# A picture containing text, design Description automatically generated[https://avatars2.githubusercontent.com/u/4156894?v=3&s=100](http://www.calstatela.edu/centers/hipic) CIS 5200 Term Project Tutorial

#### Authors:  [Ragi Dave](https://www.linkedin.com/in/ragi-dave-0ab43720b/), [An Mach](https://www.linkedin.com/in/andrea-mach/), [Ankita Hasmukhbhai Savaliya](https://www.linkedin.com/in/ankita-savaliya-0a4709257/), [Bhumika Suvagia](https://www.linkedin.com/in/bhumika-suvagia-937b16252/)

#### Instructor: Dr. [Jongwook Woo](https://www.linkedin.com/in/jongwook-woo-7081a85)

#### Date: 05/17/2023

**Lab Tutorial**

Ragi Dave ([rdave4@calstatela.edu](mailto:rdave4@calstatela.edu))

An Mach ([amach3@calstatela.edu](mailto:amach3@calstatela.edu))

Ankita Hasmukhbhai Savaliya ([asavali2@calstatela.edu](mailto:asavali2@calstatela.edu))

Bhumika Suvagia ([bsuvagi@calstatela.edu](mailto:bsuvagi@calstatela.edu))

05/17/2023

**Flight Prices Analysis using Hive**

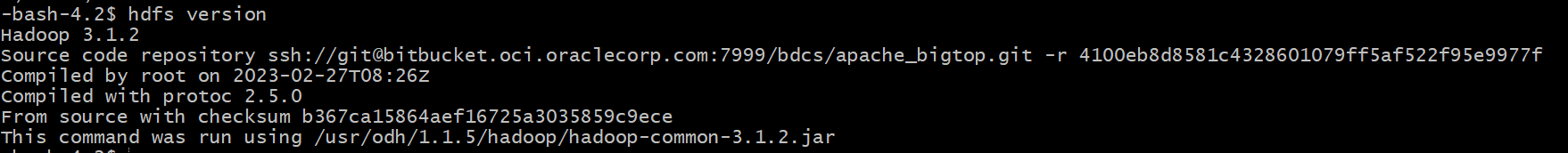
**Objectives**

In this hands-on lab, you will learn how to:

* Upload the data into HDFS using -put command.
* Create Hive tables and perform data engineering in Hive using HiveQL
* Query data using HiveQL
* Secure copy tables created with HiveQL from Hadoop to local computers
* Create visualizations with Excel and Tableau

**Platform Spec**

hdfs version



lscpu

Text

Description automatically generated

hdfs dfsadmin -report

**Text

Description automatically generated**

* Cluster Version: Hadoop 3.1.2
* CPU Speed: 1995.312 MHz
* Number of CPU cores: 8 cores \* 5 nodes = 40 cores
* Number of nodes: 5 (2 Master and 3 Worker)
* Total Memory Size: 390.71 GB

**Dataset Details**

* DATASET NAME: Flight Prices
* DATASET URL:<https://www.kaggle.com/datasets/dilwong/flightprices>

<https://www.kaggle.com/datasets/mike90/airport-codes>

* TOTAL SIZE: 31.09 GB
* COUNTRIES CONSIDERED:USA
* NUMBER OF FILES: 2
* FILE FORMAT: CSV
* GITHUB LINK: <https://github.com/amach3/Flight-Price-Analysis.git>

**Step 1: Download Kaggle dataset in Google Colab**

Instruction was found at <https://www.kaggle.com/general/156610>.

**Note: We realized the steps in this section are not necessary in this project. We could have downloaded the zip file directly from Kaggle then unzip the file on a local computer. You can skip to Step 2 section.**

1. Go to your kaggle account, Scroll to API section and Click Expire API Token to remove previous tokens

2. Click on Create New API Token - It will download kaggle.json file on your machine.

Graphical user interface, text, application, email

Description automatically generated

Graphical user interface, text, application

Description automatically generated

3.Go to your Google Colab project file and run the following commands to mount your Google Drive files Following code make mount your google drive

from google.colab import drive

drive.mount('/content/gdrive')

Graphical user interface, text, website

Description automatically generated

• Now upload the kaggle.json file

from google.colab import files

files.upload() #this will prompt you to upload the kaggle.json

Text

Description automatically generated

• make sure kaggle.json file is present

!ls -lha kaggle.json

Graphical user interface, website

Description automatically generated

• Install kaggle API client

!pip install -q kaggle

Text

Description automatically generated with medium confidence

• kaggle API client expects the file to be in ~/.kaggle

• so move it there

!mkdir -p ~/.kaggle

!cp kaggle.json ~/.kaggle/

Text

Description automatically generated

• we need to set permissions

!chmod 600 /root/.kaggle/kaggle.json

Logo

Description automatically generated

• check your directory before downloading the datasets

!pwd

Graphical user interface, application

Description automatically generated

• list all available datasets

!kaggle datasets list

Text

Description automatically generated

• download the required dataset from kaggle by copying the API command from the dataset page (<https://www.kaggle.com/datasets/dilwong/flightprices?select=itineraries.csv>)

Graphical user interface, text, application, chat or text message

Description automatically generated

!kaggle datasets download -d dilwong/flightprices

Graphical user interface, text, chat or text message

Description automatically generated

Graphical user interface, text, application

Description automatically generated

• check if the zip file is in the current working directory

Text

Description automatically generated with medium confidence

* Copy the zip file to Google Drive

!cp flightprices.zip /content/gdrive/MyDrive

Text

Description automatically generated with medium confidence

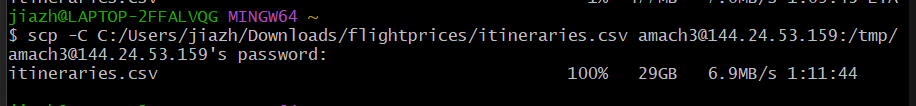
Graphical user interface, text, application, chat or text message

Description automatically generated

**Step 2: Secure copy csv files from local computer to Linux serve**

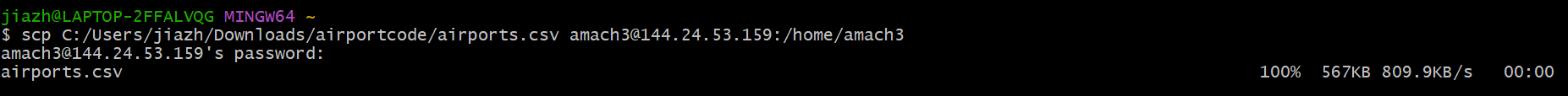
* Download the flightprices.zip file from Google Drive then unzip it in Download folder, then secure copy it to tmp folder on Linux server.

scp -C C:/Users/jiazh/Downloads/flightprices/itineraries.csv [amach3@144.24.53.159:/tmp](mailto:amach3@144.24.53.159:/tmp)



