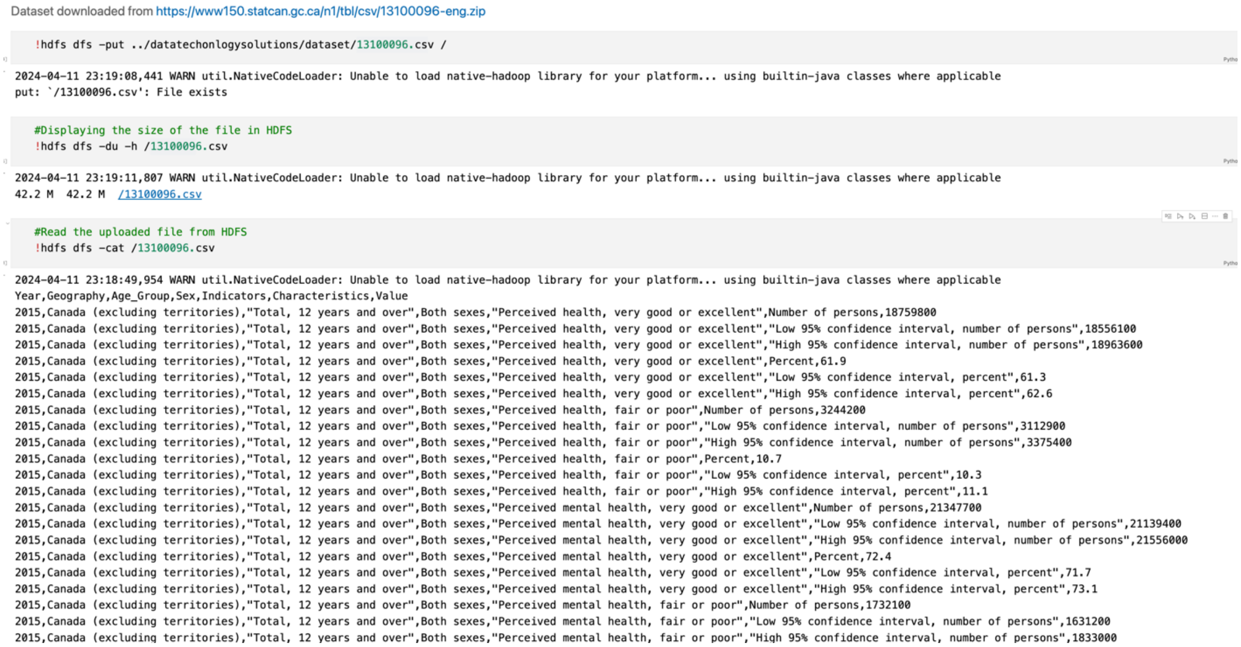
Final Project:

Overview:

Perform a large-scale data analysis using PySpark and Hive. Tailor the analysis based on your interests and the characteristics of the chosen dataset.

Steps:

1. **Data Ingestion:**
    - Obtain a diverse and large dataset. (Use Public data sets)
    - Upload the dataset to HDFS.



Dataset downloaded from https://www150.statcan.gc.ca/n1/tbl/csv/13100096-eng.zip

```
!hdfs dfs -put ../datatechonlogysolutions/dataset/13100096.csv /
```

```
2024-04-11 23:19:08,441 WARN util.NativeCodeLoader: Unable to load native-hadoop library for your platform... using builtin-java classes where applicable
put: `/13100096.csv': File exists
```

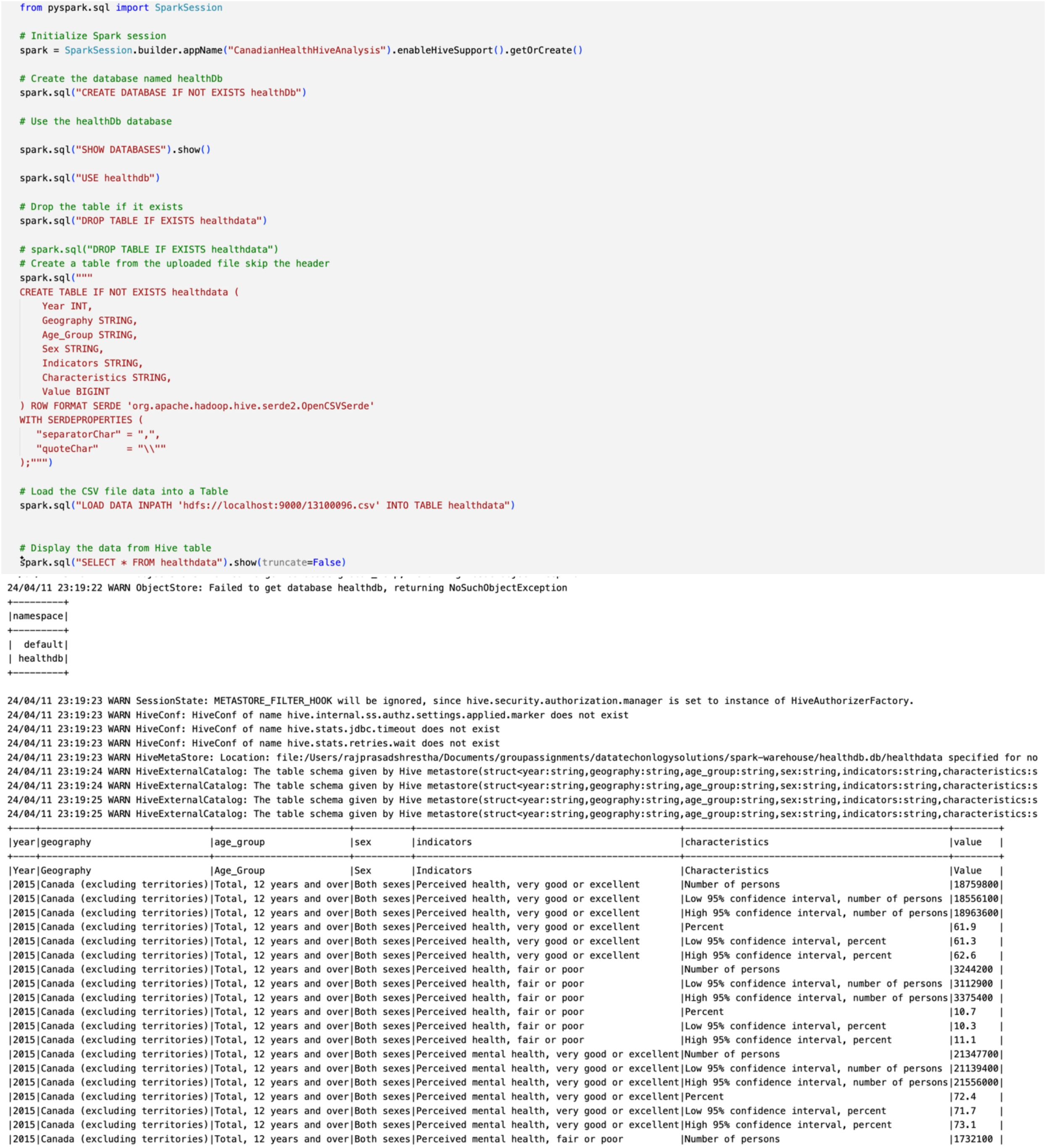
```
#Displaying the size of the file in HDFS
!hdfs dfs -du -h /13100096.csv
```

```
2024-04-11 23:19:11,807 WARN util.NativeCodeLoader: Unable to load native-hadoop library for your platform... using builtin-java classes where applicable
42.2 M  42.2 M  /13100096.csv
```

```
#Read the uploaded file from HDFS
!hdfs dfs -cat /13100096.csv
```

```
2024-04-11 23:18:49,954 WARN util.NativeCodeLoader: Unable to load native-hadoop library for your platform... using builtin-java classes where applicable
Year,Geography,Age_Group,Sex,Indicators,Characteristics,Value
2015,Canada (excluding territories),"Total, 12 years and over",Both sexes,"Perceived health, very good or excellent",Number of persons,18759800
2015,Canada (excluding territories),"Total, 12 years and over",Both sexes,"Perceived health, very good or excellent","Low 95% confidence interval, number of persons",18556100
2015,Canada (excluding territories),"Total, 12 years and over",Both sexes,"Perceived health, very good or excellent","High 95% confidence interval, number of persons",18963600
2015,Canada (excluding territories),"Total, 12 years and over",Both sexes,"Perceived health, very good or excellent",Percent,61.9
2015,Canada (excluding territories),"Total, 12 years and over",Both sexes,"Perceived health, very good or excellent","Low 95% confidence interval, percent",61.3
2015,Canada (excluding territories),"Total, 12 years and over",Both sexes,"Perceived health, very good or excellent","High 95% confidence interval, percent",62.6
2015,Canada (excluding territories),"Total, 12 years and over",Both sexes,"Perceived health, fair or poor",Number of persons,3244200
2015,Canada (excluding territories),"Total, 12 years and over",Both sexes,"Perceived health, fair or poor","Low 95% confidence interval, number of persons",3112900
2015,Canada (excluding territories),"Total, 12 years and over",Both sexes,"Perceived health, fair or poor","High 95% confidence interval, number of persons",3375400
2015,Canada (excluding territories),"Total, 12 years and over",Both sexes,"Perceived health, fair or poor",Percent,10.7
2015,Canada (excluding territories),"Total, 12 years and over",Both sexes,"Perceived health, fair or poor","Low 95% confidence interval, percent",10.3
2015,Canada (excluding territories),"Total, 12 years and over",Both sexes,"Perceived health, fair or poor","High 95% confidence interval, percent",11.1
2015,Canada (excluding territories),"Total, 12 years and over",Both sexes,"Perceived mental health, very good or excellent",Number of persons,21347700
2015,Canada (excluding territories),"Total, 12 years and over",Both sexes,"Perceived mental health, very good or excellent","Low 95% confidence interval, number of persons",21139400
2015,Canada (excluding territories),"Total, 12 years and over",Both sexes,"Perceived mental health, very good or excellent","High 95% confidence interval, number of persons",21556000
2015,Canada (excluding territories),"Total, 12 years and over",Both sexes,"Perceived mental health, very good or excellent",Percent,72.4
2015,Canada (excluding territories),"Total, 12 years and over",Both sexes,"Perceived mental health, very good or excellent","Low 95% confidence interval, percent",71.7
2015,Canada (excluding territories),"Total, 12 years and over",Both sexes,"Perceived mental health, very good or excellent","High 95% confidence interval, percent",73.1
2015,Canada (excluding territories),"Total, 12 years and over",Both sexes,"Perceived mental health, fair or poor",Number of persons,1732100
2015,Canada (excluding territories),"Total, 12 years and over",Both sexes,"Perceived mental health, fair or poor","Low 95% confidence interval, number of persons",1631200
2015,Canada (excluding territories),"Total, 12 years and over",Both sexes,"Perceived mental health, fair or poor","High 95% confidence interval, number of persons",1833000
```

2. **Data Exploration with Hive:**
    - Create a Hive table to read the dataset.
    - Explore the structure of the data using Hive queries.
    - Identify any missing or inconsistent data.

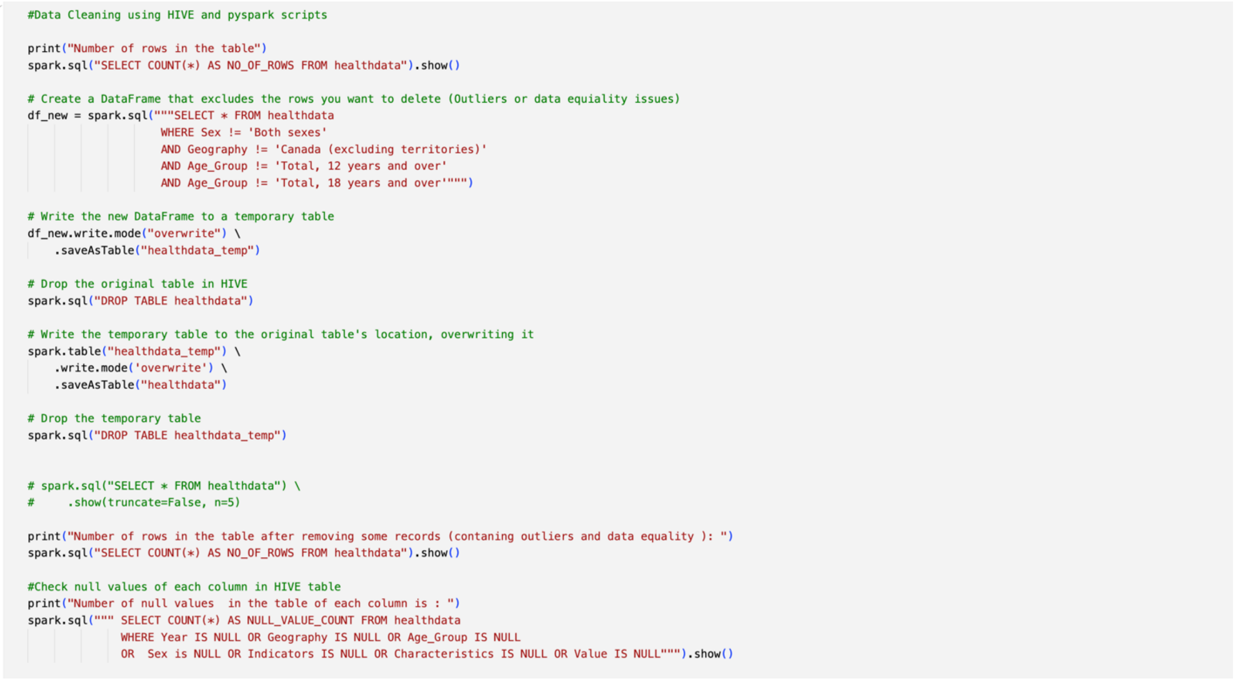


```
from pyspark.sql import SparkSession

# Initialize Spark session
spark = SparkSession.builder.appName("CanadianHealthHiveAnalysis").enableHiveSupport().getOrCreate()

# Create the database named healthDb
spark.sql("CREATE DATABASE IF NOT EXISTS healthDb")

# Use the healthDb database

spark.sql("SHOW DATABASES").show()

spark.sql("USE healthdb")

# Drop the table if it exists
spark.sql("DROP TABLE IF EXISTS healthdata")

# spark.sql("DROP TABLE IF EXISTS healthdata")
# Create a table from the uploaded file skip the header
spark.sql("""
CREATE TABLE IF NOT EXISTS healthdata (
    Year INT,
    Geography STRING,
    Age_Group STRING,
    Sex STRING,
    Indicators STRING,
    Characteristics STRING,
    Value BIGINT
) ROW FORMAT SERDE 'org.apache.hadoop.hive.serde2.OpenCSVSerde'
WITH SERDEPROPERTIES (
   "separatorChar" = ",",
   "quoteChar"     = "\\""
);""")

# Load the CSV file data into a Table
spark.sql("LOAD DATA INPATH 'hdfs://localhost:9000/13100096.csv' INTO TABLE healthdata")


# Display the data from Hive table
spark.sql("SELECT * FROM healthdata").show(truncate=False)
```

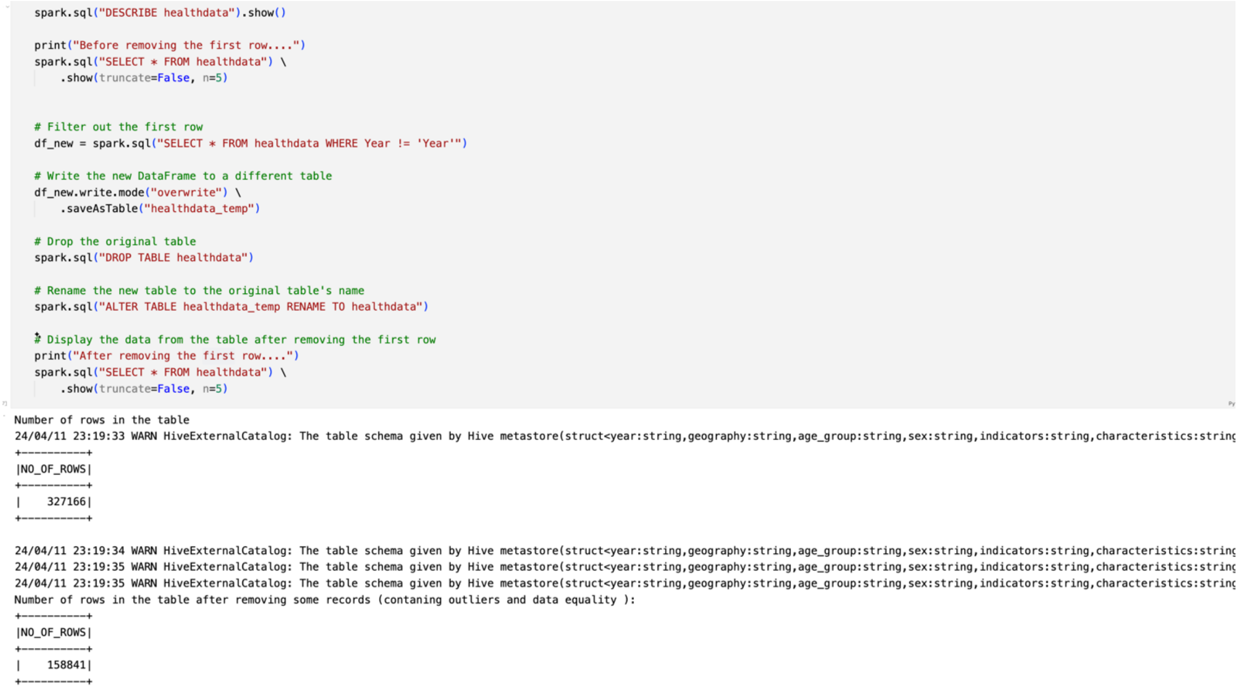
```
24/04/11 23:19:22 WARN ObjectStore: Failed to get database healthdb, returning NoSuchObjectException
+---------+
|namespace|
+---------+
|  default|
| healthdb|
+---------+


24/04/11 23:19:23 WARN SessionState: METASTORE_FILTER_HOOK will be ignored, since hive.security.authorization.manager is set to instance of HiveAuthorizerFactory.
24/04/11 23:19:23 WARN HiveConf: HiveConf of name hive.internal.ss.authz.settings.applied.marker does not exist
24/04/11 23:19:23 WARN HiveConf: HiveConf of name hive.stats.jdbc.timeout does not exist
24/04/11 23:19:23 WARN HiveConf: HiveConf of name hive.stats.retries.wait does not exist
24/04/11 23:19:23 WARN HiveMetaStore: Location: file:/Users/rajprasadshrestha/Documents/groupassignments/datatechonlogysolutions/spark-warehouse/healthdb.db/healthdata specified for no
24/04/11 23:19:24 WARN HiveExternalCatalog: The table schema given by Hive metastore(struct<year:string,geography:string,age_group:string,sex:string,indicators:string,characteristics:s
24/04/11 23:19:24 WARN HiveExternalCatalog: The table schema given by Hive metastore(struct<year:string,geography:string,age_group:string,sex:string,indicators:string,characteristics:s
24/04/11 23:19:25 WARN HiveExternalCatalog: The table schema given by Hive metastore(struct<year:string,geography:string,age_group:string,sex:string,indicators:string,characteristics:s
24/04/11 23:19:25 WARN HiveExternalCatalog: The table schema given by Hive metastore(struct<year:string,geography:string,age_group:string,sex:string,indicators:string,characteristics:s
+----+------------------------------+------------------------+----------+-----------------------------------------------+-----------------------------------------------+--------+
|year|geography                     |age_group               |sex       |indicators                                     |characteristics                                |value   |
+----+------------------------------+------------------------+----------+-----------------------------------------------+-----------------------------------------------+--------+
|Year|Geography                     |Age_Group               |Sex       |Indicators                                     |Characteristics                                |Value   |
|2015|Canada (excluding territories)|Total, 12 years and over|Both sexes|Perceived health, very good or excellent       |Number of persons                              |18759800|
|2015|Canada (excluding territories)|Total, 12 years and over|Both sexes|Perceived health, very good or excellent       |Low 95% confidence interval, number of persons |18556100|
|2015|Canada (excluding territories)|Total, 12 years and over|Both sexes|Perceived health, very good or excellent       |High 95% confidence interval, number of persons|18963600|
|2015|Canada (excluding territories)|Total, 12 years and over|Both sexes|Perceived health, very good or excellent       |Percent                                        |61.9    |
|2015|Canada (excluding territories)|Total, 12 years and over|Both sexes|Perceived health, very good or excellent       |Low 95% confidence interval, percent           |61.3    |
|2015|Canada (excluding territories)|Total, 12 years and over|Both sexes|Perceived health, very good or excellent       |High 95% confidence interval, percent          |62.6    |
|2015|Canada (excluding territories)|Total, 12 years and over|Both sexes|Perceived health, fair or poor                 |Number of persons                              |3244200 |
|2015|Canada (excluding territories)|Total, 12 years and over|Both sexes|Perceived health, fair or poor                 |Low 95% confidence interval, number of persons |3112900 |
|2015|Canada (excluding territories)|Total, 12 years and over|Both sexes|Perceived health, fair or poor                 |High 95% confidence interval, number of persons|3375400 |
|2015|Canada (excluding territories)|Total, 12 years and over|Both sexes|Perceived health, fair or poor                 |Percent                                        |10.7    |
|2015|Canada (excluding territories)|Total, 12 years and over|Both sexes|Perceived health, fair or poor                 |Low 95% confidence interval, percent           |10.3    |
|2015|Canada (excluding territories)|Total, 12 years and over|Both sexes|Perceived health, fair or poor                 |High 95% confidence interval, percent          |11.1    |
|2015|Canada (excluding territories)|Total, 12 years and over|Both sexes|Perceived mental health, very good or excellent|Number of persons                              |21347700|
|2015|Canada (excluding territories)|Total, 12 years and over|Both sexes|Perceived mental health, very good or excellent|Low 95% confidence interval, number of persons |21139400|
|2015|Canada (excluding territories)|Total, 12 years and over|Both sexes|Perceived mental health, very good or excellent|High 95% confidence interval, number of persons|21556000|
|2015|Canada (excluding territories)|Total, 12 years and over|Both sexes|Perceived mental health, very good or excellent|Percent                                        |72.4    |
|2015|Canada (excluding territories)|Total, 12 years and over|Both sexes|Perceived mental health, very good or excellent|Low 95% confidence interval, percent           |71.7    |
|2015|Canada (excluding territories)|Total, 12 years and over|Both sexes|Perceived mental health, very good or excellent|High 95% confidence interval, percent          |73.1    |
|2015|Canada (excluding territories)|Total, 12 years and over|Both sexes|Perceived mental health, fair or poor          |Number of persons                              |1732100 |
```

