New York University

CS-GY 6513-D : BIG DATA (Prof. Amit Patel)

Fall 2023

GHW5: Spark Assignment

Yogya Sharma (ys5250)

Using the survey.csv uploaded under the /shared/ folder in JupyterHub (and also attached with assignment) , **answer the following questions using both Spark dataframe and SparkSQL**. Eliminate rows that have non-integer 'Value' column value.

Importing libraries, creating a spark session, eliminating rows that have non-integer 'Value' column value. Moreover, normalizing the Value column to Dollars, and removing the rows with "Percentage"as an entry in the Units column.

In [7]:
```
from pyspark.sql import SparkSession
from pyspark.sql.functions import col, avg, when, count, isnan, stddev, sum, regexp_replace, mean
from pyspark.sql.types import IntegerType


# Create a Spark session
spark = SparkSession.builder.appName("SurveyAnalysis").getOrCreate()
```

In [8]:
```
# Load the CSV file into a Spark DataFrame
file_path = "shared/survey.csv"
df = spark.read.csv(file_path, header=True, inferSchema=True)

df = df.filter(col("Value").cast("int").isNotNull())

# Normalizing values to Dollars

# Multiply 'Value' by a million for rows with 'Dollars (millions)' in 'Units'
df = df.withColumn(
    "Value",
    when(col("Units") == "Dollars (millions)", col("Value") * 1_000_000)
    .otherwise(col("Value"))
)

# Filter rows where 'Units' is not 'Percentage'
df = df.filter(col("Units") != "Percentage")

# Register the DataFrame as a temporary table for SparkSQL queries
df.createOrReplaceTempView("survey_table")
```

1. **Question 1**:

   a. Count unique industry names.

Spark DF:

```
unique_industries_df = df.select("Industry_name_NZSIOC").distinct().count()
print(f"Count of unique industry names using DataFrame: {unique_industries_df}")
```

SparkSQL:

```
unique_industries_sql = spark.sql("""
   SELECT COUNT(DISTINCT `Industry_name_NZSIOC`) as UniqueIndustries
   FROM survey_table
""")
unique_industries_sql.show()
```

   b. Calculate total income by industry and year.

Spark DF:

```
total_income_by_industry_df = df.groupBy("Industry_name_NZSIOC",
"Year").agg(sum("Value").alias("TotalIncomeDollars")).orderBy("Industry_name_N
ZSIOC", "Year")
total_income_by_industry_df.show()
```

In [6]:
```
# Spark Dataframe
total_income_by_industry_df = df.groupBy("Industry_name_NZSIOC", "Year") \
    .agg(sum("Value").alias("TotalIncomeDollars")) \
    .orderBy("Industry_name_NZSIOC", "Year")

total_income_by_industry_df.show()
```

```
[Stage 24:============================>                            (1 + 1) / 2]
```

```
+--------------------+----+-----------------+
|Industry_name_NZSIOC|Year|TotalIncomeDollars|
+--------------------+----+-----------------+
|       Accommodation|2013|           3.199E9|
|       Accommodation|2014|           2.811E9|
|       Accommodation|2015|           3.159E9|
|       Accommodation|2016|           3.637E9|
|       Accommodation|2017|           3.825E9|
|       Accommodation|2018|           4.228E9|
|       Accommodation|2019|           2.619E9|
|       Accommodation|2020|           2.708E9|
|       Accommodation|2021|           3.806E9|
|Accommodation and...|2013|           2.786E9|
|Accommodation and...|2014|           3.329E9|
|Accommodation and...|2015|           2.751E9|
|Accommodation and...|2016|           2.924E9|
|Accommodation and...|2017|           2.391E9|
|Accommodation and...|2018|           2.613E9|
|Accommodation and...|2019|           2.765E9|
|Accommodation and...|2020|           3.142E9|
|Accommodation and...|2021|           2.133E9|
|Administrative an...|2013|           2.116E9|
|Administrative an...|2014|           2.455E9|
+--------------------+----+-----------------+
only showing top 20 rows
```

SparkSQL:

```
total_income_by_industry_sql = spark.sql("""
    SELECT
        Industry_name_NZSIOC,
        Year,
        SUM(Value) AS TotalIncomeDollars
    FROM survey_table
    GROUP BY Industry_name_NZSIOC, Year
    ORDER BY Industry_name_NZSIOC, Year
""")
total_income_by_industry_sql.show()
```

