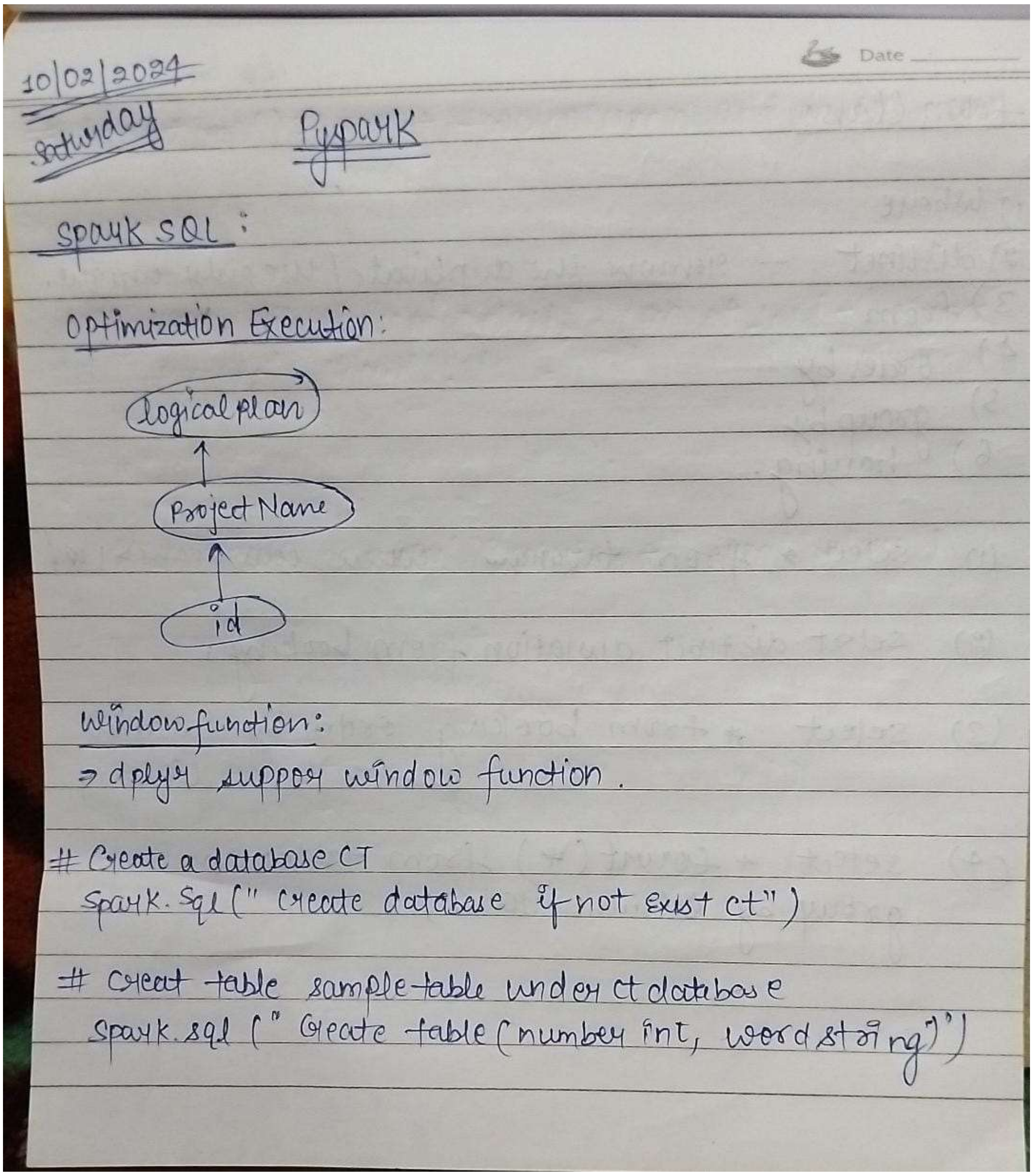
Name: Abhishek Kanoujia

DATA ENGINEERING BATCH 1

DAY 16 ASSIGNMENT

Class hand written notes:-



10|02|2024

Saturday

Pyspark

Spark SQL :

Optimization Execution:

$$\begin{tikzcd} \text{logical plan} \\ \text{Project Name} \arrow[u] \\ \text{id} \arrow[u] \end{tikzcd}$$

Window function:

⇒ dplyr suppor window function.

