Zeppelin

FINISHED

```
// Imports
import org.apache.spark.sql.functions._
import org.joda.time.format.DateTimeFormat

// Load data - adjust the path to the location of your data
val inputPath =  "/Users/mmohamar/Desktop/airlines"
val airTraffic = sqlContext.read
        .format("com.databricks.spark.csv")
        .option("header", "true") // Use first line of all files as header
        .option("delimiter", ",")
        .option("inferSchema", "true") // Automatically infer data types
        .load(inputPath)

// Add extra features
val calcDayOfYear = udf(
    (dayOfMonth: Int, month: Int, year: Int) => {
        val dateFormat = DateTimeFormat.forPattern("dd/MM/yyyy")
        dateFormat.parseDateTime(s"$dayOfMonth/$month/$year").getDayOfYear()
    }
)
val calcRoute = udf(
    (origin: String, dest: String) => s"$origin - $dest"
)
val calcHourOfArrival = udf(
    (arrTime: String) => arrTime.slice(0,arrTime.size-2)
)
val featuredTraffic = airTraffic
    .withColumn("DayOfYear", calcDayOfYear(airTraffic("DayOfMonth"), airTraffic("Month"), 
    .withColumn("HourOfArr", calcHourOfArrival(airTraffic("ArrTime")))
    .withColumn("Route", calcRoute(airTraffic("Origin"), airTraffic("Dest")))

// Register as Spark SQL Table
featuredTraffic.registerTempTable("air_traffic")
// sqlContext.cacheTable("air_traffic")
```

```
import org.apache.spark.sql.functions._
import org.joda.time.format.DateTimeFormat
inputPath: String = /Users/mmohamar/Desktop/airlines
airTraffic: org.apache.spark.sql.DataFrame = [Year: int, Month: int ... 27 more fields]
calcDayOfYear: org.apache.spark.sql.expressions.UserDefinedFunction = UserDefinedFunction(<
function3>,IntegerType,Some(List(IntegerType, IntegerType, IntegerType)))
calcRoute: org.apache.spark.sql.expressions.UserDefinedFunction = UserDefinedFunction(<func
tion2>,StringType,Some(List(StringType, StringType)))
calcHourOfArrival: org.apache.spark.sql.expressions.UserDefinedFunction = UserDefinedFuncti
on(<function1>,StringType,Some(List(StringType)))
featuredTraffic: org.apache.spark.sql.DataFrame = [Year: int, Month: int ... 30 more fields
]
warning: there was one deprecation warning; re-run with -deprecation for details
```

FINISHED

```
%sql
select DayOfYear, count(*) as NrOfFlights, avg(DepDelay) as AvgDepDelay, avg(ArrDelay) as 
```

DayOfYear	NrOfFlights	AvgDepDelay
148	382,896	6.27944
243	391,906	6.88937
31	393,094	7.28031
85	397,077	7.40247
137	397,458	8.57223
251	389,784	4.09237
65	400,274	8.86551
53	397,646	12.41472
255	395,408	5.00076

FINISHED

```
%sql
select HourOfArr, count(*) as NrFlights, avg(ArrDelay) as AvgArrDelay from air_traffic whe
(Friday)|6(Saturday)|7(Sunday)} and Origin = "${org=ATL,ATL|PHX|PIT}" group by HourOfArr
```

org

ATL

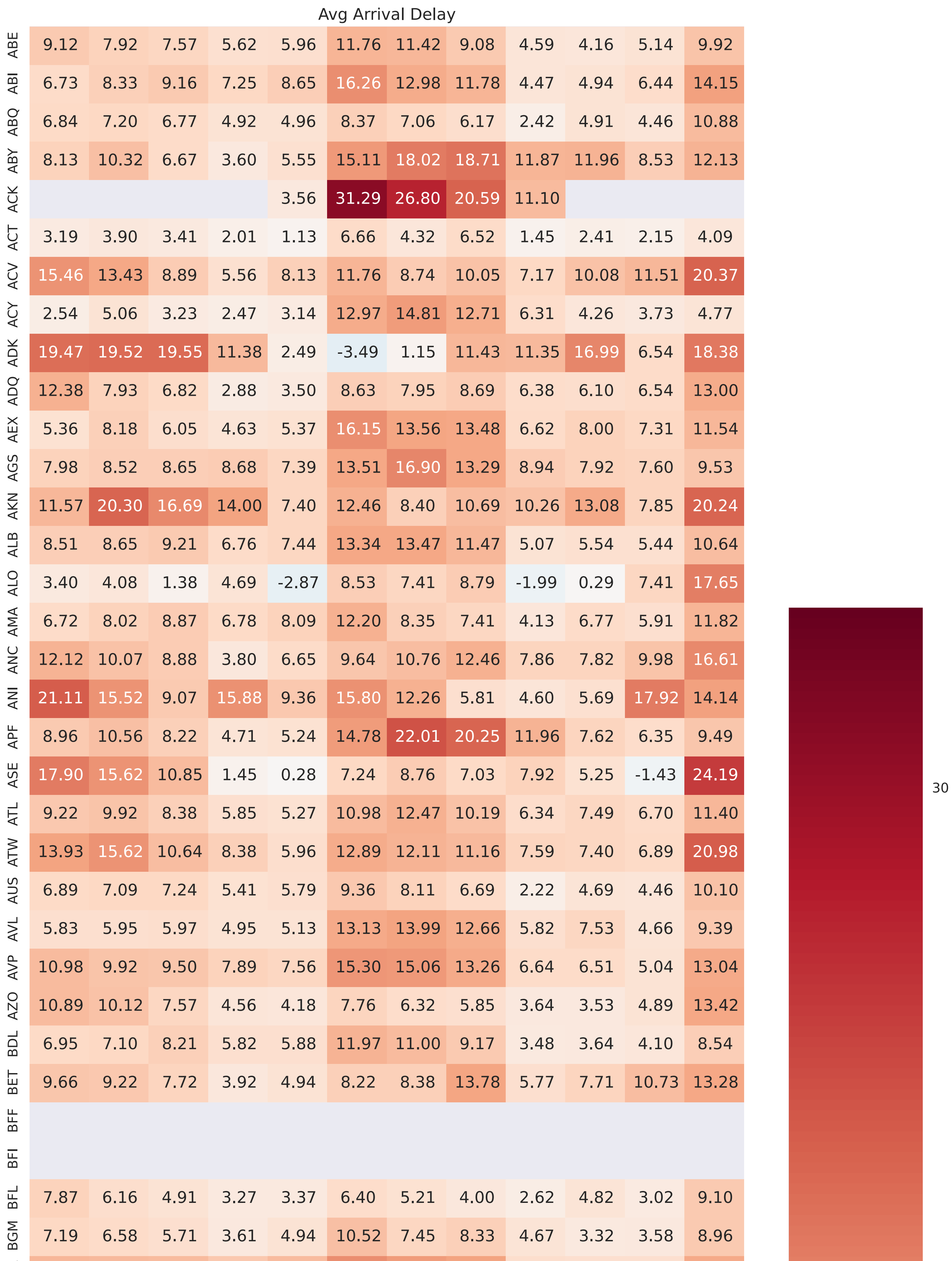
day

Monday

HourOfArr	NrFlights
7	13,386
15	61,916
11	74,132
3	274
8	34,363
22	67,786
28	3
16	66,777
5	15

