CHALLENGES AND BUSINESS RECOMMENDATIONS IN AVIATION INDUSTRY

TRENT UNIVERSITY

Praveen Kumar Marichamy, SaiKarthik Kandikonda

Big-Data Analytics

CHALLENGES AND BUSINESS RECOMMENDATIONS IN AVIATION INDUSTRY

Praveen Kumar Marichamy (0663980)
Saikarthik Kandikonda (0671359)

TRENT UNIVERSITY

Introduction:

In this fast and ever-changing world, everyone knows the value of time and money moreover people started travelling from one place to another for various reasons like business, leisure, sports etc., In this case analysing the past airline services data became in-evitable for most of the companies to give better services for the people in terms of no delay in operating the airlines, providing best price for all the various class passengers and overall satisfaction of the customers as well. Lot of countries trying to build their airports as a hub for transits to attract people for tourism also. In bigger picture it indeed helps to boost the country's economy as well. Countries such as Singapore and Dubai are the most important players in terms of airline industry and tourism related services.

Motivation for Choosing the Project:

Myself (Praveen) worked in a logistics software-based domain as a performance and test engineering analyst where I dealt with clients like Virgin Atlantic and KLM in Airline Industry as well as Maersk in shipping Industry. Karthik has worked as a data analyst and Business intelligence strategist for Changi Singapore airport. His project was based on analyzing the customer satisfaction index, overall rating and comparing the performances across all the employees working in the airport.

Both were pretty familiar with the airline industry domain and we know it's one of the un-avoidable and profitable industries in the world. According to the current scenario we can even split the business into two categories post covid and pre covid. We are expecting many business outcomes like merger of small airlines, fewer number of travelers across the world, flight price changes etc.,

Research problem and Dataset Details:

Internet provides us a lot of datasets for airline domain either structured or semi-structured. We have chosen Airline on time performance dataset from ***http://stat-computing.org/dataexpo/2009/*** a collection of all the logs of domestic flights from the period of October 1987 to April 2008 in the United States of America. The records represent the individual flights where various flight related details such as the time and the date of arrival were mentioned. Moreover, the time taken for the taxi to runway were provided in additional.

We are going to address some of the inferences from the dataset as well as the recommendations for the passengers to choose the flights wisely.

1. What would be the best day of the week the day the passenger can choose to fly so that he/she can avoid delays in their journeys?
2. Predicting the top 20 busiest routes in USA and the number of flights travelled in a year
3. To predict the peak hours of the particular airport which will help the individual to book the flight on non-peak hours of the day.
4. Predicting the best Airline one can choose between the 2 Places based on the past data. This will help the passenger to get rid of delays and last-minute flight cancellations.

CHALLENGES AND BUSINESS RECOMMENDATIONS IN AVIATION INDUSTRY

Praveen Kumar Marichamy (0663980)
Saikarthik Kandikonda (0671359)

List of Activities

1. Data Gathering
2. Defining Problem Statement
3. Architecture and Data Pipeline
4. Establishing connections across various services
5. Loading the data and designing the schema
6. Performing Analysis
7. Plotting appropriate graphs
8. Storing the data back to No-Sql DB from Spark
9. Report and PPT

Activities	***Contributors***
Data Gathering	SaiKarthik
Defining Problem Statement	Praveen
Architecture and Data Pipeline	Praveen, SaiKarthik
Establishing connections across various services	Praveen, SaiKarthik
Loading the data and designing the schema	Praveen, SaiKarthik
Performing Analysis	Praveen(2 ProblemStatements) Karthik (2 Problem Statements)
Plotting appropriate graphs	Praveen(2 ProblemStatements) Karthik (2 Problem Statements)
Storing the data back to No-Sql DB from Spark	SaiKarthik
Report and PPT	Praveen

CHALLENGES AND BUSINESS RECOMMENDATIONS IN AVIATION INDUSTRY

Praveen Kumar Marichamy (0663980)
Saikarthik Kandikonda (0671359)

Approach Overview:

- As the data being big enough and semi-structured, we choose the MongoDB as the source for storing the output from the PySpark data frame.
- The initial approach being the data from the data source (***dataexpo 2009***) is sent to the spark RDD container using pyspark using ipython Jupyter notebook.
- Data being analyzed by using spark sqlcontext due to the variety of operations can be carried out inside the data.
- Once the data was cleaned and transformed, we have sent the data in to the MongoDB to store and can view the result for the future.

**Firstly, this approach was carried out in the Horton works hdp using oracle virtual machine but due to the limitations of RAM and ROM occupied during the process made us to install the software's in our local system and interconnected them accordingly.*

Architecture Diagram for Airline Project:

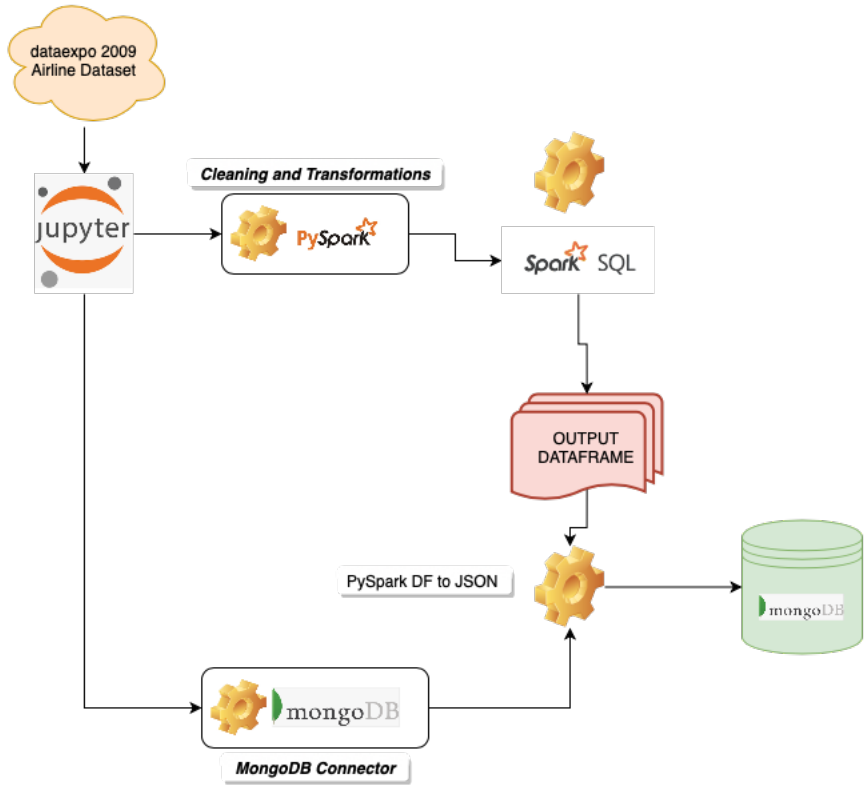


Diagram has drawn using **drawio architecture diagram software*

Praveen Kumar Marichamy (0663980)
Saikarthik Kandikonda (0671359)

Pyspark:

Spark is widely used as a computing engine after MapReduce framework. The main advantage of using the spark it supports stream processing whereas Hadoop supports batch processing. The default language of spark is Scala and in our project, we have installed pyspark on top of spark for programming.

