**Netaji Subhas University of Technology**

****

# **REPORT FILE**

**COMPUTER HARDWARE AND SOFTWARE**

**SUBMITTED BY:-**

**Raveesh Gupta (2021UCS1581)**

**Devashis Soren (2021UCS1595)**

**Aryan Gangwar (2021UCS1606)**

## 

## **Project Title: Spark**

**Task 1:**

**Explore RDD in spark**

**1)What is Spark:**

Apache Spark is an open-source framework for large-scale data processing. It excels at handling big data workloads efficiently, making it a popular tool in data science and analytics. Here's a breakdown of Spark's key features:

* **Speed:** Spark leverages in-memory computation, significantly speeding up data processing compared to traditional disk-based methods. This is especially beneficial for iterative tasks like machine learning.
* **Scalability:** Spark can be easily scaled up by distributing tasks across clusters of machines. This allows you to handle datasets that wouldn't fit on a single machine.
* **Unified Platform:** Spark offers a unified platform for various data processing tasks, including batch processing, real-time streaming, SQL queries, machine learning, and graph processing. This eliminates the need for separate tools for different stages of the data pipeline.
* **Ease of Use:** Spark provides APIs in popular programming languages like Python, Scala, Java, and R. This makes it accessible to developers and data scientists with varying backgrounds.
* **Open-Source:** Being open-source, Spark is free to use and has a large community that contributes to its development and offers support.

In essence, Spark provides a powerful and versatile toolkit for handling big data. It simplifies complex data processing tasks and enables large-scale data analytics efficiently.

**2)What is pySpark:**

PySpark is the bridge between the Apache Spark framework and the Python programming language. It acts as a Python API (Application Programming Interface) that allows you to utilise Spark's functionalities within Python code.

Here's a more detailed explanation of PySpark:

* **Enables Python for Big Data:** Since Apache Spark is primarily written in Scala, PySpark makes Spark accessible to programmers comfortable with Python. This is advantageous because Python is a popular and relatively easy-to-learn language, especially for data science and analysis tasks.
* **Spark Functionalities in Python:** PySpark provides access to all the core functionalities of Spark. This includes working with large datasets using Spark's distributed processing capabilities, in-memory computation for speed, and fault tolerance mechanisms.
* **Familiar Syntax:** By using PySpark, you can leverage Spark's power while writing code in Python's clear and concise syntax. This can improve development speed and code readability for Python programmers.
* **Spark Libraries:** PySpark also grants access to Spark's built-in libraries, such as:  
  + Spark SQL: Enables working with structured data using SQL-like queries.
  + DataFrames: Offer a powerful data structure for distributed processing of large datasets, similar to Pandas DataFrames but optimised for Spark's architecture.
  + MLlib: Provides tools and algorithms for machine learning tasks on big data.
  + Spark Streaming: Facilitates real-time data processing for handling continuous data streams.