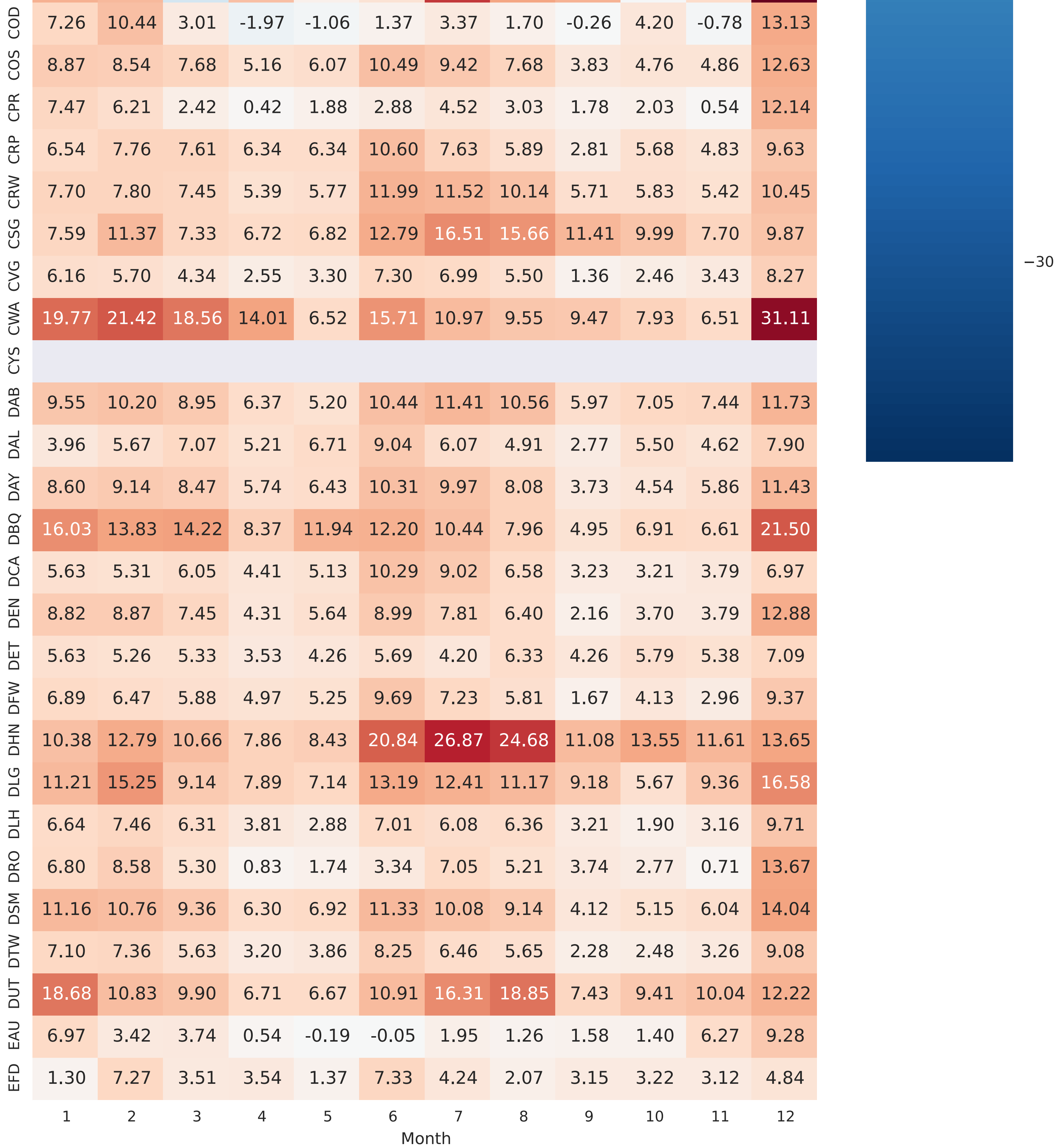
FINISHED

```
%pyspark
import matplotlib
import matplotlib.pyplot as plt
import seaborn as sns

import StringIO
def show(p):
  img = StringIO.StringIO()
  p.savefig(img, format='svg')
  img.seek(0)
  print "%html " + img.buf

df = sqlContext.sql("SELECT Dest, Month, count(*) as NrOfFlights, avg(ArrDelay) as AvgArrD
data = df.toPandas()

value = "AvgArrDelay"
x = "Dest"
grouping = ["Month"]

heatmap_data = data.pivot_table(values=value, index=x, columns=grouping)
heatmap_data = heatmap_data[0:100]

a4_dims = (len(heatmap_data.columns),50)
fig, ax = plt.subplots(figsize=a4_dims)
ax.set_title("Avg Arrival Delay")
sns.heatmap(heatmap_data, ax=ax, annot=True, fmt=".02f")

show(plt)
```



