databricks FinalProject

```
from pyspark.sql import SparkSession


spark = SparkSession.builder.appName('FLIGHTSSQL').getOrCreate()


df = spark.read.csv('/FileStore/tables/2008.csv',inferSchema=True,header=True)
display(df)
```

Year	Month	DayofMonth	DayOfWeek	DepTime	CRSDepTime	ArrTime	CRSArrTime	UniqueC
2008	1	3	4	2003	1955	2211	2225	WN
2008	1	3	4	754	735	1002	1000	WN
2008	1	3	4	628	620	804	750	WN
2008	1	3	4	926	930	1054	1100	WN
2008	1	3	4	1829	1755	1959	1925	WN
2008	1	3	4	1940	1915	2121	2110	WN
2008	1	3	4	1937	1830	2037	1940	WN
2008	1	3	4	1039	1040	1132	1150	WN
2008	1	3	4	617	615	652	650	WN
2008	1	3	4	1620	1620	1639	1655	WN
2008	1	3	4	706	700	916	915	WN
2008	1	3	4	1644	1510	1845	1725	WN
2008	1	3	4	1426	1430	1426	1425	WN
2008	1	3	4	715	715	720	710	WN
2008	1	3	4	1702	1700	1651	1655	WN
2008	1	3	4	1029	1020	1021	1010	WN
2008	1	3	4	1452	1425	1640	1625	WN
2008	1	3	4	754	745	940	955	WN
2008	1	3	4	1323	1255	1526	1510	WN
2008	1	3	4	1416	1325	1512	1435	WN
2008	1	3	4	706	705	807	810	WN
2008	1	3	4	1657	1625	1754	1735	WN
2008	1	3	4	1900	1840	1956	1950	WN
2008	1	3	4	1039	1030	1133	1140	WN
2008	1	3	4	801	800	902	910	WN
2008	1	3	4	1520	1455	1619	1605	WN
2008	1	3	4	1422	1255	1657	1610	WN
2008	1	3	4	1954	1925	2239	2235	WN
2008	1	3	4	636	635	921	945	WN
2008	1	3	4	734	730	958	1020	WN
2008	1	3	4	2107	1945	2334	2230	WN
2008	1	3	4	1008	1005	1234	1255	WN

2008	1	3	4	712	710	953	1000	WN
2008	1	3	4	1312	1300	1546	1550	WN
2008	1	3	4	1449	1430	1715	1720	WN
2008	1	3	4	1634	1555	1859	1845	WN
2008	1	3	4	831	830	935	955	WN
2008	1	3	4	1812	1650	1927	1815	WN
2008	1	3	4	1127	1105	1235	1230	WN
2008	1	3	4	1424	1355	1531	1520	WN
2008	1	3	4	1326	1230	1559	1530	WN
2008	1	3	4	1749	1725	2019	2030	WN
2008	1	3	4	726	720	958	1020	WN
2008	1	3	4	646	640	929	955	WN
2008	1	3	4	1153	1140	1428	1440	WN
2008	1	3	4	1528	1510	1802	1810	WN
2008	1	3	4	634	635	907	935	WN
2008	1	3	4	831	830	1148	1140	WN
2008	1	3	4	1450	1435	1806	1745	WN
2008	1	3	4	2245	1730	2354	1850	WN
2008	1	3	4	615	615	724	735	WN
2008	1	3	4	1150	1145	1303	1305	WN
2008	1	3	4	2025	1940	2135	2100	WN
2008	1	3	4	1038	945	1314	1225	WN
2008	1	3	4	1900	1850	2123	2045	WN
2008	1	3	4	700	700	851	900	WN
2008	1	3	4	948	925	959	940	WN
2008	1	3	4	646	620	725	655	WN
2008	1	3	4	1110	1040	1136	1110	WN
2008	1	3	4	1535	1535	1603	1610	WN
2008	1	3	4	1919	1915	1942	1950	WN
2008	1	3	4	1053	1055	1245	1240	WN
2008	1	3	4	1433	1440	1623	1625	WN
2008	1	3	4	2015	2010	2158	2155	WN
2008	1	3	4	2139	2130	2244	2240	WN
2008	1	3	4	1500	1500	1602	1615	WN
2008	1	3	4	850	850	1000	1000	WN
2008	1	3	4	646	645	752	755	WN
2008	1	3	4	1221	1220	1328	1330	WN
2008	1	3	4	1738	1730	1841	1840	WN
2008	1	3	4	1813	1735	1936	1905	WN
2008	1	3	4	802	750	1001	955	WN
2008	1	3	4	1820	1825	1946	1955	WN
2008	1	3	4	821	820	953	945	WN

2008	1	3	4	1734	1650	1941	1905	WN
2008	1	3	4	712	700	926	915	WN
2008	1	3	4	1318	1310	1410	1400	WN
2008	1	3	4	958	900	1052	950	WN
2008	1	3	4	1859	1850	1950	1945	WN
2008	1	3	4	1538	1445	1753	1710	WN
2008	1	3	4	933	935	1151	1200	WN
2008	1	3	4	2248	2125	102	2345	WN
2008	1	3	4	1327	1230	1550	1500	WN

Showing the first 1000 rows.

```
df.count()
```

```
Out[4]: 7009728
```

```
df.createOrReplaceTempView("FLIGHTS")
```

```
spark.sql("describe FLIGHTS").show(50)
```

```
+-----------------+---------+-------+
|         col_name|data_type|comment|
+-----------------+---------+-------+
|             Year|      int|   null|
|            Month|      int|   null|
|       DayofMonth|      int|   null|
|        DayOfWeek|      int|   null|
|          DepTime|   string|   null|
|       CRSDepTime|      int|   null|
|          ArrTime|   string|   null|
|       CRSArrTime|      int|   null|
|    UniqueCarrier|   string|   null|
|        FlightNum|      int|   null|
|          TailNum|   string|   null|
|ActualElapsedTime|   string|   null|
|   CRSElapsedTime|   string|   null|
|           AirTime|   string|   null|
|         ArrDelay|   string|   null|
|         DepDelay|   string|   null|
|           Origin|   string|   null|
|             Dest|   string|   null|
|         Distance|      int|   null|
|           TaxiIn|   string|   null|
|          TaxiOut|   string|   null|
|        Cancelled|      int|   null|
| CancellationCode|   string|   null|
|         Diverted|      int|   null|
|     CarrierDelay|   string|   null|
|     WeatherDelay|   string|   null|
|         NASDelay|   string|   null|
|    SecurityDelay|   string|   null|
|LateAircraftDelay|   string|   null|
+-----------------+---------+-------+
```

```
## BUSINESS QUESTIONS ##
## CANCELLATIONS


# What is the most common reason for Cancellation? (CancellationCode)
# reason for cancellation (A = carrier, B = weather, C = NAS, D = security)

spark.sql("SELECT CASE  \
                    WHEN CancellationCode == 'A' THEN 'Carrier' \
                    WHEN CancellationCode == 'B' THEN 'Weather' \
                    WHEN CancellationCode == 'C' THEN 'NAS' \
                    WHEN CancellationCode == 'D' THEN 'Security' \
                    ELSE 'N/A' \
                  END AS cancellationReason, count(*) as totalCancellations \
                  FROM FLIGHTS \
                  WHERE Cancelled == 1 AND CancellationCode IS NOT NULL \
            GROUP BY CancellationCode ORDER BY totalCancellations DESC").show()

# Based on the results, Weather is the #1 cause of flight cancellations with a total of 54,904,
however, the second-place
# is very close. With a total of 54,330 canceled flights, the carrier is the #2 reason for flight
cancellations.

# From the results we could also highly how Security reasons are the less common cause of flight
cancellation, with just 12 occurrences
# during the whole year 2008.

+------------------+------------------+
|cancellationReason|totalCancellations|
+------------------+------------------+
|           Weather|             54904|
|           Carrier|             54330|
|               NAS|             28188|
|          Security|                12|
+------------------+------------------+
```

```
# What are the routes (Origin, Dest) with more cancellations for every cancellation code?

# CancellationCode == 'A' means Carrier
spark.sql("SELECT UniqueCarrier, Origin, Dest, count(*) as totalCarrierCancellations \
            FROM FLIGHTS \
            WHERE Cancelled = 1 AND CancellationCode == 'A' \
            GROUP BY UniqueCarrier, Origin, Dest  ORDER BY totalCarrierCancellations DESC").show()

# From results we can notice two important elements:
# First, the carriers in the top 20 of Carrier's caused cancellations are WN (Southwest Airlines
Co.), UA (United Air Lines Inc.)
# HA (Hawaiian Airlines Inc.), US Airways Inc. (US), AA (American Airlines Inc.)
# From the business point of view this carriers should revise the operations of this routes and
determine
# with is causing such a frequent cancellations so they can reduce the cancellation %.
# The reasons could be aircraft conditions, cargo loading logistics, connecting flights, and some
others.

# The second observation, is that the tuples Origin, Dest and Dest Origin for the same Carrier
repeat on the results.
# Eg: HOU-DAL / DAL-HOU, SFO-LAX / LAX-SFO. This is quite logic, because if the same airplane
serves the round-trip route, then
# if one of the flights is cancelled, the other will be affected too. For the business analysis,
the carrier should find causes
# of the cancellation in each of the locations, maybe one location is causing the cancellation more
than the other.
```

```
+-------------+------+----+-------------------------+
|UniqueCarrier|Origin|Dest|totalCarrierCancellations|
+-------------+------+----+-------------------------+
|           WN|   HOU| DAL|                      329|
|           WN|   DAL| HOU|                      275|
|           UA|   SFO| LAX|                      227|
|           UA|   LAX| SFO|                      225|
|           WN|   LAS| PHX|                      206|
|           UA|   LGA| ORD|                      205|
|           HA|   OGG| HNL|                      202|
|           HA|   HNL| OGG|                      196|
|           WN|   PHX| LAS|                      177|
|           UA|   DCA| ORD|                      176|
|           UA|   ORD| LGA|                      175|
|           US|   BOS| PHL|                      173|
|           AA|   ORD| LGA|                      171|
|           WN|   LAX| OAK|                      168|
|           WN|   OAK| LAX|                      166|
|           AA|   LGA| ORD|                      165|
|           UA|   ORD| DCA|                      157|
|           US|   PHL| BOS|                      157|
|           AA|   DFW| LGA|                      155|
|           AA|   LGA| DFW|                      153|
+-------------+------+----+-------------------------+
only showing top 20 rows
```

```
# What are the routes (Origin, Dest) with more cancellations for every cancellation code?

# CancellationCode == 'B' means Weather
spark.sql("SELECT Origin, Dest, count(*) as totalWeatherCancellations \
            FROM FLIGHTS \
            WHERE Cancelled = 1 AND CancellationCode == 'B' \
            GROUP BY Origin, Dest ORDER BY totalWeatherCancellations DESC").show()

# From the user's point of view, these routes should expect delays due to bad condition weather.
# Examples of the Cities in these routes are: Aspen, CO (Montain location), New York, Boston,
Chicago.

# From the operational point of view, airlines should consider cancellation costs per weather
conditions in these routes
# Therefore, these may impact the tickets costs on this cities.

+------+----+------------------------+
|Origin|Dest|totalWeatherCancellations|
+------+----+------------------------+
|   ASE| DEN|                     276|
|   DEN| ASE|                     263|
|   LGA| ORD|                     208|
|   ORD| LGA|                     207|
|   BOS| JFK|                     185|
|   JFK| BOS|                     182|
|   HOU| DAL|                     172|
|   DAL| HOU|                     162|
|   LGA| BOS|                     156|
|   SUN| SLC|                     152|
|   BOS| LGA|                     151|
|   ORD| CVG|                     151|
|   EWR| ORD|                     142|
|   ORD| EWR|                     141|
|   CVG| ORD|                     141|
|   DFW| ORD|                     137|
|   ORD| MSN|                     133|
|   ORD| DFW|                     132|
|   MSN| ORD|                     129|
|   LGA| DCA|                     125|
+------+----+------------------------+
only showing top 20 rows


# What are the routes (Origin, Dest) with more cancellations for every cancellation code?

# CancellationCode == 'C' means National Airspace System (NAS)
spark.sql("SELECT Origin, Dest, count(*) as totalNASCancellations \
            FROM FLIGHTS \
            WHERE Cancelled = 1 AND CancellationCode == 'C' \
            GROUP BY Origin, Dest ORDER BY totalNASCancellations DESC").show()

# Some of the airport that most repeat in the top 20 results are:
# La Guardia in New York,
# Chicago O'Hare International in Chicago
# From the operational point of view, airlines should consider operation costs of cancellations on
these airports,
# These NAS cancellation may be explained due to heavy traffic volume at this airports.
```

```
+------+----+---------------------+
|Origin|Dest|totalNASCancellations|
+------+----+---------------------+
|   BOS| LGA|                  392|
|   LGA| BOS|                  385|
|   DCA| LGA|                  340|
|   LGA| DCA|                  328|
|   LGA| ORD|                  274|
|   ORD| LGA|                  273|
|   ORD| MSN|                  186|
|   MSN| ORD|                  179|
|   LGA| ATL|                  179|
|   ATL| LGA|                  171|
|   MKE| ORD|                  158|
|   ORD| MKE|                  158|
|   GRB| ORD|                  149|
|   PIA| ORD|                  144|
|   ORD| GRB|                  138|
|   ORD| PIA|                  138|
|   CLT| EWR|                  132|
|   EWR| CLT|                  131|
|   AZO| ORD|                  126|
|   CVG| ORD|                  122|
+------+----+---------------------+
only showing top 20 rows


# What are the routes (Origin, Dest) with more cancellations for every cancellation code?

# CancellationCode == 'D' means Security
spark.sql("SELECT Origin, Dest, count(*) as totalSecurityCancellations \
            FROM FLIGHTS \
            WHERE Cancelled = 1 AND CancellationCode == 'D' \
            GROUP BY Origin, Dest ORDER BY totalSecurityCancellations DESC").show()

# The results show a maximun of 1 cancellation per route, therefore the security cancellations are
very uncommon.
# This report doen't lead to any business decision or conclusion.

+------+----+--------------------------+
|Origin|Dest|totalSecurityCancellations|
+------+----+--------------------------+
|   FSM| DFW|                         1|
|   LAX| MIA|                         1|
|   ATL| DHN|                         1|
|   DFW| FSM|                         1|
|   OAK| BUR|                         1|
|   DFW| LEX|                         1|
|   MIA| LAX|                         1|
|   YUM| LAX|                         1|
|   ATL| HOU|                         1|
|   LEX| DFW|                         1|
|   DHN| ATL|                         1|
|   BUR| OAK|                         1|
+------+----+--------------------------+
```

```
#Distinct carriers on the dataset
df.select("UniqueCarrier").distinct().show()
# In Total 20 Carriers
```

```
+-------------+
|UniqueCarrier|
+-------------+
|           UA|
|           AA|
|           NW|
|           EV|
|           B6|
|           DL|
|           OO|
|           F9|
|           YV|
|           US|
|           AQ|
|           MQ|
|           OH|
|           HA|
|           XE|
|           AS|
|           FL|
|           CO|
|           WN|
|           9E|
+-------------+
```

```
## MODELS ##
# K-Mean clustering for Carriers to get groups of airlines based on the following features:
# AvgCarrierDelay: Average time in minutes on Carrier's delays for all its flights during 2008.
# AvgArrDelay: Average time in minutes on arrival delays for flights operated by the Carrier in
2008.
# AvgDepDelay: Average time in minutes on departure delays for flights operated by the Carrier in
2008.
# NumFlights: Number of flights operated by the airline during 2008. Gives an idea of the airline's
operation volume.
# PrgtCancellation: Percentage of flights cancellations for the carrier during 2008. It was
considered the pergentage and not the
# total number of cancellations because not all airlines have the same volumen of flights,
therefore, the percentage is more
# significant for do comparison between the carriers.

# Note: This code was prepared using the Incident Management example from class as a reference.
# Also, it was used the following two links for reference:
# https://rsandstroem.github.io/sparkkmeans.html
# https://runawayhorse001.github.io/LearningApacheSpark/clustering.html

# First, let's aggregate the data by Carrier to get the cluster's features:
from pyspark.sql import functions as F

A1 = df.groupBy("UniqueCarrier") \
        .agg(F.sum("Cancelled"), F.avg("CarrierDelay"), F.avg("ArrDelay"), F.avg("DepDelay"),
F.count("FlightNum"))\
        .withColumnRenamed("sum(Cancelled)","NumCancellation")\
        .withColumnRenamed("avg(CarrierDelay)","AvgCarrierDelay")\
        .withColumnRenamed("avg(ArrDelay)","AvgArrDelay")\
        .withColumnRenamed("avg(DepDelay)","AvgDepDelay")\
        .withColumnRenamed("count(FlightNum)","NumFlights") \
        .withColumn('PrgtCancellation', (F.col("NumCancellation") / F.col('NumFlights')) * 100 )


# Define the features columns
FEATURES_COL = ['AvgCarrierDelay', 'AvgArrDelay', 'AvgDepDelay', 'NumFlights','PrgtCancellation']

# Convert all data columns to float
for col in df.columns:
    if col in FEATURES_COL:
        A1 = A1.withColumn(col,A1[col].cast('float'))

A1.show()
```

```
+-------------+---------------+------------------+------------------+-------------------+---------
-+--------------------+
|UniqueCarrier|NumCancellation|   AvgCarrierDelay|       AvgArrDelay|        AvgDepDelay|NumFlight
s|   PrgtCancellation|
+-------------+---------------+------------------+------------------+-------------------+---------
-+--------------------+
|           UA|          10541| 14.81968088772486|11.291322186680183|  14.11257661236138|    44951
5|  2.344971802943172|
|           AA|          17440|17.309764206497203|12.607194035713981| 13.280898264437912|    60488
5| 2.8831926729874273|
|           NW|           2906| 19.14381361858634| 7.368539129929383|  6.463235656670833|    34765
2| 0.8358933646289969|
|           EV|           5026|25.375897713272636|10.208002415713782| 11.922537970871462|    28057
```

```
5| 1.7913213935667824|
|           B6|           3205|13.766349064461735| 11.08418439051813| 12.653395748122113|    19609
1| 1.6344452320606249|
|           DL|           6813|14.277822743828409| 7.855163154883384|  8.007765572702564|    45193
1|  1.507531016903023|
|           OO|          12436| 15.94580265095729| 6.598884736868734| 7.4564427592619955|    56715
9| 2.1926831805543068|
|           F9|            303|12.708679359935184| 6.108246666107523|  5.919601516833923|     9576
2|0.31640943171612956|
|           YV|           9219| 32.59070160608622|11.775181433600808| 12.000675279875033|    25493
0|  3.616286823833994|
|           US|           6582|13.376279647426236|2.8481100056939694|  5.717489671893907|    45358
9| 1.4510933907127377|
|           AQ|             42|  22.1497461928934|-2.888673890608875|-1.3977829337458108|      780
0| 0.5384615384615384|
|           MQ|          18331|15.001680273066366| 9.890667945776498| 10.695641776641581|    49069
3| 3.7357370086795614|
|           OH|           6462|  20.4775193208208|11.817467683998007| 11.536153117856601|    19760
7| 3.2701270703973035|
|           HA|            570| 33.07478976561102|1.2644089394236424| 0.4552013450206487|     6182
6|  0.921942224953903|
|           XE|           9992|13.859438116586826|10.635404800035221| 11.395866476493499|    37451
0|  2.668019545539505|
|           AS|           2139|15.960019874130507| 4.804346362093581|  6.848722010417226|    15110
2| 1.4156000582388057|
|           FL|           2236| 7.660307649810956|  9.09137538507922|  9.262713040260852|    26168
4| 0.8544656914446431|
|           CO|           3702|13.637260941585554|  10.9790372802913|  13.18522978602152|    29845
5|  1.240387998190682|
|           WN|          12389|10.288646095459985|  5.17967817300539| 10.383034750411133|   120175
4| 1.0309098201462197|
|           9E|           7100|19.864633213075326| 4.111134703250211|  6.765859659983622|    26220
8| 2.7077739809616794|
+-------------+---------------+------------------+------------------+-------------------+---------
-+--------------------+


# Drop the null values
A1 = A1.na.drop()
A1.show()

+-------------+---------------+------------------+------------------+-------------------+---------
-+--------------------+
|UniqueCarrier|NumCancellation|   AvgCarrierDelay|       AvgArrDelay|        AvgDepDelay|NumFlight
s|   PrgtCancellation|
+-------------+---------------+------------------+------------------+-------------------+---------
-+--------------------+
|           UA|          10541| 14.81968088772486|11.291322186680183|  14.11257661236138|    44951
5|  2.344971802943172|
|           AA|          17440|17.309764206497203|12.607194035713981| 13.280898264437912|    60488
5| 2.8831926729874273|
|           NW|           2906| 19.14381361858634| 7.368539129929383|  6.463235656670833|    34765
2| 0.8358933646289969|
|           EV|           5026|25.375897713272636|10.208002415713782| 11.922537970871462|    28057
5| 1.7913213935667824|
|           B6|           3205|13.766349064461735| 11.08418439051813| 12.653395748122113|    19609
1| 1.6344452320606249|
|           DL|           6813|14.277822743828409| 7.855163154883384|  8.007765572702564|    45193
```

```
1|  1.507531016903023|
|           OO|          12436| 15.94580265095729| 6.598884736868734| 7.4564427592619955|    56715
9| 2.1926831805543068|
|           F9|            303|12.708679359935184| 6.108246666107523|  5.919601516833923|     9576
2|0.31640943171612956|
|           YV|           9219| 32.59070160608622|11.775181433600808| 12.000675279875033|    25493
0|  3.616286823833994|
|           US|           6582|13.376279647426236|2.8481100056939694|  5.717489671893907|    45358
9| 1.4510933907127377|
|           AQ|             42|  22.1497461928934|-2.888673890608875|-1.3977829337458108|      780
0| 0.5384615384615384|
|           MQ|          18331|15.001680273066366| 9.890667945776498| 10.695641776641581|    49069
3| 3.7357370086795614|
|           OH|           6462|  20.4775193208208|11.817467683998007| 11.536153117856601|    19760
7| 3.2701270703973035|
|           HA|            570| 33.07478976561102|1.2644089394236424| 0.4552013450206487|     6182
6|  0.921942224953903|
|           XE|           9992|13.859438116586826|10.635404800035221| 11.395866476493499|    37451
0|  2.668019545539505|
|           AS|           2139|15.960019874130507| 4.804346362093581|  6.848722010417226|    15110
2| 1.4156000582388057|
|           FL|           2236| 7.660307649810956|  9.09137538507922|  9.262713040260852|    26168
4| 0.8544656914446431|
|           CO|           3702|13.637260941585554|  10.9790372802913|  13.18522978602152|    29845
5|  1.240387998190682|
|           WN|          12389|10.288646095459985|  5.17967817300539| 10.383034750411133|   120175
4| 1.0309098201462197|
|           9E|           7100|19.864633213075326| 4.111134703250211|  6.765859659983622|    26220
8| 2.7077739809616794|
+-------------+---------------+------------------+------------------+-------------------+---------
-+--------------------+
```

