120A3051
Shreya Idate
Batch: E3

EXPERIMENT NO. 10

Aim: Exploratory data analysis using Apache Spark and Pandas
Theory:

What is Apache Spark?

Apache Spark is an open-source unified analytics engine for large-scale data processing. Spark provides an interface for programming clusters with implicit data parallelism and fault tolerance. Originally developed at the University of California, Berkeley's AMPLab, the Spark codebase was later donated to the Apache Software Foundation, which has maintained it since.

Spark Core

Spark Core is the foundation of the overall project. It provides distributed task dispatching, scheduling, and basic I/O functionalities, exposed through an application programming interface (for Java, Python, Scala, .NETand R) centered on the RDD abstraction (the Java API is available for other JVM languages, but is also usable for some other non-JVM languages that can connect to the JVM, such as Julia). This interface mirrors a functional/higher-order model of programming: a "driver" program invokes parallel operations such as map, filter or reduce on an RDD by passing a function to Spark, which then schedules the function's execution in parallel on the cluster.These operations, and additional ones such as joins, take RDDs as input and produce new RDDs. RDDs are immutable and their operations are lazy; fault-tolerance is achieved by keeping track of the "lineage" of each RDD (the sequence of operations that produced it) so that it can be reconstructed in the case of data loss. RDDs can contain any type of Python, .NET, Java, or Scala objects.

Besides the RDD-oriented functional style of programming, Spark provides two restricted forms of shared variables: broadcast variables reference read-only data that needs to be available on all nodes, while accumulators can be used to program reductions in an imperative style.

A typical example of RDD-centric functional programming is the following Scala program that computes the frequencies of all words occurring in a set of text files and prints the most common ones. Each map, flatMap (a variant of map) and reduceByKey takes an anonymous function that performs a simple operation on a single data item (or a pair of items), and applies its argument to transform an RDD into a new RDD.

Spark Streaming

Spark Streaming uses Spark Core's fast scheduling capability to perform streaming analytics. It ingests data in mini-batches and performs RDD transformations on those mini-batches of data. This design enables the same set of application code written for batch analytics to be used in streaming analytics, thus facilitating easy implementation of lambda architecture. However, this convenience comes with the penalty of latency equal to the mini-batch duration. Other streaming data engines that process event by event rather than in mini-batches include Storm and the streaming component of Flink.Spark Streaming has support built-in to consume from Kafka, Flume, Twitter, ZeroMQ, Kinesis, and TCP/IP sockets.

In Spark 2.x, a separate technology based on Datasets, called Structured Streaming, that has a higher-level interface is also provided to support streaming.

Spark can be deployed in a traditional on-premises data center as well as in the cloud.

MLlib Machine Learning Library

Spark MLlib is a distributed machine-learning framework on top of Spark Core that, due in large part to the distributed memory-based Spark architecture, is as much as nine times as fast as the disk-based implementation used by Apache Mahout (according to benchmarks done by the MLlib developers against the alternating least squares (ALS) implementations, and before Mahout itself gained a Spark interface), and scales better than Vowpal Wabbit.Many common machine learning and statistical algorithms have been implemented and are shipped with MLlib which simplifies large scale machine learning pipelines.

