EXPERIMENT 4

AIM : Implementing Relational Algorithm on Pig.

Step 1: Start Apache Pig

Open a terminal in Hadoop and start Pig in **interactive mode**

Step 2: Load Datasets

Let's assume we have two datasets:

1. Employees Data (employees.txt)
2. Locations Data (locations.txt)

Load Data into Pig

```
pig
employees = LOAD 'employees.txt' USING PigStorage(',')
        AS (ID:int, Name:chararray, Department:chararray);

locations = LOAD 'locations.txt' USING PigStorage(',')
        AS (ID:int, City:chararray);
```

```
Table 1 (Employees):
+---+--------+----------+
| ID|    Name|Department|
+---+--------+----------+
|  1|   Alice|        HR|
|  2|     Bob|        IT|
|  3|Charlie|   Finance|
+---+--------+----------+

Table 2 (Locations):
+---+--------------+
| ID|          City|
+---+--------------+
|  1|    New York|
|  2|San Francisco|
|  3|Los Angeles|
+---+--------------+
```

Step 3: Perform Relational Algebra Operations

1. JOIN Operation (Combining both tables based on ID)

```
joined_data = JOIN employees BY ID, locations BY ID;
DUMP joined_data;
```

2. FILTER Operation (Select Employees from IT Department)

```
it_employees = FILTER employees BY Department == 'IT';
DUMP it_employees;
```

3. GROUP BY Operation (Group Employees by Department)

```
grouped_data = GROUP employees BY Department;
DUMP grouped_data;
```

4. FOREACH (PROJECT) Operation (Project only Employee Names)

```
projected_data = FOREACH employees GENERATE Name;
DUMP projected_data;
```

Step 4: Store Output

If you want to **store the output** instead of using DUMP:

```
pig
STORE joined_data INTO 'output/joined_data' USING PigStorage(',');
STORE it_employees INTO 'output/it_employees' USING PigStorage(',');
STORE grouped_data INTO 'output/grouped_data' USING PigStorage(',');
STORE projected_data INTO 'output/projected_data' USING PigStorage(',');
```

```
Joined Table (Employees + Locations):
+---+-------+----------+-------------+
| ID|   Name|Department|         City|
+---+-------+----------+-------------+
|  1|  Alice|        HR|   New York|
|  2|    Bob|        IT|San Francisco|
|  3|Charlie|   Finance|Los Angeles|
+---+-------+----------+-------------+

Filtered Employees (Only IT Department):
+---+----+----------+
| ID|Name|Department|
+---+----+----------+
|  2| Bob|        IT|
+---+----+----------+
```

Step 5: Exit Pig

Once done, exit the Pig interactive shell:

bash

quit;

EXPERIMENT 5

AIM : Implementing database operations on Hive.

Step 1: Start Hive

Before executing any queries, ensure that Hadoop and Hive are properly set up. Open the terminal and start Hive:

Step 2: Create a Database

To organize data, create a new database in Hive.

```
CREATE DATABASE company_db;
USE company_db;
```

Step 3: Create Tables in Hive

We will create two tables:

1. **employees** (Stores employee details)
2. **departments** (Stores department details)

```
CREATE TABLE employees (
    emp_id INT,
    name STRING,
    age INT,
    dept_id INT
)
ROW FORMAT DELIMITED
FIELDS TERMINATED BY ','
STORED AS TEXTFILE;
CREATE TABLE departments (
    dept_id INT,
    dept_name STRING
)
ROW FORMAT DELIMITED
FIELDS TERMINATED BY ','
STORED AS TEXTFILE;
```

Step 4: Load Data into Tables

The employee and department details should be stored in files and uploaded to HDFS.

1. Upload Sample Data to HDFS

Save the following data in local files:

employees.txt

departments.txt

Now, move these files to HDFS:

bash

```
hdfs dfs -mkdir /user/hive/warehouse/user_raj
hdfs dfs -put employees.txt /user/hive/warehouse/user_raj/
hdfs dfs -put departments.txt /user/hive/warehouse/user_raj/
```

2. Load Data into Hive Tables

```
LOAD DATA INPATH '/user/hive/warehouse/user_raj/employees.txt' INTO
TABLE employees;
LOAD DATA INPATH '/user/hive/warehouse/user_raj/departments.txt'
INTO TABLE departments;
```

Step 5: Perform Database Operations

Now, we will perform SQL-like queries in Hive.

1. Select Data from Tables

```
SELECT * FROM employees;
```

2. Perform a JOIN Operation

```
SELECT e.emp_id, e.name, e.age, d.dept_name
FROM employees e
JOIN departments d
ON e.dept_id = d.dept_id;
```

Output:

emp_id	name	age	dept_name
1	Alice	25	HR
2	Bob	30	IT
3	Charlie	28	HR
4	David	35	Finance
5	Eva	27	IT

3. Filter Data (Retrieve Employees Older Than 28)

```
SELECT * FROM employees WHERE age > 28;
```

emp_id	name	age	dept_id
2	Bob	30	102
4	David	35	103

4. Grouping and Aggregation

Find the number of employees in each department:

sql

CopyEdit

```
SELECT d.dept_name, COUNT(e.emp_id) AS num_employees
FROM employees e
JOIN departments d ON e.dept_id = d.dept_id
GROUP BY d.dept_name;
```

Step 6: Delete Data and Drop Tables

To delete data from a table:

```
DELETE FROM employees WHERE age < 28;
```

Note: DELETE only works with **transactional tables**. If using non-ACID tables,

```
INSERT OVERWRITE TABLE employees SELECT * FROM employees
WHERE age >= 28;
```

To drop a table:

```
DROP TABLE employees;
DROP TABLE departments;
```

Step 7: Exit Hive

After performing all operations, exit Hive: