**Madiha Aimon Tappal**

[**madihaaimon@gmail.com**](mailto:madihaaimon@gmail.com)

**Data Engineering Batch – 1**

**Day – 13 Assignment**

**PySpark**

**PySpark: -**

PySpark is the Python API for Apache Spark, an open-source distributed computing system designed for big data processing and analytics. Spark provides a fast and general-purpose cluster-computing framework for large-scale data processing.

Here are some key theoretical concepts related to PySpark:

1. **Resilient Distributed Datasets (RDD):**
   * RDD is the fundamental data structure in Spark, representing an immutable distributed collection of objects.
   * It allows parallel processing of data across a cluster, dividing the data into partitions that can be processed independently.
   * RDDs are fault-tolerant, meaning if a partition is lost due to a node failure, Spark can reconstruct it.
2. **Transformations:**
   * Transformations in PySpark are operations applied to an RDD to create a new RDD.
   * Examples of transformations include **map**, **filter**, **flatMap**, **union**, and more.
   * Transformations are lazily evaluated, meaning they are not executed immediately but only when an action is triggered.
3. **Actions:**
   * Actions are operations that trigger the execution of transformations and return a result to the driver program or write data to external storage.
   * Examples of actions include **count**, **collect**, **reduce**, **saveAsTextFile**, etc.
4. **SparkContext:**
   * SparkContext is the entry point for any Spark functionality in a PySpark application.
   * It manages the connection to the Spark cluster and coordinates the execution of operations on RDDs.
5. **Driver Program:**
   * The driver program is the main program that runs the user’s Spark code and creates the SparkContext.
   * It defines transformations and actions on RDDs, which are then executed on the cluster.
6. **Executor:**
   * Executors are processes that run on worker nodes and are responsible for executing tasks.
   * They manage the data stored on their node and perform computations requested by the driver program.
7. **SparkSession:**
   * SparkSession is the entry point for DataFrame and SQL functionality in PySpark.
   * It is used to create DataFrames, which provide a more high-level and structured API compared to RDDs.
8. **DataFrame and SQL:**
   * DataFrames are distributed collections of data organized into named columns, similar to a table in a relational database.
   * PySpark supports SQL queries using Spark SQL, allowing users to query DataFrames using SQL syntax.
9. **Caching and Persistence:**
   * RDDs and DataFrames can be cached in memory to speed up iterative algorithms and multiple computations on the same dataset.
   * Persistence allows you to control the storage level and choose whether to store the data in memory, on disk, or both.
10. **Broadcast Variables:**
    * Broadcast variables allow the efficient sharing of large read-only variables among worker nodes.
    * They are used to cache a value or an object in serialized form to avoid redundant data transfer over the network.

**What is Apache spark: -**

Apache Spark is an open-source distributed computing system that is designed for big data processing and analytics. It provides a fast and general-purpose cluster computing framework for large-scale data processing. Spark was developed to address the limitations of the MapReduce programming model, offering improved performance and ease of use.

Key features of Apache Spark include:

1. **Speed:** Spark is known for its speed, primarily due to its ability to perform in-memory processing. It can cache intermediate data in memory, reducing the need to read and write to disk.
2. **Ease of Use:** Spark provides high-level APIs in Java, Scala, Python, and R, making it accessible to a wide range of developers. It also includes a built-in set of high-level libraries for tasks such as SQL queries, machine learning, graph processing, and stream processing.
3. **Generality:** Spark is designed to handle a wide range of data processing workloads, including batch processing, iterative algorithms, interactive queries, and streaming.
4. **Fault Tolerance:** Spark provides fault tolerance through resilient distributed datasets (RDDs). RDDs are immutable distributed collections of objects, and they automatically recover from node failures.
5. **Compatibility:** Spark is compatible with Hadoop Distributed File System (HDFS) and can run on existing Hadoop clusters. It can also be integrated with other data sources and storage systems.
6. **Spark SQL:** Spark includes a module called Spark SQL that allows developers to query structured data using SQL as well as Data Frame APIs. This facilitates seamless integration with existing SQL-based tools and databases.
7. **Machine Learning Library (MLlib):** Spark includes a machine learning library for scalable and distributed machine learning tasks. It supports various algorithms for classification, regression, clustering, and collaborative filtering.
8. **GraphX:** Spark's GraphX library provides a distributed graph processing framework for graph analytics.

**Who uses PySpark?**

* PySpark is widely used in the field of data science and machine learning due to its ability to efficiently handle large-scale data processing in distributed computing environments.
* PySpark is a powerful tool for data scientists and machine learning practitioners who need to work with large-scale datasets in distributed computing environments.
* Its integration with Spark's ecosystem, scalability, and support for various machine learning tasks make it a valuable tool in the data science and machine learning toolbox.

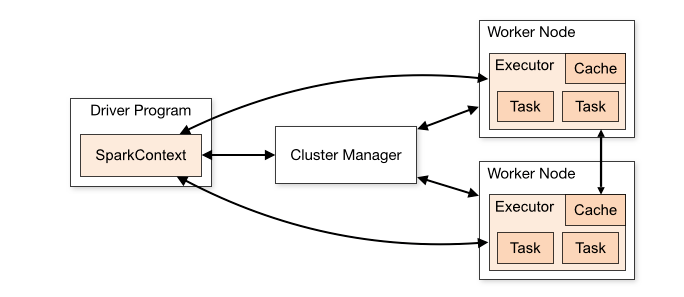
**What are features of pySpark?**

* In memory computation
* Immutable
* Distributed processing using parallelize
* Can be used with many cluster managers (Spark, YARN, Mesos etc.)
* Cache & Persistence
* Lazy evaluation
* Inbuild-Optimization when using Data Frames, Fault-tolerant

**Advantages of Pyspark: -**

* PySpark is a general purpose, in memory, distributed processing engine that allows you to process data efficiently in a distributed fashion.
* Applications running on PySpark are 100X faster than traditional systems.
* You will get great benefits from using PySpark for data integration.
* Using PySpark we can process data from Hadoop HDFS, AWS S3 and many file systems.
* PySpark also is used to process real-time data using streaming & Katka.

