Department of Computer Engineering
College of Engineering
Polytechnic University of the Philippines Sta. Mesa

CMPE 40163: Big Data using PySpark

Data Manipulation of WHO Suicide Statistics

Submitted by:

Merque, John Ric C.
BSCOE 3-2

Submitted to:

EDCEL B. ARTIFICIO

I. **Introduction**

WHO Suicide Statistics

Basic historical (1979-2016) data by country, year and demographic groups

Suicide is a significant public health issue, with approximately 700,000 people dying by suicide each year worldwide according to the World Health Organization (WHO). For each suicide, there are an estimated 20 suicide attempts. It ranks among the leading causes of death, affecting diverse demographics across different regions. Interestingly, suicide rates tend to exhibit considerable differences based on factors such as year, age, gender, and geographic location and these are the perspectives this analysis aims to look at.

The primary objective of this analysis on suicide statistics is to understand the patterns and trends associated with suicidal behavior in the Philippines. This involves identifying high-risk groups through demographic analysis by grouping the data by gender or age-groups, understanding the probable presence of underlying causes by monitoring trends over time, which also seeks to evaluate the effectiveness of the country's public mental health interventions. By analyzing this data, one can gain thorough understanding of the real trajectory of Philippine suicide data and how it develops over time and for whom it turns out to be more prevalent.

Analyzing and manipulating suicide data is crucial for creating evidence-based interventions, developing early warning systems, and reducing stigma associated with mental health issues. Reliable data enables the design of effective prevention strategies tailored to specific at-risk populations. Additionally, it helps improve public awareness and support for mental health initiatives.

Furthermore, global comparisons of suicide data can reveal successful prevention strategies that can be adapted and implemented in various contexts, contributing to the global effort to mitigate this public health crisis.

Reference:
"Suicide." Accessed: May 24, 2024. [Online]. Available: https://www.who.int/health-topics/suicide#tab=tab_1

II. Data Dictionary

Here is the outline of data variables and high-level information about the dataset:

```
Data Dictionary
Number of rows: 43776
Number of columns: 6
Column 'country': 0 null values
Column 'year': 0 null values
Column 'sex': 0 null values
Column 'age': 0 null values
Column 'suicides_no': 2256 null values
Column 'population': 5460 null values
```

```
root
 |-- country: string (nullable = true)
 |-- year: integer (nullable = true)
 |-- sex: string (nullable = true)
 |-- age: string (nullable = true)
 |-- suicides_no: integer (nullable = true)
 |-- population: integer (nullable = true)
```

III. Data Manipulation using an RDD

1. *Header Removal*

A. RDD Functions Used

Transformation: filter()

Actions: first(), count()

I initially checked the first elements of the RDD using the first() function and the number of row elements using count(). Then, using the filter() function I temporarily removed the header to have a uniform data type for all elements and stored it in a variable

B. Screenshots

Before:

```
# RDD Data Manipulation
rdd = sc.textFile("who_suicide_statistics.csv")
print(f"first elements of the rdd: {rdd.first()}")
print(f"number of elements of the rdd: {rdd.count()}")
```

```
first elements of the rdd: country,year,sex,age,suicides_no,population
number of elements of the rdd: 43777
```

After:

```
# Remove header
header = rdd.first()
data_rdd = rdd.filter(lambda row: row != header)
print(f"first elements of the rdd after removing the header: {data_rdd.first()}")
print(f"number of elements of the rdd after removing the header: {data_rdd.count()}")
```

```
first elements of the rdd after removing the header: Albania,1985,female,15-24 years,,277900
number of elements of the rdd after removing the header: 43776
```

2. *Transformation of the RDD to a Dictionary*

A. RDD Functions Used:
Transformation: map(), split()
Actions: take()

Using the take() method, I peeked at the structure of the first 10 elements of the raw imported CSV file and realized it is just a list with comma-separated values and would be hard to extract value in this form. So I made a helper function and leveraged it to transform the data into a dictionary using the map() function.

B. Screenshots

Before:

```
# Using take() method
print(data_rdd.take(10))
```

```
['Albania,1985,female,15-24 years,,277900', 'Albania,1985,female,25-34 years,,246800', 'Albania,1985,female,35-54 yea
rs,,267500', 'Albania,1985,female,5-14 years,,298300', 'Albania,1985,female,55-74 years,,138700', 'Albania,1985,femal
e,75+ years,,34200', 'Albania,1985,male,15-24 years,,301400', 'Albania,1985,male,25-34 years,,264200', 'Albania,1985,
male,35-54 years,,296700', 'Albania,1985,male,5-14 years,,325800']
```

After:

```
# Helper function to parse a CSV row.
def parse_csv(row):
    reader = csv.DictReader(StringIO(row), fieldnames=header.split(','))
    return next(reader)

# Parse each row into a dictionary
parsed_rdd = data_rdd.map(parse_csv)
```

```
# Using take() method after converting to dictionary
print(parsed_rdd.take(10))
```

```
[{'country': 'Albania', 'year': '1985', 'sex': 'female', 'age': '15-24 years', 'suicides_no': '', 'population': '2779
00'}, {'country': 'Albania', 'year': '1985', 'sex': 'female', 'age': '25-34 years', 'suicides_no': '', 'population':
'246800'}, {'country': 'Albania', 'year': '1985', 'sex': 'female', 'age': '35-54 years', 'suicides_no': '', 'populati
on': '267500'}, {'country': 'Albania', 'year': '1985', 'sex': 'female', 'age': '5-14 years', 'suicides_no': '', 'popu
lation': '298300'}, {'country': 'Albania', 'year': '1985', 'sex': 'female', 'age': '55-74 years', 'suicides_no': '',
'population': '138700'}, {'country': 'Albania', 'year': '1985', 'sex': 'female', 'age': '75+ years', 'suicides_no':
'', 'population': '34200'}, {'country': 'Albania', 'year': '1985', 'sex': 'male', 'age': '15-24 years', 'suicides_n
o': '', 'population': '301400'}, {'country': 'Albania', 'year': '1985', 'sex': 'male', 'age': '25-34 years', 'suicide
s_no': '', 'population': '264200'}, {'country': 'Albania', 'year': '1985', 'sex': 'male', 'age': '35-54 years', 'suic
ides_no': '', 'population': '296700'}, {'country': 'Albania', 'year': '1985', 'sex': 'male', 'age': '5-14 years', 'su
icides_no': '', 'population': '325800'}]
```

