On-Time Flight Performance Analysis

Executive Business Review - VP & Director of Data Science

John Hupperts

expedia group™

Itinerary

- Company Overview
- Dataset – Background
- Dataset – Technical Synopsis
- Analysis – Questions, Wrangling, Insights
- Business Actions, Further Work
- Appendix: Databricks Platform

Company Overview

- Online Marketplaces for Travel Services
- Portfolio of Consumer Brands
  - Airlines
  - Lodging
  - Car Rental
  - Cruises
  - Experience Packages
- Business Synergies – Leverage Data Across Markets
- Merchants: Buy & Sell
- Agents: Broker for Fees

expedia group™

SEC Form 10-K 2021

Dataset – Background

- **DOT BTS On-Time Performance of Domestic Flights**
  - Categorical Flight Delay-Causes & Cancellations
  - Delayed: Arrived >15min Late
  - Possibly Multiple Delay-Causes Per Flight

Delay Cause	Delay Definition
Air Carrier	Airline's fault
Extreme Weather	Bad weather
National Aviation System (NAS)	Air Traffic Control
Late-Arriving Aircraft	Plane arrives late from previous flight
Security	Security breach, dangerous events, or >29 min TSA lines

Dataset – Technical Synopsis

- Data resided on Databricks simulated file system
- 121G Directory, 1920 CSV files
- Each CSV ~65M, ~645k rows

```
%sh
du --human-readable /dbfs/databricks-datasets/airlines/                    # size of directory
ls -s --human-readable /dbfs/databricks-datasets/airlines/part-00000       # size of 1 data-file
wc --lines /dbfs/databricks-datasets/airlines/part-00000                   # rows in 1 data-file
ls /dbfs/databricks-datasets/airlines/part-* | wc --lines                  # count of data-files
file /dbfs/databricks-datasets/airlines/part-00000                         # data-file type
```

```
121G    /dbfs/databricks-datasets/airlines/
65M /dbfs/databricks-datasets/airlines/part-00000
645919 /dbfs/databricks-datasets/airlines/part-00000
1920
/dbfs/databricks-datasets/airlines/part-00000: CSV text
```

expedia group™

Dataset – Technical Synopsis

- Inferred Schema
  - Integers & Strings

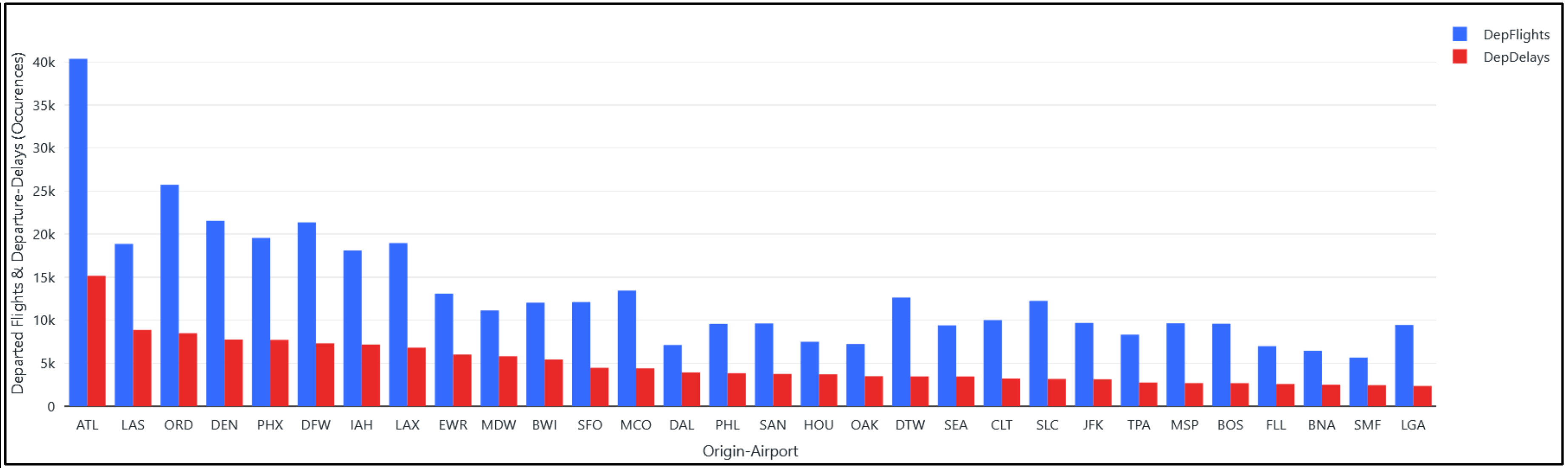
```
%python
from pyspark.sql import SparkSession
import pandas as pd


spark = SparkSession.builder.getOrCreate()
df_demo = spark.read.csv('/databricks-datasets/airlines/part-01918', sep=',', header=True, schema=df_first.schema)
df_demo.printSchema()
df_demo.display()
```

```
root
 |-- Year: integer (nullable = true)
 |-- Month: integer (nullable = true)
 |-- DayofMonth: integer (nullable = true)
 |-- DayOfWeek: integer (nullable = true)
 |-- DepTime: string (nullable = true)
 |-- CRSDepTime: integer (nullable = true)
 |-- ArrTime: string (nullable = true)
 |-- CRSArrTime: integer (nullable = true)
 |-- UniqueCarrier: string (nullable = true)
 |-- FlightNum: integer (nullable = true)
 |-- TailNum: string (nullable = true)
 |-- ActualElapsedTime: string (nullable = true)
 |-- CRSElapsedTime: integer (nullable = true)
 |-- AirTime: string (nullable = true)
 |-- ArrDelay: string (nullable = true)
 |-- DepDelay: string (nullable = true)
 |-- Origin: string (nullable = true)
 |-- Dest: string (nullable = true)
 |-- Distance: string (nullable = true)
 |-- TaxiIn: string (nullable = true)
```