In [7]:
```
# SparkSQL
total_income_by_industry_sql = spark.sql("""
    SELECT
        Industry_name_NZSIOC,
        Year,
        SUM(Value) AS TotalIncomeDollars
    FROM survey_table
    GROUP BY Industry_name_NZSIOC, Year
    ORDER BY Industry_name_NZSIOC, Year
""")
total_income_by_industry_sql.show()
```

```
+--------------------+----+------------------+
|Industry_name_NZSIOC|Year|TotalIncomeDollars|
+--------------------+----+------------------+
|       Accommodation|2013|           3.199E9|
|       Accommodation|2014|           2.811E9|
|       Accommodation|2015|           3.159E9|
|       Accommodation|2016|           3.637E9|
|       Accommodation|2017|           3.825E9|
|       Accommodation|2018|           4.228E9|
|       Accommodation|2019|           2.619E9|
|       Accommodation|2020|           2.708E9|
|       Accommodation|2021|           3.806E9|
|Accommodation and...|2013|           2.786E9|
|Accommodation and...|2014|           3.329E9|
|Accommodation and...|2015|           2.751E9|
|Accommodation and...|2016|           2.924E9|
|Accommodation and...|2017|           2.391E9|
|Accommodation and...|2018|           2.613E9|
|Accommodation and...|2019|           2.765E9|
|Accommodation and...|2020|           3.142E9|
|Accommodation and...|2021|           2.133E9|
|Administrative an...|2013|           2.116E9|
|Administrative an...|2014|           2.455E9|
```

c. Show the top 5 industries by total income in 2021.

Spark DF:

```
top5_industries_2021 = df.filter(col("Year") == 2021) \
    .groupBy("Industry_name_NZSIOC").agg({"Value": "sum"}) \
    .orderBy("sum(Value)", ascending=False).limit(5)

top5_industries_2021.show()
```

In [52]:
```
# Spark Dataframe
top5_industries_2021 = df.filter(col("Year") == 2021) \
    .groupBy("Industry_name_NZSIOC").agg({"Value": "sum"}) \
    .orderBy("sum(Value)", ascending=False).limit(5)

# Show the result
top5_industries_2021.show()
```

```
+--------------------+----------+
|Industry_name_NZSIOC|sum(Value)|
+--------------------+----------+
|Public Order, Saf...| 1.5834E10|
|Polymer Product a...| 1.2458E10|
|            Printing| 1.0972E10|
|Fabricated Metal ...| 1.0208E10|
|Non-Metallic Mine...| 1.0014E10|
+--------------------+----------+
```

SparkSQL:

```
top5_industries_2021_sql = spark.sql("""
```

```
    SELECT Industry_name_NZSIOC, SUM(Value) AS TotalIncome
    FROM survey_table
    WHERE Year = 2021
    GROUP BY Industry_name_NZSIOC
    ORDER BY TotalIncome DESC
    LIMIT 5
""")
top5_industries_2021_sql.show()
```

In [25]:
```
# SparkSQL
top5_industries_2021_sql = spark.sql("""
    SELECT Industry_name_NZSIOC, SUM(Value) AS TotalIncome
    FROM survey_table
    WHERE Year = 2021
    GROUP BY Industry_name_NZSIOC
    ORDER BY TotalIncome DESC
    LIMIT 5
""")
top5_industries_2021_sql.show()
```

```
+--------------------+-----------+
|Industry_name_NZSIOC|TotalIncome|
+--------------------+-----------+
|Public Order, Saf...|  1.5834E10|
|Polymer Product a...|  1.2458E10|
|            Printing|  1.0972E10|
|Fabricated Metal ...|  1.0208E10|
|Non-Metallic Mine...|  1.0014E10|
+--------------------+-----------+
```

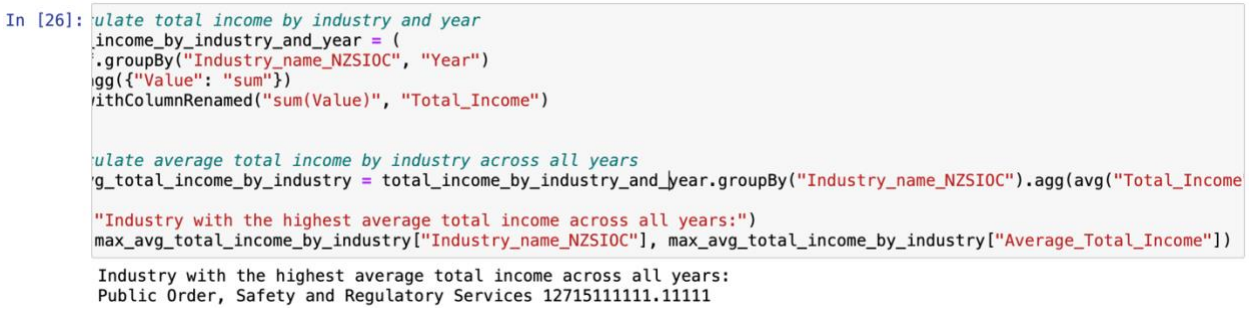
2. **Question 2**: Find the industry with the highest average total income across all years.

