Cours : Big Data

Classe : IPSL

Auteur : Dr. Djibril MBOUP

E-mail : mboupdjibril.m2itic@gmail.com

Date limite : Avant le 27/07/2024 à 23 : 59.

Fait par : Mariame Diallo Ing3Info

Projet : Ingestion de données dans Big Data

I. Ingestion de données avec Apache Scoop

❖ Aperçu :

Apache Sqoop est un outil en ligne de commande pour transférer des données en masse depuis des bases de données relationnelles vers Hive. Il prend en charge le chargement différentiel d'une table ou d'une requête SQL depuis la dernière importation.

❖ Travail à faire :

Nous allons télécharger les scripts SQL de la base de données retail_db.sql sur ce lien drive ci-dessous :

https://drive.google.com/file/d/1CHwWhfJn4edCuAOHiWr6iyT4wJ-zPNbU/view?usp=share_link

Retail DB est une base de données qui contient des données de ventes d'une entreprise de commerce. Cette base de données comporte 6 tables :

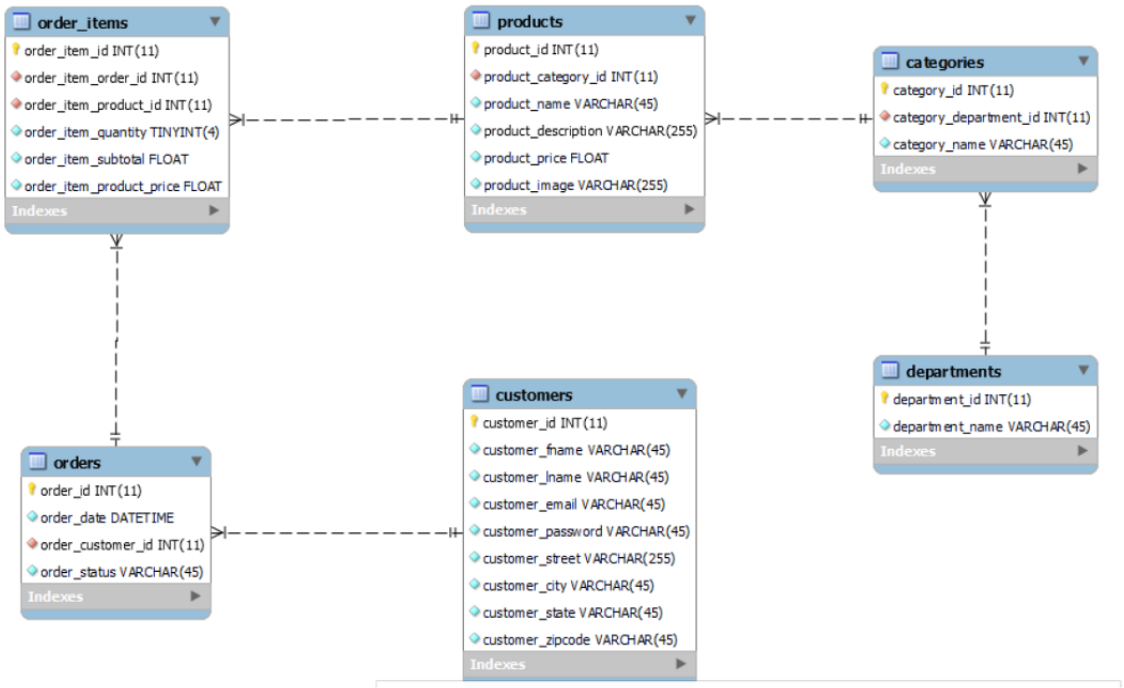

Figure 1: Schéma de la base de données Retail DB

- Démarrons MySQL en mode console

$ mysql -u root –p

- Création d'un compte utilisateur admin

mysql> CREATE user retail_user identified by 'hadoop';

- Création de la base de données retail_db

mysql> CREATE user retail_user identified by 'hadoop';

```
Microsoft Windows [version 10.0.22631.3737]
(c) Microsoft Corporation. Tous droits réservés.

C:\Users\bmd tech\hadoopVagrant>mysql -u root -p
Enter password:
Welcome to the MariaDB monitor.  Commands end with ; or \g.
Your MySQL connection id is 22
Server version: 8.3.0 MySQL Community Server - GPL

Copyright (c) 2000, 2018, Oracle, MariaDB Corporation Ab and others.

Type 'help;' or '\h' for help. Type '\c' to clear the current input statement.

MySQL [(none)]>  CREATE user retail_user identified by 'hadoop';
Query OK, 0 rows affected (0.024 sec)

MySQL [(none)]>  CREATE database retail_db;
Query OK, 1 row affected (0.015 sec)
```

- Ajoutons les droits d'utilisateur sur la base de données retail_db

mysql> GRANT ALL ON retail_db.* to retail_dba;

mysql> flush privileges;

```
MySQL [(none)]> GRANT ALL PRIVILEGES ON retail_db.* TO 'retail_user'@'%';
Query OK, 0 rows affected (0.004 sec)

MySQL [(none)]> flush privileges;
Query OK, 0 rows affected (0.002 sec)

MySQL [(none)]>
```

- Se connecter en tant que retail_user

$ mysql -u retail_user –p hadoop

mysql> USE retail_db;

```
C:\Users\bmd tech\hadoopVagrant>mysql -u retail_user -p
Enter password: ******
Welcome to the MariaDB monitor.  Commands end with ; or \g.
Your MySQL connection id is 46
Server version: 8.3.0 MySQL Community Server - GPL

Copyright (c) 2000, 2018, Oracle, MariaDB Corporation Ab and others.

Type 'help;' or '\h' for help. Type '\c' to clear the current input statement.

MySQL [(none)]> USE retail_db;
Database changed
MySQL [retail_db]>
```

- Chargement

mysql> source C:/Users/bmd tech/Desktop/TP3_BigData/retail_db.sql;

```
Database changed
MySQL [retail_db]> source C:/Users/bmd tech/Desktop/TP3_BigData/retail_db.sql;
Query OK, 0 rows affected (0.000 sec)

Query OK, 0 rows affected (0.000 sec)

Query OK, 0 rows affected (0.000 sec)

Query OK, 0 rows affected, 1 warning (0.000 sec)

Query OK, 0 rows affected (0.000 sec)

Query OK, 0 rows affected (0.000 sec)

Query OK, 0 rows affected (0.000 sec)

Query OK, 0 rows affected (0.000 sec)

Query OK, 0 rows affected (0.000 sec)

Query OK, 0 rows affected (0.000 sec)
```

- Affichage des tables

mysql> show tables;

- Démarrons le Vagrant VM

```
C:\Users\bmd tech\hadoopVagrant>vagrant up
Bringing machine 'default' up with 'virtualbox' provider...
==> default: Checking if box 'SopeKhadim/hadoopVM' version '2.0' is up to date...
==> default: Machine already provisioned. Run `vagrant provision` or use the `--provision`
==> default: flag to force provisioning. Provisioners marked to run always will still run.

C:\Users\bmd tech\hadoopVagrant>
```

- Se connecter

```
C:\Users\bmd tech\hadoopVagrant>vagrant ssh
Last login: Wed Jul 24 06:26:32 2024 from 10.0.2.2
[vagrant@10 ~]$
```

- Démarrons Hadoop

```
[vagrant@10 ~]$ start-dfs.sh
Starting namenodes on [localhost]
Starting datanodes
Starting secondary namenodes [10.0.2.15]
[vagrant@10 ~]$ start-all.sh
WARNING: Attempting to start all Apache Hadoop daemons as vagrant in 10 seconds.
WARNING: This is not a recommended production deployment configuration.
WARNING: Use CTRL-C to abort.
Starting namenodes on [localhost]
localhost: namenode is running as process 4937.  Stop it first.
Starting datanodes
localhost: datanode is running as process 5056.  Stop it first.
Starting secondary namenodes [10.0.2.15]
10.0.2.15: secondarynamenode is running as process 5257.  Stop it first.
Starting resourcemanager
resourcemanager is running as process 4302.  Stop it first.
Starting nodemanagers
localhost: nodemanager is running as process 4414.  Stop it first.
[vagrant@10 ~]$
```

- Vérifier si la machine virtuelle et votre machine locale sont dans le même réseau. Si c'est OK, vous pouvez tester avec la commande Sqoop ci-dessous :

sqoop list-databases \ --connect "jdbc:mysql://192.168.1.21:3306" \ --username retail_user \ --password hadoop

```
[vagrant@10 ~]$ sqoop list-tables --connect "jdbc:mysql://192.168.1.21:3306/retail_db?useSSL=false" --username retail_user --password
 hadoop
Warning: /usr/lib/sqoop/../hbase does not exist! HBase imports will fail.
Please set $HBASE_HOME to the root of your HBase installation.
Warning: /usr/lib/sqoop/../hcatalog does not exist! HCatalog jobs will fail.
Please set $HCAT_HOME to the root of your HCatalog installation.
Warning: /usr/lib/sqoop/../accumulo does not exist! Accumulo imports will fail.
Please set $ACCUMULO_HOME to the root of your Accumulo installation.
Warning: /usr/lib/sqoop/../zookeeper does not exist! Accumulo imports will fail.
Please set $ZOOKEEPER_HOME to the root of your Zookeeper installation.
2024-07-27 00:29:29,460 INFO sqoop.Sqoop: Running Sqoop version: 1.4.7
2024-07-27 00:29:29,611 WARN tool.BaseSqoopTool: Setting your password on the command-line is insecure. Consider using -P instead.
2024-07-27 00:29:29,778 INFO manager.MySQLManager: Preparing to use a MySQL streaming resultset.
categories
customers
departments
order_items
orders
products
[vagrant@10 ~]$
```

- Afficher la liste des tables contenues dans retail_db

sqoop list-tables \ --connect "jdbc:mysql://host_address:3306/retail_db" \ --username retail_user \ --password hadoop

```
[vagrant@10 ~]$ sqoop list-tables --connect "jdbc:mysql://192.168.1.21:3306/retail_db" --username retail_user --password hadoop
Warning: /usr/lib/sqoop/../hbase does not exist! HBase imports will fail.
Please set $HBASE_HOME to the root of your HBase installation.
Warning: /usr/lib/sqoop/../hcatalog does not exist! HCatalog jobs will fail.
Please set $HCAT_HOME to the root of your HCatalog installation.
Warning: /usr/lib/sqoop/../accumulo does not exist! Accumulo imports will fail.
Please set $ACCUMULO_HOME to the root of your Accumulo installation.
Warning: /usr/lib/sqoop/../zookeeper does not exist! Accumulo imports will fail.
Please set $ZOOKEEPER_HOME to the root of your Zookeeper installation.
2024-07-27 00:32:17,322 INFO sqoop.Sqoop: Running Sqoop version: 1.4.7
2024-07-27 00:32:17,478 WARN tool.BaseSqoopTool: Setting your password on the command-line is insecure. Consider using -P instead.
2024-07-27 00:32:17,628 INFO manager.MySQLManager: Preparing to use a MySQL streaming resultset.
Sat Jul 27 00:32:17 UTC 2024 WARN: Establishing SSL connection without server's identity verification is not recommended. According t
o MySQL 5.5.45+, 5.6.26+ and 5.7.6+ requirements SSL connection must be established by default if explicit option isn't set. For comp
liance with existing applications not using SSL the verifyServerCertificate property is set to 'false'. You need either to explicitly
 disable SSL by setting useSSL=false, or set useSSL=true and provide truststore for server certificate verification.
categories
customers
departments
order_items
orders
products
[vagrant@10 ~]$
```

- Importer chaque table de la base de données retail_db dans Hive en utilisant la requête sqoop ci-dessous : Il faut remplace la variable tablename par le nom de la table que vous voulez importer.

sqoop import \ --connect "jdbc:mysql://@IP_hostname:3306/retail_db" \ --username=retail_user \ --password=hadoop \ --table tablename \ --as-parquetfile \ --target-dir=/user/hive/warehouse/retail_db.db/{tablename} \ --delete-target-dir

- Order_items

```
2024-07-27 00:38:18,823 INFO mapreduce.ImportJobBase: Transferred 1.6101 MB in 49.5789 seconds (33.2554 KB/sec)
2024-07-27 00:38:18,840 INFO mapreduce.ImportJobBase: Retrieved 172198 records.
```

- Products

```
2024-07-27 00:40:20,479 INFO mapreduce.ImportJobBase: Transferred 69.2285 KB in 38.0419 seconds (1.8198 KB/sec)
2024-07-27 00:40:20,486 INFO mapreduce.ImportJobBase: Retrieved 1345 records.
```

- Categories

```
2024-07-27 00:41:59,019 INFO mapreduce.ImportJobBase: Transferred 10.8271 KB in 36.0246 seconds (307.7618 bytes/sec)
2024-07-27 00:41:59,034 INFO mapreduce.ImportJobBase: Retrieved 58 records.
```

- Orders

```
2024-07-27 00:43:30,069 INFO mapreduce.ImportJobBase: Transferred 586.6133 KB in 36.9966 seconds (15.8559 KB/sec)
2024-07-27 00:43:30,083 INFO mapreduce.ImportJobBase: Retrieved 68883 records.
```

- Customers

```
2024-07-27 00:45:12,418 INFO mapreduce.ImportJobBase: Transferred 360.6367 KB in 33.9146 seconds (10.6337 KB/sec)
2024-07-27 00:45:12,431 INFO mapreduce.ImportJobBase: Retrieved 12435 records.
```

- Departments

```
2024-07-27 00:46:45,651 INFO mapreduce.ImportJobBase: Transferred 7.5078 KB in 34.3488 seconds (223.8216 bytes/sec)
2024-07-27 00:46:45,674 INFO mapreduce.ImportJobBase: Retrieved 6 records.
```

- Vérifions si les données ont été intégrées dans Warehouse de hive

hdfs dfs -ls user/hive/warehouse/retail_retail_db.db

```
[vagrant@10 ~]$ hdfs dfs -ls /user/hive/warehouse/retail_db
Found 6 items
drwxr-xr-x   - vagrant supergroup          0 2024-07-27 00:41 /user/hive/warehouse/retail_db/categories
drwxr-xr-x   - vagrant supergroup          0 2024-07-27 00:45 /user/hive/warehouse/retail_db/customers
drwxr-xr-x   - vagrant supergroup          0 2024-07-27 00:46 /user/hive/warehouse/retail_db/departments
drwxr-xr-x   - vagrant supergroup          0 2024-07-27 00:38 /user/hive/warehouse/retail_db/order_items
drwxr-xr-x   - vagrant supergroup          0 2024-07-27 00:43 /user/hive/warehouse/retail_db/orders
drwxr-xr-x   - vagrant supergroup          0 2024-07-27 00:40 /user/hive/warehouse/retail_db/products
[vagrant@10 ~]$
```

```
[vagrant@10 ~]$ hdfs dfs -ls /user/hive/warehouse/retail_db/order_items
Found 6 items
drwxr-xr-x   - vagrant supergroup          0 2024-07-27 00:37 /user/hive/warehouse/retail_db/order_items/.metadata
drwxr-xr-x   - vagrant supergroup          0 2024-07-27 00:38 /user/hive/warehouse/retail_db/order_items/.signals
-rw-r--r--   1 vagrant supergroup     415558 2024-07-27 00:38 /user/hive/warehouse/retail_db/order_items/2f9d047d-de95-496c-a0d8-9f06
b5bb173c.parquet
-rw-r--r--   1 vagrant supergroup     415617 2024-07-27 00:38 /user/hive/warehouse/retail_db/order_items/3651c6c2-2f1d-4a33-8089-be20
51ffa5e4.parquet
-rw-r--r--   1 vagrant supergroup     415333 2024-07-27 00:38 /user/hive/warehouse/retail_db/order_items/478b38e6-8d60-4874-9634-7e40
38004d8d.parquet
-rw-r--r--   1 vagrant supergroup     431795 2024-07-27 00:38 /user/hive/warehouse/retail_db/order_items/a1d99c51-d79a-40b3-ba66-5f86
6e819faa.parquet
[vagrant@10 ~]$
```

- Se connecter dans hive et vérifions si les tables ont été crée

$ hive

$ show tables ;

#Les tables ne sont pas crées donc, on dois les crées.

II. Data processing avec Apache Hive

❖ Apercu :

Qu‘est-ce que Apache Hive ? Hive fournit un mécanisme permettant d‘interroger, de créer et de gérer de grands ensembles de données stockés sur Hadoop, à l‘aide d‘instructions de type SQL. Il permet également d‘ajouter une structure aux données existantes qui résident sur HDFS. Dans ce billet, je décrirai une approche pratique sur la façon d‘ingérer des données dans Hive, avec la plateforme d‘intégration élastique SnapLogic, sans avoir besoin d‘écrire du code.