Analysis
Delay Occurrences

- By ratio of departure-delay occurrences to departures, which origin-airports have the most departure-delays?
- For each origin-airport, what proportion of total departures is delayed by each delay-cause?
- Irregular departure-delay ratios among top 30

```
%python
depart_delays_df = spark.sql("""
    WITH origin_flights AS (
        SELECT Origin, COUNT(*) DepFlights
        FROM airlines
        GROUP BY Origin
    ),
    depart_delays AS (
        SELECT
            Origin,
            SUM(CASE WHEN CarrierDelay>0 THEN 1 ELSE 0 END) CarrierDelays,
            SUM(CASE WHEN WeatherDelay>0 THEN 1 ELSE 0 END) WeatherDelays,
            SUM(CASE WHEN NASDelay>0 THEN 1 ELSE 0 END) NASDelays,
            SUM(CASE WHEN SecurityDelay>0 THEN 1 ELSE 0 END) SecurityDelays,
            SUM(CASE WHEN LateAircraftDelay>0 THEN 1 ELSE 0 END) LateAircraftDelays,
            COUNT(*) DepDelays
        FROM airlines
        WHERE IsDepDelayed = 'YES'
        GROUP BY Origin
    )
    SELECT depart_delays.*, origin_flights.DepFlights
    FROM depart_delays
    RIGHT JOIN origin_flights
        ON origin_flights.Origin = depart_delays.Origin
    ORDER BY DepDelays DESC;
""")
depart_delays_df.limit(30).display()  # just display 30 rows in chart
depart_delays_df.createOrReplaceTempView("depart_delays")
```



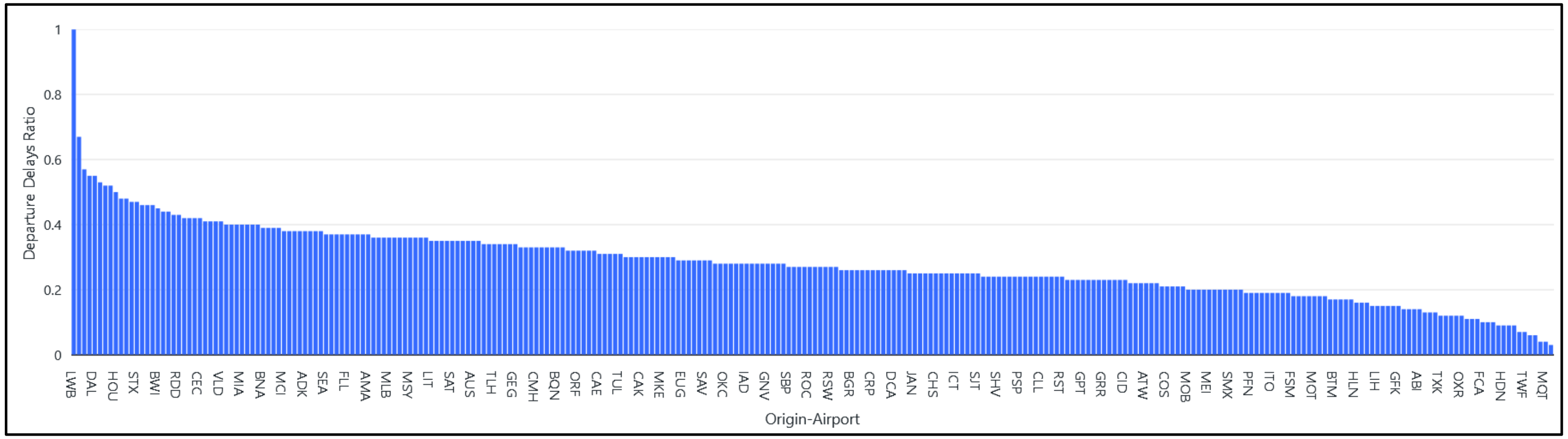
expedia group™

Analysis

Delay Occurrences

- By ratio of departure-delay occurrences to departures, which origin-airports have the most departure-delays? (Cont.)
- For each origin-airport, what proportion of total departures is delayed by each delay-cause? (Cont.)
- Very broad range of delay-ratios across airports

```
%python
depart_delays_ratios_df = spark.sql("""
    SELECT
        Origin,
        ROUND((CarrierDelays / DepFlights), 2) CarrierDelaysRatio,
        ROUND((WeatherDelays / DepFlights), 2) WeatherDelaysRatio,
        ROUND((NASDelays / DepFlights), 2) NASDelaysRatio,
        ROUND((SecurityDelays / DepFlights), 2) SecurityDelaysRatio,
        ROUND((LateAircraftDelays / DepFlights), 2) LateAircraftDelaysRatio,
        ROUND((DepDelays / DepFlights), 2) DepDelaysRatio,
        DepDelays,
        DepFlights
    FROM depart_delays
    ORDER BY DepDelaysRatio DESC;
""")
depart_delays_ratios_df.display()
depart_delays_ratios_df.createOrReplaceTempView("depart_delays_ratios")
```

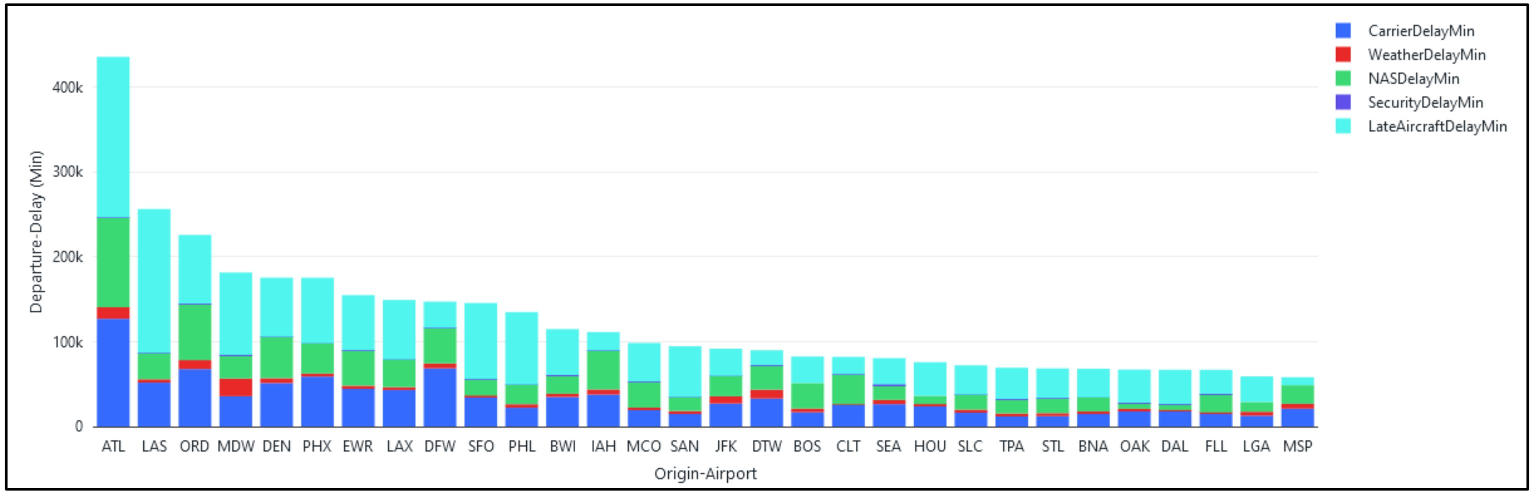


Analysis

Delay Time

- By summation of departure delay-time, which origin-airports have the most delay-time?
- For each origin-airport, how much delay-time is each delay-cause responsible for?
- Irregular skew of delay-causes among airports

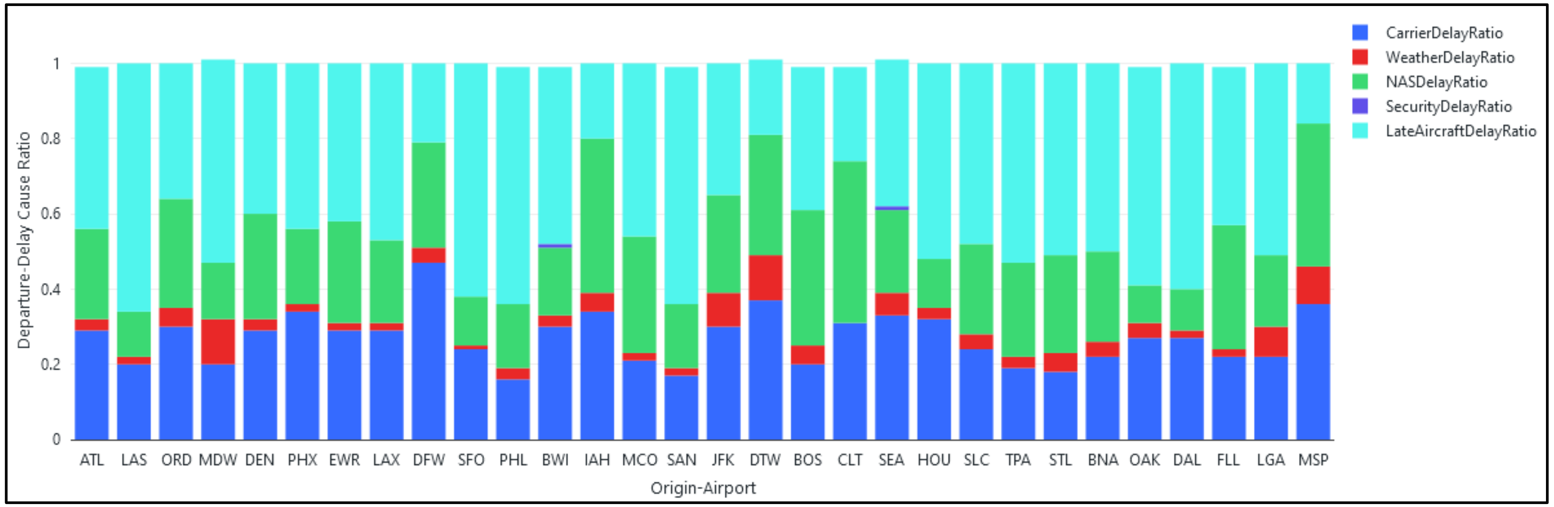
```
%python
depart_delays_minutes_df = spark.sql("""
    WITH depart_delays AS (
        SELECT
            Origin,
            SUM(CarrierDelay) CarrierDelayMin,
            SUM(WeatherDelay) WeatherDelayMin,
            SUM(NASDelay) NASDelayMin,
            SUM(SecurityDelay) SecurityDelayMin,
            SUM(LateAircraftDelay) LateAircraftDelayMin,
            COUNT(*) DepDelays
        FROM airlines
        WHERE IsDepDelayed = 'YES'
        GROUP BY Origin
    )
    SELECT *, (CarrierDelayMin + WeatherDelayMin + NASDelayMin + SecurityDelayMin + LateAircraftDelayMin) TotalDelayMin
    FROM depart_delays
    ORDER BY TotalDelayMin DESC;
""")
depart_delays_minutes_df.limit(30).display()  # just display 30 rows in chart
depart_delays_minutes_df.createOrReplaceTempView("depart_delays_minutes")
```



Analysis
Delay Time

- For each origin-airport, what's the distribution of total departure delay-time among delay-causes?
- Different skews of delay-causes among airports, better visualized independent of volume

```
%python
depart_delays_minutes_ratios_df = spark.sql("""
    SELECT
        Origin,
        ROUND((CarrierDelayMin / TotalDelayMin), 2) CarrierDelayRatio,
        ROUND((WeatherDelayMin / TotalDelayMin), 2) WeatherDelayRatio,
        ROUND((NASDelayMin / TotalDelayMin), 2) NASDelayRatio,
        ROUND((SecurityDelayMin / TotalDelayMin), 2) SecurityDelayRatio,
        ROUND((LateAircraftDelayMin / TotalDelayMin), 2) LateAircraftDelayRatio,
        TotalDelayMin
    FROM depart_delays_minutes
    ORDER BY TotalDelayMin DESC;
""")
depart_delays_minutes_ratios_df.limit(30).display()  # just display 30 rows in chart
depart_delays_minutes_ratios_df.createOrReplaceTempView("depart_delays_minutes_ratios")
```

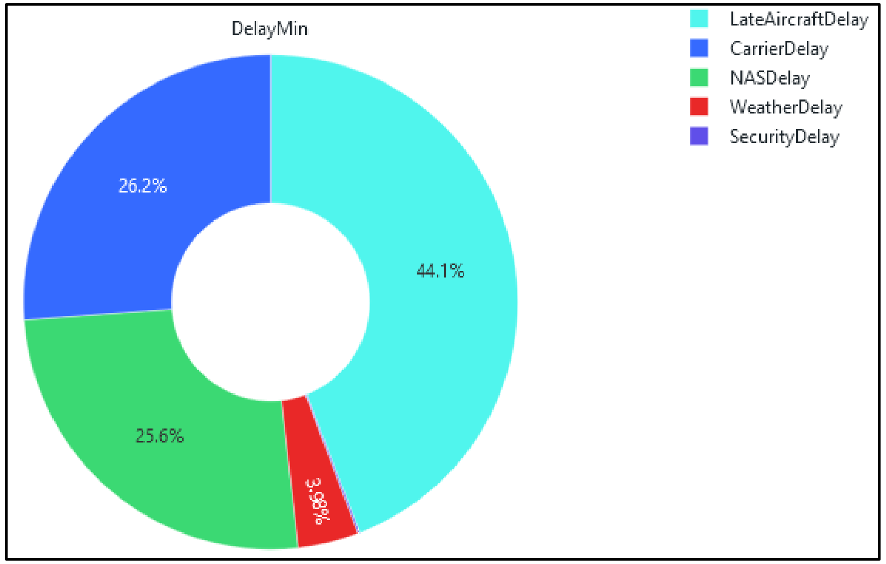


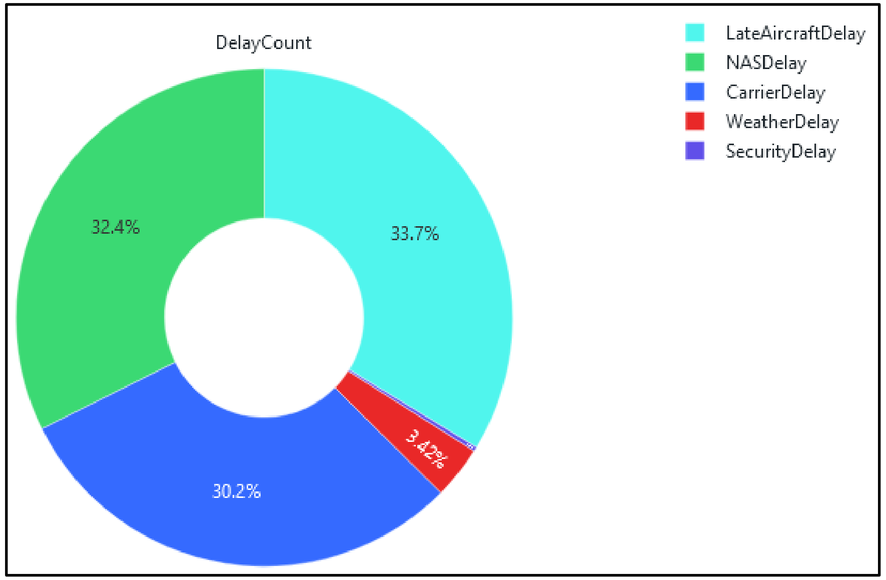
Analysis

Delay Time

- Among all origin-airports, what's the distribution of total departure delay-time among delay-causes?
- Few Security delays

```
%python
# this is an ugly transformation to make the pie chart work
depart_delays_minutes_df = spark.sql("""
    SELECT
        'CarrierDelay' DelayCause, SUM(CarrierDelay) DelayMin, SUM(CASE WHEN CarrierDelay>0 THEN 1 ELSE 0 END) DelayCount
    FROM airlines
    WHERE IsDepDelayed = 'YES'
    UNION ALL
    SELECT
        'WeatherDelay' DelayCause, SUM(WeatherDelay) DelayMin, SUM(CASE WHEN WeatherDelay>0 THEN 1 ELSE 0 END) DelayCount
    FROM airlines
    WHERE IsDepDelayed = 'YES'
    UNION ALL
    SELECT
        'NASDelay' DelayCause, SUM(NASDelay) DelayMin, SUM(CASE WHEN NASDelay>0 THEN 1 ELSE 0 END) DelayCount
    FROM airlines
    WHERE IsDepDelayed = 'YES'
    UNION ALL
    SELECT
        'SecurityDelay' DelayCause,SUM(SecurityDelay) DelayMin, SUM(CASE WHEN SecurityDelay>0 THEN 1 ELSE 0 END) DelayCount
    FROM airlines
    WHERE IsDepDelayed = 'YES'
    UNION ALL
    SELECT
        'LateAircraftDelay' DelayCause, SUM(LateAircraftDelay) DelayMin, SUM(CASE WHEN LateAircraftDelay>0 THEN 1 ELSE 0 END) DelayCount