* Download the airport.csv file from Kaggle and unzip it in Download folder. Then secure copy it to linux.

scp C:/Users/jiazh/Downloads/airportcode/airports.csv amach3@144.24.53.159:/home/amach3



**Step 3: Copy files from Linux server to HDFS**

* Make FlightData directory and put itineraries.csv file here

hdfs dfs -mkdir FlightData

hdfs dfs -put /tmp/itineraries.csv /user/amach3/FlightData/

hdfs dfs -ls FlightData/



* Make the FlightData directory public

hdfs dfs -chmod -R go+rx /user/amach3/FlightData

* Make Airport directory and put Airport.csv file here

hdfs dfs -mkdir Airport

hdfs dfs -put airports.csv Airport/

hdfs dfs -ls Airport/

Text

Description automatically generated

* make the Airport directory public

hdfs dfs -chmod -R go+rx /user/amach3/Airport

**Step 4: Create tables in Hive**

1. Airport table

Beeline

Use amach3;

DROP TABLE IF EXISTS Airport;

CREATE EXTERNAL TABLE IF NOT EXISTS Airport

(Name string, City string, Country string, IATA string, ICAO string,

Latitude double, Longitude double)

ROW FORMAT DELIMITED FIELDS TERMINATED BY ','

STORED AS TEXTFILE LOCATION '/user/amach3/Airport/'

TBLPROPERTIES ('skip.header.line.count'='1');

SELECT \* FROM Airport LIMIT 10;

A screenshot of a computer

Description automatically generated with medium confidence

1. FlightData table

DROP TABLE IF EXISTS FlightData;

CREATE EXTERNAL TABLE IF NOT EXISTS FlightData

(FlightID string, SearchDate date, FlightDate date, StartingAirport string,

DestinationAirport string, FareBasisCode string, TravelDuration string,

ElapsedDays int, IsBasicEconomy boolean, IsRefundable boolean,

IsNonStop boolean, BaseFare double, TotalFare double, SeatsRemaining int,

TotalTravelDistance double, SegmentsDepartureTimeEpochSeconds string,

SegmentsDepartureTimeRaw string, SegmentsArrivalTimeEpochSeconds string,

SegmentsArrivalTimeRaw string, SegmentsArrivalAirportCode string,

SegmentsDepartureAirportCode string, SegmentsAirlineName string,

SegmentsAirlineCode string, SegmentsEquipmentDescription string,

SegmentsDurationInSeconds string, SegmentsDistance double,

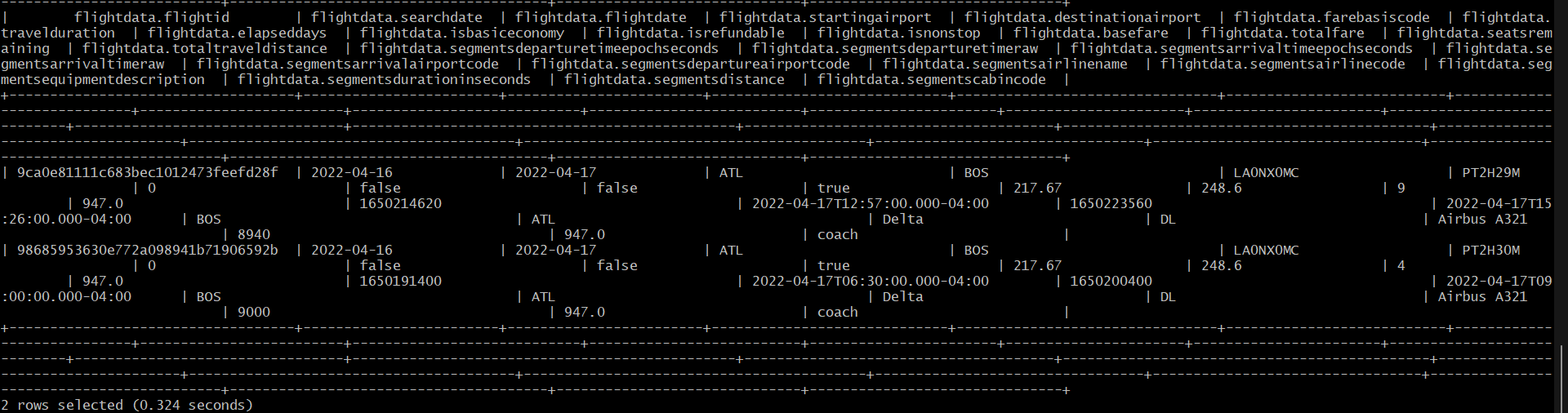
SegmentsCabinCode string)

ROW FORMAT DELIMITED FIELDS TERMINATED BY ','

STORED AS TEXTFILE LOCATION '/user/amach3/FlightData/'

TBLPROPERTIES ('skip.header.line.count'='1');

SELECT \* FROM FlightData LIMIT 2;



Since the dataset is too large, Dr. Woo advised us to reduce our data size to 2 – 3 GB by sampling our data.

SELECT COUNT(\*) FROM FlightData; --82138753 rows

--10% is about 8,000,000 rows

CREATE EXTERNAL TABLE IF NOT EXISTS FlightData2

(FlightID string, SearchDate date, FlightDate date, StartingAirport string,

DestinationAirport string, FareBasisCode string, TravelDuration string,

ElapsedDays int, IsBasicEconomy boolean, IsRefundable boolean,

IsNonStop boolean, BaseFare double, TotalFare double, SeatsRemaining int,

TotalTravelDistance int, SegmentsDepartureTimeEpochSeconds string,

SegmentsDepartureTimeRaw string, SegmentsArrivalTimeEpochSeconds string,

SegmentsArrivalTimeRaw string, SegmentsArrivalAirportCode string,

SegmentsDepartureAirportCode string, SegmentsAirlineName string,

SegmentsAirlineCode string, SegmentsEquipmentDescription string,

SegmentsDurationInSeconds string, SegmentsDistance int,

SegmentsCabinCode string)

ROW FORMAT DELIMITED FIELDS TERMINATED BY ','

STORED AS TEXTFILE LOCATION '/user/amach3/FlightData2/'

TBLPROPERTIES ('skip.header.line.count'='1');

INSERT OVERWRITE TABLE FlightData2

SELECT \* FROM FlightData

WHERE rand() <= 0.1

distribute by rand()

SORT BY rand()

LIMIT 8000000;

SELECT COUNT(\*) FROM FlightData2;

Graphical user interface, application

Description automatically generated

**Step 5:**  **Data Engineering**

1. Drop unusable columns

ALTER TABLE FlightData2 REPLACE COLUMNS (FlightID string, SearchDate date, FlightDate date, StartingAirport string,

DestinationAirport string, FareBasisCode string, TravelDuration string,

ElapsedDays int, IsBasicEconomy boolean, IsRefundable boolean,

IsNonStop boolean, BaseFare double, TotalFare double, SeatsRemaining int,

TotalTravelDistance int, SegmentsAirlineName string, SegmentsEquipmentDescription string);

SELECT \* FROM FlightData2 LIMIT 2;

Graphical user interface

Description automatically generated with medium confidence

1. Remove any duplicate rows from FlightData2 table

INSERT OVERWRITE TABLE FlightData2

SELECT DISTINCT \* FROM FlightData2;

-- Check number of rows after removing duplicates

SELECT COUNT(\*) FROM FlightData2;

A picture containing text

Description automatically generated

1. Check null values

SELECT COUNT(\*) FROM FlightData2 where TotalTravelDistance IS NULL;

A picture containing text

Description automatically generated

SELECT count(\*) from FlightData2 where FlightDate IS NULL;

A picture containing text

Description automatically generated

SELECT count(\*) from FlightData2 where StartingAirport IS NULL;

A picture containing graphical user interface

Description automatically generated

SELECT count(\*) from FlightData2 where DestinationAirport IS NULL;

A picture containing text

Description automatically generated

SELECT count(\*) from FlightData2 where TravelDuration IS NULL;

A picture containing text

Description automatically generated

SELECT count(\*) from FlightData2 where IsBasicEconomy IS NULL;

A picture containing text

Description automatically generated

SELECT count(\*) from FlightData2 where IsNonStop IS NULL;

A picture containing graphical user interface

Description automatically generated

SELECT count(\*) from FlightData2 where BaseFare IS NULL;

A picture containing text

Description automatically generated

SELECT count(\*) from FlightData2 where TotalFare IS NULL;

A picture containing graphical user interface

Description automatically generated

SELECT count(\*) from FlightData2 where SeatsRemaining IS NULL;

Text

Description automatically generated

SELECT count(\*) from FlightData2 where SegmentsAirlineName IS NULL;

Text

Description automatically generated

SELECT count(\*) from FlightData2 where SegmentsEquipmentDescription IS NULL;

A picture containing text

Description automatically generated

Since our dataset is huge (almost 8 million rows of data), we decided to remove rows with null values detected above.

INSERT OVERWRITE TABLE FlightData2

SELECT \* FROM FlightData2 WHERE TotalTravelDistance IS NOT NULL;

INSERT OVERWRITE TABLE FlightData2

SELECT \* FROM FlightData2 WHERE FlightDate IS NOT NULL;

INSERT OVERWRITE TABLE FlightData2

SELECT \* FROM FlightData2 WHERE StartingAirport IS NOT NULL;

INSERT OVERWRITE TABLE FlightData2

SELECT \* FROM FlightData2 WHERE DestinationAirport IS NOT NULL;

INSERT OVERWRITE TABLE FlightData2

SELECT \* FROM FlightData2 WHERE TravelDuration IS NOT NULL;

INSERT OVERWRITE TABLE FlightData2

SELECT \* FROM FlightData2 WHERE IsBasicEconomy IS NOT NULL;

INSERT OVERWRITE TABLE FlightData2

SELECT \* FROM FlightData2 WHERE IsNonStop IS NOT NULL;

INSERT OVERWRITE TABLE FlightData2

SELECT \* FROM FlightData2 WHERE BaseFare IS NOT NULL;

INSERT OVERWRITE TABLE FlightData2

SELECT \* FROM FlightData2 WHERE TotalFare IS NOT NULL;

INSERT OVERWRITE TABLE FlightData2

SELECT \* FROM FlightData2 WHERE SeatsRemaining IS NOT NULL;

INSERT OVERWRITE TABLE FlightData2

SELECT \* FROM FlightData2 WHERE SegmentsAirlineName IS NOT NULL;

INSERT OVERWRITE TABLE FlightData2

SELECT \* FROM FlightData2 WHERE SegmentsEquipmentDescription IS NOT NULL;

SELECT COUNT(\*) from FlightData2;

A picture containing graphical user interface

Description automatically generated

1. Add FlightMonth column by getting the month from FlightDate column

ALTER TABLE FlightData2 ADD COLUMNS (FlightMonth int);

INSERT OVERWRITE TABLE FlightData2

SELECT FlightID, SearchDate, FlightDate, StartingAirport,

DestinationAirport, FareBasisCode string, TravelDuration,

ElapsedDays, IsBasicEconomy, IsRefundable,

IsNonStop, BaseFare, TotalFare, SeatsRemaining,

TotalTravelDistance, SegmentsAirlineName , SegmentsEquipmentDescription,

MONTH(FlightDate) AS FlightMonth FROM FlightData2;

SELECT FlightDate, FlightMonth FROM FlightData2 LIMIT 10;

A picture containing shape

Description automatically generated

1. Add a FlightRoute column that concatenate startingAirport with destinationAirport columns

ALTER TABLE FlightData2 ADD COLUMNS (FlightRoute string);

INSERT OVERWRITE TABLE FlightData2

SELECT FlightID, SearchDate, FlightDate, StartingAirport,

DestinationAirport, FareBasisCode string, TravelDuration,

ElapsedDays, IsBasicEconomy, IsRefundable,

IsNonStop, BaseFare, TotalFare, SeatsRemaining,

TotalTravelDistance, SegmentsAirlineName , SegmentsEquipmentDescription, FlightMonth,

CONCAT(StartingAirport, '-', DestinationAirport) AS FlightRoute

FROM FlightData2;

SELECT StartingAirport, DestinationAirport, FlightRoute FROM FlightData2 limit 10;

A picture containing diagram

Description automatically generated

**Step 6:** **Write analysis queries in Hive**

**Analysis 1: Top 10 most popular flight routes**

The analysis is conducted by following these steps:

* Select the flightRoute column from the FlightData2 table.
* Count the FlightID and assign the alias "FlightCount" to the count column.
* Group the FlightRoute column to group the flight data by unique flight routes.
* Sort the flightCount in descending order.
* Limit the results to the top 10 flight routes.
* Insert the resulting data into the "FlightRoute" subdirectory of "FlightData2".

INSERT OVERWRITE DIRECTORY '/user/amach3/FlightData2/FlightRoute/'

ROW FORMAT DELIMITED FIELDS TERMINATED BY ','

SELECT FlightRoute, COUNT(FlightID) AS FlightCount

FROM FlightData2

GROUP BY FlightRoute

ORDER BY FlightCount DESC

LIMIT 10;

By following the above steps, we got the top 10 most popular flight routes in the Unites States between April to November 2022 and its count same as below.

A picture containing diagram

Description automatically generated

**Analysis 2: Least 10 popular flight routes**

The analysis is conducted by following these steps:

* Select the flightRoute column from the FlightData2 table.
* Count the FlightID and assign the alias "FlightCount" to the count column.
* Group the FlightRoute column to group the flight data by unique flight routes.
* Sort the flightCount in ascending order.
* Limit the results to the top 10 flight routes.
* Insert the resulting data into the " LeastPopularRoute" subdirectory of "FlightData2".

INSERT OVERWRITE DIRECTORY '/user/amach3/FlightData2/LeastPopularRoute/'

ROW FORMAT DELIMITED FIELDS TERMINATED BY ','

SELECT FlightRoute, COUNT(FlightID) AS FlightCount

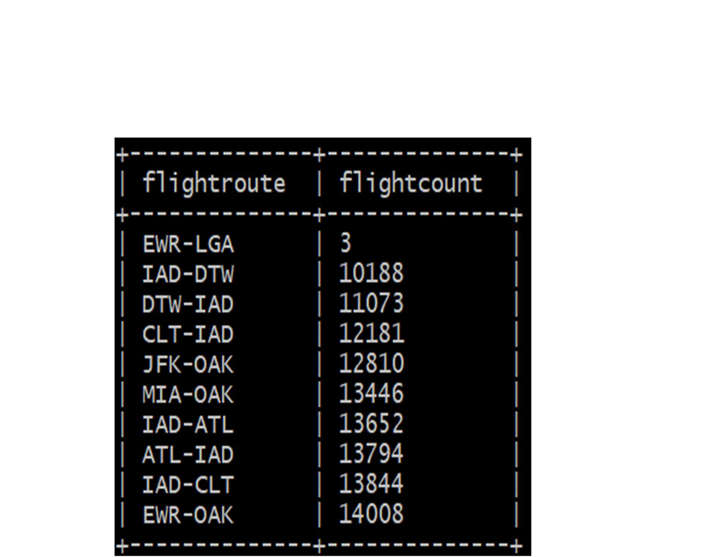
FROM FlightData2

GROUP BY FlightRoute

ORDER BY FlightCount ASC

LIMIT 10;

By following the above steps, we got the least 10 popular flight routes in the United States same as below.



**Analysis 3: Top 15 Most Expensive Routes:**

**-**Below mentioned query is for calculate and write the average flight price for top 15 flight routes in the FlightData2 table to a directory in HDFS.

**-** This is the SELECT statement that calculates the average flight price for each flight route in the FlightData2 table.

**-T**he ROUND() function is used to round the average price to two decimal places.

-The results are the group by flight route sorted in descending order by flight price,limited to top 15 results.

Top 15 most expensive routes

INSERT OVERWRITE DIRECTORY '/user/amach3/FlightData2/FlightPrice/'

ROW FORMAT DELIMITED FIELDS TERMINATED BY ','

SELECT FlightRoute, ROUND(AVG(TotalFare),2) AS FlightPrice

FROM FlightData2

GROUP BY FlightRoute

ORDER BY FlightPrice DESC LIMIT 15;

A picture containing diagram

Description automatically generated

**Analysis 4: Average Airfare and Average Travel Distance over a Month in 2022**

To carry out the mentioned analysis, the following steps were performed:

* Calculate the average of the TotalFare and Travel Distance data for each month using the AVG() function.
* Round the calculated averages to two decimal places using the ROUND() function.
* Take the resulting data, including the month, average TotalFare, and average Travel Distance.
* Insert the resulting data into subdirectory "PriceDistanceOverTime" of FlightData2.

By following these steps, the "PriceDistanceOverTime" contains information about the month, average TotalFare, and average Travel Distance for the peroid between March to December 2022.

INSERT OVERWRITE DIRECTORY '/user/amach3/FlightData2/PriceDistanceOverTime/'

ROW FORMAT DELIMITED FIELDS TERMINATED BY ','

SELECT FlightMonth, ROUND(AVG(TotalFare),2) AS AverageAirfare, ROUND(AVG(TotalTravelDistance),2) AS AverageTravelDistance

FROM FlightData2

GROUP BY FlightMonth;

Execute the following command to confirm :

SELECT FlightMonth, ROUND(AVG(TotalFare),2) AS AverageAirfare, ROUND(AVG(TotalTravelDistance),2) AS AverageTravelDistance

FROM FlightData2

GROUP BY FlightMonth;

A picture containing text

Description automatically generated

**Analysis 5: Average Airfare by Days per Month**

The following steps were performed:

* Calculate the average of the TotalFare and Travel Distance data for each month using the AVG() function.
* These averages were then rounded to two decimal places using the ROUND() function.
* Additionally, the FlightDate column was utilized to determine the weeks and days of the flights.
* The resulting data consisted of the FlightDate, average TotalFare, and average Travel Distance.
* Insert the resulting data into subdirectory "Days" of FlightData2.

As a result, the "Days" subdirectory now contains information about the month, average TotalFare, and average Travel Distance for the period spanning from April to November 2022.

INSERT OVERWRITE DIRECTORY '/user/amach3/FlightData2/Days/'

ROW FORMAT DELIMITED FIELDS TERMINATED BY ','

SELECT FlightDate, ROUND(AVG(TotalFare),2) AS AverageAirfare, ROUND(AVG(TotalTravelDistance),2) AS AverageTravelDistance

FROM FlightData2

GROUP BY FlightDate;

Execute the following command to confirm :

SELECT FlightDate, ROUND(AVG(TotalFare),2) AS AverageAirfare, ROUND(AVG(TotalTravelDistance),2) AS AverageTravelDistance

FROM FlightData2

GROUP BY FlightDate;

A screenshot of a computer screen

Description automatically generated with low confidence

A screen shot of a computer

Description automatically generated with low confidence

A picture containing text, screenshot, book, symmetry

Description automatically generated

A screen shot of a computer

Description automatically generated with low confidence

**Analysis 6: Most popular destinations over a month in 2022**

For this analysis, we joined two tables 1)FlightData2 and 2)Airport.

* The FlightData2 table was used to obtain information such as the Destination airport, FlightDate, and FlightID. The FlightDate column was formatted using the date\_format() function to represent the month as 'MM-01-yyyy' format. This formatting was done to calculate the total number of flights per month, ensuring that the month portion remains consistent with a value of 01 rather than representing different dates within the month.
* The Airport table was utilized to retrieve additional data, specifically the Latitude and Longitude values associated with each Destination airport.
* To combine the information from both tables, a JOIN operation was performed based on the common field DestinationAirport in the FlightData2 table and the IATA column in the Airport table. These columns contain a 3-character airport code that facilitates the matching of records.
* The resulting data contains the following fields: DestinationAirport, formatted\_date, Latitude, Longitude, and FlightCount.
* Insert the resulting data into subdirectory " Destination" of FlightData2.

As a result, the " Destination" subdirectory now contains information about DestinationAirport, formatted\_date, Latitude, Longitude, and FlightCount from April to November 2022.

INSERT OVERWRITE DIRECTORY '/user/amach3/FlightData2/Destination/'

ROW FORMAT DELIMITED FIELDS TERMINATED BY ','

SELECT DestinationAirport, date\_format(FlightDate,'MM-01-yyyy') AS formatted\_date,Latitude, Longitude, COUNT(FlightID) AS FlightCount

FROM FlightData2 F

JOIN Airport A

ON F.DestinationAirport = A.IATA

GROUP BY DestinationAirport,date\_format(FlightDate,'MM-01-yyyy'), Latitude, Longitude;

To see the result:

SELECT DestinationAirport, date\_format(FlightDate,'MM-01-yyyy') AS formatted\_date,Latitude, Longitude, COUNT(FlightID) AS FlightCount

FROM FlightData2 F

JOIN Airport A

ON F.DestinationAirport = A.IATA

GROUP BY DestinationAirport,date\_format(FlightDate,'MM-01-yyyy'), Latitude, Longitude;

A screen shot of a computer screen

Description automatically generated with low confidence

A screen shot of a computer screen

Description automatically generated with low confidence

**Analysis 7: Distribution of Basic Economy tickets**

The following steps performed:

* In FlightData2,isBasicEconomy field contains Boolean value such as ‘Y’ and ‘N’. Bu using COUNT() function, it will calculate the total number of purchased tickets and saved as a count.
* The query groups the data by the isBasicEconomy column, which separates the records based on whether they are classified as basic economy or not.
* The resulting data contains isBasicEconomy and count.
* Insert the resulting data into subdirectory "BasicEconomy" of FlightData2.

As a result, the "BasicEconomy " subdirectory now contains information about isBasicEconomy and count.

INSERT OVERWRITE DIRECTORY '/user/amach3/FlightData2/BasicEconomy/'

ROW FORMAT DELIMITED FIELDS TERMINATED BY ','

SELECT isBasicEconomy, COUNT(\*) as count

FROM FlightData2

GROUP BY isBasicEconomy;

A picture containing diagram

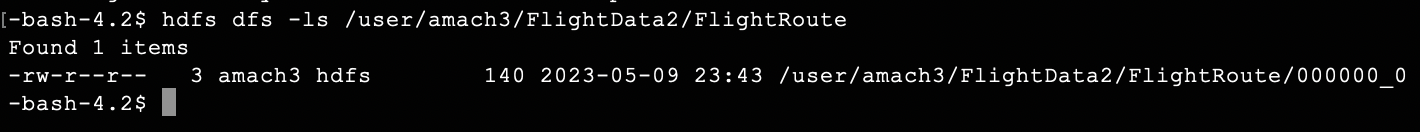
Description automatically generated

**Step 7: Export query results to HDFS, secure copy text files to local computer**

1. **Top 10 most popular flight routes (FlightRoute.txt)**

Open another terminal with git bash, minty, or putty, which is to connect the HADOOP CLUSTER to download the output file 000000\_0 at the HDFS path “/user/amach3/FlightData2/FlightRoute”:

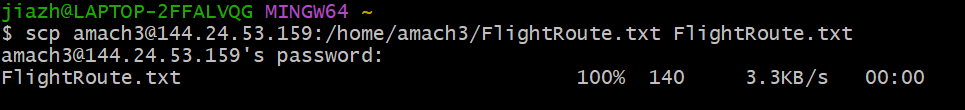
hdfs dfs -ls /user/amach3/FlightData2/FlightRoute

****

Download the file to the local file systems:

hdfs dfs -get /user/amach3/FlightData2/FlightRoute/000000\_0 FlightRoute.txt

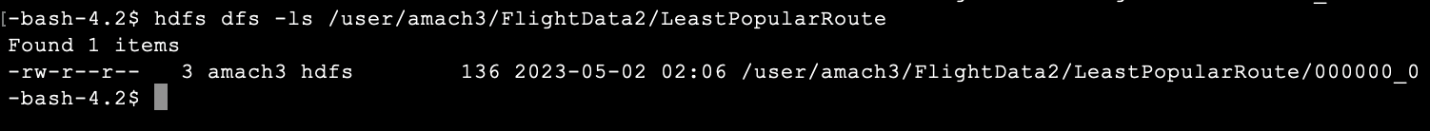
scp [amach3@144.24.53.159:/home/amach3/FlightRoute.txt FlightRoute.txt](mailto:amach3@144.24.53.159:/home/amach3/FlightRoute.txt%20FlightRoute.txt)



1. **Least 10 popular flight routes (LeastPopularRoute.txt)**

Download the output file 000000\_0 at the HDFS path “/user/amach3/FlightData2/LeastPopularRoute”

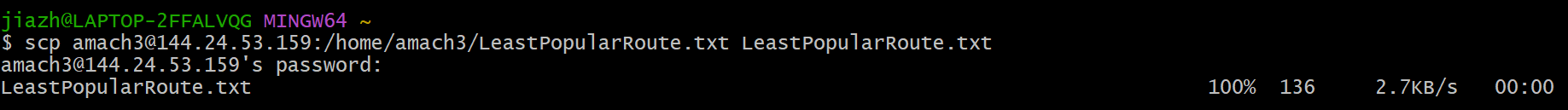
hdfs dfs -ls /user/amach3/FlightData2/LeastPopularRoute



Download the file to the local file systems:

hdfs dfs -get /user/amach3/FlightData2/LeastPopularRoute/000000\_0 LeastPopularRoute.txt

scp [amach3@144.24.53.159:/home/amach3/LeastPopularRoute.txt LeastPopularRoute.txt](amach3@144.24.53.159:/home/amach3/LeastPopularRoute.txt%20LeastPopularRoute.txt)

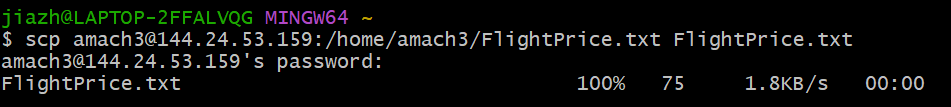


1. **Top 15 most expensive routes (FlightPrice.txt)**

Download the output file “000000\_0” to “FlightPrice.txt” using the following hdfs command:

hdfs dfs -get /user/amach3/FlightData2/FlightPrice/000000\_0 FlightPrice.txt

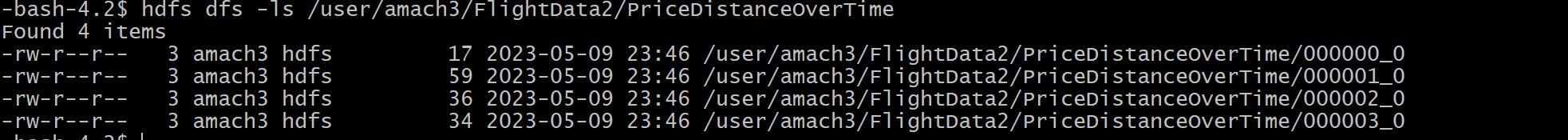
scp amach3@144.24.53.159:/home/amach3/FlightPrice.txt FlightPrice.txt



**4. Airfare and Travel Distance over time (PriceDistanceOverTime.txt)**

* This command lists all the files inside the subdirectory "PriceDistanceOverTime" in HDFS.

hdfs dfs -ls /user/amach3/FlightData2/PriceDistanceOverTime



* Download the output files using -get command to the local file systems and rename them as PriceDistanceOverTime.txt.
* For these steps, we need to concatenate the files into the local Linux file system for downloading in step.

hdfs dfs -get /user/amach3/FlightData2/PriceDistanceOverTime/000000\_0

hdfs dfs -get /user/amach3/FlightData2/PriceDistanceOverTime/000001\_0

hdfs dfs -get /user/amach3/FlightData2/PriceDistanceOverTime/000002\_0

hdfs dfs -get /user/amach3/FlightData2/PriceDistanceOverTime/000003\_0

cat 000000\_0 000002\_0 000003\_0 000001\_0 > PriceDistanceOverTime.txt

* Open another terminal with git bash in order to import the output file using your lab computer (or your PC/Laptop) - you have to download the file to your lab computer (or your PC/Laptop). For example, your output file at the oracle cloud server is located at /home/amach3/PriceDistanceOverTime.txt. And, remotely copied to the file “PriceDistanceOverTime.txt”.
* You will be prompted for your credentials. Provide your password.

scp amach3@144.24.53.159:/home/amach3/ PriceDistanceOverTime.txt PriceDistanceOverTime.txt

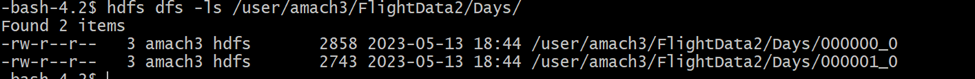
Text

Description automatically generated

**5**. **Average Airfare by Days per Month (Days.txt)**

* This command lists all the files inside the subdirectory "Days" in HDFS.

hdfs dfs -ls /user/amach3/FlightData2/Days/

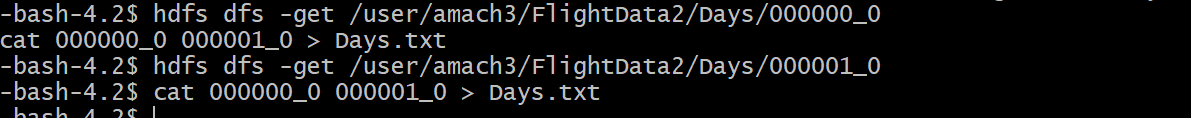


* Download the output files to the local file systems and rename them as Days.txt.
* For these steps, we need to concatenate the csv files into the local Linux file system for downloading in step.

hdfs dfs -get /user/amach3/FlightData2/Days/000000\_0

hdfs dfs -get /user/amach3/FlightData2/Days/000001\_0

cat 000000\_0 000001\_0 > Days.txt



* Open another terminal with git bash in order to import the output file using your lab computer (or your PC/Laptop) - you have to download the file to your lab computer (or your PC/Laptop). For example, your output file at the oracle cloud server is located at /home/amach3/ Days.txt. And, remotely copied to the file “Days.txt”.
* You will be prompted for your credentials. Provide your password.

scp amach3@144.24.53.159:/home/amach3/Days.txt Days.txt

A black background with white text

Description automatically generated with low confidence

**6. Most popular destinations over a month in 2022(Destination.txt)**

* This command lists all the files inside the subdirectory "Destination" in HDFS.

hdfs dfs -ls /user/amach3/FlightData2/Destination/

Text

Description automatically generated with medium confidence

* Download the output files to the local file systems and rename them as Destination.txt.
* For these steps, we need to concatenate the csv files into the local Linux file system for downloading in step.

hdfs dfs -get /user/amach3/FlightData2/Destination/000000\_0

hdfs dfs -get /user/amach3/FlightData2/Destination/000001\_0

hdfs dfs -get /user/amach3/FlightData2/Destination/000002\_0

hdfs dfs -get /user/amach3/FlightData2/Destination/000003\_0

hdfs dfs -get /user/amach3/FlightData2/Destination/000004\_0

hdfs dfs -get /user/amach3/FlightData2/Destination/000005\_0

hdfs dfs -get /user/amach3/FlightData2/Destination/000006\_0

hdfs dfs -get /user/amach3/FlightData2/Destination/000007\_0

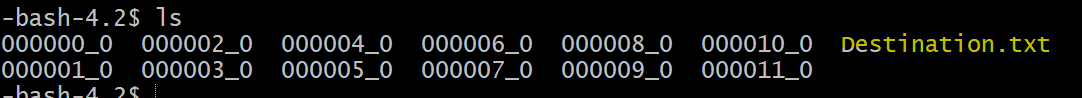
hdfs dfs -get /user/amach3/FlightData2/Destination/000008\_0

hdfs dfs -get /user/amach3/FlightData2/Destination/000009\_0

hdfs dfs -get /user/amach3/FlightData2/Destination/000010\_0

hdfs dfs -get /user/amach3/FlightData2/Destination/000011\_0

cat 000000\_0 000002\_0 000003\_0 000001\_0 000004\_0 000005\_0 000006\_0 000007\_0 000008\_0 000009\_0 000010\_0 000011\_0 > Destination.txt



* Open another terminal with git bash in order to import the output file using your lab computer (or your PC/Laptop) - you have to download the file to your lab computer (or your PC/Laptop). For example, your output file at the oracle cloud server is located at /home/amach3/ Destination.txt. And, remotely copied to the file “Destination.txt”.
* You will be prompted for your credentials. Provide your password.

scp amach3@144.24.53.159:/home/amach3/Destination.txt Destination.txt

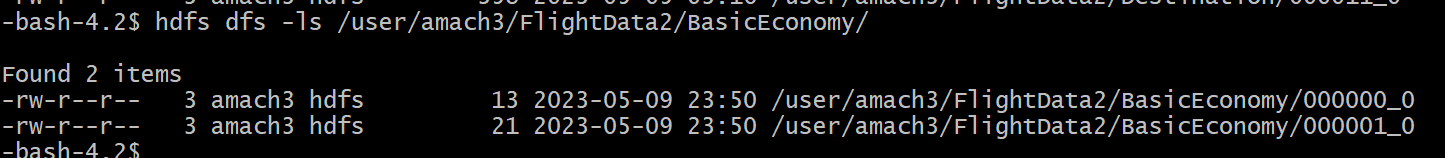
Text

Description automatically generated with medium confidence

1. **Distribution of basic economy tickets (BasicEconomy.txt)**

* This command lists all the files inside the subdirectory "BasicEconomy" in HDFS.

hdfs dfs -ls /user/amach3/FlightData2/BasicEconomy/



* Download the output files to the local file systems and rename them as BasicEconomy.txt.
* For these steps, we need to concatenate the csv files into the local Linux file system for downloading in step.

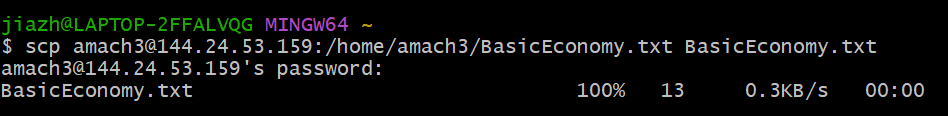
hdfs dfs -get /user/amach3/FlightData2/BasicEconomy/000000\_0

hdfs dfs -get /user/amach3/FlightData2/BasicEconomy/000001\_0

cat 000000\_0 000001\_0 > BasicEconomy.txt

* Open another terminal with git bash in order to import the output file using your lab computer (or your PC/Laptop) - you have to download the file to your lab computer (or your PC/Laptop). For example, your output file at the oracle cloud server is located at /home/amach3/ BasicEconomy.txt. And, remotely copied to the file “BasicEconomy.txt”.
* You will be prompted for your credentials. Provide your password.

scp amach3@144.24.53.159:/home/amach3/BasicEconomy.txt BasicEconomy.txt



**Step 8: Create visualizations.**

**1. Top 10 most popular flight routes (Bar chart)**

Open FlightRoute.txt in the Microsoft excel and insert the column name FlightRoutes in first column and FlightCount for the second column and save as FlightRoute.xlsx.

A screenshot of a spreadsheet

Description automatically generated

Open your Tableau to connect your server. You need to select Microsoft Excel File to open the file FlightRoute.xlsx file.

**A screenshot of a computer

Description automatically generated with medium confidence**

You will see the following data at data source.

**A screenshot of a computer

Description automatically generated**

Select Sheet 1 next to Data Source, which will present the following frame. Then drag the FlightCount to **Columns**, and FlightRoutes to **Rows**, which shows top 10 most popular flight routes in the USA.

Graphical user interface, application

Description automatically generated

A picture containing bar chart

Description automatically generated

**2. Least 10 popular flight routes (bar chart)**

Open the LeastPopularRoute.txt in the Microsoft excel and insert the column name Flight Route in first column and Number of Flights for the second column and save as LeastPopularRoute.xlsx

Open your Tableau to connect your server. You need to select Microsoft Excel File to open the file LeastPopularRoute.xlsx file same as above.

Select Sheet 1 next to Data Source, which will present the following frame. Then drag the Number of Flights to **Columns**, and Flight Route to **Rows**, which shows least 10 most popular flight routes in the USA.

Chart

Description automatically generated

Chart, bar chart

Description automatically generated

1. Top 15 most expensive routes (Heat map)

-To show this Analysis we have choose to create a heat map in Tableau.

First step, we decide to show Airfare in each rectangle, we choose Airfare to color and then other two measure are in Text and in Size.

Reason to add those two measures in text is to show flight routes and flight fares clearly in the given rectangle

We have choose the color shade dark pink to highlight the most expensive flight route.

Chart, treemap chart

Description automatically generated

Chart, treemap chart

Description automatically generated

1. Airfare and Travel Distance over time (dual axis chart in Tableau)

Because the measures have been aggregated using HiveQL, we do not need to aggregate data within Tableau. Go to **Analysis** tab and uncheck **Aggregate Measures**

A screenshot of a computer

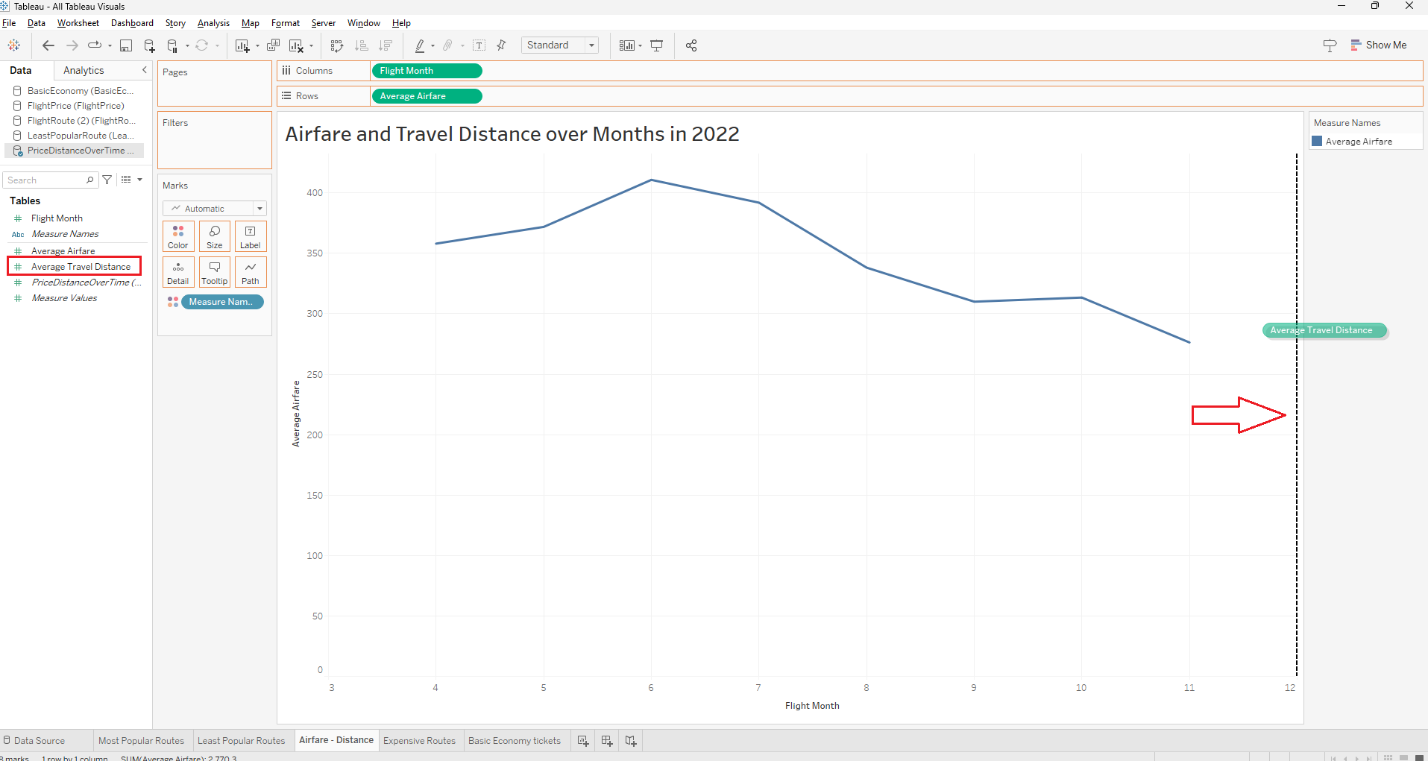
Description automatically generated with medium confidence

Drag Flight Month to **Columns**

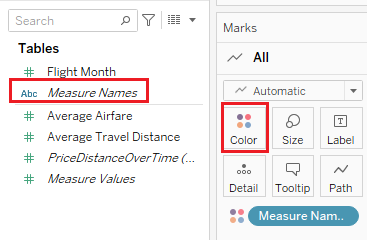
Drag Average Airfare to **Rows**

A picture containing text, screenshot, plot, line

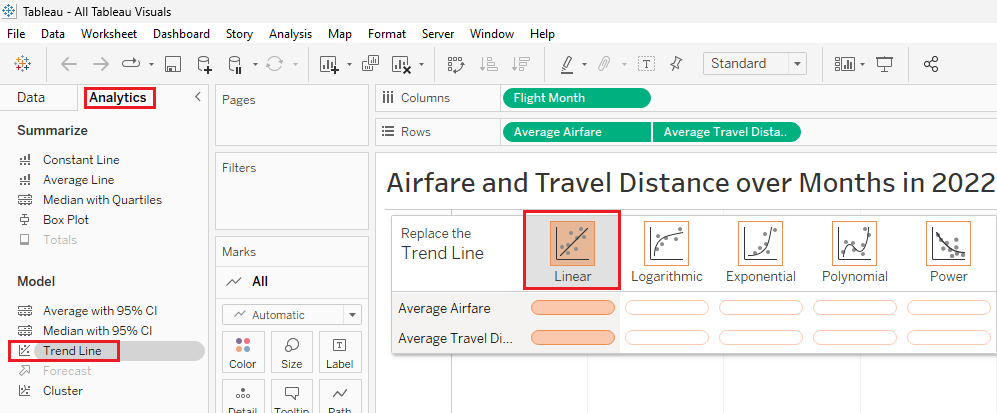
Description automatically generated



Drag Measure Names to **Color**



Click on **Analytics** tab, drag **Trend Line** to **Linear** type to add the dotted trend line to line chart.



Chart, line chart

Description automatically generated

Chart, line chart

Description automatically generated

1. Average Airfare by Days per Month (bar chart in Power BI)

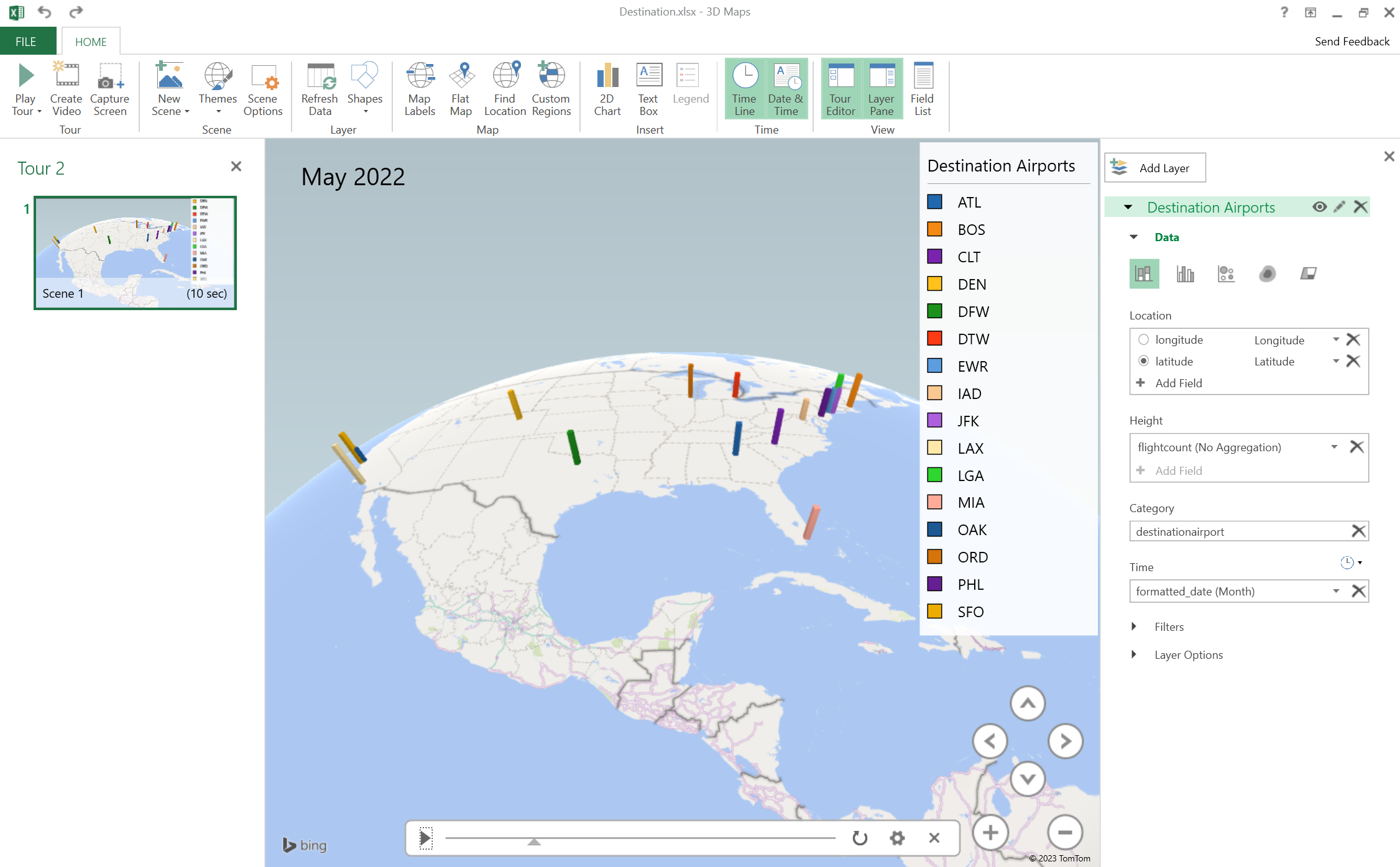
A screenshot of a graph

Description automatically generated with medium confidence

A picture containing text, screenshot, plot, diagram

Description automatically generated

1. Which are the most popular destinations over the months?



A screenshot of a computer screen

Description automatically generated with low confidence

1. Distribution of basic economy tickets (pie chart in Tableau)

Graphical user interface, chart, application, pie chart

Description automatically generated

Chart, pie chart

Description automatically generated