Course: Data Management and Big Data

CRN Number: 20997

Assignment: Final Project

***Project Name*: US Accidents- Countrywide Traffic Accidents Analysis.**

Email: taneja.ri@husky.neu.edu

Rishabh Taneja

PROFESSOR: VALERIY SHEVCHENKO

INTRODUCTION

This is a countrywide car accident dataset, which covers 49 states of the United States. The data is collected from February 2016 to December 2019, using several data providers, including two APIs that provide streaming traffic incident data. These APIs broadcast traffic data captured by a variety of entities, such as the US and state departments of transportation, law enforcement agencies, traffic cameras, and traffic sensors within the road-networks. Currently, there are about 3.0 million accident records in this dataset.

This data has been collected in real-time, using multiple Traffic APIs. Currently, it contains data that is collected from February 2016 to December 2019 for the Contiguous United States.

SUMMARY

- In this project, I decided to analyse the accidents dataset related to United States of America.
- To handle the dataset requirements of million+ records, I decided to analyse the dataset using DataBricks which is a cloud-based tool that makes use of PySpark to process the data.
- I also described on reasons to choose databricks over traditional tool such as jupyter notebook by demonstrating feasibility, flexibility and speed.
- I was successfully able to analyse the business problems associated with the accidents.
- Finally concluded in the end with insights and possible recommendations to solve them.

ANALYSIS

Identify a Real-World Problem that can be solved

The number of accidents taking place daily has always been a big problem. But recently, the number has been rising quickly where researchers, the police, the government and all organizations related to roadways have been working on finding a solution that can solve this issue. The number of deaths per year is nearly 1.25 Million which is haunting and a matter of concern. Thus, I decided to pick up this dataset to find valuable insights out of the data, find meaningful patterns and see if I can provide some analysis that might help make important decisions to curb this problem and bring the number of deaths per year down to as low as possible.

BUSINESS QUESTIONS TO BE SOLVED (PROJECT PROPOSAL) –

1. Is there any relationship between the attributes, especially severity?
2. Which states contribute the most to the accidents data?
3. Does weather really have an impact on accidents?
4. What time of the year or week contributes more towards the accident?

DATASET DESCRIPTION

This is a countrywide car accident dataset, which covers 49 states of the United States. The accident data are collected from February 2016 to December 2019, using several data providers,

including two APIs that provide streaming traffic incident data. These APIs broadcast traffic data captured by a variety of entities, such as the US and state departments of transportation, law enforcement agencies, traffic cameras, and traffic sensors within the road-networks. Currently, there are about 3.0 million accident records in this dataset.

DATASET SIZE: 3 Million+ records and 20+ Attributes

TOOLS & LANGUAGES TO BE USED: DataBricks, Python - PySpark, SQL.

FIELDS TO BE USED:

ID - This is a unique identifier of the accident record.

Source - Indicates the source of the accident report (i.e. the API which reported the accident.).

TMC - A traffic accident may have a Traffic Message Channel (TMC) code which provides a more detailed description of the event.

Severity - Shows the severity of the accident, a number between 1 and 4, where 1 indicates the least impact on traffic (i.e., short delay as a result of the accident) and 4 indicates a significant impact on traffic (i.e., long delay).

Start_Time - Shows the start time of the accident in the local time zone.

End_Time - Shows end time of the accident in the local time zone.

Distance(mi) - The length of the road extent affected by the accident.

Description - Shows a natural language description of the accident.

County - Shows the county in the address field.

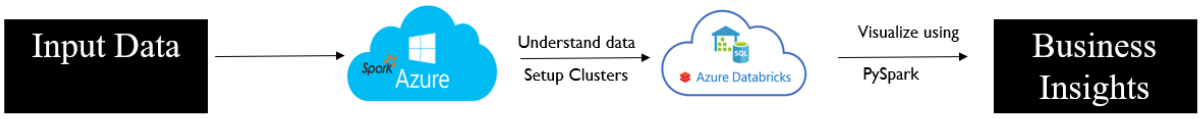
State - Shows the state in the address field.

Zipcode - Shows the zip code in the address field.

Country - Shows the country in the address field.

Timezone - Shows timezone based on the location of the accident (eastern, central, etc.)

DATA ANALYTICS PROCESS WORKFLOW



- Import data into Databricks Azure.
- Import data as RDD and clean using PySpark.
- Visualize using PySpark.
- Generate Business insights

WHY DATABRICKS AZURE (METHODOLOGY/TOOL)

The dataset was imported using Jupyter notebook at the beginning to check the capability of the local system and then to data bricks Azure to understand the most important feature of big data processing system i.e. scalability and a significant 20 second time difference was observed with Azure being faster than Jupyter Notebook. This is because Spark in Data Bricks Azure uses a

Directed Acyclic Spark Streaming service which is much faster than the traditional T-SQL DW connectors used in Jupyter Notebooks.

In [4]:

```
import pandas as pd
import time
start_time = time.time()

df= pd.read_csv("C:\\Users\\Arun Kumar Gorantla\\Downloads\\us-accidents\\US_Accidents_Dec19.csv")
end_time = time.time()
print(end_time-start_time)
```

```
19.63708472251892
```

```
# Create a view or table

temp_table_name = "US_Accidents_Dec19_csv"

df.createOrReplaceTempView(temp_table_name)
```

```
Command took 0.12 seconds -- by gorantla.a@husky.neu.edu at 3/23/2020, 12:06:09 AM on My Cluster
```

Content

Proof of Analysis Using DataBricks:

Step1 - Loading the dataset, creating a schema and checking the datatypes of each attribute

```
# File location and type
file_location = "/FileStore/tables/accidents.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.
accidents_df = spark.read.format(file_type) \
  .option("inferSchema", infer_schema) \
  .option("header", first_row_is_header) \
  .option("sep", delimiter) \
  .load(file_location)
```

In the first step, I have loaded the Dataset onto the Databricks platform and created a schema. Since it is CSV file, I am working on parameters pertaining to it else these parameters would be ignored.