3. *Filter to Philippine Data Only*

A. RDD Functions Used:
Transformation: filter()
Actions: collect()

Using the collect() method, I looked at existing data and found multiple countries, including the Philippines. I aim for Philippine data so I use filter() to filter suicide data for the Philippines only.

B. Screenshots

Before:

```
# Using collect() method
parsed_rdd.collect()
```

```
[{'country': 'Albania',
  'year': '1985',
  'sex': 'female',
  'age': '15-24 years',
  'suicides_no': '',
  'population': '277900'},
 {'country': 'Albania',
  'year': '1985',
  'sex': 'female',
  'age': '25-34 years',
  'suicides_no': '',
  'population': '246800'},
 {'country': 'Albania',
  'year': '1985',
  'sex': 'female',
  'age': '35-54 years',
  'suicides_no': '',
  'population': '267500'},
```

After:

```
# Filter the dataset for rows where the country is 'Philippines'
philippines_rdd = parsed_rdd.filter(lambda row: row['country'] == 'Philippines')
philippines_rdd.collect()
```

```
[{'country': 'Philippines',
  'year': '1980',
  'sex': 'female',
  'age': '15-24 years',
  'suicides_no': '',
  'population': '5015000'},
 {'country': 'Philippines',
  'year': '1980',
  'sex': 'female',
  'age': '25-34 years',
  'suicides_no': '',
  'population': '3417900'},
 {'country': 'Philippines',
  'year': '1980',
  'sex': 'female',
  'age': '35-54 years',
  'suicides_no': '',
  'population': '3865100'},
 {'country': 'Philippines',
  'year': '1980',
  'sex': 'female',
  'age': '5-14 years',
  'suicides_no': '',
  'population': '6017800'},
 {'country': 'Philippines',
```

4. *Removing Null Values*

A. RDD Functions Used:

Transformation: filter()

Actions: count()

Using the filter() method, I removed null values and used count() for verification

B. Screenshots

Before:

```
# Empty suicides number rows
empty_suicides_no_rdd = philippines_rdd.filter(lambda row: row['suicides_no'] == '')
print(f"empty suicide number rows count: {empty_suicides_no_rdd.count()}")
```

```
empty suicide number rows count: 96
```

After:

```
# Filter rows with no null or empty string values
philippines_rdd = philippines_rdd.filter(lambda d: all(v not in (None, '') for v in d.values()))
empty_suicides_no_rdd = philippines_rdd.filter(lambda row: row['suicides_no'] == '')
print(f"empty suicide number rows count: {empty_suicides_no_rdd.count()}")
```

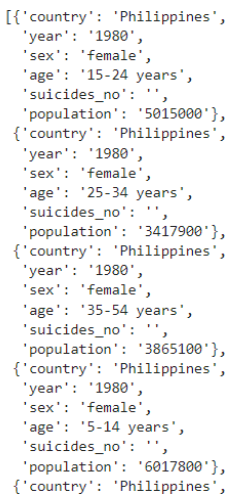
```
empty suicide number rows count: 0
```

5. *Grouping Suicide Rate Values per Age Group*
   A. RDD Functions Used:
   Transformation: flatMap(), map(), groupBy(), combineByKey()
   Actions: collect()

   Using the groupBy() method, I grouped the data by age column, made a list of values using map method and then used flatMap() to concatenate values. I combined the values using combinedByKey() using the mapped RDD that includes the calculated suicide rate standardized for 100 000 population per age group

   B. Screenshots

Before:



```
[{'country': 'Philippines',
  'year': '1980',
  'sex': 'female',
  'age': '15-24 years',
  'suicides_no': '',
  'population': '5015000'},
 {'country': 'Philippines',
  'year': '1980',
  'sex': 'female',
  'age': '25-34 years',
  'suicides_no': '',
  'population': '3417900'},
 {'country': 'Philippines',
  'year': '1980',
  'sex': 'female',
  'age': '35-54 years',
  'suicides_no': '',
  'population': '3865100'},
 {'country': 'Philippines',
  'year': '1980',
  'sex': 'female',
  'age': '5-14 years',
  'suicides_no': '',
  'population': '6017800'},
 {'country': 'Philippines',
```

After:

```
# Group By
grouped_by_age_rdd = philippines_rdd.groupBy(lambda row: row['age'])
grouped_by_age_rdd = grouped_by_age_rdd.map(lambda x: (x[0], list(x[1])))
flattened_rdd = grouped_by_age_rdd.flatMap(lambda x: [(x[0], entry) for entry in x[1]])

# Compute suicide rates
suicide_rates_rdd = flattened_rdd.map(lambda x: (
    x[0],  # age group
    float(x[1]['suicides_no']) / float(x[1]['population']) * 100000  # suicide rate per 100,000
))

# Aggregate by age group, calculate average suicide rate per age group
# using formula: Suicide rate = Total Number of suicides per Age Group / Total Population per Age Group) × 100,000

aggregated_rdd = suicide_rates_rdd.combineByKey(
    lambda value: (value, 1),
    lambda acc, value: (acc[0] + value, acc[1] + 1),
    lambda acc1, acc2: (acc1[0] + acc2[0], acc1[1] + acc2[1])
).map(lambda x: (x[0], x[1][0] / x[1][1]))

# Collect and print results
results = aggregated_rdd.collect()
for result in results:
    print(f"Age Group: {result[0]}, Average Suicide Rate: {result[1]:.2f}%")
```

```
Age Group: 15-24 years, Average Suicide Rate: 2.84%
Age Group: 5-14 years, Average Suicide Rate: 0.16%
Age Group: 55-74 years, Average Suicide Rate: 2.46%
Age Group: 75+ years, Average Suicide Rate: 4.04%
Age Group: 25-34 years, Average Suicide Rate: 2.78%
Age Group: 35-54 years, Average Suicide Rate: 2.24%
```

6. *Sorting Suicide Rate Values per Age Group*
   A. RDD Functions Used:
   Transformation: sortBy()
   Actions: collect()

   Using the sortBy() method, I ordered the grouped data in descending order to know which age group has higher or lower suicide rate. I found that people aged 75+ from the PH has the most suicide rate from the data set and people aged 5-14 has the least.

B. Screenshots

Before:

```
Age Group: 15-24 years, Average Suicide Rate: 2.84%
Age Group: 5-14 years, Average Suicide Rate: 0.16%
Age Group: 55-74 years, Average Suicide Rate: 2.46%
Age Group: 75+ years, Average Suicide Rate: 4.04%
Age Group: 25-34 years, Average Suicide Rate: 2.78%
Age Group: 35-54 years, Average Suicide Rate: 2.24%
```

After:

```
# Sort by suicide rate in descending order
sorted_rdd = aggregated_rdd.sortBy(lambda x: x[1], ascending=False)
results = sorted_rdd.collect()
for result in results:
    print(f"Age Group: {result[0]}, Average Suicide Rate: {result[1]:.2f}%")
```

```
Age Group: 75+ years, Average Suicide Rate: 4.04%
Age Group: 15-24 years, Average Suicide Rate: 2.84%
Age Group: 25-34 years, Average Suicide Rate: 2.78%
Age Group: 55-74 years, Average Suicide Rate: 2.46%
Age Group: 35-54 years, Average Suicide Rate: 2.24%
Age Group: 5-14 years, Average Suicide Rate: 0.16%
```

C. Reflection about data manipulation using an RDD:

While working with the data manipulation using RDD, I learned that this way, data has to be carefully cleaned, or else deriving insights from it would be challenging. In case of null values, and headers that are of different data types, these had to be manually and programmatically treated in RDDs as it does not have shorthand commands for it.

Moreover, manipulating Data as RDD can be insightful, from the chunk of data presented before transformation and after, it almost seems magical how connections and trends show themselves. This manipulation showed how 75+ aged Filipino people tend to commit suicide the most.

Nevertheless, in my experience, as powerful as RDDs are, they can be complex to manage and lack some advanced optimizations. However, it can still deliver insights but with more explicit coding.

IV. **Data Manipulation using a Dataframe**

1. *Removing Null Values*
   A. Functions Used: dropna()

      Using dropna() I removed the elements with null values.

   B. Screenshots

Before:

```
# Show the first few rows of the DataFrame
print("Before dropping nulls:")
df.show()
```

```
Before dropping nulls:
+-------+----+------+-----------+-----------+----------+
|country|year|   sex|        age|suicides_no|population|
+-------+----+------+-----------+-----------+----------+
|Albania|1985|female|15-24 years|       NULL|    277900|
|Albania|1985|female|25-34 years|       NULL|    246800|
|Albania|1985|female|35-54 years|       NULL|    267500|
|Albania|1985|female| 5-14 years|       NULL|    298300|
|Albania|1985|female|55-74 years|       NULL|    138700|
|Albania|1985|female|  75+ years|       NULL|     34200|
|Albania|1985|  male|15-24 years|       NULL|    301400|
|Albania|1985|  male|25-34 years|       NULL|    264200|
|Albania|1985|  male|35-54 years|       NULL|    296700|
|Albania|1985|  male| 5-14 years|       NULL|    325800|
|Albania|1985|  male|55-74 years|       NULL|    132500|
|Albania|1985|  male|  75+ years|       NULL|     21100|
|Albania|1986|female|15-24 years|       NULL|    283900|
|Albania|1986|female|25-34 years|       NULL|    252100|
|Albania|1986|female|35-54 years|       NULL|    273200|
|Albania|1986|female| 5-14 years|       NULL|    304700|
|Albania|1986|female|55-74 years|       NULL|    141700|
|Albania|1986|female|  75+ years|       NULL|     34900|
|Albania|1986|  male|15-24 years|       NULL|    306700|
|Albania|1986|  male|25-34 years|       NULL|    269000|
+-------+----+------+-----------+-----------+----------+
only showing top 20 rows
```

After:

```
# Drop rows where any of the specified columns have null val
columns_to_check = df.columns
cleaned_df = df.dropna(subset=columns_to_check)
print("After dropping nulls:")
cleaned_df.show()
```

```
After dropping nulls:
+-------+----+------+-----------+-----------+----------+
|country|year|   sex|        age|suicides_no|population|
+-------+----+------+-----------+-----------+----------+
|Albania|1987|female|15-24 years|         14|    289700|
|Albania|1987|female|25-34 years|          4|    257200|
|Albania|1987|female|35-54 years|          6|    278800|
|Albania|1987|female| 5-14 years|          0|    311000|
|Albania|1987|female|55-74 years|          0|    144600|
|Albania|1987|female|  75+ years|          1|     35600|
|Albania|1987|  male|15-24 years|         21|    312900|
|Albania|1987|  male|25-34 years|          9|    274300|
|Albania|1987|  male|35-54 years|         16|    308000|
|Albania|1987|  male| 5-14 years|          0|    338200|
|Albania|1987|  male|55-74 years|          1|    137500|
|Albania|1987|  male|  75+ years|          1|     21800|
|Albania|1988|female|15-24 years|          8|    295600|
|Albania|1988|female|25-34 years|          5|    262400|
|Albania|1988|female|35-54 years|          4|    284500|
|Albania|1988|female| 5-14 years|          0|    317200|
|Albania|1988|female|55-74 years|          3|    147500|
|Albania|1988|female|  75+ years|          2|     36400|
|Albania|1988|  male|15-24 years|         17|    319200|
|Albania|1988|  male|25-34 years|          5|    279900|
+-------+----+------+-----------+-----------+----------+
only showing top 20 rows
```

2. *Filter for Philippine Data*
   A. Functions Used: filter()

      Using filter() method I filtered the Suicide Data for Philippines only.

B. Screenshots

Before:

```
# Show the first few rows of the DataFrame before filtering
print("Before filtering:")
cleaned_df.show()
```

```
Before filtering:
+-------+----+------+-----------+-----------+----------+
|country|year|   sex|        age|suicides_no|population|
+-------+----+------+-----------+-----------+----------+
|Albania|1987|female|15-24 years|         14|    289700|
|Albania|1987|female|25-34 years|          4|    257200|
|Albania|1987|female|35-54 years|          6|    278800|
|Albania|1987|female| 5-14 years|          0|    311000|
|Albania|1987|female|55-74 years|          0|    144600|
|Albania|1987|female|  75+ years|          1|     35600|
|Albania|1987|  male|15-24 years|         21|    312900|
|Albania|1987|  male|25-34 years|          9|    274300|
|Albania|1987|  male|35-54 years|         16|    308000|
|Albania|1987|  male| 5-14 years|          0|    338200|
|Albania|1987|  male|55-74 years|          1|    137500|
|Albania|1987|  male|  75+ years|          1|     21800|
|Albania|1988|female|15-24 years|          8|    295600|
|Albania|1988|female|25-34 years|          5|    262400|
|Albania|1988|female|35-54 years|          4|    284500|
|Albania|1988|female| 5-14 years|          0|    317200|
|Albania|1988|female|55-74 years|          3|    147500|
|Albania|1988|female|  75+ years|          2|     36400|
|Albania|1988|  male|15-24 years|         17|    319200|
|Albania|1988|  male|25-34 years|          5|    279900|
+-------+----+------+-----------+-----------+----------+
only showing top 20 rows
```

After:

```
# Filter
# Filter the DataFrame to only include rows where the country is "Philippines"
filtered_df = cleaned_df.dropDuplicates().filter(df['country'] == 'Philippines')
print("After filtering for Philippines:")
filtered_df.show()
```

```
After filtering for Philippines:
+-----------+----+------+-----------+-----------+----------+
|    country|year|   sex|        age|suicides_no|population|
+-----------+----+------+-----------+-----------+----------+
|Philippines|1993|female|  75+ years|         10|    395945|
|Philippines|2006|female| 5-14 years|         28|  10121349|
|Philippines|2002|female|15-24 years|        114|   7903007|
|Philippines|1992|female|  75+ years|          2|    385751|
|Philippines|2008|  male|55-74 years|        144|   3349085|
|Philippines|1993|female|35-54 years|         54|   5754827|
|Philippines|1993|  male| 5-14 years|          0|   8673303|
|Philippines|1999|  male|25-34 years|        199|   5840463|
|Philippines|2009|  male|25-34 years|        383|   7144648|
|Philippines|2002|  male|25-34 years|        267|   6171400|
|Philippines|2000|  male|35-54 years|        294|   7449254|
|Philippines|2002|  male|55-74 years|         93|   2525030|
|Philippines|2010|  male|25-34 years|        400|   7256791|
|Philippines|2002|female|55-74 years|         27|   2885746|
|Philippines|2002|  male| 5-14 years|         22|  10117670|
|Philippines|2003|  male| 5-14 years|         21|  10247489|
|Philippines|2006|female|25-34 years|         91|   6566402|
|Philippines|1999|female|35-54 years|         57|   7196381|
|Philippines|2011|  male| 5-14 years|         44|  10788943|
|Philippines|1998|female|55-74 years|         23|   2539996|
+-----------+----+------+-----------+-----------+----------+
only showing top 20 rows
```

3. Adding Suicide Rate Columns

A. Functions Used: withColumn()

Using withColumn() function I added a column that calculates suicide rate from suicide number and population.

B. Screenshots

Before:

```
+-----------+----+------+-----------+-----------+----------+
|    country|year|   sex|        age|suicides_no|population|
+-----------+----+------+-----------+-----------+----------+
|Philippines|1993|female|  75+ years|         10|    395945|
|Philippines|2006|female| 5-14 years|         28|  10121349|
|Philippines|2002|female|15-24 years|        114|   7903007|
|Philippines|1992|female|  75+ years|          2|    385751|
|Philippines|2008|  male|55-74 years|        144|   3349085|
|Philippines|1993|female|35-54 years|         54|   5754827|
|Philippines|1993|  male| 5-14 years|          0|   8673303|
|Philippines|1999|  male|25-34 years|        199|   5840463|
|Philippines|2009|  male|25-34 years|        383|   7144648|
|Philippines|2002|  male|25-34 years|        267|   6171400|
|Philippines|2000|  male|35-54 years|        294|   7449254|
|Philippines|2002|  male|55-74 years|         93|   2525030|
|Philippines|2010|  male|25-34 years|        400|   7256791|
|Philippines|2002|female|55-74 years|         27|   2885746|
|Philippines|2002|  male| 5-14 years|         22|  10117670|
|Philippines|2003|  male| 5-14 years|         21|  10247489|
|Philippines|2006|female|25-34 years|         91|   6566402|
|Philippines|1999|female|35-54 years|         57|   7196381|
|Philippines|2011|  male| 5-14 years|         44|  10788943|
|Philippines|1998|female|55-74 years|         23|   2539996|
+-----------+----+------+-----------+-----------+----------+
only showing top 20 rows
```

After:

```
# Add column called suicide rate
# using formula: Suicide rate = ( suicides number / population ) × 100,000

filtered_df = filtered_df.withColumn("suicide_rate", (col("suicides_no") / col("population")) * 100000 )
filtered_df = filtered_df.withColumn("suicide_rate", format_number("suicide_rate", 2))
filtered_df.show()
```

```
+-----------+----+------+-----------+-----------+----------+------------+
|    country|year|   sex|        age|suicides_no|population|suicide_rate|
+-----------+----+------+-----------+-----------+----------+------------+
|Philippines|1993|female|  75+ years|         10|    395945|        2.53|
|Philippines|2006|female| 5-14 years|         28|  10121349|        0.28|
|Philippines|2002|female|15-24 years|        114|   7903007|        1.44|
|Philippines|1992|female|  75+ years|          2|    385751|        0.52|
|Philippines|2008|  male|55-74 years|        144|   3349085|        4.30|
|Philippines|1993|female|35-54 years|         54|   5754827|        0.94|
|Philippines|1993|  male| 5-14 years|          0|   8673303|        0.00|
|Philippines|1999|  male|25-34 years|        199|   5840463|        3.41|
|Philippines|2009|  male|25-34 years|        383|   7144648|        5.36|
|Philippines|2002|  male|25-34 years|        267|   6171400|        4.33|
|Philippines|2000|  male|35-54 years|        294|   7449254|        3.95|
|Philippines|2002|  male|55-74 years|         93|   2525030|        3.68|
|Philippines|2010|  male|25-34 years|        400|   7256791|        5.51|
|Philippines|2002|female|55-74 years|         27|   2885746|        0.94|
|Philippines|2002|  male| 5-14 years|         22|  10117670|        0.22|
|Philippines|2003|  male| 5-14 years|         21|  10247489|        0.20|
|Philippines|2006|female|25-34 years|         91|   6566402|        1.39|
|Philippines|1999|female|35-54 years|         57|   7196381|        0.79|
|Philippines|2011|  male| 5-14 years|         44|  10788943|        0.41|
|Philippines|1998|female|55-74 years|         23|   2539996|        0.91|
+-----------+----+------+-----------+-----------+----------+------------+
only showing top 20 rows
```

4. *Using Select Distinct*

A. Functions Used: select(), distinct()

Using select() and distinct() functions I looked for the distinct age groups the data set has.

B. Screenshots

```
# Select Distinct Age Groups

select_age_groups_df = filtered_df.select("age").distinct()
select_age_groups_df.show()
```

```
+-----------+
|        age|
+-----------+
|55-74 years|
|25-34 years|
| 5-14 years|
|  75+ years|
|15-24 years|
|35-54 years|
+-----------+
```

5. *Grouping by Age and Ordering it by Average Suicide Rate and Total Suicide Number*

A. Functions Used: groupBy(), orderBy()

Using groupBy() I grouped the data set by age and ordered it descending using orderBy(). I first used total suicide number as a parameter then average suicide rate and see the difference.

B. Screenshots

Before:

```
# Group By Age

# Cast the 'suicides_no' column to integers, 'suicide_rate' column to float (or double)
filtered_df = filtered_df.withColumn('suicides_no', col('suicides_no').cast('integer'))
filtered_df = filtered_df.withColumn('suicide_rate', col('suicide_rate').cast('double'))

# Group by the 'age' column, sum the 'suicides_no' column, and get the average of the 'suicide_rate' column
grouped_by_age_df = filtered_df.groupBy('age').agg(
    sum('suicides_no').alias('total_suicides_no'),
    round(avg('suicide_rate'), 2).alias('average_suicide_rate')
)

grouped_by_age_df.show()
```

```
+-----------+-----------------+--------------------+
|        age|total_suicides_no|average_suicide_rate|
+-----------+-----------------+--------------------+
|55-74 years|             2168|                2.46|
|25-34 years|             5358|                2.78|
| 5-14 years|              493|                0.16|
|  75+ years|              488|                4.04|
|15-24 years|             7107|                2.84|
|35-54 years|             5716|                2.24|
+-----------+-----------------+--------------------+
```

After:

```
# Order By "average_suicide_rate" descending
grouped_by_age_df = grouped_by_age_df.orderBy("average_suicide_rate", ascending=False)
print("Ordered age groups by average suicide rate per 100 000 population")
grouped_by_age_df.show()
```

```
Ordered age groups by average suicide rate per 100 000 population
+-----------+-----------------+--------------------+
|        age|total_suicides_no|average_suicide_rate|
+-----------+-----------------+--------------------+
|  75+ years|              488|                4.04|
|15-24 years|             7107|                2.84|
|25-34 years|             5358|                2.78|
|55-74 years|             2168|                2.46|
|35-54 years|             5716|                2.24|
| 5-14 years|              493|                0.16|
+-----------+-----------------+--------------------+
```

```
# Order By "total_suicides_no" descending
grouped_by_age_df = grouped_by_age_df.orderBy("total_suicides_no", ascending=False)
print("Ordered age groups by most number of suicides")
grouped_by_age_df.show()
```

```
Ordered age groups by most number of suicides
+-----------+-----------------+--------------------+
|        age|total_suicides_no|average_suicide_rate|
+-----------+-----------------+--------------------+
|15-24 years|             7107|                2.84|
|35-54 years|             5716|                2.24|
|25-34 years|             5358|                2.78|
|55-74 years|             2168|                2.46|
| 5-14 years|              493|                0.16|
|  75+ years|              488|                4.04|
+-----------+-----------------+--------------------+
```

6. *Grouping by Sex and Ordering it by Average Suicide Rate*

A. Functions Used: groupBy(), orderBy()

Using groupBy() I grouped the data set by sex and ordered it descending using orderBy().

B. Screenshots

Before:

```
# Group By sex

# Group by the 'sex' column, sum the 'suicides_no' column, and get the average of the 'suicide_rate' column
grouped_by_sex_df = filtered_df.groupBy('sex').agg(
    sum('suicides_no').alias('total_suicides_no'),
    round(avg('suicide_rate'), 2).alias('average_suicide_rate')
)

# Show the grouped DataFrame
grouped_by_sex_df.show()
```

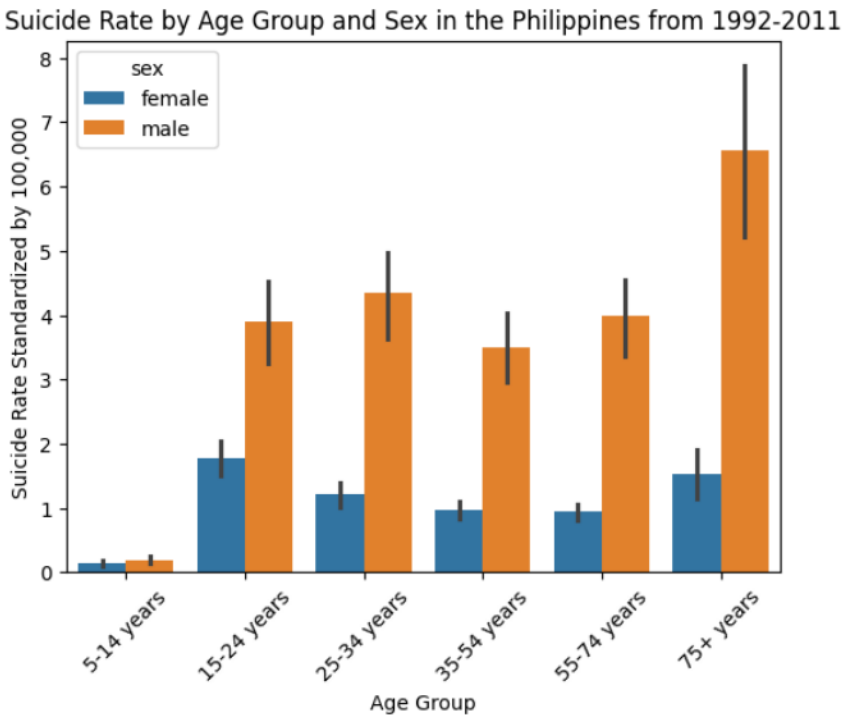
```
+------+-----------------+--------------------+
|   sex|total_suicides_no|average_suicide_rate|
+------+-----------------+--------------------+
|female|             5273|                1.09|
|  male|            16057|                3.75|
+------+-----------------+--------------------+
```

After:

```
# Order By "average_suicide_rate" descending
grouped_by_sex_df = grouped_by_sex_df.orderBy(col('average_suicide_rate').desc())
grouped_by_sex_df.show()
```

```
+------+-----------------+--------------------+
|   sex|total_suicides_no|average_suicide_rate|
+------+-----------------+--------------------+
|  male|            16057|                3.75|
|female|             5273|                1.09|
+------+-----------------+--------------------+
```

Visualization using Seaborn Bar Plot:



7. *Grouping by Year and Ordering it by Ascending Years*

A. Functions Used: orderBy(), groupBy()

Using groupBy() I grouped the data set by year and ordered it descending using orderBy().

B. Screenshots

Before:

```
# Group By year

# Group by the 'year' column, sum the 'suicides_no' column, and get the average of the 'suicide_rate' column
grouped_by_year_df = filtered_df.groupBy('year').agg(
    sum('suicides_no').alias('total_suicides_no'),
    round(avg('suicide_rate'), 2).alias('average_suicide_rate')
)

# Show the grouped DataFrame
grouped_by_year_df.show()
```

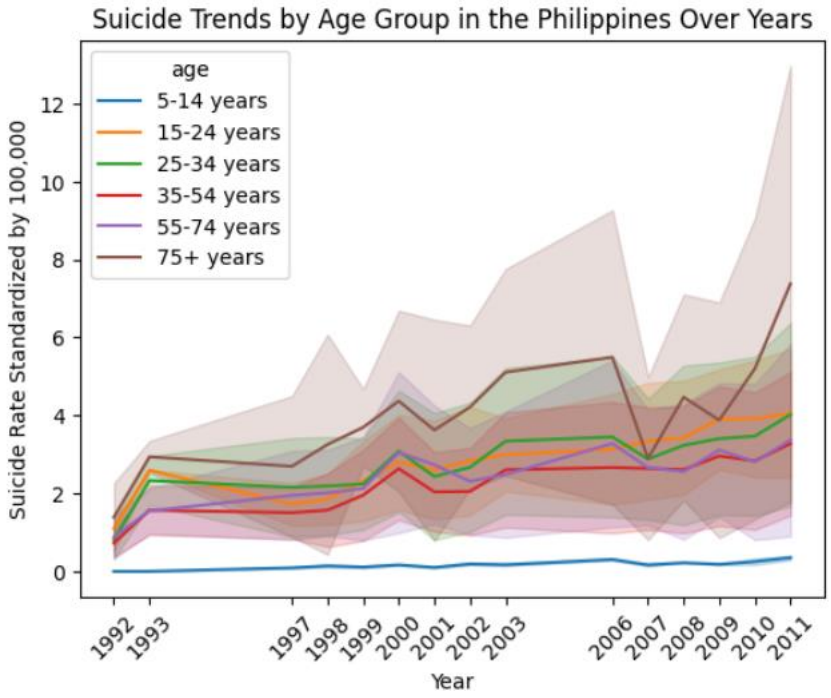
```
+----+-----------------+--------------------+
|year|total_suicides_no|average_suicide_rate|
+----+-----------------+--------------------+
|2003|             1544|                2.78|
|2007|             1698|                2.43|
|2006|             1788|                3.05|
|1997|              814|                1.69|
|1998|              883|                1.84|
|2009|             2074|                2.91|
|2001|             1204|                2.26|
|1992|              347|                0.82|
|2000|             1403|                2.68|
|2010|             2116|                3.08|
|2011|             2449|                3.74|
|2008|             1820|                2.75|
|1999|             1043|                2.07|
|2002|             1301|                2.38|
|1993|              846|                1.83|
+----+-----------------+--------------------+
```

After:

```
# Order By year ascending
grouped_by_year_df = grouped_by_year_df.orderBy(col('year').desc())
grouped_by_year_df.show()
```

```
+----+-----------------+--------------------+
|year|total_suicides_no|average_suicide_rate|
+----+-----------------+--------------------+
|2011|             2449|                3.74|
|2010|             2116|                3.08|
|2009|             2074|                2.91|
|2008|             1820|                2.75|
|2007|             1698|                2.43|
|2006|             1788|                3.05|
|2003|             1544|                2.78|
|2002|             1301|                2.38|
|2001|             1204|                2.26|
|2000|             1403|                2.68|
|1999|             1043|                2.07|
|1998|              883|                1.84|
|1997|              814|                1.69|
|1993|              846|                1.83|
|1992|              347|                0.82|
+----+-----------------+--------------------+
```

Visualization using Seaborn Line Plot



In retrospect, since I chose suicide data with different demographic variables, I thought of several questions related to the relationships between such demographics and the number of suicide cases as well as the suicide rate. These includes the trajectory of suicide data over the years in the Philippines, the age group most susceptible to suicide or if the two biological sexes have extreme differences in their suicide values. As such, the following questions were formulated:

Questions:
According to the WHO Suicide Statistics (with constraints for unavailable data):
1. What age group has the highest suicide rate in the Philippines from 1992-2011?
2. Are Filipino men more likely to commit suicide than Filipino women?
3. In what year from 1992-2011 has the highest recorded number of suicides in the Philippines? lowest?
4. In what year from 1992-2011 has the highest calculated suicide rate in the Philippines? lowest?

SQL queries:

1. What age group has the highest suicide rate in the Philippines from 1992-2011?

```
# SQL Queries

# 1. What age group has the highest suicide rate in the Philippines throughout the years?

filtered_df.createOrReplaceTempView("temp_view1")
query_result = spark.sql("""
    SELECT age, ROUND(AVG(suicide_rate), 2) AS avg_suicide_rate
    FROM temp_view1
    WHERE country = 'Philippines'
    GROUP BY age
    ORDER BY avg_suicide_rate DESC
    LIMIT 1
""")
query_result.show()
```

```
+---------+----------------+
|      age|avg_suicide_rate|
+---------+----------------+
|75+ years|            4.04|
+---------+----------------+
```

Answer: According to the WHO Suicide Statistics, Filipinos aged 75+ has the highest suicide rate in the Philippines from 1992-2011 with rating as high as 4.04% (rating standardized to 100 000 population).

2. Are Filipino men more likely to commit suicide than Filipino women?

```
# 2. Are Filipino men more likely to commit suicide than Filipino women?

filtered_df.createOrReplaceTempView("temp_view2")
query_result = spark.sql("""
    SELECT sex, ROUND(AVG(suicide_rate), 2) AS avg_suicide_rate
    FROM temp_view2
    WHERE country = 'Philippines'
    GROUP BY sex
    ORDER BY avg_suicide_rate DESC
""")

# Show the result
query_result.show()
```

```
+------+----------------+
|   sex|avg_suicide_rate|
+------+----------------+
|  male|            3.75|
|female|            1.09|
+------+----------------+
```

Answer: According to the WHO Suicide Statistics, Filipino men are approximately three to four times more likely to commit suicide with 3.75% average suicide rate from 1992-2011 compared to Filipino women with 1.09%.

3. In what year from 1992-2011 has the highest recorded number of suicides in the Philippines? lowest?

```
# 3. In what year from 1992-2011 has the highest recorded number of suicide in the Philippines? lowest? ( with constr

# For the highest recorded number of suicides
filtered_df.createOrReplaceTempView("temp_view3")
query_result_highest = spark.sql("""
    SELECT year, SUM(suicides_no) AS total_suicides
    FROM temp_view3
    WHERE country = 'Philippines' AND year BETWEEN 1992 AND 2011
    GROUP BY year
    ORDER BY total_suicides DESC
    LIMIT 1
""")

print("The highest recorded number of suicide in the Philippines from 1992-2011")
query_result_highest.show()

# For the lowest recorded number of suicides
filtered_df.createOrReplaceTempView("temp_view4")
query_result_highest = spark.sql("""
    SELECT year, SUM(suicides_no) AS total_suicides
    FROM temp_view4
    WHERE country = 'Philippines' AND year BETWEEN 1992 AND 2011
    GROUP BY year
    ORDER BY total_suicides
    LIMIT 1
""")

print("The lowest recorded number of suicide in the Philippines from 1992-2011")
query_result_highest.show()
```

```
The highest recorded number of suicide in the Philippines from 1992-2011
+----+--------------+
|year|total_suicides|
+----+--------------+
|2011|          2449|
+----+--------------+

The lowest recorded number of suicide in the Philippines from 1992-2011
+----+--------------+
|year|total_suicides|
+----+--------------+
|1992|           347|
+----+--------------+
```

Answer: According to the WHO Suicide Statistics, the highest recorded number of suicides in the Philippines from 1992-2011 is from year 2011 with 2449 total cases. The lowest in comparison is 1992 with 347 cases of suicide.

It might seem that as we progress throughout the years, suicide number is increasing and it's easy to say people are more probable to suicidal behaviors, however it might be due to the increasing population so we must check suicide rate as well.

4. In what year from 1992-2011 has the highest calculated suicide rate in the Philippines? lowest?

```
# 4. In what year from 1992-2011 has the highest measured suicide rate in the Philippines? lowest? (with constraints ;

# For the highest recorded suicide rate
query_result_highest = spark.sql("""
    SELECT year, AVG(suicide_rate) AS avg_suicide_rate
    FROM temp_view3
    WHERE country = 'Philippines' AND year BETWEEN 1992 AND 2011
    GROUP BY year
    ORDER BY avg_suicide_rate DESC
    LIMIT 1
""")

print("The year with the highest recorded suicide rate in the Philippines from 1992-2011:")
query_result_highest.show()

# For the lowest recorded suicide rate
query_result_lowest = spark.sql("""
    SELECT year, AVG(suicide_rate) AS avg_suicide_rate
    FROM temp_view4
    WHERE country = 'Philippines' AND year BETWEEN 1992 AND 2011
    GROUP BY year
    ORDER BY avg_suicide_rate ASC
    LIMIT 1
""")

print("The year with the lowest recorded suicide rate in the Philippines from 1992-2011:")
query_result_lowest.show()
```

```
The year with the highest recorded suicide rate in the Philippines from 1992-2011:
+----+----------------+
|year|avg_suicide_rate|
+----+----------------+
|2011|          3.7425|
+----+----------------+

The year with the lowest recorded suicide rate in the Philippines from 1992-2011:
+----+----------------+
|year|avg_suicide_rate|
+----+----------------+
|1992|          0.8175|
+----+----------------+
```

Answer: According to the WHO Suicide Statistics, the highest calculated suicide rate in the Philippines from 1992-2011 is from year 2011 with 3.74%. The lowest in comparison is in year 1992 with 0.82% rate of suicide.

This is parallel with the results of the previous query, but now using a probability value that also considers the population, hence it is safe to assume that chances of suicidal behavior among Filipinos indeed tends to increase throughout the years from 1992-2011.

C. Reflection about manipulating data from a dataframe using SQL

Using Data Frames is a lot more convenient than RDDs as you no longer have to programmatically treat headers and null values to clean the data before extracting value. There are already shorthand commands to handle them.

The tabular structure of data also calls for lesser effort to organize data and is intuitive. It does not need to be transformed into Python dictionaries. Generally, code is easier to write for manipulating the data.

Moreover, SQL engines like Spark SQL made it easy to extract information and derive insights from the data set. SQL is a very familiar language to learn and comprehend, it is not complexly structured and the syntax is human-readable so it makes it easier to learn and uncover hidden bits of information.

Furthermore, data becomes more contextual when queried using SQL and is isolated from the big chunk of data that is hard to decipher on its own. Using this tool feels like a voyage of uncovering some hidden treasure chest of untapped knowledge.

Generally, I had an insightful time uncovering trends and answering questions I had in mind about the nuances and trajectories of suicide behaviors in the Philippines. I have always wondered about this topic and some of my questions were gracefully answered.

V. **Synthesis and Moving Forward**

A. What are the things that you learned about Big Data?

Big Data can be overwhelming on its own. With lots of information to digest it is not just possible to scroll through every record and find insights. That is why I learned to see how much tools such as RDDs and Data frames SQL are crucial for uncovering untapped knowledge hidden from the chunks of data that is big enough that by itself, does not make any meaningful sense. The nature of manipulating the data is so we can transform it into something that could give us value, and in my case discovering the suicide trends in the Philippines throughout the years.

Technically, it also occurred to me how null values from unavailable data can constrict insights and affect its reliability. In this analysis, I only had data from 1992-2011. It is unfortunate because I would want to it to extend into the present to tap into world events like pandemic and generally the virtualization of the social landscape and how it affects mental health and thus suicidal behaviors. But then this is the only data available and I realized that data analysis is only as valuable as the data we can extract.

Moreover, it is also important to look out for the biases in assuming trends. In my case the number of suicides vs suicide rate. It is easy to assume number of suicides for measuring how likely suicidal behavior is, but a confounding variable for it is the population. Bigger population means bigger numbers but it does not necessarily mean bigger probability. So, I had to use suicide rate for most part, which is not readily available so I learned to calculate it from the available columns and standardized it based on the conventional population value.

Lastly, I learned that (Big) Data can be reduced to a more digestible data and thus, valuable data, one that has clear insights, and hence, more value. I also learned that doing such, while overwhelming, can be a lot of fun.

B. How do you plan to use the things you learned moving forward?

I will take these learnings to heart. The technical ones I will try hard to remember, but I don't think I will forget how fun it is to extract insights from a data set. So, I will use this memory to motivate myself into learning more about Big Data, explore more areas that this world has to offer and really look on how I can be able provide value by doing so.

C. What are your areas for improvement?

First is having more out-of-the-box explorations. Exploring data from other countries and comparing how it differs to the Philippines. It could be used to relate to GDP, level of technology or any other data sets that may show correlations with suicide data.

It is also an area to improve on code refactoring and building more efficient code with less need for processing power.

I could also use more contextual insights from real world scenarios, like what major world events happened in what year that may have caused the spike or downfall in the suicide data of a particular country, and many more.