("header", "False").option("inferSchema", "true").csv(data_file)
```

Figure 5 : Utility Code

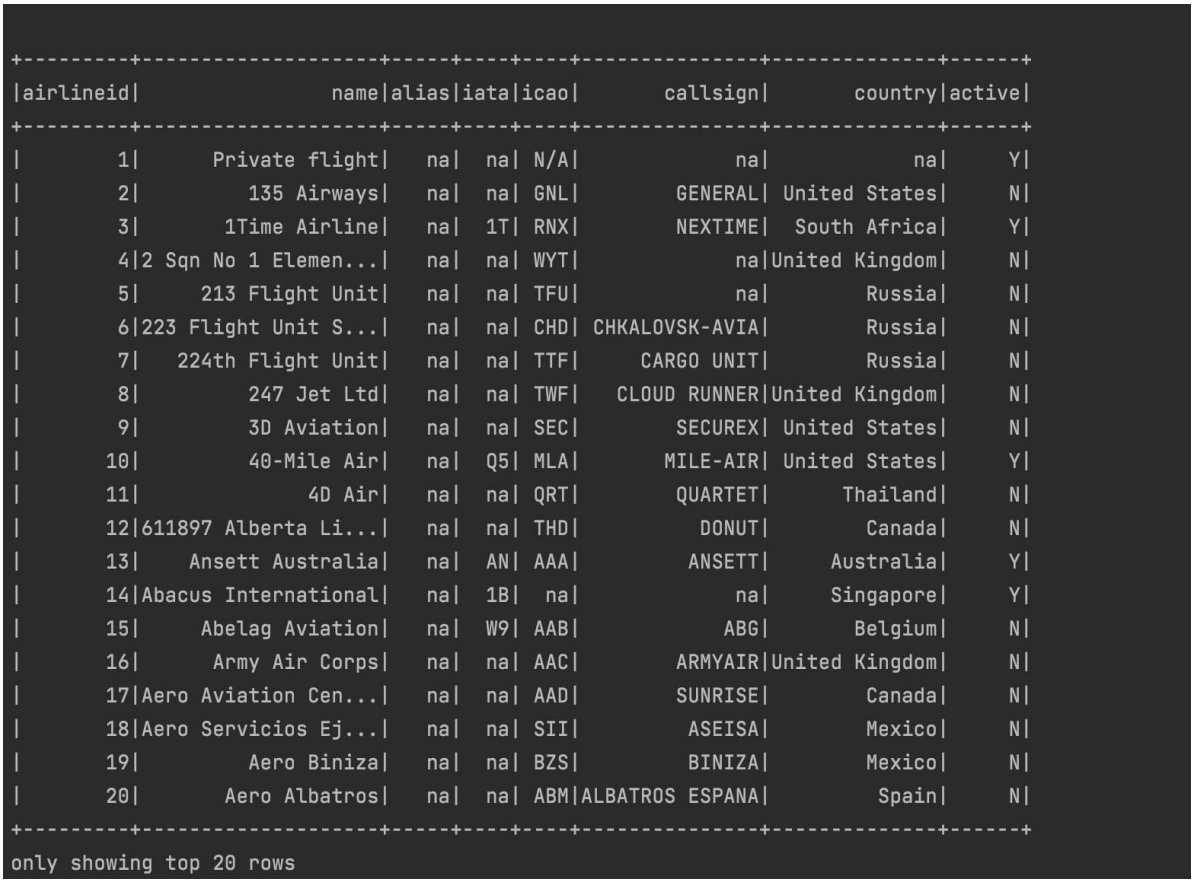
7 *Code for creating the Spark session*

```python
import sys
from pyspark import SparkConf,SparkContext
from pyspark.sql import SparkSession
from lib.util import get_spark_app_config,load_df
spark_conf = get_spark_app_config()
spark = SparkSession.builder.config(conf=spark_conf).getOrCreate()
```

Figure 6: Code for creating the Spark session

8 Transformation and Cleaning

Doing some transformation and cleaning work like replace strings like “\N” and “- “with na and transformation by replacing all null values with strings like na. You can find the output in the screen below after this transformation and cleaning.



```
+---------+--------------------+-----+----+----+--------------+-------------+------+
|airlineid|                name|alias|iata|icao|       callsign|      country|active|
+---------+--------------------+-----+----+----+--------------+-------------+------+
|        1|      Private flight|   na|  na| N/A|            na|           na|     Y|
|        2|         135 Airways|   na|  na| GNL|       GENERAL| United States|     N|
|        3|       1Time Airline|   na|  1T| RNX|       NEXTIME|  South Africa|     Y|
|        4|2 Sqn No 1 Elemen...|   na|  na| WYT|            na|United Kingdom|     N|
|        5|     213 Flight Unit|   na|  na| TFU|            na|       Russia|     N|
|        6|223 Flight Unit S...|   na|  na| CHD| CHKALOVSK-AVIA|       Russia|     N|
|        7|   224th Flight Unit|   na|  na| TTF|    CARGO UNIT|       Russia|     N|
|        8|         247 Jet Ltd|   na|  na| TWF|   CLOUD RUNNER|United Kingdom|     N|
|        9|         3D Aviation|   na|  na| SEC|       SECUREX| United States|     N|
|       10|         40-Mile Air|   na|  Q5| MLA|      MILE-AIR| United States|     Y|
|       11|              4D Air|   na|  na| QRT|       QUARTET|     Thailand|     N|
|       12|611897 Alberta Li...|   na|  na| THD|         DONUT|       Canada|     N|
|       13|    Ansett Australia|   na|  AN| AAA|        ANSETT|    Australia|     Y|
|       14|Abacus International|   na|  1B|  na|            na|    Singapore|     Y|
|       15|     Abelag Aviation|   na|  W9| AAB|           ABG|      Belgium|     N|
|       16|      Army Air Corps|   na|  na| AAC|       ARMYAIR|United Kingdom|     N|
|       17|Aero Aviation Cen...|   na|  na| AAD|       SUNRISE|       Canada|     N|
|       18|Aero Servicios Ej...|   na|  na| SII|        ASEISA|       Mexico|     N|
|       19|         Aero Biniza|   na|  na| BZS|        BINIZA|       Mexico|     N|
|       20|       Aero Albatros|   na|  na| ABM|ALBATROS ESPANA|        Spain|     N|
+---------+--------------------+-----+----+----+--------------+-------------+------+
only showing top 20 rows
```

Figure 7: Transformation and Cleaning

9 *Complete Project Code:*