Program:

```
[1] !pip install pyspark

Looking in indexes: https://pypi.org/simple, https://us-python.pkg.dev/colab-wheels/public/simple/
Requirement already satisfied: pyspark in /usr/local/lib/python3.9/dist-packages (3.4.0)
Requirement already satisfied: py4j==0.10.9.7 in /usr/local/lib/python3.9/dist-packages (from pyspark) (0.10.9.7)
```

```
[2] from google.colab import drive

drive.mount('/content/drive')

Drive already mounted at /content/drive; to attempt to forcibly remount, call drive.mount("/content/drive", force_remount=True).
```

```
import pandas as pd
from pyspark.sql import SparkSession

# create a SparkSession
spark = SparkSession.builder.appName("Test").getOrCreate()
```

```
[4] spark

SparkSession - in-memory
SparkContext
Spark UI

Version
    v3.4.0
Master
    local[*]
AppName
    Test
```

```
[5] path = '/content/drive/MyDrive/Colab Notebooks/Customers.csv'
```

```
[6] from pyspark.sql.types import StructType, StructField, StringType, FloatType, IntegerType

customSchema = StructType(
                          [StructField("CustomerID",IntegerType(),True),
                           StructField("Gender",StringType(),True),
                           StructField("Age",IntegerType(),True),
                           StructField("Annual Income ($)",IntegerType(),True),
                           StructField("Spending Score (1-100)",IntegerType(),True),
                           StructField("Profession",StringType(),True),
                           StructField("Work Experience",IntegerType(),True),
                           StructField("Family Size",IntegerType(),True)
                          ]
                         )
df = spark.read.csv(path,schema=customSchema,header=True)
```

```
[7] df.printSchema()
```

```
root
 |-- CustomerID: integer (nullable = true)
 |-- Gender: string (nullable = true)
 |-- Age: integer (nullable = true)
 |-- Annual Income ($): integer (nullable = true)
 |-- Spending Score (1-100): integer (nullable = true)
 |-- Profession: string (nullable = true)
 |-- Work Experience: integer (nullable = true)
 |-- Family Size: integer (nullable = true)
```

```
df.show()
```

