Name: Jonathan Lawrence
Date: 9/20/19
Assignment: 4-2

In [1]:
```
from pyspark.context import SparkContext
from pyspark.sql.session import SparkSession
from pyspark.sql.functions import count, avg
sc = SparkContext('local')
spark = SparkSession(sc)
```

1. Flight Data

In [2]:
```
flights_parquet_path = "C:\\Users\\Jonathan\\Desktop\\Master's Stuff\\Master's Program\\DSC650-T301 Big Data\\dsc650-master\\data\\domestic-flights\\flights.parquet"
airport_codes_path = "C:\\Users\\Jonathan\\Desktop\\Master's Stuff\\Master's Program\\DSC650-T301 Big Data\\dsc650-master\\data\\airport-codes\\airport-codes.csv"
```

In [3]:
```
df_flights = spark.read.parquet(flights_parquet_path)
df_airportCodes = spark.read.load(
  airport_codes_path,
  format="csv",
  sep=",",
  inferSchema=True,
  header=True
)
```

In [4]:
```
print(df_flights.printSchema()) # Print the schema
print(df_airportCodes.printSchema()) # Print the schema
```

```
root
 |-- origin_airport_code: string (nullable = true)
 |-- destination_airport_code: string (nullable = true)
 |-- origin_city: string (nullable = true)
 |-- destination_city: string (nullable = true)
 |-- passengers: long (nullable = true)
 |-- seats: long (nullable = true)
 |-- flights: long (nullable = true)
 |-- distance: double (nullable = true)
 |-- origin_population: long (nullable = true)
 |-- destination_population: long (nullable = true)
 |-- flight_year: long (nullable = true)
 |-- flight_month: long (nullable = true)
 |-- __index_level_0__: long (nullable = true)

None
root
 |-- ident: string (nullable = true)
 |-- type: string (nullable = true)
 |-- name: string (nullable = true)
 |-- elevation_ft: double (nullable = true)
 |-- continent: string (nullable = true)
 |-- iso_country: string (nullable = true)
 |-- iso_region: string (nullable = true)
 |-- municipality: string (nullable = true)
 |-- gps_code: string (nullable = true)
 |-- iata_code: string (nullable = true)
 |-- local_code: string (nullable = true)
 |-- coordinates: string (nullable = true)

None
```

a) Join the data

In [5]:
```
# Origin
joinExpression = df_flights.origin_airport_code == df_airportCodes.iata_code
joinType = "left_outer"

origin_join = df_flights.join(df_airportCodes, joinExpression, joinType)
print(origin_join.printSchema()) # Print the schema of the joined
```

```
root
 |-- origin_airport_code: string (nullable = true)
 |-- destination_airport_code: string (nullable = true)
 |-- origin_city: string (nullable = true)
 |-- destination_city: string (nullable = true)
 |-- passengers: long (nullable = true)
 |-- seats: long (nullable = true)
 |-- flights: long (nullable = true)
 |-- distance: double (nullable = true)
 |-- origin_population: long (nullable = true)
 |-- destination_population: long (nullable = true)
 |-- flight_year: long (nullable = true)
 |-- flight_month: long (nullable = true)
 |-- __index_level_0__: long (nullable = true)
 |-- ident: string (nullable = true)
 |-- type: string (nullable = true)
 |-- name: string (nullable = true)
 |-- elevation_ft: double (nullable = true)
 |-- continent: string (nullable = true)
 |-- iso_country: string (nullable = true)
 |-- iso_region: string (nullable = true)
 |-- municipality: string (nullable = true)
 |-- gps_code: string (nullable = true)
 |-- iata_code: string (nullable = true)
 |-- local_code: string (nullable = true)
 |-- coordinates: string (nullable = true)

None
```

b. Rename and Remove Columns

In [6]:
```
# b. Rename and Remove Columns
origin_join = (origin_join
   .withColumnRenamed('type','origin_airport_type')
   .withColumnRenamed('name', 'origin_airport_name')
   .withColumnRenamed('elevation_ft', 'origin_airport_elevation_ft')
   .withColumnRenamed('iso_region', 'origin_airport_region')
   .withColumnRenamed('municipality', 'origin_airport_municipality')
   .withColumnRenamed('gps_code', 'origin_airport_gps_code')
   .withColumnRenamed('coordinates', 'origin_airport_coordinates'))

columns_to_drop = ['__index_level_0__', 'ident', 'local_code', 'continent', 'i
so_country', 'iata_code']
origin_join = origin_join.drop(*columns_to_drop)
print(origin_join.printSchema()) # Print the schema of the joined
```

```
root
 |-- origin_airport_code: string (nullable = true)
 |-- destination_airport_code: string (nullable = true)
 |-- origin_city: string (nullable = true)
 |-- destination_city: string (nullable = true)
 |-- passengers: long (nullable = true)
 |-- seats: long (nullable = true)
 |-- flights: long (nullable = true)
 |-- distance: double (nullable = true)
 |-- origin_population: long (nullable = true)
 |-- destination_population: long (nullable = true)
 |-- flight_year: long (nullable = true)
 |-- flight_month: long (nullable = true)
 |-- origin_airport_type: string (nullable = true)
 |-- origin_airport_name: string (nullable = true)
 |-- origin_airport_elevation_ft: double (nullable = true)
 |-- origin_airport_region: string (nullable = true)
 |-- origin_airport_municipality: string (nullable = true)
 |-- origin_airport_gps_code: string (nullable = true)
 |-- origin_airport_coordinates: string (nullable = true)

None
```

c. Join to Destination Airport

In [7]:
```
# Destination
joinExpression = df_flights.destination_airport_code == df_airportCodes.iata_c
ode
joinType = "left_outer"

destination_join = df_flights.join(df_airportCodes, joinExpression, joinType)

# Rename
destination_join = (destination_join
   .withColumnRenamed('type','destination_airport_type')
   .withColumnRenamed('name', 'destination_airport_name')
   .withColumnRenamed('elevation_ft', 'destination_airport_elevation_ft')
   .withColumnRenamed('iso_region', 'destination_airport_region')
   .withColumnRenamed('municipality', 'destination_airport_municipality')
   .withColumnRenamed('gps_code', 'destination_airport_gps_code')
   .withColumnRenamed('coordinates', 'destination_airport_coordinates'))

# Remove
columns_to_drop = ['__index_level_0__', 'ident', 'local_code', 'continent', 'i
so_country', 'iata_code']
destination_join = destination_join.drop(*columns_to_drop)

print(destination_join.printSchema()) # Print the schema of the joined
```

```
root
 |-- origin_airport_code: string (nullable = true)
 |-- destination_airport_code: string (nullable = true)
 |-- origin_city: string (nullable = true)
 |-- destination_city: string (nullable = true)
 |-- passengers: long (nullable = true)
 |-- seats: long (nullable = true)
 |-- flights: long (nullable = true)
 |-- distance: double (nullable = true)
 |-- origin_population: long (nullable = true)
 |-- destination_population: long (nullable = true)
 |-- flight_year: long (nullable = true)
 |-- flight_month: long (nullable = true)
 |-- destination_airport_type: string (nullable = true)
 |-- destination_airport_name: string (nullable = true)
 |-- destination_airport_elevation_ft: double (nullable = true)
 |-- destination_airport_region: string (nullable = true)
 |-- destination_airport_municipality: string (nullable = true)
 |-- destination_airport_gps_code: string (nullable = true)
 |-- destination_airport_coordinates: string (nullable = true)

None
```

d. Top Ten Airports

- Rank (1-10)
- Name
- IATA code
- Total Inbound Passengers
- Total Inbound Flights
- Average Daily Passengers
- Average Inbound Flights

Initial join

In [8]:
```
# Inbound only (destination)

destination_join_2008 = destination_join.where("flight_year == '2008'") # Year
2008 only
#destination_join_2008.show(10)


#joinExpression = df_flights.origin_airport_code == df_airportCodes.iata_code
#joinType = 'outer'

#inner_join = df_flights.join(df_airportCodes, joinExpression, joinType)
#inner_join_2008 = inner_join.where("flight_year == '2008'") # Year 2008 only
#inner_join_2008.show(10)
```

Setting up the columns

In [9]:
```
from pyspark.sql import functions as F, Window
from pyspark.sql.functions import desc, dense_rank

# Group by name and destination airport code, then use aggregate functions to
 achieve mathematical calculations
top_ten = destination_join_2008.groupBy('destination_airport_name', 'destinati
on_airport_code').agg(F.sum('passengers'), F.sum('flights'), F.sum('passenger
s') / 365, F.avg('flights'))

# Order with Window, then Rank
windowA = Window.orderBy(F.desc("sum(passengers)"))
top_ten_ranked = top_ten.withColumn("Rank", F.dense_rank().over(windowA))

# Rename columns
top_ten_ranked = top_ten_ranked.withColumnRenamed(
    "destination_airport_name", "Name").withColumnRenamed(
    "destination_airport_code", "IATA code").withColumnRenamed(
    "sum(passengers)", "Total Inbound Passengers").withColumnRenamed(
    "sum(flights)", "Total Inbound Flights").withColumnRenamed(
    "(sum(passengers) / 365)", "Average Daily Passengers").withColumnRenamed(
    "avg(flights)", "Average Inbound Flights")

# Reorder columns
top_ten_ranked = top_ten_ranked.select("Rank","Name","IATA code","Total Inboun
d Passengers","Total Inbound Flights","Average Daily Passengers","Average Inbo
und Flights")

# Show result
print(top_ten_ranked.printSchema()) # Print the schema
top_ten_ranked.show(10)
```

```
root
 |-- Rank: integer (nullable = true)
 |-- Name: string (nullable = true)
 |-- IATA code: string (nullable = true)
 |-- Total Inbound Passengers: long (nullable = true)
 |-- Total Inbound Flights: long (nullable = true)
 |-- Average Daily Passengers: double (nullable = true)
 |-- Average Inbound Flights: double (nullable = true)

None
+----+--------------------+---------+-----------------------+---------------
------+-----------------------+----------------------+
|Rank|                Name|IATA code|Total Inbound Passengers|Total Inbound F
lights|Average Daily Passengers|Average Inbound Flights|
+----+--------------------+---------+-----------------------+---------------
------+-----------------------+----------------------+
|   1|Hartsfield Jackso...|      ATL|                35561795|
395192|       97429.57534246576|      45.03612535612535|
|   2|Chicago O'Hare In...|      ORD|                26398793|
356570|        72325.4602739726|      37.61683721911594|
|   3|Dallas Fort Worth...|      DFW|                22883558|
270243|       62694.67945205479|      52.89547856723429|
|   4|Los Angeles Inter...|      LAX|                19741782|
215000|       54087.07397260274|      46.39620198532585|
|   5|McCarran Internat...|      LAS|                18262263|
164123|       50033.59726027397|      42.22356573192694|
|   6|Phoenix Sky Harbo...|      PHX|                17305718|
181259|       47412.92602739726|     40.981008365362875|
|   7|Charlotte Douglas...|      CLT|                15038489|
205040|        41201.3397260274|      33.32899869960988|
|   8|George Bush Inter...|      IAH|                14870717|
214245|        40741.6904109589|      36.86886938564791|
|   9|Orlando Internati...|      MCO|                14581086|
131710|      39948.180821917806|     36.749441964285715|
|  10|Detroit Metropoli...|      DTW|                14228887|
191910|      38983.252054794524|      34.03866619368571|
+----+--------------------+---------+-----------------------+---------------
------+-----------------------+----------------------+
only showing top 10 rows
```

e. User Defined Functions

In [10]:
```
# Provided functions

from pyspark.sql.functions import udf

@udf('double')
def get_latitude(coordinates):
    split_coords = coordinates.split(',')
    if len(split_coords) != 2:
        return None

    return float(split_coords[0].strip())


@udf('double')
def get_longitude(coordinates):
    split_coords = coordinates.split(',')
    if len(split_coords) != 2:
        return None

    return float(split_coords[1].strip())
```

In [12]:
```
# Verify Schema (before modification)
#print(origin_join.printSchema()) # Print the schema
#print(destination_join.printSchema()) # Print the schema

# Origin
origin_join = origin_join.withColumn(
  'origin_airport_longitude',
  get_longitude(origin_join['origin_airport_coordinates'])
).withColumn(
  'origin_airport_latitude',
  get_latitude(origin_join['origin_airport_coordinates'])
)

# Destination
destination_join = destination_join.withColumn(
  'destination_airport_longitude',
  get_longitude(destination_join['destination_airport_coordinates'])
).withColumn(
  'destination_airport_latitude',
  get_latitude(destination_join['destination_airport_coordinates'])
)

# Verify Schema
print(origin_join.printSchema()) # Print the schema
print(destination_join.printSchema()) # Print the schema
```

```
root
 |-- origin_airport_code: string (nullable = true)
 |-- destination_airport_code: string (nullable = true)
 |-- origin_city: string (nullable = true)
 |-- destination_city: string (nullable = true)
 |-- passengers: long (nullable = true)
 |-- seats: long (nullable = true)
 |-- flights: long (nullable = true)
 |-- distance: double (nullable = true)
 |-- origin_population: long (nullable = true)
 |-- destination_population: long (nullable = true)
 |-- flight_year: long (nullable = true)
 |-- flight_month: long (nullable = true)
 |-- origin_airport_type: string (nullable = true)
 |-- origin_airport_name: string (nullable = true)
 |-- origin_airport_elevation_ft: double (nullable = true)
 |-- origin_airport_region: string (nullable = true)
 |-- origin_airport_municipality: string (nullable = true)
 |-- origin_airport_gps_code: string (nullable = true)
 |-- origin_airport_coordinates: string (nullable = true)
 |-- origin_airport_longitude: double (nullable = true)
 |-- origin_airport_latitude: double (nullable = true)

None
root
 |-- origin_airport_code: string (nullable = true)
 |-- destination_airport_code: string (nullable = true)
 |-- origin_city: string (nullable = true)
 |-- destination_city: string (nullable = true)
 |-- passengers: long (nullable = true)
 |-- seats: long (nullable = true)
 |-- flights: long (nullable = true)
 |-- distance: double (nullable = true)
 |-- origin_population: long (nullable = true)
 |-- destination_population: long (nullable = true)
 |-- flight_year: long (nullable = true)
 |-- flight_month: long (nullable = true)
 |-- destination_airport_type: string (nullable = true)
 |-- destination_airport_name: string (nullable = true)
 |-- destination_airport_elevation_ft: double (nullable = true)
 |-- destination_airport_region: string (nullable = true)
 |-- destination_airport_municipality: string (nullable = true)
 |-- destination_airport_gps_code: string (nullable = true)
 |-- destination_airport_coordinates: string (nullable = true)
 |-- destination_airport_longitude: double (nullable = true)
 |-- destination_airport_latitude: double (nullable = true)

None
```