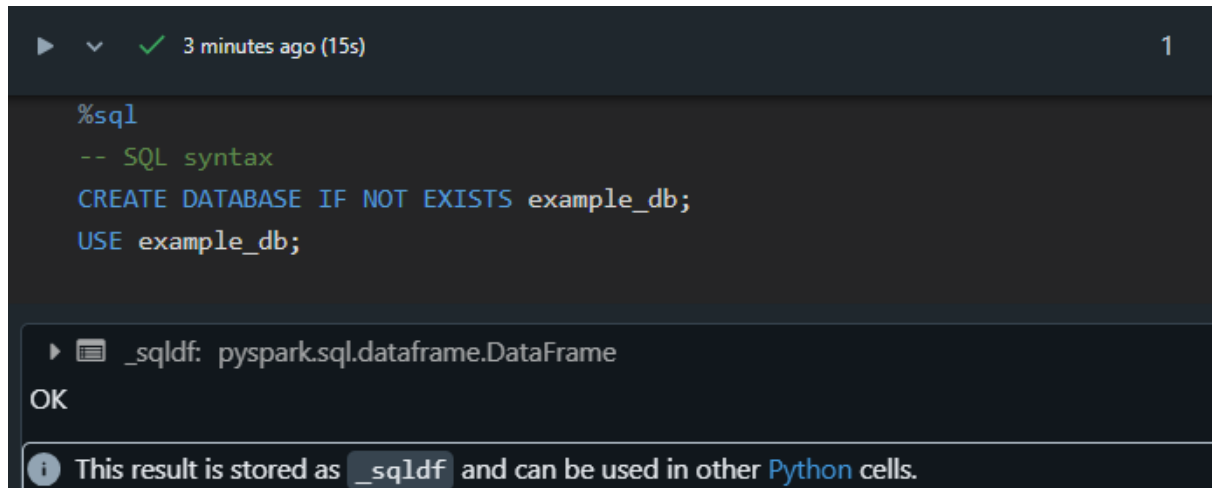


Daily Assignment-4

Name: **Harish Er**

1. Creating Databases, Tables, and Views

- Create a Database

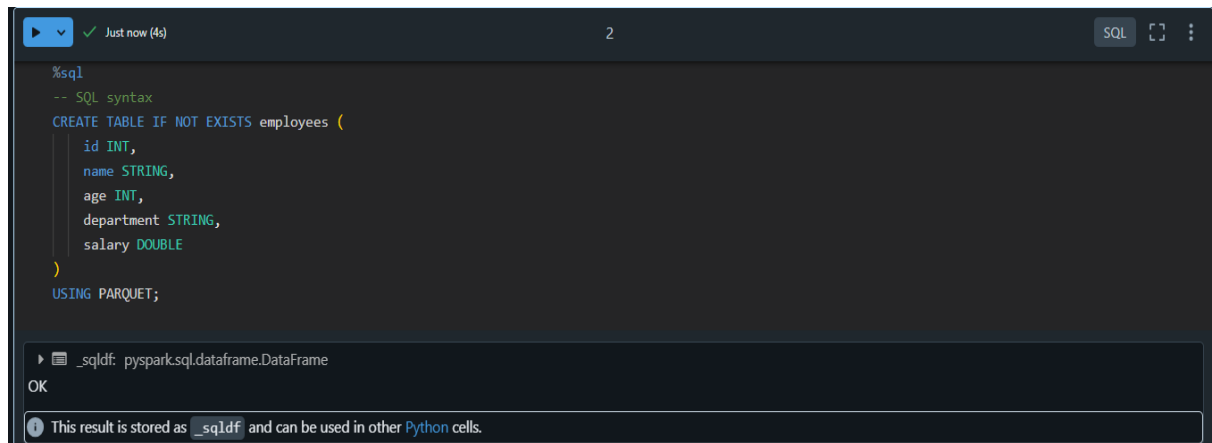


The screenshot shows a Databricks SQL cell with the following SQL code:

```
%sql
-- SQL syntax
CREATE DATABASE IF NOT EXISTS example_db;
USE example_db;
```

The execution status is "3 minutes ago (15s)" with a green checkmark. The output shows a table icon, the variable name `_sqldf`, the type `pyspark.sql.dataframe.DataFrame`, and the text "OK". A message at the bottom states: "This result is stored as `_sqldf` and can be used in other Python cells."