Displaying the top 1000 rows of the dataset

```
display(accidents_df)
```

ID	Source	TMC	Severity	Start_Time	End_Time	Start_Lat	Start_Lng	End_Lat	End_Lng	Distance(mi)	Description	Number	Street
A-1	MapQuest	201	3	2016-02-08T05:46:00.000+0000	2016-02-08T11:00:00.000+0000	39.865147	-84.058723	null	null	0.01	Right lane blocked due to accident on I-70 Eastbound at Exit 41 OH-235 State Route 4.	null	I-70 E
A-2	MapQuest	201	2	2016-02-08T06:07:59.000+0000	2016-02-08T06:37:59.000+0000	39.928059	-82.831184	null	null	0.01	Accident on Brice Rd at	2584	Brice F

Showing the first 1000 rows.

In the picture above, all the 1000 rows of the dataset are displayed. It has the information for every accident that took place in the United States from February 2016 to December 2019. The information in these rows has the description of the accident, where it happened, the number, street, city, county, state, zip code, timezone etc.

Creating a table so I can run SQL commands

```
# Create a view or table

accidents_table = "accidents_csv"

accidents_df.createOrReplaceTempView(accidents_table)
```

In the above picture, a table is created to run the SQL commands.

Using the select SQL command I check the contents of the table.

```
%sql

/* Query the created temp table in a SQL cell */

select * from `accidents_csv`
```

ID	Source	TMC	Severity	Start_Time	End_Time	Start_Lat	Start_Lng	End_Lat	End_Lng	Distance(mi)	Description	Number	Street
A-1	MapQuest	201	3	2016-02-08T05:46:00.000+0000	2016-02-08T11:00:00.000+0000	39.865147	-84.058723	null	null	0.01	Right lane blocked due to accident on I-70 Eastbound at Exit 41 OH-235 State Route 4.	null	I-70 E
A-2	MapQuest	201	2	2016-02-08T06:07:59.000+0000	2016-02-08T06:37:59.000+0000	39.928059	-82.831184	null	null	0.01	Accident on Brice Rd at	2584	Brice R

Showing the first 1000 rows.

Storing the data frame into cache memory to increase the speed

```
accidents_df.cache()
```

```
Out[5]: DataFrame[ID: string, Source: string, TMC: double, Severity: int, Start_Time: timestamp, End_Time: timestamp, Start_Lat: double, Start_Lng: double, End_Lat: double, End_Lng: double, Distance(mi): double, Description: string, Number: double, Street: string, Side: string, City: string, County: string, State: string, Zipcode: string, Country: string, Timezone: string, Airport_Code: string, Weather_Timestamp: timestamp, Temperature(F): double, Wind_Chill(F): double, Humidity(%): double, Pressure(in): double, Visibility(mi): double, Wind_Direction: string, Wind_Speed(mph): double, Precipitation(in): double, Weather_Condition: string, Amenity: boolean, Bump: boolean, Crossing: boolean, Give_Way: boolean, Junction: boolean, No_Exit: boolean, Railway: boolean, Roundabout: boolean, Station: boolean, Stop: boolean, Traffic_Calming: boolean, Traffic_Signal: boolean, Turning_Loop: boolean, Sunrise_Sunset: string, Civil_Twilight: string, Nautical_Twilight: string, Astronomical_Twilight: string]
```

Step2 - Performing the Exploratory Data Analysis

Converting the dataframe to pandas and checking the statistics of each attribute

```
##descriptive statistics#
accidents_df.describe().toPandas().transpose()
```

Out[6]:

	0	1	2	3	4
summary	count	mean	stddev	min	max
ID	2974335	None	None	A-1	A-999999
Source	2974335	None	None	Bing	MapQuest-Bing
TMC	2246264	207.83163198982845	20.32958633116194	200.0	406.0
Severity	2974335	2.360190092911525	0.5414733263296426	1	4
Start_Lat	2974335	36.493605002031636	4.918848998696759	24.555269	49.002201
Start_Lng	2974335	-95.4262537385967	17.218805933372185	-124.623833	-67.113167
End_Lat	728071	37.58087071324656	5.00475695787924	24.57011	49.075
End_Lng	728071	-99.97603191130777	18.41664728374778	-124.497829	-67.109242
Distance(mi)	2974335	0.2855653569462957	1.5483920254566632	0.0	333.630004883
Description	2974334	None	None	middle lane blocked due to accident on I-95 N...	~6 lane blocked due to accident on I-5 Southbo...
Number	1056730	5837.003543951624	15159.278074287868	0.0	9999997.0
Street	2974335	None	None	1 Mile Rd	william Carey Dr
Side	2974335	None	None		R
City	2974252	None	None	Aaronsburg	Zwingle
County	2974335	None	None	Abbeville	Yuma
State	2974335	None	None	AL	WY
Zipcode	2973455	59328.94336883425	30637.325538650803	01001	99401-9712
Country	2974335	None	None	US	US
Timezone	2971172	None	None	US/Central	US/Pacific
Airport_Code	2968644	None	None	K01M	KZZV
Temperature(F)	2918272	62.35120314350398	18.78854911265782	-77.8	170.6
Wind_Chill(F)	1121712	51.32684860284989	25.19127055656769	-65.9	115.0
Humidity(%)	2915162	65.40541554808961	22.556763456537613	1.0	100.0
Pressure(in)	2926193	29.831895008292193	0.7213808414176774	0.0	33.04
Visibility(mi)	2908644	9.15076997047429	2.892113743736492	0.0	140.0
Wind_Direction	2929234	None	None	CALM	West
Wind_Speed(mph)	2533495	8.298063781457618	5.138545725098774	0.0	822.8
Precipitation(in)	975977	0.020494509604222225	0.23577039560945023	0.0	25.0
Weather_Condition	2908403	None	None	Blowing Dust	Wintry Mix / Windy
Sunrise_Sunset	2974242	None	None	Day	Night
Civil_Twilight	2974242	None	None	Day	Night
Nautical_Twilight	2974242	None	None	Day	Night
Astronomical_Twilight	2974242	None	None	Day	Night

This is the descriptive statistics performed on the attributes to check if there are any unusual stats of any feature that must be handled before further analysis. There was no such unusual behaviour noticed.

Importing all the necessary pyspark libraries

```
import pandas as pd
import numpy as np
from pyspark import SparkConf, SparkContext
from pyspark.sql import SQLContext
from pyspark.sql.functions import dayofmonth,hour,dayofyear,weekofyear,month,year,format_number,date_format,mean, date_format, datediff, to_date, lit
import math
from pyspark.sql.functions import mean as _mean, stddev as stddev, col
from pyspark.sql.types import IntegerType
import matplotlib.pyplot as plt
from pyspark.sql.functions import unix_timestamp, from_unixtime, date_format
from pyspark.sql.functions import date_format
from pyspark.sql.functions import *
from pyspark.sql.types import *
from pyspark.sql.functions import monotonically_increasing_id
import matplotlib.pyplot as plt
import seaborn as sns
import datetime
plt.style.use('fivethirtyeight')
from pyspark.sql.functions import isnan, when, count, col
```

After importing the libraries, I created a pandas data frame to start performing the data analysis using the command. Almost all the libraries are important to run the commands however, there are some libraries which sets the base which are pandas, numpy for computation, pyspark.sql.functions and matplotlib for visualisation.

accidents_pd = accidents_df.toPandas()

```
print('Rows     :',accidents_df.count())
print('Columns  :',len(accidents_df.columns))
print('\nFeatures :\n     :',accidents_df.columns)
print('\nMissing values    :',accidents_pd.isnull().values.sum())
print('\nUnique values :  \n',accidents_pd.nunique())
```

```
Rows     : 2974335
Columns  : 49

Features :
     : ['ID', 'Source', 'TMC', 'Severity', 'Start_Time', 'End_Time', 'Start_Lat', 'Start_Lng', 'End_Lat', 'End_Lng', 'Distance(mi)', 'Description', 'Numbe
r', 'Street', 'Side', 'City', 'County', 'State', 'Zipcode', 'Country', 'Timezone', 'Airport_Code', 'Weather_Timestamp', 'Temperature(F)', 'Wind_Chill(F)',
'Humidity(%)', 'Pressure(in)', 'Visibility(mi)', 'Wind_Direction', 'Wind_Speed(mph)', 'Precipitation(in)', 'Weather_Condition', 'Amenity', 'Bump', 'Crossin
g', 'Give_Way', 'Junction', 'No_Exit', 'Railway', 'Roundabout', 'Station', 'Stop', 'Traffic_Calming', 'Traffic_Signal', 'Turning_Loop', 'Sunrise_Sunset', 'C
ivil_Twilight', 'Nautical_Twilight', 'Astronomical_Twilight']

Missing values    : 11817022

Unique values :
 ID                     2974335
Source                       3
TMC                         21
Severity                     4
Start_Time             2743101
End_Time               2761499
Start_Lat              1002359
Start_Lng               985099
```

Here, I tried to find unique values and null values if there are any to handle them.

All attributes with the object data type are collected and stored using the below commands.

```
#collecting all the attributes with object datatypes
accidents_pd.select_dtypes(exclude=['int','float']).columns

Out[92]: Index(['ID', 'Source', 'Severity', 'Start_Time', 'End_Time', 'Description',
       'Street', 'Side', 'City', 'County', 'State', 'Zipcode', 'Country',
       'Timezone', 'Airport_Code', 'Weather_Timestamp', 'Wind_Direction',
       'Weather_Condition', 'Amenity', 'Bump', 'Crossing', 'Give_Way',
       'Junction', 'No_Exit', 'Railway', 'Roundabout', 'Station', 'Stop',
       'Traffic_Calming', 'Traffic_Signal', 'Turning_Loop', 'Sunrise_Sunset',
       'Civil_Twilight', 'Nautical_Twilight', 'Astronomical_Twilight',
       'DayOfWeek'],
      dtype='object')
```

The first few rows of the description column are checked.

```
#First few rows from description column
accidents_pd['Description'].head()

Out[11]: 0    Right lane blocked due to accident on I-70 Eas...
1    Accident on Brice Rd at Tussing Rd. Expect del...
2    Accident on OH-32 State Route 32 Westbound at ...
3    Accident on I-75 Southbound at Exits 52 52B US...
4    Accident on McEwen Rd at OH-725 Miamisburg Cen...
Name: Description, dtype: object
```

Working with unique values to determine the actual source of data and number of time-zones considered

```
print(accidents_pd['Source'].unique())
print(accidents_pd['Description'].unique())
print(accidents_pd['Timezone'].unique())
print(accidents_pd['Amenity'].unique())

['MapQuest' 'MapQuest-Bing' 'Bing']
['Right lane blocked due to accident on I-70 Eastbound at Exit 41 OH-235 State Route 4.'
 'Accident on Brice Rd at Tussing Rd. Expect delays.'
 'Accident on OH-32 State Route 32 Westbound at Dela Palma Rd. Expect delays.'
 ... 'Ramp closed to Bristol St - Road closed due to accident.'
 'At Friars Rd - Accident. Center lane blocked.'
 'Ramp closed to The City Dr/Exit 14A - Road closed due to accident.']
['US/Eastern' 'US/Pacific' None 'US/Central' 'US/Mountain']
[False  True]
```

