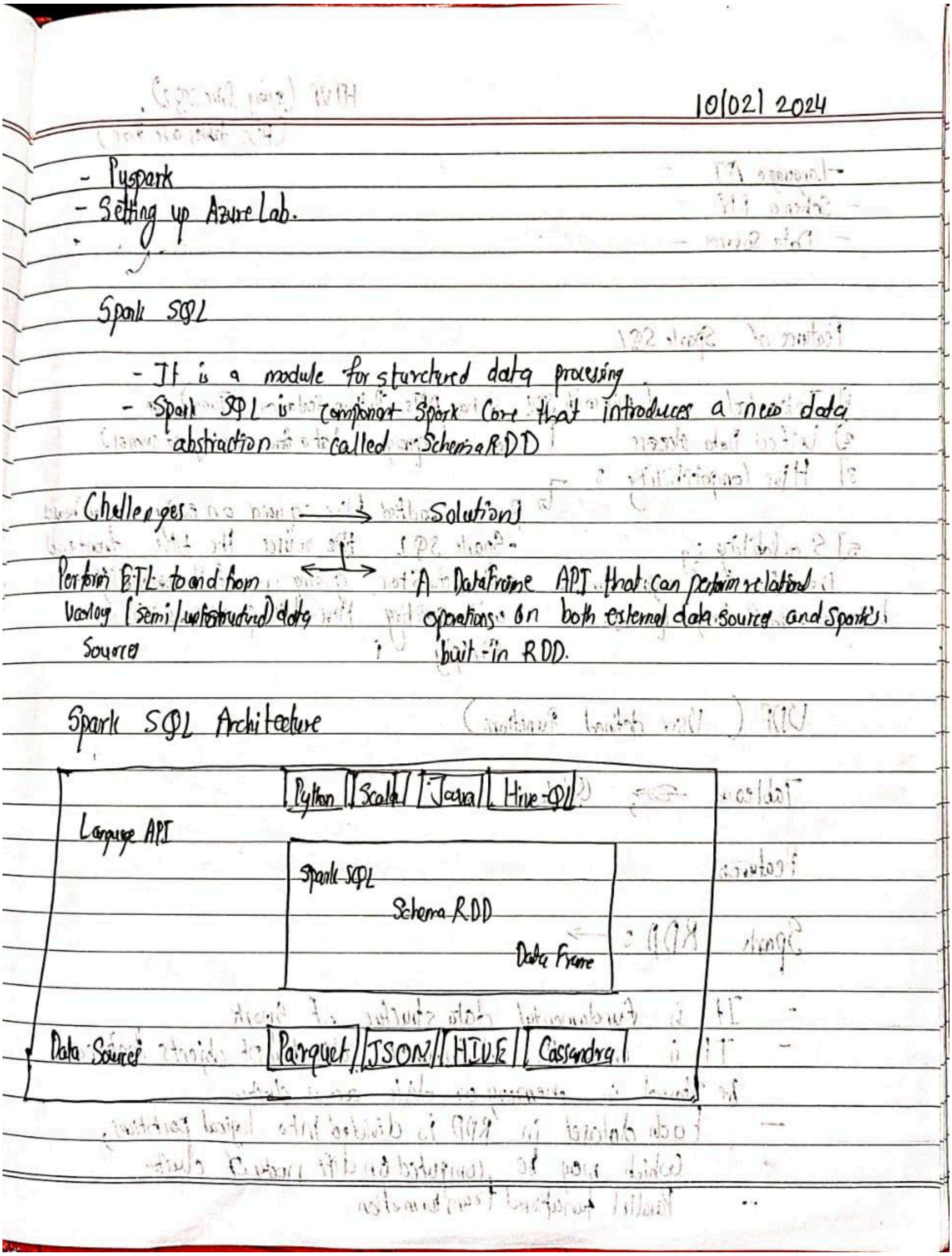
Name : Parth Nandedkar
Date : 10 Feb 2024
Topics : PySpark
Batch : Data Engineering Batch-1

Handwritten Notes :



10|02| 2024

- Pyspark
- Setting up Azure Lab.

Spark SQL

- It is a module for sturctured data processing
- Spark SQL is component Spark Core that introduces a new data abstraction called SchemaRDD

Challenges	Solutions
Perform ETL to and from Various (semi/unstructured) data Source	A DataFrame API that can perform relational operations on both external data source and Spark's built-in RDD.

Spark SQL Architecture

Language API

Python | Scala | Java | Hive-QL

Spark SQL

Schema RDD

Data Frame

Data Source

Parquet | JSON | HIVE | Cassandra

HIVE (query like SQL).
(Hive tables are there)

- Language API -
- Schema RDD -
- Data Sources -

Features of Spark SQL

1] Integrated (mix with SQL & has APIs Python, Scala, and Java) can -
2] Unified Data Access (Load and query data from variety of sources)
3] Hive Compatibility : → Run unmodified Hive queries on existing warehouses
- Spark SQL ~~the~~ reuses the Hive frontend and MetaStore, giving you full compatibility with existing Hive data, queries & UDFs
5] Scalability :-
Use the same engine for both interactive & long queries.
Simply

UDF (User defined functions)

Tableau ~~Qo~~, Qlik

Features

Sparks RDD : →

- It is fundamental data structure of Spark
- It is immutable distributed collection of objects that can be stored in memory or disk across cluster
- Each dataset in RDD is divided into logical partitions, which may be computed on diff nodes of cluster
- Parallel functional transformation.

- Automatically rebuild failures
- RDD contains any type Python, Java, Scala objects including user defined classes

- Formally, an RDD is read-only, partitioned collection of records.

- RDDs can be created through deterministic operations on either data on stable storage or other RDD

- RDD is fault-tolerant collection of elements that can be operated on in parallel.

Dataset & Dataframe :

- A ~~da~~ distributed collection of data, which organized into named columns.
- Conceptually, it is equivalent to relational tables with good optimization techniques.
- A DataFrame can be constructed from an array of different sources such as Hive tables, Structured data files, external databases.
- This API was designed for modern Big data

Data frame →

Data is organized into named columns, like table in a relational ~~das~~ database.

DataFrames

Features of DataFrame :

- Ability to process the data in the size of kilobytes to Petabytes on a single node s cluster to large cluster.

- Supports diff data formates (Avro, csv, elastic search, Casendra) and storage systems (HDFS, HIVE, mysql etc)

- State of art optimization & code generation through the Spark SQL Catalyst optimizer (tree transformation framework)

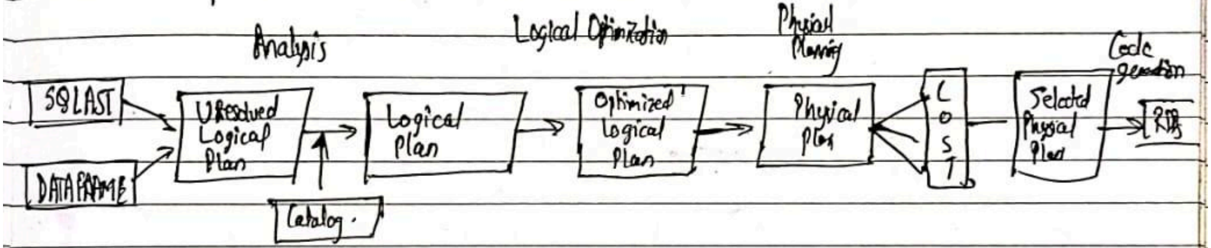
- Can be easily integrated with all Big data tools and frameworks via Spark core.

- Provides API for Python, Java, Scala & R programing.

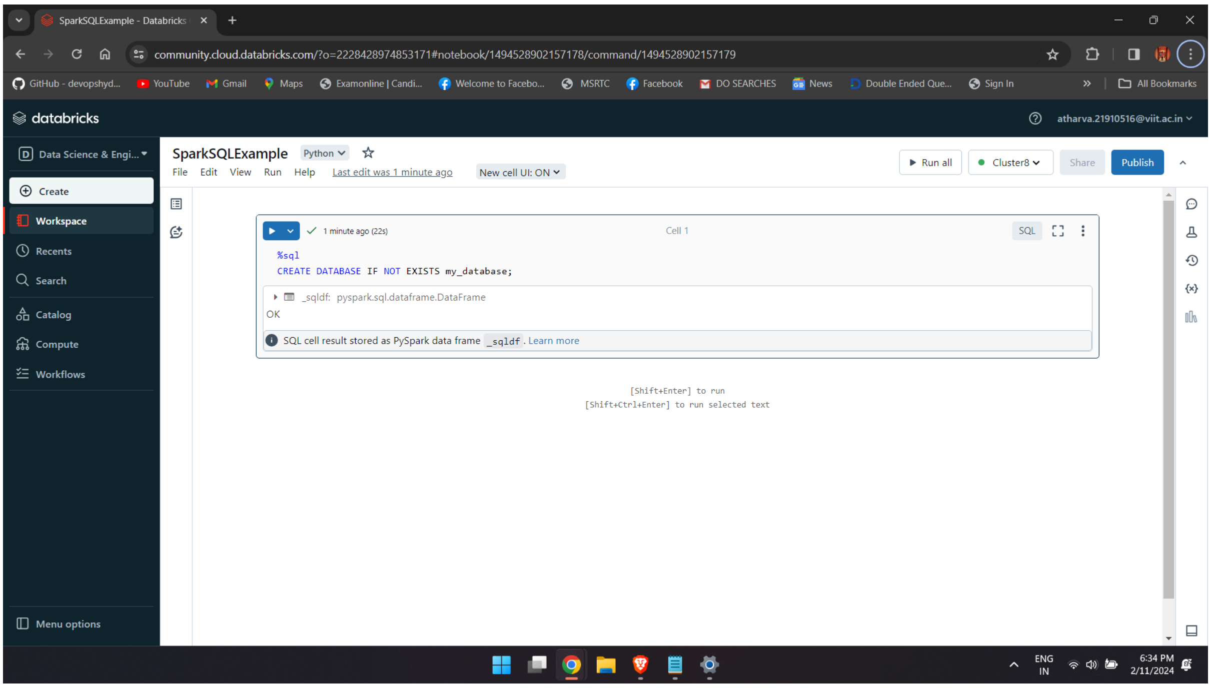
Spark SQL

1. Write less code
2. Read less data
3.

Plan Optimization & Execution :



Creating Database using Spark SQL :



SparkSQLExample

Cell 1 — 1 minute ago (22s)

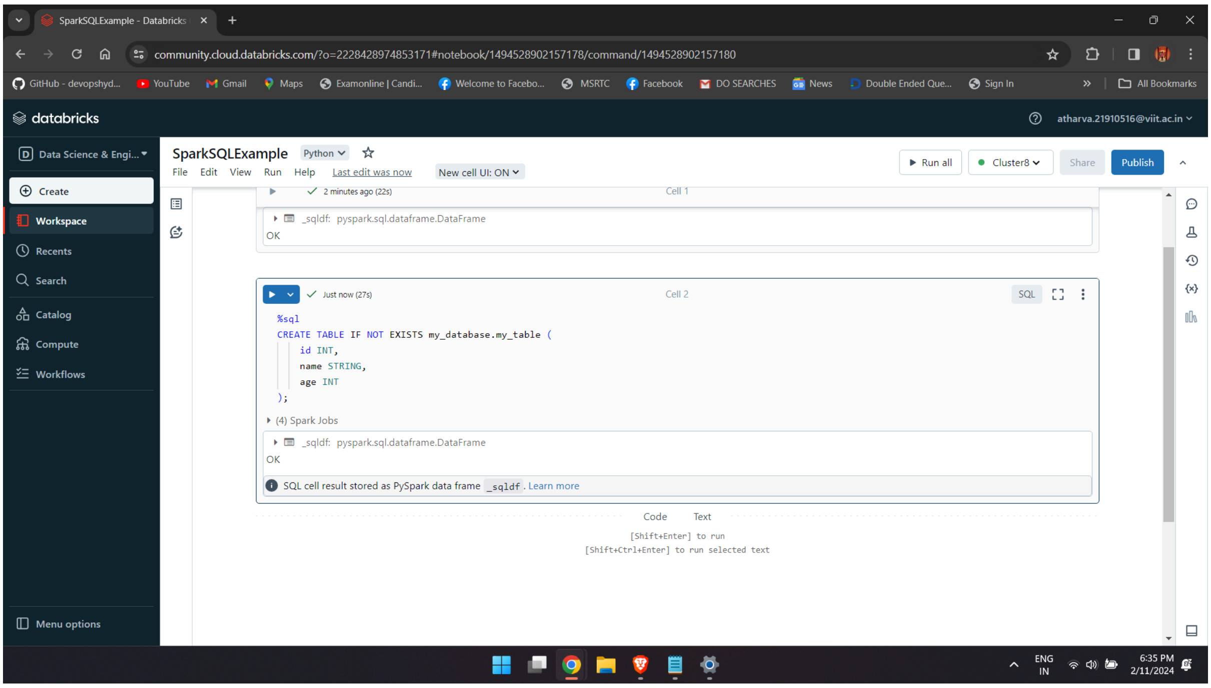
```
%sql
CREATE DATABASE IF NOT EXISTS my_database;
```

```
_sqldf:  pyspark.sql.dataframe.DataFrame
OK
```

SQL cell result stored as PySpark data frame `_sqldf`. Learn more

```
[Shift+Enter] to run
[Shift+Ctrl+Enter] to run selected text
```

Creating Table using Spark SQL :



SparkSQLExample

Cell 1 — 2 minutes ago (22s)

```
_sqldf:  pyspark.sql.dataframe.DataFrame
OK
```

Cell 2 — Just now (27s)

```
%sql
CREATE TABLE IF NOT EXISTS my_database.my_table (
    id INT,
    name STRING,
    age INT
);
```

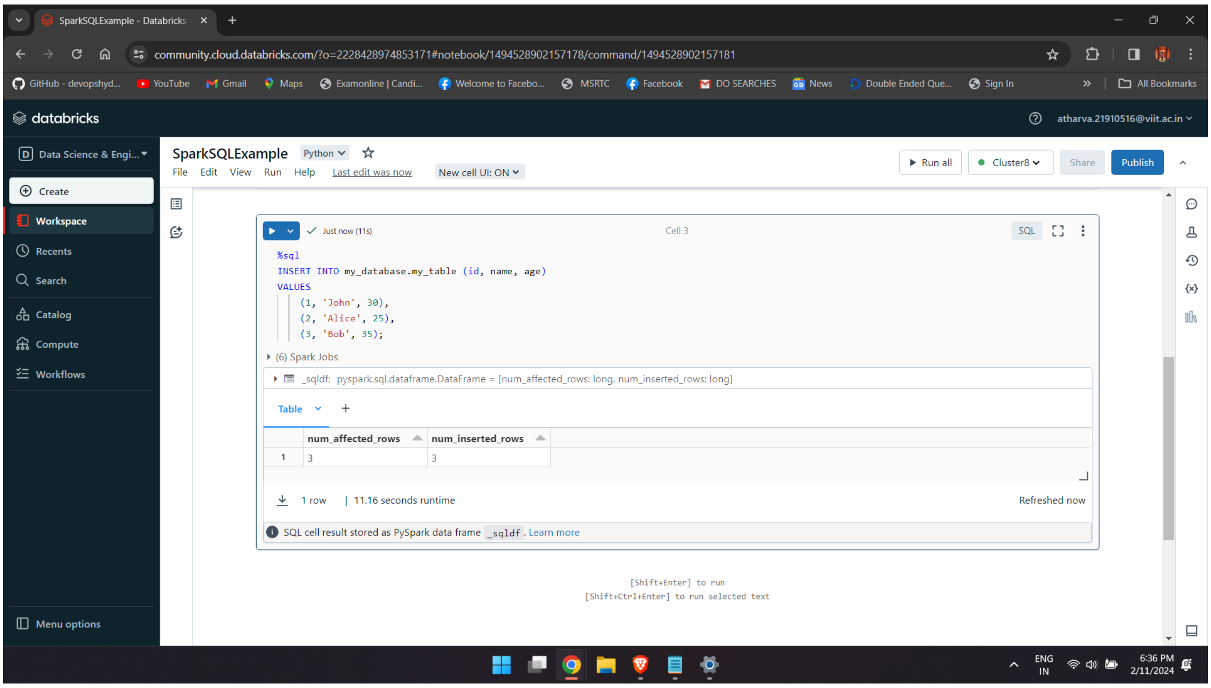
(4) Spark Jobs

```
_sqldf:  pyspark.sql.dataframe.DataFrame
OK
```

SQL cell result stored as PySpark data frame `_sqldf`. Learn more

```
[Shift+Enter] to run
[Shift+Ctrl+Enter] to run selected text
```

Adding data to table :



```
%sql
INSERT INTO my_database.my_table (id, name, age)
VALUES
    (1, 'John', 30),
    (2, 'Alice', 25),
    (3, 'Bob', 35);
```

(6) Spark Jobs

_sqldf: pyspark.sql.dataframe.DataFrame = [num_affected_rows: long, num_inserted_rows: long]

Table

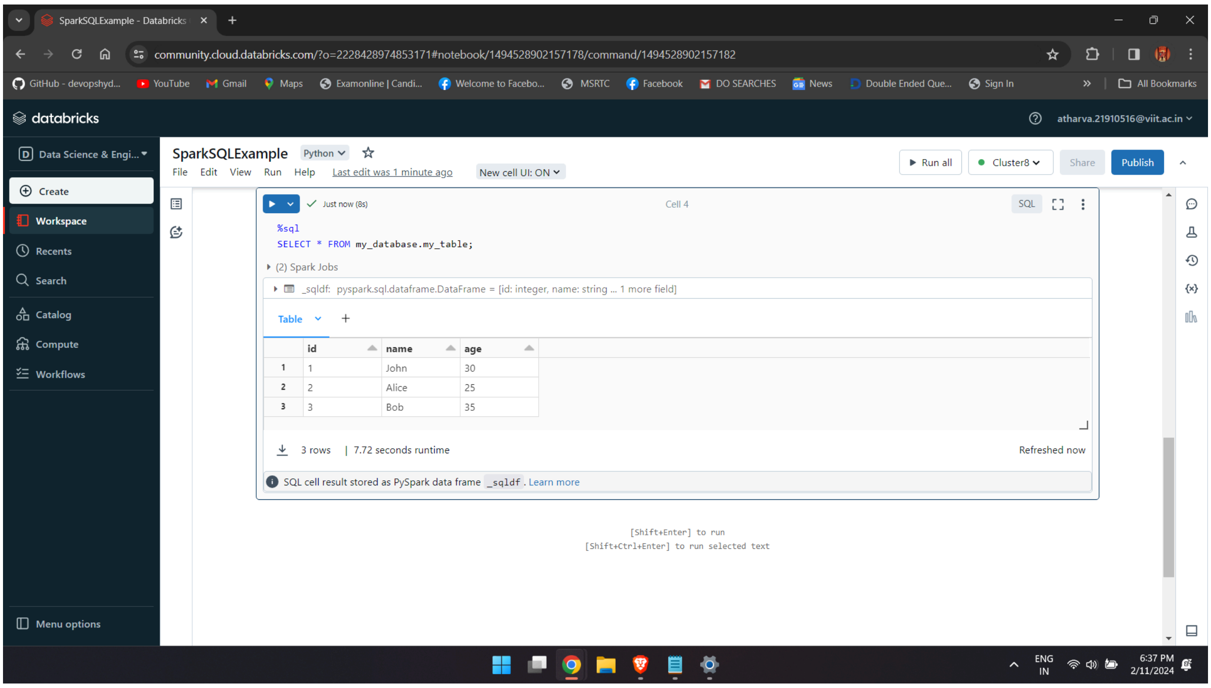
	num_affected_rows	num_inserted_rows
1	3	3

1 row | 11.16 seconds runtime   Refreshed now

SQL cell result stored as PySpark data frame `_sqldf`. Learn more

```
[Shift+Enter] to run
[Shift+Ctrl+Enter] to run selected text
```

Fetching Data from table :



```
%sql
SELECT * FROM my_database.my_table;
```

(2) Spark Jobs

_sqldf: pyspark.sql.dataframe.DataFrame = [id: integer, name: string ... 1 more field]

Table

	id	name	age
1	1	John	30
2	2	Alice	25
3	3	Bob	35

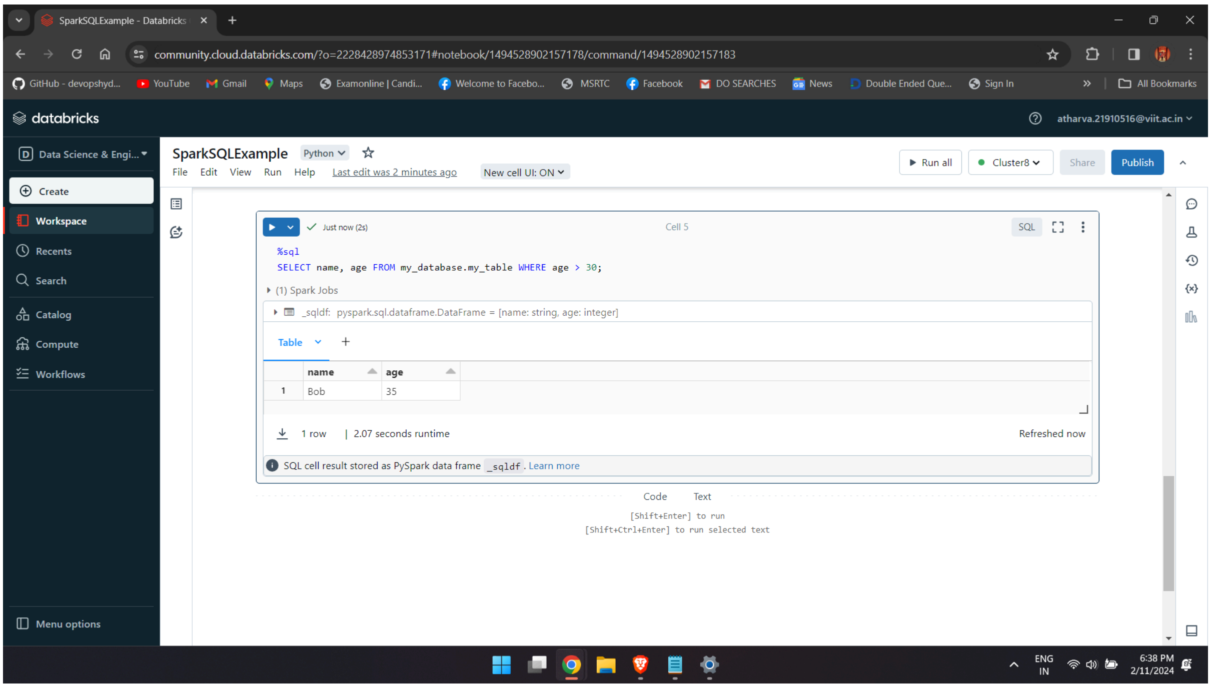
3 rows | 7.72 seconds runtime

Refreshed now

SQL cell result stored as PySpark data frame `_sqldf`. Learn more

```
[Shift+Enter] to run
[Shift+Ctrl+Enter] to run selected text
```

Filtering Data from table :



```
%sql
SELECT name, age FROM my_database.my_table WHERE age > 30;
```

(1) Spark Jobs

_sqldf: pyspark.sql.dataframe.DataFrame = [name: string, age: integer]

Table

	name	age
1	Bob	35

1 row | 2.07 seconds runtime

Refreshed now

SQL cell result stored as PySpark data frame `_sqldf`. Learn more

```
[Shift+Enter] to run
[Shift+Ctrl+Enter] to run selected text
```