Problem description

This project aims to analyze the data of deaths in United states form the year 2005 till 2015. The main analyze is done over Age, gender, cause of death, date and race.

Project pipeline (Khaled - Moustafa)

- we preferred to work on Google Colab as it has decent computation power in addition to the virsitily of data transfer and handling.

Processing steps

- The steps of each analysis idea is as follows:
    1. load the data
    2. explore features
    3. clean the data,convert to the proper type and fill/remove the nulls.
    4. group and aggregate.
    5. save the reduced data for visualization and further analysis.

Data exploration

The Data contained lots of convoluted features that had the following problems:

1. Some data are just nulls.
2. There are lots of mislabeling in the values of each column (like putting a random sting in the years section).
3. All the data were strings which made it hard for us to make any arithmetic analysis without converting it into proper numeric values.

Knowing we have these problems, we focused our attention to get the results in a categorical manner and disregard all the previous outliers because:

1. The mislabeled rows are very few and will not get to be in the top 10 counts anyway.
2. Some columns is properly nullified like (130_infant_cause_recode) which should be only filled if the dead person is infant.
3. We didn't need to handle all the columns as many of them will be dropped anyway.

The columns and types of the data-frame are as follows :

```
[
  ("resident_status", "string"),
  ("education_1989_revision", "string"),
  ("education_2003_revision", "string"),
  ("education_reporting_flag", "string"),
  ("month_of_death", "string"),
  ("sex", "string"),
  ("detail_age_type", "string"),
  ("detail_age", "string"),
  ("age_substitution_flag", "string"),
  ("age_recode_52", "string"),
  ("age_recode_27", "string"),
  ("age_recode_12", "string"),
  ("infant_age_recode_22", "string"),
  ("place_of_death_and_decedents_status", "string"),
  ("marital_status", "string"),
  ("day_of_week_of_death", "string"),
  ("current_data_year", "string"),
  ("injury_at_work", "string"),
  ("manner_of_death", "string"),
  ("method_of_disposition", "string"),
  ("autopsy", "string"),
  ("activity_code", "string"),
  ("place_of_injury_for_causes_w00_y34_except_y06_and_y07_", "string"),
  ("icd_code_10th_revision", "string"),
  ("358_cause_recode", "string"),
  ("113_cause_recode", "string"),
  ("130_infant_cause_recode", "string"),
  ("39_cause_recode", "string"),
  ("number_of_entity_axis_conditions", "string"),
  ("entity_condition_1", "string"),
  ("entity_condition_2", "string"),
  ("entity_condition_3", "string"),
  ("entity_condition_4", "string"),
  ("entity_condition_5", "string"),
  ("entity_condition_6", "string"),
  ("entity_condition_7", "string"),
  ("entity_condition_8", "string"),
  ("entity_condition_9", "string"),
  ("entity_condition_10", "string"),
  ("entity_condition_11", "string"),
  ("entity_condition_12", "string"),
  ("entity_condition_13", "string"),
  ("entity_condition_14", "string"),
  ("entity_condition_15", "string"),
  ("entity_condition_16", "string"),
  ("entity_condition_17", "string"),
  ("entity_condition_18", "string"),
  ("entity_condition_19", "string"),
  ("entity_condition_20", "string"),
  ("number_of_record_axis_conditions", "string"),
  ("record_condition_1", "string"),
  ("record_condition_2", "string"),
  ("record_condition_3", "string"),
  ("record_condition_4", "string"),
```

```
  ("record_condition_5", "string"),
  ("record_condition_6", "string"),
  ("record_condition_7", "string"),
  ("record_condition_8", "string"),
  ("record_condition_9", "string"),
  ("record_condition_10", "string"),
  ("record_condition_11", "string"),
  ("record_condition_12", "string"),
  ("record_condition_13", "string"),
  ("record_condition_14", "string"),
  ("record_condition_15", "string"),
  ("record_condition_16", "string"),
  ("record_condition_17", "string"),
  ("record_condition_18", "string"),
  ("record_condition_19", "string"),
  ("record_condition_20", "string"),
  ("race", "string"),
  ("bridged_race_flag", "string"),
  ("race_imputation_flag", "string"),
  ("race_recode_3", "string"),
  ("race_recode_5", "string"),
  ("hispanic_origin", "string"),
  ("hispanic_originrace_recode", "string")
]
```

```
+-----------------+-------+
|current_data_year|  count|
+-----------------+-------+
|             2012|2547864|
|             2014|2631171|
|             2013|2601452|
|             2005|2452506|
|            V89.9|      8|
|             null|   4517|
|             2009|2441219|
|             2006|2430725|
|      N15.8-N15.9|      4|
|             2011|2519842|
|             2008|2476811|
|             2007|2428343|
|            U04)"|      3|
|          V89.9)"|      4|
|             2015|2718198|
|             2010|2472542|
+-----------------+-------+
```

The years before cleaning

```
+-----------------+-------+
|current_data_year|  count|
+-----------------+-------+
|             2012|2547864|
|             2014|2631171|
|             2013|2601452|
|             2005|2452506|
|             2009|2441219|
|             2006|2430725|
|             2011|2519842|
|             2008|2476811|
|             2007|2428343|
|             2015|2718198|
|             2010|2472542|
+-----------------+-------+
```

The years after cleaning

Analytics questions

The questions answered in the `main.ipynb` are as follows:

Getting the top 10 causes of death for each race.

Getting the top 10 causes of death for each gender.

The distributaion of death per weekday.

The distributaion of death per month.

Correlation between cause of death and the season timing.

Correlation between cause of death, engagement and the month.

Cause of death distribution for each year.

These data has been processed in almost the same manner by grouping and

Getting the top 10 causes of death for each race.

This process is done by grouping all causes of death together and each race and then aggregate to get the count of each row. Then sort all the rows and get the top 20 causes of death and save them in a CSV for each race.

```
df_deaths=df.groupBy('race','358_cause_recode','113_cause_recode','130_infant_cau
se_recode','39_cause_recode').count()
for race_id in codes['race'].keys():
  path=f'/content/drive/MyDrive/Colab Notebooks/BigData/Final
project/results/COD_race/race_{race_id}.csv'
  df_deaths.filter(df_deaths.race==race_id)\
           .sort(F.desc("count")).limit(10)\
           .toPandas()\
           .to_csv(path,header=True)
```

Getting the top 10 causes of death for each gender.

This process is done by grouping all causes of death together and each gender and then aggregate to get the count of each row. Then sort all the rows and get the top 20 causes of death and save them in a CSV for each gender.

```
df_deaths_gender=df.groupBy('sex','358_cause_recode','113_cause_recode','130_infa
nt_cause_recode','39_cause_recode').count()
for gender_id in codes['sex'].keys():
  path=f'/content/drive/MyDrive/Colab Notebooks/BigData/Final
project/results/COD_gender/gender_{gender_id}.csv'
  df_deaths_gender.filter(df_deaths_gender.sex==gender_id)\
           .sort(F.desc("count")).limit(20)\
           .toPandas()\
           .to_csv(path,header=True)
```

The distribution of death per weekday

This process is done by grouping all number of deaths happened in a specific week day and count them. The results was so simple that it didn't need saving in a CSV.

```
day_death=df.groupBy('day_of_week_of_death').count().sort(F.desc("count")).limit(
7)
day_death.show()
```

```
+--------------------+-------+
|day_of_week_of_death|  count|
+--------------------+-------+
|                   7|4012103|
|                   6|3996471|
|                   2|3960250|
|                   5|3945930|
|                   4|3939328|
|                   1|3933510|
|                   3|3931885|
+--------------------+-------+
```

Better visualizations to be made in The business document

The distribution of death per month

This process is done by grouping all number of deaths happened in a specific month and count them. The results was so simple that it didn't need saving in a CSV.

```
month_death=df.groupBy('month_of_death').count().sort(F.desc("count")).limit(12)
month_death.show()
```

```
+--------------+-------+
|month_of_death|  count|
+--------------+-------+
|            01|2571615|
|            03|2494294|
|            12|2479590|
|            02|2324679|
|            04|2296119|
|            10|2292374|
|            11|2285476|
|            05|2278865|
|            07|2205194|
|            08|2191365|
|            06|2154291|
|            09|2146811|
+--------------+-------+
```

Better visualizations to be made in The business document.