**PySpark Architecture: -**



PySpark, being the Python API for Apache Spark, operates within the broader Apache Spark architecture. Here's an overview of the key components and their interactions:

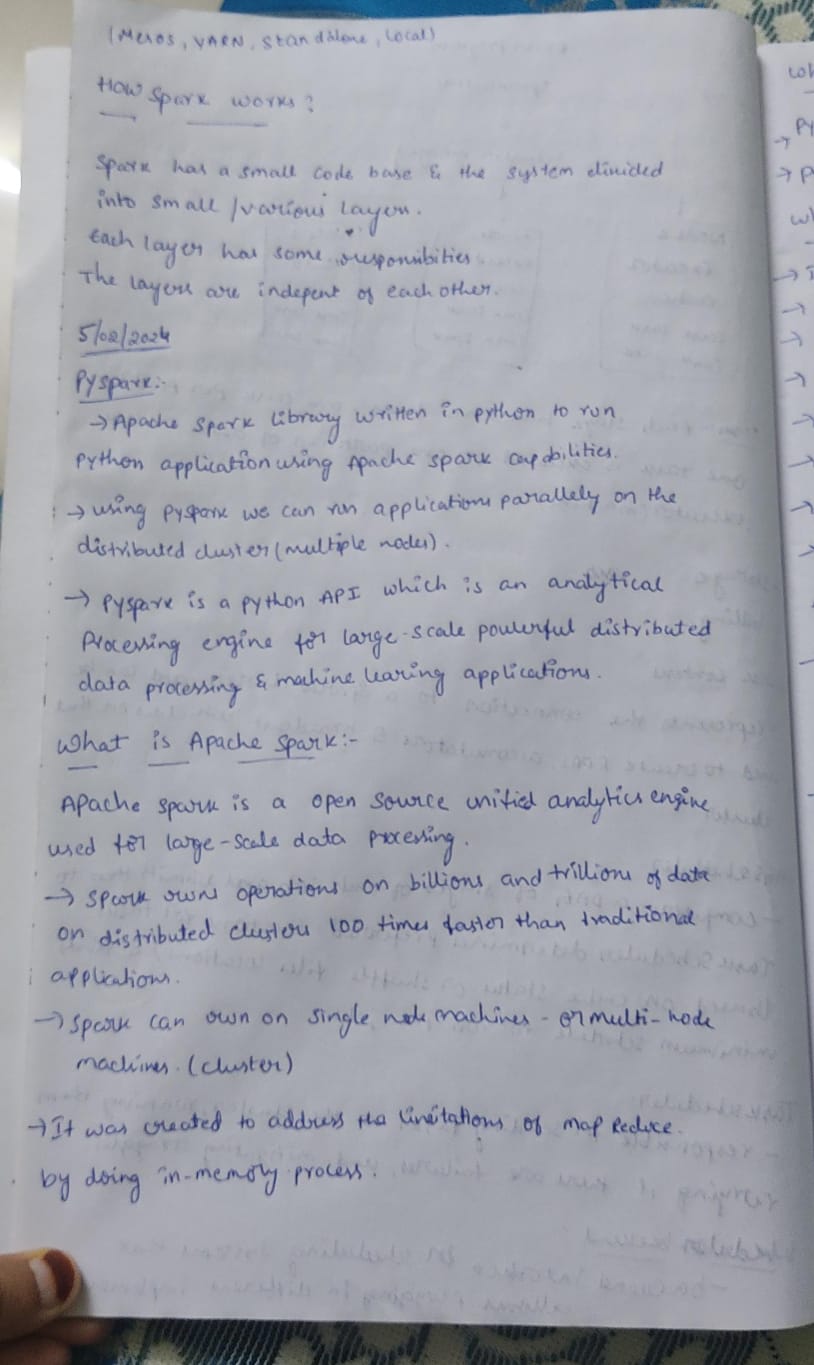
1. **Driver Program:**
   * The driver program is the main application that runs the user's PySpark code. It contains the SparkContext, which is the entry point for interacting with the Spark cluster.
2. **SparkContext:**
   * The SparkContext is responsible for coordinating the execution of tasks across the Spark cluster. It establishes communication with the cluster manager and ensures the proper distribution of tasks to worker nodes.
3. **Cluster Manager:**
   * The cluster manager is responsible for managing resources across the Spark cluster. Apache Spark supports various cluster managers such as Standalone, Apache Mesos, and Apache Hadoop YARN.
4. **Worker Nodes:**
   * Worker nodes are individual machines within the Spark cluster. They are responsible for executing tasks assigned by the driver program. Each worker node has its own executor processes.
5. **Executors:**
   * Executors are processes running on worker nodes that are responsible for executing tasks. They manage the local storage and perform computations as instructed by the driver program.
6. **Resilient Distributed Datasets (RDD):**
   * RDD is the fundamental data structure in Spark. It represents an immutable distributed collection of objects partitioned across the nodes in the cluster. RDDs are fault-tolerant and can be cached for performance optimization.
7. **DataFrames and Spark SQL:**
   * DataFrames and Spark SQL provide higher-level abstractions for working with structured data. They are built on top of RDDs and offer a more user-friendly interface for data manipulation and querying.
8. **SparkSession:**
   * SparkSession is the entry point for using DataFrames and Spark SQL in PySpark. It provides a unified interface and consolidates the functionality of the earlier SparkContext, SQLContext, and HiveContext.
9. **RDD and DataFrame Operations:**
   * Transformations and actions on RDDs and DataFrames define the processing logic. Transformations create new RDDs or DataFrames, while actions trigger the execution of the computation plan.
10. **Cluster Mode:**
    * PySpark can be run in various cluster modes, such as local mode for testing on a single machine, standalone mode with its own cluster manager, or on existing clusters managed by Apache Mesos or Apache Hadoop YARN.
11. **Driver and Executor Communication:**
    * The driver program communicates with the executors to send tasks and collect results. Data is exchanged between the driver and executors during task execution.
12. **Shuffle Operations:**
    * Shuffle operations involve redistributing and exchanging data between partitions during certain operations, like groupByKey or reduceByKey. Efficient handling of shuffling is crucial for performance.

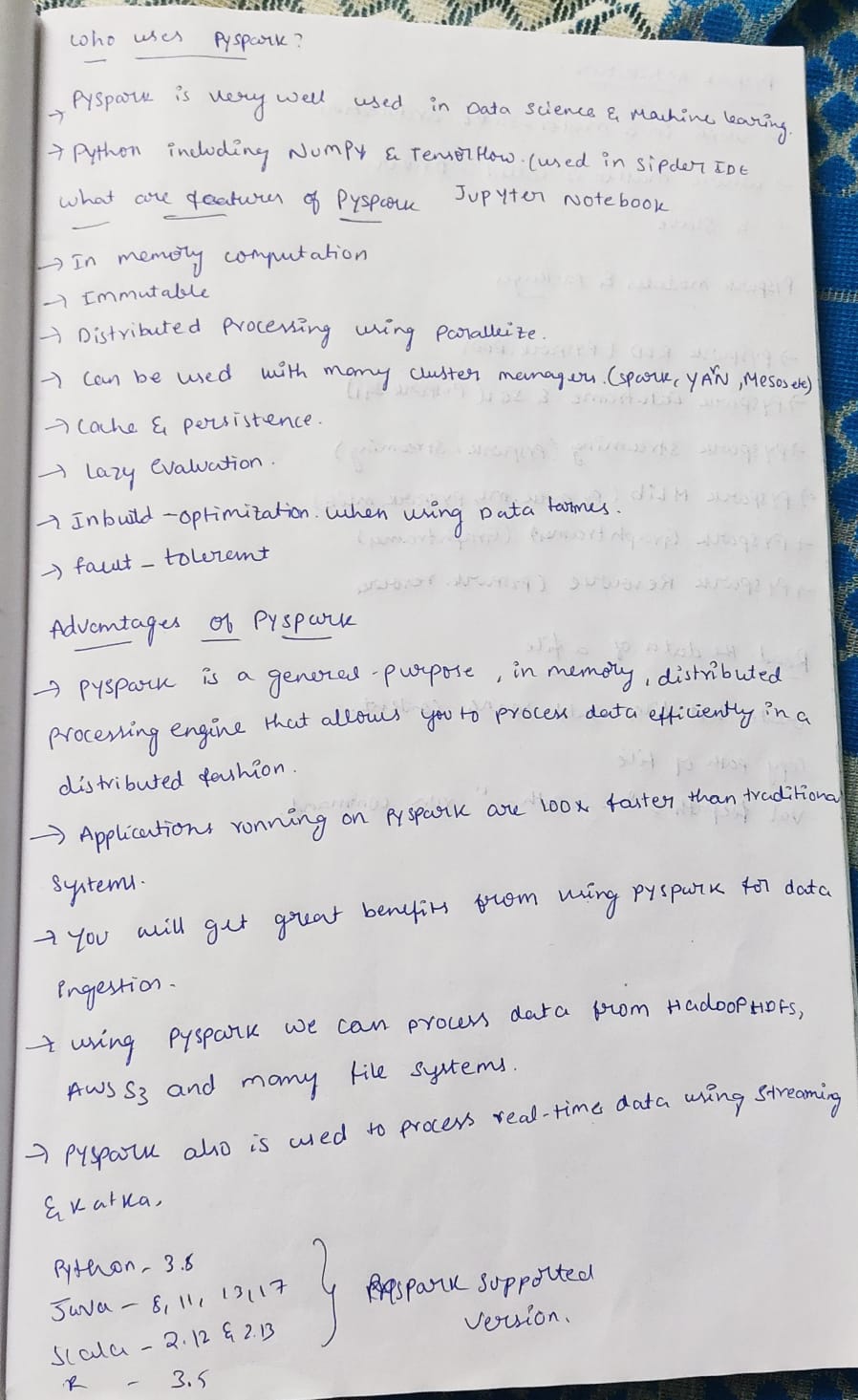
**Pyspark modules and packages: -**

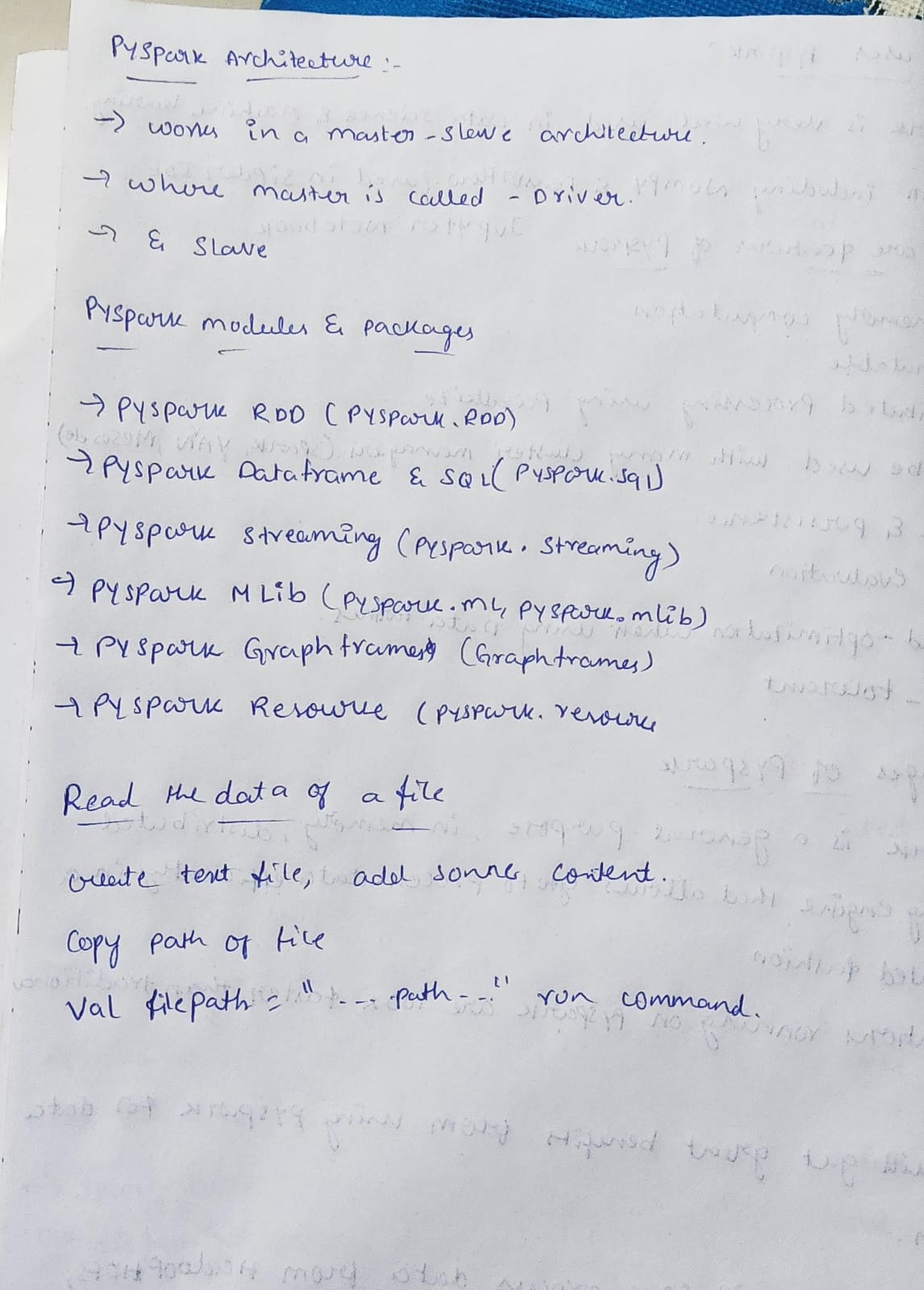
PySpark provides several modules and packages that cover various aspects of distributed data processing, machine learning, graph processing, and more. Some of the key modules and packages in PySpark include:

* **pyspark.sql:**
  + This module provides the DataFrame API and Spark SQL functionality. It allows users to work with structured data using a high-level, SQL-like interface. The **SparkSession** class, part of this module, is the entry point for using DataFrames and Spark SQL.
* **pyspark.rdd:**
* The RDD (Resilient Distributed Dataset) module is a fundamental part of PySpark, representing an immutable distributed collection of objects. It includes operations for creating, transforming, and processing RDDs.
* **pyspark. streaming:**
* This module supports real-time data processing through Spark Streaming. It enables the processing of live data streams and provides functionality for windowed computations and stateful processing.
* **pyspark.ml:**
* The **ml** module in PySpark provides the MLlib library, which includes a set of machine learning algorithms and utilities. It supports various tasks such as classification, regression, clustering, and collaborative filtering.
* **pyspark. mllib:**
* The **mllib** module is the predecessor to the **ml** module and provides machine learning algorithms in RDD-based API. While **mllib** is still available, it's recommended to use the newer **ml** module for DataFrames-based machine learning.
* **pyspark.graphx:**
* The **graphx** module supports graph processing using GraphX. It allows users to represent and process graph data efficiently.
* **pyspark.sql.functions:**
* The **functions** module provides a set of built-in functions that can be used with DataFrames. These functions are similar to SQL functions and enable users to perform various data manipulations.
* **pyspark.sql.types:**
* The **types** module provides data types that can be used to define the structure of the data in DataFrames.
* **pyspark.sql.window:**
* The **window** module supports window functions for DataFrames. Window functions allow users to perform calculations across a specified range of rows related to the current row.
* **pyspark.sql.streaming:**
* The **streaming** module provides functionality for structured streaming, allowing users to process data streams in a similar manner to batch processing.