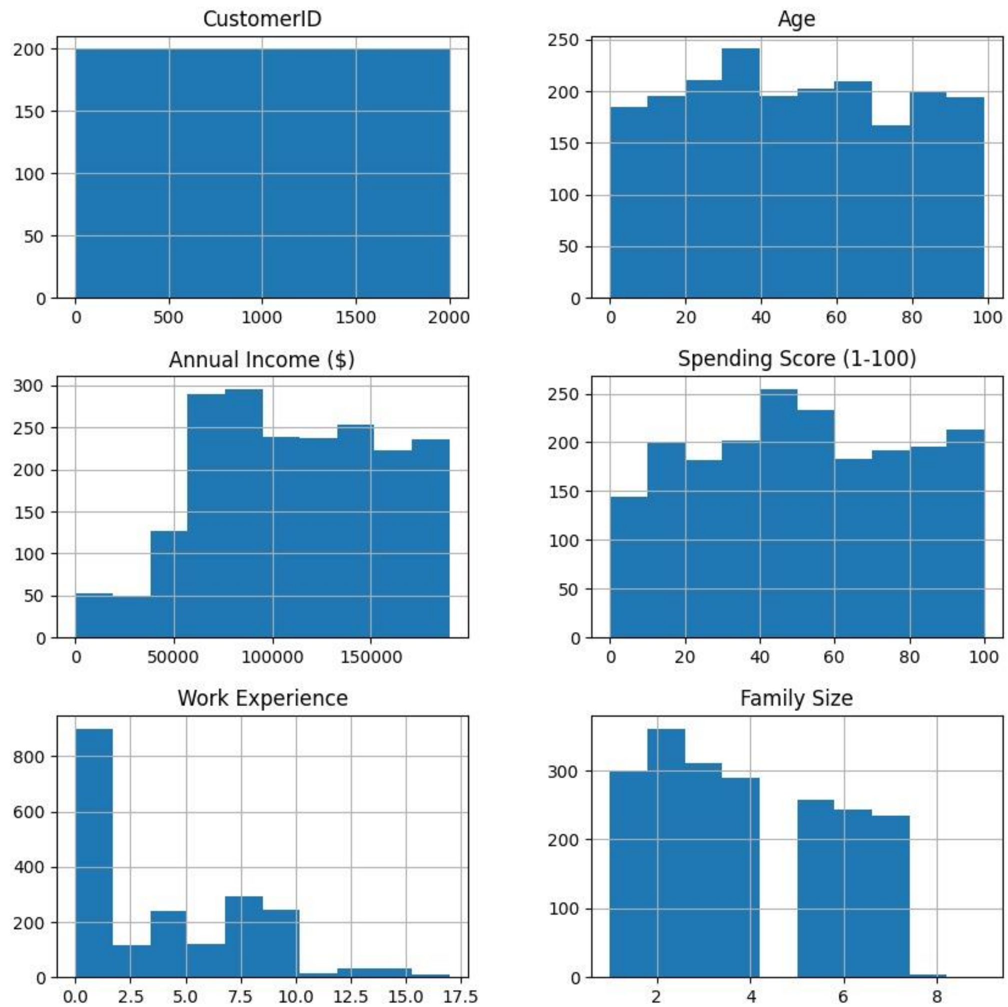
```
+----------+------+---+-----------------+----------------------+-------------+---------------+-----------+
|CustomerID|Gender|Age|Annual Income ($)|Spending Score (1-100)|   Profession|Work Experience|Family Size|
+----------+------+---+-----------------+----------------------+-------------+---------------+-----------+
|         1|  Male| 19|            15000|                    39|   Healthcare|              1|          4|
|         2|  Male| 21|            35000|                    81|     Engineer|              3|          3|
|         3|Female| 20|            86000|                     6|     Engineer|              1|          1|
|         4|Female| 23|            59000|                    77|       Lawyer|              0|          2|
|         5|Female| 31|            38000|                    40|Entertainment|              2|          6|
|         6|Female| 22|            58000|                    76|       Artist|              0|          2|
|         7|Female| 35|            31000|                     6|   Healthcare|              1|          3|
|         8|Female| 23|            84000|                    94|   Healthcare|              1|          3|
|         9|  Male| 64|            97000|                     3|     Engineer|              0|          3|
|        10|Female| 30|            98000|                    72|       Artist|              1|          4|
|        11|  Male| 67|             7000|                    14|     Engineer|              1|          3|
|        12|Female| 35|            93000|                    99|   Healthcare|              4|          4|
|        13|Female| 58|            80000|                    15|    Executive|              0|          5|
|        14|Female| 24|            91000|                    77|       Lawyer|              1|          1|
|        15|  Male| 37|            19000|                    13|       Doctor|              0|          1|
|        16|  Male| 22|            51000|                    79|   Healthcare|              1|          2|
|        17|Female| 35|            29000|                    35|    Homemaker|              9|          5|
|        18|  Male| 20|            89000|                    66|   Healthcare|              1|          6|
|        19|  Male| 52|            20000|                    29|Entertainment|              1|          4|
|        20|Female| 35|            62000|                    98|       Artist|              0|          1|
+----------+------+---+-----------------+----------------------+-------------+---------------+-----------+
only showing top 20 rows
```

```
[9]  df.describe().show()

+-------+-----------------+------+-----------------+-----------------+--------------------+----------+------------------+------------------+
|summary|       CustomerID|Gender|              Age|Annual Income ($)|Spending Score (1-100)|Profession|   Work Experience|       Family Size|
+-------+-----------------+------+-----------------+-----------------+--------------------+----------+------------------+------------------+
|  count|             2000|  2000|             2000|             2000|                2000|      1965|              2000|              2000|
|   mean|           1000.5|  null|            48.96|      110731.8215|             50.9625|      null|            4.1025|            3.7685|
| stddev|577.4945887192364|  null|28.429747189565916|45739.53668828386|   27.93466066346952|      null|3.9222041753070958|1.9707485062375214|
|    min|                1|Female|                0|                0|                   0|    Artist|                 0|                 1|
|    max|             2000|  Male|               99|           189974|                 100| Marketing|                17|                 9|
+-------+-----------------+------+-----------------+-----------------+--------------------+----------+------------------+------------------+
```

```
[10] df.dtypes

[('CustomerID', 'int'),
 ('Gender', 'string'),
 ('Age', 'int'),
 ('Annual Income ($)', 'int'),
 ('Spending Score (1-100)', 'int'),
 ('Profession', 'string'),
 ('Work Experience', 'int'),
 ('Family Size', 'int')]
```