# Create a database CT

spark.sql(" create database if not exist ct")

# Creat table sample-table under ct database

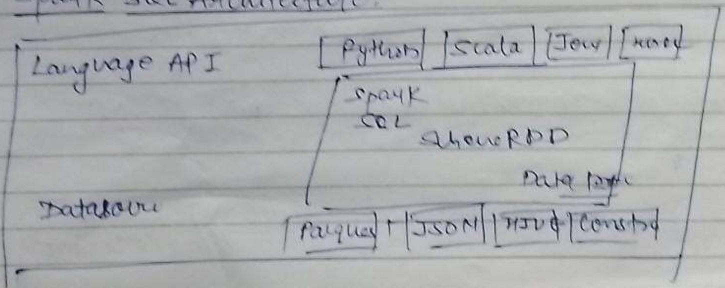
spark.sql (" Create table(number int, word string")")

Insert:
using the sampleview. spark.sql ("INSERT INTO TABLE
ct. sampleTable select * from sample view).

→ Spark SQL
→ setting up Azure lab

Spark SQL:
→ module for structure data processing
→ spark SQL is component spark core that Introduce a new data abstraction called RDD

Spark SQL Architecture:



⇒ Language API
⇒ schema RDD
⇒ Data source

feature of SQL
⇒ Integrated
⇒ unified Data Access
⇒ Have compatibility
⇒ scalibility

UDF (User defined functions)

~~UDF (User defined~~
Tableaue → Qlide.

Spark RDD:
→ fundamental data structure of spark
⇒ immutable distributed collection of object that is store in memory disk
⇒ Parallel functional transformation

Date

⇒ Automatically rebuild Architecure

⇒ RDD contains any type python; Java, Scala, including user defined classes.