**Notes Written: -**

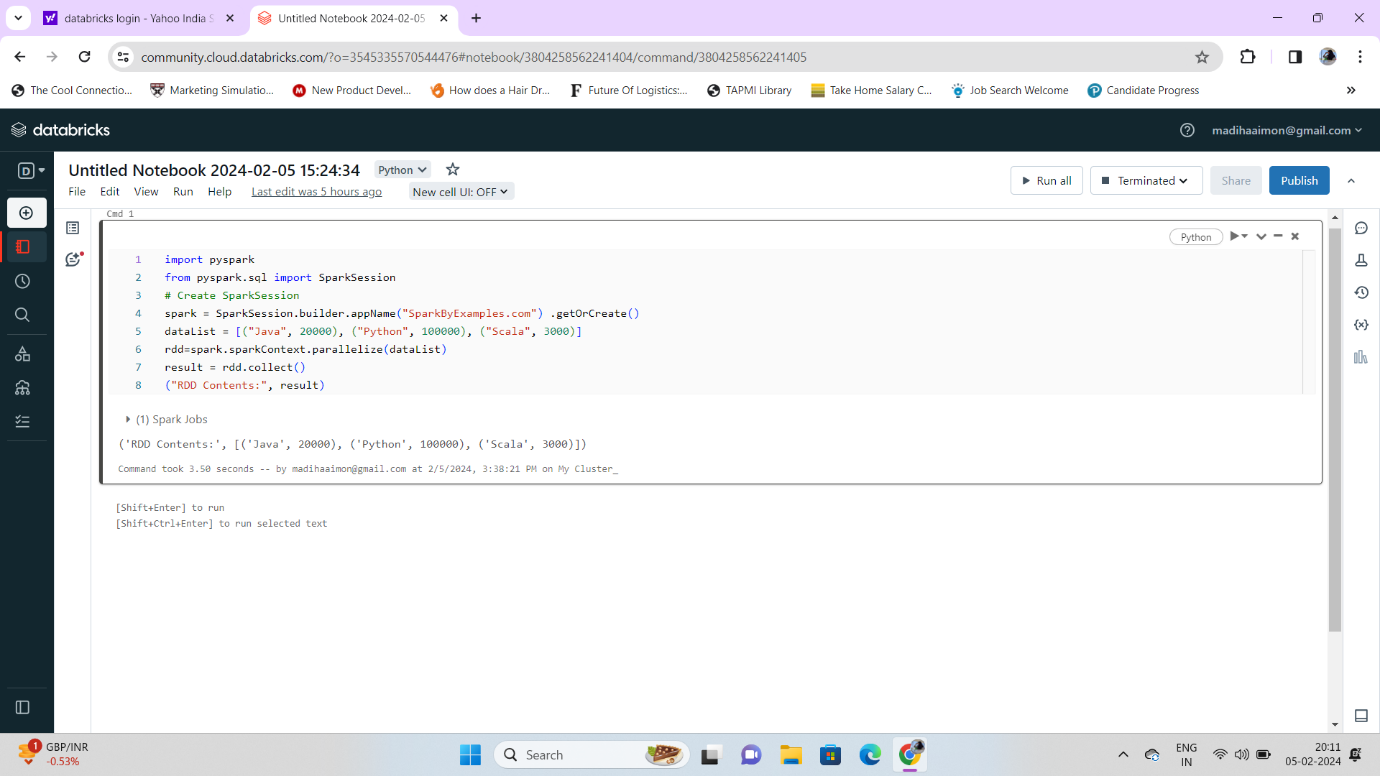