3.**Data Preprocessing and Cleaning:**

- Use Hive to clean and preprocess the data.
- Handle missing values, outliers, or any data quality issues.

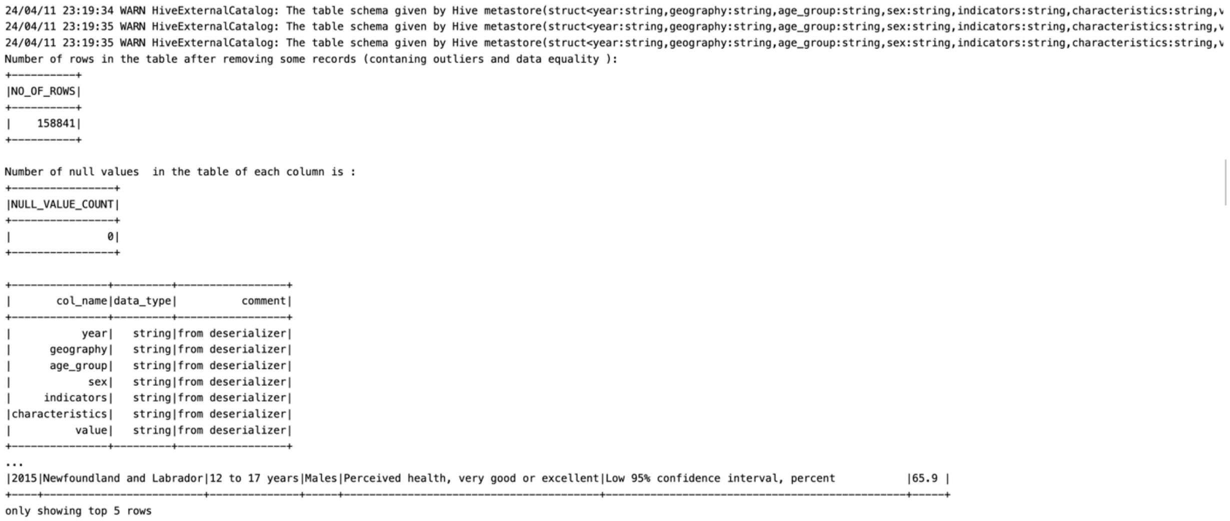


```
#Data Cleaning using HIVE and pyspark scripts

print("Number of rows in the table")
spark.sql("SELECT COUNT(*) AS NO_OF_ROWS FROM healthdata").show()

# Create a DataFrame that excludes the rows you want to delete (Outliers or data equiality issues)
df_new = spark.sql("""SELECT * FROM healthdata
                    WHERE Sex != 'Both sexes'
                    AND Geography != 'Canada (excluding territories)'
                    AND Age_Group != 'Total, 12 years and over'
                    AND Age_Group != 'Total, 18 years and over'""")

# Write the new DataFrame to a temporary table
df_new.write.mode("overwrite") \
    .saveAsTable("healthdata_temp")

# Drop the original table in HIVE
spark.sql("DROP TABLE healthdata")

# Write the temporary table to the original table's location, overwriting it
spark.table("healthdata_temp") \
    .write.mode('overwrite') \
    .saveAsTable("healthdata")

# Drop the temporary table
spark.sql("DROP TABLE healthdata_temp")


# spark.sql("SELECT * FROM healthdata") \
#     .show(truncate=False, n=5)

print("Number of rows in the table after removing some records (contaning outliers and data equality ): ")
spark.sql("SELECT COUNT(*) AS NO_OF_ROWS FROM healthdata").show()

#Check null values of each column in HIVE table
print("Number of null values  in the table of each column is : ")
spark.sql(""" SELECT COUNT(*) AS NULL_VALUE_COUNT FROM healthdata
              WHERE Year IS NULL OR Geography IS NULL OR Age_Group IS NULL
              OR  Sex is NULL OR Indicators IS NULL OR Characteristics IS NULL OR Value IS NULL""").show()
```



```
spark.sql("DESCRIBE healthdata").show()

print("Before removing the first row....")
spark.sql("SELECT * FROM healthdata") \
    .show(truncate=False, n=5)


# Filter out the first row
df_new = spark.sql("SELECT * FROM healthdata WHERE Year != 'Year'")

# Write the new DataFrame to a different table
df_new.write.mode("overwrite") \
    .saveAsTable("healthdata_temp")

# Drop the original table
spark.sql("DROP TABLE healthdata")

# Rename the new table to the original table's name
spark.sql("ALTER TABLE healthdata_temp RENAME TO healthdata")

# Display the data from the table after removing the first row
print("After removing the first row....")
spark.sql("SELECT * FROM healthdata") \
    .show(truncate=False, n=5)
```