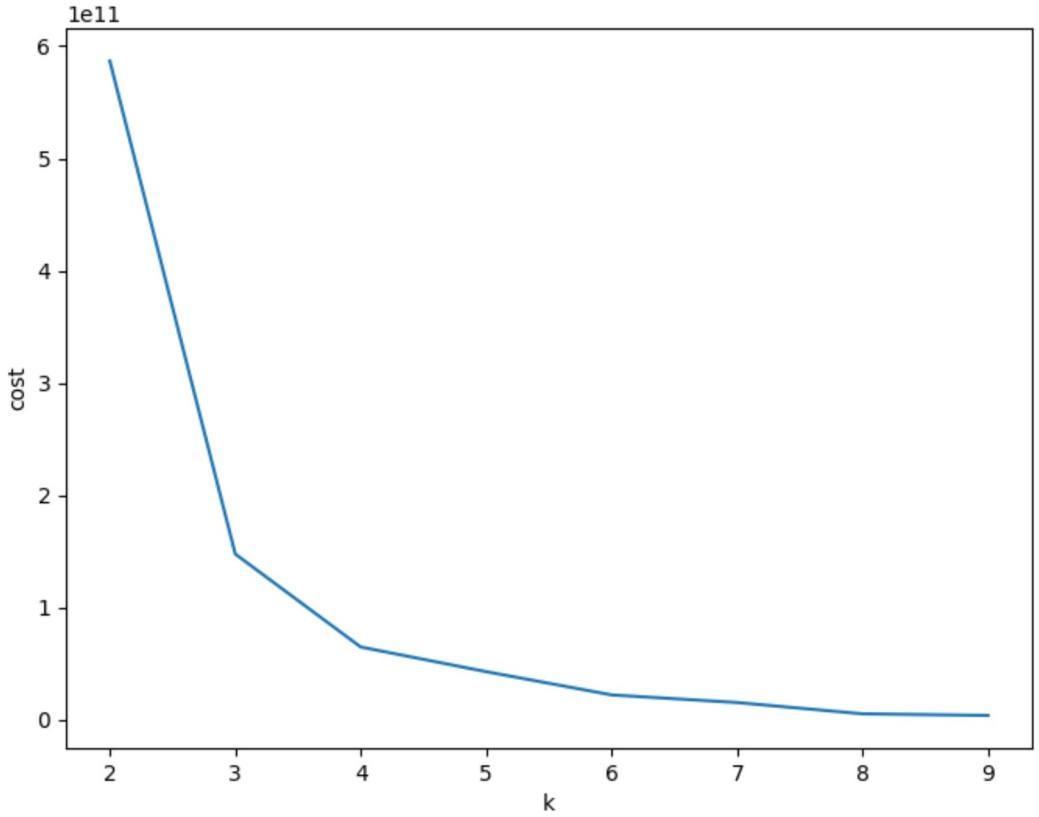
```
# Import the required libraries for the KMeans
from pyspark.ml.clustering import KMeans
from pyspark.ml.feature import VectorAssembler


# Vector assembler is used to create a vector of input features to be used in the clustering
assembler = VectorAssembler(inputCols=FEATURES_COL,
                            outputCol="features")

df_kmeans = assembler.transform(A1).select('UniqueCarrier', 'features')
df_kmeans.show()
```

```
+-------------+--------------------+
|UniqueCarrier|            features|
+-------------+--------------------+
|           UA|[14.8196808877248...|
|           AA|[17.3097642064972...|
|           NW|[19.1438136185863...|
|           EV|[25.3758977132726...|
|           B6|[13.7663490644617...|
|           DL|[14.2778227438284...|
|           OO|[15.9458026509572...|
|           F9|[12.7086793599351...|
|           YV|[32.5907016060862...|
|           US|[13.3762796474262...|
```

```
|           AQ|[22.1497461928934...|
|           MQ|[15.0016802730663...|
|           OH|[20.4775193208208...|
|           HA|[33.0747897656110...|
|           XE|[13.8594381165868...|
|           AS|[15.9600198741305...|
|           FL|[7.66030764981095...|
|           CO|[13.6372609415855...|
|           WN|[10.2886460954599...|
|           9E|[19.8646332130753...|
+-------------+--------------------+
```

```
# Optimize choice of K using the Elbow method to determine the optimal number of clusters for
k-means clustering
# To optimize k we cluster a fraction of the data for different choices of k and look for an
"elbow" in the cost function.
import numpy as np
cost = np.zeros(10)
for k in range(2,10):
    kmeans = KMeans().setK(k).setSeed(1).setFeaturesCol("features")
    model = kmeans.fit(df_kmeans)
    cost[k] = model.computeCost(df_kmeans) # requires Spark 2.0 or later
```

```
# Show the elbow graph to identify the optimum K
import numpy as np
import matplotlib.mlab as mlab
import matplotlib.pyplot as plt
import seaborn as sbs
from matplotlib.ticker import MaxNLocator

fig, ax = plt.subplots(1,1, figsize =(8,6))
ax.plot(range(2,10),cost[2:10])
ax.set_xlabel('k')
ax.set_ylabel('cost')
ax.xaxis.set_major_locator(MaxNLocator(integer=True))
plt.show()
display()
```




```
# Based on the elbow plot, let choose 6 as the optimal K.
# Train the machine learning model
k = 6
kmeans = KMeans().setK(k).setSeed(1).setFeaturesCol("features")
model = kmeans.fit(df_kmeans)
centers = model.clusterCenters()

print("Cluster Centers: ")
for center in centers:
    print(center)
```

```
Cluster Centers:
[1.43688659e+01 7.97131582e+00 9.63336841e+00 4.61432000e+05
 2.25983330e+00]
[2.09733088e+01 2.32208202e+00 2.95643548e+00 7.91225000e+04
 7.98103313e-01]
[1.02886461e+01 5.17967817e+00 1.03830348e+01 1.20175400e+06
 1.03090982e+00]
[1.90532385e+01 9.86662618e+00 1.10466521e+01 2.50221429e+05
 2.15925831e+00]
[1.65016259e+01 9.00197196e+00 8.92955107e+00 3.61081000e+05
 1.75195646e+00]
[1.66277834e+01 9.60303939e+00 1.03686705e+01 5.86022000e+05
 2.53793793e+00]
```

```
# Get the final cost
wssse = model.computeCost(df_kmeans)
print("The within set sum of squared error of the mode is {}".format(wssse))
```

```
The within set sum of squared error of the mode is 22309731753.266678
```

```
# Assign clusters to the carriers using the transform method,
# which adds 'prediction' column to the dataframe. The prediction value is an integer between 0 and
k.
transformed = model.transform(df_kmeans).select('UniqueCarrier', 'prediction')
rows = transformed.collect()
df_pred = sqlContext.createDataFrame(rows)
df_pred.show()
```

```
+-------------+----------+
|UniqueCarrier|prediction|
+-------------+----------+
|           UA|         0|
|           AA|         5|
|           NW|         4|
|           EV|         3|
|           B6|         3|
|           DL|         0|
|           OO|         5|
|           F9|         1|
|           YV|         3|
|           US|         0|
|           AQ|         1|
|           MQ|         0|
|           OH|         3|
|           HA|         1|
|           XE|         4|
|           AS|         1|
|           FL|         3|
|           CO|         3|
|           WN|         2|
|           9E|         3|
+-------------+----------+
```

```
# Join the prediction with the original data
df_pred = df_pred.join(A1, 'UniqueCarrier')
df_pred.show()
```

```
+-------------+----------+---------------+------------------+------------------+------------------
-+----------+-------------------+
|UniqueCarrier|prediction|NumCancellation|   AvgCarrierDelay|       AvgArrDelay|        AvgDepDela
y|NumFlights|   PrgtCancellation|
+-------------+----------+---------------+------------------+------------------+------------------
-+----------+-------------------+
|           UA|         0|          10541| 14.81968088772486|11.291322186680183|  14.1125766123613
8|    449515|  2.344971802943172|
|           AA|         5|          17440|17.309764206497203|12.607194035713981| 13.28089826443791
2|    604885| 2.8831926729874273|
|           NW|         4|           2906| 19.14381361858634| 7.368539129929383|  6.46323565667083
3|    347652| 0.8358933646289969|
```

```
|          EV|         3|          5026|25.375897713272636|10.208002415713782| 11.92253797087146
2|    280575| 1.7913213935667824|
|          B6|         3|          3205|13.766349064461735| 11.08418439051813| 12.65339574812211
3|    196091| 1.6344452320606249|
|          DL|         0|          6813|14.277822743828409| 7.855163154883384|  8.00776557270256
4|    451931|  1.507531016903023|
|          OO|         5|         12436| 15.94580265095729| 6.598884736868734| 7.456442759261995
5|    567159| 2.1926831805543068|
|          F9|         1|           303|12.708679359935184| 6.108246666107523|  5.91960151683392
3|     95762|0.31640943171612956|
|          YV|         3|          9219| 32.59070160608622|11.775181433600808| 12.00067527987503
3|    254930|  3.616286823833994|
|          AQ|         1|            42|  22.1497461928934|-2.888673890608875|-1.397782933745810
8|      7800| 0.5384615384615384|
|          US|         0|          6582|13.376279647426236|2.8481100056939694|  5.71748967189390
7|    453589| 1.4510933907127377|
|          MQ|         0|         18331|15.001680273066366| 9.890667945776498| 10.69564177664158
1|    490693| 3.7357370086795614|
|          OH|         3|          6462|  20.4775193208208|11.817467683998007| 11.53615311785660
1|    197607| 3.2701270703973035|
|          HA|         1|           570| 33.07478976561102|1.2644089394236424| 0.455201345020648
7|     61826|  0.921942224953903|
|          XE|         4|          9992|13.859438116586826|10.635404800035221| 11.39586647649349
9|    374510|  2.668019545539505|
|          AS|         1|          2139|15.960019874130507| 4.804346362093581|  6.84872201041722
6|    151102| 1.4156000582388057|
|          CO|         3|          3702|13.637260941585554|  10.9790372802913|  13.1852297860215
2|    298455|  1.240387998190682|
|          FL|         3|          2236| 7.660307649810956|  9.09137538507922|  9.26271304026085
2|    261684| 0.8544656914446431|
|          WN|         2|         12389|10.288646095459985|  5.17967817300539| 10.38303475041113
3|   1201754| 1.0309098201462197|
|          9E|         3|          7100|19.864633213075326| 4.111134703250211|  6.76585965998362
2|    262208| 2.7077739809616794|
+------------+----------+---------------+------------------+------------------+------------------
-+----------+-------------------+
```

```
# Get carriers names, to do a better interpretation
carriers = spark.read.csv('/FileStore/tables/carriers.csv',inferSchema=True,header=True)
carriers = carriers.withColumnRenamed("Code", "UniqueCarrier")
df_pred = df_pred.join(carriers, 'UniqueCarrier').orderBy(["prediction"],ascending=[True])
display(df_pred)
```

UniqueCarrier	prediction	NumCancellation	AvgCarrierDelay	AvgArrDelay	AvgDepDelay	I
DL	0	6813	14.277822743828409	7.855163154883384	8.007765572702564	4
MQ	0	18331	15.001680273066366	9.890667945776498	10.695641776641581	4
US	0	6582	13.376279647426236	2.8481100056939694	5.717489671893907	4
UA	0	10541	14.81968088772486	11.291322186680183	14.11257661236138	4
F9	1	303	12.708679359935184	6.108246666107523	5.919601516833923	9
AS	1	2139	15.960019874130507	4.804346362093581	6.848722010417226	1
AQ	1	42	22.1497461928934	-2.888673890608875	-1.3977829337458108	7
HA	1	570	33.07478976561102	1.2644089394236424	0.4552013450206487	6

WN	2	12389	10.288646095459985	5.17967817300539	10.383034750411133	1
CO	3	3702	13.637260941585554	10.9790372802913	13.18522978602152	2
YV	3	9219	32.59070160608622	11.775181433600808	12.000675279875033	2
EV	3	5026	25.375897713272636	10.208002415713782	11.922537970871462	2
9E	3	7100	19.864633213075326	4.111134703250211	6.765859659983622	2
B6	3	3205	13.766349064461735	11.08418439051813	12.653395748122113	1
OH	3	6462	20.4775193208208	11.817467683998007	11.536153117856601	1
FL	3	2236	7.660307649810956	9.09137538507922	9.262713040260852	2
XE	4	9992	13.859438116586826	10.635404800035221	11.395866476493499	3

```
+----------+-----+
|prediction|count|
+----------+-----+
|         0|    4|
|         1|    4|
|         2|    1|
|         3|    7|
|         4|    2|
|         5|    2|
+----------+-----+
```

databricks Project (1)

Set Spark Context

```
from pyspark.sql import SparkSession

spark = SparkSession.builder.appName('project').getOrCreate()
```

Read input files

```
#1. Airline data for 2008 has details of Aircrafts from various airports in US along with the
information of Delay and Cancellation
#2. Airport codes and descriptions file has all airports around the world
#3. Airline codes and descriptions file has airline carrier codes and description

airline_df = spark.read.csv('/FileStore/tables/2008_csv-db05f.bz2', inferSchema=True, header=True)

airports_df= spark.read.csv('/FileStore/tables/airports.csv', inferSchema=True, header=True)

airlines_df = spark.read.csv('/FileStore/tables/carriers.csv', inferSchema=True, header=True)
```

View Dataframes and Join the Dataframes to get Airport des

```
display(airline_df)
```

Year	Month	DayofMonth	DayOfWeek	DepTime	CRSDepTime	ArrTime	CRSArrTime	UniqueC
2008	1	3	4	2003	1955	2211	2225	WN
2008	1	3	4	754	735	1002	1000	WN
2008	1	3	4	628	620	804	750	WN
2008	1	3	4	926	930	1054	1100	WN
2008	1	3	4	1829	1755	1959	1925	WN
2008	1	3	4	1940	1915	2121	2110	WN
2008	1	3	4	1937	1830	2037	1940	WN
2008	1	3	4	1039	1040	1132	1150	WN
2008	1	3	4	617	615	652	650	WN
2008	1	3	4	1620	1620	1639	1655	WN
2008	1	3	4	706	700	916	915	WN
2008	1	3	4	1644	1510	1845	1725	WN
2008	1	3	4	1426	1430	1426	1425	WN

2008	1	3	4	715	715	720	710	WN
2008	1	3	4	1702	1700	1651	1655	WN
2008	1	3	4	1029	1020	1021	1010	WN
2008	1	3	4	1452	1425	1640	1625	WN
2008	1	3	4	754	745	940	955	WN
2008	1	3	4	1323	1255	1526	1510	WN
2008	1	3	4	1416	1325	1512	1435	WN
2008	1	3	4	706	705	807	810	WN
2008	1	3	4	1657	1625	1754	1735	WN
2008	1	3	4	1900	1840	1956	1950	WN
2008	1	3	4	1039	1030	1133	1140	WN
2008	1	3	4	801	800	902	910	WN
2008	1	3	4	1520	1455	1619	1605	WN
2008	1	3	4	1422	1255	1657	1610	WN
2008	1	3	4	1954	1925	2239	2235	WN
2008	1	3	4	636	635	921	945	WN
2008	1	3	4	734	730	958	1020	WN
2008	1	3	4	2107	1945	2334	2230	WN
2008	1	3	4	1008	1005	1234	1255	WN
2008	1	3	4	712	710	953	1000	WN
2008	1	3	4	1312	1300	1546	1550	WN
2008	1	3	4	1449	1430	1715	1720	WN
2008	1	3	4	1634	1555	1859	1845	WN
2008	1	3	4	831	830	935	955	WN
2008	1	3	4	1812	1650	1927	1815	WN
2008	1	3	4	1127	1105	1235	1230	WN
2008	1	3	4	1424	1355	1531	1520	WN
2008	1	3	4	1326	1230	1559	1530	WN
2008	1	3	4	1749	1725	2019	2030	WN
2008	1	3	4	726	720	958	1020	WN
2008	1	3	4	646	640	929	955	WN
2008	1	3	4	1153	1140	1428	1440	WN
2008	1	3	4	1528	1510	1802	1810	WN
2008	1	3	4	634	635	907	935	WN
2008	1	3	4	831	830	1148	1140	WN
2008	1	3	4	1450	1435	1806	1745	WN
2008	1	3	4	2245	1730	2354	1850	WN
2008	1	3	4	615	615	724	735	WN
2008	1	3	4	1150	1145	1303	1305	WN
2008	1	3	4	2025	1940	2135	2100	WN
2008	1	3	4	1038	945	1314	1225	WN
2008	1	3	4	1900	1850	2123	2045	WN

2008	1	3	4	700	700	851	900	WN
2008	1	3	4	948	925	959	940	WN
2008	1	3	4	646	620	725	655	WN
2008	1	3	4	1110	1040	1136	1110	WN
2008	1	3	4	1535	1535	1603	1610	WN
2008	1	3	4	1919	1915	1942	1950	WN
2008	1	3	4	1053	1055	1245	1240	WN
2008	1	3	4	1433	1440	1623	1625	WN
2008	1	3	4	2015	2010	2158	2155	WN
2008	1	3	4	2139	2130	2244	2240	WN
2008	1	3	4	1500	1500	1602	1615	WN
2008	1	3	4	850	850	1000	1000	WN
2008	1	3	4	646	645	752	755	WN
2008	1	3	4	1221	1220	1328	1330	WN
2008	1	3	4	1738	1730	1841	1840	WN
2008	1	3	4	1813	1735	1936	1905	WN
2008	1	3	4	802	750	1001	955	WN
2008	1	3	4	1820	1825	1946	1955	WN
2008	1	3	4	821	820	953	945	WN
2008	1	3	4	1734	1650	1941	1905	WN
2008	1	3	4	712	700	926	915	WN
2008	1	3	4	1318	1310	1410	1400	WN
2008	1	3	4	958	900	1052	950	WN
2008	1	3	4	1859	1850	1950	1945	WN
2008	1	3	4	1538	1445	1753	1710	WN
2008	1	3	4	933	935	1151	1200	WN
2008	1	3	4	2248	2125	102	2345	WN
2008	1	3	4	1327	1230	1550	1500	WN
2008	1	3	4	624	625	846	850	WN
2008	1	3	4	1614	1600	1833	1825	WN
2008	1	3	4	1917	1915	2136	2140	WN
2008	1	3	4	1832	1655	148	30	WN
2008	1	3	4	1229	1155	1633	1555	WN
2008	1	3	4	1256	1240	1724	1720	WN
2008	1	3	4	2118	2015	144	50	WN
2008	1	3	4	905	850	1334	1330	WN
2008	1	3	4	1739	1640	114	25	WN
2008	1	3	4	906	905	1426	1430	WN
2008	1	3	4	816	815	1339	1340	WN
2008	1	3	4	1325	1240	1841	1810	WN
2008	1	3	4	1506	1440	2030	2010	WN
2008	1	3	4	2039	1930	155	55	WN

2008	1	3	4	924	920	1209	1210	WN
2008	1	3	4	1611	1535	1849	1825	WN
2008	1	3	4	1824	1715	117	25	WN

Showing the first 1000 rows.

```
display(airports_df)
```

iata	airport	city
00M	Thigpen	Bay Springs
00R	Livingston Municipal	Livingston
00V	Meadow Lake	Colorado Springs
01G	Perry-Warsaw	Perry
01J	Hilliard Airpark	Hilliard
01M	Tishomingo County	Belmont
02A	Gragg-Wade	Clanton
02C	Capitol	Brookfield
02G	Columbiana County	East Liverpool
03D	Memphis Memorial	Memphis
04M	Calhoun County	Pittsboro
04Y	Hawley Municipal	Hawley
05C	Griffith-Merrillville	Griffith
05F	Gatesville - City/County	Gatesville
05U	Eureka	Eureka
06A	Moton Municipal	Tuskegee
06C	Schaumburg	Chicago/Schaumburg
06D	Rolla Municipal	Rolla
06M	Eupora Municipal	Eupora
06N	Randall	Middletown
06U	Jackpot/Hayden	Jackpot
07C	Dekalb County	Auburn
07F	Gladewater Municipal	Gladewater
07G	Fitch H Beach	Charlotte
07K	Central City Municipal	Central City
08A	Wetumpka Municipal	Wetumpka
08D	Stanley Municipal	Stanley
08K	Harvard State	Harvard
08M	Carthage-Leake County	Carthage
09A	Butler-Choctaw County	Butler
09J	Jekyll Island	Jekyll Island
09K	Sargent Municipal	Sargent

09M	Charleston Municipal	Charleston
09W	South Capitol Street	Washington
0A3	Smithville Municipal	Smithville
0A8	Bibb County	Centreville
0A9	Elizabethton Municipal	Elizabethton
0AK	Pilot Station	Pilot Station
0B1	Col. Dyke	Bethel
0B4	Hartington Municipal	Hartington
0B5	Turners Falls	Montague
0B7	Warren-Sugar Bush	Warren
0B8	Elizabeth	Fishers Island
0C0	Dacy	Chicago/Harvard
0C4	Pender Municipal	Pender
0D1	South Haven Municipal	South Haven
0D8	Gettysburg Municipal	Gettysburg
0E0	Moriarty	Moriarty
0E8	Crownpoint	Crownpoint
0F2	Bowie Municipal	Bowie
0F4	Loup City Municipal	Loup City
0F7	Fountainhead Lodge Airpark	Eufaula
0F8	William R Pogue Municipal	Sand Springs
0F9	Tishomingo Airpark	Tishomingo
0G0	North Buffalo Suburban	Lockport
0G3	Tecumseh Municipal	Tecumseh
0G6	Williams County	Bryan
0G7	Finger Lakes Regional	Seneca Falls
0H1	Trego Wakeeney	Wakeeney
0I8	Cynthiana-Harrison County	Cynthiana
0J0	Abbeville Municipal	Abbeville
0J4	Florala Municipal	Florala
0J6	Headland Municipal	Headland
0K7	Humboldt Municipal	Humboldt
0L5	Goldfield	Goldfield
0L7	Jean	Jean
0L9	Echo Bay	Overton
0M0	Dumas Municipal	Dumas
0M1	Scott	Parsons
0M4	Benton County	Camden
0M5	Humphreys County	Waverly
0M6	Panola County	Batesville
0M8	Byerley	Lake Providence
0O3	Calaveras Co-Maury Rasmussen	San Andreas

0O4	Corning Municipal	Corning
0O5	University	Davis
0Q5	Shelter Cove	Shelter Cove
0Q6	Shingletown	Shingletown
0R0	Columbia-Marion County	Columbia
0R1	Atmore Municipal	Atmore
0R3	Abbeville Chris Crusta Memorial	Abbeville
0R4	Concordia Parish	Vidalia
0R5	David G Joyce	Winnfield

Showing the first 1000 rows.

```
display(airlines_df)
```

Code	Description
02Q	Titan Airways
04Q	Tradewind Aviation
05Q	Comlux Aviation, AG
06Q	Master Top Linhas Aereas Ltd.
07Q	Flair Airlines Ltd.
09Q	Swift Air, LLC
0BQ	DCA
0CQ	ACM AIR CHARTER GmbH
0FQ	Maine Aviation Aircraft Charter, LLC
0GQ	Inter Island Airways, d/b/a Inter Island Air
0HQ	Polar Airlines de Mexico d/b/a Nova Air
0J	JetClub AG
0JQ	Vision Airlines
0KQ	Mokulele Flight Services, Inc.
0LQ	Metropix UK, LLP.
0MQ	Multi-Aero, Inc. d/b/a Air Choice One
0Q	Flying Service N.V.
16	PSA Airlines Inc.
17	Piedmont Airlines
1I	Sky Trek Int'l Airlines
2E	Smokey Bay Air Inc.
2F	Frontier Flying Service
2M	Midway Express Airlines
2O	Island Air Service
2R	Regal Air
2T	Canada 3000 Airlines Ltd.
2U	Valley Air Express Inc.