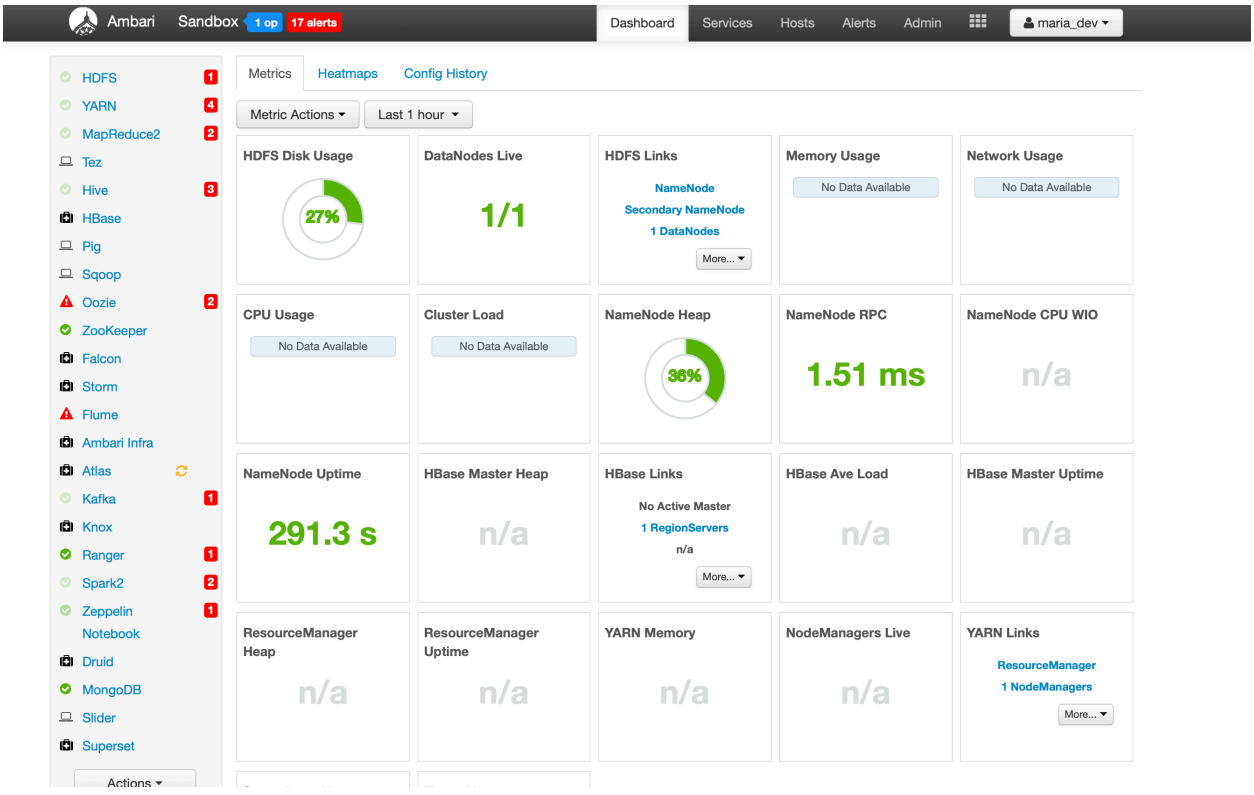
1. Pyspark is a great language for performing exploratory data analysis and building machine learning pipelines too.
2. Easy to integrate and work with RDD's in python programming language.

PySpark SQL Context:

SQL Context is used to enable and run applications using SQL queries and running SQL functions and returns all the outputs/ results as a data frame.

1. Spark SQL is a spark module for structured data processing and also acts as a distributed SQL query engine.

Spark and Mongo Installed in HDP:

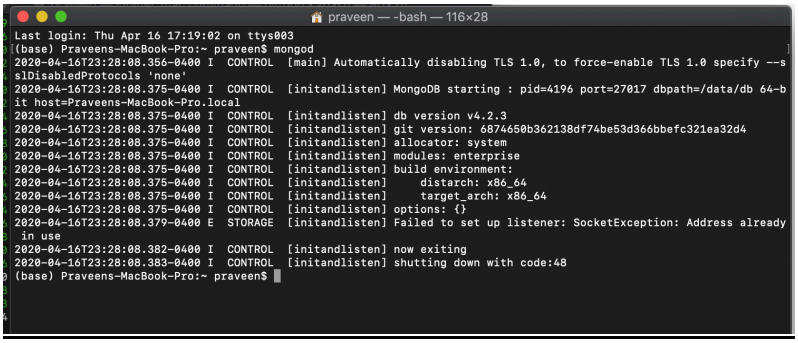


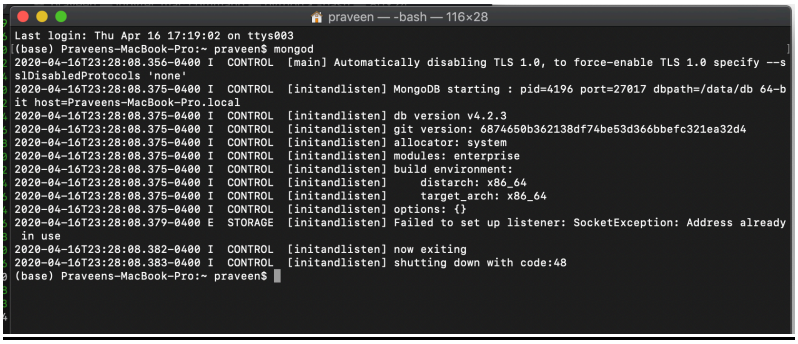
**As mentioned, due to the limitations we switched back to our local system and software's for analyzing the data*

CHALLENGES AND BUSINESS RECOMMENDATIONS IN AVIATION INDUSTRY

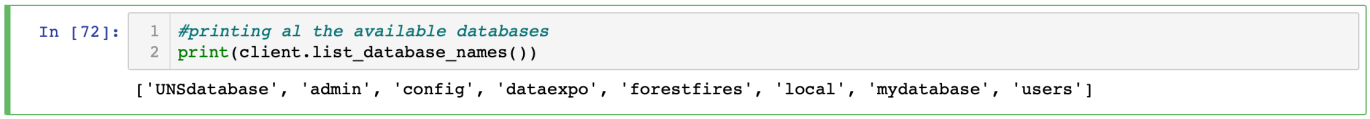
Praveen Kumar Marichamy (0663980)
Saikarthik Kandikonda (0671359)

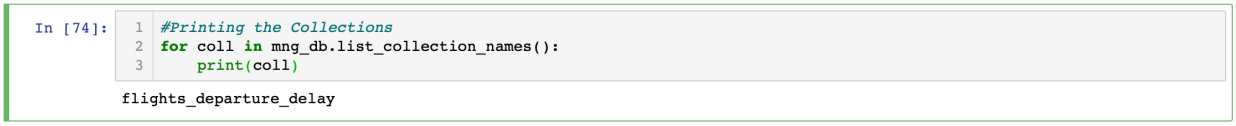
MongoDB and Server Screenshot:





MongoDB Connector in Ipython Notebook:



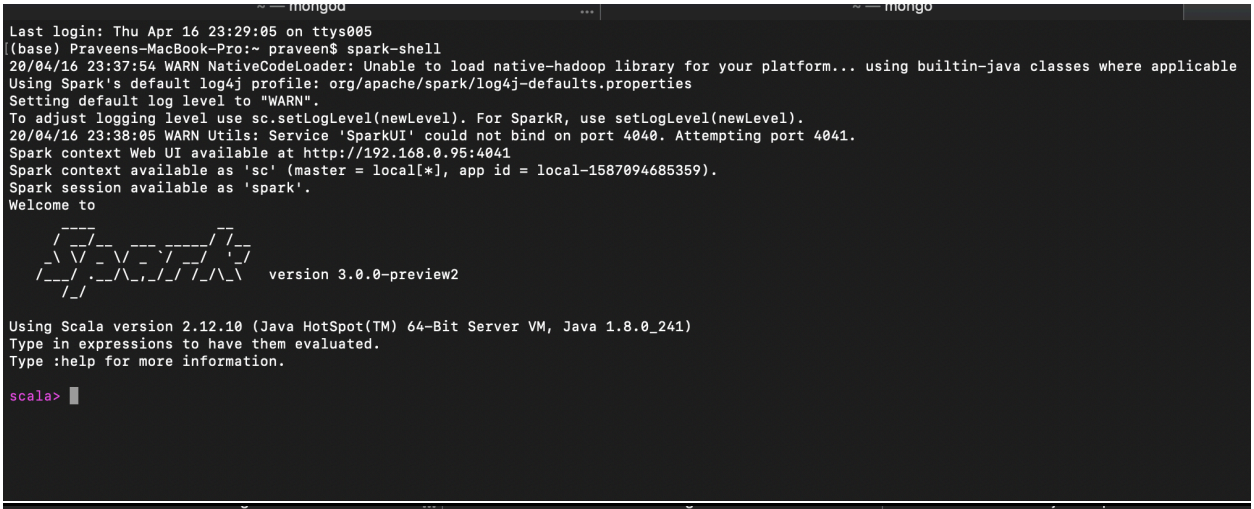


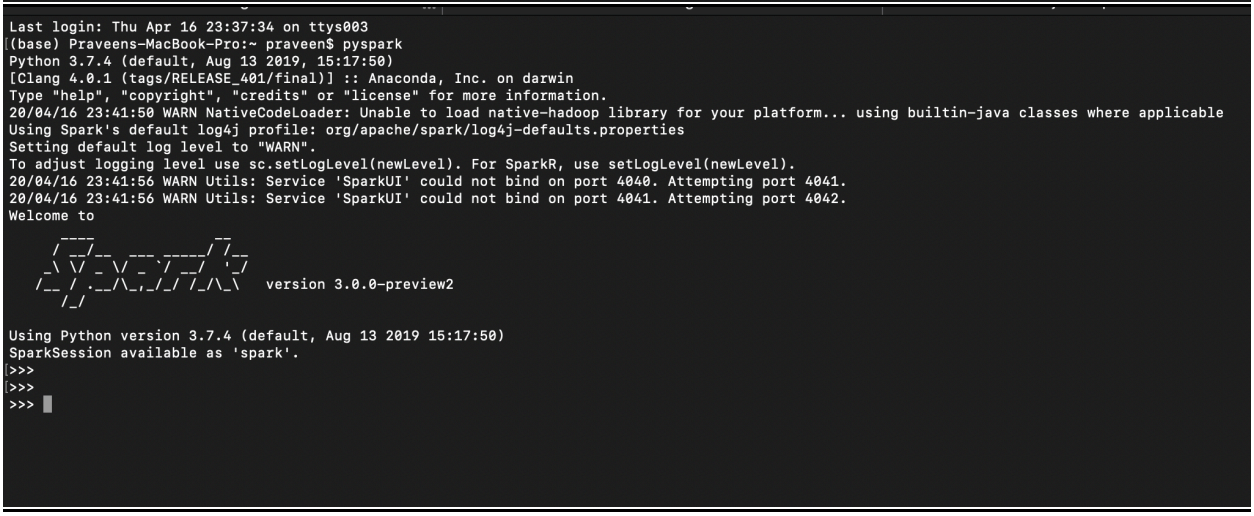
- Our Database dataexpo was created successfully.
- Our Database collection flights_departure_delay was successfully created.

CHALLENGES AND BUSINESS RECOMMENDATIONS IN AVIATION INDUSTRY

Praveen Kumar Marichamy (0663980)
Saikarthik Kandikonda (0671359)

Spark Installation and Pyspark Shell:





Spark, Pyspark Connector and SQLContext in Ipython Notebook:

CHALLENGES AND BUSINESS RECOMMENDATIONS IN AVIATION INDUSTRY

Praveen Kumar Marichamy (0663980)
Saikarthik Kandikonda (0671359)

In [2]:
```
from pyspark.sql import SQLContext
from pyspark import Row
from pyspark.context import SparkContext
from pyspark.sql.session import SparkSession
from pyspark.sql import Row
```

In [4]:
```
import pyspark
#sc = pyspark.SparkContext('local[*]')
txt = sc.textFile('/Users/praveen/Big-Data/Project/flights.csv')
sqlContext= SQLContext(sc)
```

In [5]:
```
df = sqlContext.read.format('com.databricks.spark.csv').options(header='true').load('/Users/praveen/Big-Data/Projec
```

In [6]:
```
df = sqlContext.read.load('/Users/praveen/Big-Data/Project/flights.csv',
                          format='com.databricks.spark.csv',
                          header='true',
                          inferSchema='true')
```

Business Recommendations and Inferences from the Dataset:

1. *Sample head of airline dataset with columns and rows printed.*

In [8]:
```
df.createOrReplaceTempView("airline")

sqlDF = sqlContext.sql("SELECT * FROM airline limit 1")
sqlDF.show()
```

```
+----+-----+---+-----------+-------+-------------+-----------+--------------+-------------------+--------------------+
--------------+---------------+--------+----------+--------------+------------+--------+--------+---------+-------+--
--------------+------------+-------------+--------+---------+-------------------+---------------+--------------+---
----------+-------------------+--------------+
|YEAR|MONTH|DAY|DAY_OF_WEEK|AIRLINE|FLIGHT_NUMBER|TAIL_NUMBER|ORIGIN_AIRPORT|DESTINATION_AIRPORT|SCHEDULED_DEPARTURE|
DEPARTURE_TIME|DEPARTURE_DELAY|TAXI_OUT|WHEELS_OFF|SCHEDULED_TIME|ELAPSED_TIME|AIR_TIME|DISTANCE|WHEELS_ON|TAXI_IN|SC
HEDULED_ARRIVAL|ARRIVAL_TIME|ARRIVAL_DELAY|DIVERTED|CANCELLED|CANCELLATION_REASON|AIR_SYSTEM_DELAY|SECURITY_DELAY|AIR
LINE_DELAY|LATE_AIRCRAFT_DELAY|WEATHER_DELAY|
+----+-----+---+-----------+-------+-------------+-----------+--------------+-------------------+--------------------+
--------------+---------------+--------+----------+--------------+------------+--------+--------+---------+-------+--
--------------+------------+-------------+--------+---------+-------------------+---------------+--------------+---
----------+-------------------+--------------+
|2015|    1|  1|          4|     AS|           98|     N407AS|           ANC|                SEA|                   5|
2354|           -11|      21|        15|           205|         194|     169|    1448|      404|      4|
430|          408|          -22|       0|        0|               null|            null|          null|          null|
null|          null|
+----+-----+---+-----------+-------+-------------+-----------+--------------+-------------------+--------------------+
--------------+---------------+--------+----------+--------------+------------+--------+--------+---------+-------+--
--------------+------------+-------------+--------+---------+-------------------+---------------+--------------+---
----------+-------------------+--------------+
```

Next we tried to analyze the data types of all the variables in the dataset

Praveen Kumar Marichamy (0663980)
Saikarthik Kandikonda (0671359)

```
df.dtypes
```

```
[('YEAR', 'int'),
 ('MONTH', 'int'),
 ('DAY', 'int'),
 ('DAY_OF_WEEK', 'int'),
 ('AIRLINE', 'string'),
 ('FLIGHT_NUMBER', 'int'),
 ('TAIL_NUMBER', 'string'),
 ('ORIGIN_AIRPORT', 'string'),
 ('DESTINATION_AIRPORT', 'string'),
 ('SCHEDULED_DEPARTURE', 'int'),
 ('DEPARTURE_TIME', 'int'),
 ('DEPARTURE_DELAY', 'int'),
 ('TAXI_OUT', 'int'),
 ('WHEELS_OFF', 'int'),
 ('SCHEDULED_TIME', 'int'),
 ('ELAPSED_TIME', 'int'),
 ('AIR_TIME', 'int'),
 ('DISTANCE', 'int'),
 ('WHEELS_ON', 'int'),
 ('TAXI_IN', 'int'),
 ('SCHEDULED_ARRIVAL', 'int'),
 ('ARRIVAL_TIME', 'int'),
 ('ARRIVAL_DELAY', 'int'),
 ('DIVERTED', 'int'),
 ('CANCELLED', 'int'),
 ('CANCELLATION_REASON', 'string'),
 ('AIR_SYSTEM_DELAY', 'int'),
 ('SECURITY_DELAY', 'int'),
 ('AIRLINE_DELAY', 'int'),
 ('LATE_AIRCRAFT_DELAY', 'int'),
 ('WEATHER_DELAY', 'int')]
```

We need to import pyspark sql package in order to perform operations in PYSPARK SQL.

```
from pyspark.sql.functions import *
```

Created a new Pyspark data frame with all the required columns so that we can reduce the unnecessary variables.

```
df_flight = df.select(df['YEAR'], df['MONTH'],df['DAY'] ,df['DAY_OF_WEEK'] , df['AIRLINE'] ,df['ORIGIN_AIRPORT'],df['DESTINATION
df_flight.show(15, truncate=False)
```

```
+----+-----+---+-----------+-------+--------------+-------------------+-------------------+--------------+---------------+-----
-------+-------------+---------+
|YEAR|MONTH|DAY|DAY_OF_WEEK|AIRLINE|ORIGIN_AIRPORT|DESTINATION_AIRPORT|SCHEDULED_DEPARTURE|DEPARTURE_TIME|DEPARTURE_DELAY|ARRIV
AL_TIME|ARRIVAL_DELAY|CANCELLED|
+----+-----+---+-----------+-------+--------------+-------------------+-------------------+--------------+---------------+-----
-------+-------------+---------+
|2015|1    |1  |4          |AS     |ANC           |SEA                |5                  |2354          |-11            |408
|-22         |0        |
|2015|1    |1  |4          |AA     |LAX           |PBI                |10                 |2             |-8             |741
|-9          |0        |
|2015|1    |1  |4          |US     |SFO           |CLT                |20                 |18            |-2             |811
|5           |0        |
|2015|1    |1  |4          |AA     |LAX           |MIA                |20                 |15            |-5             |756
|-9          |0        |
|2015|1    |1  |4          |AS     |SEA           |ANC                |25                 |24            |-1             |259
|-21         |0        |
|2015|1    |1  |4          |DL     |SFO           |MSP                |25                 |20            |-5             |610
|8           |0        |
|2015|1    |1  |4          |NK     |LAS           |MSP                |25                 |19            |-6             |509
|-17         |0        |
|2015|1    |1  |4          |US     |LAX           |CLT                |30                 |44            |14             |753
|-10         |0        |
|2015|1    |1  |4          |AA     |SFO           |DFW                |30                 |19            |-11            |532
|-13         |0        |
|2015|1    |1  |4          |DL     |LAS           |ATL                |30                 |33            |3              |656
|-15         |0        |
|2015|1    |1  |4          |DL     |DEN           |ATL                |30                 |24            |-6             |453
```

Praveen Kumar Marichamy (0663980)
Saikarthik Kandikonda (0671359)

In order to perform Spark Sql queries we need a Table upon which we can perform analysis, data frames can not be used to perform analysis directly.so we have created a temporary view upon which we can perform analysis.

```
df_flight.createOrReplaceTempView("flight_1")

sqlDF = sqlContext.sql("SELECT * FROM flight_1 limit 1")
sqlDF.show()
```

Analysis 1:

What would be the best day of the week the day the passenger can choose to fly so that he/she can avoid delays in their journeys?

```
# This is used to analyse the average delay time for each day of the week
sqlDF1 = sqlContext.sql("SELECT DAY_OF_WEEK ,avg(DEPARTURE_DELAY) FROM flight_1 WHERE DEPARTURE_DELAY > 0
                         GROUP BY DAY_OF_WEEK ORDER BY DAY_OF_WEEK   ")
sqlDF1.show()
```

```
+-----------+--------------------+
|DAY_OF_WEEK|avg(DEPARTURE_DELAY)|
+-----------+--------------------+
|          1|   35.29113493909654|
|          2|   33.79152860987517|
|          3|  32.064168899565736|
|          4|   32.36675169788299|
|          5|  31.244123389063574|
|          6|  31.288367895222905|
|          7|   32.36801816735181|
+-----------+--------------------+
```

As shown in the above image the output data is not in the much presentable way, we can not plots graphs directly on the spark data frame, in order to overcome this we have converted it to a pandas data frame as shown in the figure below.

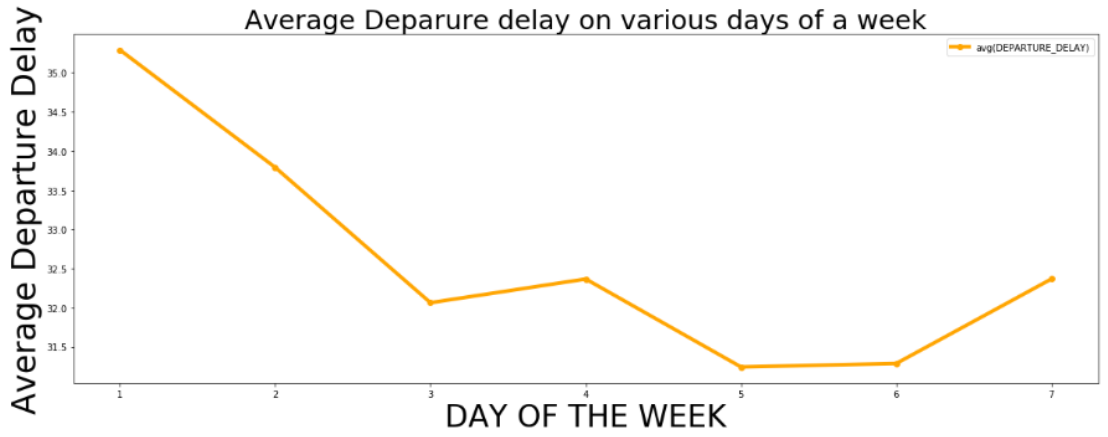
Praveen Kumar Marichamy (0663980)
Saikarthik Kandikonda (0671359)

```
Day_of_the_week = sqlDF1.toPandas()
Day_of_the_week.reset_index(drop=True, inplace=True)
Day_of_the_week
```

	DAY_OF_WEEK	avg(DEPARTURE_DELAY)
0	1	35.291135
1	2	33.791529
2	3	32.064169
3	4	32.366752
4	5	31.244123
5	6	31.288368
6	7	32.368018

Graph:

In [116]:
```
plt.plot( 'DAY_OF_WEEK', 'avg(DEPARTURE_DELAY)', data=Day_of_the_week, marker='o', markerfacecolor='orange',
         color='orange', linewidth=4)
plt.legend()
plt.xlabel("DAY OF THE WEEK", fontsize = 33)
plt.ylabel("Average Departure Delay" , fontsize =35)
plt.title("Average Deparure delay on various days of a week" , fontsize =29)
plt.show()
```



Praveen Kumar Marichamy (0663980)
Saikarthik Kandikonda (0671359)

Inference: *From the above graph we can clearly say that the average flight delays are more on day 1, 2 of the week.*

Inserting the data into MongoDB:

In [73]:
```
import pymongo
mng_client = pymongo.MongoClient('localhost', 27017)
mng_db = mng_client['dataexpo']
collection_name = 'flights_departure_delay'
db_cm = mng_db[collection_name]

#Insert Data
db_cm.insert(data)
```

...

```
MongoDB Enterprise > db.flights_departure_delay.find()
{ "_id" : ObjectId("5e98d710f8a287a285ebf6cb"), "DAY_OF_WEEK" : { "0" : 5, "1" : 6, "2" : 3, "3" : 4, "4" : 7, "5" : 2, "6" : 1 }, "avg(DEPARTURE_DELAY)" : { "0" : 31.2441233891, "1" : 31.2883678952, "2" : 32.8641688996, "3" : 32.3667516979, "4" : 32.3688181674, "5" : 33.7915286099, "6" : 35.2911349391 } }
```

Analysis 2:

Analysing the most busiest routes in USA

Praveen Kumar Marichamy (0663980)
Saikarthik Kandikonda (0671359)

In [43]:
```
# This is used to analyse the top 20 busiest routes in USA and the number of flights travelled in a Year
sqlDF4 = sqlContext.sql("SELECT  ORIGIN_AIRPORT , DESTINATION_AIRPORT , count(*)
                         FROM flight_1 GROUP BY ORIGIN_AIRPORT ,DESTINATION_AIRPORT ORDER BY count(*) DESC  ")
sqlDF4.show()
```

```
+--------------+-------------------+--------+
|ORIGIN_AIRPORT|DESTINATION_AIRPORT|count(1)|
+--------------+-------------------+--------+
|           SFO|                LAX|   13744|
|           LAX|                SFO|   13457|
|           JFK|                LAX|   12016|
|           LAX|                JFK|   12015|
|           LAS|                LAX|    9715|
|           LGA|                ORD|    9639|
|           LAX|                LAS|    9594|
|           ORD|                LGA|    9575|
|           SFO|                JFK|    8440|
|           JFK|                SFO|    8437|
|           OGG|                HNL|    8313|
|           HNL|                OGG|    8282|
|           LAX|                ORD|    8256|
|           ATL|                LGA|    8234|
|           LGA|                ATL|    8215|
|           ATL|                MCO|    8202|
|           MCO|                ATL|    8202|
|           SFO|                LAS|    7995|
|           ORD|                LAX|    7941|
|           LAS|                SFO|    7870|
+--------------+-------------------+--------+
only showing top 20 rows
```

Inference: *From the above grid we can capture the busiest routes and this will help us to arrange more number of flights in case required.*

MongoDB Insertion:

```
MongoDB Enterprise > db.busyroutes.find()
{ "_id" : ObjectId("5e9a3d93b54c749f3b5a4817"), "ORIGIN_AIRPORT" : "SFO", "DESTINATION_AIRPORT" : "LAX", "count(1)" : 13744 }
{ "_id" : ObjectId("5e9a3d93b54c749f3b5a4818"), "ORIGIN_AIRPORT" : "LAX", "DESTINATION_AIRPORT" : "SFO", "count(1)" : 13457 }
{ "_id" : ObjectId("5e9a3d93b54c749f3b5a4819"), "ORIGIN_AIRPORT" : "JFK", "DESTINATION_AIRPORT" : "LAX", "count(1)" : 12016 }
{ "_id" : ObjectId("5e9a3d93b54c749f3b5a481a"), "ORIGIN_AIRPORT" : "LAX", "DESTINATION_AIRPORT" : "JFK", "count(1)" : 12015 }
{ "_id" : ObjectId("5e9a3d93b54c749f3b5a481b"), "ORIGIN_AIRPORT" : "LAS", "DESTINATION_AIRPORT" : "LAX", "count(1)" : 9715 }
{ "_id" : ObjectId("5e9a3d93b54c749f3b5a481c"), "ORIGIN_AIRPORT" : "LGA", "DESTINATION_AIRPORT" : "ORD", "count(1)" : 9639 }
{ "_id" : ObjectId("5e9a3d93b54c749f3b5a481d"), "ORIGIN_AIRPORT" : "LAX", "DESTINATION_AIRPORT" : "LAS", "count(1)" : 9594 }
{ "_id" : ObjectId("5e9a3d93b54c749f3b5a481e"), "ORIGIN_AIRPORT" : "ORD", "DESTINATION_AIRPORT" : "LGA", "count(1)" : 9575 }
{ "_id" : ObjectId("5e9a3d93b54c749f3b5a481f"), "ORIGIN_AIRPORT" : "SFO", "DESTINATION_AIRPORT" : "JFK", "count(1)" : 8440 }
{ "_id" : ObjectId("5e9a3d93b54c749f3b5a4820"), "ORIGIN_AIRPORT" : "JFK", "DESTINATION_AIRPORT" : "SFO", "count(1)" : 8437 }
{ "_id" : ObjectId("5e9a3d93b54c749f3b5a4821"), "ORIGIN_AIRPORT" : "OGG", "DESTINATION_AIRPORT" : "HNL", "count(1)" : 8313 }
{ "_id" : ObjectId("5e9a3d93b54c749f3b5a4822"), "ORIGIN_AIRPORT" : "HNL", "DESTINATION_AIRPORT" : "OGG", "count(1)" : 8282 }
{ "_id" : ObjectId("5e9a3d93b54c749f3b5a4823"), "ORIGIN_AIRPORT" : "LAX", "DESTINATION_AIRPORT" : "ORD", "count(1)" : 8256 }
{ "_id" : ObjectId("5e9a3d93b54c749f3b5a4824"), "ORIGIN_AIRPORT" : "ATL", "DESTINATION_AIRPORT" : "LGA", "count(1)" : 8234 }
{ "_id" : ObjectId("5e9a3d93b54c749f3b5a4825"), "ORIGIN_AIRPORT" : "LGA", "DESTINATION_AIRPORT" : "ATL", "count(1)" : 8215 }
{ "_id" : ObjectId("5e9a3d93b54c749f3b5a4826"), "ORIGIN_AIRPORT" : "ATL", "DESTINATION_AIRPORT" : "MCO", "count(1)" : 8202 }
{ "_id" : ObjectId("5e9a3d93b54c749f3b5a4827"), "ORIGIN_AIRPORT" : "MCO", "DESTINATION_AIRPORT" : "ATL", "count(1)" : 8202 }
{ "_id" : ObjectId("5e9a3d93b54c749f3b5a4828"), "ORIGIN_AIRPORT" : "SFO", "DESTINATION_AIRPORT" : "LAS", "count(1)" : 7995 }
{ "_id" : ObjectId("5e9a3d93b54c749f3b5a4829"), "ORIGIN_AIRPORT" : "ORD", "DESTINATION_AIRPORT" : "LAX", "count(1)" : 7941 }
{ "_id" : ObjectId("5e9a3d93b54c749f3b5a482a"), "ORIGIN_AIRPORT" : "DFW", "DESTINATION_AIRPORT" : "ORD", "count(1)" : 7870 }
```

Praveen Kumar Marichamy (0663980)
Saikarthik Kandikonda (0671359)

```
In [54]:
from bson import json_util
data = json_util.loads(BUSY_ROUTES_Json)
```

```
In [55]:
import pymongo
mng_client = pymongo.MongoClient('localhost', 27017)
mng_db = mng_client['dataexpo']
collection_name = 'busyroutes'
db_cm = mng_db[collection_name]

#Insert Data
db_cm.insert(data)
```

```
/opt/anaconda3/lib/python3.7/site-packages/ipykernel_launcher.py:8: DeprecationWarning: insert is deprecated. Use ins
ert_one or insert_many instead.
```

```
Out[55]: [ObjectId('5e9a3d93b54c749f3b5a4817'),
 ObjectId('5e9a3d93b54c749f3b5a4818'),
 ObjectId('5e9a3d93b54c749f3b5a4819'),
 ObjectId('5e9a3d93b54c749f3b5a481a'),
 ObjectId('5e9a3d93b54c749f3b5a481b'),
 ObjectId('5e9a3d93b54c749f3b5a481c'),
 ObjectId('5e9a3d93b54c749f3b5a481d'),
 ObjectId('5e9a3d93b54c749f3b5a481e'),
 ObjectId('5e9a3d93b54c749f3b5a481f'),
 ObjectId('5e9a3d93b54c749f3b5a4820'),
 ObjectId('5e9a3d93b54c749f3b5a4821'),
 ObjectId('5e9a3d93b54c749f3b5a4822'),
 ObjectId('5e9a3d93b54c749f3b5a4823'),
 ObjectId('5e9a3d93b54c749f3b5a4824'),
 ObjectId('5e9a3d93b54c749f3b5a4825'),
 ObjectId('5e9a3d93b54c749f3b5a4826')]
```

Analysis 3:

Analysing the best Airline one can choose based on the past data. This will help the passenger to get rid of delays and last minute flight cancellations.

```
In [95]:
# This is used to analyse the average delays by airline , number of times the flight got cancelled in a year
sqlDF2 = sqlContext.sql("SELECT AIRLINE ,avg(DEPARTURE_DELAY), sum(CANCELLED)/12 FROM flight_1 WHERE DEPARTURE_DELAY > 0 GROUP B
sqlDF2.show()
```

```
+-------+--------------------+--------------------------------------------------------------------+
|AIRLINE|avg(DEPARTURE_DELAY)|(CAST(sum(CAST(CANCELLED AS BIGINT)) AS DOUBLE) / CAST(12 AS DOUBLE))|
+-------+--------------------+--------------------------------------------------------------------+
|     HA|  16.844038518812667|                                                                 0.5|
|     AS|  26.045976219988063|                                                2.0833333333333335|
|     WN|   26.95237708779179|                                                18.666666666666668|
|     US|  28.500615360025574|                                                 9.416666666666666|
|     DL|   29.68744224907333|                                                               6.5|
|     VX|  30.285983147268915|                                                1.0833333333333333|
|     UA|   32.60218281036835|                                                             25.75|
|     AA|  34.370396577526186|                                                              29.5|
|     B6|    37.6171211334398|                                                 4.083333333333333|
|     OO|   39.20327908982818|                                                32.583333333333336|
|     MQ|   40.16399931715853|                                                41.166666666666664|
|     EV|  40.840515135641006|                                                32.833333333333336|
|     NK|  41.924149052583076|                                                 4.666666666666667|
|     F9|   44.54076748918121|                                                2.8333333333333335|
+-------+--------------------+--------------------------------------------------------------------+
```

Praveen Kumar Marichamy (0663980)
Saikarthik Kandikonda (0671359)

Converting it to pandas data frame

In [96]:
```
Airline_Delay = sqlDF2.toPandas()
```

In [97]:
```
Airline_Delay.columns = ['Airline','Avg_Departure_Delay','No_Of_Cancellations']
Airline_Delay.reset_index(drop=True, inplace=True)
Airline_Delay
```

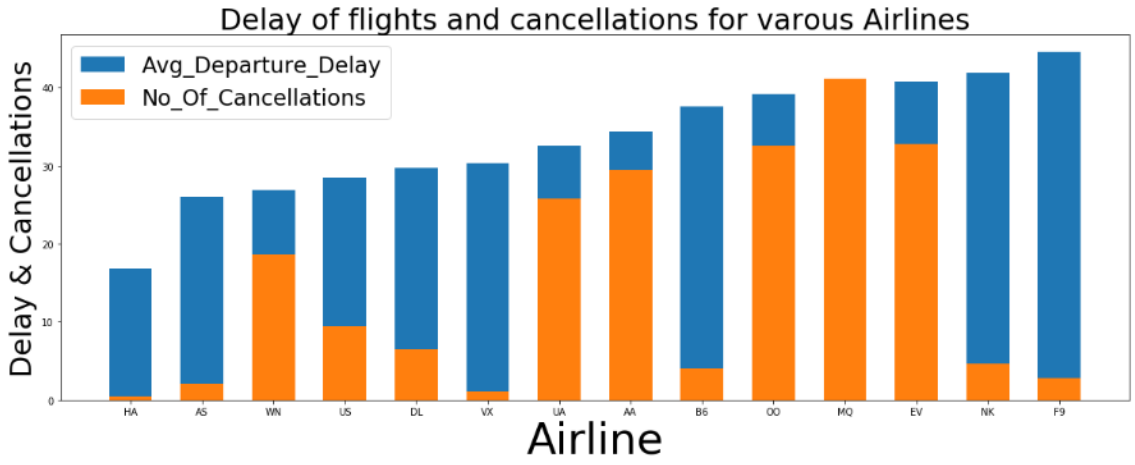
Out[97]:

	Airline	Avg_Departure_Delay	No_Of_Cancellations
0	HA	16.844039	0.500000
1	AS	26.045976	2.083333
2	WN	26.952377	18.666667
3	US	28.500615	9.416667
4	DL	29.687442	6.500000
5	VX	30.285983	1.083333
6	UA	32.602183	25.750000
7	AA	34.370397	29.500000
8	B6	37.617121	4.083333
9	OO	39.203279	32.583333
10	MQ	40.163999	41.166667
11	EV	40.840515	32.833333
12	NK	41.924149	4.666667
13	F9	44.540767	2.833333

Graph:

In [121]:
```
p1=plt.bar('Airline', 'Avg_Departure_Delay' ,width =0.6,data = Airline_Delay )
p2=plt.bar('Airline', 'No_Of_Cancellations' ,width =0.6,data = Airline_Delay )
plt.xlabel('Airline', fontsize=45)
plt.ylabel('Delay & Cancellations', fontsize=30)
plt.title('Delay of flights and cancellations for varous Airlines',   fontsize =30)
plt.legend((p1[0], p2[0]), ('Avg_Departure_Delay', 'No_Of_Cancellations'),prop={"size":23})
```

Out[121]: `<matplotlib.legend.Legend at 0x1dbeb4c91c8>`



Praveen Kumar Marichamy (0663980)
Saikarthik Kandikonda (0671359)

Inference : *The above analysis is useful when someone wants to be strategic in terms of choosing a Airline to avoid last minute cancellations and delays in journey. This also helps the companies to improve their services based on the data, which will eventually help them in improving the business eventually.*

MongoDB Insertion:

In [60]:
```
import pymongo
mng_client = pymongo.MongoClient('localhost', 27017)
mng_db = mng_client['dataexpo']
collection_name = 'delay'
db_cm = mng_db[collection_name]

#Insert Data
db_cm.insert(data)
```

```
/opt/anaconda3/lib/python3.7/site-packages/ipykernel_launcher.py:8: DeprecationWarning: insert is deprecated. Use ins
ert_one or insert_many instead.
```

Out[60]:
```
[ObjectId('5e9a3eb5b54c749f3b5a69b9'),
 ObjectId('5e9a3eb5b54c749f3b5a69ba'),
 ObjectId('5e9a3eb5b54c749f3b5a69bb'),
 ObjectId('5e9a3eb5b54c749f3b5a69bc'),
 ObjectId('5e9a3eb5b54c749f3b5a69bd'),
 ObjectId('5e9a3eb5b54c749f3b5a69be'),
 ObjectId('5e9a3eb5b54c749f3b5a69bf'),
 ObjectId('5e9a3eb5b54c749f3b5a69c0'),
 ObjectId('5e9a3eb5b54c749f3b5a69c1'),
 ObjectId('5e9a3eb5b54c749f3b5a69c2'),
 ObjectId('5e9a3eb5b54c749f3b5a69c3'),
 ObjectId('5e9a3eb5b54c749f3b5a69c4'),
 ObjectId('5e9a3eb5b54c749f3b5a69c5'),
 ObjectId('5e9a3eb5b54c749f3b5a69c6')]
```

```
MongoDB Enterprise > db.delay.find()
{ "_id" : ObjectId("5e9a3eb5b54c749f3b5a69b9"), "AIRLINE" : "HA", "avg(DEPARTURE_DELAY)" : 16.8440385188, "sum(CANCELLED)" : 6 }
{ "_id" : ObjectId("5e9a3eb5b54c749f3b5a69ba"), "AIRLINE" : "AS", "avg(DEPARTURE_DELAY)" : 26.04597622, "sum(CANCELLED)" : 25 }
{ "_id" : ObjectId("5e9a3eb5b54c749f3b5a69bb"), "AIRLINE" : "WN", "avg(DEPARTURE_DELAY)" : 26.9523770878, "sum(CANCELLED)" : 224 }
{ "_id" : ObjectId("5e9a3eb5b54c749f3b5a69bc"), "AIRLINE" : "US", "avg(DEPARTURE_DELAY)" : 28.50061536, "sum(CANCELLED)" : 113 }
{ "_id" : ObjectId("5e9a3eb5b54c749f3b5a69bd"), "AIRLINE" : "DL", "avg(DEPARTURE_DELAY)" : 29.6874422491, "sum(CANCELLED)" : 78 }
{ "_id" : ObjectId("5e9a3eb5b54c749f3b5a69be"), "AIRLINE" : "VX", "avg(DEPARTURE_DELAY)" : 30.2859831473, "sum(CANCELLED)" : 13 }
{ "_id" : ObjectId("5e9a3eb5b54c749f3b5a69bf"), "AIRLINE" : "UA", "avg(DEPARTURE_DELAY)" : 32.6021828104, "sum(CANCELLED)" : 309 }
{ "_id" : ObjectId("5e9a3eb5b54c749f3b5a69c0"), "AIRLINE" : "AA", "avg(DEPARTURE_DELAY)" : 34.3703965775, "sum(CANCELLED)" : 354 }
{ "_id" : ObjectId("5e9a3eb5b54c749f3b5a69c1"), "AIRLINE" : "B6", "avg(DEPARTURE_DELAY)" : 37.6171211334, "sum(CANCELLED)" : 49 }
{ "_id" : ObjectId("5e9a3eb5b54c749f3b5a69c2"), "AIRLINE" : "OO", "avg(DEPARTURE_DELAY)" : 39.2032790898, "sum(CANCELLED)" : 391 }
{ "_id" : ObjectId("5e9a3eb5b54c749f3b5a69c3"), "AIRLINE" : "MQ", "avg(DEPARTURE_DELAY)" : 40.1639993172, "sum(CANCELLED)" : 494 }
{ "_id" : ObjectId("5e9a3eb5b54c749f3b5a69c4"), "AIRLINE" : "EV", "avg(DEPARTURE_DELAY)" : 40.8405151356, "sum(CANCELLED)" : 394 }
{ "_id" : ObjectId("5e9a3eb5b54c749f3b5a69c5"), "AIRLINE" : "NK", "avg(DEPARTURE_DELAY)" : 41.9241490526, "sum(CANCELLED)" : 56 }
{ "_id" : ObjectId("5e9a3eb5b54c749f3b5a69c6"), "AIRLINE" : "F9", "avg(DEPARTURE_DELAY)" : 44.5407674892, "sum(CANCELLED)" : 34 }
```

Praveen Kumar Marichamy (0663980)
Saikarthik Kandikonda (0671359)

Analysis 4:

Analysis of which month has highest number of cancellations, which helps the passangers in planning their Journey accordingly

```
In [46]: # This is used to analyse the number of times teh flights got cancelled in a month
sqlDF6 = sqlContext.sql("SELECT MONTH, (sum(case when CANCELLED  in (1) then 1 end)/sum(case when CANCELLED  in (1,0) then 1 end
sqlDF6.show()
```

```
+-----+--------------------------------------------------------------------------------------------------------------------------------
-------------------------------------------------------------------+
|MONTH|((CAST(sum(CAST(CASE WHEN (CANCELLED IN (1)) THEN 1 END AS BIGINT)) AS DOUBLE) / CAST(sum(CAST(CASE WHEN (CANCELLED IN
(1, 0)) THEN 1 END AS BIGINT)) AS DOUBLE)) * CAST(100 AS DOUBLE))|
+-----+--------------------------------------------------------------------------------------------------------------------------------
-------------------------------------------------------------------+
|    1|
2.5495352875089368|
|    2|
4.780389150751064|
|    3|
2.1815860023160263|
|    4|
0.9316686969623891|
|    5|
1.1456901807470126|
|    6|
1.8098936885911208|
|    7|
0.9229563794606677|
|    8|
0.9895482394973126|
|    9|
0.44628838617817984|
|   10|
0.5047669001265003|
```

Converting the Spark DF to Pandas data frame for further analysis.

```
In [47]: cancellation = sqlDF6.toPandas()
```

```
In [48]: cancellation
```

Praveen Kumar Marichamy (0663980)
Saikarthik Kandikonda (0671359)

```
#Renaming the column name
cancellation.columns = ['Month', 'Chances of Flight getting cancelled']
```

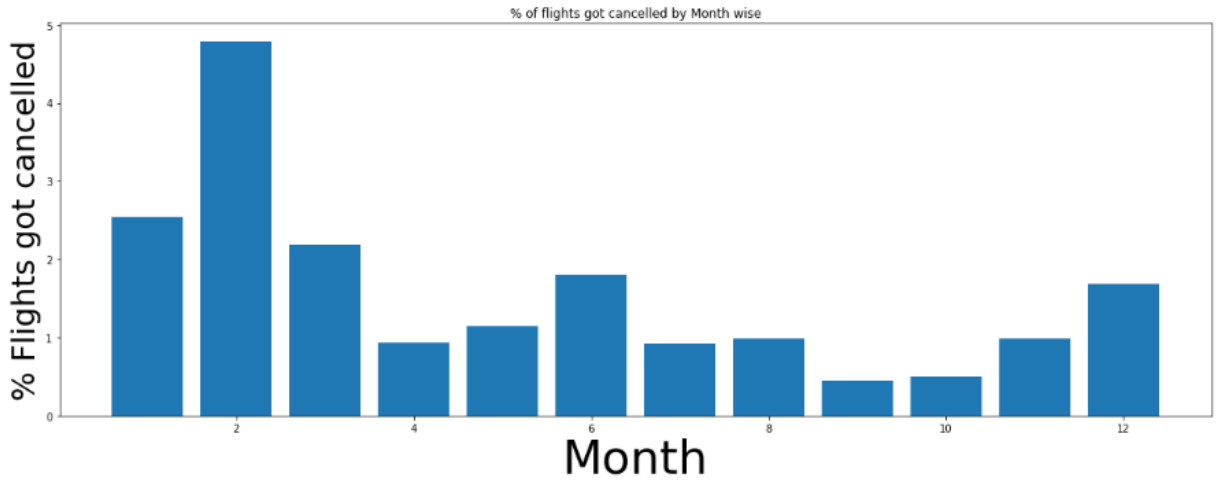
```
cancellation.reset_index(drop=True, inplace=True)
cancellation
```

	Month	Chances of Flight getting cancelled
0	1	2.549535
1	2	4.780389
2	3	2.181586
3	4	0.931669
4	5	1.145690
5	6	1.809894
6	7	0.922956
7	8	0.989548
8	9	0.446288
9	10	0.504767
10	11	0.982751
11	12	1.682491

Graph:

```
plt.bar('Month', 'Chances of Flight getting cancelled' ,width =0.8, data = cancellation )
plt.xlabel('Month', fontsize=45)
plt.ylabel('% Flights got cancelled', fontsize=30)
plt.title('% of flights got cancelled by Month wise')
```

```
Text(0.5, 1.0, '% of flights got cancelled by Month wise')
```



Inference: *From the above graph we can infer that the probability of the flights getting cancel are more in the months of January, February.*

Praveen Kumar Marichamy (0663980)
Saikarthik Kandikonda (0671359)

MongoDB Insertion:

In [73]:
```
import pymongo
mng_client = pymongo.MongoClient('localhost', 27017)
mng_db = mng_client['dataexpo']
collection_name = 'cancel'
db_cm = mng_db[collection_name]

#Insert Data
db_cm.insert(data)
```

```
/opt/anaconda3/lib/python3.7/site-packages/ipykernel_launcher.py:8: DeprecationWarning: insert is deprecated. Use ins
ert_one or insert_many instead.
```

Out[73]:
```
[ObjectId('5e9a3fb4b54c749f3b5a69c8'),
 ObjectId('5e9a3fb4b54c749f3b5a69c9'),
 ObjectId('5e9a3fb4b54c749f3b5a69ca'),
 ObjectId('5e9a3fb4b54c749f3b5a69cb'),
 ObjectId('5e9a3fb4b54c749f3b5a69cc'),
 ObjectId('5e9a3fb4b54c749f3b5a69cd'),
 ObjectId('5e9a3fb4b54c749f3b5a69ce'),
 ObjectId('5e9a3fb4b54c749f3b5a69cf'),
 ObjectId('5e9a3fb4b54c749f3b5a69d0'),
 ObjectId('5e9a3fb4b54c749f3b5a69d1'),
 ObjectId('5e9a3fb4b54c749f3b5a69d2'),
 ObjectId('5e9a3fb4b54c749f3b5a69d3')]
```

click to scroll output; double click to hide

```
MongoDB Enterprise > db.cancel.find()
{ "_id" : ObjectId("5e9a3fb4b54c749f3b5a69c8"), "Month" : 1, "Chances_of_Flight_getting_cancelled" : 2.5495352875 }
{ "_id" : ObjectId("5e9a3fb4b54c749f3b5a69c9"), "Month" : 2, "Chances_of_Flight_getting_cancelled" : 4.7803891508 }
{ "_id" : ObjectId("5e9a3fb4b54c749f3b5a69ca"), "Month" : 3, "Chances_of_Flight_getting_cancelled" : 2.1815860023 }
{ "_id" : ObjectId("5e9a3fb4b54c749f3b5a69cb"), "Month" : 4, "Chances_of_Flight_getting_cancelled" : 0.931668697 }
{ "_id" : ObjectId("5e9a3fb4b54c749f3b5a69cc"), "Month" : 5, "Chances_of_Flight_getting_cancelled" : 1.1456901807 }
{ "_id" : ObjectId("5e9a3fb4b54c749f3b5a69cd"), "Month" : 6, "Chances_of_Flight_getting_cancelled" : 1.8098936886 }
{ "_id" : ObjectId("5e9a3fb4b54c749f3b5a69ce"), "Month" : 7, "Chances_of_Flight_getting_cancelled" : 0.9229563795 }
{ "_id" : ObjectId("5e9a3fb4b54c749f3b5a69cf"), "Month" : 8, "Chances_of_Flight_getting_cancelled" : 0.9895482395 }
{ "_id" : ObjectId("5e9a3fb4b54c749f3b5a69d0"), "Month" : 9, "Chances_of_Flight_getting_cancelled" : 0.4462883862 }
{ "_id" : ObjectId("5e9a3fb4b54c749f3b5a69d1"), "Month" : 10, "Chances_of_Flight_getting_cancelled" : 0.5047669001 }
{ "_id" : ObjectId("5e9a3fb4b54c749f3b5a69d2"), "Month" : 11, "Chances_of_Flight_getting_cancelled" : 0.9827511048 }
{ "_id" : ObjectId("5e9a3fb4b54c749f3b5a69d3"), "Month" : 12, "Chances_of_Flight_getting_cancelled" : 1.6824906621 }
```