- Create a Table

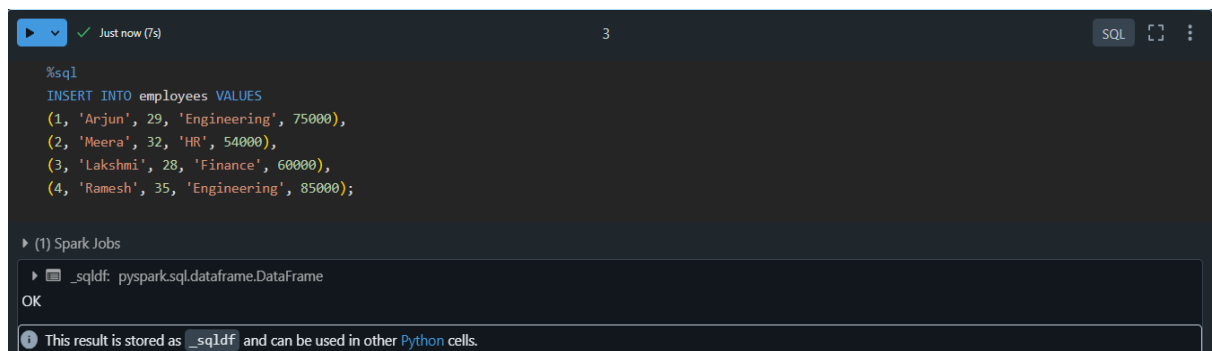


The screenshot shows a Databricks SQL cell with the following SQL code:

```
%sql
-- SQL syntax
CREATE TABLE IF NOT EXISTS employees (
  id INT,
  name STRING,
  age INT,
  department STRING,
  salary DOUBLE
)
USING PARQUET;
```

The execution status is "Just now (4s)" with a green checkmark. The output shows a table icon, the variable name `_sqldf`, the type `pyspark.sql.dataframe.DataFrame`, and the text "OK". A message at the bottom states: "This result is stored as `_sqldf` and can be used in other Python cells."

- Insert Data into the Table



The screenshot shows a Databricks SQL cell with the following SQL code:

```
%sql
INSERT INTO employees VALUES
(1, 'Arjun', 29, 'Engineering', 75000),
(2, 'Meera', 32, 'HR', 54000),
(3, 'Lakshmi', 28, 'Finance', 60000),
(4, 'Ramesh', 35, 'Engineering', 85000);
```

The execution status is "Just now (7s)" with a green checkmark. The output shows "(1) Spark Jobs" and a table icon, the variable name `_sqldf`, the type `pyspark.sql.dataframe.DataFrame`, and the text "OK". A message at the bottom states: "This result is stored as `_sqldf` and can be used in other Python cells."