⇒ formally on RDD is read only

⇒ RDD is fault tolerace

Dataset & Dataframe

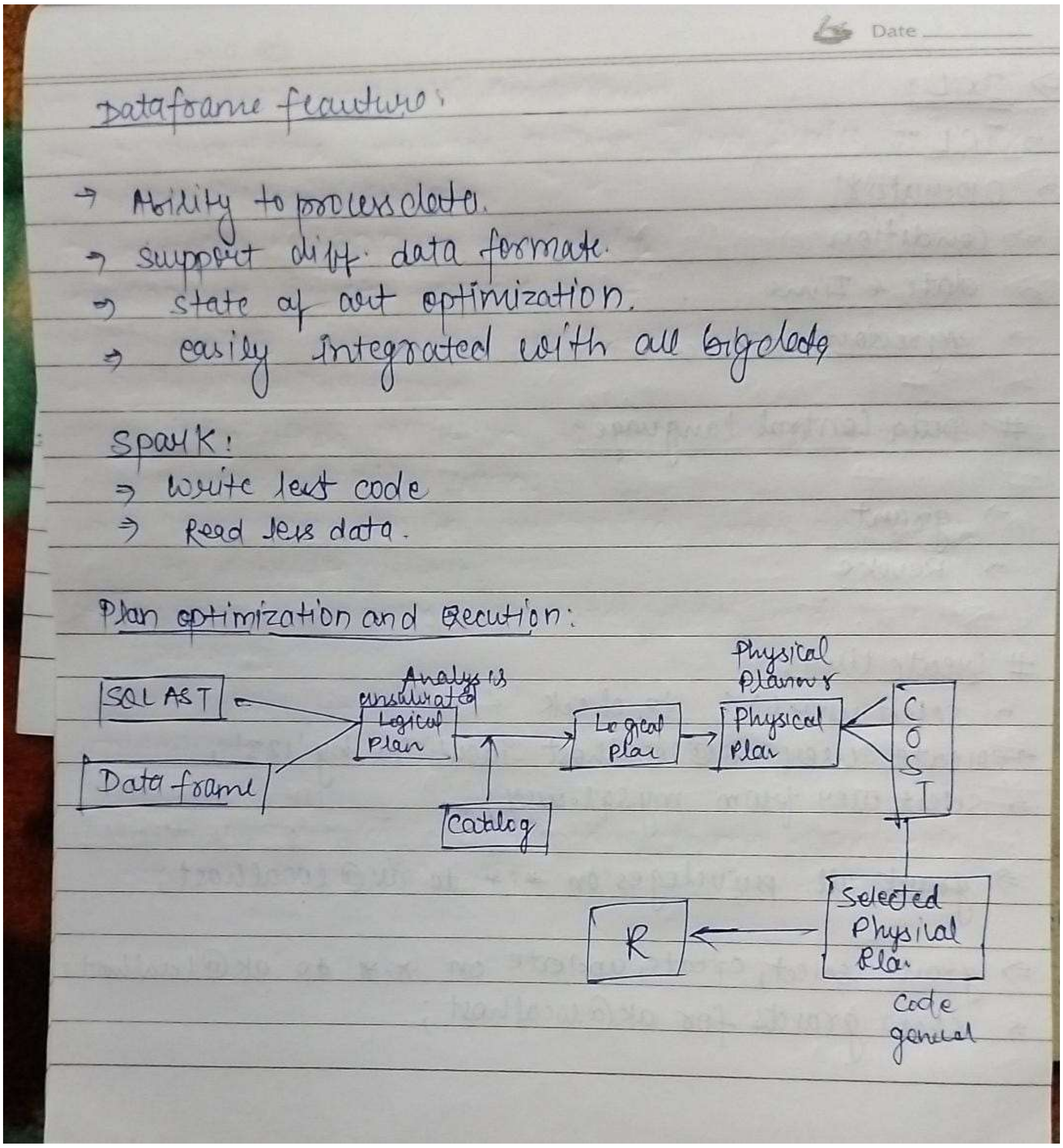
⇒ A distribut~~ed~~ion collection of data which organised into name and column.

⇒ data frame can be constructed from an array of different source.

⇒ such as five table, structured data field, external databases.

Dataframe:

Data is organised into named column like table in relocation database



Date

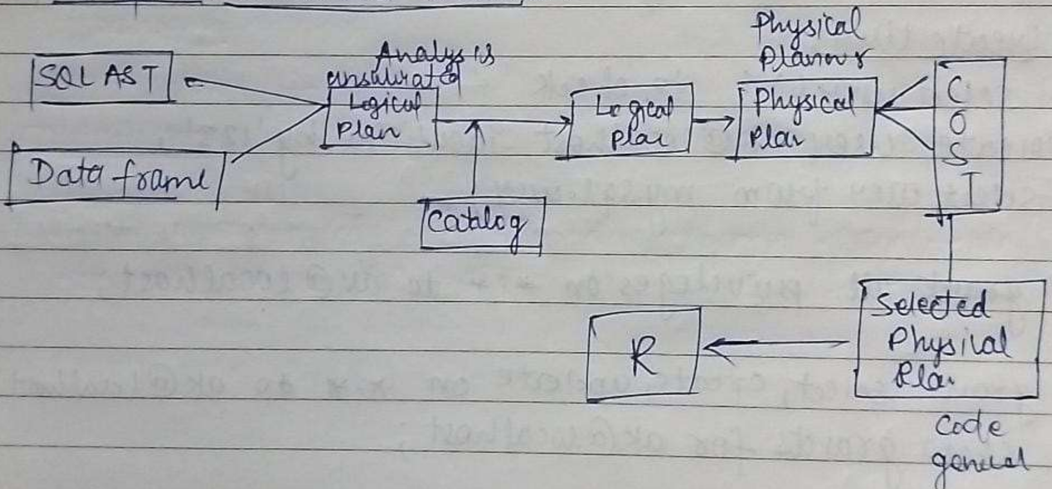
Dataframe features:

- → Ability to process data.
- → support diff. data formate.
- → state of art optimization.
- → easily integrated with all bigdata