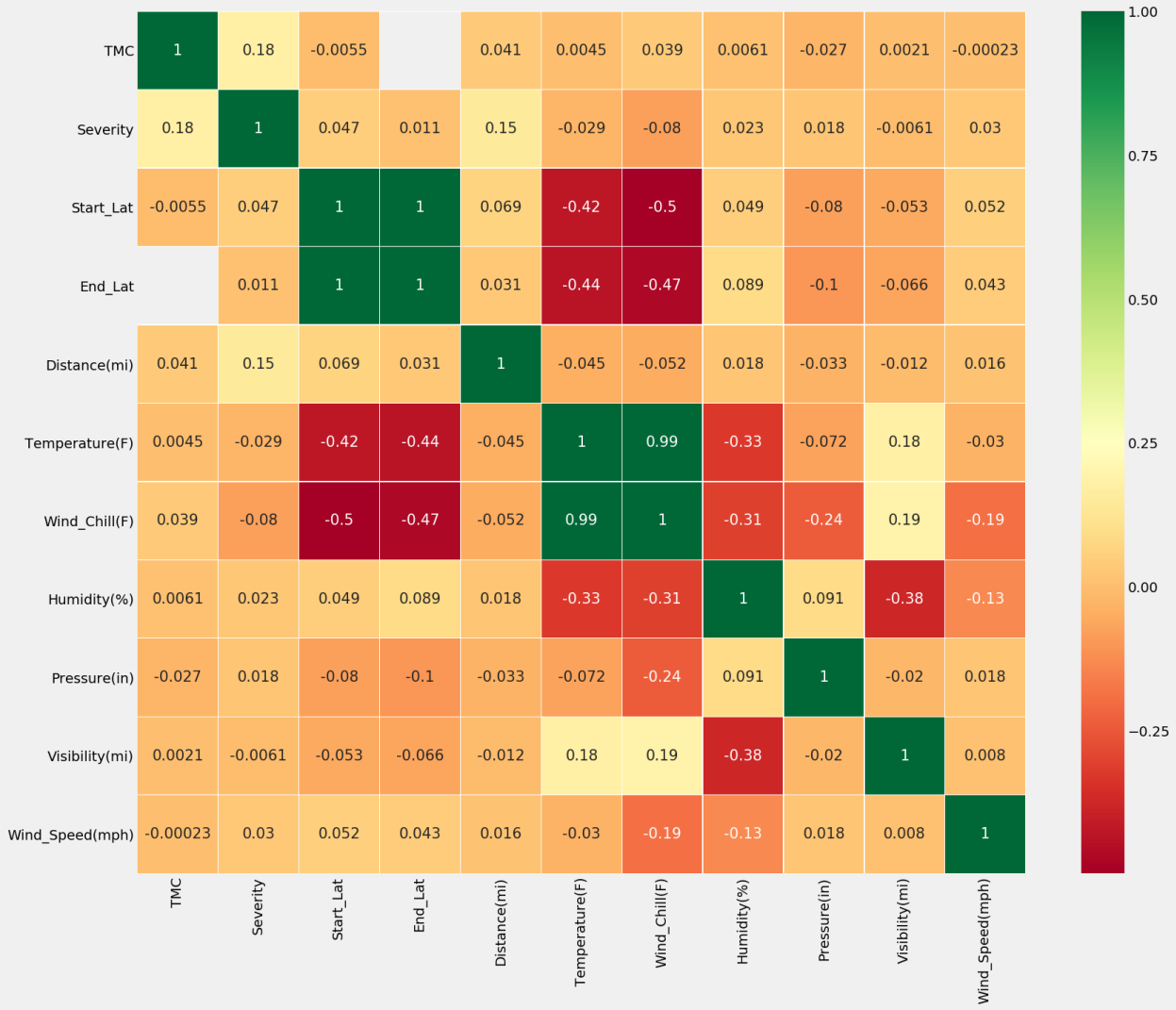
Step3- Analyzing the data visually to solve business questions

Business Question 1- Is there any relationship among the attributes, especially severity?

In order to identify the factors affecting the severity a heat map was generated using the below commands.

```
#Creating a heatmap to find the correlation among each attribute contributing to the accident
fig=sns.heatmap(accidents_pd[['TMC','Severity','Start_Lat','End_Lat','Distance(mi)','Temperature(F)','Wind_Chill(F)','Humidity(%)','Pressure(in)','Visibility
(mi)','Wind_Speed(mph)']].corr(),annot=True,cmap='RdYlGn',linewidths=0.2,annot_kws={'size':15})
fig=plt.gcf()
fig.set_size_inches(18,15)
plt.xticks(fontsize=14)
plt.yticks(fontsize=14)
plt.show()
display()
```

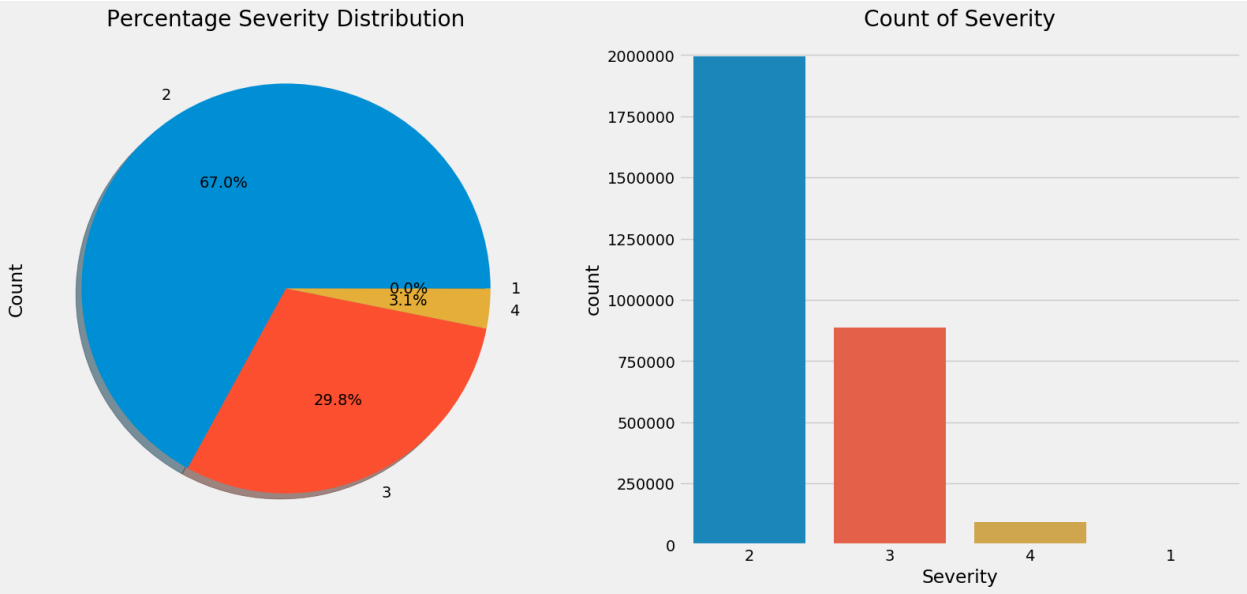
Heat Map:



We observe that variables “temperature” and “Wind_Chill” are positively correlated (0.99). “Start_Lat” and “End_Lat” latitudes have slight negative correlation with Temperature and “Wnd_Chill” respectively. As we are focusing on Severity, we notice that “Severity” and “Visibility” has a negative correlation which we would see if ‘weather’ plays any role in it or not.