```
import sys
from pyspark import
SparkConf,SparkContextfrom pyspark.sql
import SparkSession
from lib.util import
get_spark_app_config,load_dfspark_conf =
get_spark_app_config()
#print(spark_conf.toDebugString())
spark =
SparkSession.builder.config(conf=spark_conf).getOrCreate()#
airline module
airlines_load_df =
load_df(spark,"data/airlines/airlines.csv")# Changing to
new column names
airlines_column_name_list =
["airlineid","name","alias","iata","icao","callsign","country","active"]
airlines_raw_df = airlines_load_df.toDF(*airlines_column_name_list)
airlines_df = airlines_raw_df.replace('\\N',"na").replace("-
```

Figure 8: Project Code

```
# airports module
airports_load_df = load_df(spark,"data/airports/airports.csv")
airports_column_name_list =
["airportid","name","city","country","iata","icao","latitude","longitude","al
titude","timezone","dst","tzdatabase","type","source"]
airports_raw_df = airports_load_df.toDF(*airports_column_name_list)
airports_df = airports_raw_df.replace('\\N',"na").replace("-
","na").fillna("na")
airports_df.write.mode("overwrite").option("header",True).csv("output/airport
s/")
airports_df.createTempView("airports")
# countries module
countries_load_df = load_df(spark,"data/countries/countries.csv")
countries_column_name_list = ["name","isocode","dafifcode"]
countries_raw_df = countries_load_df.toDF(*countries_column_name_list)
countries_df = countries_raw_df.replace('\\N',"na").replace("-
","na").fillna("na")
countries_df.write.mode("overwrite").option("header",True).csv("output/countr
ies/")
countries df.createTempView("countries")

# planes module
planes_load_df = load_df(spark,"data/planes/planes.csv")
planes_column_name_list = ["name","iata","icao"]
planes_raw_df = planes_load_df.toDF(*planes_column_name_list)
planes_df = planes_raw_df.replace('\\N',"na").replace("-","na").fillna("na")
planes_df.write.mode("overwrite").option("header",True).csv("output/planes/")
planes_df.createTempView("planes")

# routes module
routes_load_df = load_df(spark,"data/routes/routes.csv")
routes_column_name_list =
["airline","airlineid","sourceairport","sourceairportid","destinationairport"
,"destinationairportid","codeshare","stops","equipment"]
routes_raw_df = routes_load_df.toDF(*routes_column_name_list)
routes_df = routes_raw_df.replace('\\N',"na").replace("-","na").fillna("na")
routes_df.write.mode("overwrite").option("header",True).csv("output/routes/")
routes_df.createTempView("routes")

# Find list of Airports operating in the Country X
spark.sql("select *, count(*) over () as count from airports where country =
'Greenland'").show(100)

# Find the list of Airlines having X stops
spark.sql("select * from routes where stops > 0").show(100)

# List of Airlines operating with code share
spark.sql("select * , count(*) over() as count from routes where codeshare !=
'na' ").show(100)

# Find the list of Active Airlines in the United States
spark.sql("select *, count(*) over() as count from airlines where country =
'United States' and active = 'Y'").show(100)

# Which country (or) territory has the highest number of Airports
spark.sql("select count(*) as cnt, country from airports group by country
order by cnt desc ").show(20)
```

Figure 9: Complete Project Code

```
# # The top k cities with most incoming airlines
spark.sql("""select * from (select airports.airportid, airports.name,
airports.city, airports.country, tb2.incoming_flight_count from
airports inner join (select count(*) as incoming_flight_count,
destinationairportid from routes group by destinationairportid ) tb2
on airports.airportid = tb2.destinationairportid) otb order by
otb.incoming_flight_count desc""").show(100)

# # The top k cities with most outgoing airlines
spark.sql("""select * from (select airports.airportid, airports.name,
airports.city, airports.country, tb2.outgoing_flight_count from
airports inner join (select count(*) as outgoing_flight_count,
sourceairportid from routes group by sourceairportid ) tb2
on airports.airportid = tb2.sourceairportid) otb order by
otb.outgoing_flight_count desc""").show(100)

#Trip that connects two cities X and Y
spark.sql("""select * from routes where sourceairportid = '2613' and
destinationairportid='2531' """).show(100)

spark.sql("""select * from routes where sourceairportid = '2613' and
destinationairportid='2531' and stops < 1 """).show(100)

spark.sql("""select destinationairport from routes where stops = 1
""").show(100)
```

Figure 10: Complete Project Code

```
import configparser
from pyspark import SparkConf
def get_spark_app_config():
    conf = SparkConf()
    config = configparser.ConfigParser()
    config.read("spark.conf")
    for (key, val) in config.items("SPARK_APP_CONFIGS"):
        conf.set(key,val)
    return conf

def load_df(spark, data_file):
    # You are telling header is there in this file
    # You assume the data type by yourself by specifying inferSchema
    return spark.read.option("header", "False").option("inferSchema",
"true").csv(data_file)
```

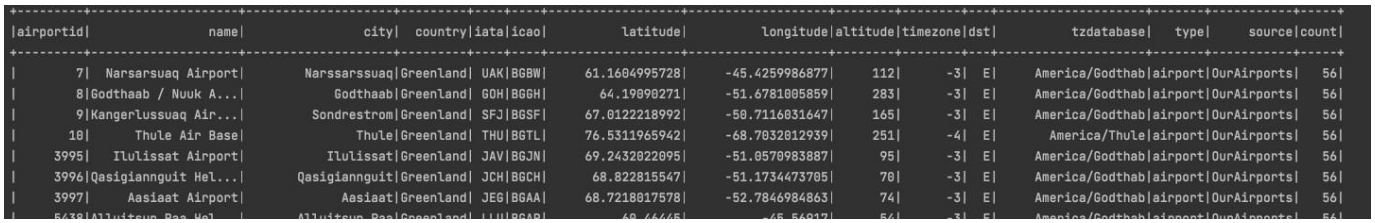
Figure 11: Spark Session Configuration Code

10 Project Output Screenshots

10.1 Find a list of Airports operating in the Country X

```
spark.sql("select *, count(*) over () as count from airports where country =
'Greenland'").show(100)
```

Output:



airportid	name	city	country	iata	icao	latitude	longitude	altitude	timezone	dst	tzdatabase	type	source	count
7	Narsarsuaq Airport	Narssarssuaq	Greenland	UAK	BGBW	61.1604995728	-45.4259986877	112	-3	E	America/Godthab	airport	OurAirports	56
8	Godthaab / Nuuk A...	Godthaab	Greenland	GOH	BGGH	64.19090271	-51.6781005859	283	-3	E	America/Godthab	airport	OurAirports	56
9	Kangerlussuaq Air...	Sondrestrom	Greenland	SFJ	BGSF	67.0122218992	-50.7116031647	165	-3	E	America/Godthab	airport	OurAirports	56
10	Thule Air Base	Thule	Greenland	THU	BGTL	76.5311965942	-68.7032012939	251	-4	E	America/Thule	airport	OurAirports	56
3995	Ilulissat Airport	Ilulissat	Greenland	JAV	BGJN	69.2432022095	-51.0570983887	95	-3	E	America/Godthab	airport	OurAirports	56
3996	Qasigiannguit Hel...	Qasigiannguit	Greenland	JCH	BGCH	68.822815547	-51.1734473705	70	-3	E	America/Godthab	airport	OurAirports	56
3997	Aasiaat Airport	Aasiaat	Greenland	JEG	BGAA	68.7218017578	-52.7846984863	74	-3	E	America/Godthab	airport	OurAirports	56
5438	Alluitsup Paa Hel...	Alluitsup Paa	Greenland	LLU	BGAA	60.46445	-45.56917	54	-3	E	America/Godthab	airport	OurAirports	56

Figure 10.1: Output for list of Airports operating in the Country X (‘GREENLAND’)

10.2 Find the list of Airlines having X stops

```
spark.sql("select * from routes where stops > 0").show(100)
```

Output:

airline	airlineid	sourceairport	sourceairportid	destinationairport	destinationairportid	codeshare	stops	equipment
5T	1623	YRT	132	YEK	50	na	1	ATR
AC	330	ABJ	253	BRU	302	na	1	333
AC	330	YVR	156	YBL	30	na	1	BEH
CU	1936	FCO	1555	HAV	1909	na	1	767
FL	1316	HOU	3566	SAT	3621	na	1	735
FL	1316	MCO	3878	HOU	3566	na	1	73W
FL	1316	MCO	3878	ORF	3611	na	1	717
SK	4319	ARN	737	GEV	715	na	1	ATP
WN	4547	BOS	3448	MCO	3878	na	1	73W
WN	4547	MCO	3878	BOS	3448	na	1	73W
WN	4547	MCO	3878	CAK	4112	na	1	73C 73W

Figure 10.2: Output for list of Airlines having X stops

10.3 List of Airlines operating with codeshare

spark.sql("select * , count(*) over() as count from routes where codeshare != 'na' ").show(100)

Output:

airline	airlineid	sourceairport	sourceairportid	destinationairport	destinationairportid	codeshare	stops	equipment	count
2P	897	GES	2402	MNL	2397	Y	0	320	14597
2P	897	MNL	2397	GES	2402	Y	0	320	14597
4M	3201	DFW	3670	EZE	3988	Y	0	777	14597
4M	3201	EZE	3988	DFW	3670	Y	0	777	14597
4M	3201	EZE	3988	JFK	3797	Y	0	777	14597
4M	3201	JFK	3797	EZE	3988	Y	0	777	14597
5N	503	ARH	4362	CSH	6110	Y	0	AN4	14597
5N	503	ARH	4362	MMK	2949	Y	0	AN4	14597
5N	503	ARH	4362	USK	4369	Y	0	AN4	14597
5N	503	CSH	6110	ARH	4362	Y	0	AN4	14597
5N	503	MMK	2949	ARH	4362	Y	0	AN4	14597

Figure 10.3: Output for list of Airlines operating with codeshare

10.4 Find the list of Active Airlines in the United States

spark.sql("select *, count(*) over() as count from airlines where country = 'United States' andactive = 'Y'").show(100)

Output:

airlineid	name	alias	iata	icao	callsign	country	active	count
10	40-Mile Air	na	Q5	MLA	MILE-AIR	United States	Y	156
22	Aloha Airlines	na	AQ	AAH	ALOHA	United States	Y	156
24	American Airlines	na	AA	AAL	AMERICAN	United States	Y	156
35	Allegiant Air	na	G4	AAY	ALLEGIANT	United States	Y	156
109	Alaska Central Ex...	na	KO	AER	ACE AIR	United States	Y	156
149	Air Cargo Carriers	na	2Q	SNC	NIGHT CARGO	United States	Y	156
210	Airlift Internati...	na	na	AIR	AIRLIFT	United States	Y	156
281	America West Airl...	na	HP	AWE	CACTUS	United States	Y	156
282	Air Wisconsin	na	ZW	AWI	AIR WISCONSIN	United States	Y	156
287	Allegheny Commute...	na	na	ALO	ALLEGHENY	United States	Y	156
295	Air Sunshine	na	na	RSI	AIR SUNSHINE	United States	Y	156

Figure 10.4: Output for list of active Airlines in the United States

10.5 Which country (or) territory has the highest number of Airports

spark.sql("select count(*) as cnt, country from airports group by country order by cnt desc").show(20)

Output:

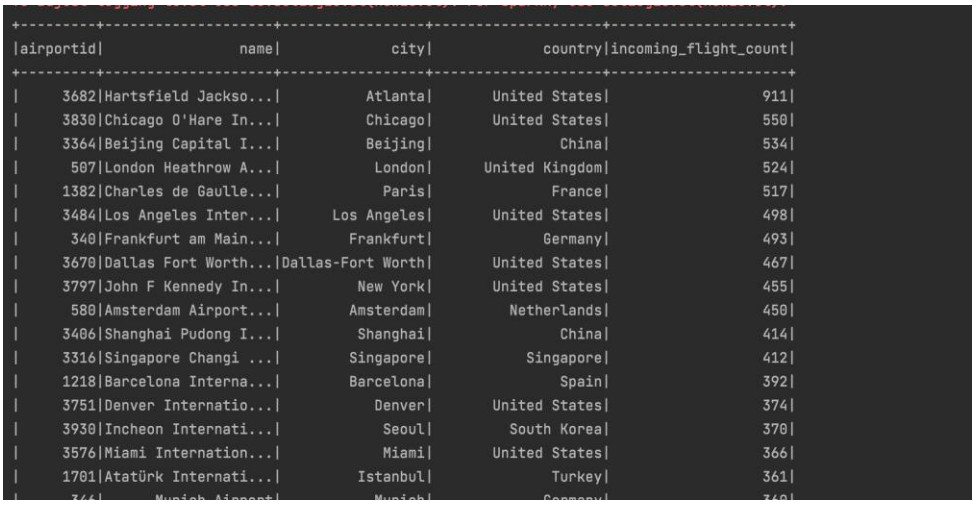
cnt	country
1512	United States
430	Canada
334	Australia
264	Russia
264	Brazil
249	Germany
241	China
217	France
167	United Kingdom
148	India
145	Indonesia

Figure 10.5: Output for the countries with highest number of Airports

10.6 The top K cities with most Incoming Airlines

spark.sql("""select * from (select airports.airportid, airports.name, airports.city,airports.country, tb2.incoming_flight_count from airports inner join (select count (*) as incoming_flight_count, destinationairportid from routesgroup by destinationairportid) tb2 on airports.airportid = tb2.destinationairportid) otb order by otb.incoming_flight_countdesc""").show(100)

Output:



airportid	name	city	country	incoming_flight_count
3682	Hartsfield Jackso...	Atlanta	United States	911
3830	Chicago O'Hare In...	Chicago	United States	550
3364	Beijing Capital I...	Beijing	China	534
507	London Heathrow A...	London	United Kingdom	524
1382	Charles de Gaulle...	Paris	France	517
3484	Los Angeles Inter...	Los Angeles	United States	498
340	Frankfurt am Main...	Frankfurt	Germany	493
3670	Dallas Fort Worth...	Dallas-Fort Worth	United States	467
3797	John F Kennedy In...	New York	United States	455
580	Amsterdam Airport...	Amsterdam	Netherlands	450
3406	Shanghai Pudong I...	Shanghai	China	414
3316	Singapore Changi ...	Singapore	Singapore	412
1218	Barcelona Interna...	Barcelona	Spain	392
3751	Denver Internatio...	Denver	United States	374
3930	Incheon Internati...	Seoul	South Korea	370
3576	Miami Internation...	Miami	United States	366
1701	Atatürk Internati...	Istanbul	Turkey	361

Figure 10.6: Output for the top cities with most incoming Airlines

10.7 The top K cities with most Outgoing Airlines

spark.sql("""select * from (select airports.airportid, airports.name, airports.city,airports.country, tb2.outgoing_flight_count from airports inner join (select count (*) as outgoing_flight_count, sourceairportid from routes groupby sourceairportid) tb2 on airports.airportid = tb2.sourceairportid) otb order by otb.outgoing_flight_count desc""").show(100)

Output:

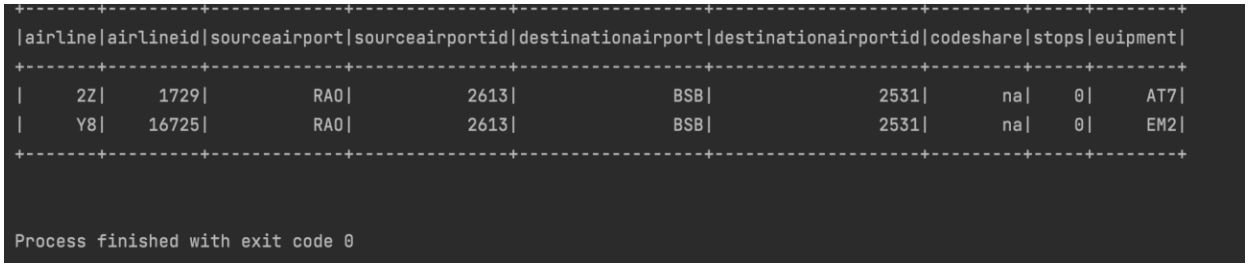
airportid	name	city	country	outgoing_flight_count
3682	Hartsfield Jackso...	Atlanta	United States	915
3830	Chicago O'Hare In...	Chicago	United States	558
3364	Beijing Capital I...	Beijing	China	535
507	London Heathrow A...	London	United Kingdom	527
1382	Charles de Gaulle...	Paris	France	524
340	Frankfurt am Main...	Frankfurt	Germany	497
3484	Los Angeles Inter...	Los Angeles	United States	492
3670	Dallas Fort Worth...	Dallas-Fort Worth	United States	469
3797	John F Kennedy In...	New York	United States	456
580	Amsterdam Airport...	Amsterdam	Netherlands	453
3406	Shanghai Pudong I...	Shanghai	China	411
3316	Singapore Changi ...	Singapore	Singapore	408
1218	Barcelona Interna...	Barcelona	Spain	391

Figure 10.7: Output for top cities with most outgoing Airlines

10.8 Trip that connects two cities X and Y

spark.sql("""select * from routes where sourceairportid = '2613' and destinationairportid='2531' """").show(100)

Output:



airline	airlineid	sourceairport	sourceairportid	destinationairport	destinationairportid	codeshare	stops	equipment
2Z	1729	RAO	2613	BSB	2531	na	0	AT7
Y8	16725	RAO	2613	BSB	2531	na	0	EM2

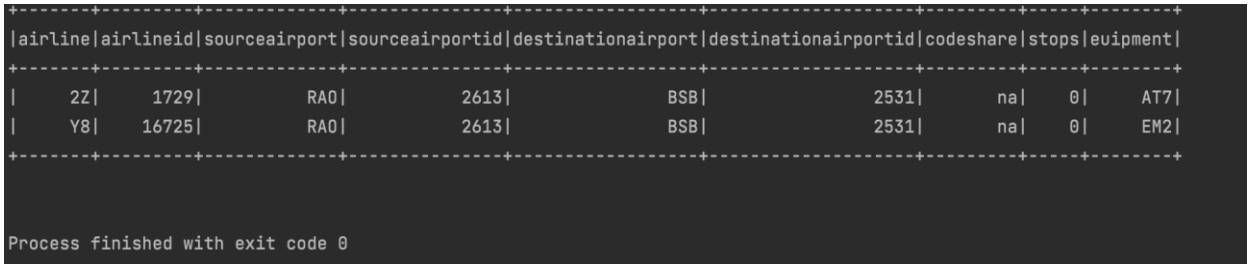
Process finished with exit code 0

Figure 10.8: Output for trip that connects two cities X and Y

10.9 Trip that connects X and Y with less than Z stops

spark.sql("""select * from routes where sourceairportid = '2613' and destinationairportid='2531' and stops < 1 """).show(100)

Output:



```
+-------+---------+-------------+--------------+-----------------+-------------------+---------+-----+---------+
|airline|airlineid|sourceairport|sourceairportid|destinationairport|destinationairportid|codeshare|stops|equipment|
+-------+---------+-------------+--------------+-----------------+-------------------+---------+-----+---------+
|     2Z|     1729|          RAO|          2613|              BSB|               2531|       na|    0|      AT7|
|     Y8|    16725|          RAO|          2613|              BSB|               2531|       na|    0|      EM2|
+-------+---------+-------------+--------------+-----------------+-------------------+---------+-----+---------+


Process finished with exit code 0
```

Figure 10.9: Output for trip that connects X and Y with less than Z stops

10.10 All the cities reachable within d hops of a city

spark.sql("""select destinationairport from routes where stops = 1 """).show(100)

Output:

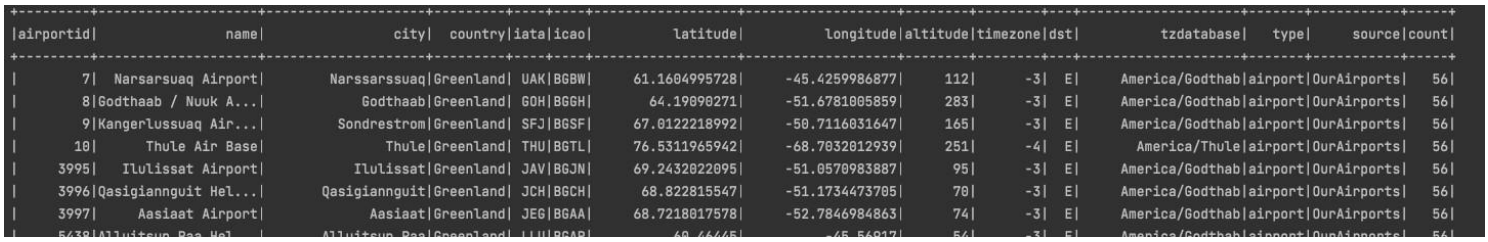
```
+------------------+
|destinationairport|
+------------------+
|               YEK|
|               BRU|
|               YBL|
|               HAV|
|               SAT|
|               HOU|
|               ORF|
|               GEV|
|               MCO|
|               BOS|
|               CAK|
+------------------+