Correlation between cause of death and the season timing

We split the months of the year to correspond to every season, then group by each cause of death and count them. the final steps are to sort them in descending order to get the to `X` results and then save each season data in a csv file.

```
seasons={"summer":("06","07","08"),
         "fall":("09","10","11"),
         "winter":("11","12","10"),
         "spring":("03","04","05")}

for season in seasons.keys():
  path=f'/content/drive/MyDrive/Colab Notebooks/BigData/Final
project/results/COD_season/season_{season}.csv'
  m1,m2,m3=seasons[season]
  df.filter((df.month_of_death==m1) | (df.month_of_death==m2) |
(df.month_of_death==m3)).groupBy('358_cause_recode','113_cause_recode','130_infan
t_cause_recode','39_cause_recode').count()\
    .sort(F.desc("count")).limit(20)\
    .toPandas()\
    .to_csv(path,header=True)
```

Correlation between cause of death, engagement and the month.

We wanted to see if the activity done in each season differs. However, most of this column was filled with nulls, hence we couldn't get any strong correlations from it.

```
seasons={"summer":("06","07","08"),
         "fall":("09","10","11"),
         "winter":("11","12","10"),
         "spring":("03","04","05")}
for season in seasons.keys():
  path=f'/content/drive/MyDrive/Colab Notebooks/BigData/Final
project/results/COD_activities/season_{season}.csv'
  m1,m2,m3=seasons[season]
  df.filter((df.month_of_death==m1) | (df.month_of_death==m2) |
(df.month_of_death==m3)).groupBy('activity_code','358_cause_recode','113_cause_re
code','130_infant_cause_recode','39_cause_recode').count()\
    .sort(F.desc("count")).limit(20)\
    .toPandas()\
    .to_csv(path,header=True)
```

Cause of death distribution for each year

This Task is different as gets the top `X` causes of death for each year. This is important for capturing the trend of the cause of death across the years.

```
years=
["2005","2006","2007","2008","2009","2010","2011","2012","2013","2014","2015"]
for year in years:
  path=f'/content/drive/MyDrive/Colab Notebooks/BigData/Final
project/results/COD_per_year/year_{year}.csv'
  df.filter(df.current_data_year==year).groupBy('current_data_year','358_cause_re
code','113_cause_recode','130_infant_cause_recode','39_cause_recode').count()\
    .sort(F.desc("count")).limit(20)\
    .toPandas()\
    .to_csv(path,header=True)
```

Machine learning Task

We decided to make trend fitting over the 2015 data give the data of the previous years. we used XGBoost algorithm for prediction.

Pipeline

- Loading and setting up the session
  - Intiating spark
  - Loading the libraries
- Loading and intial cleaning.
  - Loading The dataset from the file path.
  - Exploring the dataset columns and the datatype of each one.
  - Xounting the number of nulls in each column.
  - Filtering the sex and year columns.
- Preparing The training dataset
  - Loading codes and relabling the columns (sex , cause_recode).
  - Casting the values of each column to the proper datatype.
  - Renaming columns.
  - and filtering non relavent causes.
  - Taking a small sample to process on.
  - changing the months into continous numerical values.
- Extracting the features from the dataset
  We extracted many features to help the model know the average values of the same state during the same period in previous years.
  - get the average sex death count for each month.
  - get the average amont of deaths for each month.
  - get the average resident status count for each month.
  - get the average race code count for each month.
  - get the average cause of death count for each month.
- Generate average count and Create time lag features
  - The lag values are 1,6 amd 12 months that are being added to the months column.
  - a left join is made to merge the laged values to the corresponding months.
  - The same process is made for the last 5 features mentioned in the previous point.
- pretraining step
  - fill out the null values that came out from the left join with zeros.
- Modeling and Fine Tuning
  - we used XGBoost model to train on the data and get the prediction

- Final model

Results and evaluation

The data set was too big for the model to train on making it very hard to get train given the narrow time frame (it may be done on the disscussion day).

Any Enhancement and future work

- we could use a big cluster to train on just to save the time needed for
- more complex features can be inferred once we fix the age columns in the data set. It is a challenge as it needs hand crafted coice of each value.