EXERCISE 4 – Loading and Comples data Transformations using Pig

Step 1: Load the employee dataset

employee\_data = LOAD 'employee\_data.csv' USING PigStorage(',')

AS (emp\_id:int, emp\_name:chararray, department\_id:int, salary:float, age:int);

-- Step 2: Load the department dataset

department\_data = LOAD 'department\_data.csv' USING PigStorage(',')

AS (department\_id:int, department\_name:chararray);

-- Step 3: Filter employees with salary greater than 70,000 and age greater than 30

filtered\_employees = FILTER employee\_data BY salary > 70000 AND age > 30;

-- Step 4: Join the employee data with department data based on department\_id

joined\_data = JOIN filtered\_employees BY department\_id, department\_data BY department\_id;

-- Step 5: Group the joined data by department

grouped\_by\_department = GROUP joined\_data BY department\_data::department\_name;

-- Step 6: Calculate total salary, average salary, and number of employees per department

department\_aggregates = FOREACH grouped\_by\_department GENERATE

group AS department\_name,

COUNT(joined\_data) AS employee\_count,

SUM(joined\_data.salary) AS total\_salary, AVG(joined\_data.salary) AS avg\_salary;

-- Step 7: Filter out departments with fewer than 2 employees

filtered\_departments = FILTER department\_aggregates BY employee\_count >= 2;

-- Step 8: Store the result in a new file filtered\_department\_summary

STORE filtered\_departments INTO 'filtered\_department\_summary' USING PigStorage(',');

Exercise 4.1 Customer and Transactions

-- Load the datasets

customers = LOAD 'customers.csv' USING PigStorage(',')

AS (customer\_id:int, name:chararray, location:chararray);

transactions = LOAD 'transactions.csv' USING PigStorage(',')

AS (transaction\_id:int, customer\_id:int, item:chararray, amount:float);

-- Filter transactions where amount >= 1000

high\_value\_transactions = FILTER transactions BY amount >= 1000;

-- Group filtered transactions by customer\_id

grouped\_transactions = GROUP high\_value\_transactions BY customer\_id;

-- Join customers with filtered transactions on customer\_id

customer\_transactions = JOIN customers BY customer\_id,

high\_value\_transactions BY customer\_id;

-- Calculate total spending per customer

customer\_spending = FOREACH (GROUP customer\_transactions BY customers::customer\_id)

GENERATE group AS customer\_id,

SUM(customer\_transactions.high\_value\_transactions::amount) AS total\_spent;

-- Sort customers by total spending in descending order

sorted\_customers = ORDER customer\_spending BY total\_spent DESC;

-- Retrieve the top 3 spenders

top\_spenders = LIMIT sorted\_customers 3;

-- Display the top 3 spenders

DUMP top\_spenders;

-- Group customers by location and calculate total spending per location

location\_spending = FOREACH (GROUP customer\_transactions BY customers::location)

GENERATE group AS location,

SUM(customer\_transactions.high\_value\_transactions::amount) AS total\_spent\_by\_location;

-- Display total spending per location on high-value items

DUMP location\_spending;