❖ Travail a faire :

- Se connecter dans hive et créer les tables

```
hive> CREATE EXTERNAL TABLE IF NOT EXISTS customers (
    > customer_id int,
    > customer_fname STRING,
    > customer_lname STRING,
    > customer_email STRING,
    > customer_password STRING,
    > customer_street STRING,
    > customer_city STRING,
    > customer_state STRING,
    > customer_zipcode STRING
    > )
    > ROW FORMAT DELIMITED
    > FIELDS TERMINATED BY ','
    > STORED AS PARQUET
    > LOCATION 'hdfs:///user/hive/warehouse/retail_db.db/customers';
OK
Time taken: 0.998 seconds
```

```
hive> CREATE EXTERNAL TABLE IF NOT EXISTS orders (
    > order_id INT,
    > order_date STRING,
    > order_customer_id INT,
    > order_status STRING
    > )
    > ROW FORMAT DELIMITED
    > FIELDS TERMINATED BY ','
    > STORED AS PARQUET
    > LOCATION 'hdfs:///user/hive/warehouse/retail_db.db/orders';
OK
Time taken: 0.464 seconds
```

```
hive> CREATE EXTERNAL TABLE IF NOT EXISTS products(
    >     product_id INT,
    >     product_category_id INT,
    >     product_name STRING,
    >     product_description STRING,
    >     product_price FLOAT, -- Removed NOT NULL constraint
    >     product_image STRING
    > )
    > STORED AS PARQUET
    > LOCATION 'hdfs:///user/hive/warehouse/retail_db/products';
OK
Time taken: 0.383 seconds
```

```
hive>
    > CREATE EXTERNAL TABLE IF NOT EXISTS categories (
    > category_id INT,
    > category_department_id INT,
    > category_name STRING
    > )
    > ROW FORMAT DELIMITED
    > FIELDS TERMINATED BY ','
    > STORED AS PARQUET
    > LOCATION 'hdfs:///user/hive/warehouse/retail_db.db/categories';
OK
Time taken: 0.248 seconds
```

```
hive> CREATE EXTERNAL TABLE IF NOT EXISTS departments (
    > department_id INT ,
    > department_name STRING
    > )
    > ROW FORMAT DELIMITED
    > FIELDS TERMINATED BY ','
    > STORED AS PARQUET
    > LOCATION 'hdfs:///user/hive/warehouse/retail_db.db/departments';
OK
Time taken: 1.767 seconds
```

```
hive> CREATE EXTERNAL TABLE IF NOT EXISTS order_items (
    > order_item_id INT,
    > order_item_order_id INT,
    > order_item_product_id INT,
    > order_item_quantity INT,
    > order_item_subtotal DOUBLE ,
    > order_item_product_price DOUBLE
    > )
    > ROW FORMAT DELIMITED
    > FIELDS TERMINATED BY ','
    > STORED AS PARQUET
    > LOCATION 'hdfs:///user/hive/warehouse/retail_db.db/order_items';
OK
Time taken: 0.263 seconds
```

- Lister les tables dans hive avec show tables

```
hive>
    > show tables;
OK
categories
customers
departments
order_items
orders
products
Time taken: 0.315 seconds, Fetched: 6 row(s)
hive>
```

Exercices : Répondre aux questions en fournissant la requête SQL correspondant à chaque question.

1. Trouver le nombre total de commandes passées par chaque client au cours de l'année 2014. Le statut de la commande doit être COMPLET, le format order_date est au format unix_timestamp
   - hive>
     > SELECT order_customer_id AS customer_id, COUNT(*) AS total_orders
     > FROM orders
     > WHERE order_status = 'COMPLET'
     > AND YEAR(FROM_UNIXTIME(CAST(order_date AS BIGINT))) = 2014
     > GROUP BY order_customer_id;

```
MapReduce Jobs Launched:
Stage-Stage-1: Map: 1  Reduce: 1   Cumulative CPU: 7.94 sec   HDFS Read: 17610 HDFS Write: 87 SUCCESS
Total MapReduce CPU Time Spent: 7 seconds 940 msec
OK
Time taken: 85.025 seconds
```

2. Afficher le nom et le prénom des clients qui n'ont passé aucune commande, triés par customer_lname puis customer_fname.
   - hive> SELECT c.customer_lname, c.customer_fname
     > FROM customers c
     > LEFT JOIN orders o ON c.customer_id = o.order_customer_id

> WHERE o.order_id IS NULL
> ORDER BY c.customer_lname, c.customer_fname;

```
MapReduce Jobs Launched:
Stage-Stage-2: Map: 1  Reduce: 1   Cumulative CPU: 10.27 sec   HDFS Read: 18965 HDFS Write: 87 SUCCESS
Total MapReduce CPU Time Spent: 10 seconds 270 msec
OK
Time taken: 108.986 seconds
hive>
```