Process finished with exit code 0
```

Figure 10.10: Output for all the cities reachable within d hops of a city

10.11 Find list of Airports operating in the Country X

```
spark.sql("select *, count(*) over () as count from airports where country =
'Greenland'").show(100)
```

Output:



airportid	name	city	country	iata	icao	latitude	longitude	altitude	timezone	dst	tzdatabase	type	source	count
7	Narsarsuaq Airport	Narssarssuaq	Greenland	UAK	BGBW	61.1604995728	-45.4259986877	112	-3	E	America/Godthab	airport	OurAirports	56
8	Godthaab / Nuuk A...	Godthaab	Greenland	GOH	BGGH	64.19090271	-51.6781005859	283	-3	E	America/Godthab	airport	OurAirports	56
9	Kangerlussuaq Air...	Sondrestrom	Greenland	SFJ	BGSF	67.0122218992	-50.7116031647	165	-3	E	America/Godthab	airport	OurAirports	56
10	Thule Air Base	Thule	Greenland	THU	BGTL	76.5311965942	-68.7032012939	251	-4	E	America/Thule	airport	OurAirports	56
3995	Ilulissat Airport	Ilulissat	Greenland	JAV	BGJN	69.2432022095	-51.0570983887	95	-3	E	America/Godthab	airport	OurAirports	56
3996	Qasigiannguit Hel...	Qasigiannguit	Greenland	JCH	BGCH	68.822815547	-51.1734473705	70	-3	E	America/Godthab	airport	OurAirports	56
3997	Aasiaat Airport	Aasiaat	Greenland	JEG	BGAA	68.7218017578	-52.7846984863	74	-3	E	America/Godthab	airport	OurAirports	56
5438	Alluitsup Paa Hel...	Alluitsup Paa	Greenland	LLU	BGAP	60.46445	-45.54917	54	-3	E	America/Godthab	airport	OurAirports	56

Figure 10.11: Output for list of Airports operating in the country X

10.12 *Find the list of Airlines having X stops*