37	Zeal 320
3C	Regions Air, Inc.
3F	Pacific Airways, Inc.
3M	Gulfstream Int
3Z	Tatonduk Flying Service
4B	Olson Air Service
4E	Tanana Air Service
4E (1)	British Airtours Limited
4H	Belize Trans Air
4M	LAN Argentina
4M (1)	Lan Dominica
4N	Air North
4R	Regent Air Corporation
4S	Sol Air (Aero Hunduras)
4S (1)	Conner Air Lines Inc.
4T	Belair Airlines Ltd.
4W	Warbelow
4Y	Yute Air Aka Flight Alaska
5B	Bellair Inc.
5C	C.A.L Cargo Airlines
5D	Aerolitoral
5F	Arctic Circle Air Service
5G	Skyservice Airlines, Inc.
5G (1)	Queen Air
5J	Private Jet Expeditions
5X	United Parcel Service
5Y	Atlas Air Inc.
6A	Aviacsa Airlines
6B	Britannia Airways A.B.
6C	Cape Smythe Air Service
6F	Laker Airways Inc.
6H	Israir Airlines
6P	Pacific East Air Inc.
6R	Aerounion Aerotransporte de Carga Union SA de CV
6U	Air Ukraine
6Y	Nicaraguense De Aviacion Sa
7B	Krasair
7F	First Air
7G	MK Airlines Ltd.
7G (1)	Bellair Inc. (1)
7H	Era Aviation
7M	Aeromonterrey S.A.

7N	Inland Aviation Services
7P	Apa International Air S.A.
7S	Arctic Transportation
7S (1)	Skystar International Inc.
7Z	Lb Limited
8C	Air Transport International
8D	Servant Air Inc.
8E	Bering Air Inc.
8F	Wilburs Inc

Showing the first 1000 rows.

```
airline_df_1=airline_df.join(airports_df, airline_df.Origin==airports_df.iata)
```

```
airline_df_1=airline_df_1.withColumnRenamed('airport', 'Origin_airport').\
                          withColumnRenamed('iata', 'Origin_iata').\
                          select('Year', 'Month', 'DayofMonth', 'DayofWeek', 'DepTime', 'ArrTime',
'UniqueCarrier', 'FlightNum', 'TailNum', 'ActualElapsedTime', 'AirTime', 'ArrDelay', 'DepDelay',
'Origin', 'Dest', 'Distance', 'Cancelled', 'CancellationCode', 'Diverted', 'Origin_airport')
```

```
display(airline_df_1)
```

Year	Month	DayofMonth	DayofWeek	DepTime	ArrTime	UniqueCarrier	FlightNum	TailNum
2008	1	3	4	2003	2211	WN	335	N712SW
2008	1	3	4	754	1002	WN	3231	N772SW
2008	1	3	4	628	804	WN	448	N428WN
2008	1	3	4	926	1054	WN	1746	N612SW
2008	1	3	4	1829	1959	WN	3920	N464WN
2008	1	3	4	1940	2121	WN	378	N726SW
2008	1	3	4	1937	2037	WN	509	N763SW
2008	1	3	4	1039	1132	WN	535	N428WN
2008	1	3	4	617	652	WN	11	N689SW
2008	1	3	4	1620	1639	WN	810	N648SW

2008	1	3	4	706	916	WN	100	N690SW
2008	1	3	4	1644	1845	WN	1333	N334SW
2008	1	3	4	1426	1426	WN	829	N476WN
2008	1	3	4	715	720	WN	1016	N765SW
2008	1	3	4	1702	1651	WN	1827	N420WN
2008	1	3	4	1029	1021	WN	2272	N263WN
2008	1	3	4	1452	1640	WN	675	N286WN
2008	1	3	4	754	940	WN	1144	N778SW
2008	1	3	4	1323	1526	WN	4	N674AA
2008	1	3	4	1416	1512	WN	54	N643SW
2008	1	3	4	706	807	WN	68	N497WN
2008	1	3	4	1657	1754	WN	623	N724SW
2008	1	3	4	1900	1956	WN	717	N786SW
2008	1	3	4	1039	1133	WN	1244	N714CB
2008	1	3	4	801	902	WN	2101	N222WN
2008	1	3	4	1520	1619	WN	2553	N394SW
2008	1	3	4	1422	1657	WN	188	N215WN
2008	1	3	4	1954	2239	WN	1754	N243WN
2008	1	3	4	636	921	WN	2275	N454WN
2008	1	3	4	734	958	WN	550	N712SW
2008	1	3	4	2107	2334	WN	362	N798SW
2008	1	3	4	1008	1234	WN	543	N736SA
2008	1	3	4	712	953	WN	1112	N795SW
2008	1	3	4	1312	1546	WN	1397	N247WN

2008	1	3	4	1449	1715	WN	3398	N707SA
2008	1	3	4	1634	1859	WN	3480	N443WN
2008	1	3	4	831	935	WN	300	N753SW
2008	1	3	4	1812	1927	WN	422	N779SW
2008	1	3	4	1127	1235	WN	1837	N704SW
2008	1	3	4	1424	1531	WN	2871	N709SW
2008	1	3	4	1326	1559	WN	1056	N459WN
2008	1	3	4	1749	2019	WN	2175	N621SW
2008	1	3	4	726	958	WN	3319	N206WN
2008	1	3	4	646	929	WN	3667	N280WN
2008	1	3	4	1153	1428	WN	2006	N241WN
2008	1	3	4	1528	1802	WN	3858	N200WN
2008	1	3	4	634	907	WN	3928	N459WN
2008	1	3	4	831	1148	WN	534	N286WN
2008	1	3	4	1450	1806	WN	3244	N475WN
2008	1	3	4	2245	2354	WN	186	N792SW
2008	1	3	4	615	724	WN	971	N202WN
2008	1	3	4	1150	1303	WN	2124	N646SW
2008	1	3	4	2025	2135	WN	3154	N252WN
2008	1	3	4	1038	1314	WN	1035	N346SW
2008	1	3	4	1900	2123	WN	205	N299WN
2008	1	3	4	700	851	WN	449	N528SW
2008	1	3	4	948	959	WN	3430	N487WN
2008	1	3	4	646	725	WN	1580	N243WN

2008	1	3	4	1110	1136	WN	2195	N479WN
2008	1	3	4	1535	1603	WN	2804	N255WN
2008	1	3	4	1919	1942	WN	3428	N215WN
2008	1	3	4	1053	1245	WN	433	N264LV
2008	1	3	4	1433	1623	WN	1331	N714CB
2008	1	3	4	2015	2158	WN	3504	N436WN
2008	1	3	4	2139	2244	WN	378	N726SW
2008	1	3	4	1500	1602	WN	640	N399WN
2008	1	3	4	850	1000	WN	1396	N387SW
2008	1	3	4	646	752	WN	2189	N405WN
2008	1	3	4	1221	1328	WN	3312	N685SW
2008	1	3	4	1738	1841	WN	3948	N467WN
2008	1	3	4	1813	1936	WN	54	N643SW
2008	1	3	4	802	1001	WN	2272	N263WN
2008	1	3	4	1820	1946	WN	549	N363SW
2008	1	3	4	821	953	WN	3604	N257WN
2008	1	3	4	1734	1941	WN	23	N521SW
2008	1	3	4	712	926	WN	1232	N663SW
2008	1	3	4	1318	1410	WN	977	N376SW
2008	1	3	4	958	1052	WN	1574	N791SW
2008	1	3	4	1859	1950	WN	2019	N392SW
2008	1	3	4	1538	1753	WN	500	N799SW
2008	1	3	4	933	1151	WN	778	N607SW
2008	1	3	4	2248	102	WN	890	N618WN

2008	1	3	4	1327	1550	WN	1171	N682SW
2008	1	3	4	624	846	WN	1320	N456WN
2008	1	3	4	1614	1833	WN	1925	N509SW
2008	1	3	4	1917	2136	WN	2457	N293
2008	1	3	4	1832	148	WN	302	N473WN

Showing the first 1000 rows.

```
airline_df_2=airline_df_1.join(airlines_df, airline_df_1.UniqueCarrier==airlines_df.Code).\
              select('Year', 'Month', 'DayofMonth', 'DayofWeek', 'DepTime', 'ArrTime',
'UniqueCarrier', 'FlightNum', 'TailNum', 'ActualElapsedTime', 'AirTime', 'ArrDelay', 'DepDelay',
'Origin', 'Dest', 'Distance', 'Cancelled', 'CancellationCode', 'Diverted', 'Origin_airport',
'Description')
```

```
display(airline_df_2)
```

Year	Month	DayofMonth	DayofWeek	DepTime	ArrTime	UniqueCarrier	FlightNum	TailNum
2008	1	3	4	2003	2211	WN	335	N712SW
2008	1	3	4	754	1002	WN	3231	N772SW
2008	1	3	4	628	804	WN	448	N428WN
2008	1	3	4	926	1054	WN	1746	N612SW
2008	1	3	4	1829	1959	WN	3920	N464WN
2008	1	3	4	1940	2121	WN	378	N726SW
2008	1	3	4	1937	2037	WN	509	N763SW
2008	1	3	4	1039	1132	WN	535	N428WN
2008	1	3	4	617	652	WN	11	N689SW
2008	1	3	4	1620	1639	WN	810	N648SW
2008	1	3	4	706	916	WN	100	N690SW
2008	1	3	4	1644	1845	WN	1333	N334SW

2008	1	3	4	1426	1426	WN	829	N476WN
2008	1	3	4	715	720	WN	1016	N765SW
2008	1	3	4	1702	1651	WN	1827	N420WN
2008	1	3	4	1029	1021	WN	2272	N263WN
2008	1	3	4	1452	1640	WN	675	N286WN
2008	1	3	4	754	940	WN	1144	N778SW
2008	1	3	4	1323	1526	WN	4	N674AA
2008	1	3	4	1416	1512	WN	54	N643SW
2008	1	3	4	706	807	WN	68	N497WN
2008	1	3	4	1657	1754	WN	623	N724SW
2008	1	3	4	1900	1956	WN	717	N786SW
2008	1	3	4	1039	1133	WN	1244	N714CB
2008	1	3	4	801	902	WN	2101	N222WN
2008	1	3	4	1520	1619	WN	2553	N394SW
2008	1	3	4	1422	1657	WN	188	N215WN
2008	1	3	4	1954	2239	WN	1754	N243WN
2008	1	3	4	636	921	WN	2275	N454WN
2008	1	3	4	734	958	WN	550	N712SW
2008	1	3	4	2107	2334	WN	362	N798SW
2008	1	3	4	1008	1234	WN	543	N736SA
2008	1	3	4	712	953	WN	1112	N795SW
2008	1	3	4	1312	1546	WN	1397	N247WN
2008	1	3	4	1449	1715	WN	3398	N707SA
2008	1	3	4	1634	1859	WN	3480	N443WN

2008	1	3	4	831	935	WN	300	N753SW
2008	1	3	4	1812	1927	WN	422	N779SW
2008	1	3	4	1127	1235	WN	1837	N704SW
2008	1	3	4	1424	1531	WN	2871	N709SW
2008	1	3	4	1326	1559	WN	1056	N459WN
2008	1	3	4	1749	2019	WN	2175	N621SW
2008	1	3	4	726	958	WN	3319	N206WN
2008	1	3	4	646	929	WN	3667	N280WN
2008	1	3	4	1153	1428	WN	2006	N241WN
2008	1	3	4	1528	1802	WN	3858	N200WN
2008	1	3	4	634	907	WN	3928	N459WN
2008	1	3	4	831	1148	WN	534	N286WN
2008	1	3	4	1450	1806	WN	3244	N475WN
2008	1	3	4	2245	2354	WN	186	N792SW
2008	1	3	4	615	724	WN	971	N202WN
2008	1	3	4	1150	1303	WN	2124	N646SW
2008	1	3	4	2025	2135	WN	3154	N252WN
2008	1	3	4	1038	1314	WN	1035	N346SW
2008	1	3	4	1900	2123	WN	205	N299WN
2008	1	3	4	700	851	WN	449	N528SW
2008	1	3	4	948	959	WN	3430	N487WN
2008	1	3	4	646	725	WN	1580	N243WN
2008	1	3	4	1110	1136	WN	2195	N479WN
2008	1	3	4	1535	1603	WN	2804	N255WN

2008	1	3	4	1919	1942	WN	3428	N215WN
2008	1	3	4	1053	1245	WN	433	N264LV
2008	1	3	4	1433	1623	WN	1331	N714CB
2008	1	3	4	2015	2158	WN	3504	N436WN
2008	1	3	4	2139	2244	WN	378	N726SW
2008	1	3	4	1500	1602	WN	640	N399WN
2008	1	3	4	850	1000	WN	1396	N387SW
2008	1	3	4	646	752	WN	2189	N405WN
2008	1	3	4	1221	1328	WN	3312	N685SW
2008	1	3	4	1738	1841	WN	3948	N467WN
2008	1	3	4	1813	1936	WN	54	N643SW
2008	1	3	4	802	1001	WN	2272	N263WN
2008	1	3	4	1820	1946	WN	549	N363SW
2008	1	3	4	821	953	WN	3604	N257WN
2008	1	3	4	1734	1941	WN	23	N521SW
2008	1	3	4	712	926	WN	1232	N663SW
2008	1	3	4	1318	1410	WN	977	N376SW
2008	1	3	4	958	1052	WN	1574	N791SW
2008	1	3	4	1859	1950	WN	2019	N392SW
2008	1	3	4	1538	1753	WN	500	N799SW
2008	1	3	4	933	1151	WN	778	N607SW
2008	1	3	4	2248	102	WN	890	N618WN
2008	1	3	4	1327	1550	WN	1171	N682SW
2008	1	3	4	624	846	WN	1320	N456WN

2008	1	3	4	1614	1833	WN	1925	N509SW
2008	1	3	4	1917	2136	WN	2457	N293
2008	1	3	4	1832	148	WN	302	N473WN
2008	1	3	4	1229	1633	WN	1079	N351SW
2008	1	3	4	1256	1724	WN	155	N238WN

Showing the first 1000 rows.

Display Cancellations by Origin, Airline, Cancellation Code

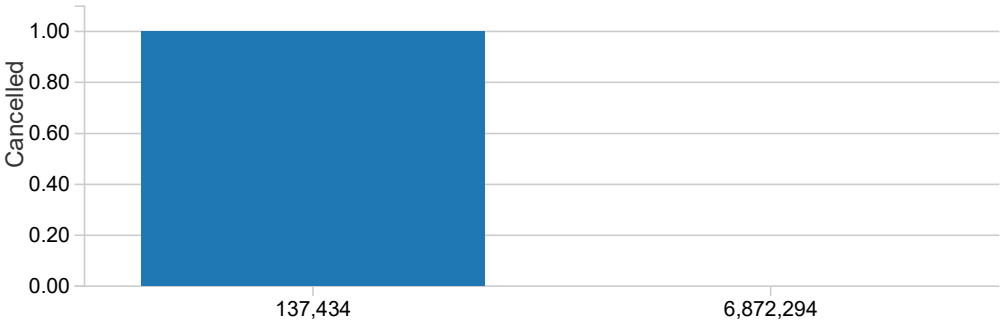
```
from pyspark.sql import functions as F
from pyspark.sql.window import Window


display(airline_df_2.filter("Cancelled=='1'").groupby('Cancelled').count().\
       withColumn('count', F.format_number('count',0)).\
       withColumnRenamed('count', 'Count of Cancelled'))
```

Cancelled	Count of Cancelled
1	137,434

```
display(airline_df_2.filter("Cancelled=='0'").groupby('Cancelled').count().\
       withColumn('count', F.format_number('count',0)).\
       withColumnRenamed('count', 'Count of Not Cancelled'))
```

Cancelled	Count of Not Cancelled
0	6,872,294

```
tmp_1=airline_df_2.groupby('Cancelled').count().\
       withColumn('count', F.format_number('count',0)).\
       select('Cancelled', 'count')
display(tmp_1)
```




```
display(airline_df_2.filter("Cancelled=='1'").groupby('Origin','Origin_airport').count().sort('coun
t', ascending=False).\
       withColumn('count', F.format_number('count', 0)).\
       withColumnRenamed('count', 'Count of Cancellation by Origin Airport'))
```

Origin	Origin_airport	(
ORD	Chicago O'Hare International	1
DFW	Dallas-Fort Worth International	7
ATL	William B Hartsfield-Atlanta Intl	5
LGA	LaGuardia	5
EWR	Newark Intl	4
BOS	Gen Edw L Logan Intl	3
IAH	George Bush Intercontinental	3
JFK	John F Kennedy Intl	3
LAX	Los Angeles International	2
SFO	San Francisco International	2
DCA	Ronald Reagan Washington National	2
DEN	Denver Intl	2
DTW	Detroit Metropolitan-Wayne County	2
IAD	Washington Dulles International	2
LAS	McCarran International	2
CLT	Charlotte/Douglas International	1
PHL	Philadelphia Intl	1
PHX	Phoenix Sky Harbor International	1
CVG	Cincinnati Northern Kentucky Intl	1
MSP	Minneapolis-St Paul Intl	1
HOU	William P Hobby	1
MEM	Memphis International	1
CLE	Cleveland-Hopkins Intl	1
STL	Lambert-St Louis International	1
RDU	Raleigh-Durham International	1
SAN	San Diego International-Lindbergh	1

MDW	Chicago Midway	1
MKE	General Mitchell International	1
SEA	Seattle-Tacoma Intl	1
SLC	Salt Lake City Intl	1
MCO	Orlando International	1
BWI	Baltimore-Washington International	1
DAL	Dallas Love	9
OAK	Metropolitan Oakland International	9
MCI	Kansas City International	8
MSY	New Orleans International	8
SJC	San Jose International	8
BNA	Nashville International	7
MIA	Miami International	7
CMH	Port Columbus Intl	7
PIT	Pittsburgh International	7
IND	Indianapolis International	7
AUS	Austin-Bergstrom International	7
TPA	Tampa International	7
SNA	John Wayne /Orange Co	7
PDX	Portland Intl	6
MSN	Dane County Regional	6
BUF	Buffalo Niagara Intl	6
BDL	Bradley International	5
DSM	Des Moines International	5
SAT	San Antonio International	5
FLL	Fort Lauderdale-Hollywood Int'l	5
BUR	Burbank-Glendale-Pasadena	5
SMF	Sacramento International	5
RIC	Richmond International	4
JAX	Jacksonville International	4
ASE	Aspen-Pitkin Co/Sardy	4
OMA	Eppley Airfield	4
GRR	Kent County International	4
XNA	Northwest Arkansas Regional	4
HNL	Honolulu International	4
CID	Eastern Iowa	4
GSO	Piedmont Triad International	4
ICT	Wichita Mid-Continent	4
ROC	Greater Rochester Int'l	4
SGF	Springfield-Branson Regional	4
GSP	Greenville-Spartanburg	4
CHS	Charleston AFB/International	4

SDF	Louisville International-Standiford	4
OKC	Will Rogers World	3
CAE	Columbia Metropolitan	3
BHM	Birmingham International	3
PVD	Theodore F Green State	3
GRB	Austin Straubel International	3
DAY	James M Cox Dayton Intl	3
SAV	Savannah International	3
TUL	Tulsa International	3

```
display(airline_df_2.filter("Cancelled=='1'").groupby('UniqueCarrier','Description').count().sort('
count', ascending=False).\
       withColumn('count', F.format_number('count', 0)).\
       withColumnRenamed('count', 'Count of Cancellation by Airline'))
```

UniqueCarrier	Description
MQ	American Eagle Airlines Inc.
AA	American Airlines Inc.
OO	Skywest Airlines Inc.
WN	Southwest Airlines Co.
UA	United Air Lines Inc.
XE	Expressjet Airlines Inc.
YV	Mesa Airlines Inc.
9E	Pinnacle Airlines Inc.
DL	Delta Air Lines Inc.

```
display(airline_df_2.filter("Cancelled=='1'").groupby('Origin_airport','Description').count().sort(
'count', ascending=False).\
       withColumn('count', F.format_number('count', 0)).\
       withColumnRenamed('count', 'Count of Cancellation by Origin Airport and Airline'))
```

Origin_airport	Description
Dallas-Fort Worth International	American Airlines Inc.
Chicago O'Hare International	American Eagle Airlines Inc.
Chicago O'Hare International	American Airlines Inc.
Chicago O'Hare International	United Air Lines Inc.
Newark Intl	Expressjet Airlines Inc.
Chicago O'Hare International	Skywest Airlines Inc.
Dallas-Fort Worth International	American Eagle Airlines Inc.
William B Hartsfield-Atlanta Intl	Atlantic Southeast Airlines
William B Hartsfield-Atlanta Intl	Delta Air Lines Inc.
George Bush Intercontinental	Expressjet Airlines Inc.

Chicago O'Hare International	Mesa Airlines Inc.
William P Hobby	Southwest Airlines Co.
Detroit Metropolitan-Wayne County	Pinnacle Airlines Inc.
Denver Intl	United Air Lines Inc.
LaGuardia	Delta Air Lines Inc.
LaGuardia	American Airlines Inc.
John F Kennedy Intl	JetBlue Airways
Chicago Midway	Southwest Airlines Co.
John F Kennedy Intl	Comair Inc.
Washington Dulles International	Mesa Airlines Inc.
George Bush Intercontinental	Continental Air Lines Inc.
McCarran International	Southwest Airlines Co.
San Francisco International	Skywest Airlines Inc.
San Francisco International	United Air Lines Inc.
Memphis International	Pinnacle Airlines Inc.
Gen Edw L Logan Intl	US Airways Inc. (Merged with America West 9/05. Reporting for both starting 1
Los Angeles International	Southwest Airlines Co.
LaGuardia	American Eagle Airlines Inc.
LaGuardia	Comair Inc.
Cincinnati Northern Kentucky Intl	Comair Inc.
Newark Intl	Continental Air Lines Inc.
Philadelphia Intl	US Airways Inc. (Merged with America West 9/05. Reporting for both starting 1
Phoenix Sky Harbor International	Southwest Airlines Co.
Metropolitan Oakland International	Southwest Airlines Co.
Detroit Metropolitan-Wayne County	Northwest Airlines Inc.
William B Hartsfield-Atlanta Intl	AirTran Airways Corporation
Charlotte/Douglas International	US Airways Inc. (Merged with America West 9/05. Reporting for both starting 1
Charlotte/Douglas International	Mesa Airlines Inc.
Dallas Love	Southwest Airlines Co.
Lambert-St Louis International	American Airlines Inc.
LaGuardia	US Airways Inc. (Merged with America West 9/05. Reporting for both starting 1
Ronald Reagan Washington National	US Airways Inc. (Merged with America West 9/05. Reporting for both starting 1
Salt Lake City Intl	Skywest Airlines Inc.
General Mitchell International	Skywest Airlines Inc.
Denver Intl	Skywest Airlines Inc.
Los Angeles International	United Air Lines Inc.
Minneapolis-St Paul Intl	Pinnacle Airlines Inc.
Gen Edw L Logan Intl	Delta Air Lines Inc.
Ronald Reagan Washington National	Delta Air Lines Inc.
Miami International	American Airlines Inc.
San Diego International-Lindbergh	Southwest Airlines Co.
Seattle-Tacoma Intl	Alaska Airlines Inc.

Los Angeles International	American Airlines Inc.
Washington Dulles International	United Air Lines Inc.
Cleveland-Hopkins Intl	Expressjet Airlines Inc.
Minneapolis-St Paul Intl	Northwest Airlines Inc.
LaGuardia	United Air Lines Inc.
Gen Edw L Logan Intl	Comair Inc.
McCarran International	US Airways Inc. (Merged with America West 9/05. Reporting for both starting 1
Raleigh-Durham International	American Eagle Airlines Inc.
Phoenix Sky Harbor International	Mesa Airlines Inc.
Los Angeles International	Skywest Airlines Inc.
John F Kennedy Intl	American Eagle Airlines Inc.
Ronald Reagan Washington National	American Eagle Airlines Inc.
Phoenix Sky Harbor International	US Airways Inc. (Merged with America West 9/05. Reporting for both starting 1
Gen Edw L Logan Intl	American Airlines Inc.
Gen Edw L Logan Intl	American Eagle Airlines Inc.
San Jose International	Southwest Airlines Co.
Denver Intl	Mesa Airlines Inc.
Cincinnati Northern Kentucky Intl	American Eagle Airlines Inc.
San Francisco International	American Airlines Inc.
Aspen-Pitkin Co/Sardy	Skywest Airlines Inc.
Philadelphia Intl	Southwest Airlines Co.
Orlando International	Southwest Airlines Co.
Gen Edw L Logan Intl	JetBlue Airways
William B Hartsfield-Atlanta Intl	American Airlines Inc.
Burbank-Glendale-Pasadena	Southwest Airlines Co.
Ronald Reagan Washington National	United Air Lines Inc.
San Francisco International	Southwest Airlines Co.
Gen Edw L Logan Intl	United Air Lines Inc.
Austin-Bergstrom International	American Airlines Inc.
Ronald Reagan Washington National	American Airlines Inc.
Newark Intl	American Airlines Inc.
Baltimore-Washington International	Southwest Airlines Co.
John F Kennedy Intl	American Airlines Inc.
Honolulu International	Hawaiian Airlines Inc.
Dane County Regional	American Eagle Airlines Inc.
Northwest Arkansas Regional	American Eagle Airlines Inc.
Indianapolis International	Pinnacle Airlines Inc.
Ted Stevens Anchorage International	Alaska Airlines Inc.
William B Hartsfield-Atlanta Intl	Comair Inc.
New Orleans International	Southwest Airlines Co.
LaGuardia	AirTran Airways Corporation
Lambert-St Louis International	Southwest Airlines Co.

Newark Intl	United Air Lines Inc.
Cincinnati Northern Kentucky Intl	Delta Air Lines Inc.
Cleveland-Hopkins Intl	American Eagle Airlines Inc.
Port Columbus Intl	American Eagle Airlines Inc.
Eastern Iowa	American Eagle Airlines Inc.
Chicago O'Hare International	Northwest Airlines Inc.

Showing the first 1000 rows.

```
display(airline_df_2.filter("Cancelled=='1'").groupby('CancellationCode',
'Origin_airport','Description').count().sort('count', ascending=False).\
       withColumn('count', F.format_number('count', 0)).\
       withColumnRenamed('count', 'Count of Cancellation by Cancellation Code'))
```

CancellationCode	Origin_airport	Description
A	Dallas-Fort Worth International	American Airlines Inc.
C	Chicago O'Hare International	American Eagle Airlines Inc.
B	Chicago O'Hare International	American Eagle Airlines Inc.
B	Dallas-Fort Worth International	American Airlines Inc.
C	Newark Intl	Expressjet Airlines Inc.
B	Dallas-Fort Worth International	American Eagle Airlines Inc.
C	Chicago O'Hare International	Skywest Airlines Inc.
B	George Bush Intercontinental	Expressjet Airlines Inc.
A	Chicago O'Hare International	United Air Lines Inc.
A	Chicago O'Hare International	American Airlines Inc.
B	Chicago O'Hare International	United Air Lines Inc.
B	William B Hartsfield-Atlanta Intl	Atlantic Southeast Airlines
B	John F Kennedy Intl	JetBlue Airways
B	John F Kennedy Intl	Comair Inc.
B	Chicago O'Hare International	American Airlines Inc.
B	William P Hobby	Southwest Airlines Co.
A	Denver Intl	United Air Lines Inc.
B	George Bush Intercontinental	Continental Air Lines Inc.
A	William B Hartsfield-Atlanta Intl	Delta Air Lines Inc.
C	Chicago O'Hare International	American Airlines Inc.
C	Chicago O'Hare International	Mesa Airlines Inc.
B	Newark Intl	Expressjet Airlines Inc.
B	William B Hartsfield-Atlanta Intl	Delta Air Lines Inc.
B	Chicago Midway	Southwest Airlines Co.
A	San Francisco International	United Air Lines Inc.
A	Detroit Metropolitan-Wayne County	Pinnacle Airlines Inc.
B	LaGuardia	Comair Inc.
A	Los Angeles International	Southwest Airlines Co.

A	Washington Dulles International	Mesa Airlines Inc.
A	Chicago O'Hare International	Mesa Airlines Inc.
B	Cincinnati Northern Kentucky Intl	Comair Inc.
C	William B Hartsfield-Atlanta Intl	Atlantic Southeast Airlines
A	McCarran International	Southwest Airlines Co.
A	Metropolitan Oakland International	Southwest Airlines Co.
A	Phoenix Sky Harbor International	Southwest Airlines Co.
B	Newark Intl	Continental Air Lines Inc.
C	Chicago O'Hare International	United Air Lines Inc.
A	Memphis International	Pinnacle Airlines Inc.
A	William P Hobby	Southwest Airlines Co.
A	Los Angeles International	United Air Lines Inc.
C	San Francisco International	Skywest Airlines Inc.
C	LaGuardia	American Eagle Airlines Inc.
A	Charlotte/Douglas International	Mesa Airlines Inc.
A	Dallas-Fort Worth International	American Eagle Airlines Inc.
B	Chicago O'Hare International	Skywest Airlines Inc.
A	LaGuardia	American Airlines Inc.
B	Salt Lake City Intl	Skywest Airlines Inc.
A	Gen Edw L Logan Intl	US Airways Inc. (Merged with America West 9/05. Re
A	Charlotte/Douglas International	US Airways Inc. (Merged with America West 9/05. Re
A	LaGuardia	Delta Air Lines Inc.
A	William B Hartsfield-Atlanta Intl	Atlantic Southeast Airlines
C	LaGuardia	Delta Air Lines Inc.
A	Chicago O'Hare International	American Eagle Airlines Inc.
B	Detroit Metropolitan-Wayne County	Pinnacle Airlines Inc.
A	Dallas Love	Southwest Airlines Co.
A	Los Angeles International	American Airlines Inc.
A	Washington Dulles International	United Air Lines Inc.
A	Philadelphia Intl	US Airways Inc. (Merged with America West 9/05. Re
B	Gen Edw L Logan Intl	Comair Inc.
C	LaGuardia	American Airlines Inc.
B	Cleveland-Hopkins Intl	Expressjet Airlines Inc.
A	Ronald Reagan Washington National	US Airways Inc. (Merged with America West 9/05. Re
B	General Mitchell International	Skywest Airlines Inc.
A	San Diego International-Lindbergh	Southwest Airlines Co.
B	Denver Intl	Skywest Airlines Inc.
A	Minneapolis-St Paul Intl	Pinnacle Airlines Inc.
A	Lambert-St Louis International	American Airlines Inc.
A	Chicago Midway	Southwest Airlines Co.
A	Miami International	American Airlines Inc.
A	Chicago O'Hare International	Skywest Airlines Inc.

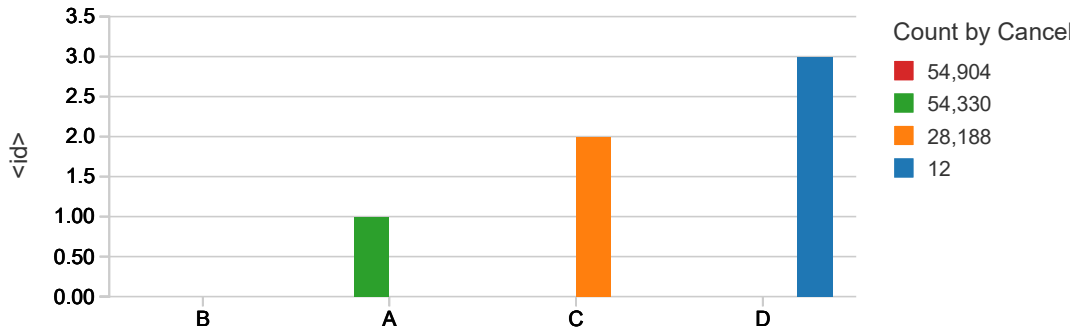
B	McCarran International	Southwest Airlines Co.
B	William B Hartsfield-Atlanta Intl	AirTran Airways Corporation
A	LaGuardia	US Airways Inc. (Merged with America West 9/05. Re
B	Dallas Love	Southwest Airlines Co.
A	McCarran International	US Airways Inc. (Merged with America West 9/05. Re
C	William B Hartsfield-Atlanta Intl	Delta Air Lines Inc.
C	Gen Edw L Logan Intl	US Airways Inc. (Merged with America West 9/05. Re
A	Phoenix Sky Harbor International	Mesa Airlines Inc.
A	San Jose International	Southwest Airlines Co.

Showing the first 1000 rows.

```
display(airline_df_2.filter("Cancelled=='1'").groupBy('CancellationCode').count().\
              withColumn('count', F.format_number('count',0)).\
              withColumnRenamed('count', 'Count by Cancellation code').\
              sort('count', ascending=False))
```

CancellationCode	Count by Cancellation cod
B	54,904
A	54,330
C	28,188
D	12

```
display(airline_df_2.filter("Cancelled=='1'").groupBy('CancellationCode').count().\
              withColumn('count', F.format_number('count',0)).\
              withColumnRenamed('count', 'Count by Cancellation code').\
              sort('count', ascending=False))
```



```
display(airline_df_2.filter("Cancelled=='1'").groupby('Month','Origin_airport','Description').count
().sort('count', ascending=False).\
       withColumn('count', F.format_number('count', 0)).\
       withColumnRenamed('count', 'Count of Cancellation by Month'))
```

Month	Origin_airport	Description
4	Dallas-Fort Worth International	American Airlines Inc.
3	Dallas-Fort Worth International	American Airlines Inc.
2	Chicago O'Hare International	American Eagle Airlines Inc
9	William P Hobby	Southwest Airlines Co.
1	Chicago O'Hare International	American Eagle Airlines Inc
12	Chicago O'Hare International	American Eagle Airlines Inc
2	Chicago O'Hare International	American Airlines Inc.
9	George Bush Intercontinental	Continental Air Lines Inc.
9	George Bush Intercontinental	Expressjet Airlines Inc.
2	Chicago O'Hare International	Skywest Airlines Inc.
3	Dallas-Fort Worth International	American Eagle Airlines Inc
4	Chicago O'Hare International	American Airlines Inc.
1	William B Hartsfield-Atlanta Intl	Atlantic Southeast Airlines
2	Chicago O'Hare International	United Air Lines Inc.
2	Newark Intl	Expressjet Airlines Inc.
1	Chicago O'Hare International	Skywest Airlines Inc.
2	Chicago Midway	Southwest Airlines Co.
2	Chicago O'Hare International	Mesa Airlines Inc.
3	Chicago O'Hare International	American Eagle Airlines Inc
1	William B Hartsfield-Atlanta Intl	Delta Air Lines Inc.
6	Chicago O'Hare International	American Eagle Airlines Inc
1	Chicago O'Hare International	American Airlines Inc.
2	Dallas-Fort Worth International	American Airlines Inc.
1	Chicago O'Hare International	United Air Lines Inc.
6	Chicago O'Hare International	United Air Lines Inc.
12	Newark Intl	Expressjet Airlines Inc.
3	Chicago O'Hare International	American Airlines Inc.
5	Dallas-Fort Worth International	American Airlines Inc.
6	Newark Intl	Expressjet Airlines Inc.
1	San Francisco International	Skywest Airlines Inc.
1	Dallas-Fort Worth International	American Airlines Inc.
12	Chicago O'Hare International	United Air Lines Inc.
12	William B Hartsfield-Atlanta Intl	Atlantic Southeast Airlines
7	Chicago O'Hare International	United Air Lines Inc.
3	Chicago O'Hare International	Skywest Airlines Inc.
8	Dallas-Fort Worth International	American Airlines Inc.
7	William B Hartsfield-Atlanta Intl	Delta Air Lines Inc.
3	George Bush Intercontinental	Expressjet Airlines Inc.
6	Dallas-Fort Worth International	American Airlines Inc.
8	Newark Intl	Expressjet Airlines Inc.
3	Newark Intl	Expressjet Airlines Inc.

7	William B Hartsfield-Atlanta Intl	Atlantic Southeast Airlines
12	Chicago O'Hare International	American Airlines Inc.
2	Washington Dulles International	Mesa Airlines Inc.
3	Chicago O'Hare International	United Air Lines Inc.
12	Detroit Metropolitan-Wayne County	Pinnacle Airlines Inc.
1	Dallas-Fort Worth International	American Eagle Airlines Inc
3	William B Hartsfield-Atlanta Intl	Delta Air Lines Inc.
12	Chicago Midway	Southwest Airlines Co.
1	Chicago O'Hare International	Mesa Airlines Inc.
12	Dallas-Fort Worth International	American Eagle Airlines Inc
12	Chicago O'Hare International	Skywest Airlines Inc.
12	McCarran International	Southwest Airlines Co.
2	William B Hartsfield-Atlanta Intl	Atlantic Southeast Airlines
2	Detroit Metropolitan-Wayne County	Pinnacle Airlines Inc.
8	George Bush Intercontinental	Expressjet Airlines Inc.
6	Chicago O'Hare International	Skywest Airlines Inc.
5	Dallas-Fort Worth International	American Eagle Airlines Inc
12	Chicago O'Hare International	Mesa Airlines Inc.
8	Chicago O'Hare International	United Air Lines Inc.
12	George Bush Intercontinental	Expressjet Airlines Inc.
3	Detroit Metropolitan-Wayne County	Pinnacle Airlines Inc.
1	Detroit Metropolitan-Wayne County	Pinnacle Airlines Inc.
7	Chicago O'Hare International	American Eagle Airlines Inc
3	Cleveland-Hopkins Intl	Expressjet Airlines Inc.
9	Dallas Love	Southwest Airlines Co.
12	Seattle-Tacoma Intl	Alaska Airlines Inc.
3	William B Hartsfield-Atlanta Intl	Atlantic Southeast Airlines
2	Charlotte/Douglas International	Mesa Airlines Inc.
8	John F Kennedy Intl	Comair Inc.
8	Chicago O'Hare International	American Eagle Airlines Inc
8	John F Kennedy Intl	JetBlue Airways
9	Chicago O'Hare International	American Eagle Airlines Inc
4	Chicago O'Hare International	American Eagle Airlines Inc
2	LaGuardia	American Airlines Inc.
7	John F Kennedy Intl	Comair Inc.
5	Chicago O'Hare International	United Air Lines Inc.
8	William B Hartsfield-Atlanta Intl	Atlantic Southeast Airlines
9	New Orleans International	Southwest Airlines Co.
2	Cincinnati Northern Kentucky Intl	Comair Inc.
7	Dallas-Fort Worth International	American Airlines Inc.
6	Dallas-Fort Worth International	American Eagle Airlines Inc
1	San Francisco International	United Air Lines Inc.

7	John F Kennedy Intl	JetBlue Airways
7	Denver Intl	United Air Lines Inc.
7	Newark Intl	Expressjet Airlines Inc.
2	Dallas-Fort Worth International	American Eagle Airlines Inc
4	Chicago O'Hare International	United Air Lines Inc.
2	Newark Intl	Continental Air Lines Inc.
5	Chicago O'Hare International	American Eagle Airlines Inc
2	Memphis International	Pinnacle Airlines Inc.
9	Dallas-Fort Worth International	American Airlines Inc

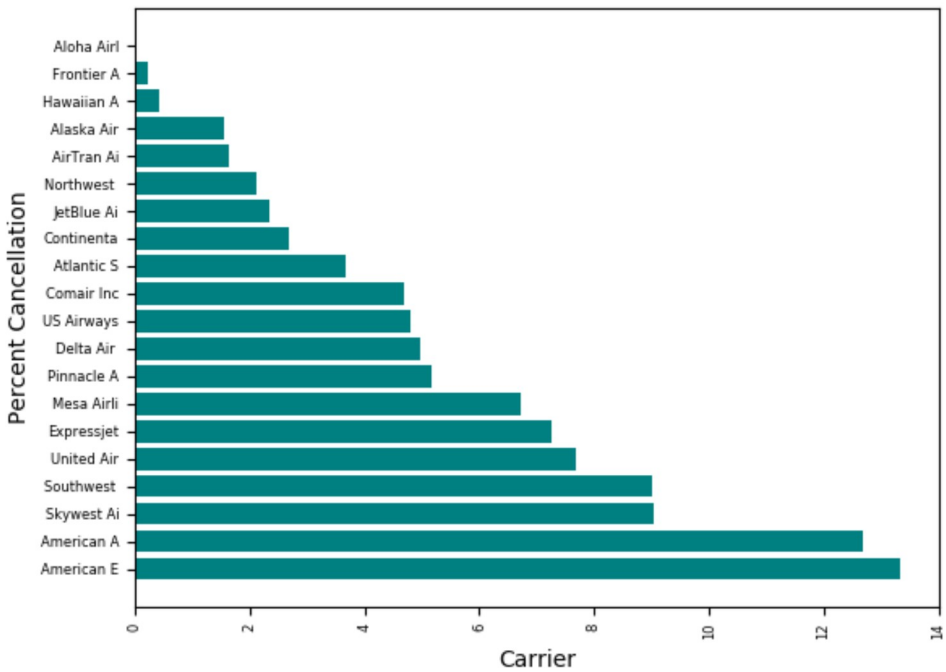
Showing the first 1000 rows.

```
tmp=airline_df_2.filter("Cancelled=='1'").groupby('UniqueCarrier','Description').count()

total = tmp.select("count").agg({"count": "sum"}).collect().pop()['sum(count)']
result = tmp.withColumn('percent', (tmp['count']/total) * 100)
result=result.withColumn('percent', F.format_number('percent',2)).\
        select('Description', 'percent')

result = result.select('Description',
result.percent.cast('float').alias('percent')).sort('percent', ascending=False)
display(result)
```

Description
American Eagle Airlines Inc.
American Airlines Inc.
Skywest Airlines Inc.
Southwest Airlines Co.
United Air Lines Inc.
Expressjet Airlines Inc.
Mesa Airlines Inc.
Pinnacle Airlines Inc.
Delta Air Lines Inc.
US Airways Inc. (Merged with America West 9/05. Reporting for both starting 10/07.)
Comair Inc.
Atlantic Southeast Airlines
Continental Air Lines Inc.
JetBlue Airways
Northwest Airlines Inc.
AirTran Airways Corporation
Alaska Airlines Inc.
Hawaiian Airlines Inc.

Frontier Airlines Inc.
Aloha Airlines Inc.

```
import matplotlib.pyplot as plt


airlines=result.toPandas()['Description'].str.slice(0,10).values.tolist()
percent = result.toPandas()['percent'].values.tolist()


plt.barh(airlines, percent, color='teal')
plt.xlabel('Carrier')
plt.ylabel('Percent Cancellation')
plt.xticks(rotation=90)
plt.tick_params(labelsize=6)
plt.show()
display()
```



Interpretation of Descriptive statistics

#1. Maximum cancellations are by CancellationCode 'B'
#2. Dallas-Fort Worth International airport has maximum cancellation with code 'A', followed by Chicago O'Hare International with code 'B' and 'C'
#3. American Airlines Inc. originating from Dallas-Fort Worth International airport has maximum cancellations followed by American Eagle Airlines Inc. orignating from Chicago O'Hare International airport
#4. American Eagle Airlines Inc. has maximum cancellations across all airports, followed closely by American Airlines Inc.
#5. Chicago O'Hare International airport has maximum cancellations followed by Dallas-Fort Worth International having only half of the leader
#6. April Month (4), has the maximum cancellations happening at Dallas-Fort Worth International airport by American Airlines Inc. closely followed by March Month (3) also by same carrier at same origin airport
#7. Maximum percentage of cancellation is by the airlines American Eagle Airlines Inc. and American Airlines Inc.

Intuitions: By now, it is somewhat clear, American Airlines In

#1. Human Error: If someone predicts that American Eagle Airlines Inc. will get cancelled among other airlines, they will be atleast 13% correct if a flight did get cancelled and 1% correct if any flight will get cancelled or not
#2. Validation Error: If our model can perform better than that in predicting a cancellation of a flight, we have a better use of our model

#Null Hypothesis:
#1. Cancellations happened due to factors attributed to an airport
#2. Cancellations happened due to factors attributed to an airline
#3. Cancellations happened due to factors attributed to

#Multi-collinearity

Chi-Square Test to verify if Cancellation is related to selecte

#In order to answer questions like, if Cancellation is related to UniqueCarrier or Origin or Destination or DayofWeek or Month or DayofMonth, we run Chi-Square test

#NULL-HYPOTHESIS: The null hypothesis in Chi-Square test is that, "the categorical values are independent"

#We can see that Cancellation varies by Airport, Airline and Month. But, we cannot conclude this for entire population, as it might have happened by chance for this sample set

#However, if strong relation is found in large sample, then we can refute our NULL hypothesis. The strength of dependence is shown by the probability or p-value

#If one variable is independent of another variable, then relative frequencies of one variable are identical over all levels of another variable

#If Chi-Square p-value is less than 5%, we can reject the NULL hypothesis and say that the variables are dependent on each other

```
from pyspark.ml.stat import ChiSquareTest
```

```
from pyspark.ml.classification import LogisticRegression
from pyspark.ml.evaluation import RegressionEvaluator
from pyspark.ml.feature import VectorAssembler
from pyspark.ml.feature import StringIndexer
from pyspark.ml.feature import StandardScaler
from pyspark.ml import Pipeline
from pyspark.ml.tuning import CrossValidator
from pyspark.ml.tuning import TrainValidationSplit
```

```
UniqueCarrierIndex = StringIndexer(inputCol='UniqueCarrier', outputCol='UniqueCarrierIndex',
handleInvalid='keep')
OriginIndex = StringIndexer(inputCol='Origin', outputCol='OriginIndex', handleInvalid='keep')
DestIndex = StringIndexer(inputCol='Dest', outputCol='DestIndex', handleInvalid='keep')
CancelledIndex = StringIndexer(inputCol='Cancelled', outputCol='CancelledIndex',
handleInvalid='keep')
CancellationCodeIndex = StringIndexer(inputCol='CancellationCode',
outputCol='CancellationCodeIndex', handleInvalid='keep')
DayofWeekIndex = StringIndexer(inputCol='DayofWeek', outputCol='DayofWeekIndex',
handleInvalid='keep')
DepTimeIndex = StringIndexer(inputCol='DepTime', outputCol='DepTimeIndex', handleInvalid='keep')
MonthIndex = StringIndexer(inputCol='Month', outputCol='MonthIndex', handleInvalid='keep')
DayofMonthIndex = StringIndexer(inputCol='DayofMonth', outputCol='DayofMonthIndex',
handleInvalid='keep')
```

```
airline_df_train, airline_df_test = airline_df.select('UniqueCarrier', 'Origin', 'Dest',
'Cancelled', 'CancellationCode','DayofWeek', 'DepTime','Month',
'DayofMonth').randomSplit([0.7,0.3], seed=100)
```

```
assembler = VectorAssembler(inputCols=['UniqueCarrierIndex', 'OriginIndex', 'DestIndex',
'CancellationCodeIndex', 'DayofWeekIndex', 'DepTimeIndex','MonthIndex', 'DayofMonthIndex'],
outputCol='features')
```

```
pipe=Pipeline(stages=[UniqueCarrierIndex,
OriginIndex,DestIndex,CancelledIndex,CancellationCodeIndex,
DayofWeekIndex,DepTimeIndex,MonthIndex,DayofMonthIndex,assembler])
```

```
p=pipe.fit(airline_df_train)
```

```
p=p.transform(airline_df_train)
```

```
r = ChiSquareTest.test(p, "features", "CancelledIndex").head()
```

```
print("pValues: " + str(r.pValues))
print("degreesOfFreedom: " + str(r.degreesOfFreedom))
print("statistics: " + str(r.statistics))
```

```
pValues: [0.0,0.0,0.0,0.0,0.0,0.0,0.0,0.0,0.0]
```

```
degreesOfFreedom: [19, 302, 303, 4, 6, 1440, 11, 30]
statistics: [22182.74291141317,30551.853383112266,29447.651788315332,4906543.000000001,1565.878718
8880342,4863239.444302091,20553.587598560807,4739.520489376625]
```

Interpretation of p-values for Chi-Square test

```
#Based on the p-values received (<5%), we can reject the NULL hypothesis and say that, Cancellation
has dependency on all the variables selected here namely
#'UniqueCarrierIndex', 'OriginIndex', 'DestIndex', 'CancellationCodeIndex', 'DayofWeekIndex',
'DepTimeIndex','MonthIndex', 'DayofMonthIndex'
```

Multi-Collinearity Test

```
#Next we need to check if independent variables have collinearity. If they do, then it is best to
remove the features with high collinearity
```

```
from pyspark.ml.linalg import Vectors
from pyspark.ml.stat import Correlation
```

```
airline_df_train_corr1 = Correlation.corr(p, "features")
```

```
print("Pearson correlation matrix:\n" + str(airline_df_train_corr1[0]))
```

```
Pearson correlation matrix:
Column<b'pearson(features)'>
```

```
display(airline_df_train_corr1)
```

pearson(features)
1.0 0.06575570036830161 ... (8 total) 0.06575570036830161 1.0 ... 0.0657252726295638 -0.24011421864393012 ... -0.0026678381935330564 -0.014908711207421689 ... 0.0068635648746472626 7.861196209730063E-4 ... -0.040400298332104526 -0.06761987574141898 ... -0.009227404522645217 -0.005782797044295207 ... 0.0014049443489825 -4.897018765511002E-4 ...

Interpretation of Multi-collinearity

```
#Based on the above multi-collinearity tests, it is evident that columns 4 and 8 have high
collinearity and they can be excluded
#They are CancellationCodeIndex and DayofMonthIndex
```

Restarting Spark Context to Run SVClassifier

```
spark.stop()
```

```
  The spark context has stopped and the driver is restarting. Your notebook will be automatically
  reattached.
```

```
from pyspark.sql import SparkSession
```

```
spark=SparkSession.builder.appName('Project1').getOrCreate()
```

```
airline_df = spark.read.csv('/FileStore/tables/2008_csv-db05f.bz2', inferSchema=True, header=True)
```

SVClassifier to predict Cancellation

```
from pyspark.ml.classification import LinearSVC
from pyspark.ml.feature import VectorAssembler,StringIndexer,StandardScaler
from pyspark.ml import Pipeline
```

```
airline_df_sample = airline_df.sample(False, 0.5,10)
print(airline_df_sample.count())
```

```
3505205
```

```
airline_df_select = airline_df_sample.select('Month', 'UniqueCarrier', 'Origin', 'Cancelled')
```

Setup the structure to pass to the Pipeline

```
#Convert the Month, Carrier and Origin to Categorical values using the StringIndexer. This is
necessary, as we are dealing with Binary classification of 0 or 1 for Cancellation label variable
```

```
month_indexer = StringIndexer(inputCol='Month',outputCol='month_index',handleInvalid='keep')
carrier_indexer =
StringIndexer(inputCol='UniqueCarrier',outputCol='carrier_index',handleInvalid='keep')
origin_indexer = StringIndexer(inputCol='Origin',outputCol='origin_index',handleInvalid='keep')
```

```
#PySpark can handle only RDD objects. Hence the input features need to be converted into a Vector
and referred with the vector name as "features"
```

```
assembler = VectorAssembler(inputCols=['month_index','carrier_index','origin_index'],
                            outputCol="unscaled_features")
```

```
scaler = StandardScaler(inputCol="unscaled_features",outputCol="features")
```

```
svc_model = LinearSVC(labelCol='Cancelled')
```

```
#Pipeline is defined here using the StringIndexers, Assember, scaler and the SVC model defined
above
```

```
pipe = Pipeline(stages=[month_indexer,carrier_indexer,origin_indexer,assembler,scaler,svc_model])
```

Split the data into Train and Test and fit the model on train d

```
train_data,test_data=airline_df_select.randomSplit([0.7,0.3])
```

This model fit ran for 2.43hrs

```
fit_model=pipe.fit(train_data)
```

```
results = fit_model.transform(test_data)
```

```
results.select(['Cancelled','prediction']).show()
```

```
+---------+----------+
|Cancelled|prediction|
+---------+----------+
|        0|       0.0|
|        0|       0.0|
|        0|       0.0|
|        0|       0.0|
|        0|       0.0|
|        0|       0.0|
|        0|       0.0|
|        0|       0.0|
|        0|       0.0|
|        1|       0.0|
|        0|       0.0|
|        0|       0.0|
|        0|       0.0|
|        0|       0.0|
|        0|       0.0|
|        1|       0.0|
|        1|       0.0|
|        0|       0.0|
|        0|       0.0|
```

```
|        0|       0.0|
+---------+----------+
only showing top 20 rows
```

Evaluation of the SVClassifier model

```
#Evaluation of the Model. Classifier models are evaluated using the following:
#Accuracy
#Recall
#F1Score
#Area Under the Curve (AUC)

#Accuracy: Model may have high accuracy by predicting all the Negative values correctly like 90%, but if 10% of values are positive and if most of them are predicted incorrectly, the model is not of much use. Hence other metrics are used
#Recall: This specifically focuses on positive values prediction, out of the 10% positives, how many are predicted correctly. In some cases, overall accuracy may be low, but, if recall is high, the model still may be heuristically/practically useful to deploy in a business scenario
#F1Score depends on True Positives and False Positives: Out of the Positives and Negatives predicted, how many are True and False within each of them. We prefer obviously maximum of True Positives and True Negatives and very minimum of False Positives and False Negatives. F1 score is calculated by taking a sum of True in numerator and total in denominator
#Area Under the Curve (AUC) is useful to know how True Positives and False Positives are distributed in a graph of X and Y axes (called ROC curve). The more the area under the curve, the best model it is. Maximum possible value of the AUC is whe the curve falling on the edges of the square opposite to them.


from pyspark.ml.evaluation import BinaryClassificationEvaluator


#Area Under the Curve


AUC = AUC_evaluator.evaluate(results)


AUC_evaluator =
BinaryClassificationEvaluator(rawPredictionCol='prediction',labelCol='Cancelled',metricName='areaUnderROC')


print("The area under the curve is {}".format(AUC))

The area under the curve is 0.5

#Interpretation: The ROC curve is plotted with TPR against the FPR where TPR is on y-axis and FPR is on the x-axis. When AUC is 0.5, it means there is 50% chance that model will be able to distinguish between positive class and negative class. This is the worst situation. When AUC is approximately 0.5, model has no discrimination capacity to distinguish between positive class and negative class. REMEDY: More features from the dataset need to be considered to improve the model performance in terms of AUC.


#2. Area under the PR
```

```
PR_evaluator =
BinaryClassificationEvaluator(rawPredictionCol='prediction',labelCol='Cancelled',metricName='areaUn
derPR')


PR = PR_evaluator.evaluate(results)
```

```
print("The area under the PR curve is {}".format(PR))
```

```
The area under the PR curve is 0.01969750732259068
```

```
#INTERPRETATION: The theoretical range of AU PR Curve score is between 0 and 1. A value less than
0.5 is not considered a good score.


#3. Accuracy


from pyspark.ml.evaluation import MulticlassClassificationEvaluator


ACC_evaluator = MulticlassClassificationEvaluator(
    labelCol="Cancelled", predictionCol="prediction", metricName="accuracy")
```

```
accuracy = ACC_evaluator.evaluate(results)


print("The accuracy of the model is {}".format(accuracy))
```

```
The accuracy of the model is 0.9803024926774093
```

```
#INTERPRETATION: The model has a phenomenal 98% accuracy. However, as explained before, once again
explaining with another example here. If there are 91 cases of non-terrorists in a data set and
model identifies 90 of them right, it is a phenomenal 90%+ accuracy. However, if there are 9
terrorists in the dataset and if the model only predicts 1 of them correctly, though the accuracy
is very high, the model is not reliable. This is the same case happening here. All the non-
cancelled flights are accurately identified, hence accuracy is high. Out of the cancelled flights,
less than 50% is predicted correctly. Hence this model needs improvement, by selecting more
features (it will make the model fit to run even longer time from the current time of 2.43 hours
with only 4 columns)


#4. Confusion Matrix


from sklearn.metrics import confusion_matrix
```

```
y_true = results.select("Cancelled")
y_true = y_true.toPandas()

y_pred = results.select("prediction")
y_pred = y_pred.toPandas()

cnf_matrix = confusion_matrix(y_true, y_pred)
print("Below is the confusion matrix: \n {}".format(cnf_matrix))
```

```
/databricks/python/lib/python3.7/site-packages/pyarrow/__init__.py:152: UserWarning: pyarrow.open_
stream is deprecated, please use pyarrow.ipc.open_stream
  warnings.warn("pyarrow.open_stream is deprecated, please use "
Below is the confusion matrix:
 [[1030493       0]
 [  20706       0]]
```

```
#INTERPRETATION: As can be seen here, True Positives are very highly correctly predicted, True
Negatives are not highly predicted. False positives are 0, but, there is a high number of false
negatives, making this model not yet usable until this is addressed.


#https://spark.apache.org/docs/2.1.1/ml-classification-regression.html#binomial-logistic-regression
#https://docs.pymc.io/notebooks/GLM-negative-binomial-regression.html
#https://data.library.virginia.edu/getting-started-with-negative-binomial-regression-modeling/
#https://docs.pymc.io/history.html
#https://towardsdatascience.com/negative-binomial-regression-f99031bb25b4
#https://stackoverflow.com/questions/46710934/pyspark-sql-utils-illegalargumentexception-ufield-
features-does-not-exist/46729342
#https://www.spss-tutorials.com/chi-square-independence-test/
#https://towardsdatascience.com/understanding-auc-roc-curve-68b2303cc9c5


#blank


#blank
```

databricks Airlines-Data-Analysis

Overview

This notebook will show you how to create and query a table or DataFrame that you uploaded to DBFS. DBFS (https://docs.databricks.com/user-guide/dbfs-databricks-file-system.html) is a Databricks File System that allows you to store data for querying inside of Databricks. This notebook assumes that you have a file already inside of DBFS that you would like to read from.

This notebook is written in **Python** so the default cell type is Python. However, you can use different languages by using the `%LANGUAGE` syntax. Python, Scala, SQL, and R are all supported.

```
# File location and type
file_location = "/FileStore/tables/2008.csv"
file_type = "csv"

# CSV options
infer_schema = "True"
first_row_is_header = "True"
delimiter = ","

# The applied options are for CSV files. For other file types, these will be ignored.
df = spark.read.format(file_type) \
  .option("inferSchema", infer_schema) \
  .option("header", first_row_is_header) \
  .option("sep", delimiter) \
  .load(file_location)

display(df)
```

Year	Month	DayofMonth	DayOfWeek	DepTime	CRSDepTime	ArrTime	CRSArrTime	UniqueC
2008	1	3	4	2003	1955	2211	2225	WN
2008	1	3	4	754	735	1002	1000	WN
2008	1	3	4	628	620	804	750	WN
2008	1	3	4	926	930	1054	1100	WN
2008	1	3	4	1829	1755	1959	1925	WN
2008	1	3	4	1940	1915	2121	2110	WN
2008	1	3	4	1937	1830	2037	1940	WN
2008	1	3	4	1039	1040	1132	1150	WN
2008	1	3	4	617	615	652	650	WN
2008	1	3	4	1620	1620	1639	1655	WN
2008	1	3	4	706	700	916	915	WN
2008	1	3	4	1644	1510	1845	1725	WN
2008	1	3	4	1426	1430	1426	1425	WN
2008	1	3	4	715	715	720	710	WN
2008	1	3	4	1702	1700	1651	1655	WN
2008	1	3	4	1029	1020	1021	1010	WN

2008	1	3	4	1452	1425	1640	1625	WN
2008	1	3	4	754	745	940	955	WN
2008	1	3	4	1323	1255	1526	1510	WN
2008	1	3	4	1416	1325	1512	1435	WN
2008	1	3	4	706	705	807	810	WN
2008	1	3	4	1657	1625	1754	1735	WN
2008	1	3	4	1900	1840	1956	1950	WN
2008	1	3	4	1039	1030	1133	1140	WN
2008	1	3	4	801	800	902	910	WN
2008	1	3	4	1520	1455	1619	1605	WN
2008	1	3	4	1422	1255	1657	1610	WN
2008	1	3	4	1954	1925	2239	2235	WN
2008	1	3	4	636	635	921	945	WN
2008	1	3	4	734	730	958	1020	WN
2008	1	3	4	2107	1945	2334	2230	WN
2008	1	3	4	1008	1005	1234	1255	WN
2008	1	3	4	712	710	953	1000	WN
2008	1	3	4	1312	1300	1546	1550	WN
2008	1	3	4	1449	1430	1715	1720	WN
2008	1	3	4	1634	1555	1859	1845	WN
2008	1	3	4	831	830	935	955	WN
2008	1	3	4	1812	1650	1927	1815	WN
2008	1	3	4	1127	1105	1235	1230	WN
2008	1	3	4	1424	1355	1531	1520	WN
2008	1	3	4	1326	1230	1559	1530	WN
2008	1	3	4	1749	1725	2019	2030	WN
2008	1	3	4	726	720	958	1020	WN
2008	1	3	4	646	640	929	955	WN
2008	1	3	4	1153	1140	1428	1440	WN
2008	1	3	4	1528	1510	1802	1810	WN
2008	1	3	4	634	635	907	935	WN
2008	1	3	4	831	830	1148	1140	WN
2008	1	3	4	1450	1435	1806	1745	WN
2008	1	3	4	2245	1730	2354	1850	WN
2008	1	3	4	615	615	724	735	WN
2008	1	3	4	1150	1145	1303	1305	WN
2008	1	3	4	2025	1940	2135	2100	WN
2008	1	3	4	1038	945	1314	1225	WN
2008	1	3	4	1900	1850	2123	2045	WN
2008	1	3	4	700	700	851	900	WN
2008	1	3	4	948	925	959	940	WN
2008	1	3	4	646	620	725	655	WN

2008	1	3	4	1110	1040	1136	1110	WN
2008	1	3	4	1535	1535	1603	1610	WN
2008	1	3	4	1919	1915	1942	1950	WN
2008	1	3	4	1053	1055	1245	1240	WN
2008	1	3	4	1433	1440	1623	1625	WN
2008	1	3	4	2015	2010	2158	2155	WN
2008	1	3	4	2139	2130	2244	2240	WN
2008	1	3	4	1500	1500	1602	1615	WN
2008	1	3	4	850	850	1000	1000	WN

Showing the first 1000 rows.

```
# File location and type
file_location = "/FileStore/tables/carriers__1_-6545c.csv"
file_type = "csv"

# CSV options
infer_schema = "True"
first_row_is_header = "True"
delimiter = ","

# The applied options are for CSV files. For other file types, these will be ignored.
carriers = spark.read.format(file_type) \
  .option("inferSchema", infer_schema) \
  .option("header", first_row_is_header) \
  .option("sep", delimiter) \
  .load(file_location)
carriers=carriers.withColumnRenamed("Code", "UniqueCarrier")
df = df.join(carriers, 'UniqueCarrier')
#display(carriers)
display(df)
```

UniqueCarrier	Year	Month	DayofMonth	DayOfWeek	DepTime	CRSDepTime	ArrTime	CRSArr
WN	2008	1	3	4	2003	1955	2211	2225
WN	2008	1	3	4	754	735	1002	1000
WN	2008	1	3	4	628	620	804	750
WN	2008	1	3	4	926	930	1054	1100
WN	2008	1	3	4	1829	1755	1959	1925
WN	2008	1	3	4	1940	1915	2121	2110
WN	2008	1	3	4	1937	1830	2037	1940
WN	2008	1	3	4	1039	1040	1132	1150

WN	2008	1	3	4	617	615	652	650
WN	2008	1	3	4	1620	1620	1639	1655
WN	2008	1	3	4	706	700	916	915
WN	2008	1	3	4	1644	1510	1845	1725
WN	2008	1	3	4	1426	1430	1426	1425
WN	2008	1	3	4	715	715	720	710
WN	2008	1	3	4	1702	1700	1651	1655
WN	2008	1	3	4	1029	1020	1021	1010
WN	2008	1	3	4	1452	1425	1640	1625
WN	2008	1	3	4	754	745	940	955
WN	2008	1	3	4	1323	1255	1526	1510
WN	2008	1	3	4	1416	1325	1512	1435
WN	2008	1	3	4	706	705	807	810
WN	2008	1	3	4	1657	1625	1754	1735
WN	2008	1	3	4	1900	1840	1956	1950
WN	2008	1	3	4	1039	1030	1133	1140
WN	2008	1	3	4	801	800	902	910
WN	2008	1	3	4	1520	1455	1619	1605
WN	2008	1	3	4	1422	1255	1657	1610
WN	2008	1	3	4	1954	1925	2239	2235
WN	2008	1	3	4	636	635	921	945
WN	2008	1	3	4	734	730	958	1020
WN	2008	1	3	4	2107	1945	2334	2230
WN	2008	1	3	4	1008	1005	1234	1255

WN	2008	1	3	4	712	710	953	1000
WN	2008	1	3	4	1312	1300	1546	1550
WN	2008	1	3	4	1449	1430	1715	1720
WN	2008	1	3	4	1634	1555	1859	1845
WN	2008	1	3	4	831	830	935	955
WN	2008	1	3	4	1812	1650	1927	1815
WN	2008	1	3	4	1127	1105	1235	1230
WN	2008	1	3	4	1424	1355	1531	1520
WN	2008	1	3	4	1326	1230	1559	1530
WN	2008	1	3	4	1749	1725	2019	2030
WN	2008	1	3	4	726	720	958	1020
WN	2008	1	3	4	646	640	929	955
WN	2008	1	3	4	1153	1140	1428	1440
WN	2008	1	3	4	1528	1510	1802	1810
WN	2008	1	3	4	634	635	907	935
WN	2008	1	3	4	831	830	1148	1140
WN	2008	1	3	4	1450	1435	1806	1745
WN	2008	1	3	4	2245	1730	2354	1850
WN	2008	1	3	4	615	615	724	735
WN	2008	1	3	4	1150	1145	1303	1305
WN	2008	1	3	4	2025	1940	2135	2100
WN	2008	1	3	4	1038	945	1314	1225
WN	2008	1	3	4	1900	1850	2123	2045
WN	2008	1	3	4	700	700	851	900

WN	2008	1	3	4	948	925	959	940
WN	2008	1	3	4	646	620	725	655
WN	2008	1	3	4	1110	1040	1136	1110
WN	2008	1	3	4	1535	1535	1603	1610
WN	2008	1	3	4	1919	1915	1942	1950
WN	2008	1	3	4	1053	1055	1245	1240
WN	2008	1	3	4	1433	1440	1623	1625
WN	2008	1	3	4	2015	2010	2158	2155
WN	2008	1	3	4	2139	2130	2244	2240
WN	2008	1	3	4	1500	1500	1602	1615
WN	2008	1	3	4	850	850	1000	1000
WN	2008	1	3	4	646	645	752	755
WN	2008	1	3	4	1221	1220	1328	1330
WN	2008	1	3	4	1738	1730	1841	1840
WN	2008	1	3	4	1813	1735	1936	1905
WN	2008	1	3	4	802	750	1001	955
WN	2008	1	3	4	1820	1825	1946	1955
WN	2008	1	3	4	821	820	953	945
WN	2008	1	3	4	1734	1650	1941	1905
WN	2008	1	3	4	712	700	926	915
WN	2008	1	3	4	1318	1310	1410	1400
WN	2008	1	3	4	958	900	1052	950
WN	2008	1	3	4	1859	1850	1950	1945
WN	2008	1	3	4	1538	1445	1753	1710

WN	2008	1	3	4	933	935	1151	1200
WN	2008	1	3	4	2248	2125	102	2345
WN	2008	1	3	4	1327	1230	1550	1500
WN	2008	1	3	4	624	625	846	850
WN	2008	1	3	4	1614	1600	1833	1825

Showing the first 1000 rows.

Exploratory Data Analysis

```
df.columns
```

```
Out[4]: ['UniqueCarrier',
 'Year',
 'Month',
 'DayofMonth',
 'DayOfWeek',
 'DepTime',
 'CRSDepTime',
 'ArrTime',
 'CRSArrTime',
 'FlightNum',
 'TailNum',
 'ActualElapsedTime',
 'CRSElapsedTime',
 'AirTime',
 'ArrDelay',
 'DepDelay',
 'Origin',
 'Dest',
 'Distance',
 'TaxiIn',
 'TaxiOut',
 'Cancelled',
 'CancellationCode',
 'Diverted',
 'CarrierDelay',
 'WeatherDelay',
 'NASDelay',
 'SecurityDelay',
 'LateAircraftDelay',
 'Description']
```

```
df.select('ArrTime','DepTime','ArrDelay','DepDelay','Distance').describe().show()
```

```
+-------+------------------+------------------+------------------+------------------+------------------+
|summary|           ArrTime|           DepTime|          ArrDelay|          DepDelay|          Distance|
```

```
+-------+------------------+------------------+------------------+------------------+-------------
----+
|  count|           7009728|           7009728|           7009728|           7009728|          700
9728|
|   mean| 1481.258226684178|1333.8300461105448|  8.16845238729114| 9.972570088930182|726.387029425
3928|
| stddev|505.22512933801625|478.06889486630547|38.501936948828586|35.311270777552814|562.101803484
0342|
|    min|                 1|                 1|                -1|                -1|
11|
|    max|                NA|                NA|                NA|                NA|
4962|
+-------+------------------+------------------+------------------+------------------+-------------
----+
```

1. Total cancelled flights

```
#Total cancelled flights
Cancelled=df.filter("Cancelled == 1").count()
print(Cancelled)
```

```
137434
```

2. % of Flights Cancelled

```
#Total Flights
Total=df.count()
#Percentage of flights cancelled
percentage= Cancelled/Total*100
print("Percentage of Flights cancelled in a year ", percentage)
```

```
Percentage of Flights cancelled in a year  1.9606181580797428
```

3. Flights Cancelled by week Analysis

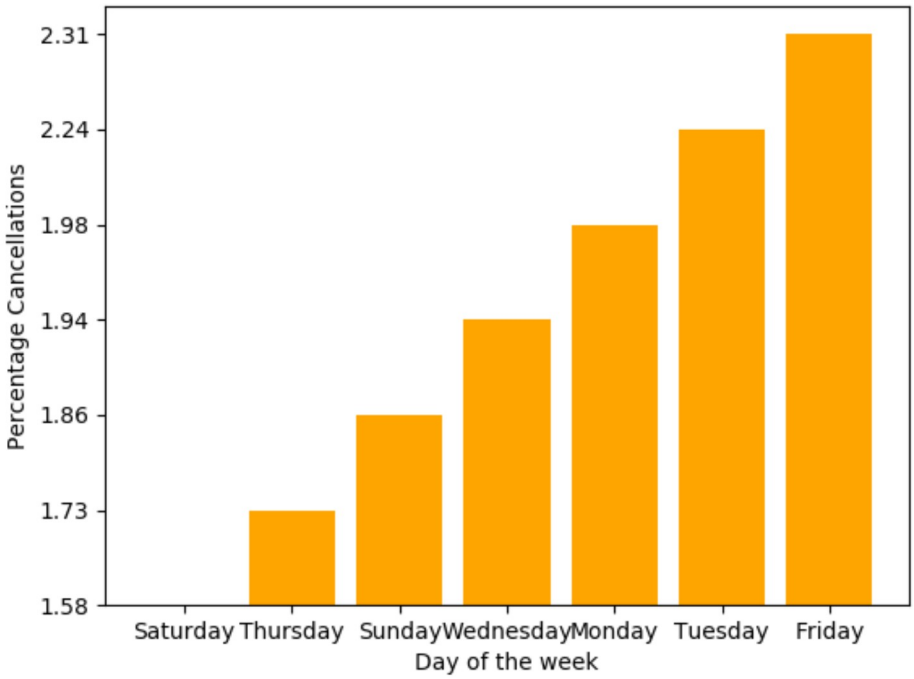
```
#Which day of the week has more cancellations?
from pyspark.sql.functions import *
import pyspark.sql.functions as f
from pyspark.sql.functions import format_number
df_week_cancelled=df.select('DayOfWeek','Cancelled').groupBy('DayOfWeek').agg(f.count("Cancelled"),
f.sum("Cancelled"))\
.withColumn("DayOfWeek",regexp_replace('DayOfWeek', '1', 'Monday'))\
.withColumn("DayOfWeek",regexp_replace('DayOfWeek', '2', 'Tuesday'))\
.withColumn("DayOfWeek",regexp_replace('DayOfWeek', '3', 'Wednesday'))\
.withColumn("DayOfWeek",regexp_replace('DayOfWeek', '4', 'Thursday'))\
.withColumn("DayOfWeek",regexp_replace('DayOfWeek', '5', 'Friday'))\
.withColumn("DayOfWeek",regexp_replace('DayOfWeek', '6', 'Saturday'))\
.withColumn("DayOfWeek",regexp_replace('DayOfWeek', '7', 'Sunday'))\
.withColumnRenamed('count(Cancelled)','scheduled')\
.withColumnRenamed('sum(Cancelled)','Times_cancelled')\
.withColumn('Percentage_cancelled',format_number(col('Times_cancelled')/col('scheduled')*100,2))\
.orderBy('Percentage_cancelled')
display(df_week_cancelled['DayOfWeek','Percentage_cancelled'])
```

DayOfWeek	Percentage_cancelled
Saturday	1.58
Thursday	1.73
Sunday	1.86
Wednesday	1.94
Monday	1.98
Tuesday	2.24
Friday	2.31

```
day=df_week_cancelled.toPandas()['DayOfWeek'].values.tolist()
percentage=df_week_cancelled.toPandas()['Percentage_cancelled'].values.tolist()
print(day,percentage)
```

```
/databricks/python/lib/python3.7/site-packages/pyarrow/__init__.py:152: UserWarning: pyarrow.open_
stream is deprecated, please use pyarrow.ipc.open_stream
  warnings.warn("pyarrow.open_stream is deprecated, please use "
['Saturday', 'Thursday', 'Sunday', 'Wednesday', 'Monday', 'Tuesday', 'Friday'] ['1.58', '1.73', '
1.86', '1.94', '1.98', '2.24', '2.31']
```

```
import matplotlib.pyplot as plt
plt.bar(day,percentage,color='orange')
plt.xlabel('Day of the week')
plt.ylabel('Percentage Cancellations')
plt.show()
display()
```



4. Flight Cancelled by Flight Number Analysis

```
#Flight number with more cancellations?
df_flightnum_cancelled=df.select('FlightNum','Cancelled').groupBy('FlightNum').agg(f.count("Cancell
ed"),f.sum("Cancelled"))\
.withColumnRenamed('count(Cancelled)','scheduled')\
.withColumnRenamed('sum(Cancelled)','Times_cancelled')\
.withColumn('Percentage_cancelled',format_number(col('Times_cancelled')/col('scheduled')*100,2))\
.orderBy('Percentage_cancelled', ascending=False)
df_flightnum_cancelled.show(n=50)
```

```
+---------+---------+---------------+--------------------+
|FlightNum|scheduled|Times_cancelled|Percentage_cancelled|
+---------+---------+---------------+--------------------+
|     7233|      302|             30|                9.93|
|     7212|      514|             51|                9.92|
|     3074|      545|             54|                9.91|
|     6047|      253|             25|                9.88|
|     6310|      142|             14|                9.86|
|     4445|      772|             76|                9.84|
|     5374|      326|             32|                9.82|
|     4370|      408|             40|                9.80|
|     5109|      296|             29|                9.80|
|     4775|      481|             47|                9.77|
|     6052|      318|             31|                9.75|
|     5946|      903|             88|                9.75|
|     3188|      394|             38|                9.64|
|     2396|      581|             56|                9.64|
|     7151|      670|             64|                9.55|
|     7390|       42|              4|                9.52|
|     5413|      296|             28|                9.46|
```

```
|     7335|      339|             32|                9.44|
|     7279|      533|             50|                9.38|
|     7182|      492|             46|                9.35|
|     6241|      193|             18|                9.33|
|     5802|      951|             88|                9.25|
|     6274|      281|             26|                9.25|
|     7244|      249|             23|                9.24|
|     7330|      282|             26|                9.22|
|     6060|      403|             37|                9.18|
|     4130|      458|             42|                9.17|
|     5166|      306|             28|                9.15|
|     7164|      351|             32|                9.12|
|     7147|      637|             58|                9.11|
|     7395|       88|              8|                9.09|
|     6239|       22|              2|                9.09|
|     5357|      300|             27|                9.00|
|     7419|       89|              8|                8.99|
|     6001|      345|             31|                8.99|
|     7394|       78|              7|                8.97|
|     7260|      235|             21|                8.94|
|     5273|      292|             26|                8.90|
|     7262|      326|             29|                8.90|
|     7457|       45|              4|                8.89|
|     7219|      350|             31|                8.86|
|     7293|      260|             23|                8.85|
|     7342|      305|             27|                8.85|
|     4250|      667|             59|                8.85|
|     7403|      181|             16|                8.84|
|     1383|      181|             16|                8.84|
|     6172|       34|              3|                8.82|
|     6262|       34|              3|                8.82|
|     7306|      239|             21|                8.79|
|     6325|       57|              5|                8.77|
+---------+---------+---------------+--------------------+
only showing top 50 rows
```

```
'''import pyspark.sql.functions as f
from pyspark.sql.window import Window
df_flightnum_cancelled.withColumn('Percentage_Cancelled',f.col('Times_Cancelled')/f.sum('Times_Canc
elled').over(Window.partitionBy())).\
withColumn('Percentage_Cancelled',col('Percentage_Cancelled')*100).orderBy('Percentage_Cancelled',a
scending=False).show(n=50)
'''
```

```
Out[12]: "import pyspark.sql.functions as f\nfrom pyspark.sql.window import Window\ndf_flightnum_c
ancelled.withColumn('Percentage_Cancelled',f.col('Times_Cancelled')/f.sum('Times_Cancelled').over
(Window.partitionBy())).withColumn('Percentage_Cancelled',col('Percentage_Cancelled')*100).orderBy
('Percentage_Cancelled',ascending=False).show(n=50)\n"
```

5.Flight Cancelled by Carrier Analysis

```
#Flight number with more cancellations?
df_carrier_cancelled=df.select('UniqueCarrier','Cancelled','Description').groupBy('UniqueCarrier','
Description').agg(f.count("Cancelled"),f.sum("Cancelled"))\
.withColumnRenamed('count(Cancelled)','scheduled')\
.withColumnRenamed('sum(Cancelled)','Times_cancelled')\
.withColumn('Percentage_cancelled',format_number(col('Times_cancelled')/col('scheduled')*100,2))\
.orderBy('Percentage_cancelled', ascending=False)
df_carrier_cancelled.show(n=50)
```

```
+-------------+--------------------+---------+---------------+--------------------+
|UniqueCarrier|         Description|scheduled|Times_cancelled|Percentage_cancelled|
+-------------+--------------------+---------+---------------+--------------------+
|           MQ|American Eagle Ai...|   490693|          18331|                3.74|
|           YV|  Mesa Airlines Inc.|   254930|           9219|                3.62|
|           OH|         Comair Inc.|   197607|           6462|                3.27|
|           AA|American Airlines...|   604885|          17440|                2.88|
|           9E|Pinnacle Airlines...|   262208|           7100|                2.71|
|           XE|Expressjet Airlin...|   374510|           9992|                2.67|
|           UA|United Air Lines ...|   449515|          10541|                2.34|
|           OO|Skywest Airlines ...|   567159|          12436|                2.19|
|           EV|Atlantic Southeas...|   280575|           5026|                1.79|
|           B6|     JetBlue Airways|   196091|           3205|                1.63|
|           DL|Delta Air Lines Inc.|   451931|           6813|                1.51|
|           US|US Airways Inc. (...|   453589|           6582|                1.45|
|           AS|Alaska Airlines Inc.|   151102|           2139|                1.42|
|           CO|Continental Air L...|   298455|           3702|                1.24|
|           WN|Southwest Airline...|  1201754|          12389|                1.03|
|           HA|Hawaiian Airlines...|    61826|            570|                0.92|
|           FL|AirTran Airways C...|   261684|           2236|                0.85|
|           NW|Northwest Airline...|   347652|           2906|                0.84|
|           AQ| Aloha Airlines Inc.|     7800|             42|                0.54|
|           F9|Frontier Airlines...|    95762|            303|                0.32|
+-------------+--------------------+---------+---------------+--------------------+
```

```
#df.select('FlightNum','Cancelled','Description').show()
```

```
'''df_fcarrier_cancelled.withColumn('Percentage_Cancelled',f.col('Times_Cancelled')/f.sum('Times_Ca
ncelled').over(Window.partitionBy())).\
withColumn('Percentage_Cancelled',col('Percentage_Cancelled')*100).orderBy('Percentage_Cancelled',a
scending=False).show(n=50)'''
```

```
Out[15]: "df_fcarrier_cancelled.withColumn('Percentage_Cancelled',f.col('Times_Cancelled')/f.sum('
Times_Cancelled').over(Window.partitionBy())).withColumn('Percentage_Cancelled',col('Percentage_Ca
ncelled')*100).orderBy('Percentage_Cancelled',ascending=False).show(n=50)"
```

6. Analysis of Arrival Delay

```
Total=df.select('ArrDelay').count()
Delayed=df.select('ArrDelay').filter("ArrDelay> 0").count()
print(Delayed)
```

```
2979504
```

```
Ontime=df.select('ArrDelay').filter("ArrDelay<= 0").count()
print(Ontime)
```

```
3875525
```

% of Flights Delayed

```
print(Ontime/Total)
```

```
0.5528780859970601
```

7. Flights Delayed more than 10 mins

```
df_delay_Ontime = df.withColumn(
    'status',
    #f.when(f.col("ArrDelay").between(5, 10), '05-10')\
    f.when(f.col("ArrDelay")> 10, 'Delayed')\
      .otherwise('On time')).select('Month','status','ArrDelay')
display(df_delay_Ontime)
```

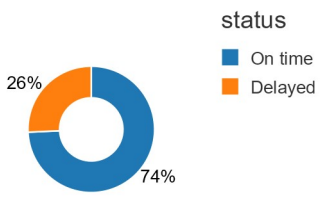
Month	status
1	On time
1	On time
1	Delayed
1	On time
1	Delayed
1	Delayed
1	Delayed
1	On time
1	On time
1	On time
1	On time
1	Delayed
1	On time
1	On time
1	On time
1	Delayed
1	Delayed
1	On time
1	Delayed
1	Delayed
1	On time
1	Delayed
1	On time
1	On time
1	On time
1	Delayed
1	Delayed
1	On time

1	On time
1	On time
1	Delayed
1	On time
1	On time
1	On time
1	On time
1	Delayed
1	On time
1	Delayed
1	On time
1	Delayed
1	Delayed
1	On time
1	On time
1	On time
1	On time
1	On time
1	On time
1	On time
1	On time
1	Delayed
1	Delayed
1	On time
1	On time
1	Delayed
1	Delayed
1	Delayed
1	On time
1	Delayed
1	Delayed
1	Delayed
1	On time
1	On time
1	On time
1	On time
1	On time
1	On time
1	On time
1	On time
1	On time
1	On time
1	On time

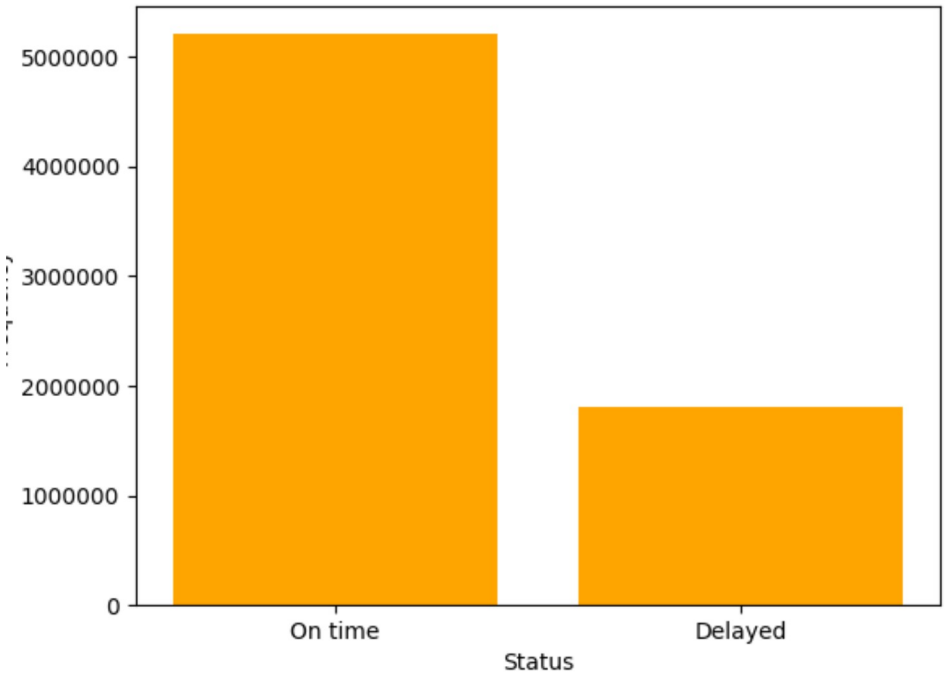
1	Delayed
1	On time
1	On time
1	On time
1	Delayed
1	Delayed
1	On time
1	Delayed
1	On time

Showing the first 1000 rows.

```
df_delvsot=df_delay_Ontime.groupBy('status').count()\
               .withColumnRenamed('count','Frequency')
print(df_delvsot.show())
display(df_delvsot)
```

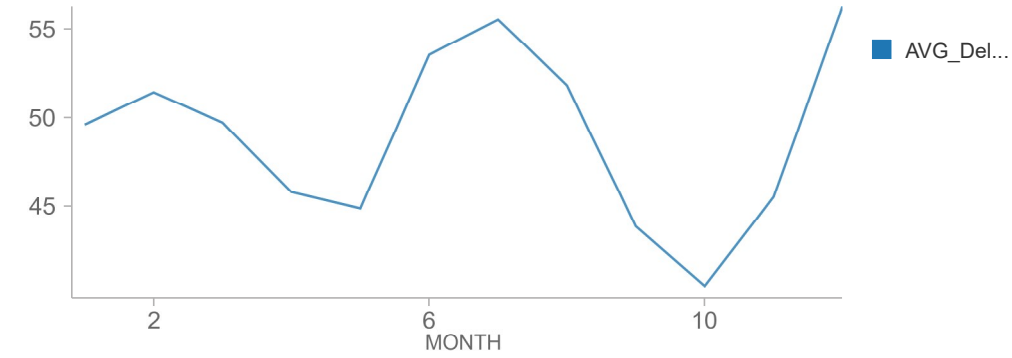



```
#using Pandas
status= df_delvsot.toPandas()['status'].values.tolist()
frequency=df_delvsot.toPandas()['Frequency'].values.tolist()
plt.bar(status,frequency,color='orange')
plt.xlabel('Status')
plt.ylabel('Frequency')
plt.show()
display()
```

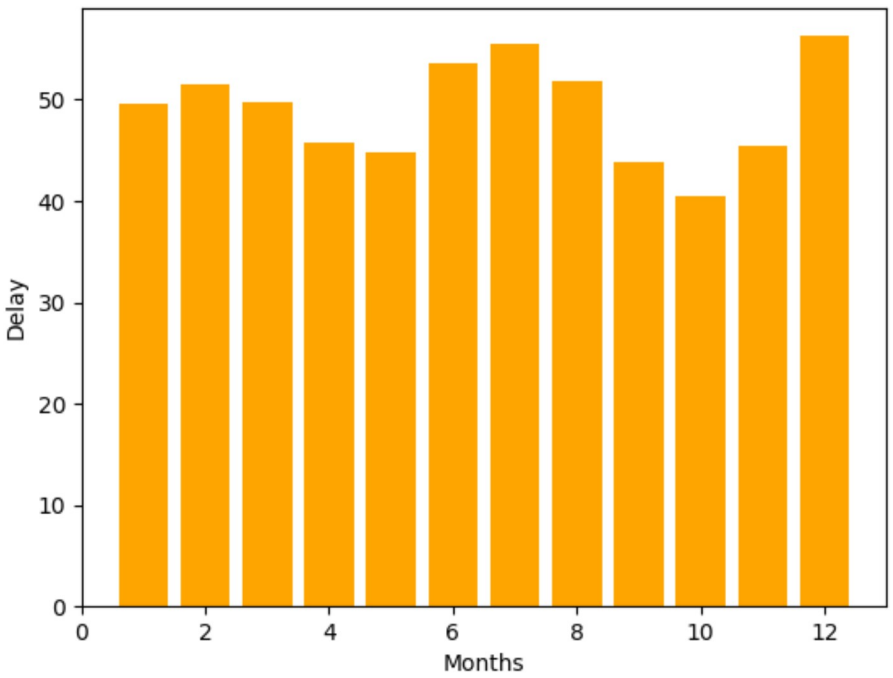


8. Average Delay By Month

```
df_avgdelay_month=df_delay_Ontime.select('ArrDelay','Month').filter("status ==
'Delayed'").groupBy('Month').agg(f.avg("ArrDelay"))\
.withColumnRenamed('avg(ArrDelay)','AVG_Delay')\
.orderBy('AVG_Delay',ascending=False)
df_avgdelay_month.show()
display(df_avgdelay_month)
```

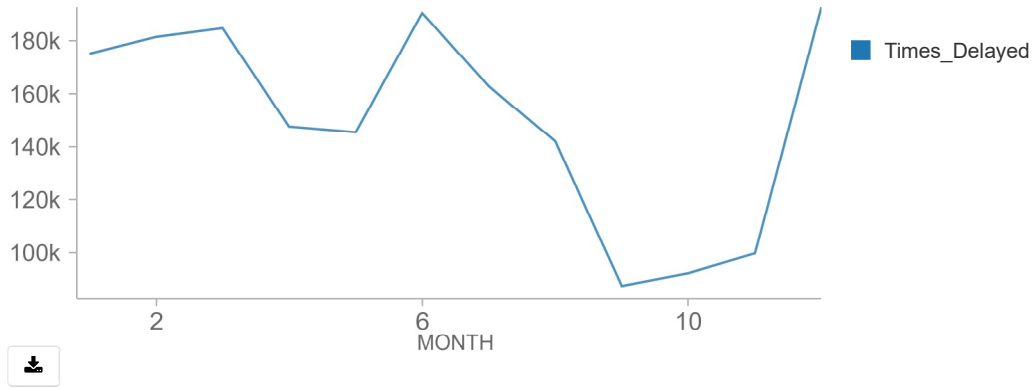


```
#Using pandas
Months= df_avgdelay_month.toPandas()['Month'].values.tolist()
Delay=df_avgdelay_month.toPandas()['AVG_Delay'].values.tolist()
plt.bar(Months,Delay,color='orange')
plt.xlabel('Months')
plt.ylabel('Delay')
plt.show()
display()
```

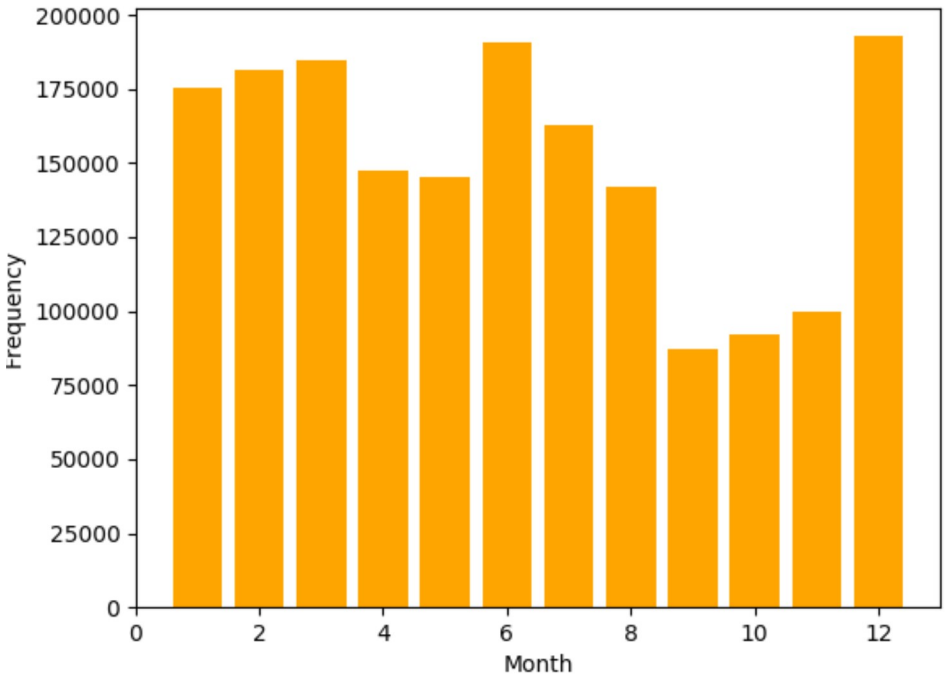


9.Number of times Delayed By Month

```
df_delay_month=df_delay_Ontime.filter("status == 'Delayed'").groupBy('Month').count()\
.withColumnRenamed('count','Times_Delayed')\
.orderBy('Times_Delayed',ascending=False)
display(df_delay_month)
```



```
#Using pandas
Month= df_delay_month.toPandas()['Month'].values.tolist()
frequency=df_delay_month.toPandas()['Times_Delayed'].values.tolist()
plt.bar(Month,frequency,color='orange')
plt.xlabel('Month')
plt.ylabel('Frequency')
plt.show()
display()
```



```
df_delay = df.withColumn(
    'DelayRange',
    #f.when(f.col("ArrDelay").between(5, 10), '05-10')\
    f.when(f.col("ArrDelay").between(11, 15), '10-15')
     .when(f.col("ArrDelay").between(16, 20), '15-20')\
     .when(f.col("ArrDelay").between(21, 25), '20-25')\
     .when(f.col("ArrDelay").between(25, 30), '25-30')\
     .when(f.col("ArrDelay").between(31, 35), '30-35')\
     .when(f.col("ArrDelay").between(36,40), '35-40')\
     .when(f.col("ArrDelay").between(41, 45), '40-45')\
     .when(f.col("ArrDelay").between(46,50), '45-50')\
     .when(f.col("ArrDelay").between(51,55), '51-55')\
     .when(f.col("ArrDelay").between(56,60), '55-60')\
     .when(f.col("ArrDelay").between(61, 65), '60-65')\
     .when(f.col("ArrDelay").between(66, 70), '65-70')\
     .when(f.col("ArrDelay").between(71, 75), '70-75')\
     .when(f.col("ArrDelay").between(75, 80), '75-80')\
     .when(f.col("ArrDelay").between(81, 85), '80-85')\
     .when(f.col("ArrDelay").between(85, 90), '85-90')\
     .when(f.col("ArrDelay").between(91, 95), '90-95')\
     .when(f.col("ArrDelay").between(96 ,100), '95-100')
     .when(f.col("ArrDelay")>105, '>105')\
    .otherwise('On time')
)
display(df_delay)
```

UniqueCarrier	Year	Month	DayofMonth	DayOfWeek	DepTime	CRSDepTime	ArrTime	CRSArr
WN	2008	1	3	4	2003	1955	2211	2225
WN	2008	1	3	4	754	735	1002	1000
WN	2008	1	3	4	628	620	804	750
WN	2008	1	3	4	926	930	1054	1100
WN	2008	1	3	4	1829	1755	1959	1925
WN	2008	1	3	4	1940	1915	2121	2110
WN	2008	1	3	4	1937	1830	2037	1940
WN	2008	1	3	4	1039	1040	1132	1150
WN	2008	1	3	4	617	615	652	650
WN	2008	1	3	4	1620	1620	1639	1655
WN	2008	1	3	4	706	700	916	915
WN	2008	1	3	4	1644	1510	1845	1725

WN	2008	1	3	4	1426	1430	1426	1425
WN	2008	1	3	4	715	715	720	710
WN	2008	1	3	4	1702	1700	1651	1655
WN	2008	1	3	4	1029	1020	1021	1010
WN	2008	1	3	4	1452	1425	1640	1625
WN	2008	1	3	4	754	745	940	955
WN	2008	1	3	4	1323	1255	1526	1510
WN	2008	1	3	4	1416	1325	1512	1435
WN	2008	1	3	4	706	705	807	810
WN	2008	1	3	4	1657	1625	1754	1735
WN	2008	1	3	4	1900	1840	1956	1950
WN	2008	1	3	4	1039	1030	1133	1140
WN	2008	1	3	4	801	800	902	910
WN	2008	1	3	4	1520	1455	1619	1605
WN	2008	1	3	4	1422	1255	1657	1610
WN	2008	1	3	4	1954	1925	2239	2235
WN	2008	1	3	4	636	635	921	945
WN	2008	1	3	4	734	730	958	1020
WN	2008	1	3	4	2107	1945	2334	2230
WN	2008	1	3	4	1008	1005	1234	1255
WN	2008	1	3	4	712	710	953	1000
WN	2008	1	3	4	1312	1300	1546	1550
WN	2008	1	3	4	1449	1430	1715	1720
WN	2008	1	3	4	1634	1555	1859	1845

WN	2008	1	3	4	831	830	935	955
WN	2008	1	3	4	1812	1650	1927	1815
WN	2008	1	3	4	1127	1105	1235	1230
WN	2008	1	3	4	1424	1355	1531	1520
WN	2008	1	3	4	1326	1230	1559	1530
WN	2008	1	3	4	1749	1725	2019	2030
WN	2008	1	3	4	726	720	958	1020
WN	2008	1	3	4	646	640	929	955
WN	2008	1	3	4	1153	1140	1428	1440
WN	2008	1	3	4	1528	1510	1802	1810
WN	2008	1	3	4	634	635	907	935
WN	2008	1	3	4	831	830	1148	1140
WN	2008	1	3	4	1450	1435	1806	1745
WN	2008	1	3	4	2245	1730	2354	1850
WN	2008	1	3	4	615	615	724	735
WN	2008	1	3	4	1150	1145	1303	1305
WN	2008	1	3	4	2025	1940	2135	2100
WN	2008	1	3	4	1038	945	1314	1225
WN	2008	1	3	4	1900	1850	2123	2045
WN	2008	1	3	4	700	700	851	900
WN	2008	1	3	4	948	925	959	940
WN	2008	1	3	4	646	620	725	655
WN	2008	1	3	4	1110	1040	1136	1110
WN	2008	1	3	4	1535	1535	1603	1610

WN	2008	1	3	4	1919	1915	1942	1950
WN	2008	1	3	4	1053	1055	1245	1240
WN	2008	1	3	4	1433	1440	1623	1625
WN	2008	1	3	4	2015	2010	2158	2155
WN	2008	1	3	4	2139	2130	2244	2240
WN	2008	1	3	4	1500	1500	1602	1615
WN	2008	1	3	4	850	850	1000	1000
WN	2008	1	3	4	646	645	752	755
WN	2008	1	3	4	1221	1220	1328	1330
WN	2008	1	3	4	1738	1730	1841	1840
WN	2008	1	3	4	1813	1735	1936	1905
WN	2008	1	3	4	802	750	1001	955
WN	2008	1	3	4	1820	1825	1946	1955
WN	2008	1	3	4	821	820	953	945
WN	2008	1	3	4	1734	1650	1941	1905
WN	2008	1	3	4	712	700	926	915
WN	2008	1	3	4	1318	1310	1410	1400
WN	2008	1	3	4	958	900	1052	950
WN	2008	1	3	4	1859	1850	1950	1945
WN	2008	1	3	4	1538	1445	1753	1710
WN	2008	1	3	4	933	935	1151	1200
WN	2008	1	3	4	2248	2125	102	2345
WN	2008	1	3	4	1327	1230	1550	1500
WN	2008	1	3	4	624	625	846	850

WN	2008	1	3	4	1614	1600	1833	1825
WN	2008	1	3	4	1917	1915	2136	2140
WN	2008	1	3	4	1832	1655	148	30
WN	2008	1	3	4	1229	1155	1633	1555
WN	2008	1	3	4	1256	1240	1724	1720

Showing the first 1000 rows.

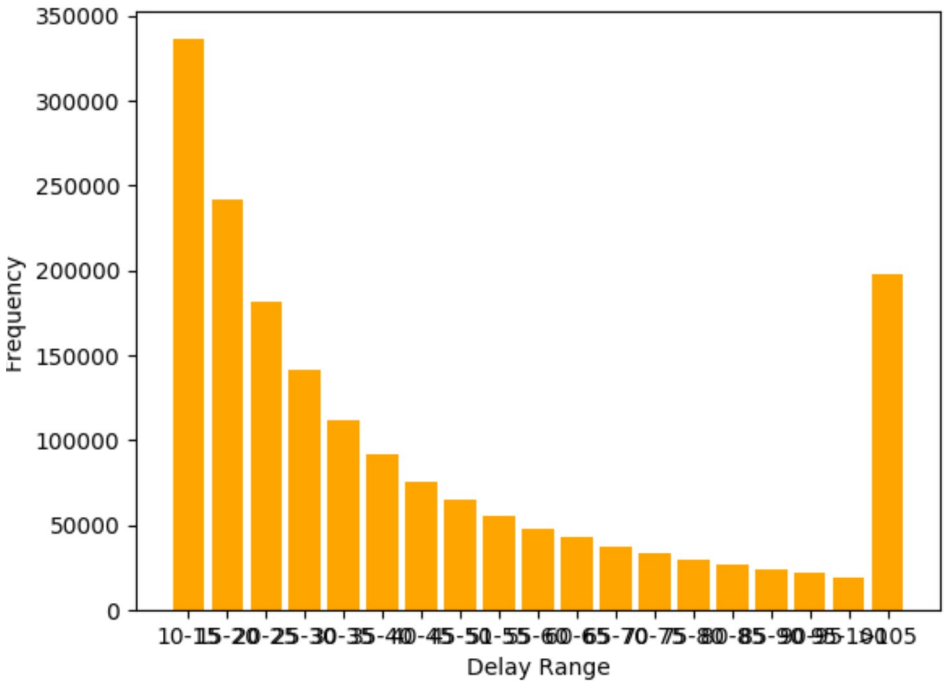
```
#x= df_test.toPandas()['DelayRange'].values.tolist()
```

```
df_delay_frequency=df_delay.select('DelayRange').filter("DelayRange !='On
time'").groupBy('DelayRange').agg(f.count("DelayRange")).\
withColumnRenamed('count(DelayRange)','Frequency')\
.orderBy('DelayRange')
df_delay_frequency.show()
```

```
+----------+---------+
|DelayRange|Frequency|
+----------+---------+
|     10-15|   336163|
|     15-20|   241678|
|     20-25|   181826|
|     25-30|   141289|
|     30-35|   112379|
|     35-40|    91750|
|     40-45|    76030|
|     45-50|    64794|
|     51-55|    55924|
|     55-60|    48203|
|     60-65|    42798|
|     65-70|    37364|
|     70-75|    33490|
|     75-80|    30028|
|     80-85|    27127|
|     85-90|    24022|
|     90-95|    21925|
|    95-100|    19672|
|      >105|   197940|
+----------+---------+
```

```
x1= df_delay_frequency.toPandas()['DelayRange'].values.tolist()
y1=df_delay_frequency.toPandas()['Frequency'].values.tolist()
```

```
plt.bar(x1,y1,color='orange')
plt.xlabel('Delay Range')
plt.ylabel('Frequency')
plt.show()
display()
```



Model- Predicting Delays

```
from pyspark.ml.regression import LinearRegression
```

Feature Selection

```
df_model=df.select('UniqueCarrier','Dest','Origin','Month','DayOfWeek','ArrTime','Distance','DepDel
ay')
df_model=df_model.filter("DepDelay>0")
```

```
df_model.printSchema()
```

```
root
 |-- UniqueCarrier: string (nullable = true)
 |-- Dest: string (nullable = true)
 |-- Origin: string (nullable = true)
 |-- Month: integer (nullable = true)
 |-- DayOfWeek: integer (nullable = true)
```

```
 |-- ArrTime: string (nullable = true)
 |-- Distance: integer (nullable = true)
 |-- DepDelay: string (nullable = true)
```

```
from pyspark.sql.types import IntegerType
df_model = df_model.withColumn("DepDelay", df_model["DepDelay"].cast(IntegerType())).\
                withColumn("ArrTime", df_model["ArrTime"].cast(IntegerType()))
df_model.printSchema()
```

```
root
 |-- UniqueCarrier: string (nullable = true)
 |-- Dest: string (nullable = true)
 |-- Origin: string (nullable = true)
 |-- Month: integer (nullable = true)
 |-- DayOfWeek: integer (nullable = true)
 |-- ArrTime: integer (nullable = true)
 |-- Distance: integer (nullable = true)
 |-- DepDelay: integer (nullable = true)
```

```
df_model.describe().show()
```

```
+-------+-------------+-------+-------+------------------+------------------+------------------+--
--------------+------------------+
|summary|UniqueCarrier|   Dest| Origin|             Month|         DayOfWeek|           ArrTime|
Distance|          DepDelay|
+-------+-------------+-------+-------+------------------+------------------+------------------+--
--------------+------------------+
|  count|      2700974|2700974|2700974|           2700974|           2700974|           2692327|
2700974|           2700974|
|   mean|         null|   null|   null|6.1661148904062015|3.9751600718851794|1579.5555001305563|77
6.000673460759| 31.70540108864432|
| stddev|         null|   null|   null| 3.455846708763988|1.9954671838826175| 529.8201694993356|58
3.022972919806|48.779906120990525|
|    min|           9E|    ABE|    ABE|                 1|                 1|                 1|
11|                 1|
|    max|           YV|    YUM|    YUM|                12|                 7|              2400|
4962|              2467|
+-------+-------------+-------+-------+------------------+------------------+------------------+--
--------------+------------------+
```

```
display(df_model)
```

UniqueCarrier	Dest	Origin	Month	DayOfWeek
WN	TPA	IAD	1	4
WN	TPA	IAD	1	4
WN	BWI	IND	1	4
WN	BWI	IND	1	4
WN	JAX	IND	1	4
WN	LAS	IND	1	4
WN	MCI	IND	1	4
WN	MCO	IND	1	4

WN	MCO	IND	1	4
WN	MDW	IND	1	4
WN	MDW	IND	1	4
WN	PHX	IND	1	4
WN	PHX	IND	1	4
WN	TPA	IND	1	4
WN	BWI	ISP	1	4
WN	BWI	ISP	1	4
WN	BWI	ISP	1	4
WN	BWI	ISP	1	4
WN	BWI	ISP	1	4
WN	BWI	ISP	1	4
WN	BWI	ISP	1	4
WN	FLL	ISP	1	4
WN	FLL	ISP	1	4
WN	FLL	ISP	1	4
WN	LAS	ISP	1	4
WN	MCO	ISP	1	4
WN	MCO	ISP	1	4
WN	MCO	ISP	1	4
WN	MCO	ISP	1	4
WN	MCO	ISP	1	4
WN	MCO	ISP	1	4
WN	MDW	ISP	1	4
WN	MDW	ISP	1	4
WN	MDW	ISP	1	4
WN	MDW	ISP	1	4
WN	PBI	ISP	1	4
WN	PBI	ISP	1	4
WN	PBI	ISP	1	4
WN	RSW	ISP	1	4
WN	TPA	ISP	1	4
WN	TPA	ISP	1	4
WN	BWI	JAN	1	4
WN	BWI	JAN	1	4
WN	HOU	JAN	1	4
WN	HOU	JAN	1	4
WN	HOU	JAN	1	4
WN	MCO	JAN	1	4
WN	MDW	JAN	1	4
WN	BHM	JAX	1	4
WN	BNA	JAX	1	4

WN	BNA	JAX	1	4
WN	BNA	JAX	1	4
WN	BWI	JAX	1	4
WN	FLL	JAX	1	4
WN	FLL	JAX	1	4
WN	FLL	JAX	1	4
WN	FLL	JAX	1	4
WN	HOU	JAX	1	4
WN	IND	JAX	1	4
WN	ORF	JAX	1	4
WN	PHL	JAX	1	4
WN	PHL	JAX	1	4
WN	TPA	JAX	1	4
WN	TPA	JAX	1	4
WN	TPA	JAX	1	4
WN	ABQ	LAS	1	4
WN	ABQ	LAS	1	4
WN	ABQ	LAS	1	4
WN	ABQ	LAS	1	4
WN	ABQ	LAS	1	4
WN	ALB	LAS	1	4
WN	AMA	LAS	1	4
WN	AUS	LAS	1	4
WN	AUS	LAS	1	4
WN	AUS	LAS	1	4
WN	BDL	LAS	1	4
WN	BHM	LAS	1	4
WN	BNA	LAS	1	4
WN	BNA	LAS	1	4
WN	BNA	LAS	1	4
WN	BNA	LAS	1	4
WN	BOI	LAS	1	4
WN	BOI	LAS	1	4
WN	BUF	LAS	1	4
WN	BUR	LAS	1	4
WN	BUR	LAS	1	4
WN	BUR	LAS	1	4
WN	BUR	LAS	1	4
WN	BUR	LAS	1	4
WN	BUR	LAS	1	4
WN	BUR	LAS	1	4
WN	BUR	LAS	1	4