3. Afficher les détails des top 5 clients par revenue pour chaque mois. Vous devez obtenir tous les détails du client ainsi que le mois et les revenus par mois. Les données doivent être triées par mois dans l'ordre croissant et les revenus par mois dans l'ordre décroissant
   - hive> WITH MonthlyRevenue AS (
     > SELECT c.customer_id, c.customer_fname, c.customer_lname,
     > DATE_FORMAT(FROM_UNIXTIME(CAST(o.order_date AS BIGINT)), '%Y-%m') AS month,
     > SUM(oi.order_item_subtotal) AS monthly_revenue
     > FROM customers c
     > JOIN orders o ON c.customer_id = o.order_customer_id
     > JOIN order_items oi ON o.order_id = oi.order_item_order_id
     > WHERE o.order_status IN ('COMPLET', 'CLOSED')
     > GROUP BY c.customer_id, c.customer_fname, c.customer_lname,
     > DATE_FORMAT(FROM_UNIXTIME(CAST(o.order_date AS BIGINT)), '%Y-%m')
     >)
     > SELECT customer_id, customer_fname, customer_lname, month, monthly_revenue
     > FROM (
     > SELECT customer_id, customer_fname, customer_lname, month, monthly_revenue,
     > ROW_NUMBER() OVER (PARTITION BY month ORDER BY monthly_revenue DESC) AS rank
     > FROM MonthlyRevenue
     >) ranked
     > WHERE rank <= 5
     > ORDER BY month, monthly_revenue DESC;

```
MapReduce Jobs Launched:
Stage-Stage-3: Map: 1  Reduce: 1   Cumulative CPU: 12.36 sec   HDFS Read: 27096 HDFS Write: 96 SUCCESS
Stage-Stage-4: Map: 1  Reduce: 1   Cumulative CPU: 10.9 sec   HDFS Read: 11052 HDFS Write: 96 SUCCESS
Stage-Stage-5: Map: 1  Reduce: 1   Cumulative CPU: 9.21 sec   HDFS Read: 8744 HDFS Write: 87 SUCCESS
Total MapReduce CPU Time Spent: 32 seconds 470 msec
OK
Time taken: 397.811 seconds
```

4. Trouver toutes les commandes terminées ou fermées (completed ou closed), puis calculez le revenu total pour chaque jour pour chaque département. La sortie doit afficher : order_date, department_name et order_revenue
   - hive> WITH DepartmentRevenue AS (
     > SELECT DATE(FROM_UNIXTIME(CAST(o.order_date AS BIGINT))) AS order_date,

```
>        d.department_name,
>        SUM(oi.order_item_subtotal) AS order_revenue
>   FROM orders o
>   JOIN order_items oi ON o.order_id = oi.order_item_order_id
>   JOIN products p ON oi.order_item_product_id = p.product_id
>   JOIN categories c ON p.product_category_id = c.category_id
>   JOIN departments d ON c.category_department_id = d.department_id
>   WHERE o.order_status IN ('COMPLET', 'CLOSED')
>      GROUP BY DATE(FROM_UNIXTIME(CAST(o.order_date AS BIGINT))),
d.department_name
> )
> SELECT order_date, department_name, order_revenue
> FROM DepartmentRevenue
> ORDER BY order_date, department_name;
```

```
MapReduce Jobs Launched:
Stage-Stage-13: Map: 1   Cumulative CPU: 8.42 sec   HDFS Read: 11870 HDFS Write: 96 SUCCESS
Stage-Stage-15: Map: 1   Cumulative CPU: 7.55 sec   HDFS Read: 8506 HDFS Write: 96 SUCCESS
Stage-Stage-10: Map: 1   Cumulative CPU: 5.13 sec   HDFS Read: 8499 HDFS Write: 96 SUCCESS
Stage-Stage-4: Map: 1  Reduce: 1   Cumulative CPU: 8.02 sec   HDFS Read: 7357 HDFS Write: 96 SUCCESS
Stage-Stage-5: Map: 1  Reduce: 1   Cumulative CPU: 3.12 sec   HDFS Read: 8293 HDFS Write: 87 SUCCESS
Total MapReduce CPU Time Spent: 32 seconds 240 msec
OK
Time taken: 703.092 seconds
hive>
```

5. Trouver le rank de chaque catégorie par revenue obtenue dans chaque département à partir de toutes les transactions. Affichez les résultats par department_name et classez-les par ordre croissant.
   -
```
hive> WITH CategoryRevenue AS (
   >   SELECT d.department_name, c.category_name,
   >        SUM(oi.order_item_subtotal) AS total_revenue
   >   FROM orders o
   >   JOIN order_items oi ON o.order_id = oi.order_item_order_id
   >   JOIN products p ON oi.order_item_product_id = p.product_id
   >   JOIN categories c ON p.product_category_id = c.category_id
   >   JOIN departments d ON c.category_department_id = d.department_id
   >   WHERE o.order_status IN ('COMPLET', 'CLOSED')
   >   GROUP BY d.department_name, c.category_name
   > )
   > SELECT department_name, category_name, total_revenue,
   >      RANK() OVER (PARTITION BY department_name ORDER BY total_revenue
DESC) AS rank
   > FROM CategoryRevenue
   > ORDER BY department_name, rank;
```

```
MapReduce Jobs Launched:
Stage-Stage-14: Map: 1   Cumulative CPU: 2.69 sec   HDFS Read: 11381 HDFS Write: 96 SUCCESS
Stage-Stage-16: Map: 1   Cumulative CPU: 4.98 sec   HDFS Read: 8655 HDFS Write: 96 SUCCESS
Stage-Stage-11: Map: 1   Cumulative CPU: 1.94 sec   HDFS Read: 7204 HDFS Write: 96 SUCCESS
Stage-Stage-4: Map: 1  Reduce: 1   Cumulative CPU: 3.38 sec   HDFS Read: 7261 HDFS Write: 96 SUCCESS
Stage-Stage-5: Map: 1  Reduce: 1   Cumulative CPU: 3.29 sec   HDFS Read: 9868 HDFS Write: 96 SUCCESS
Stage-Stage-6: Map: 1  Reduce: 1   Cumulative CPU: 3.23 sec   HDFS Read: 8389 HDFS Write: 87 SUCCESS
Total MapReduce CPU Time Spent: 19 seconds 510 msec
OK
Time taken: 410.16 seconds
```

6. Afficher le pourcentage de chaque catégorie par revenue dans chaque département. Afficher les résultats par department_name et pourcentage par ordre décroissant.
   - hive> WITH TotalRevenue AS (
     > SELECT d.department_name,
     > SUM(oi.order_item_subtotal) AS department_total
     > FROM orders o
     > JOIN order_items oi ON o.order_id = oi.order_item_order_id
     > JOIN products p ON oi.order_item_product_id = p.product_id
     > JOIN categories c ON p.product_category_id = c.category_id
     > JOIN departments d ON c.category_department_id = d.department_id
     > WHERE o.order_status IN ('COMPLET', 'CLOSED')
     > GROUP BY d.department_name
     >),
     > CategoryRevenue AS (
     > SELECT d.department_name, c.category_name,
     > SUM(oi.order_item_subtotal) AS category_total
     > FROM orders o
     > JOIN order_items oi ON o.order_id = oi.order_item_order_id
     > JOIN products p ON oi.order_item_product_id = p.product_id
     > JOIN categories c ON p.product_category_id = c.category_id
     > JOIN departments d ON c.category_department_id = d.department_id
     > WHERE o.order_status IN ('COMPLET', 'CLOSED')
     > GROUP BY d.department_name, c.category_name
     >)
     > SELECT c.department_name, c.category_name,
     > (c.category_total / t.department_total * 100) AS percentage
     > FROM CategoryRevenue c
     > JOIN TotalRevenue t ON c.department_name = t.department_name
     > ORDER BY c.department_name, percentage DESC;

```
MapReduce Jobs Launched:
Stage-Stage-25: Map: 1   Cumulative CPU: 2.75 sec   HDFS Read: 11393 HDFS Write: 96 SUCCESS
Stage-Stage-27: Map: 1   Cumulative CPU: 3.74 sec   HDFS Read: 8873 HDFS Write: 96 SUCCESS
Stage-Stage-31: Map: 1   Cumulative CPU: 2.91 sec   HDFS Read: 11394 HDFS Write: 96 SUCCESS
Stage-Stage-33: Map: 1   Cumulative CPU: 2.25 sec   HDFS Read: 8729 HDFS Write: 96 SUCCESS
Stage-Stage-22: Map: 1   Cumulative CPU: 1.7 sec   HDFS Read: 7215 HDFS Write: 96 SUCCESS
Stage-Stage-28: Map: 1   Cumulative CPU: 1.58 sec   HDFS Read: 7155 HDFS Write: 96 SUCCESS
Stage-Stage-4: Map: 1  Reduce: 1   Cumulative CPU: 2.92 sec   HDFS Read: 7265 HDFS Write: 96 SUCCESS
Stage-Stage-14: Map: 1  Reduce: 1   Cumulative CPU: 3.07 sec   HDFS Read: 6991 HDFS Write: 96 SUCCESS
Stage-Stage-19: Map: 1   Cumulative CPU: 1.65 sec   HDFS Read: 7388 HDFS Write: 96 SUCCESS
Stage-Stage-6: Map: 1  Reduce: 1   Cumulative CPU: 3.16 sec   HDFS Read: 7884 HDFS Write: 87 SUCCESS
Total MapReduce CPU Time Spent: 25 seconds 730 msec
OK
Time taken: 741.803 seconds
```

7. Afficher tous les clients qui ont passé une commande d'un montant supérieur à 200 $.
   - hive> SELECT DISTINCT c.customer_id, c.customer_fname, c.customer_lname
     > FROM customers c
     > JOIN orders o ON c.customer_id = o.order_customer_id
     > JOIN order_items oi ON o.order_id = oi.order_item_order_id
     > WHERE oi.order_item_subtotal > 200;

```
MapReduce Jobs Launched:
Stage-Stage-3: Map: 1  Reduce: 1   Cumulative CPU: 8.65 sec   HDFS Read: 23968 HDFS Write: 87 SUCCESS
Total MapReduce CPU Time Spent: 8 seconds 650 msec
OK
Time taken: 77.015 seconds
hive>
```

8. Afficher les clients de la "customers" dont les noms customer_fname commence par "Rich"
   - hive> SELECT customer_id, customer_fname, customer_lname
     > FROM customers
     > WHERE customer_fname LIKE 'Rich%';

```
hive>
    > SELECT customer_id, customer_fname, customer_lname
    > FROM customers
    > WHERE customer_fname LIKE 'Rich%';
OK
Time taken: 0.152 seconds
hive>
```

9. Fournir le nombre total de clients dans chaque état (state) dont le prénom commence par « M »
   - hive> SELECT customer_state, COUNT(*) AS total_customers
     > FROM customers
     > WHERE customer_fname LIKE 'M%'
     > GROUP BY customer_state;

```
MapReduce Jobs Launched:
Stage-Stage-1: Map: 1  Reduce: 1   Cumulative CPU: 4.65 sec   HDFS Read: 16571 HDFS Write: 87 SUCCESS
Total MapReduce CPU Time Spent: 4 seconds 650 msec
OK
Time taken: 63.359 seconds
hive>
```

10. Trouver le produit le plus cher dans chaque catégorie
    - hive> SELECT c.category_name, MAX(p.product_price) AS max_price
      > FROM products p
      > JOIN categories c ON p.product_category_id = c.category_id
      > GROUP BY c.category_name;

```
MapReduce Jobs Launched:
Stage-Stage-2: Map: 1  Reduce: 1   Cumulative CPU: 13.09 sec   HDFS Read: 35991 HDFS Write: 87 SUCCESS
Total MapReduce CPU Time Spent: 13 seconds 90 msec
OK
Time taken: 81.684 seconds
hive>
```

11. Trouvez les 10 meilleurs produits qui ont généré les revenus les plus élevés.

- hive> SELECT p.product_id, p.product_name, SUM(oi.order_item_subtotal) AS total_revenue
  > FROM orders o
  > JOIN order_items oi ON o.order_id = oi.order_item_order_id
  > JOIN products p ON oi.order_item_product_id = p.product_id
  > WHERE o.order_status IN ('COMPLET', 'CLOSED')
  > GROUP BY p.product_id, p.product_name
  > ORDER BY total_revenue DESC
  > LIMIT 10;

```
MapReduce Jobs Launched:
Stage-Stage-3: Map: 1  Reduce: 1   Cumulative CPU: 8.54 sec   HDFS Read: 23533 HDFS Write: 96 SUCCESS
Stage-Stage-4: Map: 1  Reduce: 1   Cumulative CPU: 3.26 sec   HDFS Read: 8433 HDFS Write: 87 SUCCESS
Total MapReduce CPU Time Spent: 11 seconds 800 msec
OK
Time taken: 132.448 seconds
hive>
```

Sources :

https://blog.stacklabs.com/code/data_ingestion_state_the_art/#:~:text=Apache%20Sqoop%20est%20un%20outil,SQL%20depuis%20la%20derni%C3%A8re%20importation.

https://www.snaplogic.com/fr/blog/big-data-ingestion-patterns-ingesting-data-from-cloud-ground-sources-into-hive