Spark DF:

```
# Calculate total income by industry and year
total_income_by_industry_and_year = (
    df.groupBy("Industry_name_NZSIOC", "Year")
    .agg({"Value": "sum"})
    .withColumnRenamed("sum(Value)", "Total_Income")
)

# Calculate average total income by industry across all years
max_avg_total_income_by_industry =
total_income_by_industry_and_year.groupBy("Industry_name_NZSIOC").agg(avg("Total_In
come").alias("Average_Total_Income")).orderBy(col("Average_Total_Income").desc()).first(
)

print("Industry with the highest average total income across all years:")
print(max_avg_total_income_by_industry["Industry_name_NZSIOC"],
max_avg_total_income_by_industry["Average_Total_Income"])
```



SparkSQL:

```
# Calculate total income by industry and year
spark.sql("""
    CREATE OR REPLACE TEMPORARY VIEW total_income_by_industry_and_year_view
AS
    SELECT Industry_name_NZSIOC, Year, SUM(Value) AS Total_Income
    FROM survey_table
    GROUP BY Industry_name_NZSIOC, Year
""")
```

```
# Calculate average total income by industry across all years
spark.sql("""
    CREATE OR REPLACE TEMPORARY VIEW avg_total_income_by_industry_view AS
    SELECT Industry_name_NZSIOC, AVG(Total_Income) AS Average_Total_Income
    FROM total_income_by_industry_and_year_view
    GROUP BY Industry_name_NZSIOC
""")

# Find the industry with the highest average total income
top_avg_total_income_industry = spark.sql("""
    SELECT Industry_name_NZSIOC, Average_Total_Income
    FROM avg_total_income_by_industry_view
    ORDER BY Average_Total_Income DESC
    LIMIT 1
""")

top_avg_total_income_industry.show()
```

In [11]:

```
# Spark SQL
# Calculate total income by industry and year
spark.sql("""
    CREATE OR REPLACE TEMPORARY VIEW total_income_by_industry_and_year_view AS
    SELECT Industry_name_NZSIOC, Year, SUM(Value) AS Total_Income
    FROM survey_table
    GROUP BY Industry_name_NZSIOC, Year
""")

# Calculate average total income by industry across all years
spark.sql("""
    CREATE OR REPLACE TEMPORARY VIEW avg_total_income_by_industry_view AS
    SELECT Industry_name_NZSIOC, AVG(Total_Income) AS Average_Total_Income
    FROM total_income_by_industry_and_year_view
    GROUP BY Industry_name_NZSIOC
""")

# Find the industry with the highest average total income
top_avg_total_income_industry = spark.sql("""
    SELECT Industry_name_NZSIOC, Average_Total_Income
    FROM avg_total_income_by_industry_view
    ORDER BY Average_Total_Income DESC
    LIMIT 1
""")

top_avg_total_income_industry.show()
```

```
+--------------------+--------------------+
|Industry_name_NZSIOC|Average_Total_Income|
+--------------------+--------------------+
|Public Order, Saf...|1.271511111111111E10|
+--------------------+--------------------+
```

3. **Question 3**: Determine the industry that had the highest total income in any single year and provide the year in which it occurred.

Spark DF:

```
total_income_by_industry_year_df = df.groupBy("Industry_name_NZSIOC", "Year").agg({"Value": "sum"}).withColumnRenamed("sum(Value)", "TotalIncome").orderBy("TotalIncome", ascending = False).limit(1)

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

```
In [12]: # Spark DataFrame
total_income_by_industry_year_df = df.groupBy("Industry_name_NZSIOC", "Year").agg({"Value": "sum"}).withColumnRename

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

```
+--------------------+----+-----------+
|Industry_name_NZSIOC|Year|TotalIncome|
+--------------------+----+-----------+
|Public Order, Saf...|2021|  1.5834E10|
+--------------------+----+-----------+
```

SparkSQL:

```
total_income_by_industry_year_sql = spark.sql("""
   SELECT Industry_name_NZSIOC, Year, SUM(Value) AS TotalIncome
   FROM survey_table
   GROUP BY Industry_name_NZSIOC, Year
   ORDER BY TotalIncome DESC
   LIMIT 1
""")

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

```
In [13]: # Spark SQL
total_income_by_industry_year_sql = spark.sql("""
    SELECT Industry_name_NZSIOC, Year, SUM(Value) AS TotalIncome
    FROM survey_table
    GROUP BY Industry_name_NZSIOC, Year
    ORDER BY TotalIncome DESC
    LIMIT 1
""")

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

```
+--------------------+----+-----------+
|Industry_name_NZSIOC|Year|TotalIncome|
+--------------------+----+-----------+
|Public Order, Saf...|2021|  1.5834E10|
+--------------------+----+-----------+
```

4. **Question 4**:
    a. Calculate the median total income for each industry in 2021.