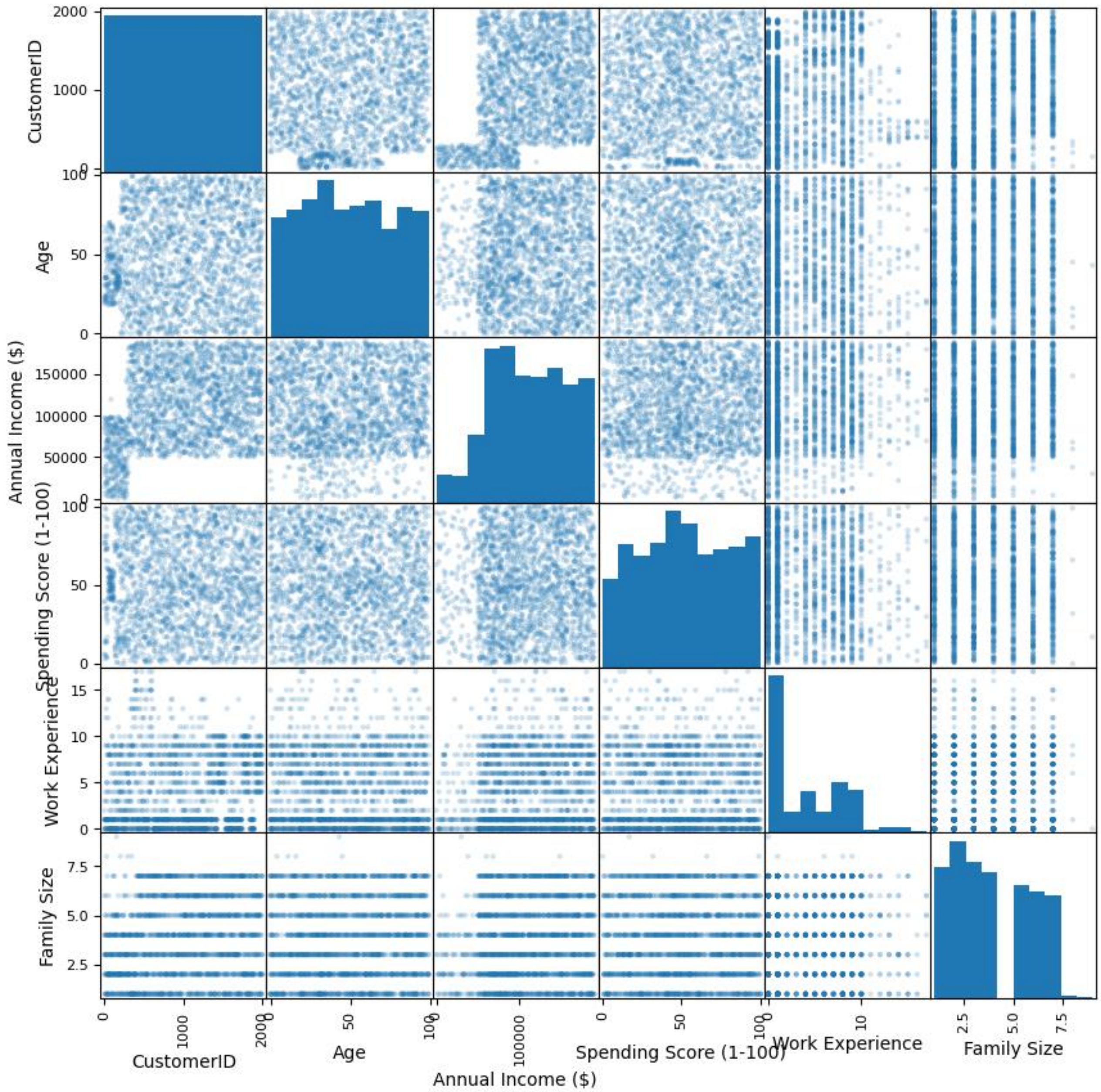
```
[11] filtered_df = df.filter((df['Age']>=20) & (df['Age']<30)).show()

+----------+------+---+-----------------+----------------------+----------+---------------+-----------+
|CustomerID|Gender|Age|Annual Income ($)|Spending Score (1-100)|Profession|Work Experience|Family Size|
+----------+------+---+-----------------+----------------------+----------+---------------+-----------+
|         2|  Male| 21|            35000|                    81|  Engineer|              3|          3|
|         3|Female| 20|            86000|                     6|  Engineer|              1|          1|
|         4|Female| 23|            59000|                    77|    Lawyer|              0|          2|
|         6|Female| 22|            58000|                    76|    Artist|              0|          2|
|         8|Female| 23|            84000|                    94|Healthcare|              1|          3|
|        14|Female| 24|            91000|                    77|    Lawyer|              1|          1|
|        16|  Male| 22|            51000|                    79|Healthcare|              1|          2|
|        18|  Male| 20|            89000|                    66|Healthcare|              1|          6|
|        22|  Male| 25|             4000|                    73|Healthcare|              3|          4|
|        26|  Male| 29|            52000|                    82|    Artist|              1|          3|
|        30|Female| 23|            20000|                    87|    Artist|              5|          4|
|        32|Female| 21|            34000|                    73|    Doctor|              1|          2|
|        36|Female| 21|            95000|                    81|Healthcare|              3|          4|
|        40|Female| 20|            69000|                    75|    Artist|              8|          2|
|        42|  Male| 24|            85000|                    92|Healthcare|              0|          2|
|        46|Female| 24|             3000|                    65|    Lawyer|              4|          2|
|        48|Female| 27|            71000|                    47|Healthcare|             12|          1|
|        49|Female| 29|            78000|                    42|Healthcare|              0|          4|
|        59|Female| 27|            57000|                    51|    Artist|              1|          3|
|        76|  Male| 26|            49000|                    54| Homemaker|              0|          3|
+----------+------+---+-----------------+----------------------+----------+---------------+-----------+
only showing top 20 rows
```

```
[12] pandas_df = df.toPandas()
     pandas_df.hist(figsize=(10, 10))

array([[<Axes: title={'center': 'CustomerID'}>,
        <Axes: title={'center': 'Age'}>],
       [<Axes: title={'center': 'Annual Income ($)'}>,
        <Axes: title={'center': 'Spending Score (1-100)'}>],
       [<Axes: title={'center': 'Work Experience'}>,
        <Axes: title={'center': 'Family Size'}>]], dtype=object)
```




```
pd.plotting.scatter_matrix(pandas_df, alpha=0.2, figsize=(10, 10))
```

```
array([[<Axes: xlabel='CustomerID', ylabel='CustomerID'>,
        <Axes: xlabel='Age', ylabel='CustomerID'>,
        <Axes: xlabel='Annual Income ($)', ylabel='CustomerID'>,
        <Axes: xlabel='Spending Score (1-100)', ylabel='CustomerID'>,
        <Axes: xlabel='Work Experience', ylabel='CustomerID'>,
        <Axes: xlabel='Family Size', ylabel='CustomerID'>],
       [<Axes: xlabel='CustomerID', ylabel='Age'>,
        <Axes: xlabel='Age', ylabel='Age'>,
        <Axes: xlabel='Annual Income ($)', ylabel='Age'>,
        <Axes: xlabel='Spending Score (1-100)', ylabel='Age'>,
        <Axes: xlabel='Work Experience', ylabel='Age'>,
        <Axes: xlabel='Family Size', ylabel='Age'>],
       [<Axes: xlabel='CustomerID', ylabel='Annual Income ($)'>,
        <Axes: xlabel='Age', ylabel='Annual Income ($)'>,
        <Axes: xlabel='Annual Income ($)', ylabel='Annual Income ($)'>,
        <Axes: xlabel='Spending Score (1-100)', ylabel='Annual Income ($)'>,
        <Axes: xlabel='Work Experience', ylabel='Annual Income ($)'>,
        <Axes: xlabel='Family Size', ylabel='Annual Income ($)'>],
       [<Axes: xlabel='CustomerID', ylabel='Spending Score (1-100)'>,
        <Axes: xlabel='Age', ylabel='Spending Score (1-100)'>,
        <Axes: xlabel='Annual Income ($)', ylabel='Spending Score (1-100)'>,
        <Axes: xlabel='Spending Score (1-100)', ylabel='Spending Score (1-100)'>,
        <Axes: xlabel='Work Experience', ylabel='Spending Score (1-100)'>,
        <Axes: xlabel='Family Size', ylabel='Spending Score (1-100)'>],
```



Conclusion: Successfully performed exploratory data analysis using Apache Spark and Pandas.