**Exercise 5: Advanced Data Transformations using Pig**

**Aim:**

To learn how to perform advanced data transformations using Pig, including joins and grouping operations.

**Procedure:**

* Start the Pig CLI.
* Load the dataset into Pig with the appropriate schema.
* Filter the dataset to include only relevant records.
* Group the data by a specified attribute.
* Perform joins between two datasets based on a common key.
* Execute aggregate functions to analyze the grouped data.
* Store the results in a new dataset.

**Program:**

**-- Load the product data**

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

AS (product\_id:chararray, product\_name:chararray, price:float, category:chararray);

**-- Load the sales data**

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

AS (order\_id:int, product\_id:chararray, quantity:int, order\_date:chararray);

**-- Filter products in the 'Electronics' category**

filtered\_products = FILTER product\_data BY category == 'Electronics';

**-- Group sales data by product\_id**

grouped\_sales = GROUP sales\_data BY product\_id;

**-- Join filtered products with grouped sales**

joined\_data = JOIN filtered\_products BY product\_id, grouped\_sales BY product\_id;

**-- Calculate total sales per product**

total\_sales = FOREACH joined\_data GENERATE

filtered\_products::product\_name,

SUM(sales\_data.quantity \* filtered\_products.price) AS total\_sales;

**-- Store the result in a new dataset**

STORE total\_sales INTO 'total\_sales\_data' USING PigStorage(',');

**Sample Dataset:**

**product\_data.csv**

|  |  |  |  |
| --- | --- | --- | --- |
| product\_id | product\_name | price | Category |
| P001 | Laptop | 500.0 | Electronics |
| P002 | Smartphone | 300.0 | Electronics |
| P003 | Refrigerator | 700.0 | Appliances |
| P004 | Headphones | 100.0 | Electronics |
| P005 | Microwave | 150.0 | Appliances |

**sales\_data.csv**

|  |  |  |  |
| --- | --- | --- | --- |
| order\_id | product\_id | quantity | order\_date |
| 1 | P001 | 2 | 2024-01-15 |
| 2 | P002 | 1 | 2024-02-05 |
| 3 | P003 | 1 | 2024-01-25 |
| 4 | P004 | 5 | 2024-03-12 |
| 5 | P005 | 3 | 2024-01-30 |

**Output:**

* **Filtered Products in Electronics Category:**

|  |  |  |  |
| --- | --- | --- | --- |
| product\_id | product\_name | price | Category |
| P001 | Laptop | 500.0 | Electronics |
| P002 | Smartphone | 300.0 | Electronics |
| P004 | Headphones | 100.0 | Electronics |

* **Total Sales per Product:**

|  |  |
| --- | --- |
| product\_name | total\_sales |
| Laptop | 1000.0 |
| Smartphone | 300.0 |
| Headphones | 500.0 |

**Result:**

Successfully executed advanced data transformations in Pig, including filtering, grouping, joining datasets, and calculating total sales for products in the Electronics category. The results were stored in a new dataset for further analysis.

**Exercise 5.1: Advanced Data Transformations using Pig**

This exercise focuses on advanced data transformations using Pig, where you will perform tasks such as loading datasets, calculating revenue, filtering data, grouping by categories, and identifying top products based on revenue.

**Dataset:**

* **products.csv**: { product\_id, product\_name, category, price }
* **sales.csv**: { sale\_id, product\_id, customer\_id, quantity, sale\_date }

**Tasks & Pig Script:**

1. **Load the Datasets:**
   * Load the products.csv and sales.csv files into Pig using the appropriate schema.

pig

Copy code

-- Load products dataset

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

AS (product\_id:chararray, product\_name:chararray, category:chararray, price:float);

-- Load sales dataset

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

AS (sale\_id:int, product\_id:chararray, customer\_id:chararray, quantity:int, sale\_date:chararray);

1. **Calculate Total Revenue Per Sale:**
   * Calculate the total revenue for each sale by multiplying the quantity sold by the product price.

pig

Copy code

-- Join products and sales datasets to calculate total revenue per sale

joined\_data = JOIN sales BY product\_id, products BY product\_id;

-- Calculate total revenue per sale

revenue\_per\_sale = FOREACH joined\_data GENERATE

sales::sale\_id,

sales::product\_id,

(sales::quantity \* products::price) AS total\_revenue;

1. **Filter Sales for High-Value Products:**
   * Filter out sales where the product price is greater than or equal to $500.

pig

Copy code

-- Filter sales where product price is >= 500

high\_value\_sales = FILTER joined\_data BY products::price >= 500;

1. **Group Sales by Category:**
   * Group the sales by product category and calculate the total quantity sold and total revenue for each category.

pig

Copy code

-- Group sales by product category

grouped\_by\_category = GROUP joined\_data BY products::category;

-- Calculate total quantity sold and total revenue per category

category\_totals = FOREACH grouped\_by\_category GENERATE

group AS category,

SUM(sales::quantity) AS total\_quantity,

SUM(sales::quantity \* products::price) AS total\_revenue;