```
spark.sql("select * from routes where stops > 0").show(100)
```

Output:

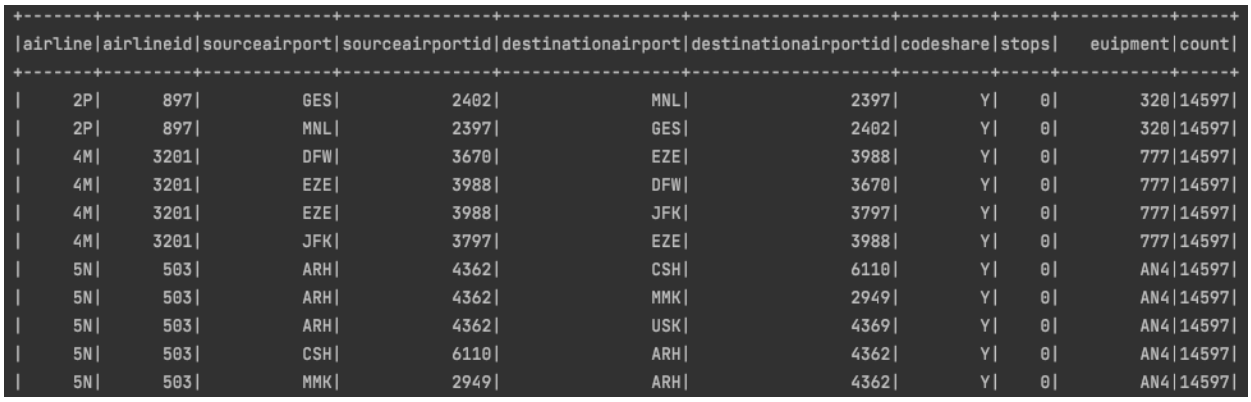
airline	airlineid	sourceairport	sourceairportid	destinationairport	destinationairportid	codeshare	stops	equipment
5T	1623	YRT	132	YEK	50	na	1	ATR
AC	330	ABJ	253	BRU	302	na	1	333
AC	330	YVR	156	YBL	30	na	1	BEH
CU	1936	FCO	1555	HAV	1909	na	1	767
FL	1316	HOU	3566	SAT	3621	na	1	735
FL	1316	MCO	3878	HOU	3566	na	1	73W
FL	1316	MCO	3878	ORF	3611	na	1	717
SK	4319	ARN	737	GEV	715	na	1	ATP
WN	4547	BOS	3448	MCO	3878	na	1	73W
WN	4547	MCO	3878	BOS	3448	na	1	73W
WN	4547	MCO	3878	CAK	4112	na	1	73C 73W

Figure 10.12: Output for the list of Airlines having X stops

10.13 List of Airlines operating with code share

spark.sql("select * , count(*) over() as count from routes where codeshare != 'na' ").show(100)

Output:



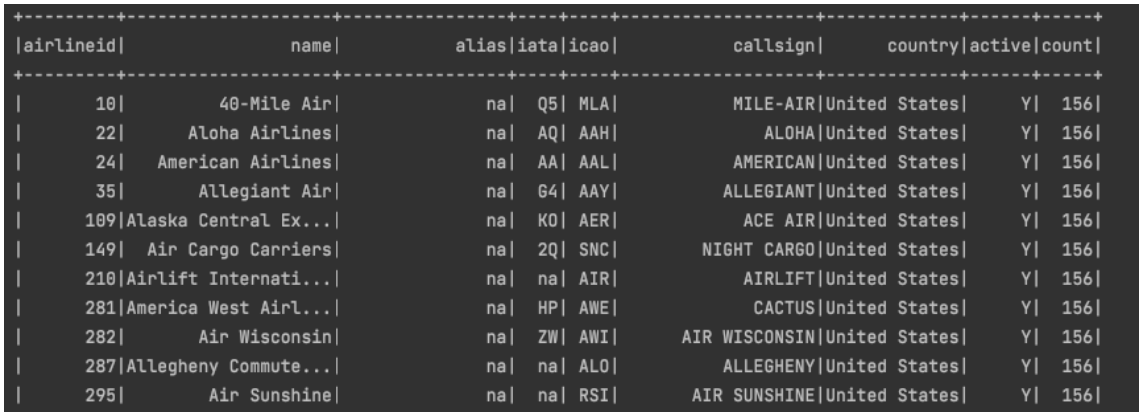
airline	airlineid	sourceairport	sourceairportid	destinationairport	destinationairportid	codeshare	stops	equipment	count
2P	897	GES	2402	MNL	2397	Y	0	320	14597
2P	897	MNL	2397	GES	2402	Y	0	320	14597
4M	3201	DFW	3670	EZE	3988	Y	0	777	14597
4M	3201	EZE	3988	DFW	3670	Y	0	777	14597
4M	3201	EZE	3988	JFK	3797	Y	0	777	14597
4M	3201	JFK	3797	EZE	3988	Y	0	777	14597
5N	503	ARH	4362	CSH	6110	Y	0	AN4	14597
5N	503	ARH	4362	MMK	2949	Y	0	AN4	14597
5N	503	ARH	4362	USK	4369	Y	0	AN4	14597
5N	503	CSH	6110	ARH	4362	Y	0	AN4	14597
5N	503	MMK	2949	ARH	4362	Y	0	AN4	14597

Figure 10.13: Output for Airlines operating with code share

10.14 Find the list of Active Airlines in the United States

spark.sql("select *, count(*) over() as count from airlines where country = 'United States' andactive = 'Y'").show(100)

Output:



airlineid	name	alias	iata	icao	callsign	country	active	count
10	40-Mile Air	na	Q5	MLA	MILE-AIR	United States	Y	156
22	Aloha Airlines	na	AQ	AAH	ALOHA	United States	Y	156
24	American Airlines	na	AA	AAL	AMERICAN	United States	Y	156
35	Allegiant Air	na	G4	AAY	ALLEGIANT	United States	Y	156
109	Alaska Central Ex...	na	KO	AER	ACE AIR	United States	Y	156
149	Air Cargo Carriers	na	2Q	SNC	NIGHT CARGO	United States	Y	156
210	Airlift Internati...	na	na	AIR	AIRLIFT	United States	Y	156
281	America West Airl...	na	HP	AWE	CACTUS	United States	Y	156
282	Air Wisconsin	na	ZW	AWI	AIR WISCONSIN	United States	Y	156
287	Allegheny Commute...	na	na	ALO	ALLEGHENY	United States	Y	156
295	Air Sunshine	na	na	RSI	AIR SUNSHINE	United States	Y	156

Figure 10.14: Output for list of active airlines in the United States

10.15 Which country (or) territory has the highest number of Airports

spark.sql("select count(*) as cnt, country from airports group by country order by cnt desc").show(20)

Output:

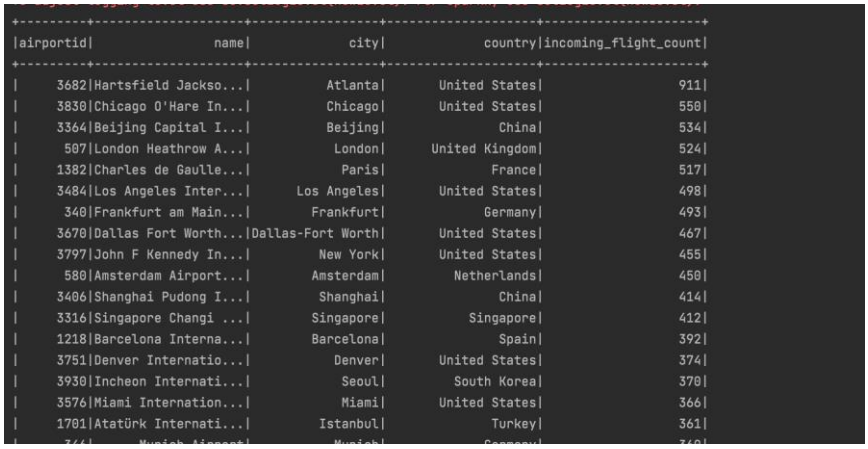
cnt	country
1512	United States
430	Canada
334	Australia
264	Russia
264	Brazil
249	Germany
241	China
217	France
167	United Kingdom
148	India
145	Indonesia

Figure 10.15: Output for multiple countries having highest number of Airports

10.16 The top K cities with most incoming Airlines

spark.sql("""select * from (select airports.airportid, airports.name, airports.city,airports.country, tb2.incoming_flight_count from airports inner join (select count (*) as incoming_flight_count, destinationairportid from routesgroup by destinationairportid) tb2 on airports.airportid = tb2.destinationairportid) otb order by otb.incoming_flight_count desc""").show(100)

Output:



airportid	name	city	country	incoming_flight_count
3682	Hartsfield Jackso...	Atlanta	United States	911
3830	Chicago O'Hare In...	Chicago	United States	550
3364	Beijing Capital I...	Beijing	China	534
507	London Heathrow A...	London	United Kingdom	524
1382	Charles de Gaulle...	Paris	France	517
3484	Los Angeles Inter...	Los Angeles	United States	498
340	Frankfurt am Main...	Frankfurt	Germany	493
3670	Dallas Fort Worth...	Dallas-Fort Worth	United States	467
3797	John F Kennedy In...	New York	United States	455
580	Amsterdam Airport...	Amsterdam	Netherlands	450
3406	Shanghai Pudong I...	Shanghai	China	414
3316	Singapore Changi ...	Singapore	Singapore	412
1218	Barcelona Interna...	Barcelona	Spain	392
3751	Denver Internatio...	Denver	United States	374
3930	Incheon Internati...	Seoul	South Korea	370
3576	Miami Internation...	Miami	United States	366
1701	Atatürk Internati...	Istanbul	Turkey	361

Figure 10.16: Output for top K cities with most incoming Airlines

10.17 The top K cities with most outgoing Airlines

spark.sql("""select * from (select airports.airportid, airports.name, airports.city,airports.country, tb2.outgoing_flight_count from airports inner join (select count (*) as outgoing_flight_count, sourceairportid from routes groupby sourceairportid) tb2 on airports.airportid = tb2.sourceairportid) otb order by otb.outgoing_flight_count desc""").show(100)

Output:

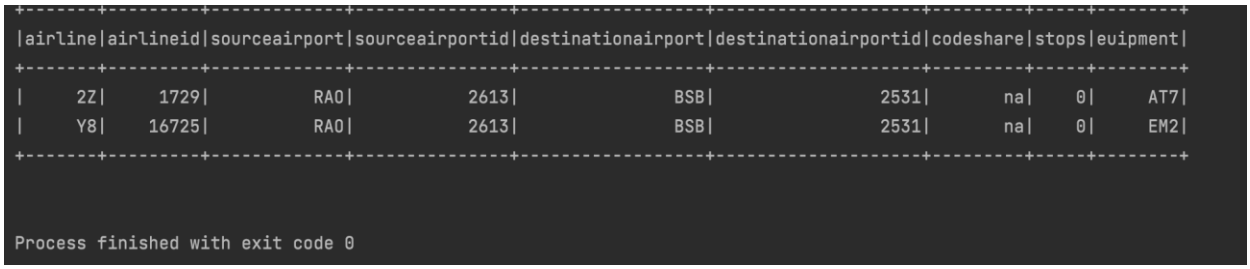
airportid	name	city	country	outgoing_flight_count
3682	Hartsfield Jackso...	Atlanta	United States	915
3830	Chicago O'Hare In...	Chicago	United States	558
3364	Beijing Capital I...	Beijing	China	535
507	London Heathrow A...	London	United Kingdom	527
1382	Charles de Gaulle...	Paris	France	524
340	Frankfurt am Main...	Frankfurt	Germany	497
3484	Los Angeles Inter...	Los Angeles	United States	492
3670	Dallas Fort Worth...	Dallas-Fort Worth	United States	469
3797	John F Kennedy In...	New York	United States	456
580	Amsterdam Airport...	Amsterdam	Netherlands	453
3406	Shanghai Pudong I...	Shanghai	China	411
3316	Singapore Changi ...	Singapore	Singapore	408
1218	Barcelona Interna...	Barcelona	Spain	391

Figure 10.17: Output for top K cities with most outgoing Airlines

10.18 Trip that connects two cities X and Y

spark.sql("""select * from routes where sourceairportid = '2613' and destinationairportid='2531' """).show(100)

Output:



airline	airlineid	sourceairport	sourceairportid	destinationairport	destinationairportid	codeshare	stops	equipment
2Z	1729	RAO	2613	BSB	2531	na	0	AT7
Y8	16725	RAO	2613	BSB	2531	na	0	EM2

Process finished with exit code 0

Figure 10.18: Output for trip that connects two cities X and Y

10.19 Trip that connects X and Y with less than Z stops

```
spark.sql("""select * from routes where sourceairportid = '2613' and
destinationairportid='2531' and stops < 1 """).show(100)
```

Output:

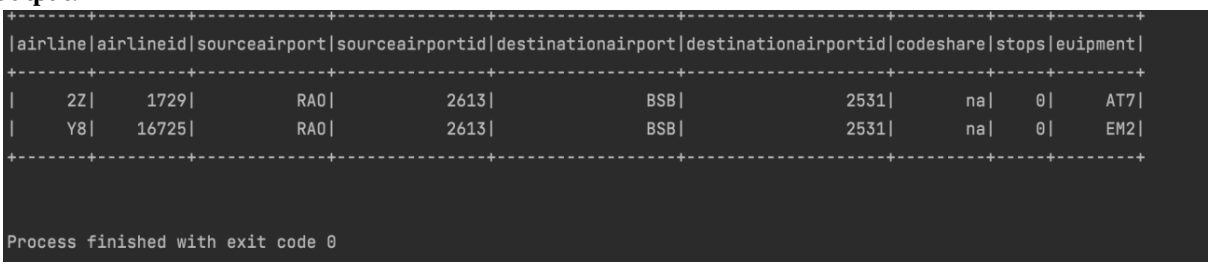


Figure 10.19: Output for trip that connects X and Y with less than Z stops

10.20 *All the cities reachable within d hops of a city*