WN	BUR	LAS	1	4
WN	BUR	LAS	1	4
WN	BUR	LAS	1	4
WN	BWI	LAS	1	4
WN	BWI	LAS	1	4
WN	BWI	LAS	1	4
WN	CLE	LAS	1	4
WN	CMH	LAS	1	4

Showing the first 1000 rows.

```
df_model=df_model.dropna()
```

```
print(df_model.select('UniqueCarrier').distinct().count())
print(df_model.select('Dest').distinct().count())
print(df_model.select('Origin').distinct().count())
```

```
20
302
303
```

```
from pyspark.ml.feature import VectorAssembler,StringIndexer,StandardScaler
from pyspark.ml import Pipeline
```

```
UniqueCarrier_indexer =
StringIndexer(inputCol='UniqueCarrier',outputCol='UniqueCarrier_index',handleInvalid='keep')
Dest_indexer = StringIndexer(inputCol='Dest',outputCol='Dest_index',handleInvalid='keep')
Origin_indexer = StringIndexer(inputCol='Origin',outputCol='Origin_index',handleInvalid='keep')
```

```
assembler = VectorAssembler(inputCols=
['UniqueCarrier_index','Dest_index','Origin_index','Month','ArrTime','Distance','DayOfWeek'],
                            outputCol="features")
```

```
lr = LinearRegression(labelCol='DepDelay')
```

```
pipe = Pipeline(stages=[UniqueCarrier_indexer,Dest_indexer,
                        Origin_indexer,assembler])
```

```
pre_model=pipe.fit(df_model)
```

```
preprocessed_data = pre_model.transform(df_model)
```

```
preprocessed_data.show()

df_vector =preprocessed_data.select('features')
vector_col = "features"
```

```
+-------------+----+------+-----+---------+-------+--------+--------+------------------+---------
-+------------+--------------------+
|UniqueCarrier|Dest|Origin|Month|DayOfWeek|ArrTime|Distance|DepDelay|UniqueCarrier_index|Dest_inde
x|Origin_index|            features|
+-------------+----+------+-----+---------+-------+--------+--------+------------------+---------
-+------------+--------------------+
|           WN| TPA|   IAD|    1|        4|   2211|     810|       8|               0.0|      23.
0|        27.0|[0.0,23.0,27.0,1....|
|           WN| TPA|   IAD|    1|        4|   1002|     810|      19|               0.0|      23.
0|        27.0|[0.0,23.0,27.0,1....|
|           WN| BWI|   IND|    1|        4|    804|     515|       8|               0.0|      14.
0|        50.0|[0.0,14.0,50.0,1....|
|           WN| BWI|   IND|    1|        4|   1959|     515|      34|               0.0|      14.
0|        50.0|[0.0,14.0,50.0,1....|
|           WN| JAX|   IND|    1|        4|   2121|     688|      25|               0.0|      53.
0|        50.0|[0.0,53.0,50.0,1....|
|           WN| LAS|   IND|    1|        4|   2037|    1591|      67|               0.0|       5.
0|        50.0|[0.0,5.0,50.0,1.0...|
|           WN| MCI|   IND|    1|        4|    652|     451|       2|               0.0|      31.
0|        50.0|[0.0,31.0,50.0,1....|
|           WN| MCO|   IND|    1|        4|    916|     828|       6|               0.0|      10.
0|        50.0|[0.0,10.0,50.0,1....|
|           WN| MCO|   IND|    1|        4|   1845|     828|      94|               0.0|      10.
0|        50.0|[0.0,10.0,50.0,1....|
|           WN| MDW|   IND|    1|        4|   1651|     162|       2|               0.0|      18.
0|        50.0|[0.0,18.0,50.0,1....|
|           WN| MDW|   IND|    1|        4|   1021|     162|       9|               0.0|      18.
0|        50.0|[0.0,18.0,50.0,1....|
|           WN| PHX|   IND|    1|        4|   1640|    1489|      27|               0.0|       6.
0|        50.0|[0.0,6.0,50.0,1.0...|
|           WN| PHX|   IND|    1|        4|    940|    1489|       9|               0.0|       6.
0|        50.0|[0.0,6.0,50.0,1.0...|
|           WN| TPA|   IND|    1|        4|   1526|     838|      28|               0.0|      23.
0|        50.0|[0.0,23.0,50.0,1....|
|           WN| BWI|   ISP|    1|        4|   1512|     220|      51|               0.0|      14.
0|        78.0|[0.0,14.0,78.0,1....|
|           WN| BWI|   ISP|    1|        4|    807|     220|       1|               0.0|      14.
0|        78.0|[0.0,14.0,78.0,1....|
|           WN| BWI|   ISP|    1|        4|   1754|     220|      32|               0.0|      14.
0|        78.0|[0.0,14.0,78.0,1....|
|           WN| BWI|   ISP|    1|        4|   1956|     220|      20|               0.0|      14.
0|        78.0|[0.0,14.0,78.0,1....|
|           WN| BWI|   ISP|    1|        4|   1133|     220|       9|               0.0|      14.
0|        78.0|[0.0,14.0,78.0,1....|
|           WN| BWI|   ISP|    1|        4|    902|     220|       1|               0.0|      14.
0|        78.0|[0.0,14.0,78.0,1....|
+-------------+----+------+-----+---------+-------+--------+--------+------------------+---------
-+------------+--------------------+
only showing top 20 rows
```

```
print(preprocessed_data.stat.corr('DepDelay','Month'))
print(preprocessed_data.stat.corr('DepDelay','DayOfWeek'))
print(preprocessed_data.stat.corr('DepDelay','UniqueCarrier_index'))
print(preprocessed_data.stat.corr('DepDelay','Dest_index'))
print(preprocessed_data.stat.corr('DepDelay','Origin_index'))
print(preprocessed_data.stat.corr('DepDelay','ArrTime'))
print(preprocessed_data.stat.corr('DepDelay','Distance'))
```

```
-0.0061332077333529945
0.009874264782662692
0.06168428828096052
-0.011142870468241457
0.043770409443521244
-0.008168415447568163
-0.016546249750661216
```

```
'''from pyspark.ml.stat import Correlation
matrix = Correlation.corr(df_vector,vector_col)'''
```

```
Out[98]: 'from pyspark.ml.stat import Correlation\nmatrix = Correlation.corr(df_vector,vector_co
l)'
```

```
'''matrix.collect()[0]["pearson({})".format(vector_col)].values'''
```

```
Out[99]: 'matrix.collect()[0]["pearson({})".format(vector_col)].values'
```

```
train_data,test_data=preprocessed_data.randomSplit([0.8,0.2])
```

```
lrModel = lr.fit(train_data)
```

```
print("Coefficients: {} Intercept: {}".format(lrModel.coefficients,lrModel.intercept))
```

```
Coefficients: [0.5927692059750387,-0.018219435665087873,0.038815060565711657,-0.0893646849588594
1,-0.0006490763116706497,-0.001078903814074715,0.23982274903884576] Intercept: 29.42397091549423
```

```
predicted=lrModel.transform(test_data)
```

```
predicted.show()
```

```
+-------------+----+------+-----+---------+-------+--------+--------+------------------+---------
-+------------+--------------------+------------------+
|UniqueCarrier|Dest|Origin|Month|DayOfWeek|ArrTime|Distance|DepDelay|UniqueCarrier_index|Dest_inde
x|Origin_index|            features|        prediction|
+-------------+----+------+-----+---------+-------+--------+--------+------------------+---------
-+------------+--------------------+------------------+
|           9E| ABE|   DTW|    1|        2|   2326|     424|      38|              14.0|     146.
0|         9.0|[14.0,146.0,9.0,1...| 33.83511183213857|
|           9E| ABE|   DTW|    1|        5|   1551|     424|      23|              14.0|     146.
0|         9.0|[14.0,146.0,9.0,1...| 35.05761422079986|
|           9E| ALB|   DTW|    1|        4|   1234|     488|      22|              14.0|      78.
0|         9.0|[14.0,78.0,9.0,1....|  36.1934204436858|
|           9E| ALB|   DTW|    1|        4|   1255|     488|      61|              14.0|      78.
```

```
0|         9.0|[14.0,78.0,9.0,1....| 36.17978984114072|
|           9E| ALB|   MSP|    1|        3|   1644|     979|      11|               14.0|      78.
0|        15.0|[14.0,78.0,15.0,1...| 35.39062499754558|
|           9E| ALO|   MSP|    1|        3|   2325|     166|       5|               14.0|     283.
0|        15.0|[14.0,283.0,15.0,...|32.090768518797596|
|           9E| ALO|   MSP|    1|        4|   2331|     166|      14|               14.0|     283.
0|        15.0|[14.0,283.0,15.0,...|32.326696809966414|
|           9E| ATL|   HOU|    1|        1|   1302|     696|     172|               14.0|       0.
0|        25.0|[14.0,0.0,25.0,1....| 37.24755996497636|
|           9E| ATL|   HOU|    1|        3|   1037|     696|       3|               14.0|       0.
0|        25.0|[14.0,0.0,25.0,1....| 37.89921068564678|
|           9E| ATL|   HOU|    1|        3|   1557|     696|       5|               14.0|       0.
0|        25.0|[14.0,0.0,25.0,1....| 37.56169100357804|
|           9E| ATL|   HOU|    1|        4|   1645|     696|      40|               14.0|       0.
0|        25.0|[14.0,0.0,25.0,1....| 37.74439503718987|
|           9E| ATL|   HOU|    1|        5|   1557|     696|      19|               14.0|       0.
0|        25.0|[14.0,0.0,25.0,1....|38.041336501655735|
|           9E| ATL|   HOU|    1|        5|   1600|     696|      17|               14.0|       0.
0|        25.0|[14.0,0.0,25.0,1....| 38.01342622025389|
|           9E| ATL|   HOU|    1|        6|   1350|     696|     105|               14.0|       0.
0|        25.0|[14.0,0.0,25.0,1....|  38.4155180472104|
|           9E| ATL|   HOU|    1|        7|   1445|     696|     149|               14.0|       0.
0|        25.0|[14.0,0.0,25.0,1....| 38.59367854664053|
|           9E| ATL|   HOU|    1|        7|   1551|     696|       3|               14.0|       0.
0|        25.0|[14.0,0.0,25.0,1....| 38.52487645760345|
|           9E| ATL|   IAH|    1|        1|   1752|     689|       6|               14.0|       0.
0|         5.0|[14.0,0.0,5.0,1.0...| 36.18672674010886|
|           9E| ATL|   IAH|    1|        2|   2140|     689|      15|               14.0|       0.
0|         5.0|[14.0,0.0,5.0,1.0...| 36.17470788021949|
|           9E| ATL|   IAH|    1|        4|   1803|     689|      13|               14.0|       0.
0|         5.0|[14.0,0.0,5.0,1.0...| 36.87309209533019|
|           9E| ATL|   IAH|    1|        4|   2127|     689|       9|               14.0|       0.
0|         5.0|[14.0,0.0,5.0,1.0...|36.662791370348906|
+-------------+----+------+-----+---------+-------+--------+--------+-------------------+---------
-+------------+--------------------+------------------+
only showing top 20 rows
```

```
  test_results = lrModel.evaluate(test_data)
```

```
test_results.residuals.show()
```

```
+-------------------+
|          residuals|
+-------------------+
|  4.164888167861427|
|-12.057614220799863|
|-14.193420443685802|
| 24.820210158859283|
| -24.39062499754558|
|-27.090768518797596|
|-18.326696809966414|
| 134.75244003502365|
| -34.89921068564678|
| -32.56169100357804|
|  2.255604962810132|
|-19.041336501655735|
```

```
| -21.01342622025389|
|   66.5844819527896|
| 110.40632145335947|
| -35.52487645760345|
| -30.18672674010886|
|-21.174707880219493|
|-23.873092095330193|
|-27.662791370348906|
+-------------------+
only showing top 20 rows
```

```
print("RMSE: {}".format(test_results.rootMeanSquaredError))
print("MSE: {}".format(test_results.meanSquaredError))
```

```
RMSE: 48.32026323637107
MSE: 2334.8478392321936
```

Using OneHotEncoder

```
from pyspark.ml.feature import OneHotEncoderEstimator
data_encoder = OneHotEncoderEstimator(inputCols=
['UniqueCarrier_index','Dest_index','Origin_index'],
                                      outputCols=['UniqueCarrier_vec','Dest_vec','Origin_vec',],
                                      handleInvalid='keep')
assembler_enc = VectorAssembler(inputCols=
['UniqueCarrier_vec','Dest_vec','Origin_vec','Month','ArrTime','Distance','DayOfWeek'],
                           outputCol="features")

#assembler_enc = VectorAssembler(inputCols=['UniqueCarrier_index','Dest_index','Origin_index'],
 #                          outputCol="features")
```

```
df_model = df_model.orderBy(f.rand(seed=123))
df_model.show(n=10)
```

```
+-------------+----+------+-----+---------+-------+--------+--------+
|UniqueCarrier|Dest|Origin|Month|DayOfWeek|ArrTime|Distance|DepDelay|
+-------------+----+------+-----+---------+-------+--------+--------+
|           UA| ORD|   DEN|    5|        5|   1956|     888|      47|
|           9E| RIC|   DTW|    4|        7|   1813|     456|     150|
|           EV| SBN|   ATL|    9|        4|   1157|     566|       7|
|           OH| CVG|   LEX|    1|        2|   1958|      70|      60|
|           US| LGA|   BOS|    9|        7|   1143|     185|      32|
|           WN| PVD|   BWI|   12|        1|    926|     328|       4|
|           AS| SEA|   LAX|    7|        7|   1228|     954|      13|
|           WN| HOU|   DAL|    4|        2|   2115|     239|     114|
|           WN| SJC|   MDW|    2|        3|   1136|    1838|      10|
|           DL| ATL|   DAB|    3|        4|   1413|     366|       7|
+-------------+----+------+-----+---------+-------+--------+--------+
only showing top 10 rows
```

```
df_model=df_model.filter("DepDelay<200")
df_model.describe().show()
```

```
+-------+-------------+-------+-------+-----------------+------------------+------------------+---
```

```
--------------+------------------+
|summary|UniqueCarrier|   Dest| Origin|            Month|         DayOfWeek|           ArrTime|
Distance|          DepDelay|
+-------+-------------+-------+-------+-----------------+------------------+------------------+---
--------------+------------------+
|  count|      2653391|2653391|2653391|          2653391|           2653391|           2653391|
2653391|           2653391|
|   mean|         null|   null|   null|6.164518534961489|3.9734083668784583|1583.1839928604568|  7
75.15023379517| 27.95196750120883|
| stddev|         null|   null|   null| 3.45483253575987|1.9942789130014908| 522.4908318948118|58
2.2283397227719|35.636240611975204|
|    min|           9E|    ABE|    ABE|                1|                 1|                 1|
11|                1|
|    max|           YV|    YUM|    YUM|               12|                 7|              2400|
4962|              199|
+-------+-------------+-------+-------+-----------------+------------------+------------------+---
--------------+------------------+
```

```
pipe_encoder = Pipeline(stages=[UniqueCarrier_indexer,Dest_indexer,
                        Origin_indexer,data_encoder,assembler_enc])
pre_model=pipe_encoder.fit(df_model)
preprocessed_data = pre_model.transform(df_model)
```

```
'''from pyspark.ml.feature import PolynomialExpansion
polyExpansion = PolynomialExpansion(degree=2, inputCol="features", outputCol="polyFeatures")
polyDF = polyExpansion.transform(preprocessed_data)
polyDF.select('polyFeatures').show()'''
```

```
+--------------------+
|        polyFeatures|
+--------------------+
|[0.0,0.0,0.0,4.0,...|
|[12.0,144.0,1728....|
|[8.0,64.0,512.0,7...|
|[8.0,64.0,512.0,7...|
|[1.0,1.0,1.0,10.0...|
|[2.0,4.0,8.0,4.0,...|
|[1.0,1.0,1.0,4.0,...|
|[1.0,1.0,1.0,63.0...|
|[2.0,4.0,8.0,28.0...|
|[2.0,4.0,8.0,59.0...|
|[3.0,9.0,27.0,1.0...|
|[8.0,64.0,512.0,7...|
|[8.0,64.0,512.0,7...|
|[12.0,144.0,1728....|
|[0.0,0.0,0.0,19.0...|
|[5.0,25.0,125.0,1...|
|[16.0,256.0,4096....|
|[1.0,1.0,1.0,1.0,...|
|[14.0,196.0,2744....|
|[1.0,1.0,1.0,53.0...|
+--------------------+
only showing top 20 rows
```

```
lr = LinearRegression(labelCol='DepDelay',maxIter=100, regParam=0.8, elasticNetParam=0.8)
```

```
train_data,test_data=preprocessed_data.randomSplit([0.8,0.2])
lrModel = lr.fit(train_data)
print("Coefficients: {} Intercept: {}".format(lrModel.coefficients,lrModel.intercept))
```

```
Coefficients: [-6.280384276160379,0.13022193109331723,0.0,0.0,0.0,-1.0799384746426426,-2.000973099
4697046,-1.7475384301623833,0.46929218502184833,0.0,1.8071404435752048,0.0,8.075884987595762,3.625
685360129759,0.0,6.353378738276372,0.0,-4.528518571895247,-3.1758630439194815,0.0,0.0,0.0,7.974959
294132355,0.0,0.0,0.0,0.0,0.0,0.0,11.63442995298102,0.0,3.424806092027378,0.0,3.4769085429271094,0.0,
0.0,0.0,0.0,1.5332111618231663,0.0,0.23972600293802995,0.0,0.0,0.0,0.0,0.0,0.0,0.0,0.0,0.0,0.0,0.
0,0.0,0.0,0.0,0.0,0.0,0.0,0.0,0.0,0.0,0.0,0.0,0.0,0.0,0.0,0.0,0.0,0.0,0.0,0.0,0.0,0.0,0.0,0.0,0.0,
0.0,0.0,0.0,0.0,0.0,0.0,0.0,0.0,0.0,0.0,0.0,0.0,0.0,0.0,0.0,0.0,0.0,0.0,0.0,0.0,0.0,0.0,0.0,0.0,0.
0,0.0,0.0,0.0,0.0,0.0,0.0,0.0,0.0,0.0,0.0,0.0,0.0,0.0,0.0,0.0,0.0,0.0,0.0,0.0,0.0,0.0,0.0,0.0,0.0,
0.0,0.0,0.0,0.0,0.0,0.0,0.0,0.0,0.0,0.0,0.0,0.0,0.0,0.0,0.0,0.0,0.0,0.0,0.0,0.0,0.0,0.0,0.0,0.0,0.
0,0.0,0.0,0.0,0.0,0.0,0.0,0.0,0.0,0.0,0.0,0.0,0.0,0.0,0.0,0.0,0.0,0.0,0.0,0.0,0.0,0.0,0.0,0.0,0.0,
0.0,0.0,0.0,0.0,0.0,0.0,0.0,0.0,0.0,0.0,0.0,0.0,0.0,0.0,0.0,0.0,0.0,0.0,0.0,0.0,0.0,0.0,0.0,0.0,0.
0,0.0,0.0,0.0,0.0,0.0,0.0,0.0,0.0,0.0,0.0,0.0,0.0,0.0,0.0,0.0,0.0,0.0,0.0,0.0,0.0,0.0,0.0,0.0,0.0,
0.0,0.0,0.0,0.0,0.0,0.0,0.0,0.0,0.0,0.0,0.0,0.0,0.0,0.0,0.0,0.0,0.0,0.0,0.0,0.0,0.0,0.0,0.0,0.0,0.
0,0.0,0.0,0.0,0.0,0.0,0.0,0.0,0.0,0.0,0.0,0.0,0.0,0.0,0.0,0.0,0.0,0.0,0.0,0.0,0.0,0.0,0.0,0.0,0.0,
0.0,0.0,0.0,0.0,0.0,0.0,0.0,0.0,0.0,0.0,0.0,0.0,0.0,0.0,0.0,0.0,0.0,0.0,0.0,0.0,0.0,0.0,0.0,0.0,0.
0,0.0,0.0,0.0,0.0,0.0,0.0,0.0,0.0,0.0,0.0,0.0,0.0,0.0,0.0,0.0,0.0,0.0,0.0,0.0,0.0,0.0,0.0,0.0,0.0,
0.0,0.0,0.0,0.0,0.0,0.0,0.0,4.821732360188146,0.0,0.0,0.0,-1.2933131260242174,0.0,0.0,2.35185884429875,0.
0,2.0867754946035553,0.0,0.0,0.0,0.0,0.0,-0.03550577161548663,0.0,0.0,0.0,1.5253697520891754,0.0,
0.0,0.0,0.0,0.0,0.0,0.0,0.0,0.0,0.0,0.0,0.0,0.0,0.0,0.0,0.0,0.0,0.0,0.0,0.0,0.0,0.0,0.0,0.0,0.0,0.
0,0.0,-0.350834582889176,0.0,0.0,0.0,0.0,0.0,0.0,0.0,0.0,0.0,0.0,0.0,0.0,0.0,0.0,0.0,0.0,0.0,0.0,
0.0,0.0,0.0,0.0,0.0,0.0,0.0,0.0,0.0,0.0,0.0,0.0,0.0,0.0,0.0,0.0,0.0,0.0,0.0,0.0,0.0,0.0,0.0,0.0,0.
0,0.0,0.0,0.0,0.0,0.0,0.0,0.0,0.0,0.0,0.0,0.0,0.0,0.0,0.0,0.0,0.0,0.0,0.0,0.0,0.0,0.0,0.0,0.0,0.0,
0.0,0.0,0.0,0.0,0.0,0.0,0.0,0.0,0.0,0.0,0.0,0.0,0.0,0.0,0.0,0.0,0.0,0.0,0.0,0.0,0.0,0.0,0.0,0.0,0.
0,0.0,0.0,0.0,0.0,0.0,0.0,0.0,0.0,0.0,0.0,0.0,0.0,0.0,0.0,0.0,0.0,0.0,0.0,0.0,0.0,0.0,0.0,0.0,0.0,
0.0,0.0,0.0,0.0,0.0,0.0,0.0,0.0,0.0,0.0,0.0,0.0,0.0,0.0,0.0,0.0,0.0,0.0,0.0,0.0,0.0,0.0,0.0,0.0,0.
0,0.0,0.0,0.0,0.0,0.0,0.0,0.0,0.0,0.0,0.0,0.0,0.0,0.0,0.0,0.0,0.0,0.0,0.0,0.0,0.0,0.0,0.0,0.0,0.0,
0.0,0.0,0.0,0.0,0.0,0.0,0.0,0.0,0.0,0.0,0.0,0.0,0.0,0.0,0.0,0.0,0.0,0.0,0.0,0.0,0.0,0.0,0.0,0.0,0.
0,0.0,0.0,0.0,0.0,0.0,0.0,0.0,0.0,0.0,0.0,0.0,0.0,0.0,0.0,0.0,0.0,0.0,0.0,0.0,0.0,0.0,0.0,0.0,0.0,
0.0,0.0,0.0,0.0,0.0,0.0,0.0,0.0,0.0,0.0,0.0,0.0,0.0,0.0,0.0,0.0,0.0,0.0,0.0,0.0,0.0,0.0,0.0,0.0,0.
0,0.0,0.0,0.0,0.0,0.0,0.0,0.0,0.0,0.0,0.0,0.0,0.0,0.0,0.0,0.0,0.0,0.0,0.0015351944665376515,-0.000
9742381423745025,0.0] Intercept: 26.302624467361994
```

```
predicted=lrModel.transform(test_data)
predicted.select('DepDelay','prediction').show()
```

```
+--------+------------------+
|DepDelay|        prediction|
+--------+------------------+
|       1|28.434194391661322|
|      24|28.868420628063397|
|      80|27.356316663899904|
|      34| 28.87287646897252|
|      16|29.149269922356318|
|       8| 28.76907911323311|
|       2| 29.46153173681011|
|       9| 27.52010040067152|
|      74|28.218526725831754|
|       4|26.939008342321653|
|       1|29.234620372758975|
|      15|27.141389438584632|
|      70| 29.50920475242583|
|      33|29.233438976711362|
```

```
|      87|29.065477170173967|
|       6|27.624199309376444|
|      24|27.800483133700553|
|      23| 28.91196392747381|
|       3|28.603531310252084|
|      16|27.465897897109738|
+--------+------------------+
only showing top 20 rows
```

```
test_results = lrModel.evaluate(test_data)
test_results.residuals.show()
```

```
+-------------------+
|          residuals|
+-------------------+
|  71.15944873794689|
| 12.627806070595867|
| 2.9322761303078586|
| 39.367502210582884|
| -4.132002345082643|
|-24.710549603652048|
| 13.380615272157666|
|-26.608402273208675|
|  59.45952959568227|
|-16.111124664557945|
| -27.22195401516056|
|-13.383559613988417|
|  9.806544062511332|
|-27.508297007873338|
|-22.882851386276798|
|-23.143537256347038|
|  11.43738485149839|
|-23.186970439378726|
|  -8.64392740353771|
|-25.809396883503336|
+-------------------+
only showing top 20 rows
```

```
print("RMSE: {}".format(test_results.rootMeanSquaredError))
print("MSE: {}".format(test_results.meanSquaredError))
```

```
RMSE: 35.042989241330005
MSE: 1228.0110949679706
```

Decision Tree

```
from pyspark.ml.regression import DecisionTreeRegressor
from pyspark.ml.evaluation import RegressionEvaluator
```

```
train_data,test_data=preprocessed_data.randomSplit([0.8,0.2])
```

```
Root Mean Squared Error (RMSE) on test data = 34.0353
```

databricks Final Project

Query to get the count of all the records in the table named 'data' which has all the information about flights in the year 2008

```
%sql
select count(*) from data where Year = 2008;
```

count(1)
7009728

Routes (Origin, Destination) with the maximum overall delay and the type of delay (carrier delay, weather delay,..) in the year 2008

Inorder to get the total delay a flight has got, we have considered both the arrival delay and departure delay. We have came to a conclusion that, the total delay associated with the flight would be the maximum value of both the arrival delay and depature delay.

```
%sql
select delay.airport as source,B.airport as destination,
sum(TotalDelay) as Total_delay_in_2008,
sum(CarrierDelay) as CarrierDelay_in_2008,
sum(WeatherDelay) as WeatherDelay_in_2008,
sum(NASDelay) as NASDelay_in_2008,
sum(SecurityDelay) as SecurityDelay_in_2008,
sum(LateAircraftDelay) as LateAircraftDelay_in_2008
from
(select airport,Dest,
case when ArrDelay>DepDelay then ArrDelay else DepDelay end as TotalDelay,
CarrierDelay,WeatherDelay,NASDelay,SecurityDelay,LateAircraftDelay from data,airports
where data.Origin = airports.iata having TotalDelay>0) delay, airports B
where delay.Dest = B.iata
group by delay.airport,B.airport order by Total_delay_in_2008 desc LIMIT 10;
```

source	destination	Total_delay_in_2008	CarrierDelay_in_2008	We
Chicago O'Hare International	LaGuardia	277079	32254	875
LaGuardia	Chicago O'Hare International	255587	21452	487
Chicago O'Hare International	Newark Intl	252264	22215	665
Los Angeles International	San Francisco International	243661	32639	687
William B Hartsfield-Atlanta Intl	Newark Intl	216651	23520	482
William B Hartsfield-Atlanta Intl	LaGuardia	214998	27288	268
Newark Intl	Chicago O'Hare International	212555	20713	511

San Francisco International	Los Angeles International	201929	30445	113
LaGuardia	William B Hartsfield-Atlanta Intl	199128	22074	455
Newark Intl	William B Hartsfield-Atlanta Intl	190763	25501	113

Days of the week with maximum delay in the year 2008.

```
%sql
select case when DayOfWeek = 1 then 'Monday'
when DayOfWeek = 2 then 'Tuesday'
when DayOfWeek = 3 then 'Wednesday'
when DayOfWeek = 4 then 'Thursday'
when DayOfWeek = 5 then 'Friday'
when DayOfWeek = 6 then 'Saturday'
when DayOfWeek = 7 then 'Sunday' end as Day
,sum(TotalDelay) as Total_Delay_Day_of_Week from
(select DayOfWeek,case when ArrDelay>DepDelay then ArrDelay else DepDelay end as TotalDelay from
data having TotalDelay>0) delay
group by Day order by Total_Delay_Day_of_Week desc;
```

Day	Total_Delay_Day_of_Week
Friday	18143013
Sunday	16442530
Monday	15928673
Thursday	15748971
Tuesday	15184541
Wednesday	14336835
Saturday	11906292

The day on which a particular carrier got delayed the most.

carrier_name
US Airways Inc. (Merged with America West 9/05. Reporting for both starting 10/07.)
Pinnacle Airlines Inc.
Aloha Airlines Inc.
Skywest Airlines Inc.
American Eagle Airlines Inc.
United Air Lines Inc.
Comair Inc.
Expressjet Airlines Inc.
Frontier Airlines Inc.
Southwest Airlines Co.

Continental Air Lines Inc.
Northwest Airlines Inc.
JetBlue Airways
AirTran Airways Corporation
Hawaiian Airlines Inc.
Atlantic Southeast Airlines
Alaska Airlines Inc.
Delta Air Lines Inc.
Mesa Airlines Inc.