Spark DF:

```
# Spark Dataframe
from pyspark.sql.functions import col, median

# Filter data for the year 2021
survey_2021_df = df.filter(col("Year") == 2021)

# Calculate the median total income for each industry in 2021
median_income_by_industry_df = (
    survey_2021_df
    .groupBy("Industry_name_NZSIOC")
    .agg(median("Value").alias("Median_Income"))
    .orderBy("Median_Income", ascending=False)
)

# Display the result
median_income_by_industry_df.show()
```

In [13]:
```
# Spark Dataframe
from pyspark.sql.functions import col, median

# Filter data for the year 2021
survey_2021_df = df.filter(col("Year") == 2021)

# Calculate the median total income for each industry in 2021
median_income_by_industry_df = (
    survey_2021_df
    .groupBy("Industry_name_NZSIOC")
    .agg(median("Value").alias("Median_Income"))
    .orderBy("Median_Income", ascending=False)
)

# Display the result
median_income_by_industry_df.show()
```

```
[Stage 34:============================>                            (1 + 1) / 2]
```

```
+--------------------+-------------+
|Industry_name_NZSIOC|Median_Income|
+--------------------+-------------+
|Professional, Sci...|       7.13E8|
|Arts, Recreation ...|       5.72E8|
|Professional, Sci...|      5.285E8|
|     Wholesale Trade|      5.095E8|
|Transport, Postal...|       4.57E8|
|Advertising, Mark...|      4.275E8|
|      Other Services|       4.21E8|
|        Construction|      4.105E8|
|Retail Trade and ...|       3.76E8|
|Electricity, Gas,...|       3.61E8|
|Electricity and G...|      3.275E8|
|Other Store-Based...|       3.16E8|
|Motor Vehicle and...|       3.13E8|
|              Mining|       2.99E8|
|Arts and Recreati...|       2.87E8|
|       Manufacturing|       2.81E8|
```

SparkSQL:

```
# SparkSQL

# Filter data for the year 2021 using SparkSQL
survey_2021_sql = spark.sql("""
   SELECT *
   FROM survey_table
   WHERE Year = 2021
""")

# Create a temporary view for the filtered data
survey_2021_sql.createOrReplaceTempView("survey_2021_table")

# Calculate the median total income for each industry in 2021 using SparkSQL
median_income_by_industry_sql = spark.sql("""
   SELECT Industry_name_NZSIOC, percentile_approx(cast(Value as double), 0.5)
as Median_Income
   FROM survey_2021_table
   GROUP BY Industry_name_NZSIOC
   ORDER BY Median_Income DESC
""")

# Display the result
median_income_by_industry_sql.show()
```

In [17]:
```
# SparkSQL

# Filter data for the year 2021 using SparkSQL
survey_2021_sql = spark.sql("""
    SELECT *
    FROM survey_table
    WHERE Year = 2021
""")

# Create a temporary view for the filtered data
survey_2021_sql.createOrReplaceTempView("survey_2021_table")

# Calculate the median total income for each industry in 2021 using SparkSQL
median_income_by_industry_sql = spark.sql("""
    SELECT Industry_name_NZSIOC, percentile_approx(cast(Value as double), 0.5) as Median_Income
    FROM survey_2021_table
    GROUP BY Industry_name_NZSIOC
    ORDER BY Median_Income DESC
""")

# Display the result
median_income_by_industry_sql.show()
```

```
+--------------------+-------------+
|Industry_name_NZSIOC|Median_Income|
+--------------------+-------------+
|Professional, Sci...|       7.13E8|
|Arts, Recreation ...|       5.56E8|
|     Wholesale Trade|       4.86E8|
|      Other Services|       3.92E8|
|Advertising, Mark...|       3.72E8|
|Electricity, Gas,...|       3.61E8|
|Retail Trade and ...|       3.36E8|
|        Construction|       3.33E8|
|Electricity and G...|       3.25E8|
|Other Store-Based...|       3.16E8|
|              Mining|       2.99E8|
```

b. Determine the industry that had the highest coefficient of variation (standard deviation divided by mean) in total income from 2020 to 2021.

Spark DF:

```
from pyspark.sql import functions as F
# Filter data for the years 2020 and 2021
df_filtered = df.filter((col("Year") == 2020) | (col("Year") == 2021))

# Calculate total income for each industry in each year
df_total_income = df_filtered.groupBy("Industry_name_NZSIOC",
"Year").agg(F.sum("Value").alias("TotalIncome")).orderBy("Industry_name_NZSIO
C")

# Calculate mean and standard deviation for each industry
df_stats =
df_total_income.groupBy("Industry_name_NZSIOC").agg(F.mean("TotalIncome").al
ias("Mean"), F.stddev("TotalIncome").alias("StdDev"))

# Calculate coefficient of variation
df_cv = df_stats.withColumn("CoefficientOfVariation", col("StdDev") / col("Mean"))
```

```
# Find the industry with the highest coefficient of variation
industry_highest_cv = df_cv.orderBy(col("CoefficientOfVariation").desc()).first()

# Display the result
print("Industry with the highest coefficient of variation:")
print(industry_highest_cv)
```