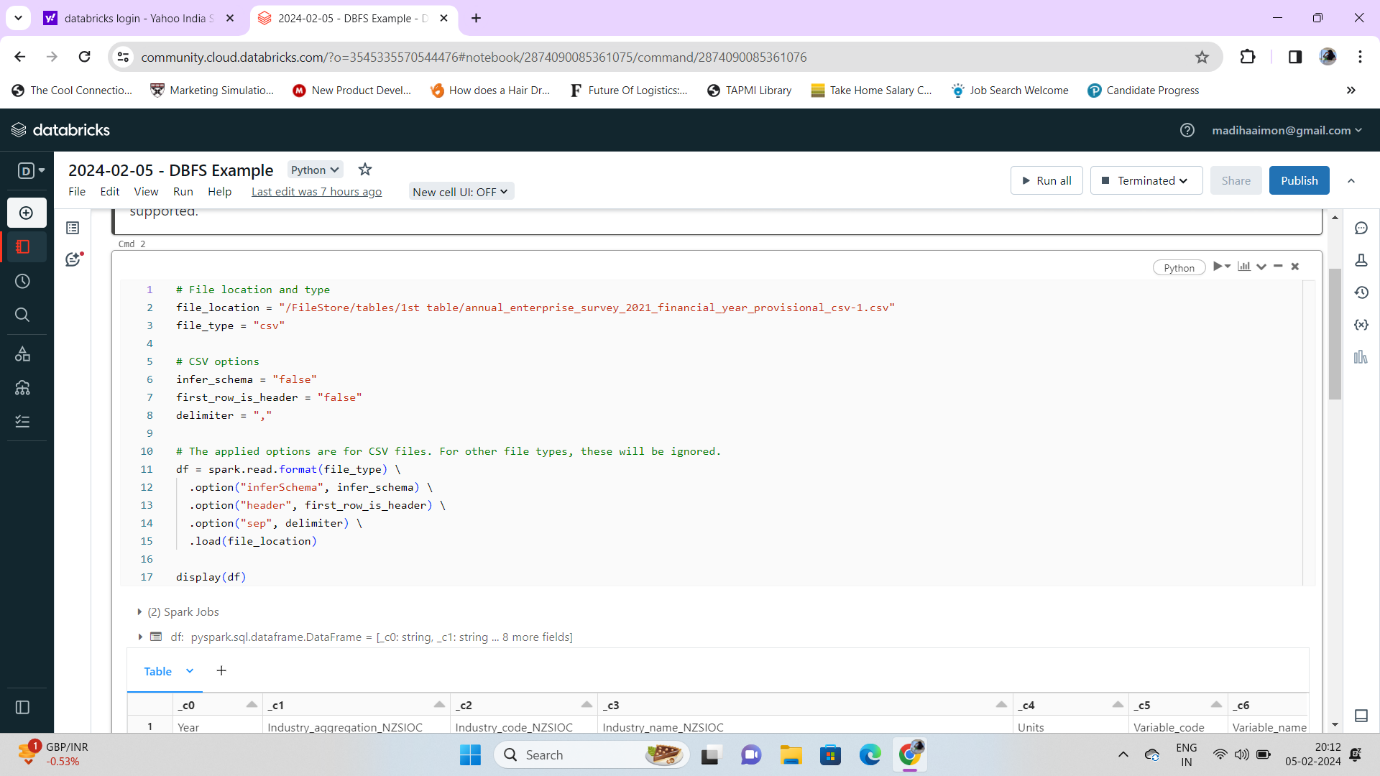


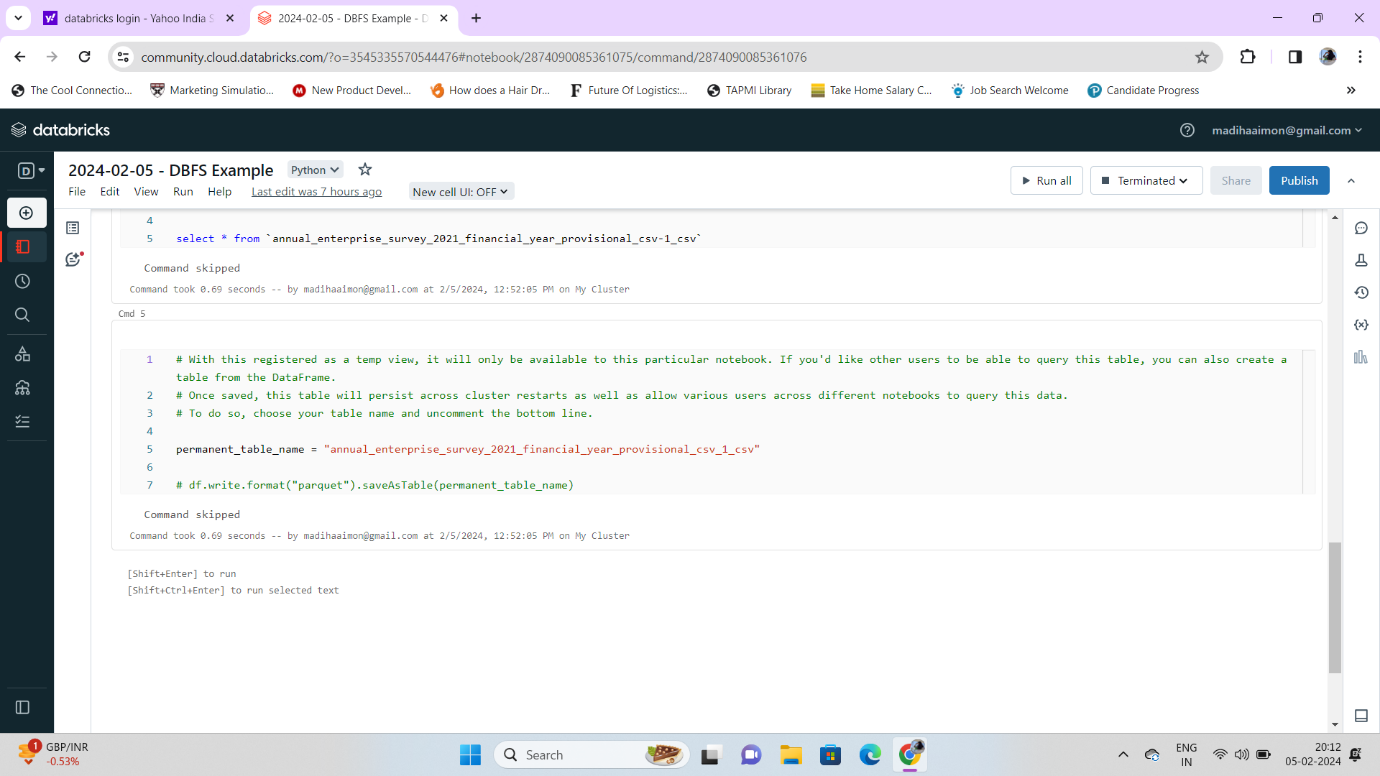
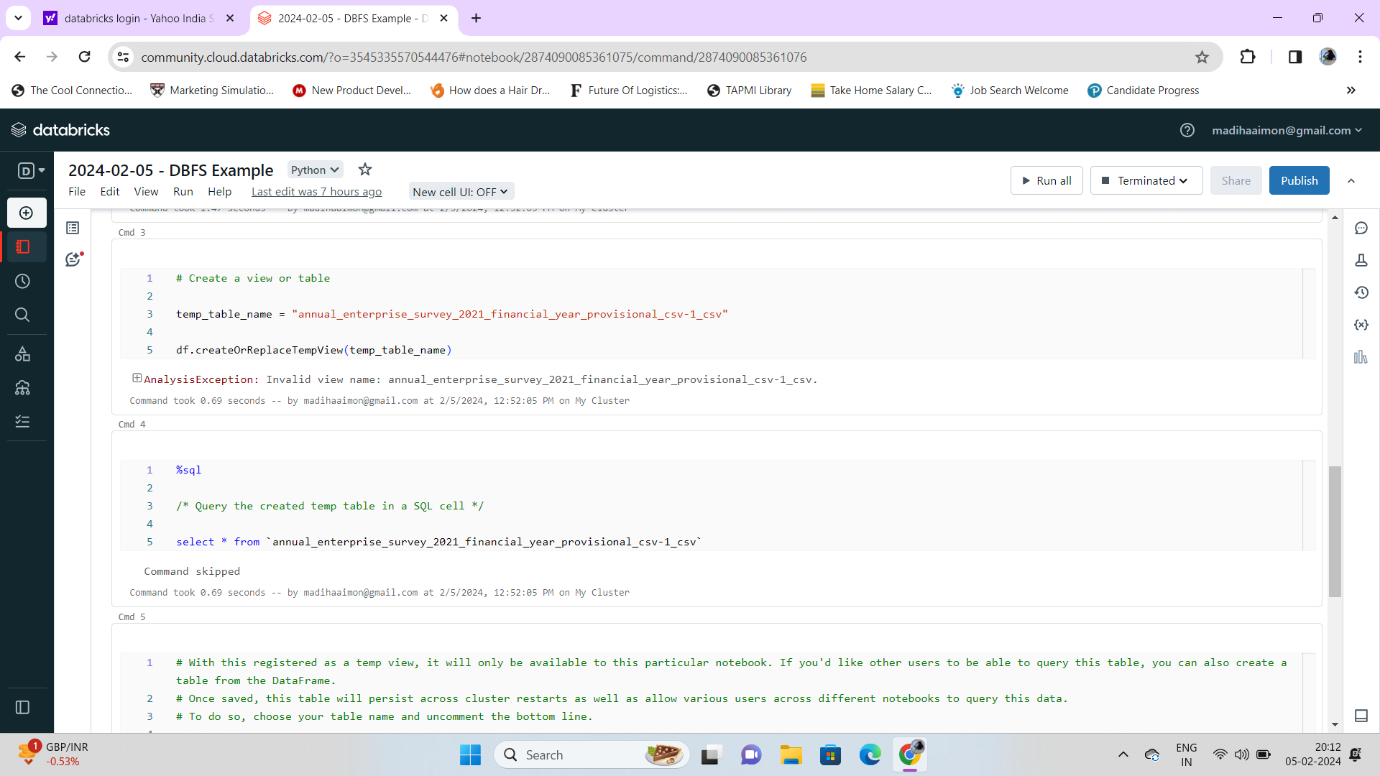
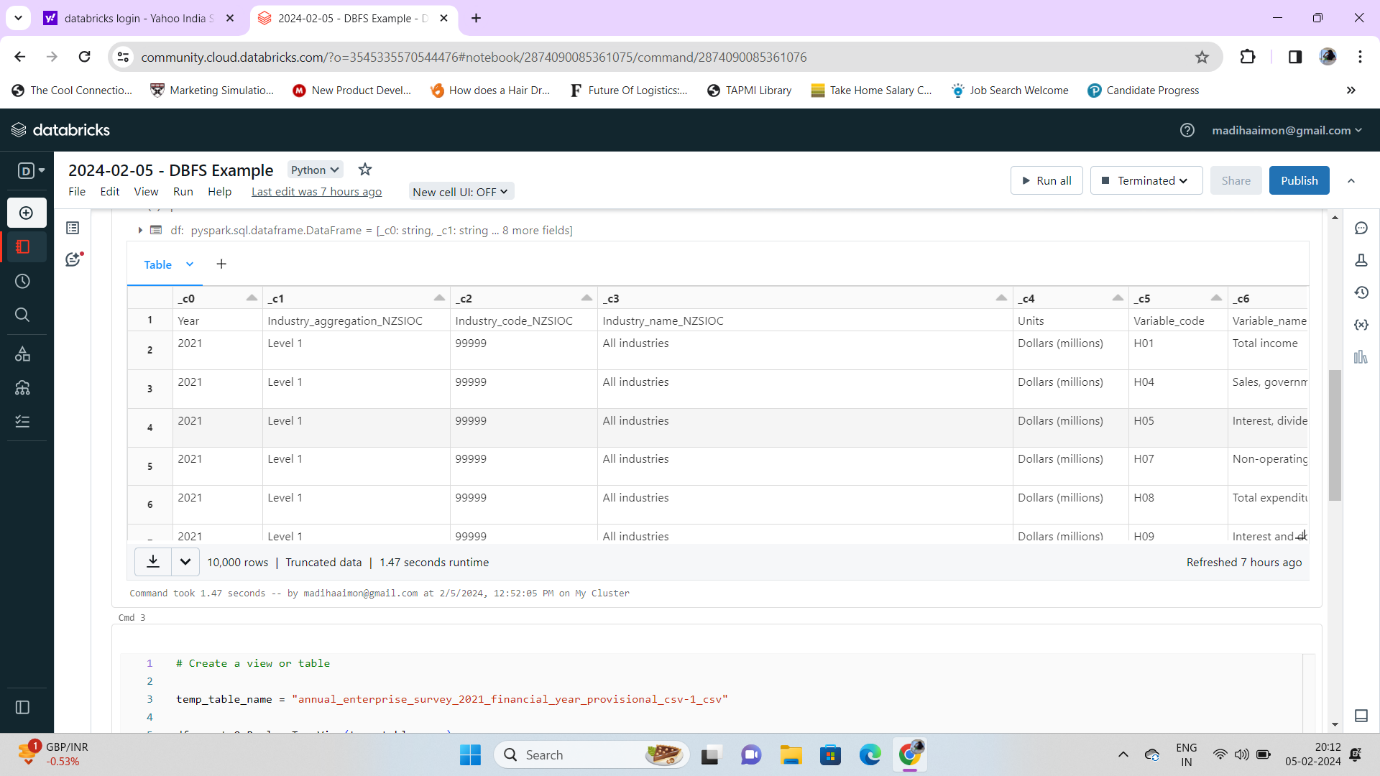


**HANDS ON PRACTICE: -**

* **DATABRICKS: -**

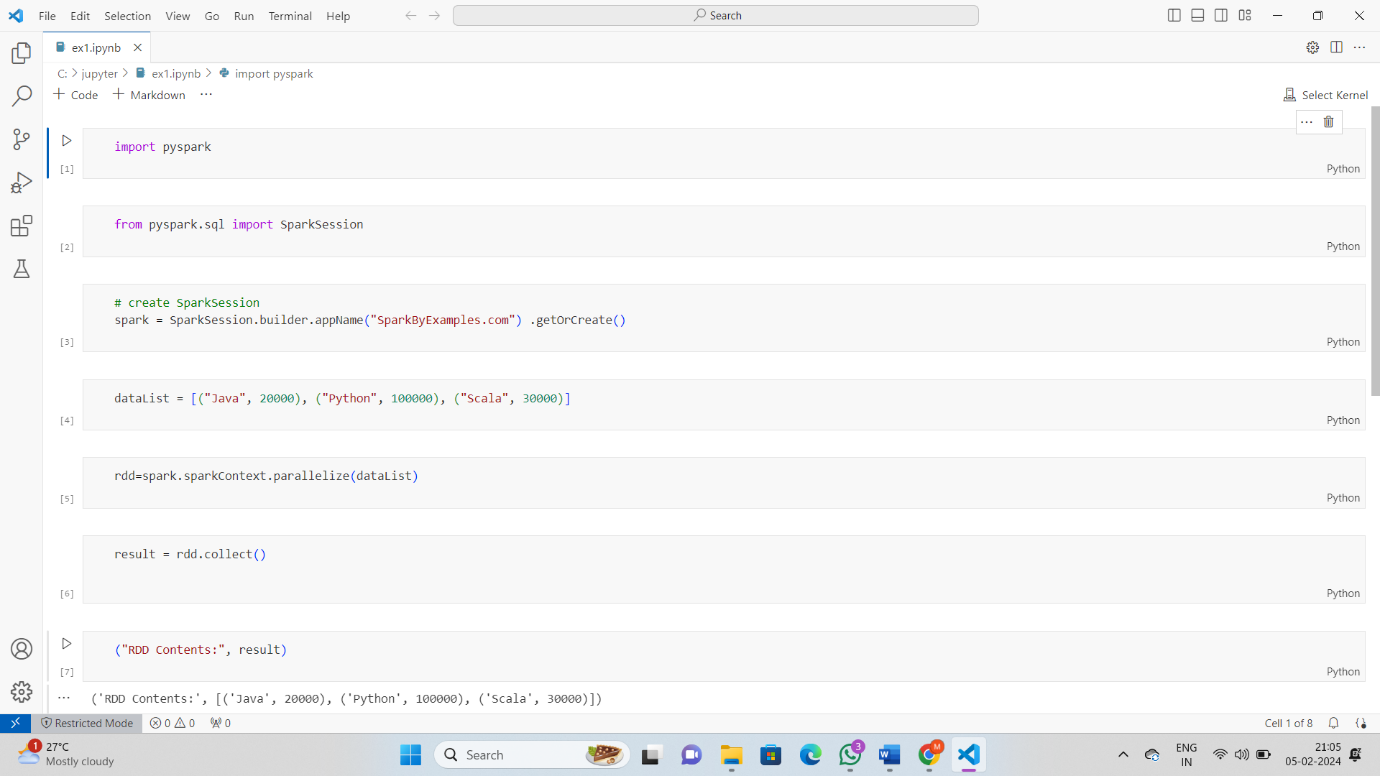
****

****

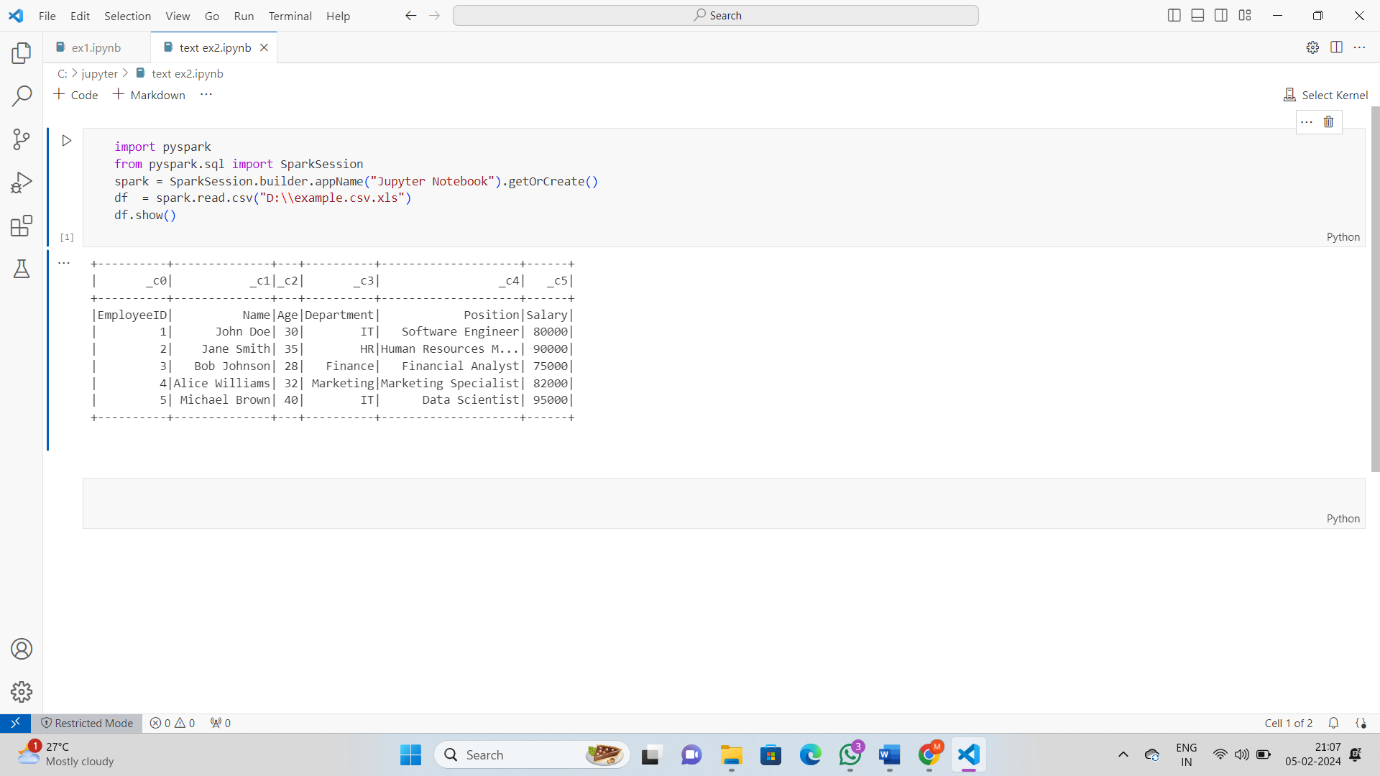
****

* **Jupyter Notebook: -**

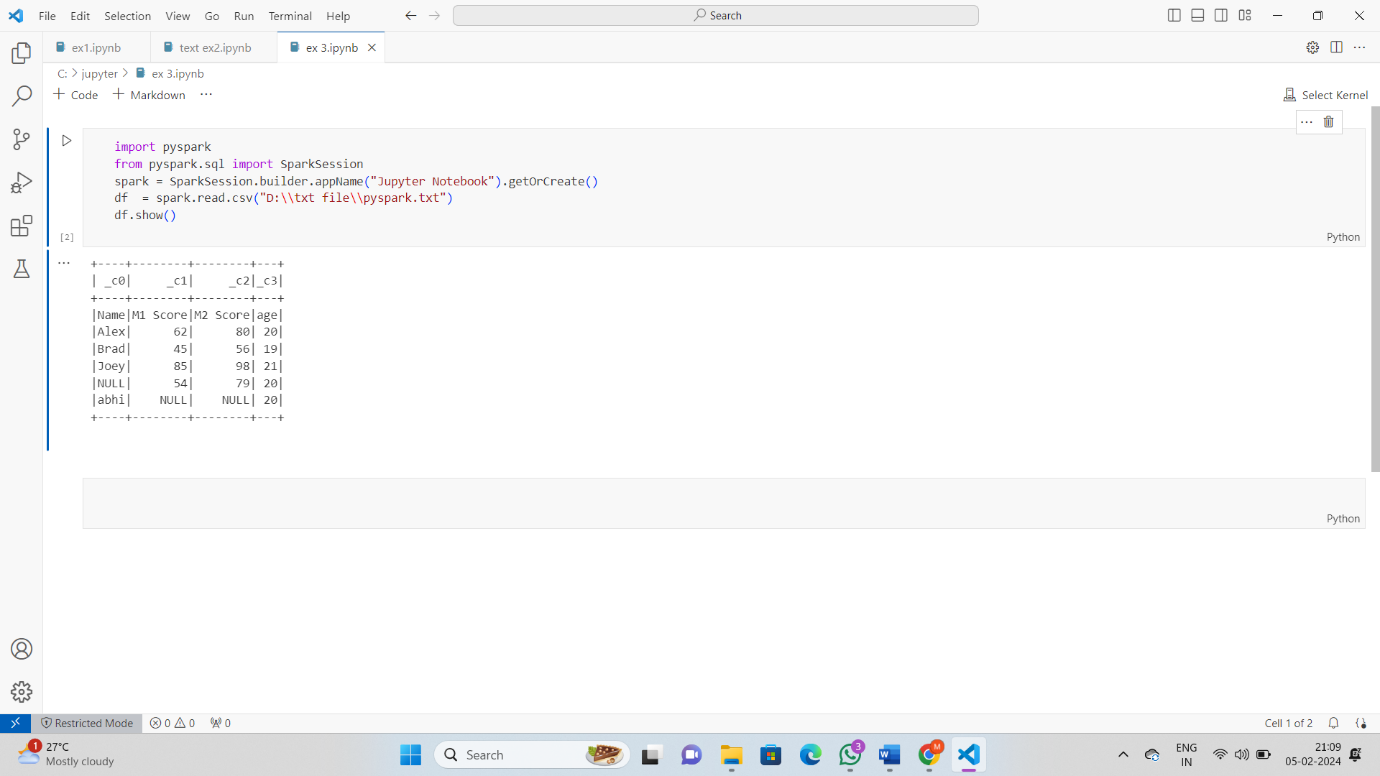
**EX: -1**

****

**EX: -2**

****

**Ex: - 3**

****