    FROM airlines
    WHERE IsDepDelayed = 'YES';
""")
depart_delays_minutes_df.display()
```





Analysis

Geography & Weather

- Are there geographic areas where a larger share of departure-delays is due to weather?
- Map IATA codes to States & Map States to FIPS

```
%python
"""
Finally. This had all but a few of the IATA codes in this dataset. The few are presumably smaller airports.
"""
html = requests.get("https://nobleaircharter.com/airport-codes-usa/").text
soup = BeautifulSoup(html, 'html.parser')

iata_aps = []
for elem in soup.select("div div[class*='fusion-text-4'] p"):
    for line in elem.text.splitlines():
        iata_aps.append(line)

org_iata_aps = []
for line in iata_aps:
    split_line = line.split(' ')
    for piece in split_line:
        if (len(piece) == 2) and (piece.upper() == piece):
            state = piece
        if "(" in piece:
            iata = piece
    org_iata_aps.append([state, iata])
    if iata == "(YUM)":
        break
org_iata_aps_df = pd.DataFrame(org_iata_aps)
org_iata_aps_df.columns = ["State", "IATA"]
org_iata_aps_df["IATA"] = org_iata_aps_df["IATA"].str.strip('(').str.strip(')')

org_iata_aps_df = spark.createDataFrame(org_iata_aps_df)
org_iata_aps_df.createOrReplaceTempView("better_states_iata")
org_iata_aps_df.display()
```

Table

	State	IATA
1	SD	ABR
2	TX	ABI
3	AK	ADK

```
%python
"""grab State and FIPS codes"""
df_states_fips = pd.read_html(requests.get("https://www.mercercountypa.gov/dps/state_fips_code_listing.htm").text)[0]
onehalf_df = df_states_fips.iloc[1:,0:2]; onehalf_df.columns = ["State", "FIPS"]
twohalf_df = df_states_fips.iloc[1:,3:5]; twohalf_df.columns = ["State", "FIPS"]
df_states_fips = pd.concat([onehalf_df, twohalf_df]).dropna().reset_index(drop=True, inplace=False)
df_states_fips = spark.createDataFrame(df_states_fips)
df_states_fips.createOrReplaceTempView("states_fips")
df_states_fips.display()
```

Table

	State	FIPS
1	AK	02
2	AL	01
3	AR	05

expedia group™

Analysis
Geography & Weather

- Are there geographic areas where a larger share of departure-delays is due to weather? (Cont.)
- Join IATA, States, & FIPS