Count of severity for each type is calculated using the below commands to generate a bar chart and pie chart reflecting the same.

```
#creating a bar graph and a pie-chart to demonstrate percentage distribution of the severity
f,ax=plt.subplots(1,2,figsize=(18,8))
accidents_pd['Severity'].value_counts().plot.pie(explode=None,autopct='%1.1f%%',ax=ax[0],shadow=True)
ax[0].set_title('Percentage Severity Distribution')
ax[0].set_ylabel('Count')
sns.countplot('Severity',data=accidents_pd,ax=ax[1],order=accidents_pd['Severity'].value_counts().index)
ax[1].set_title('Count of Severity')
plt.show()
display()
```



The range of Severity is from 1 to 4 where:

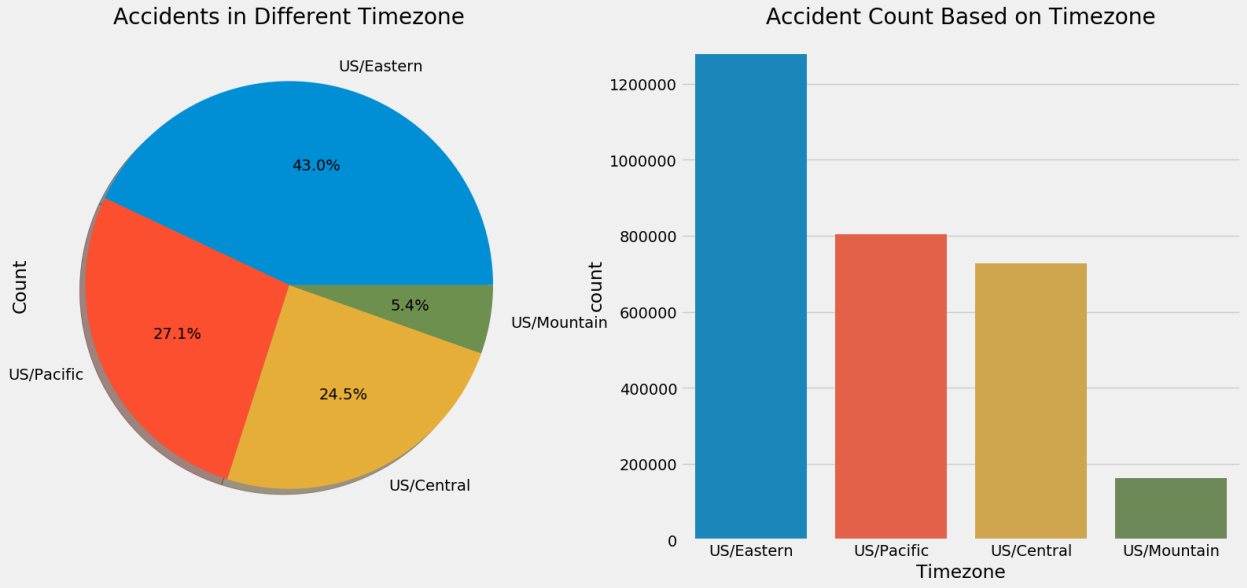
1. Indicates the accident did not have much impact on the traffic (No delay as such)
2. Indicates the accident had a minor impact on the traffic (Short delay)
3. Indicates the accident had moderate impact on the traffic (Moderate delay)
4. Indicates the accident had a very high impact on the traffic (High delay and clutter)

We notice that “2” contribute the most which basically means that the severity of the accidents was not that high and did not create much chaos for the ongoing vehicles.

Business Question 2 - Which states contribute the most to the accidents data?

Visually analyzing the percentage distribution of the number of accidents based on time zones

```
#creating a bar graph and a pie-chart to demonstrate percentage distribution of the number of accidents based on timezone
f,ax=plt.subplots(1,2,figsize=(18,8))
accidents_pd['Timezone'].value_counts().plot.pie(explode=None,autopct='%1.1f%%',ax=ax[0],shadow=True)
ax[0].set_title('Accidents in Different Timezone')
ax[0].set_ylabel('Count')
sns.countplot('Timezone',data=accidents_pd,ax=ax[1],order=accidents_pd['Timezone'].value_counts().index)
ax[1].set_title('Accident Count Based on Timezone')
plt.show()
display()
```

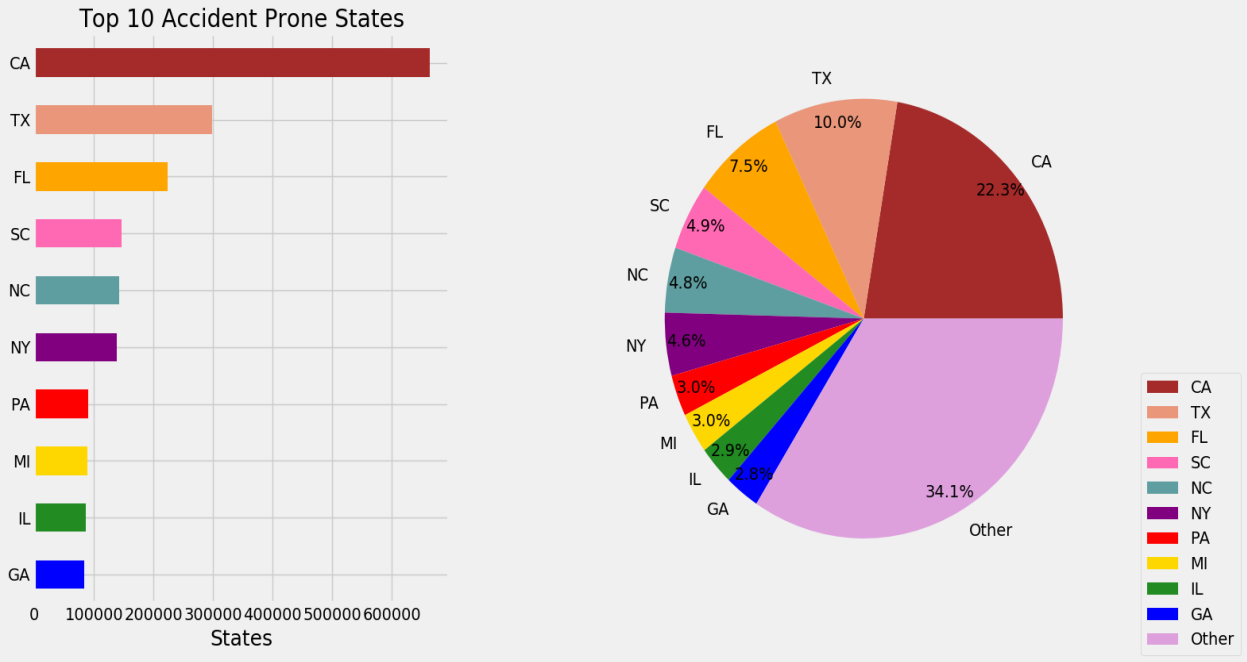


"US/Eastern" Time zone has the highest Accident count and "US/Mountain" has the least.