In [25]:
```
# Spark Dataframe

from pyspark.sql import functions as F
# Filter data for the years 2020 and 2021
df_filtered = df.filter((col("Year") == 2020) | (col("Year") == 2021))

# Calculate total income for each industry in each year
df_total_income = df_filtered.groupBy("Industry_name_NZSIOC", "Year").agg(F.sum("Value").alias("TotalIncome")).order

# Calculate mean and standard deviation for each industry
df_stats = df_total_income.groupBy("Industry_name_NZSIOC").agg(F.mean("TotalIncome").alias("Mean"), F.stddev("TotalI

# Calculate coefficient of variation
df_cv = df_stats.withColumn("CoefficientOfVariation", col("StdDev") / col("Mean"))

# Find the industry with the highest coefficient of variation
industry_highest_cv = df_cv.orderBy(col("CoefficientOfVariation").desc()).first()

# Display the result
print("Industry with the highest coefficient of variation:")
print(industry_highest_cv)
```

```
Industry with the highest coefficient of variation:
Row(Industry_name_NZSIOC='Petroleum and Coal Product Manufacturing', Mean=1073000000.0, StdDev=1480681599.8046305,
CoefficientOfVariation=1.3799455729772885)
```

SparkSQL:

```
sql_query = """
   SELECT Industry_name_NZSIOC,
        AVG(TotalIncome) AS Mean,
        STD(TotalIncome) AS StdDev,
        STD(TotalIncome) / AVG(TotalIncome) AS CoefficientOfVariation
   FROM (
      SELECT Industry_name_NZSIOC, Year, SUM(Value) AS TotalIncome
      FROM survey_table
      WHERE Year IN (2020, 2021)
      GROUP BY Industry_name_NZSIOC, Year
   )
   GROUP BY Industry_name_NZSIOC
   ORDER BY CoefficientOfVariation DESC
   LIMIT 1
"""

# Run the Spark SQL query
result_sql = spark.sql(sql_query)
```

# Display the result
print("Industry with the highest coefficient of variation (SparkSQL):")
result_sql.show()

In [29]:
```
sql_query = """
    SELECT Industry_name_NZSIOC,
           AVG(TotalIncome) AS Mean,
           STD(TotalIncome) AS StdDev,
           STD(TotalIncome) / AVG(TotalIncome) AS CoefficientOfVariation
    FROM (
        SELECT Industry_name_NZSIOC, Year, SUM(Value) AS TotalIncome
        FROM survey_table
        WHERE Year IN (2020, 2021)
        GROUP BY Industry_name_NZSIOC, Year
    )
    GROUP BY Industry_name_NZSIOC
    ORDER BY CoefficientOfVariation DESC
    LIMIT 1
"""

# Run the Spark SQL query
result_sql = spark.sql(sql_query)

# Display the result
print("Industry with the highest coefficient of variation (SparkSQL):")
result_sql.show()
```

```
Industry with the highest coefficient of variation (SparkSQL):
+--------------------+-------+--------------------+----------------------+
|Industry_name_NZSIOC|   Mean|              StdDev|CoefficientOfVariation|
+--------------------+-------+--------------------+----------------------+
|Petroleum and Coa...|1.073E9|1.4806815998046305E9|    1.3799455729772885|
+--------------------+-------+--------------------+----------------------+
```

5. **Question 5**: Find the industry with the highest year-over-year growth in total income from 2020 to 2021.

Spark DF:

```
# Calculate total income for each industry in 2020 and 2021
total_income_2020 = df.filter(col("Year") ==
2020).groupBy("Industry_name_NZSIOC").agg({"Value":
"sum"}).withColumnRenamed("sum(Value)", "TotalIncome_2020")

total_income_2021 = df.filter(col("Year") ==
2021).groupBy("Industry_name_NZSIOC").agg({"Value":
"sum"}).withColumnRenamed("sum(Value)", "TotalIncome_2021")

# Join the DataFrames on industry name
income_comparison_df = total_income_2020.join(total_income_2021,
"Industry_name_NZSIOC", "inner")

# Calculate year-over-year growth percentage
income_comparison_df = income_comparison_df.withColumn(
    "YoYGrowthPercentage",
    ((col("TotalIncome_2021") - col("TotalIncome_2020")) / col("TotalIncome_2020")) * 100
)

# Find the industry with the highest year-over-year growth
max_growth_industry =
income_comparison_df.orderBy(col("YoYGrowthPercentage").desc()).first()

# Display the result
print("Industry with the highest year-over-year growth from 2020 to 2021:")
print(f"Industry Name: {max_growth_industry['Industry_name_NZSIOC']}")
print(f"Year-over-Year Growth Percentage:
{max_growth_industry['YoYGrowthPercentage']:.2f}%")
```

In [28]:
```
# Calculate total income for each industry in 2020 and 2021
total_income_2020 = df.filter(col("Year") == 2020).groupBy("Industry_name_NZSIOC").agg({"Value": "sum"}).withColumnR

total_income_2021 = df.filter(col("Year") == 2021).groupBy("Industry_name_NZSIOC").agg({"Value": "sum"}).withColumnR

# Join the DataFrames on industry name
income_comparison_df = total_income_2020.join(total_income_2021, "Industry_name_NZSIOC", "inner")

# Calculate year-over-year growth percentage
income_comparison_df = income_comparison_df.withColumn(
    "YoYGrowthPercentage",
    ((col("TotalIncome_2021") - col("TotalIncome_2020")) / col("TotalIncome_2020")) * 100
)

# Find the industry with the highest year-over-year growth
max_growth_industry = income_comparison_df.orderBy(col("YoYGrowthPercentage").desc()).first()

# Display the result
print("Industry with the highest year-over-year growth from 2020 to 2021:")
print(f"Industry Name: {max_growth_industry['Industry_name_NZSIOC']}")
print(f"Year-over-Year Growth Percentage: {max_growth_industry['YoYGrowthPercentage']:.2f}%")
```

```
Industry with the highest year-over-year growth from 2020 to 2021:
Industry Name: Forestry and Logging
Year-over-Year Growth Percentage: 51.52%
```

SparkSQL:

```
# Create a temporary view for the survey_table
spark.sql("CREATE OR REPLACE TEMPORARY VIEW survey_table_view AS SELECT *
FROM survey_table")

# SparkSQL query to calculate total income for each industry in 2020 and 2021
total_income_sql = spark.sql("""
    SELECT
        Industry_name_NZSIOC,
        SUM(CASE WHEN Year = 2020 THEN Value END) AS TotalIncome_2020,
        SUM(CASE WHEN Year = 2021 THEN Value END) AS TotalIncome_2021
    FROM survey_table_view
    WHERE Year = 2020 OR Year = 2021
    GROUP BY Industry_name_NZSIOC
""")

# Calculate year-over-year growth percentage using SparkSQL
growth_percentage_sql = spark.sql("""
    SELECT
        Industry_name_NZSIOC,
        TotalIncome_2020,
        TotalIncome_2021,
        ((TotalIncome_2021 - TotalIncome_2020) / TotalIncome_2020) * 100 AS
YoYGrowthPercentage
    FROM (
        SELECT
```

```
        Industry_name_NZSIOC,
        SUM(CASE WHEN Year = 2020 THEN Value END) AS TotalIncome_2020,
        SUM(CASE WHEN Year = 2021 THEN Value END) AS TotalIncome_2021
      FROM survey_table_view
      WHERE Year = 2020 OR Year = 2021
      GROUP BY Industry_name_NZSIOC
   ) AS total_income_subquery
""")

# Create a temporary view for the growth_percentage_sql
growth_percentage_sql.createOrReplaceTempView("growth_percentage_sql")

# Find the industry with the highest year-over-year growth using SparkSQL
max_growth_industry_sql = spark.sql("""
   SELECT *
   FROM (
      SELECT
         Industry_name_NZSIOC,
         YoYGrowthPercentage,
         ROW_NUMBER() OVER (ORDER BY YoYGrowthPercentage DESC) as row_num
      FROM growth_percentage_sql
   ) ranked
   WHERE row_num = 1
""").first()

# Display the result
print("Industry with the highest year-over-year growth from 2020 to 2021 (SparkSQL):")
print(f"Industry Name: {max_growth_industry_sql['Industry_name_NZSIOC']}")
print(f"Year-over-Year Growth Percentage:
{max_growth_industry_sql['YoYGrowthPercentage']:.2f}%")
```

In [6]:

```
# Create a temporary view for the survey_table
spark.sql("CREATE OR REPLACE TEMPORARY VIEW survey_table_view AS SELECT * FROM survey_table")

# SparkSQL query to calculate total income for each industry in 2020 and 2021
total_income_sql = spark.sql("""
    SELECT
        Industry_name_NZSIOC,
        SUM(CASE WHEN Year = 2020 THEN Value END) AS TotalIncome_2020,
        SUM(CASE WHEN Year = 2021 THEN Value END) AS TotalIncome_2021
    FROM survey_table_view
    WHERE Year = 2020 OR Year = 2021
    GROUP BY Industry_name_NZSIOC
""")

# Calculate year-over-year growth percentage using SparkSQL
growth_percentage_sql = spark.sql("""
    SELECT
        Industry_name_NZSIOC,
        TotalIncome_2020,
        TotalIncome_2021,
        ((TotalIncome_2021 - TotalIncome_2020) / TotalIncome_2020) * 100 AS YoYGrowthPercentage
    FROM (
        SELECT
            Industry_name_NZSIOC,
            SUM(CASE WHEN Year = 2020 THEN Value END) AS TotalIncome_2020,
            SUM(CASE WHEN Year = 2021 THEN Value END) AS TotalIncome_2021
        FROM survey_table_view
        WHERE Year = 2020 OR Year = 2021
        GROUP BY Industry_name_NZSIOC
    ) AS total_income_subquery
""")

# Create a temporary view for the growth_percentage_sql
growth_percentage_sql.createOrReplaceTempView("growth_percentage_sql")
```

```
# Find the industry with the highest year-over-year growth using SparkSQL
max_growth_industry_sql = spark.sql("""
    SELECT *
    FROM (
        SELECT
            Industry_name_NZSIOC,
            YoYGrowthPercentage,
            ROW_NUMBER() OVER (ORDER BY YoYGrowthPercentage DESC) as row_num
        FROM growth_percentage_sql
    ) ranked
    WHERE row_num = 1
""").first()

# Display the result
print("Industry with the highest year-over-year growth from 2020 to 2021 (SparkSQL):")
print(f"Industry Name: {max_growth_industry_sql['Industry_name_NZSIOC']}")
print(f"Year-over-Year Growth Percentage: {max_growth_industry_sql['YoYGrowthPercentage']:.2f}%")
```

```
Industry with the highest year-over-year growth from 2020 to 2021 (SparkSQL):
Industry Name: Forestry and Logging
Year-over-Year Growth Percentage: 51.52%
```

6. **Question 6**: Identify the industry with the most significant change in rank based on total income from 2020 to 2021. The rank is determined by total income in each year.

Spark DF:

```
from pyspark.sql import SparkSession
from pyspark.sql.functions import col, rank, abs
from pyspark.sql.window import Window

# Filter data for the years 2020 and 2021
filtered_df = df.filter((col("Year") == 2020) | (col("Year") == 2021))

# Pivot the DataFrame to get total income for each industry in 2020 and 2021
pivoted_df = filtered_df.groupBy("Industry_name_NZSIOC").pivot("Year").agg({"Value":
"sum"})

# Calculate ranks for each year based on total income
window_spec_2020 = Window.orderBy(col("2020").desc())
window_spec_2021 = Window.orderBy(col("2021").desc())

rank_df = pivoted_df.withColumn("Rank_2020", rank().over(window_spec_2020)) \
                .withColumn("Rank_2021", rank().over(window_spec_2021))

# Calculate the absolute difference in ranks
rank_diff_df = rank_df.withColumn("Rank_Difference", abs(col("Rank_2020") -
col("Rank_2021")))

# Find the industry with the highest absolute difference in ranks
max_rank_diff_industry =
rank_diff_df.orderBy(col("Rank_Difference").desc()).first()["Industry_name_NZSIOC"]

print("Industry with the most significant change in rank based on total income from 2020 to
2021:")
print(f"Industry: {max_rank_diff_industry}")
print(f"Rank in 2020: {rank_df.filter(col('Industry_name_NZSIOC') ==
max_rank_diff_industry).select('Rank_2020').first()['Rank_2020']}")
print(f"Rank in 2021: {rank_df.filter(col('Industry_name_NZSIOC') ==
max_rank_diff_industry).select('Rank_2021').first()['Rank_2021']}")
```

In [58]:
```
from pyspark.sql import SparkSession
from pyspark.sql.functions import col, rank, abs
from pyspark.sql.window import Window

# Filter data for the years 2020 and 2021
filtered_df = df.filter((col("Year") == 2020) | (col("Year") == 2021))

# Pivot the DataFrame to get total income for each industry in 2020 and 2021
pivoted_df = filtered_df.groupBy("Industry_name_NZSIOC").pivot("Year").agg({"Value": "sum"})

# Calculate ranks for each year based on total income
window_spec_2020 = Window.orderBy(col("2020").desc())
window_spec_2021 = Window.orderBy(col("2021").desc())

rank_df = pivoted_df.withColumn("Rank_2020", rank().over(window_spec_2020)) \
                    .withColumn("Rank_2021", rank().over(window_spec_2021))

# Calculate the absolute difference in ranks
rank_diff_df = rank_df.withColumn("Rank_Difference", abs(col("Rank_2020") - col("Rank_2021")))

# Find the industry with the highest absolute difference in ranks
max_rank_diff_industry = rank_diff_df.orderBy(col("Rank_Difference").desc()).first()["Industry_name_NZSIOC"]

print("Industry with the most significant change in rank based on total income from 2020 to 2021:")
print(f"Industry: {max_rank_diff_industry}")
print(f"Rank in 2020: {rank_df.filter(col('Industry_name_NZSIOC') == max_rank_diff_industry).select('Rank_2020').fir
print(f"Rank in 2021: {rank_df.filter(col('Industry_name_NZSIOC') == max_rank_diff_industry).select('Rank_2021').fir
```

```
Industry with the most significant change in rank based on total income from 2020 to 2021:
Industry: Professional, Scientific and Technical Services

Rank in 2020: 19

Rank in 2021: 73
```

SparkSQL:

```
# Filter data for the years 2020 and 2021 using Spark SQL
filtered_df = spark.sql("SELECT * FROM survey_table WHERE Year IN (2020, 2021)")

# Create a temporary view for the filtered DataFrame
filtered_df.createOrReplaceTempView("filtered_table")

# Pivot the DataFrame to get total income for each industry in 2020 and 2021 using Spark SQL
pivoted_df = spark.sql("""
   SELECT
      Industry_name_NZSIOC,
      SUM(CASE WHEN Year = 2020 THEN Value END) AS `2020`,
      SUM(CASE WHEN Year = 2021 THEN Value END) AS `2021`
   FROM filtered_table
   GROUP BY Industry_name_NZSIOC
""")
```

```
# Create a temporary view for the pivoted DataFrame
pivoted_df.createOrReplaceTempView("pivoted_table")
# Calculate ranks for each year based on total income using Spark SQL
rank_df = spark.sql("""
   SELECT
      Industry_name_NZSIOC,
      `2020`,
      `2021`,
      RANK() OVER (ORDER BY `2020` DESC) AS Rank_2020,
      RANK() OVER (ORDER BY `2021` DESC) AS Rank_2021
   FROM pivoted_table
""")

# Create a temporary view for the ranked DataFrame
rank_df.createOrReplaceTempView("ranked_table")

# Calculate the absolute difference in ranks using Spark SQL
rank_diff_df = spark.sql("""
   SELECT
      Industry_name_NZSIOC,
      Rank_2020,
      Rank_2021,
      ABS(Rank_2020 - Rank_2021) AS Rank_Difference
   FROM ranked_table
""")

# Create a temporary view for the rank difference DataFrame
rank_diff_df.createOrReplaceTempView("rank_diff_table")

# Find the industry with the highest absolute difference in ranks using Spark SQL
max_rank_diff_row = spark.sql("""
   SELECT *
   FROM rank_diff_table
   ORDER BY Rank_Difference DESC
   LIMIT 1
""").first()

max_rank_diff_industry = max_rank_diff_row["Industry_name_NZSIOC"]
rank_2020 = max_rank_diff_row["Rank_2020"]
```

rank_2021 = max_rank_diff_row["Rank_2021"]

print("Industry with the most significant change in rank based on total income from 2020 to 2021:")
print(f"Industry: {max_rank_diff_industry}")
print(f"Rank in 2020: {rank_2020}")
print(f"Rank in 2021: {rank_2021}")

In [63]:

```
# Filter data for the years 2020 and 2021 using Spark SQL
filtered_df = spark.sql("SELECT * FROM survey_table WHERE Year IN (2020, 2021)")

# Create a temporary view for the filtered DataFrame
filtered_df.createOrReplaceTempView("filtered_table")

# Pivot the DataFrame to get total income for each industry in 2020 and 2021 using Spark SQL
pivoted_df = spark.sql("""
    SELECT
        Industry_name_NZSIOC,
        SUM(CASE WHEN Year = 2020 THEN Value END) AS `2020`,
        SUM(CASE WHEN Year = 2021 THEN Value END) AS `2021`
    FROM filtered_table
    GROUP BY Industry_name_NZSIOC
""")

# Create a temporary view for the pivoted DataFrame
pivoted_df.createOrReplaceTempView("pivoted_table")
# Calculate ranks for each year based on total income using Spark SQL
rank_df = spark.sql("""
    SELECT
        Industry_name_NZSIOC,
        `2020`,
        `2021`,
        RANK() OVER (ORDER BY `2020` DESC) AS Rank_2020,
        RANK() OVER (ORDER BY `2021` DESC) AS Rank_2021
    FROM pivoted_table
""")

# Create a temporary view for the ranked DataFrame
rank_df.createOrReplaceTempView("ranked_table")
```

```
# Calculate the absolute difference in ranks using Spark SQL
rank_diff_df = spark.sql("""
    SELECT
        Industry_name_NZSIOC,
        Rank_2020,
        Rank_2021,
        ABS(Rank_2020 - Rank_2021) AS Rank_Difference
    FROM ranked_table
""")

# Create a temporary view for the rank difference DataFrame
rank_diff_df.createOrReplaceTempView("rank_diff_table")

# Find the industry with the highest absolute difference in ranks using Spark SQL
max_rank_diff_row = spark.sql("""
    SELECT *
    FROM rank_diff_table
    ORDER BY Rank_Difference DESC
    LIMIT 1
""").first()

max_rank_diff_industry = max_rank_diff_row["Industry_name_NZSIOC"]
rank_2020 = max_rank_diff_row["Rank_2020"]
rank_2021 = max_rank_diff_row["Rank_2021"]

print("Industry with the most significant change in rank based on total income from 2020 to 2021:")
print(f"Industry: {max_rank_diff_industry}")
print(f"Rank in 2020: {rank_2020}")
print(f"Rank in 2021: {rank_2021}")
```

```
Industry with the most significant change in rank based on total income from 2020 to 2021:
Industry: Professional, Scientific and Technical Services
Rank in 2020: 19
Rank in 2021: 73
```

Submit a single PDF document with your code in text (no screenshot of code) and screenshot of the result. DO NOT remove questions, submit your answers underneath each question.