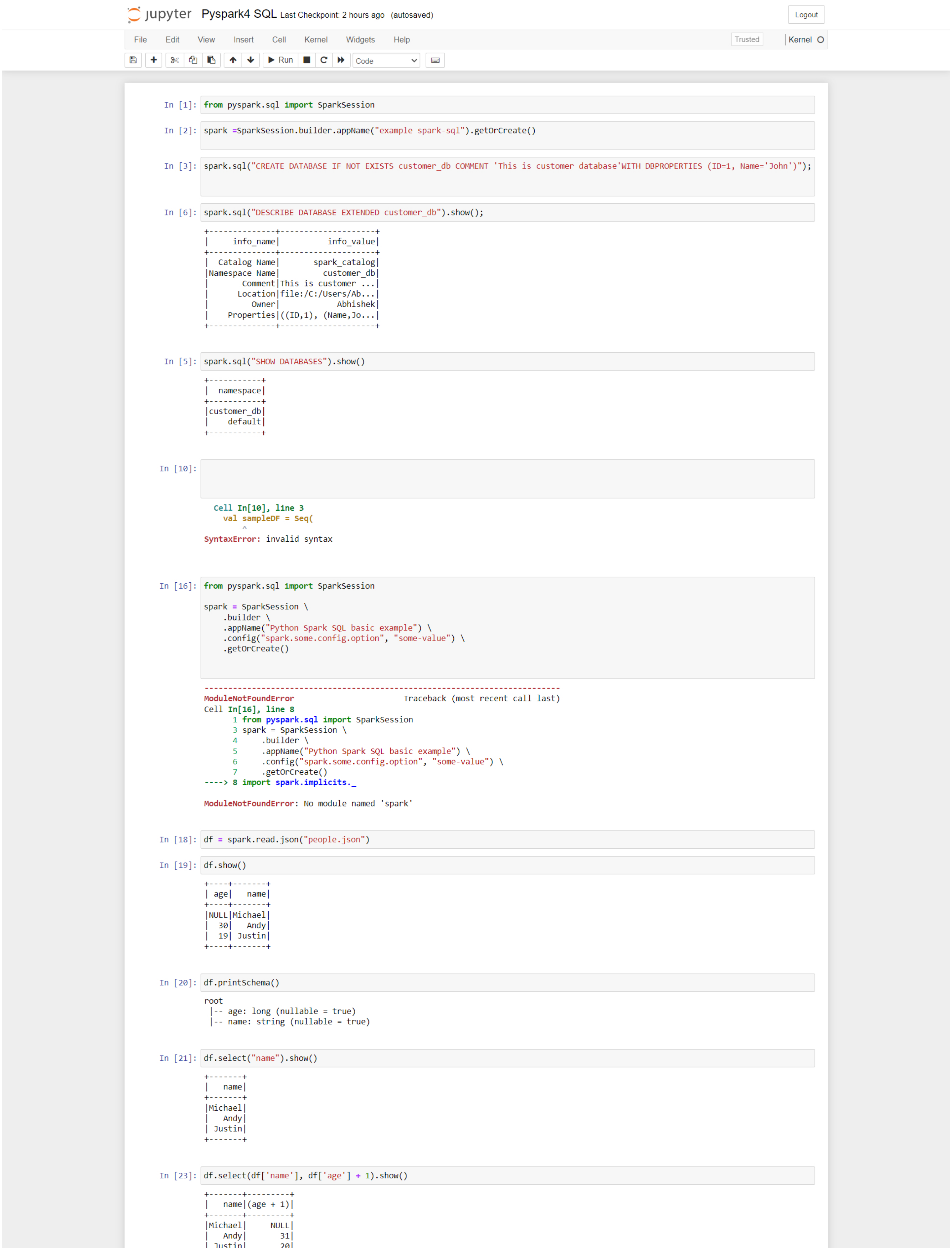
Spark:

- → write less code
- → Read less data.