- The east standard time states contribute most to accidents data at 120,0000.
- The pacific time comes second at 100,0000.
- The central time US states come in at third in the graph at 800000
- The Mountain time US states come in at fourth and the least contribution to accidents at 200000.
- States like Pennsylvania and South Carolina fall in the Eastern time Zone.

Top 10 states prone to accidents based on the data

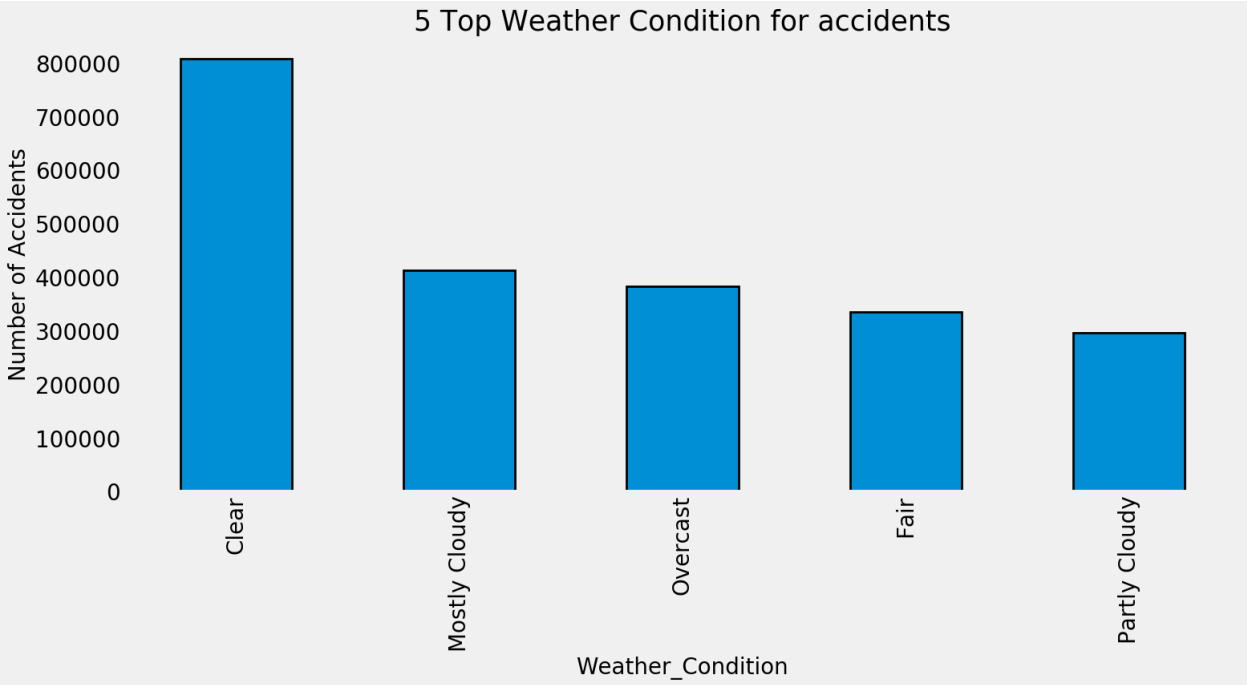
```
#Creating a bar graph and a pi-chart to see the percentage distribution of the
from pyspark.sql.functions import sum as fsum
fig,ax=plt.subplots(1,2,figsize=(15,8))
clr = ("blue", "forestgreen", "gold", "red", "purple",'cadetblue','hotpink','orange','darksalmon','brown')
accidents_pd.State.value_counts().sort_values(ascending=False)[:10].sort_values().plot(kind='barh',color=clr,ax=ax[0])
ax[0].set_title("Top 10 Accident Prone States",size=20)
ax[0].set_xlabel('States',size=18)
count=accidents_pd['State'].value_counts()
groups=list(accidents_pd['State'].value_counts().index)[:10]
counts=list(count[:10])
counts.append(count.agg('sum')-count[:10].agg('sum'))
groups.append('Other')
type_dict=pd.DataFrame({"group":groups,"counts":counts})
clr1=('brown','darksalmon','orange','hotpink','cadetblue','purple','red','gold','forestgreen','blue','plum')
qx = type_dict.plot(kind='pie', y='counts', labels=groups,colors=clr1,autopct='%1.1f%%', pctdistance=0.9, radius=1.2,ax=ax[1])
plt.legend(loc=0, bbox_to_anchor=(1.15,0.4))
plt.subplots_adjust(wspace =0.5, hspace =0)
plt.ioff()
plt.ylabel('')
display()
```



- We observe that “CA” is the most accident-prone area in United States and “TX” state has the second highest. The highest state that is prone to accidents is California at 22.3 %.
- And the least prone state is Georgia at 2.8%.

Business Question 3 - Does weather really have an impact on accidents?

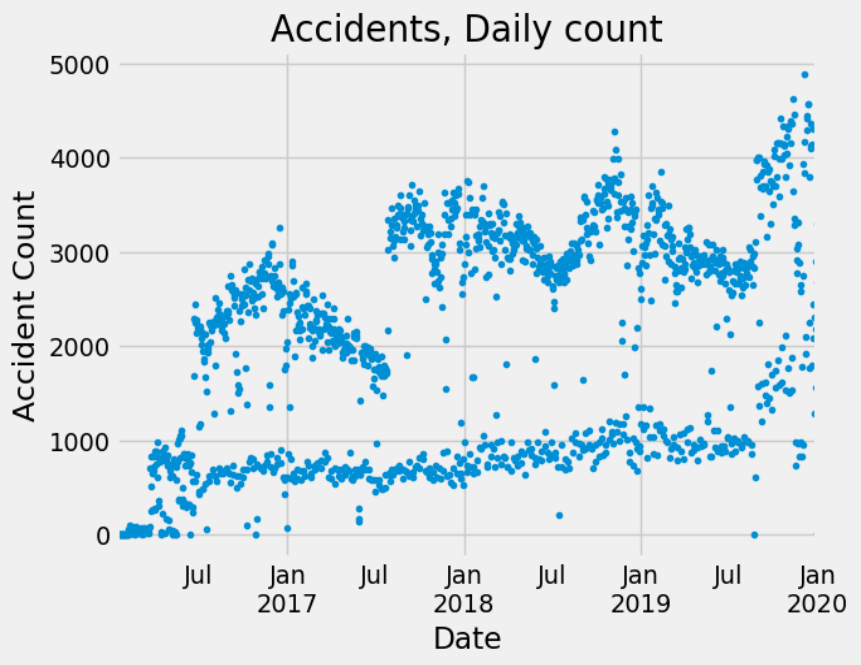
```
#Creating a bar graph based on the weather patterns during the accidents
fig, ax=plt.subplots(figsize=(16,7))
accidents_pd['Weather_Condition'].value_counts().sort_values(ascending=False).head(5).plot.bar(width=0.5,edgecolor='k',align='center',linewidth=2)
plt.xlabel('Weather_Condition',fontsize=20)
plt.ylabel('Number of Accidents',fontsize=20)
ax.tick_params(labelsize=20)
plt.title('5 Top Weather Condition for accidents',fontsize=25)
plt.grid()
plt.ioff()
display()
```



- We observe that more accidents take place in clear weather conditions and least when it is partly cloudy. The reason may be high speed or sharp turning curves that cause higher number of accidents in clear skies. The greatest number of accidents is when the weather is “Clear” which show that the way people are driving is the reason for the accidents.
- All the weather conditions contribute almost equally for the remaining accidents.

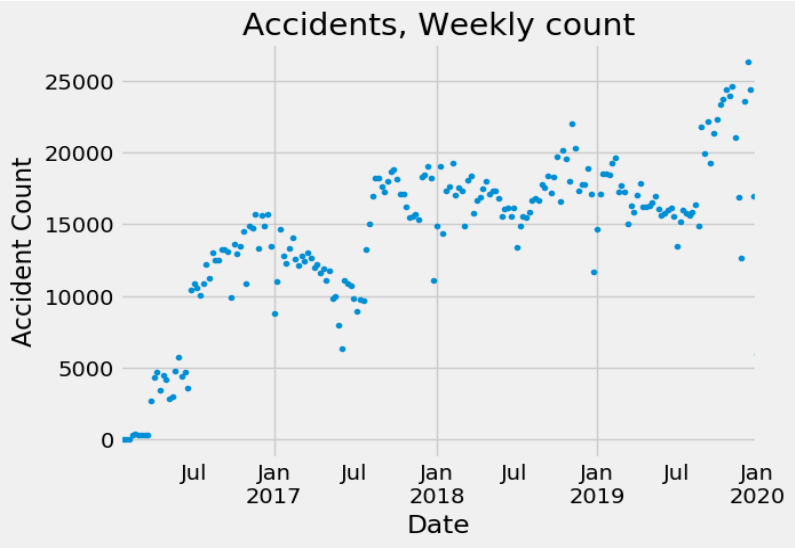
Business Question 4 - Which time of the year or week contributes more towards the accident?

```
#Plotting scatter plots for the number of number accidents for daily, weekly and yearly
freq_text = {'D':'Daily','W':'Weekly','Y':'Yearly'}
for i, (fr,text) in enumerate(freq_text.items(),1):
  sample = accidents_pd.ID['2016':].resample(fr).count()
  sample.plot(style='.')
  plt.title('Accidents, {} count'.format(text))
  plt.xlabel('Date')
  plt.ylabel('Accident Count')
  display()
```



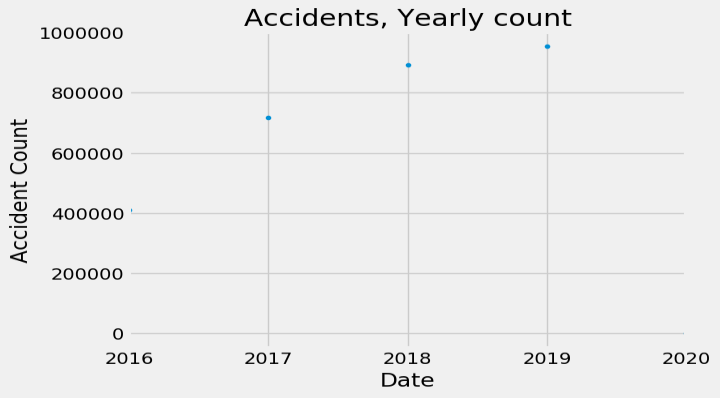
- We observe that daily accident count is higher during December and January whereas July and August have the least. There should be increased monitoring of traffic during December, January and appropriate steps should be taken to decrease the accidents in the coming years. In this case, we aggregated the daily count of the accidents and tried to show a trend over the year.
- We can notice that there is an increasing and a seasonal trend.

• We also noticed that there is a decreasing trending or a dip during the month of July and it has been the same for years 2016-2019.



• We can observe that January has the highest weekly count of accidents over the years. The reason for this might be due to fog and unclear skies in winter. In this case, we aggregated the weekly count of the accidents to see the trend over the year.

• Here, we can notice that for the year 2017 once the weekly accident count dropped in the month of July to 10,000, it suddenly increased to 15,000 which is not a trend seen in other years.

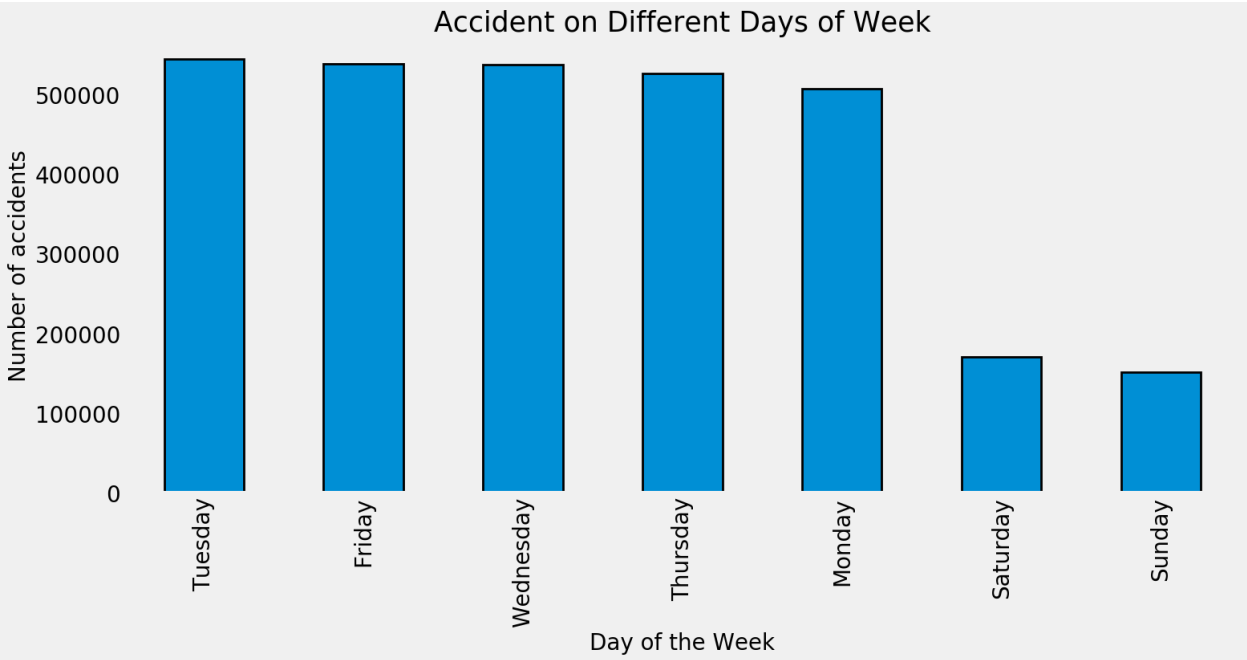


We observe that yearly count of accidents has been increasing over the years. Public awareness and proper traffic monitoring can help reduce the Accidents. Speed-limits also help to reduce the number of accidents.

Checking for the number of accidents that happen day-wise

```
#converting all the variables to their required data types to analyse
accidents_pd['Start_Time'] = pd.to_datetime(accidents_pd['Start_Time'], format="%Y/%m/%d %H:%M:%S")
accidents_pd['DayOfWeekNum'] = accidents_pd['Start_Time'].dt.dayofweek
accidents_pd['DayOfWeek'] = accidents_pd['Start_Time'].dt.weekday_name
accidents_pd['MonthDayNum'] = accidents_pd['Start_Time'].dt.day
accidents_pd['HourOfDay'] = accidents_pd['Start_Time'].dt.hour
```

```
#plotting a bar-graph for the number of accidents on different days of week
fig, ax=plt.subplots(figsize=(16,7))
accidents_pd['DayOfWeek'].value_counts(ascending=False).plot.bar(width=0.5,edgecolor='k',align='center',linewidth=2)
plt.xlabel('Day of the Week',fontsize=20)
plt.ylabel('Number of accidents',fontsize=20)
ax.tick_params(labelsize=20)
plt.title('Accident on Different Days of Week',fontsize=25)
plt.grid()
plt.ioff()
display()
```



We can observe that Tuesday and Friday have the highest number of Accidents whereas Saturday and Sunday have the least. The reason for less accidents on these days could be due to most people having a holiday on the weekends and few people do not come out or they come out at different times of the day. However, keen observation and regulation of speed and traffic should be done in order to reduce the number of accidents taking place.

Comments

Pros:

1. Using databricks platforms for the first time has been easy to navigate and understand its functionality.
2. Creating spark cluster is easy and attaching them to the notebook has been straightforward.
3. The User experience has been good, it has workspace with notebook, dashboard, cluster management.
4. We get visualizations of the data in the form of graphs, and charts to get a better and quick overall understanding of the data we are analyzing.
5. Multiple users can work on the same cluster. The sharing of this space is really useful in saving cost and time.

Cons:

1. I would say the major drawback for Databricks Is its cost, I have used the community version to complete the assignment.
2. The subscription is very pricey and not everyone can utilize all the features the platform can offer.
3. It is Time-consuming as it must process huge amounts of data.

CONCLUSION

Storing the data frame into cache memory increases the speed of the execution. We observe that Weekends have a smaller number of accidents for which the reason might be holidays. The Accidents have been increasing over the years, appropriate measures are to be taken to control the number of accidents taking place. The keen observation and regulation of speed and traffic should be done in order to reduce the number of accidents taking place. Most accidents take place when the weather is clear. Below is the stats from the work done above –

- We noticed that “Severity” and “Visibility” has a negative correlation.
- From the severity distribution, we see that the accidents did not cause much delay in the traffic and were not that severe.
- We can see that 65% of the accidents are in the category 2. 32% in the category 3 and 3% in the category 4.
- The east standard time states contribute most to accidents data at 120,0000.
- More accident happens in the Easter Time Zone. Sates like Pennsylvania and South Carolina fall in the Eastern time Zone.
- The greatest number of accidents is when the weather is “Clear” which show that the way people are driving is the reason for the accidents.
- The average number of accidents during weekdays is way more as compared to the number of accidents during the weekends.

FUTURE RECOMMENDATION

This project can be further improved by performing regression models to predict the number of accidents that could take place in the near future based on the variables that clearly define the target variable.

REFERENCES

1. Moosavi, Sobhan, Mohammad Hossein Samavatian, Srinivasan Parthasarathy, and Rajiv Ramnath. "A Countrywide Traffic Accident Dataset.", 2019.

2. Moosavi, Sobhan, Mohammad Hossein Samavatian, Srinivasan Parthasarathy, Radu Teodorescu, and Rajiv Ramnath. "Accident Risk Prediction based on Heterogeneous Sparse Data: New Dataset and Insights." In proceedings of the 27th ACM SIGSPATIAL International Conference on Advances in Geographic Information Systems, ACM, 2019.

3. Databricks Connect: Bringing the capabilities of hosted Apache Spark™ to applications and microservices - The Databricks Blog. (2020, March 18). Retrieved from https://databricks.com/blog/2019/06/14/databricks-connect-bringing-the-capabilities-of-hosted-apache-spark-to-applications-and-microservices.html

4. Moosavi, S. (2020, January 17). US Accidents (3.0 million records). Retrieved from https://www.kaggle.com/sobhanmoosavi/us-accidents

5. Gupta, A., Ankit, & Ubs. (2019, June 24). Complete Guide on Data Frames Operations in PySpark. Retrieved from https://www.analyticsvidhya.com/blog/2016/10/spark-dataframe-and-operations/

HOW TO RUN THE CODE -

1. Download the “accidents.csv” dataset.

2. Store it on your local disk

3. Open DataBricks web application and upload your “accidents.csv” dataset onto the database. Since it is a cloud-based technology, the huge dataset would be stored in their cloud and you may access it directly.

4. Load the .ipynb or the .py file and run it on the cluster you want.