**3)What is RDD:**

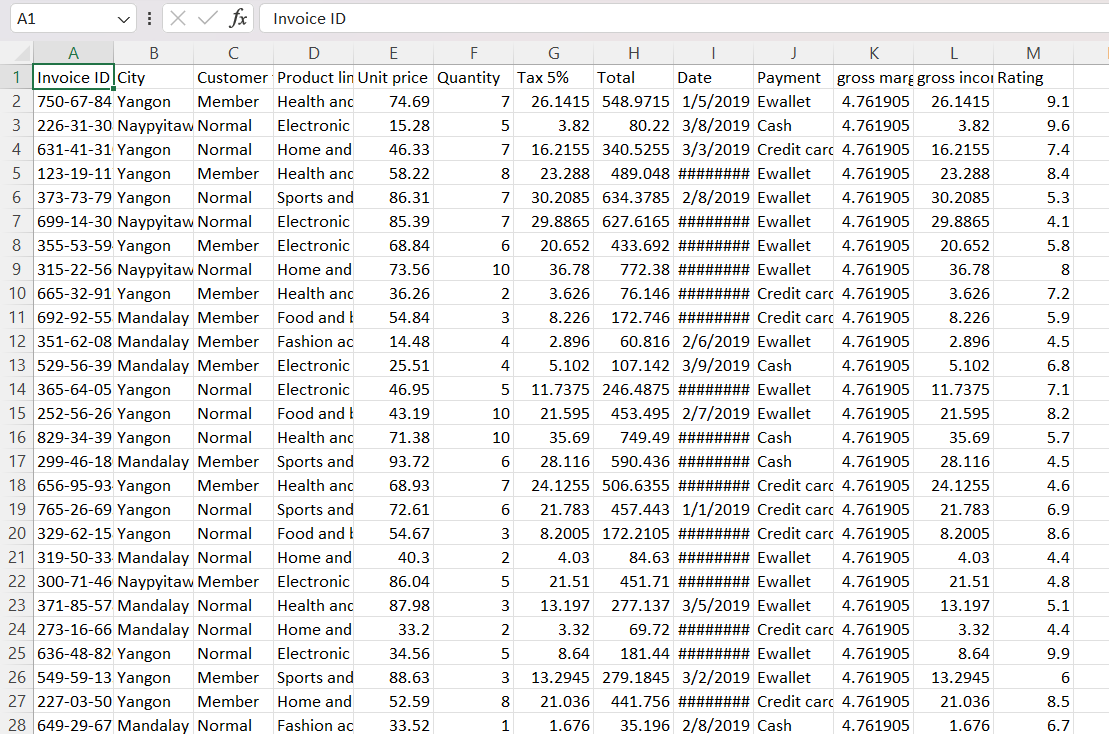
RDD (Resilient Distributed Dataset) is a fundamental data structure in Apache Spark, which is a distributed computing framework for big data processing. RDDs are fault-tolerant, immutable, and partitioned collections of records that can be operated on in parallel across multiple nodes in a cluster. Here's an overview of RDDs in Spark:

* **Resilient**: RDDs are resilient to node failures because they are able to automatically recreate partitions on different nodes when failures occur.
* **Distributed:** RDDs are distributed across multiple nodes in a cluster, which allows for parallel processing and scalability.
* **Dataset:** An RDD represents a collection of data items, which can be of any type (e.g., strings, integers, objects).
* **Immutable:** RDDs are immutable, meaning that once created, they cannot be modified. Any transformation on an RDD creates a new RDD, leaving the original RDD unchanged.
* **Partitioned:** RDDs are partitioned across multiple nodes in the cluster, allowing for parallelism and efficient data distribution.
* **Lazy Evaluation:** RDDs follow a lazy evaluation model, which means that transformations on RDDs are not executed until an action is called to materialise the results.
* **Transformations:** Transformations are operations on RDDs that create new RDDs. Examples include map, filter, flatMap, union, join, and many others.
* **Actions:** Actions are operations on RDDs that trigger the actual computation and return a result. Examples include count, collect, reduce, take, and saveAsTextFile.
* **Lineage:** RDDs track their lineage, which is the sequence of transformations applied to create them. This lineage information is used for fault tolerance and efficient recomputation of RDD partitions in case of node failures.
* **Persistence:** RDDs can be persisted in memory or on disk for faster access in iterative computations or when reusing the same dataset multiple times

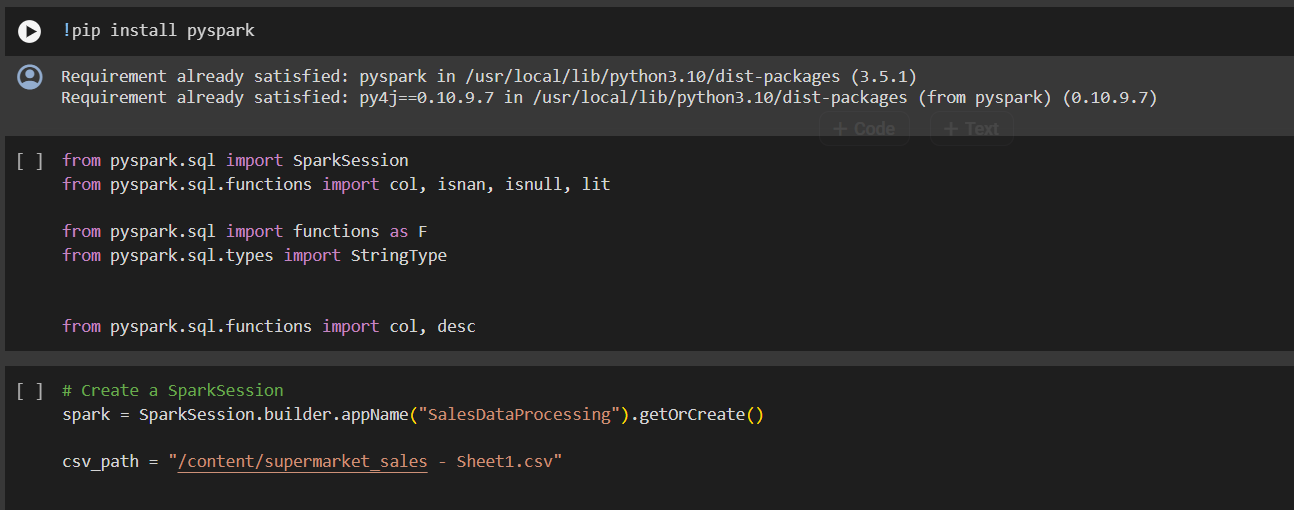
**Task 2:**

**In PySpark, create a program that reads a CSV file containing sales data, performs data cleaning by handling missing values and removing duplicates, calculates the total sales amount for each product, and finally, outputs the results to a new CSV file. Ensure to use transformations and actions in your PySpark script**

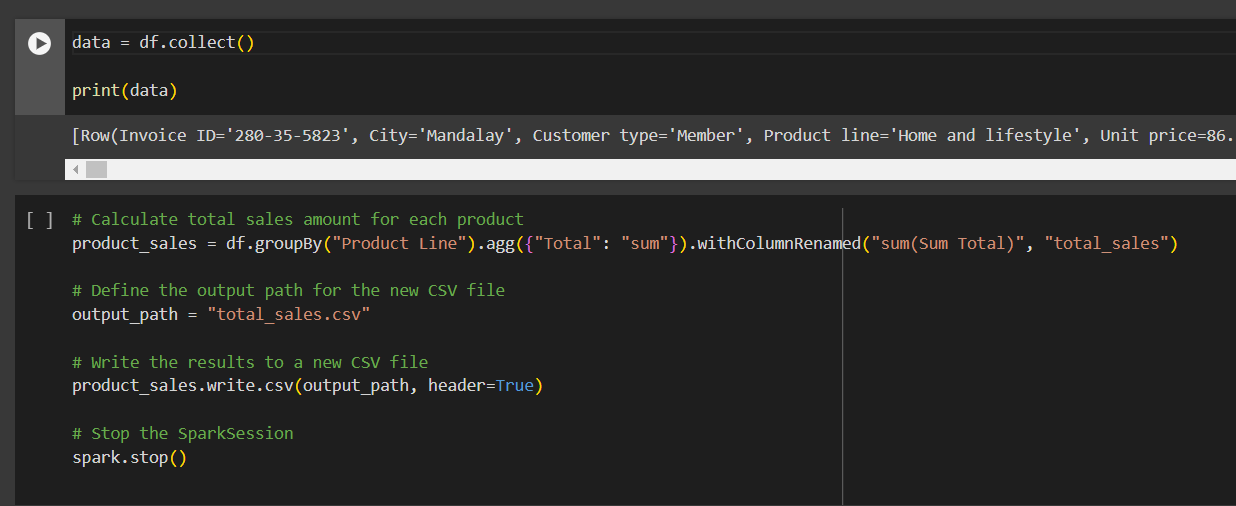
**Dataset:**

****

**Code:**

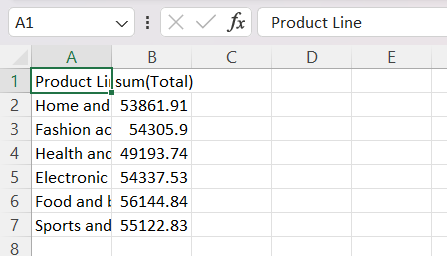
****

****

****

**Output:**

Calculated the total amount for each sales item

****

Sorted the “unit price” col.