Avg Arrival Delay
ABE 9.12 7.92 7.57 5.62 5.96 11.76 11.42 9.08 4.59 4.16 5.14 9.92
ABI 6.73 8.33 9.16 7.25 8.65 16.26 12.98 11.78 4.47 4.94 6.44 14.15
ABQ 6.84 7.20 6.77 4.92 4.96 8.37 7.06 6.17 2.42 4.91 4.46 10.88
ABY 8.13 10.32 6.67 3.60 5.55 15.11 18.02 18.71 11.87 11.96 8.53 12.13
ACK 3.56 31.29 26.80 20.59 11.10
ACT 3.19 3.90 3.41 2.01 1.13 6.66 4.32 6.52 1.45 2.41 2.15 4.09
ACV 15.46 13.43 8.89 5.56 8.13 11.76 8.74 10.05 7.17 10.08 11.51 20.37
ACY 2.54 5.06 3.23 2.47 3.14 12.97 14.81 12.71 6.31 4.26 3.73 4.77
ADK 19.47 19.52 19.55 11.38 2.49 -3.49 1.15 11.43 11.35 16.99 6.54 18.38
ADQ 12.38 7.93 6.82 2.88 3.50 8.63 7.95 8.69 6.38 6.10 6.54 13.00
AEX 5.36 8.18 6.05 4.63 5.37 16.15 13.56 13.48 6.62 8.00 7.31 11.54
AGS 7.98 8.52 8.65 8.68 7.39 13.51 16.90 13.29 8.94 7.92 7.60 9.53
AKN 11.57 20.30 16.69 14.00 7.40 12.46 8.40 10.69 10.26 13.08 7.85 20.24
ALB 8.51 8.65 9.21 6.76 7.44 13.34 13.47 11.47 5.07 5.54 5.44 10.64
ALO 3.40 4.08 1.38 4.69 -2.87 8.53 7.41 8.79 -1.99 0.29 7.41 17.65
AMA 6.72 8.02 8.87 6.78 8.09 12.20 8.35 7.41 4.13 6.77 5.91 11.82
ANC 12.12 10.07 8.88 3.80 6.65 9.64 10.76 12.46 7.86 7.82 9.98 16.61
ANI 21.11 15.52 9.07 15.88 9.36 15.80 12.26 5.81 4.60 5.69 17.92 14.14
APF 8.96 10.56 8.22 4.71 5.24 14.78 22.01 20.25 11.96 7.62 6.35 9.49
ASE 17.90 15.62 10.85 1.45 0.28 7.24 8.76 7.03 7.92 5.25 -1.43 24.19
ATL 9.22 9.92 8.38 5.85 5.27 10.98 12.47 10.19 6.34 7.49 6.70 11.40
ATW 13.93 15.62 10.64 8.38 5.96 12.89 12.11 11.16 7.59 7.40 6.89 20.98
AUS 6.89 7.09 7.24 5.41 5.79 9.36 8.11 6.69 2.22 4.69 4.46 10.10
AVL 5.83 5.95 5.97 4.95 5.13 13.13 13.99 12.66 5.82 7.53 4.66 9.39
AVP 10.98 9.92 9.50 7.89 7.56 15.30 15.06 13.26 6.64 6.51 5.04 13.04
AZO 10.89 10.12 7.57 4.56 4.18 7.76 6.32 5.85 3.64 3.53 4.89 13.42
BDL 6.95 7.10 8.21 5.82 5.88 11.97 11.00 9.17 3.48 3.64 4.10 8.54
BET 9.66 9.22 7.72 3.92 4.94 8.22 8.38 13.78 5.77 7.71 10.73 13.28
BFF
BFI
BFL 7.87 6.16 4.91 3.27 3.37 6.40 5.21 4.00 2.62 4.82 3.02 9.10
BGM 7.19 6.58 5.71 3.61 4.94 10.52 7.45 8.33 4.67 3.32 3.58 8.96
30