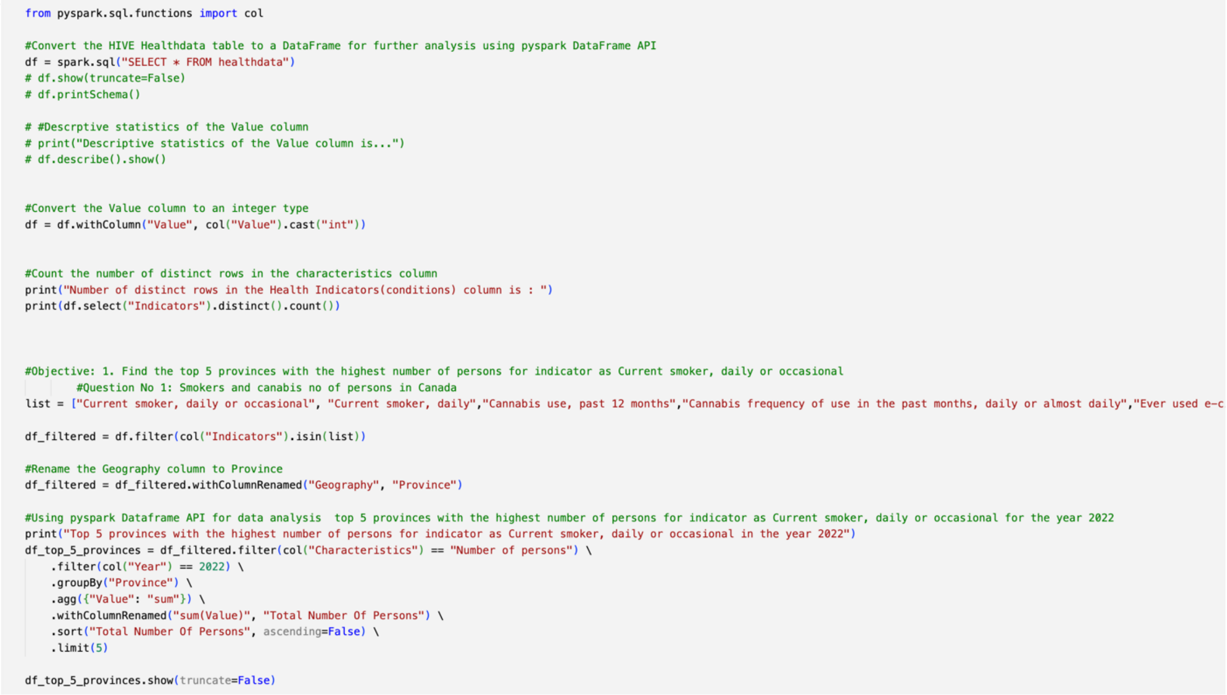
```
Number of rows in the table
24/04/11 23:19:33 WARN HiveExternalCatalog: The table schema given by Hive metastore(struct<year:string,geography:string,age_group:string,sex:string,indicators:string,characteristics:string
+----------+
|NO_OF_ROWS|
+----------+
|    327166|
+----------+

24/04/11 23:19:34 WARN HiveExternalCatalog: The table schema given by Hive metastore(struct<year:string,geography:string,age_group:string,sex:string,indicators:string,characteristics:string
24/04/11 23:19:35 WARN HiveExternalCatalog: The table schema given by Hive metastore(struct<year:string,geography:string,age_group:string,sex:string,indicators:string,characteristics:string
24/04/11 23:19:35 WARN HiveExternalCatalog: The table schema given by Hive metastore(struct<year:string,geography:string,age_group:string,sex:string,indicators:string,characteristics:string
Number of rows in the table after removing some records (contaning outliers and data equality ):
+----------+
|NO_OF_ROWS|
+----------+
|    158841|
+----------+
```



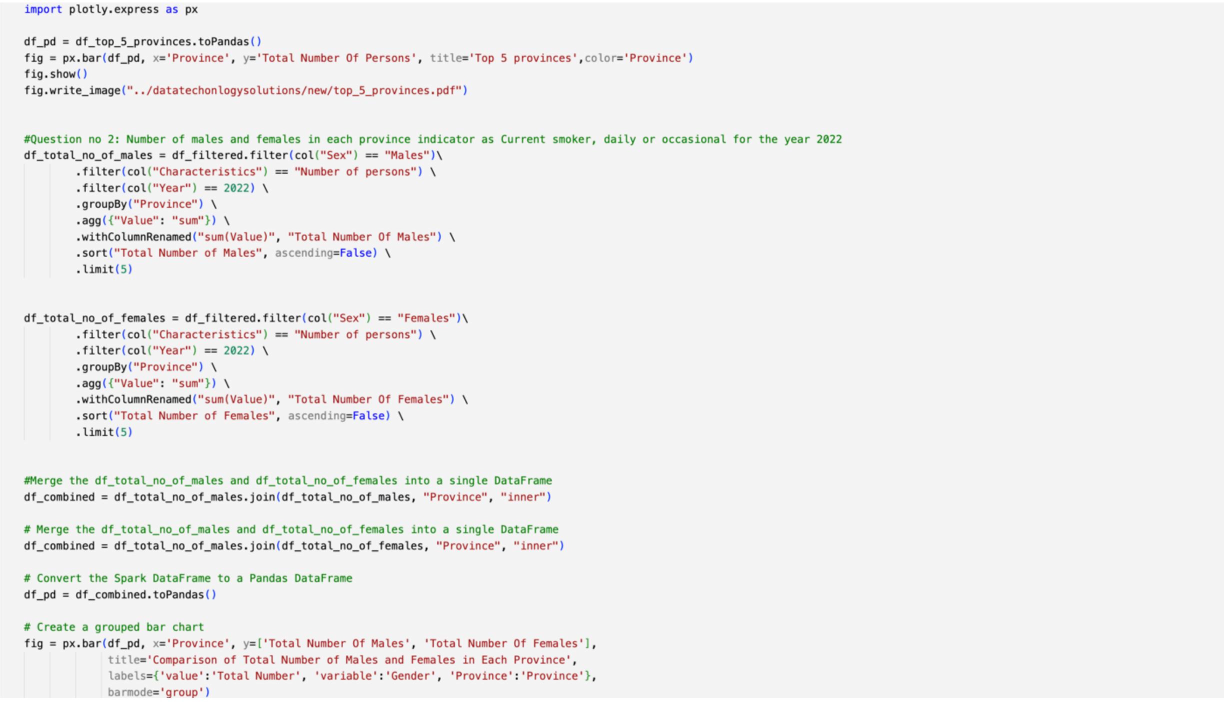
```
24/04/11 23:19:34 WARN HiveExternalCatalog: The table schema given by Hive metastore(struct<year:string,geography:string,age_group:string,sex:string,indicators:string,characteristics:string,v
24/04/11 23:19:35 WARN HiveExternalCatalog: The table schema given by Hive metastore(struct<year:string,geography:string,age_group:string,sex:string,indicators:string,characteristics:string,v
24/04/11 23:19:35 WARN HiveExternalCatalog: The table schema given by Hive metastore(struct<year:string,geography:string,age_group:string,sex:string,indicators:string,characteristics:string,v
Number of rows in the table after removing some records (contaning outliers and data equality ):
+----------+
|NO_OF_ROWS|
+----------+
|    158841|
+----------+

Number of null values  in the table of each column is :
+----------------+
|NULL_VALUE_COUNT|
+----------------+
|               0|
+----------------+

+---------------+---------+-----------------+
|       col_name|data_type|          comment|
+---------------+---------+-----------------+
|           year|   string|from deserializer|
|      geography|   string|from deserializer|
|      age_group|   string|from deserializer|
|            sex|   string|from deserializer|
|     indicators|   string|from deserializer|
|characteristics|   string|from deserializer|
|          value|   string|from deserializer|
+---------------+---------+-----------------+
...
|2015|Newfoundland and Labrador|12 to 17 years|Males|Perceived health, very good or excellent|Low 95% confidence interval, percent           |65.9 |
+----+-------------------------+--------------+-----+----------------------------------------+-----------------------------------------------+-----+
only showing top 5 rows
```

4.Data Analysis with PySpark and Data Visualization:

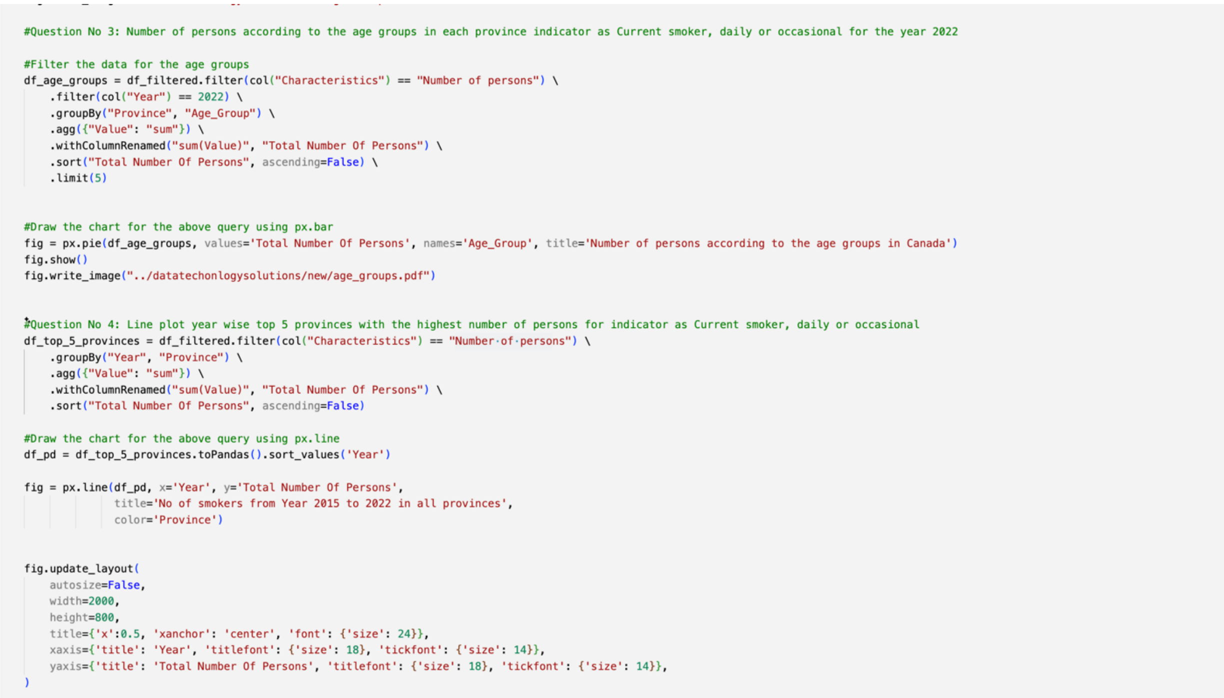
- Utilize PySpark SQL and DataFrame API for analysis.
- Calculate descriptive statistics, aggregations, or any meaningful insights.



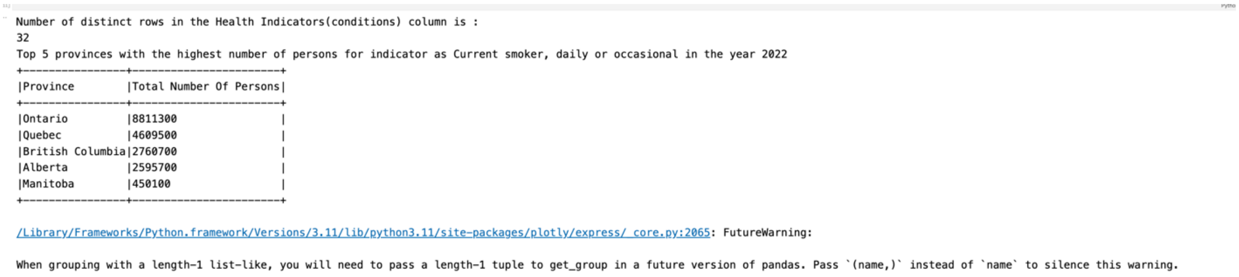
```python
from pyspark.sql.functions import col

#Convert the HIVE Healthdata table to a DataFrame for further analysis using pyspark DataFrame API
df = spark.sql("SELECT * FROM healthdata")
# df.show(truncate=False)
# df.printSchema()

# #Descrptive statistics of the Value column
# print("Descriptive statistics of the Value column is...")
# df.describe().show()


#Convert the Value column to an integer type
df = df.withColumn("Value", col("Value").cast("int"))


#Count the number of distinct rows in the characteristics column
print("Number of distinct rows in the Health Indicators(conditions) column is : ")
print(df.select("Indicators").distinct().count())



#Objective: 1. Find the top 5 provinces with the highest number of persons for indicator as Current smoker, daily or occasional
        #Question No 1: Smokers and canabis no of persons in Canada
list = ["Current smoker, daily or occasional", "Current smoker, daily","Cannabis use, past 12 months","Cannabis frequency of use in the past months, daily or almost daily","Ever used e-c

df_filtered = df.filter(col("Indicators").isin(list))

#Rename the Geography column to Province
df_filtered = df_filtered.withColumnRenamed("Geography", "Province")

#Using pyspark Dataframe API for data analysis  top 5 provinces with the highest number of persons for indicator as Current smoker, daily or occasional for the year 2022
print("Top 5 provinces with the highest number of persons for indicator as Current smoker, daily or occasional in the year 2022")
df_top_5_provinces = df_filtered.filter(col("Characteristics") == "Number of persons") \
    .filter(col("Year") == 2022) \
    .groupBy("Province") \
    .agg({"Value": "sum"}) \
    .withColumnRenamed("sum(Value)", "Total Number Of Persons") \
    .sort("Total Number Of Persons", ascending=False) \
    .limit(5)

df_top_5_provinces.show(truncate=False)
```



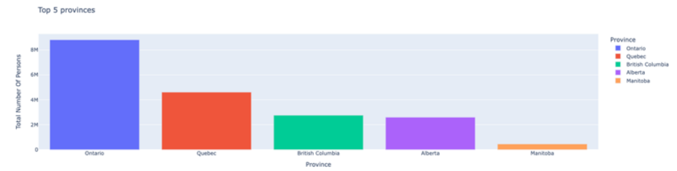
```
import plotly.express as px

df_pd = df_top_5_provinces.toPandas()
fig = px.bar(df_pd, x='Province', y='Total Number Of Persons', title='Top 5 provinces',color='Province')
fig.show()
fig.write_image("../datatechonlogysolutions/new/top_5_provinces.pdf")


#Question no 2: Number of males and females in each province indicator as Current smoker, daily or occasional for the year 2022
df_total_no_of_males = df_filtered.filter(col("Sex") == "Males")\
        .filter(col("Characteristics") == "Number of persons") \
        .filter(col("Year") == 2022) \
        .groupBy("Province") \
        .agg({"Value": "sum"}) \
        .withColumnRenamed("sum(Value)", "Total Number Of Males") \
        .sort("Total Number of Males", ascending=False) \
        .limit(5)


df_total_no_of_females = df_filtered.filter(col("Sex") == "Females")\
        .filter(col("Characteristics") == "Number of persons") \
        .filter(col("Year") == 2022) \
        .groupBy("Province") \
        .agg({"Value": "sum"}) \
        .withColumnRenamed("sum(Value)", "Total Number Of Females") \
        .sort("Total Number of Females", ascending=False) \
        .limit(5)


#Merge the df_total_no_of_males and df_total_no_of_females into a single DataFrame
df_combined = df_total_no_of_males.join(df_total_no_of_males, "Province", "inner")

# Merge the df_total_no_of_males and df_total_no_of_females into a single DataFrame
df_combined = df_total_no_of_males.join(df_total_no_of_females, "Province", "inner")

# Convert the Spark DataFrame to a Pandas DataFrame
df_pd = df_combined.toPandas()

# Create a grouped bar chart
fig = px.bar(df_pd, x='Province', y=['Total Number Of Males', 'Total Number Of Females'],
             title='Comparison of Total Number of Males and Females in Each Province',
             labels={'value':'Total Number', 'variable':'Gender', 'Province':'Province'},
             barmode='group')
```

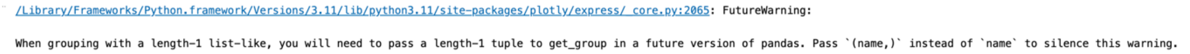


```
#Question No 3: Number of persons according to the age groups in each province indicator as Current smoker, daily or occasional for the year 2022

#Filter the data for the age groups
df_age_groups = df_filtered.filter(col("Characteristics") == "Number of persons") \
    .filter(col("Year") == 2022) \
    .groupBy("Province", "Age_Group") \
    .agg({"Value": "sum"}) \
    .withColumnRenamed("sum(Value)", "Total Number Of Persons") \
    .sort("Total Number Of Persons", ascending=False) \
    .limit(5)


#Draw the chart for the above query using px.bar
fig = px.pie(df_age_groups, values='Total Number Of Persons', names='Age_Group', title='Number of persons according to the age groups in Canada')
fig.show()
fig.write_image("../datatechonlogysolutions/new/age_groups.pdf")


#Question No 4: Line plot year wise top 5 provinces with the highest number of persons for indicator as Current smoker, daily or occasional
df_top_5_provinces = df_filtered.filter(col("Characteristics") == "Number of persons") \
    .groupBy("Year", "Province") \
    .agg({"Value": "sum"}) \
    .withColumnRenamed("sum(Value)", "Total Number Of Persons") \
    .sort("Total Number Of Persons", ascending=False)

#Draw the chart for the above query using px.line
df_pd = df_top_5_provinces.toPandas().sort_values('Year')

fig = px.line(df_pd, x='Year', y='Total Number Of Persons',
              title='No of smokers from Year 2015 to 2022 in all provinces',
              color='Province')


fig.update_layout(
    autosize=False,
    width=2000,
    height=800,
    title={'x':0.5, 'xanchor': 'center', 'font': {'size': 24}},
    xaxis={'title': 'Year', 'titlefont': {'size': 18}, 'tickfont': {'size': 14}},
    yaxis={'title': 'Total Number Of Persons', 'titlefont': {'size': 18}, 'tickfont': {'size': 14}},
)
```

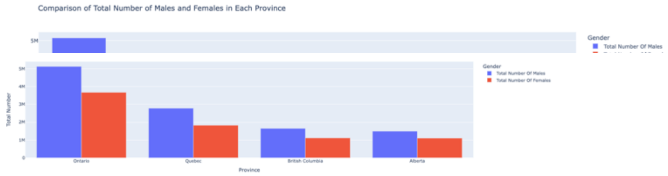


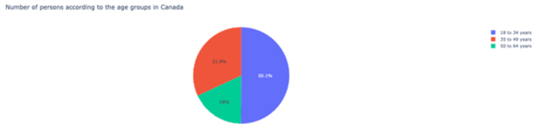
```
Number of distinct rows in the Health Indicators(conditions) column is :
32
Top 5 provinces with the highest number of persons for indicator as Current smoker, daily or occasional in the year 2022
+----------------+-----------------------+
|Province        |Total Number Of Persons|
+----------------+-----------------------+
|Ontario         |8811300                |
|Quebec          |4609500                |
|British Columbia|2760700                |
|Alberta         |2595700                |
|Manitoba        |450100                 |
+----------------+-----------------------+

/Library/Frameworks/Python.framework/Versions/3.11/lib/python3.11/site-packages/plotly/express/_core.py:2065: FutureWarning:

When grouping with a length-1 list-like, you will need to pass a length-1 tuple to get_group in a future version of pandas. Pass `(name,)` instead of `name` to silence this warning.
```



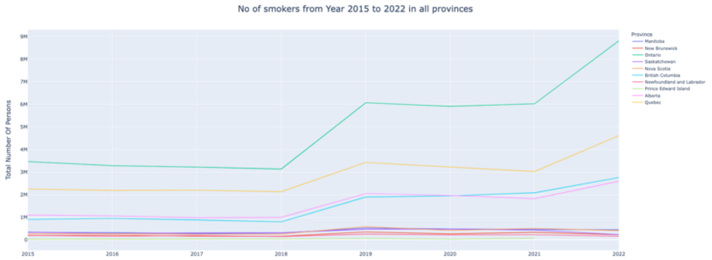


```
/Library/Frameworks/Python.framework/Versions/3.11/lib/python3.11/site-packages/plotly/express/_core.py:2065: FutureWarning:

When grouping with a length-1 list-like, you will need to pass a length-1 tuple to get_group in a future version of pandas. Pass `(name,)` instead of `name` to silence this warning.
```





```
/Library/Frameworks/Python.framework/Versions/3.11/lib/python3.11/site-packages/plotly/express/_core.py:2065: FutureWarning:

When grouping with a length-1 list-like, you will need to pass a length-1 tuple to get_group in a future version of pandas. Pass `(name,)` instead of `name` to silence this warning.
```

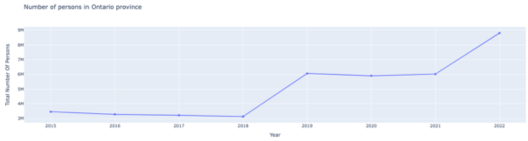


5.**Machine Learning Exploration:**

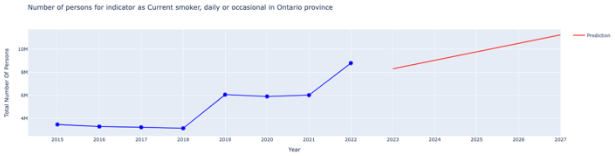
- Explore machine learning tasks using PySpark's MLlib.
- Experiment with classification, regression, or clustering based on the data characteristics.

```
#Predict the number of persons for indicator as Current smoker, daily or occasional in Ontario province for the next 5 years using linear regression
from pyspark.ml.feature import VectorAssembler
from pyspark.ml.regression import LinearRegression
from pyspark.ml import Pipeline

#Convert the Year column to a numeric type
df_ontario = df_ontario.withColumn("Year", col("Year").cast("int"))

#Create a VectorAssembler
assembler = VectorAssembler(inputCols=["Year"], outputCol="features")

#Create a LinearRegression model
lr = LinearRegression(featuresCol="features", labelCol="Total Number Of Persons")

#Create a Pipeline
pipeline = Pipeline(stages=[assembler, lr])

#Fit the model
model = pipeline.fit(df_ontario)

#Predict the number of persons for the next 5 years
df_predict = spark.createDataFrame([(2023,), (2024,), (2025,), (2026,), (2027,)], ["Year"])

#Transform the data
df_predict = model.transform(df_predict)


#draw the prediction line and the scatter plot for the number of persons for indicator as Current smoker, daily or occasional as per year in Ontario province

df_pd = df_ontario.toPandas()
df_predict_pd = df_predict.toPandas()


fig = px.scatter(df_pd, x='Year', y='Total Number Of Persons',
                 title='Number of persons for indicator as Current smoker, daily or occasional in Ontario province',
                 labels={'Total Number Of Persons': 'Total Number Of Persons', 'Year': 'Year'})

fig.update_traces(marker=dict(color='blue', size=10))  # Update the marker size and color
fig.update_traces(mode='lines+markers')

fig.add_scatter(x=df_predict_pd['Year'], y=df_predict_pd['prediction'], mode='lines',
                name='Prediction', line=dict(color='red'),text='Total Number Of Persons')  # Change the color of the prediction line

fig.update_layout(xaxis=dict(dtick=1))  # Show all the years on the x-axis

fig.show()
fig.write_image("../datatechonlogysolutions/new/ontario_prediction_innext5years.pdf")
```

```
fig.add_scatter(x=df_predict_pd['Year'], y=df_predict_pd['prediction'], mode='lines',
                name='Prediction', line=dict(color='red'),text='Total Number Of Persons')  # Change the color of the prediction line

fig.update_layout(xaxis=dict(dtick=1))  # Show all the years on the x-axis

fig.show()
fig.write_image("../datatechonlogysolutions/new/ontario_prediction_innext5years.pdf")



from pyspark.sql.functions import col

# Divide the 'prediction' values by 1,000,000 and create a new column 'prediction_in_millions'
df_predict = df_predict.withColumn('prediction_in_millions', col('prediction') / 1000000)

# Select only the 'Year' and 'prediction_in_millions' columns
df_predict = df_predict.select('Year', 'prediction_in_millions')

df_predict.show()
```



```
DEPRECATION: Loading egg at /Library/Frameworks/Python.framework/Versions/3.11/lib/python3.11/site-packages/jupyter-1.0.0-py3.11.egg is deprecated. pip 24.3 will enforce this behaviour change. A possible replacement is to use pip
Requirement already satisfied: statsmodels in /Library/Frameworks/Python.framework/Versions/3.11/lib/python3.11/site-packages (0.14.1)
Requirement already satisfied: numpy<2,>=1.18 in /Library/Frameworks/Python.framework/Versions/3.11/lib/python3.11/site-packages (from statsmodels) (1.25.2)
Requirement already satisfied: scipy!=1.9.2,>=1.4 in /Library/Frameworks/Python.framework/Versions/3.11/lib/python3.11/site-packages (from statsmodels) (1.11.2)
Requirement already satisfied: pandas!=2.1.0,>=1.0 in /Library/Frameworks/Python.framework/Versions/3.11/lib/python3.11/site-packages (from statsmodels) (2.2.1)
Requirement already satisfied: patsy>=0.5.4 in /Library/Frameworks/Python.framework/Versions/3.11/lib/python3.11/site-packages (from statsmodels) (0.5.6)
Requirement already satisfied: packaging>=21.3 in /Library/Frameworks/Python.framework/Versions/3.11/lib/python3.11/site-packages (from statsmodels) (23.1)
Requirement already satisfied: python-dateutil>=2.8.2 in /Library/Frameworks/Python.framework/Versions/3.11/lib/python3.11/site-packages (from pandas!=2.1.0,>=1.0->statsmodels) (2.8.2)
Requirement already satisfied: pytz>=2020.1 in /Library/Frameworks/Python.framework/Versions/3.11/lib/python3.11/site-packages (from pandas!=2.1.0,>=1.0->statsmodels) (2023.3)
Requirement already satisfied: tzdata>=2022.7 in /Library/Frameworks/Python.framework/Versions/3.11/lib/python3.11/site-packages (from pandas!=2.1.0,>=1.0->statsmodels) (2023.3)
Requirement already satisfied: six in /Library/Frameworks/Python.framework/Versions/3.11/lib/python3.11/site-packages (from patsy>=0.5.4->statsmodels) (1.16.0)
24/04/11 23:33:01 WARN Instrumentation: [0c031e6e] regParam is zero, which might cause numerical instability and overfitting.
```



```
+----+----------------------+
|Year|prediction_in_millions|
+----+----------------------+
|2023|     8.315342856907606|
|2024|     9.054560713998079|
|2025|     9.793778571088552|
|2026|    10.532996428179025|
|2027|    11.272214285269499|
+----+----------------------+
```

6.

- Use data visualization libraries to create visualizations.
- Generate reports summarizing key findings from the analysis.

Tools and Technologies:

- Apache Hadoop (HDFS)
- Apache Hive
- Apache Spark (PySpark)
- Data visualization libraries
- Development environment (Jupyter Notebooks or others)

Business use cases:

- Analyzing customer behaviour and preferences.
- Detecting fraudulent activities in financial transactions.
- Improving efficiency and reducing costs in the supply chain.
- Analyzing electronic health records for insights and decision-making.
- Enhancing the customer shopping experience through personalized recommendations.
- Predicting and optimizing energy consumption in a smart city.

Each student needs to present for 5 minutes about their project during the 13th(April 13th) or 14th (April 20th – 10:am – noon) week. Please book the time in advance.

Deliverables:

- Hive SQL scripts for data exploration and preprocessing.
- PySpark script for data analysis.
- Data visualization notebook or script.
- Final report PowerPoint presentation.
- Demo video showcasing the project.

Notes:

- Choose a dataset aligned with your interests or domain expertise.
- Document your code and analysis steps