```
spark.sql("""select destinationairport from routes where stops = 1 """).show(100)
```

Output:

Figure 10.20: Output for all the cities reachable within d hops of a city

11 AWS Output Screenshot

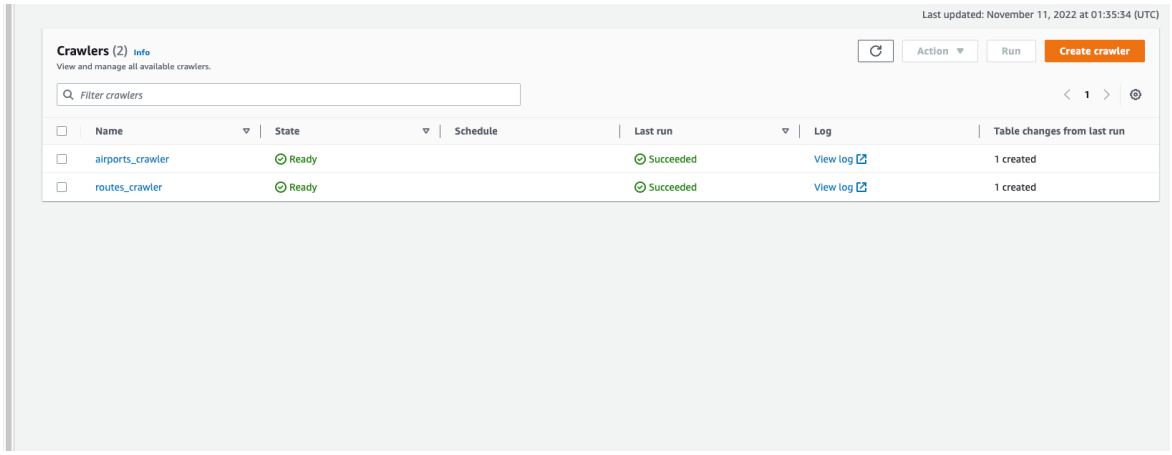


Figure 30: AWS Crawlers page

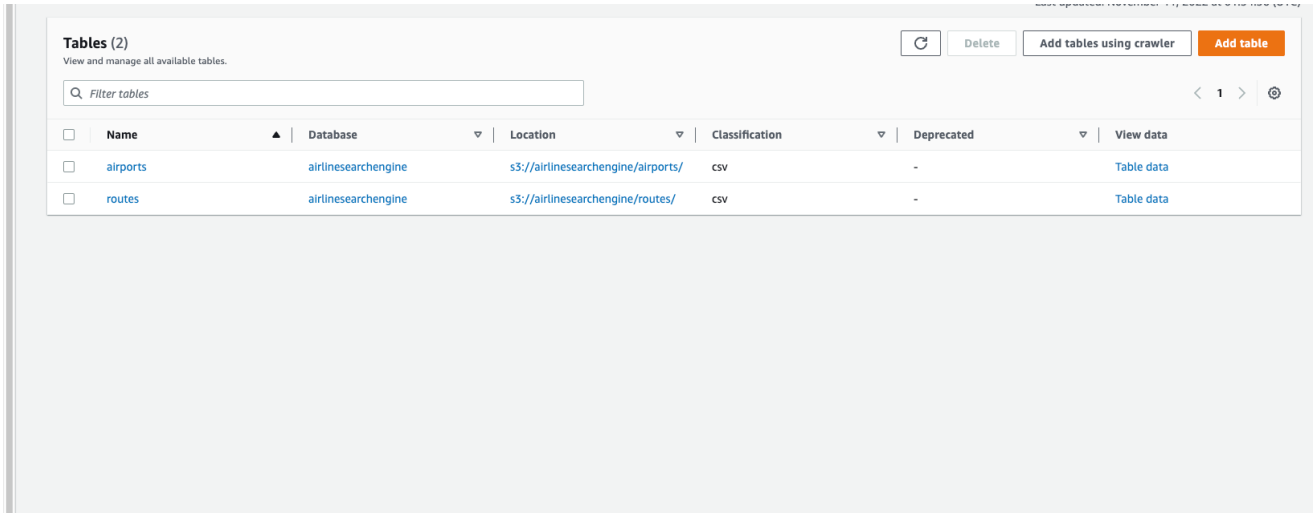


Figure 31: AWS Tables

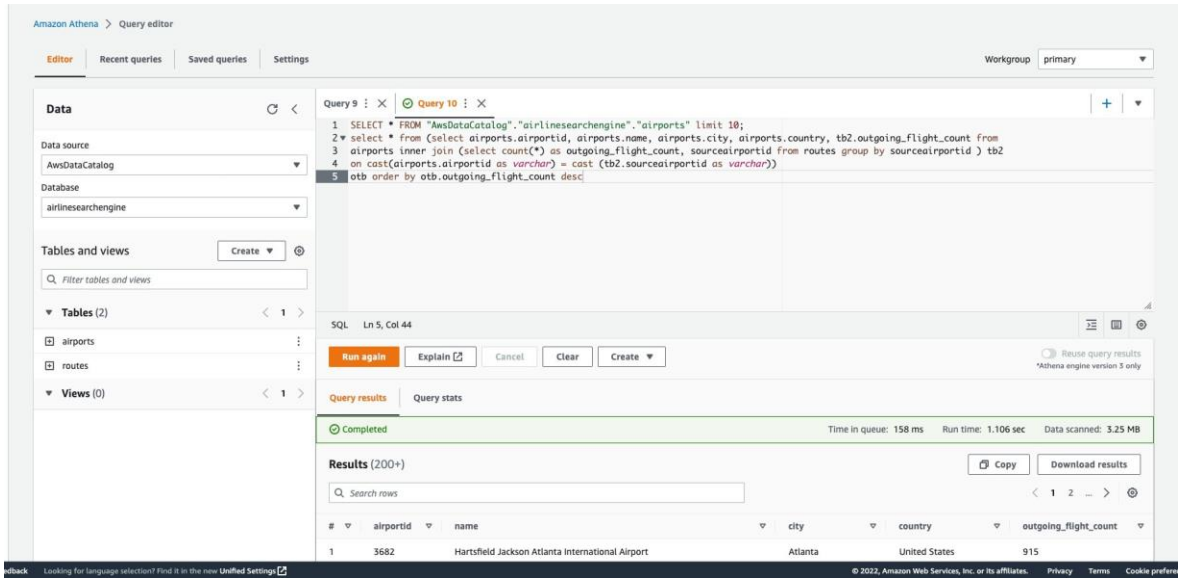


Figure 32: AWS Athena Query

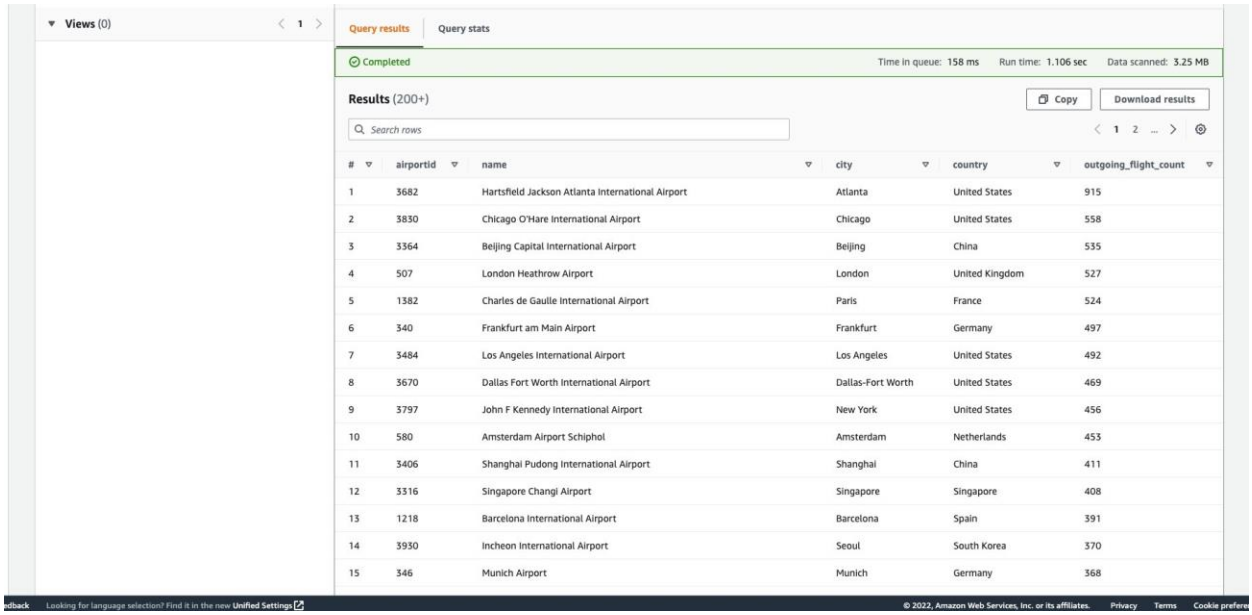


Figure 33: AWS Athena Output

12 Acknowledgement

I would like to thank my major professor, Liu Yunchuan, for having faith in me and my talents and for continuing to believe that I would be able to complete the project on schedule. This Project was completed successfully thanks to the support, ongoing direction, and insightful feedback. I also want to express my sincere gratitude to my mentor for being on my panel, working as my academic advisor, helping me make all the important choices, and having faith in me.

13 References:

[1] http://openflights.org/data.html.
[2] https://docs.aws.amazon.com/glue/latest/ug/tutorial-create-job.html
[3] https://spoddutur.github.io/spark-notes/spark-as-cloud-based-sql-engine-via-thrift-server.html
[4] https://docs.aws.amazon.com/s3/index.html
[5] https://www.iata.org/en/publications/directories/code-search/
[6] https://www.youtube.com/watch?v=8VOf1PUFE0I
[7] https://docs.aws.amazon.com/iam/index.html
[8] https://www.jetbrains.com/pycharm/learn/
[9] https://docs.python.org/3/library/index.html
[10] https://pandas.pydata.org/docs/
[11] https://spark.apache.org/docs/latest/api/python/index.html