Dest												
BGR	11.55	11.33	10.97	9.23	10.91	16.09	14.87	13.86	6.11	5.78	5.82	13.09
BHM	6.79	6.98	6.80	5.08	5.96	10.17	9.63	7.08	3.11	4.53	4.63	9.77
BIL	8.38	7.44	5.87	2.72	3.04	7.16	7.89	6.19	2.37	3.29	3.94	13.39
BIS	6.83	7.35	6.70	2.11	2.64	7.81	4.78	4.19	1.51	1.76	3.69	12.33
BJI						2.80	0.91	4.14	9.46	2.00	-7.00	
BLI	6.64	4.44	2.68	0.63	1.35	3.18	2.01	6.33	3.37	6.04	7.95	14.50
BMI	15.69	18.62	15.50	9.25	7.21	16.10	16.52	13.91	9.59	9.43	10.62	19.38
BNA	5.66	5.87	5.41	3.89	4.40	8.74	7.91	6.09	2.03	3.51	3.47	8.23
BOI	10.84	9.53	7.64	4.46	4.62	8.29	7.95	8.17	3.59	5.35	5.94	15.79
BOS	8.21	8.31	9.24	7.83	8.23	13.78	12.55	10.81	5.31	5.46	5.96	9.26
BPT	1.63	3.33	2.67	1.67	1.49	12.09	5.63	4.25	0.59	0.86	4.58	1.94
BQK	5.32	10.13	10.45	5.83	9.45	15.97	21.09	21.18	13.79	15.16	8.41	13.75
BQN	7.57	10.62	11.46	9.82	8.94	20.49	22.64	14.09	4.89	1.46	6.44	13.68
BRO	5.79	6.99	8.83	4.24	5.80	11.72	8.40	4.62	-1.70	5.90	5.74	11.27
BRW	7.63	9.81	7.93	2.49	6.86	8.97	12.78	14.33	5.73	4.87	5.65	11.68
BTM	9.27	5.37	3.85	0.45	-0.20	3.52	2.40	2.24	0.16	0.74	2.76	15.93
BTR	7.54	9.25	8.76	7.40	6.71	12.31	10.92	8.74	4.41	5.14	6.00	10.59
BTV	9.29	10.49	12.36	8.03	8.32	14.15	16.67	14.23	5.05	5.18	5.36	13.47
BUF	9.23	9.83	9.00	7.51	6.58	12.76	13.06	11.02	4.86	5.34	5.74	11.86
BUR	7.53	8.09	6.07	5.09	4.00	6.59	5.82	6.66	3.38	5.64	6.05	10.55
BWI	6.04	6.00	6.64	4.60	5.65	11.99	11.48	8.60	2.70	3.14	4.18	7.84
BZN	12.67	10.70	6.71	1.14	2.11	6.82	6.14	5.57	1.92	1.67	3.14	16.35
CAE	10.80	11.18	10.16	8.52	7.87	14.07	14.04	12.17	6.96	6.76	7.63	13.55
CAK	9.11	11.29	7.96	5.56	5.20	11.73	12.74	10.85	4.86	5.81	6.18	13.12
CBM				0.00								
CCR	2.64	1.13	1.95	-1.32	0.49	2.03	0.75	5.94	2.77	7.50	7.05	7.49
CDC	4.43	6.32	2.47	-0.32	-1.09	0.93	2.67	0.73	0.75	-0.65	1.71	8.18
CDV	12.36	9.59	7.51	6.86	9.06	13.56	13.62	19.88	14.74	8.66	7.60	17.65
CEC	15.45	10.64	7.92	4.30	5.11	10.79	8.25	11.41	6.75	9.70	14.71	18.65
CHA	7.11	7.14	6.16	4.97	5.86	11.21	11.50	8.78	5.40	5.00	4.90	9.90
CHO	6.44	5.59	3.70	3.48	3.26	9.01	11.58	9.88	6.57	5.99	5.69	5.44
CHS	8.09	9.09	9.28	7.28	6.80	11.45	12.02	9.18	5.05	6.01	5.89	10.96
CIC	17.60	15.55	9.00	9.13	11.07	15.34	14.39	10.37	11.17	13.82	16.89	20.26
CID	11.88	10.44	9.56	6.51	7.23	11.34	9.82	8.51	3.50	3.74	4.80	13.96
CKB	7.83											
CLD	3.90	3.80	3.42	1.76	2.17	2.92	2.72	1.56	2.90	3.51	1.20	2.29
CLE	7.61	7.52	7.05	4.79	5.40	10.45	9.26	7.69	2.72	3.75	4.29	9.91
CLL	3.63	5.06	3.26	1.81	3.19	9.26	5.83	7.61	-0.51	1.92	2.24	5.10
CLT	5.91	5.83	5.94	4.38	4.01	8.58	8.19	6.55	2.27	2.89	3.64	7.36
CMH	9.48	8.71	8.65	6.67	7.25	12.20	11.46	9.59	4.39	4.98	5.92	11.67
CMI	19.47	15.82	15.09	10.26	11.47	15.38	15.60	14.51	7.02	8.42	10.03	24.01
CMX	12.46	11.20	-6.45	10.49	1.79	5.06	24.55	13.74	12.11	-0.40	6.33	34.69



FINISHED

```
%pyspark
```

```
from wordcloud import WordCloud

import StringIO
def show(p):
  img = StringIO.StringIO()
  p.savefig(img, format='svg')
  img.seek(0)
  print "%html " + img.buf

# Create route frequencies
routes = sqlContext.sql("SELECT Route, count(*) as Count FROM air_traffic GROUP BY Route")
route_freq = [(x[0],x[1]) for x in routes]

# Generate word cloud image
wordcloud = WordCloud().generate_from_frequencies(route_freq)
image = wordcloud.to_image()
```

FINISHED

```
%pyspark
import matplotlib.pyplot as plt

#define some data
x = [1,2,3,4]
y = [20, 21, 20.5, 20.8]

#plot data
plt.plot(x, y, linestyle="dashed", marker="o", color="green")
```

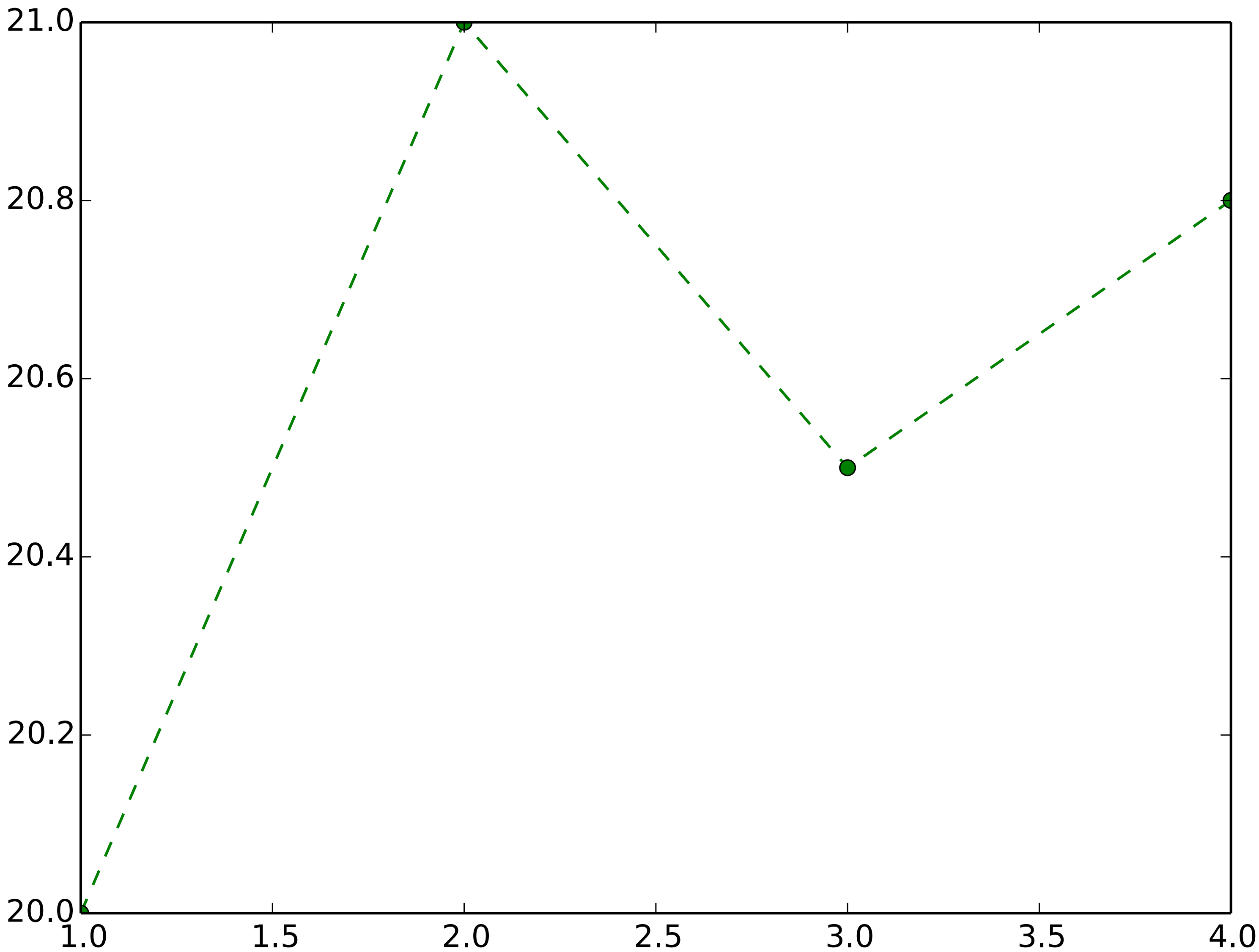
```
[<matplotlib.lines.Line2D object at 0x111558c50>]
```

FINISHED

```
%pyspark
# helper function to display in Zeppelin

import StringIO
def show(p):
  img = StringIO.StringIO()
  p.savefig(img, format='svg')
  img.seek(0)
  print "%html " + img.buf
```

FINISHED

```
%pyspark
show(plt)
```



READY