Plan optimization and Execution:



Hands on Spark SQL:-


jupyter
Pyspark4 SQL Last Checkpoint: 2 hours ago (autosaved)
Logout
File Edit View Insert Cell Kernel Widgets Help
Trusted
Kernel
Run
Code

In [1]:
```
from pyspark.sql import SparkSession
```

In [2]:
```
spark =SparkSession.builder.appName("example spark-sql").getOrCreate()
```

In [3]:
```
spark.sql("CREATE DATABASE IF NOT EXISTS customer_db COMMENT 'This is customer database'WITH DBPROPERTIES (ID=1, Name='John')");
```

In [6]:
```
spark.sql("DESCRIBE DATABASE EXTENDED customer_db").show();
```

```
+--------------+-------------------+
|     info_name|         info_value|
+--------------+-------------------+
|  Catalog Name|      spark_catalog|
|Namespace Name|        customer_db|
|       Comment|This is customer ...|
|      Location|file:/C:/Users/Ab...|
|         Owner|           Abhishek|
|    Properties|((ID,1), (Name,Jo...|
+--------------+-------------------+
```

In [5]:
```
spark.sql("SHOW DATABASES").show()
```

```
+-----------+
|  namespace|
+-----------+
|customer_db|
|    default|
+-----------+
```

In [10]:
```

```

```
  Cell In[10], line 3
    val sampleDF = Seq(
        ^
SyntaxError: invalid syntax
```

In [16]:
```
from pyspark.sql import SparkSession

spark = SparkSession \
    .builder \
    .appName("Python Spark SQL basic example") \
    .config("spark.some.config.option", "some-value") \
    .getOrCreate()
```

```
---------------------------------------------------------------------------
ModuleNotFoundError                       Traceback (most recent call last)
Cell In[16], line 8
      1 from pyspark.sql import SparkSession
      3 spark = SparkSession \
      4     .builder \
      5     .appName("Python Spark SQL basic example") \
      6     .config("spark.some.config.option", "some-value") \
      7     .getOrCreate()
----> 8 import spark.implicits._

ModuleNotFoundError: No module named 'spark'
```

In [18]:
```
df = spark.read.json("people.json")
```

In [19]:
```
df.show()
```

```
+----+-------+
| age|   name|
+----+-------+
|NULL|Michael|
|  30|   Andy|
|  19| Justin|
+----+-------+
```

In [20]:
```
df.printSchema()
```

```
root
 |-- age: long (nullable = true)
 |-- name: string (nullable = true)
```

In [21]:
```
df.select("name").show()
```

```
+-------+
|   name|
+-------+
|Michael|
|   Andy|
| Justin|
+-------+
```

In [23]:
```
df.select(df['name'], df['age'] + 1).show()
```

```
+-------+---------+
|   name|(age + 1)|
+-------+---------+
|Michael|     NULL|
|   Andy|       31|
| Justin|       20|
+-------+---------+
```

In [24]:
```
df.filter(df['age'] > 21).show()
```

```
+---+----+
|age|name|
+---+----+
| 30|Andy|
+---+----+
```

In [26]:
```
df.groupBy("age").count().show()
```

```
+----+-----+
| age|count|
+----+-----+
|  19|    1|
|NULL|    1|
|  30|    1|
+----+-----+
```

In [27]:
```
df.createOrReplaceTempView("people")
```

In [28]:
```
sqlDF = spark.sql("SELECT * FROM people")
sqlDF.show()
```

```
+----+-------+
| age|   name|
+----+-------+
|NULL|Michael|
|  30|   Andy|
|  19| Justin|
+----+-------+
```

In [29]:
```
df.createGlobalTempView("people")
```

In [30]:
```
spark.sql("SELECT * FROM global_temp.people").show()
```

```
+----+-------+
| age|   name|
+----+-------+
|NULL|Michael|
|  30|   Andy|
|  19| Justin|
+----+-------+
```

In [32]:
```
spark.newSession().sql("SELECT * FROM global_temp.people").show()
```

```
+----+-------+
| age|   name|
+----+-------+
|NULL|Michael|
|  30|   Andy|
|  19| Justin|
+----+-------+
```

In [ ]: