Some answers of Questions on Unit-4

1. Create a data file for below schemas:
Order: CustomerId, ItemId, ItemName, OrderDate, DeliveryDate
Customer: Customerld, CustomerName, Address, City, State, Country
i. Create a table for Order and Customer Data.
ii. Write a HiveQL to find number of items bought by each customer.

Answer:

```
-- Create Hive tables
-- Create the Order table
CREATE TABLE orders (
   CustomerId INT,
   ItemId INT,
   ItemName STRING,
   OrderDate DATE,
   DeliveryDate DATE
) ROW FORMAT DELIMITED
FIELDS TERMINATED BY ','
STORED AS TEXTFILE;
-- Load data into the Order table
LOAD DATA LOCAL INPATH 'order_data.txt' INTO TABLE orders;

-- Create the Customer table
CREATE TABLE customers (
   CustomerId INT,
   CustomerName STRING,
   Address STRING,
   City STRING,
   State STRING,
   Country STRING
) ROW FORMAT DELIMITED
FIELDS TERMINATED BY ','
STORED AS TEXTFILE;
-- Load data into the Customer table
LOAD DATA LOCAL INPATH 'customer_data.txt' INTO TABLE customers;

-- HiveQL query to find the number of items bought by each customer
SELECT
   c.CustomerName,  COUNT(o.ItemId) AS NumberOfItems
FROM
   customers c
JOIN
   orders o
ON
   c.CustomerId = o.CustomerId
GROUP BY
   c.CustomerName;
```

2. Write a Pig Latin script to perform the following tasks on a dataset sales_data (fields: product_id, category, amount, date):
   a. Filter the data for sales in the "Electronics" category.
   b. Calculate the total sales amount for each product_id in this category.
   c. Sort the results by total amount in descending order.

Answer:

```
-- Load the sales_data dataset
sales_data = LOAD 'sales_data' USING PigStorage(',') AS (product_id:chararray, category:chararray, amount:float, date:chararray);

-- Filter the data for sales in the 'Electronics' category
electronics_sales = FILTER sales_data BY category == 'Electronics';

-- Calculate the total sales amount for each product_id in the Electronics category
total_sales = FOREACH (GROUP electronics_sales BY product_id) GENERATE
        group AS product_id,
        SUM(electronics_sales.amount) AS total_amount;

-- Sort the results by total amount in descending order
sorted_sales = ORDER total_sales BY total_amount DESC;

-- Display the results
DUMP sorted_sales;
```

3. Explain User Defined Functions (UDFs) in Hive. Describe their purpose. Write a Hive function to convert the values of a field to uppercase.

Answer:

User Defined Functions (UDFs) in Hive allow you to extend the functionality of Hive queries by writing custom functions. Hive comes with many built-in functions for common operations (like string manipulation, mathematical functions, etc.), but in case these built-in functions are not sufficient, UDFs provide a way to implement custom logic. You can write UDFs in Java, and once created, they can be used in Hive queries just like built-in functions.

Purpose of UDFs in Hive

The main purpose of UDFs in Hive is to:

1. Extend Hive's functionality: Add custom business logic or complex transformations that are not supported by the built-in functions.
2. Custom data processing: Perform operations on data that may not be handled efficiently by HiveQL alone, such as custom string manipulations or mathematical operations.
3. Reusability: Write a custom function once, and then reuse it in multiple Hive queries.
4. Integration with external systems: UDFs allow integration with external systems or libraries, enabling more complex data processing.

Writing a UDF in Hive to Convert Field Values to Uppercase

To write a UDF in Hive to convert the values of a field to uppercase, you would follow these general steps:

1. Write the UDF in Java: The UDF will extend the UDF class from Hive's API.
2. Compile the UDF into a JAR file and
3. add it to Hive.

1. Create the UDF in Java

```
import org.apache.hadoop.hive.ql.exec.UDF;
import org.apache.hadoop.io.Text;

public class ToUpperCaseUDF extends UDF {
   public Text evaluate(Text input) {
      if (input == null) {
         return null;
      }
      return new Text(input.toString().toUpperCase());
   }
}
```

Explanation:

- **evaluate Method**: This is the main method that Hive calls to process data.
- **Input**: It accepts and processes the input (e.g., a string in this case).
- **Output**: Returns the transformed result.

2. Compile the Java Code

```
javac -cp "$(hive --auxpath)" ToUpperCaseUDF.java
jar -cf ToUpperCaseUDF.jar ToUpperCaseUDF.class
```

Explanation:

- Save the file as **ToUpperCaseUDF.java.**
- Compile it into **a .jar** file using the following commands

3. Add the UDF JAR to Hive- Upload the JAR file to a location accessible to Hive (e.g., HDFS or local path).

```
ADD JAR /path/to/ToUpperCaseUDF.jar;
```