1. **Join Products and Sales Data:**
   * Perform a join between the products and sales datasets based on product\_id to enrich the sales data with product details.

pig

Copy code

-- Already joined in step 2

-- "joined\_data" contains the result of joining sales and products datasets

1. **Calculate Top 3 Products by Revenue:**
   * For each product, calculate the total revenue and sort products based on the revenue. Extract the top 3 products.

pig

Copy code

-- Group sales by product to calculate total revenue per product

grouped\_by\_product = GROUP joined\_data BY products::product\_id;

-- Calculate total revenue per product

total\_revenue\_per\_product = FOREACH grouped\_by\_product GENERATE

group AS product\_id,

MAX(products::product\_name) AS product\_name,

SUM(sales::quantity \* products::price) AS total\_revenue;

-- Sort products by total revenue in descending order and take top 3

sorted\_products = ORDER total\_revenue\_per\_product BY total\_revenue DESC;

top\_3\_products = LIMIT sorted\_products 3;

1. **Identify Product Categories with Sales Above a Threshold:**
   * Filter out categories with total sales revenue greater than $1000.

pig

Copy code

-- Filter categories where total revenue is greater than $1000

high\_revenue\_categories = FILTER category\_totals BY total\_revenue > 1000;

**Sample Output:**

1. **Total Revenue Per Sale:**

yaml

Copy code

sale\_id product\_id total\_revenue

1 P001 1000.0

2 P002 600.0

3 P003 700.0

4 P004 500.0

1. **Filtered High-Value Sales (Products with price >= $500):**

Copy code

sale\_id product\_id product\_name price

1 P001 Laptop 500.0

3 P003 Refrigerator 700.0

1. **Grouped Sales by Category (with total quantity and revenue):**

yaml

Copy code

category total\_quantity total\_revenue

Electronics 10 2100.0

Appliances 4 850.0

1. **Top 3 Products by Revenue:**

yaml

Copy code

product\_name total\_revenue

Laptop 1000.0

Refrigerator 700.0

Smartphone 600.0

1. **Categories with Revenue Above $1000:**

yaml

Copy code

category total\_revenue

Electronics 2100.0

**Conclusion:**

In this exercise, you successfully performed advanced data transformations using Pig, including loading datasets, calculating total revenue, filtering, grouping by categories, joining data, and identifying top-performing products and categories based on revenue thresholds. This showcases the flexibility and power of Pig for processing large datasets in a streamlined and efficient manner.

**Exercise 6: Creating and Querying Tables in Hive with SQL**

**Aim:**

To learn how to create tables in Hive, load data into tables, and perform basic SQL queries such as SELECT, WHERE, and GROUP BY.

**Procedure:**

* Start the Hive CLI.
* Create a new database to organize your tables and switch to it.
* Create a new table to store sales data with appropriate columns and data types.
* Load the data into the sales\_data table from an HDFS location.
* Run SQL queries to analyze the data, including displaying all rows, calculating total sales per product, and filtering data by category.

**Program:**

CREATE DATABASE IF NOT EXISTS sales\_db;

USE sales\_db;

CREATE TABLE sales\_data (

order\_id INT,

product\_name STRING,

category STRING,

price FLOAT,

quantity INT,

order\_date STRING

) ROW FORMAT DELIMITED

FIELDS TERMINATED BY ','

STORED AS TEXTFILE;

LOAD DATA INPATH 'sales\_data.csv' INTO TABLE sales\_data;

**-- Display all data**

SELECT \* FROM sales\_data;

**-- Calculate total sales per product**

SELECT product\_name, SUM(price \* quantity) AS total\_salesFROM sales\_dataGROUP BY product\_name;

**-- Filter data for 'Electronics' category**

SELECT \* FROM sales\_dataWHERE category = 'Electronics';

**Sample Dataset:**

* **sales\_data.csv**{order\_id,product\_name,category,price,quantity,order\_date}

1,Laptop,Electronics,500.0,2,2024-01-15

2,Smartphone,Electronics,300.0,1,2024-02-05

3,Refrigerator,Appliances,700.0,1,2024-01-25

4,Headphones,Electronics,100.0,5,2024-03-12

5,Microwave,Appliances,150.0,3,2024-01-30

**Output:**

* **Displaying all rows from the table:**

| **order\_id** | **product\_name** | **category** | **price** | **quantity** | **order\_date** |
| --- | --- | --- | --- | --- | --- |
| 1 | Laptop | Electronics | 500.0 | 2 | 2024-01-15 |
| 2 | Smartphone | Electronics | 300.0 | 1 | 2024-02-05 |
| 3 | Refrigerator | Appliances | 700.0 | 1 | 2024-01-25 |
| 4 | Headphones | Electronics | 100.0 | 5 | 2024-03-12 |
| 5 | Microwave | Appliances | 150.0 | 3 | 2024-01-30 |

* **Total sales per product:**

| **product\_name** | **total\_sales** |
| --- | --- |
| Laptop | 1000.0 |
| Smartphone | 300.0 |
| Refrigerator | 700.0 |
| Headphones | 500.0 |
| Microwave | 450.0 |

* **Filtered data for the 'Electronics' category:**

| **order\_id** | **product\_name** | **category** | **price** | **quantity** | **order\_date** |
| --- | --- | --- | --- | --- | --- |
| 1 | Laptop | Electronics | 500.0 | 2 | 2024-01-15 |
| 2 | Smartphone | Electronics | 300.0 | 1 | 2024-02-05 |
| 4 | Headphones | Electronics | 100.0 | 5 | 2024-03-12 |

**Result:**

Successfully created a table in Hive, loaded sales data, and performed various SQL queries to analyze the data.

**Exercise 6.1: Creating and Querying Tables in Hive with SQL**

**DataSet**

* **products.csv**:{product\_id, product\_name, category, price} : 5Records
* **sales.csv**:{sale\_id, product\_id, customer\_id, quantity, sale\_date} :10Records

**1. Start Hive CLI**

To start the Hive CLI, you can open the terminal and type:

bash

Copy code

hive

**2. Create a Database**

Once the Hive CLI is running, create a new database:

sql

Copy code

CREATE DATABASE retail\_db;

**3. Use the Created Database**

Switch to the newly created database:

sql

Copy code

USE retail\_db;

**4. Create a Table for Products**

Now, create a table for products:

sql

Copy code

CREATE TABLE products (

product\_id INT,

product\_name STRING,

category STRING,

price FLOAT

)

ROW FORMAT DELIMITED

FIELDS TERMINATED BY ','

STORED AS TEXTFILE;

**5. Load Data into Products Table**

Assume the products.csv file is in HDFS at the path /user/hive/warehouse/retail\_db/products.csv. You can load data into the table like this:

sql

Copy code

LOAD DATA INPATH '/user/hive/warehouse/retail\_db/products.csv'

INTO TABLE products;

**6. Create a Table for Sales**

Now, create a table for sales:

sql

Copy code

CREATE TABLE sales (

sale\_id INT,

product\_id INT,

customer\_id INT,

quantity INT,

sale\_date STRING

)

ROW FORMAT DELIMITED

FIELDS TERMINATED BY ','

STORED AS TEXTFILE;

**7. Load Data into Sales Table**

Assume the sales.csv file is in HDFS at the path /user/hive/warehouse/retail\_db/sales.csv. Load data into the sales table:

sql

Copy code

LOAD DATA INPATH '/user/hive/warehouse/retail\_db/sales.csv'

INTO TABLE sales;

**8. Query to Select All Products**

To select all products:

sql

Copy code

SELECT \* FROM products;

**9. Query to Get Products with Price Greater than $500**

To get products with a price greater than $500:

sql

Copy code

SELECT \* FROM products

WHERE price > 500;

**10. Query to Count Total Sales by Product**

To count the total sales by each product:

sql

Copy code

SELECT p.product\_name, COUNT(s.sale\_id) AS total\_sales

FROM products p

JOIN sales s

ON p.product\_id = s.product\_id

GROUP BY p.product\_name;

**11. Query to Get Total Revenue by Product**

To get the total revenue by product (quantity sold \* product price):

sql

Copy code

SELECT p.product\_name, SUM(s.quantity \* p.price) AS total\_revenue

FROM products p

JOIN sales s

ON p.product\_id = s.product\_id

GROUP BY p.product\_name;

**Expected Output:**

1. **Select All Products**:

diff

Copy code

+------------+----------------+-----------+-------+

| product\_id | product\_name | category | price |

+------------+----------------+-----------+-------+

| 1 | Laptop | Electronics| 800 |

| 2 | Smartphone | Electronics| 600 |

| 3 | Headphones | Accessories| 150 |

| 4 | TV | Electronics| 1000 |

| 5 | Microwave | Home | 250 |

+------------+----------------+-----------+-------+

1. **Products with Price Greater than $500**:

diff

Copy code

+------------+----------------+-----------+-------+

| product\_id | product\_name | category | price |

+------------+----------------+-----------+-------+

| 1 | Laptop | Electronics| 800 |

| 2 | Smartphone | Electronics| 600 |

| 4 | TV | Electronics| 1000 |

+------------+----------------+-----------+-------+

1. **Count Total Sales by Product**:

diff

Copy code

+----------------+-------------+

| product\_name | total\_sales |

+----------------+-------------+

| Laptop | 3 |

| Smartphone | 2 |

| Headphones | 1 |

| TV | 2 |

| Microwave | 2 |

+----------------+-------------+

1. **Total Revenue by Product**:

diff

Copy code

+----------------+--------------+

| product\_name | total\_revenue|

+----------------+--------------+

| Laptop | 2400 |

| Smartphone | 1200 |

| Headphones | 150 |

| TV | 2000 |

| Microwave | 500 |

+----------------+--------------+