- Create Temporary Views



The screenshot shows a Databricks notebook interface. At the top, there's a toolbar with a play button, a dropdown menu, a status indicator 'Just now (2s)', a cell number '4', and a language selector 'Python'. The main code area contains the following Python code:

```
# Python code
data = [(1, 'Arjun', 29, 'Engineering', 75000),
        (2, 'Meera', 32, 'HR', 54000),
        (3, 'Lakshmi', 28, 'Finance', 60000),
        (4, 'Ramesh', 35, 'Engineering', 85000)]

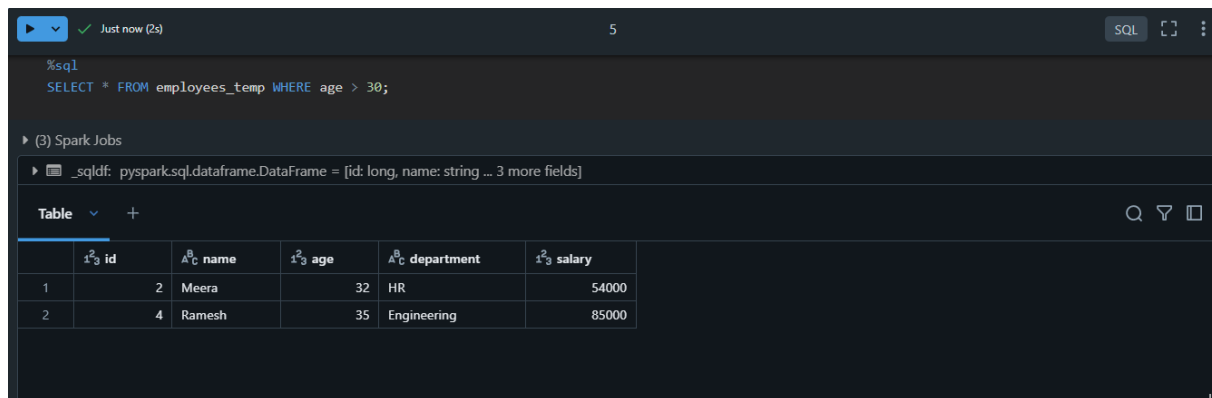
columns = ['id', 'name', 'age', 'department', 'salary']
df = spark.createDataFrame(data, columns)
df.createOrReplaceTempView("employees_temp")
```

Below the code, the variable 'df' is expanded to show its type and schema:

```
df: pyspark.sql.dataframe.DataFrame
  id: long
  name: string
  age: long
  department: string
  salary: long
```

2. Transformations

- Filter



The screenshot shows a Databricks notebook interface. At the top, there's a toolbar with a play button, a dropdown menu, a status indicator 'Just now (2s)', a cell number '5', and a language selector 'SQL'. The main code area contains the following SQL query:

```
%sql
SELECT * FROM employees_temp WHERE age > 30;
```

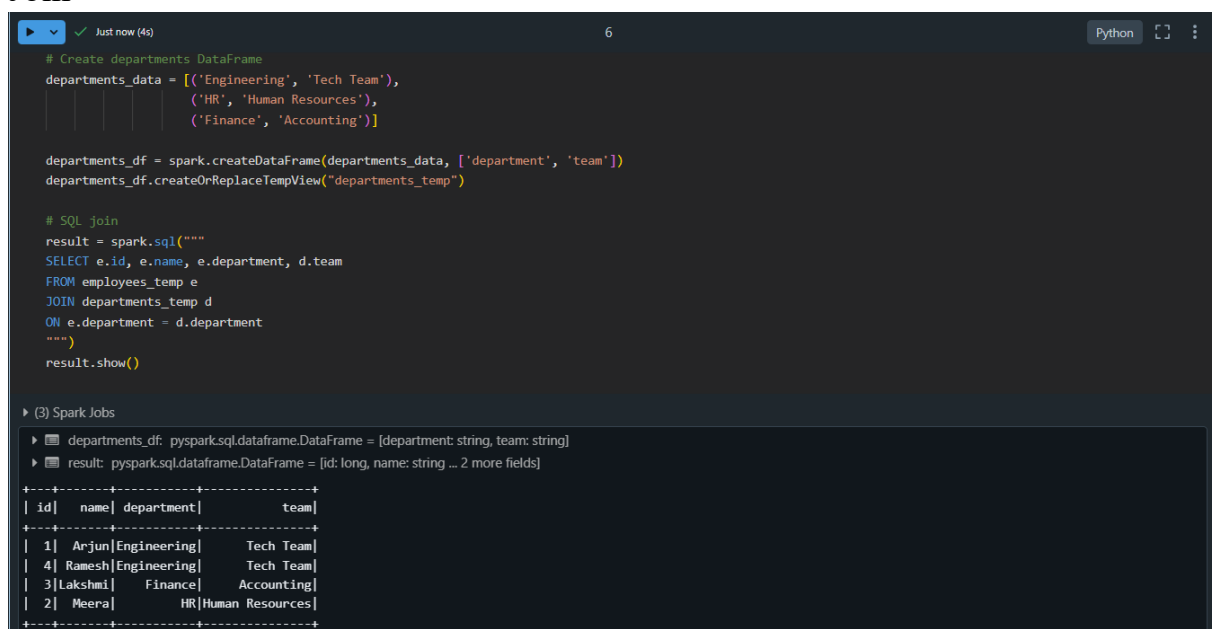
Below the code, the variable '_sqldf' is expanded to show its type and schema:

```
_sqldf: pyspark.sql.dataframe.DataFrame = [id: long, name: string ... 3 more fields]
```

Below the variable expansion, there's a table view of the results:

	¹ _{id}	² _{name}	¹ _{age}	¹ _{department}	¹ _{salary}
1	2	Meera	32	HR	54000
2	4	Ramesh	35	Engineering	85000

- Join



The screenshot shows a Databricks notebook interface. At the top, there's a toolbar with a play button, a dropdown menu, a status indicator 'Just now (4s)', a cell number '6', and a language selector 'Python'. The main code area contains the following Python code:

```
# Create departments DataFrame
departments_data = [('Engineering', 'Tech Team'),
                    ('HR', 'Human Resources'),
                    ('Finance', 'Accounting')]

departments_df = spark.createDataFrame(departments_data, ['department', 'team'])
departments_df.createOrReplaceTempView("departments_temp")

# SQL join
result = spark.sql("""
SELECT e.id, e.name, e.department, d.team
FROM employees_temp e
JOIN departments_temp d
ON e.department = d.department
""")
result.show()
```

Below the code, the variables 'departments_df' and 'result' are expanded to show their types and schemas:

```
departments_df: pyspark.sql.dataframe.DataFrame = [department: string, team: string]
result: pyspark.sql.dataframe.DataFrame = [id: long, name: string ... 2 more fields]
```

Below the variable expansion, there's a table view of the results:

	¹ _{id}	¹ _{name}	¹ _{department}	¹ _{team}
1	1	Arjun	Engineering	Tech Team
2	4	Ramesh	Engineering	Tech Team
3	3	Lakshmi	Finance	Accounting
4	2	Meera	HR	Human Resources

- GroupBy and Aggregations

Just now (2s) 7

```
%sql
SELECT department, AVG(salary) AS avg_salary, COUNT(*) AS total_employees
FROM employees_temp
GROUP BY department;
```

▶ (2) Spark Jobs

▶ _sqldf: pyspark.sql.dataframe.DataFrame = [department: string, avg_salary: double ... 1 more field]

Table ▾ +

	¹ department	^{1.2} avg_salary	^{1.2} total_employees
1	Engineering	80000	2
2	HR	54000	1
3	Finance	60000	1

- Window Functions

Just now (1s) 8

```
%sql
SELECT
  id,
  name,
  salary,
  RANK() OVER (PARTITION BY department ORDER BY salary DESC) AS rank
FROM employees_temp;
```

▶ (2) Spark Jobs

▶ _sqldf: pyspark.sql.dataframe.DataFrame = [id: long, name: string ... 2 more fields]

Table ▾ +

	^{1.2} id	^{1.2} name	^{1.2} salary	^{1.2} rank
1	4	Ramesh	85000	1
2	1	Arjun	75000	2
3	3	Lakshmi	60000	1
4	2	Meera	54000	1

3. Creating Local and Temporary Views

- Create a Temporary View

Just now (<1s) 9

```
df.createOrReplaceTempView("temp_view_name")
```

Python

- Create a Global Temporary View

Just now (<1s) 10

```
df.createOrReplaceGlobalTempView("global_temp_view_name")
```

Python

- Query the Temporary View

Just now (1s) 11

```
%sql
SELECT * FROM temp_view_name WHERE age > 30;
```

▶ (3) Spark Jobs

▶ _sqldf: pyspark.sql.dataframe.DataFrame = [id: long, name: string ... 3 more fields]

Table ▾ +

	^{1.3} id	^{1.3} name	^{1.3} age	^{1.3} department	^{1.3} salary
1	2	Meera	32	HR	54000
2	4	Ramesh	35	Engineering	85000