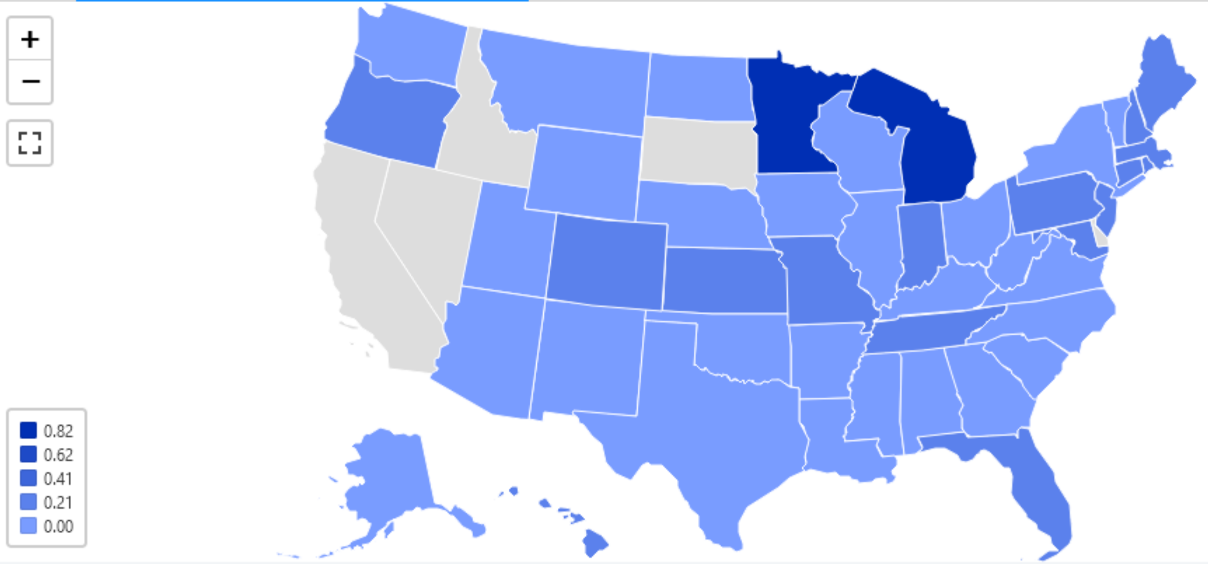
```
%python
airports_states_iata_fips = spark.sql("""
    SELECT iata.State, iata.IATA, fips.FIPS
    FROM better_states_iata iata
    LEFT JOIN states_fips fips
        ON iata.State = fips.State
    ORDER BY State ASC;
""")
airports_states_iata_fips.display()
airports_states_iata_fips.createOrReplaceTempView("airports_states_iata_fips")
```

Table

	State	IATA	FIPS
1	AK	CIK	02
2	AK	CYF	02
3	AK	VAK	02

```
%python
"""chloropleth visualization can be a little buggy. If it fails, refresh your web-browser window."""
weather_delay_chart_df = spark.sql("""
SELECT ddmr.Origin, ddmr.WeatherDelayRatio, asif.State, asif.FIPS
FROM depart_delays_minutes_ratios ddmr
LEFT JOIN airports_states_iata_fips asif
  ON ddmr.Origin = asif.IATA;
""")
weather_delay_chart_df.display()
# maybe winters are harsh in the north, tornadoes in the middle, hurricanes in Florida
```

Table   Departure-WeatherDelay Time Share by FIPS Code



Analysis

Geography & Path

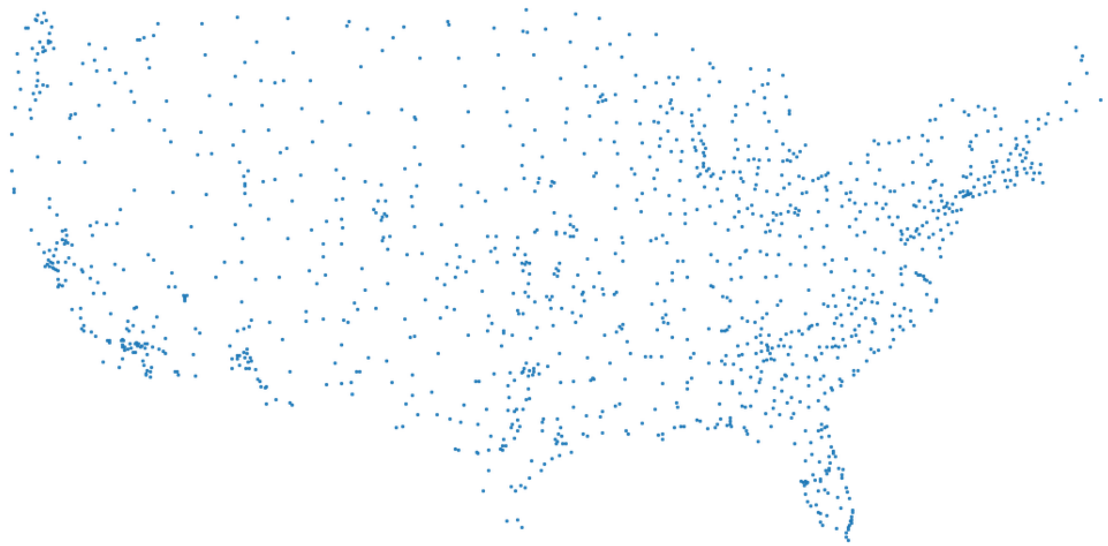
- Can insights be gleaned from a visual of a random sample of flight paths?
- Grab lat., long. coordinates for domestic airports

```
%python
"""data found at https://openflights.org/data.html available under Database Contents License"""
"""Native Databricks Map (Markers) chart too unstable"""
airports_df = pd.read_csv("https://raw.githubusercontent.com/jpatokal/openflights/master/data/airports.dat",
                          delimiter=',',
                          names=['id', 'name', 'city', 'country', 'iata', 'icao', 'lat', 'long', 'altitude', 'timezone', 'dst', 'tz',
'type', 'source'])

airports_df = spark.createDataFrame(airports_df)
airports_df.createOrReplaceTempView("airports_data")
airports_df = spark.sql("""
    SELECT iata, lat, long
    FROM airports_data
    WHERE COUNTRY = 'United States'
    AND LAT > 25 AND LAT < 50          -- Normal USA landmass
    AND LONG < -65 AND LONG > -125     -- Normal USA landmass
""")
airports_df.createOrReplaceTempView("airports_data")
airports_df.display()
pd_airports_df = airports_df.toPandas()
```

Table

	iata	lat	long
1	ORL	28.5455	-81.332901
2	BED	42.47000122	-71.28900146
3	OSC	44.451599	-83.394096

```
%python
import matplotlib.pyplot as plt

fig, ax = plt.subplots()
fig.set_size_inches(20, 10)

ax.scatter(pd_airports_df['long'], pd_airports_df['lat'], s=5)
ax.axis('off')

plt.show()
```



Analysis

Geography & Path

- Can insights be gleaned from a visual of a random sample of flight paths? (Cont.)
- Join lat., long coordinates to flight origins & destinations
- Calculate path-lines

```
%python
flights_df = spark.sql("""
    SELECT airlines.origin, airlines.dest, ad1.lat origin_lat, ad1.long origin_long, ad2.lat dest_lat, ad2.long dest_long
    FROM airlines
    INNER JOIN airports_data ad1 -- have to sacrifice losing a few iata airports
      ON airlines.ORIGIN = ad1.IATA
    INNER JOIN airports_data ad2 -- have to sacrifice losing a few iata airports
      ON airlines.dest = ad2.IATA
""")
flights_df.createOrReplaceTempView("flights")
flights_df.display()
```

Table

	origin	dest	origin_lat	origin_long	dest_lat	dest_long
1	FSM	MEM	35.33660125732422	-94.36740112304688	35.04240036010742	-89.97669982910156
2	FSM	MEM	35.33660125732422	-94.36740112304688	35.04240036010742	-89.97669982910156
3	FSM	MEM	35.33660125732422	-94.36740112304688	35.04240036010742	-89.97669982910156

```
%python
import geopandas as gpd
from shapely.geometry import LineString


pd_flights_df = flights_df.toPandas().sample(10000) # we just need a random sample of 10,000 flights for the visual, else it's too ridiculous
geometry = [LineString([[pd_flights_df.iloc[i]['origin_long'], pd_flights_df.iloc[i]['origin_lat']], [pd_flights_df.iloc[i]['dest_long'], pd_flights_df.iloc[i]['dest_lat']]]) for i in range(pd_flights_df.shape[0])]
routes = gpd.GeoDataFrame(pd_flights_df, geometry=geometry, crs='EPSG:4326')
print(routes)
```

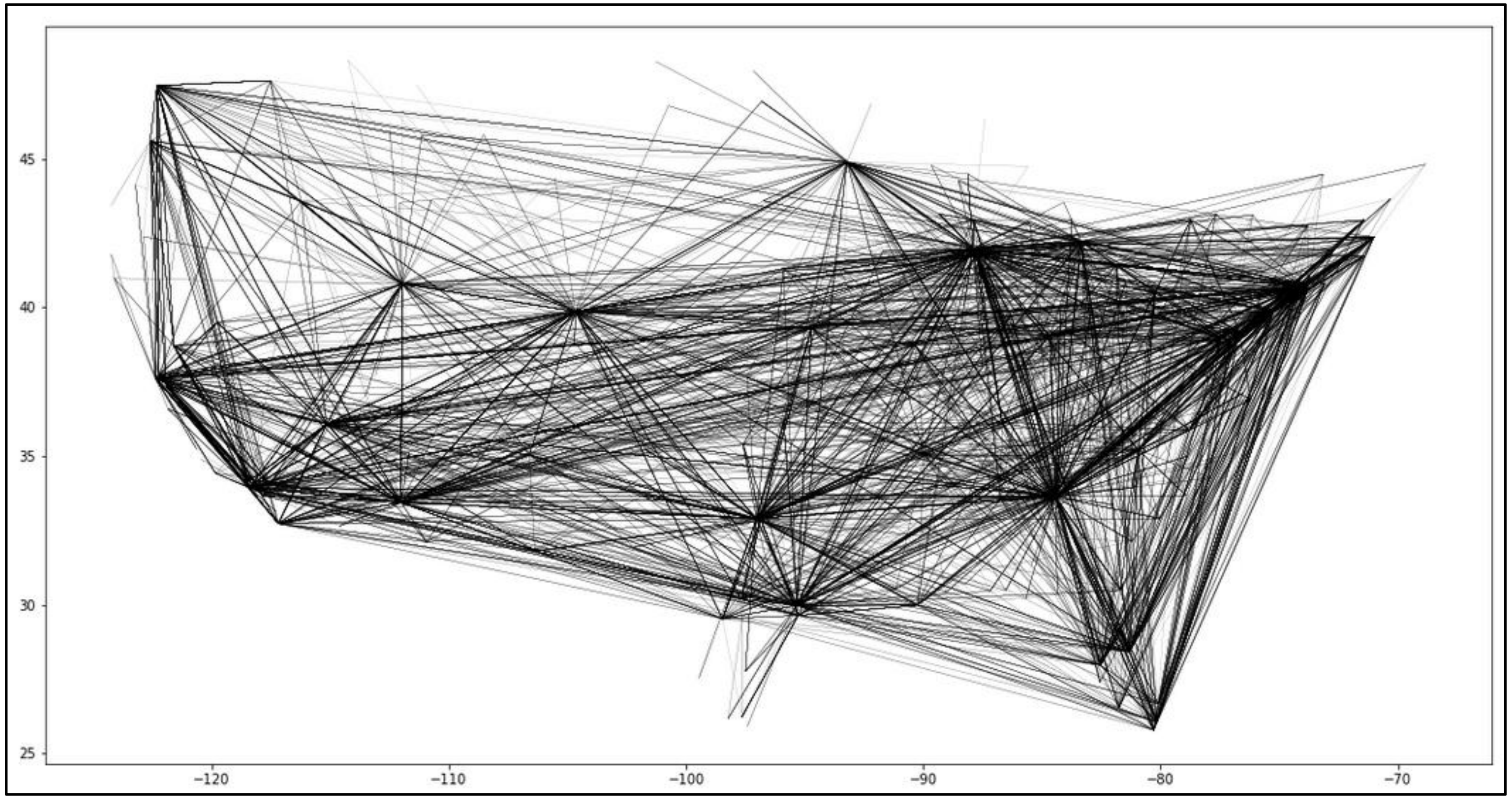
```
       origin dest  origin_lat  origin_long   dest_lat   dest_long  \
61162     MEM  JAN   35.042400   -89.976700  32.311199  -90.075897
61277     MEM  CHA   35.042400   -89.976700  35.035301  -85.203796
92519     CVG  HSV   39.048801   -84.667801  34.637199  -86.775101
85831     CLE  FLL   41.411701   -81.849800  26.072599  -80.152702
331489    BNA  MDW   36.124500   -86.678200  41.785999  -87.752403
...       ...  ...         ...          ...        ...         ...
259964    DFW  SAT   32.896801   -97.038002  29.533701  -98.469803
194000    SEA  SLC   47.449001  -122.308998  40.788399 -111.977997
124244    DCA  LGA   38.852100   -77.037697  40.777199  -73.872597
352567    IAD  PHX   38.944500   -77.455803  33.434299 -112.012001
409827    DEN  MCO   39.861698  -104.672997  28.429399  -81.308998

                                                 geometry
61162   LINESTRING (-89.97670 35.04240, -90.07590 32.3...
61277   LINESTRING (-89.97670 35.04240, -85.20380 35.0...
92519   LINESTRING (-84.66780 39.04880, -86.77510 34.6...
```

Analysis

Geography & Path

- Can insights be gleaned from a visual of a random sample of flight paths? (Cont.)
- Overlay this on a map with major cities highlighted

```
%python
fig = plt.figure()
ax = plt.axes()

fig.set_size_inches(20, 10)
# ax.patch.set_facecolor('black')

routes.plot(ax=ax, color='black', linewidth=0.1)

plt.setp(ax.spines.values(), color='black')
plt.setp([ax.get_xticklines(), ax.get_yticklines()], color='black')

plt.show()
```



Business Actions, Further Work
Expanded Use Cases, Addl. Datasets

- Unfortunately, dataset wasn’t very robust; no actionable business insights
- However, if augmented with more thorough data to feed Expedia’s offering portfolio AI…
  - Could better weight portfolio of merchandised & brokered tickets with respect to flight delay & cancellation risk
    - Example: Mid-summer is peak tornado season – more delayed/canceled flights
      - Expedia could recognize conditions & rebalance its flight ticket offerings to have a greater share of brokered tickets rather than merchandised tickets to alleviate its risk of claims for refunds by customers
      - If unable to expediently rebalance its portfolio of merchandised & brokered tickets, perhaps Expedia could sell off assets in other businesses to raise cash and shore up liquidity to repay those refunds.
    - Could offer favorable discounted lodging accommodations to customers facing canceled flights
      - Harvest some goodwill & addl. revenue from sale of lodging
  - Interesting candidate for integration: Weather data
    - Pattern of more frequent inclement weather in certain locations due to climate change, affecting travel behavior

expedia group™

Appendix: Databricks Platform

- Cohesive, well packaged, collaborative Cloud development environment for Apache Spark w/ integrated tools: Web UI, Notebooks, COS
- Multiple languages/kernels in 1 notebook
  - I used Python & PySpark mostly
  - But, in dev., was nice to use SQL against Spark tables w/o boiler plate
- If moved into production…
  - Data engineering SME should improve utilization of Spark
  - Architect set up a well optimized data lake for storage; DeltaLake
  - Job scheduler like Apache Airflow
  - Access controls for collaboration