REPUBLIQUE DU SENEGAL

Un peuple –un but –une foi

MINISTERE DE L'ENSEIGNEMENT SUPERIEURE DE LA RECHERCHE ET DE L'INNOVATION

DIRECTION GENERALE DE L'ENSEIGNEMENT SUPERIEUR

Université Gaston Berger

UFR Institut Polytechnique de Saint-Louis

Ingestion de données dans Big Data

Présenté par :

Bassirou SAGNA

Sous la direction de :

Dr. Djibril MBOUP

ING 3 Info-Telecom

Plan :

Introduction :

Dans le monde numérique actuel, la gestion et l'analyse de vastes quantités de données sont devenues des compétences cruciales pour les entreprises. La capacité à ingérer, stocker et traiter de grandes quantités de données rapidement et efficacement permet aux organisations de prendre des décisions éclairées et d'améliorer leurs performances opérationnelles. Le projet d'ingestion de données dans Big Data, que nous abordons ici, se divise en deux parties essentielles: l'ingestion des données avec Apache Sqoop et le traitement des données avec Apache Hive.

Apache Sqoop qui est un outil puissant pour le transfert de données entre les systèmes de gestion de bases de données (SGBD) relationnels et Hadoop sera la première partie pour le traitement de notre projet. Dans cette première partie du projet, nous nous concentrerons sur l'ingestion des données provenant d'une base de donnée vers le système Hadoop. Apache Sqoop facilite ce processus en permettant l'importation et l'exportation des données de manière efficace, en minimisant l'intervention manuelle et en optimisant les performances.

L'ingestion de données est la première étape cruciale dans le domaine des Big Data. Elle implique de transporter des données brutes depuis différentes sources, telles que des bases de données SQL, des systèmes ERP, des applications CRM, ou même des fichiers CSV, vers un environnement Hadoop pour un traitement et une analyse ultérieurs. Apache Sqoop se distingue par sa capacité à automatiser et à simplifier ce transfert de données, tout en garantissant l'intégrité et la cohérence des données.

Une fois que nous aurons ingérées les données dans le système Hadoop, la deuxième partie de notre projet consiste à les traiter et les analyser en utilisant Apache Hive. Hive est un outil d'entreposage de données construit sur Hadoop, qui fournit une interface de requête similaire à SQL pour faciliter l'accès et l'analyse des données stockées dans Hadoop. Grâce à Hive, les

utilisateurs peuvent écrire des requêtes SQL pour analyser de grandes quantités de données sans avoir à se plonger dans les complexités du code MapReduce.

L'objectif de cette etude est de permettre de réaliser une **Ingestion des données avec Apache Sqoop** et ensuite faire un **Data Processing avec Apache Hive** en faisant des requêtes SQL pour traiter les données importées.

PART I : Ingestion des données avec Apache Sqoop :

Apache Sqoop est un outil spécialement conçu pour transférer des données entre les systèmes de gestion de bases de données (SGBD) relationnels et Hadoop. Dans cette première partie du projet, nous nous concentrerons sur l'ingestion des données provenant de diverses sources relationnelles vers le système Hadoop

Pour la réalisation de celle-ci il faudrait en premier telecharger sur le drive via le lien suivant

https://drive.google.com/file/d/1CHwWhfJn4edCuAOHiWr6iyT4wJ-zPNbU/view?usp=share_link le téléchargement de notre base de donner va nous permettre de faire l'importation des donnée rationnel dans Hadoop.

Avant de commencer les travaux à faire nous allons en premier parler de notre base de donner Retail DB qui est une base de données qui contient des données de ventes d'une entreprise e-commerce. Cette base de données comporte 6 tables :

- Departments ;
- Categories ;
- Products ;
- Order Items ;
- Orders ;
- Customers.

Figure 1: Schéma de la base de données Retail DB

La figure 1 nous montre le schéma de notre base de donnée ainsi que les différentes tables ainsi que leur liaison.

Pour l'importation de notre base de données nous allons importer notre base de donnée et créer l'utilisateur dans notre machine hôte comme suit :

```
C:\Windows\system32>mysql -u root -p
Enter password:
Welcome to the MariaDB monitor.  Commands end with ; or \g.
Your MariaDB connection id is 8
Server version: 10.4.32-MariaDB mariadb.org binary distribution

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

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

MariaDB [(none)]>  CREATE user retail_user identified by 'hadoop';
Query OK, 0 rows affected (0.004 sec)

MariaDB [(none)]> CREATE database retail_db;
Query OK, 1 row affected (0.001 sec)

MariaDB [(none)]>  GRANT ALL ON retail_db.* to retail_dba;
Query OK, 0 rows affected (0.003 sec)

MariaDB [(none)]>  flush privileges;
Query OK, 0 rows affected (0.001 sec)
```

Figure 2 : création utilisateur et de la base de donnée

Après la création de notre utilisateur et de la base de donnée nous allons nous connecter avec le nouvel utilisateur pour voir si la création s'est bien passer comme suit :

```
C:\Windows\system32>mysql -u retail_user -phadoop
Welcome to the MariaDB monitor.  Commands end with ; or \g.
Your MariaDB connection id is 11
Server version: 10.4.32-MariaDB mariadb.org binary distribution

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

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

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

Figure 3 : connexion avec le nouvel utilisateur.

Après la connexion avec le nouvel utilisateur nous allons nous placer sur la base de donnée concerner à l'occurrence **retail_db** apres cela nous allons maintenant lancer l'importation de notre base de donnée qui se trouve sur **C:/Users/lenovo/Desktop/Documents/Cours/IPSL-Ing3/BigData** comme suit :

```
MariaDB [retail_db]> source C:/Users/lenovo/Desktop/Documents/Cours/IPSL-Ing3/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 (0.000 sec)

Query OK, 0 rows affected (0.000 sec)

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

Figure 4 : Importation de la base de données

Si nous avons fini avec l'importation nous allons vérifier si l'importations c'est bien fait en essayant de vois les tables présentes sur la base de donnée :

```
MariaDB [retail_db]>  show tables;
+---------------------+
| Tables_in_retail_db |
+---------------------+
| categories          |
| customers           |
| departments         |
| order_items         |
| orders              |
| products            |
+---------------------+
6 rows in set (0.002 sec)
```

Figure 5 : Vérifications des tables

Une fois l'importation de notre base de donnée terminer nous allons nous connecter sur notre machine vagrant et se connecter, une fois que nous nous sommes bien connecter nous allons nous lancer maintenant au lancement des service Hadoop avec la commande **start-all.sh** cette commande permet le Démarrage Complet des Services Hadoop celle-ci lance tous les services Hadoop . Cela inclut les services suivants :

- NameNode : Le serveur principal qui gère la métadonnée du système de fichiers Hadoop (HDFS).
- DataNode : Les serveurs de stockage qui stockent les blocs de données dans HDFS.
- ResourceManager : Le serveur principal qui gère les ressources de traitement et planifie les tâches dans le cadre de YARN (Yet Another Resource Negotiator).
- NodeManager : Les agents qui gèrent les ressources sur chaque nœud du cluster et supervisent les conteneurs d'application YARN.
- SecondaryNameNode : Un processus auxiliaire qui effectue des sauvegardes périodiques du NameNode et aide à la récupération en cas de défaillance.

```
[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]
Starting datanodes
Starting secondary namenodes [10.0.2.15]
Starting resourcemanager
Starting nodemanagers
```

Figure 6 : lancement des services de hadoop

Une fois le lancement des services fais nous allons donc vérifier si la machine virtuelle et votre machine locale sont dans le même réseau en testant avec la commande Sqoop ci-dessous :

**sqoop list-databases **

**--connect "jdbc:mysql://B-SAGNA3306" **

**--username retail_user **

--password hadoop

Cette commande nous permet de lister les Bases de Données c'est-à-dire qu'elle permet de récupérer une liste de toutes les bases de données présentes dans le serveur MySQL auquel nous sommes connecté. Cela peut être utile pour vérifier que nous avez accès aux bases de données attendues ou pour identifier la base de données spécifique que vous souhaitez importer avec Sqoop comme le montre ci-bien la capture suivante :

```
[vagrant@10 ~]$ sqoop list-databases --connect "jdbc:mysql://B-SAGNA:3306" --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.
2024-07-22 21:49:56,924 INFO sqoop.Sqoop: Running Sqoop version: 1.4.7
2024-07-22 21:49:57,099 WARN tool.BaseSqoopTool: Setting your password on the command-line is insecure. Consider using -P instead.
2024-07-22 21:49:57,323 INFO manager.MySQLManager: Preparing to use a MySQL streaming resultset.
information_schema
retail_db
```

Figure 7 : Liste des Base de donnée avec scoop

Après cela nous pouvons passer à la liste des tables contenues dans **retail_db** avec la commande

**sqoop list-tables **

**--connect "jdbc:mysql://B-SAGNA:3306/retail_db" **

**--username retail_user **

--password hadoop

Cette commande permet de récupérer la liste de toutes les tables dans la base de données **retail_db** sur le serveur MySQL spécifié. Cela est particulièrement utile pour vérifier la structure de la base de données et choisir les tables à importer ou à analyser comme le montre la figure ci-dessous :

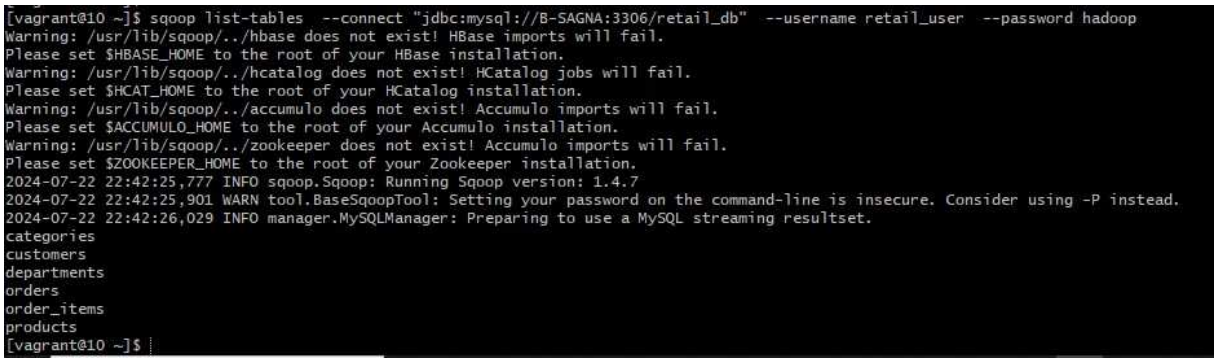


Figure 7 : Listes des Tables avec Scoop

Une fois que nous avons fini de verifier si notre machine virtuelle vois la base de donnée nous allons donc commencer à importer chaque table de la base de données **retail_db** dans Hive en utilisant la requête sqoop ci-dessous pour ce faire il faut remplace la variable **tablename** par le nom de la table que vous voulez importer :

**sqoop import **

**--connect "jdbc:mysql://B-SAGNA:3306/retail_db" **

**--username=retail_user **

**--password=hadoop **

**--table tablename **

**--as-parquetfile **

**--target-dir=/user/hive/warehouse/retail_db/{tablename} **

--delete-target-dir

La commande **sqoop import** nous permet de transférer des données depuis une base de données relationnelle vers Hadoop, en spécifiant le format de fichier et le répertoire de destination. Les options fournies permettent d'optimiser l'importation en stockant les données en format Parquet et en nettoyant le répertoire de destination avant l'importation comme le montre l'exemple ci-dessous :

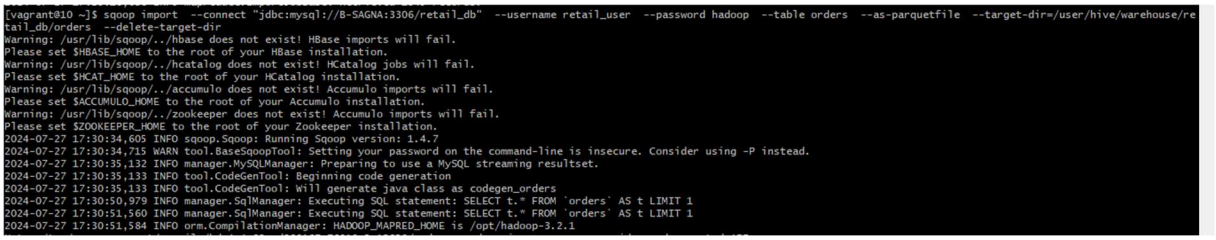


Figure 8 : Importation des tables dans Hive

Une fois l'importation terminer nous allons vérifier si les données ont été ingérés dans le warehouse de Hive avec la commande :

hdfs dfs -ls user/hive/warehouse/retail_db

Cette commande affiche la liste des fichiers et sous-répertoires situés dans le répertoire HDFS user/hive/warehouse/retail_db. Elle permet de lister fichiers et sous-répertoires, y compris les permissions, le nombre de blocs, la taille, le propriétaire, le groupe, et la date de la dernière modification comme le montre la capture suivante :

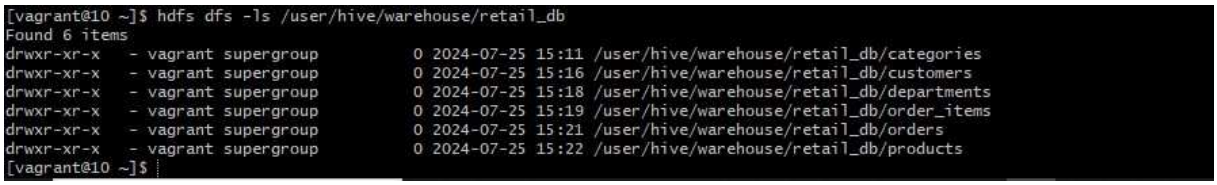


Figure 9 : Liste des fichiers importer dans le hive

Une fois cette étape terminer nous allons nous placer et voire si les tables ont été bien importe comme suit :

Figure 10 : Connexion dans Hive

Nous ne parvenons pas à voir les tables importer dans notre base de donnée

L'utilisation d'Apache Sqoop pour l'ingestion de données est une étape essentielle pour intégrer des données relationnelles dans un environnement Big Data. Sqoop simplifie et automatise ce processus, assurant un transfert de données efficace, fiable et optimisé. En maîtrisant les fonctionnalités de Sqoop, les entreprises peuvent faciliter l'intégration de leurs données opérationnelles dans Hadoop, ouvrant ainsi la voie à des analyses approfondies et à des décisions basées sur des données précises. De ce fait Apache Sqoop permet d'importer efficacement des données depuis des bases de données SQL (comme MySQL, PostgreSQL, Oracle) vers le système de fichiers Hadoop (HDFS). Cette fonctionnalité est cruciale pour intégrer des données provenant de systèmes opérationnels dans un environnement de Big Data pour une analyse approfondie.

PART II: Data Processing avec Apache Hive :

Apache Hive est un système d'entreposage de données construit sur Hadoop, qui facilite l'analyse de grandes quantités de données stockées dans le système de fichiers Hadoop (HDFS). Il fournit une interface de requête SQL-like pour effectuer des analyses complexes et gérer des ensembles de données volumineux. Dans cette partie du projet, nous allons explorer comment utiliser Apache Hive pour le traitement des données importées et effectuer des requêtes SQL.

Comme dit en amont dans notre première partie nous ne voyons pas les tables importer de ce fait nous allons demander à hive d'aller chercher pour chaque table le fichier dont nous les avons gardées à l'aide de la commande :

```
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/customers';
```

```
        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/orders';


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/order_items';
```

```
CREATE EXTERNAL TABLE IF NOT EXISTS products(
        product_id INT ,
        product_category_id INT,
        product_name STRING,
        product_description STRING,
        product_price float NOT NULL,
        product_image STRING
)
ROW FORMAT DELIMITED
FIELDS TERMINATED BY ','
STORED AS PARQUET
LOCATION 'hdfs:///user/hive/warehouse/retail_db/products';

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/categories';

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/departments';

Les commandes que nous avons fournies permettent de créer des tables externes dans Hive. Les tables externes sont des tables où les données sont stockées à un emplacement externe spécifié, en l'occurrence, dans le système de fichiers HDFS. Les commandes fournies permettent de créer des tables externes dans Hive pour diverses entités telles que les commandes, les produits, les catégories et les départements ... En utilisant le format de stockage Parquet et en définissant les emplacements dans HDFS, ces tables facilitent le traitement et l'analyse de grandes quantités de données avec Apache Hive. Une fois les tables créées, vous pouvez effectuer des requêtes HiveQL pour extraire des informations précieuses et effectuer des analyses complexes comme le montre la figures ci-dessus :

```
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/customers';
OK
Time taken: 0.183 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/orders';
OK
Time taken: 0.083 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/order_items';
OK
Time taken: 0.071 seconds
```

Figure 11 : Création des tables dans Hive

Vérifions si nous avons bien créer les tables comme dans la base de donnée **retail_db** comme suit :

```
hive> show tables;
OK
categories
customers
departments
order_items
orders
products
Time taken: 0.123 seconds, Fetched: 6 row(s)
```

Figure 12 : Vérification des tables dans hive

Une fois que tous les tables bien créer nous allons nous lancer dans l'exécution des requêtes SQL :

- **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**

```
SELECT
   o.order_customer_id,
   COUNT(*) AS total_orders
FROM
   orders o
WHERE
   o.order_status = 'COMPLETE'
   AND YEAR(FROM_UNIXTIME(UNIX_TIMESTAMP(o.order_date, 'yyyy-MM-dd HH:mm:ss'))) = 2014
GROUP BY
   o.order_customer_id
```

ORDER BY

total_orders DESC;

```
Number of reduce tasks determined at compile time: 1
In order to change the average load for a reducer (in bytes):
  set hive.exec.reducers.bytes.per.reducer=<number>
In order to limit the maximum number of reducers:
  set hive.exec.reducers.max=<number>
In order to set a constant number of reducers:
  set mapreduce.job.reduces=<number>
Starting Job = job_1722106357031_0029, Tracking URL = http://10.0.2.15:8088/proxy/application_1722106357031_0029/
Kill Command = /opt/hadoop/bin/mapred job  -kill job_1722106357031_0029
Hadoop job information for Stage-2: number of mappers: 1; number of reducers: 1
2024-07-27 21:14:22,770 Stage-2 map = 0%,  reduce = 0%
2024-07-27 21:14:36,320 Stage-2 map = 100%,  reduce = 0%, Cumulative CPU 4.98 sec
2024-07-27 21:14:49,288 Stage-2 map = 100%,  reduce = 100%, Cumulative CPU 11.89 sec
MapReduce Total cumulative CPU time: 11 seconds 890 msec
Ended Job = job_1722106357031_0029
MapReduce Jobs Launched:
Stage-Stage-1: Map: 1  Reduce: 1   Cumulative CPU: 20.34 sec   HDFS Read: 338730 HDFS Write: 96 SUCCESS
Stage-Stage-2: Map: 1  Reduce: 1   Cumulative CPU: 11.89 sec   HDFS Read: 7696 HDFS Write: 87 SUCCESS
Total MapReduce CPU Time Spent: 32 seconds 230 msec
OK
Time taken: 187.008 seconds
hive>
```

Figure 13 : Question 1

- **2-Afficher le nom et le prénom des clients qui n'ont passé aucune commande, triés par customer_lname puis customer_fname.**

SELECT

customer_lname,

customer_fname

FROM

customers

LEFT JOIN

orders

ON

customers.customer_id = orders.order_customer_id

WHERE

orders.order_id IS NULL

ORDER BY

customer_lname,

customer_fname;

Sélectionne les colonnes **customer_lname et customer_fname** de la table **customers.**

Effectue une jointure gauche **(LEFT JOIN)** entre la table **customers** et la table **orders** sur l'identifiant du client.

Filtre les résultats pour ne conserver que les clients qui n'ont aucune commande **(WHERE orders.order_id IS NULL).**

Trie les résultats par nom de famille **(customer_lname)** puis par prénom **(customer_fname).**

```
OK
Bolton  Mary
Ellison Albert
Green   Carolyn
Greene  Mary
Harrell Mary
Lewis   Mary
Mueller Mary
Patel   Matthew
Shaw    Mary
Smith   Amanda
Smith   Ashley
Smith   Carl
Smith   Emma
Smith   Grace
Smith   James
Smith   Joan
Smith   Kenneth
Smith   Kevin
Smith   Mary
Smith   Mary
Smith   Mary
Smith   Mary
Smith   Randy
Smith   Stephen
Stephens        Donna
Tanner  Jose
Vazquez Dorothy
Walker  Gary
Williams        Mary
Wolf    Alan
Time taken: 119.079 seconds, Fetched: 30 row(s)
hive>
```

Figure 14 : Question 2

- **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**

```
WITH monthly_data AS (
  SELECT
    o.order_customer_id AS customer_id,
    c.customer_fname,
    c.customer_lname,
    YEAR(FROM_UNIXTIME(UNIX_TIMESTAMP(o.order_date, 'yyyy-MM-dd HH:mm:ss'))) AS year,
    MONTH(FROM_UNIXTIME(UNIX_TIMESTAMP(o.order_date, 'yyyy-MM-dd HH:mm:ss'))) AS month,
    SUM(oi.order_item_quantity * oi.order_item_product_price) AS monthly_revenue
  FROM
    orders o
  JOIN
    order_items oi ON o.order_id = oi.order_item_order_id
  JOIN
    customers c ON o.order_customer_id = c.customer_id
  WHERE
```

```
        o.order_status = 'COMPLETE'
    GROUP BY
        o.order_customer_id,
        c.customer_fname,
        c.customer_lname,
        YEAR(FROM_UNIXTIME(UNIX_TIMESTAMP(o.order_date, 'yyyy-MM-dd HH:mm:ss'))),
        MONTH(FROM_UNIXTIME(UNIX_TIMESTAMP(o.order_date, 'yyyy-MM-dd HH:mm:ss')))
),
ranked_data AS (
    SELECT
        customer_id,
        customer_fname,
        customer_lname,
        year,
        month,
        monthly_revenue,
        ROW_NUMBER() OVER (PARTITION BY year, month ORDER BY monthly_revenue DESC) AS rank
    FROM
```

```
    monthly_data
)
SELECT
  customer_fname,
  customer_lname,
  year,
  month,
  monthly_revenue
FROM
  ranked_data
WHERE
  rank <= 5
ORDER BY
  year ASC,
  month ASC,
  monthly_revenue DESC;
```

monthly_data : Calcule les revenus mensuels par client pour chaque année et mois.
ranked_data : Attribue un rang à chaque client basé sur les revenus mensuels en u tilisant ROW_NUMBER().
Final SELECT : Sélectionne les clients avec un rang de 1 à 5 pour chaque mois e t trie les résultats par année, mois, et revenus mensuels.

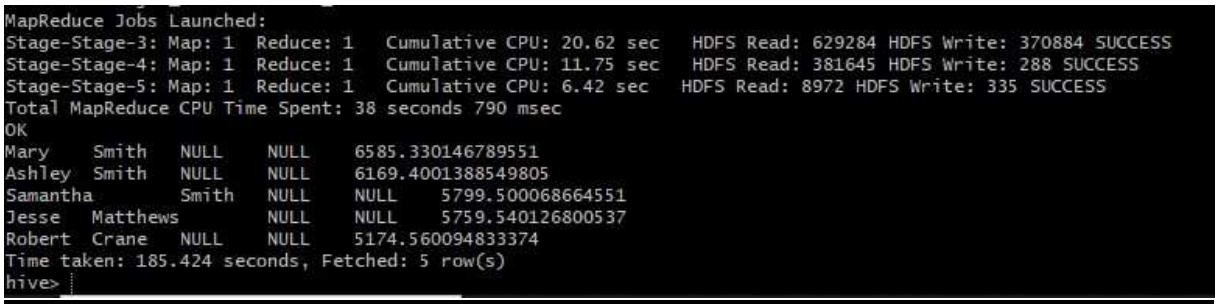


Figure 15 : Question 3

- **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 :**

```
SELECT
   DATE(FROM_UNIXTIME(UNIX_TIMESTAMP(o.order_date, 'yyyy-MM-dd HH:mm:ss'))) AS order_date,
   d.department_name,
   SUM(oi.order_item_quantity * oi.order_item_product_price) 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 ('COMPLETE', 'CLOSED')

GROUP BY

DATE(FROM_UNIXTIME(UNIX_TIMESTAMP(o.order_date, 'yyyy-MM-dd HH:mm:ss'))),

d.department_name

ORDER BY

order_date ASC,

d.department_name ASC;

Filtrage des commandes :

- La clause **WHERE** filtre les commandes avec les statuts **COMPLETE ou CLOSED.**

Joindre les tables :

- **orders est joint avec order_items** pour obtenir les articles commandés**.**
- **order_items est joint avec products** pour obtenir les informations sur les produits.
- **products est joint avec categories** pour obtenir les informations de catégorie.
- **categories est joint avec departments** pour obtenir les informations de département.

Calculer le revenu total :

- **SUM(oi.order_item_quantity * oi.order_item_product_price)** calcule le revenu total pour chaque combinaison de jour et de département.

Afficher les résultats :

- Les résultats sont regroupés par order_date et department_name.
- La sortie est triée par order_date et department_name.

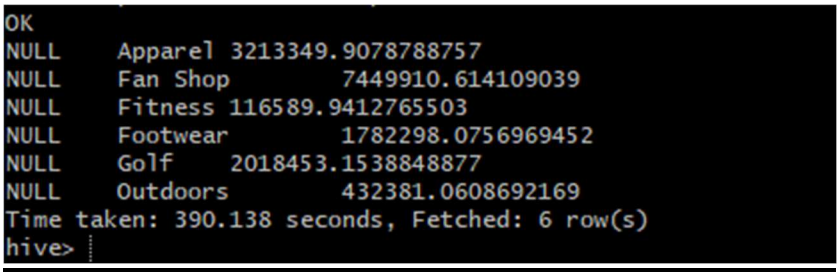


Figure 16 : Question 4

- **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.**

```
SELECT
   department_name,
   category_name,
   total_revenue,
   rank
FROM (
   SELECT
      d.department_name,
      c.category_name,
      SUM(oi.order_item_quantity * oi.order_item_product_price) AS total_revenue,
      @rank := IF(@department_name = d.department_name, @rank + 1, 1) AS
rank,
      @department_name := d.department_name
```

```
    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 ('COMPLETE', 'CLOSED')
    GROUP BY
        d.department_name,
        c.category_name
    ORDER BY
        d.department_name,
        total_revenue DESC
) AS ranked_categories
ORDER BY
    department_name ASC,
    rank ASC;
```

Calculer le revenu total :

- **SUM(oi.order_item_quantity * oi.order_item_product_price)** calcule le revenu total pour chaque catégorie dans chaque département.

Attribuer un rang :

- Utilise les variables utilisateur **(@rank et @department_name)** pour attribuer un rang à chaque catégorie dans chaque département.
- La condition IF(**@department_name = d.department_name, @rank + 1, 1**) réinitialise le rang lorsque le nom du département change.

Afficher les résultats :

- Sélectionne les colonnes department_name, category_name, total_revenue, et rank.
- Trie les résultats par department_name et rank.

```
OK
Apparel Cleats  1934378.2438316345      1
Apparel Men's Footwear  1278971.6640472412      2
Fan Shop        Fishing 3022248.9630126953      1
Fan Shop        Camping & Hiking        1802879.866027832       2
Fan Shop        Water Sports    1342783.0065917969      3
Fan Shop        Indoor/Outdoor Games    1257646.7284812927      4
Fan Shop        Hunting & Shooting      24352.049995422363      5
Fitness Baseball & Softball     40757.16142272949       1
Fitness Hockey  19557.929931640625      2
Fitness Tennis & Racquet        18490.890689849854      3
Fitness Lacrosse        16045.619770050049      4
Fitness Soccer  12438.579696655273      5
Footwear        Cardio Equipment        1639187.2152328491      1
Footwear        Electronics     49014.70033836365       2
Footwear        Boxing & MMA    35727.91002655029       3
Footwear        Strength Training       34302.709716796875      4
Footwear        Fitness Accessories     16266.32054901123       5
Golf    Women's Apparel 1387000.0       1
Golf    Shop By Sport   568171.8332672119       2
Golf    Girls' Apparel  63281.32061767578       3
Outdoors        Electronics     112141.70235443115      1
Outdoors        Accessories     59151.329458236694      2
Outdoors        Golf Shoes      49168.0 3
Outdoors        Golf Gloves     47238.17037010193       4
Outdoors        Kids' Golf Clubs        42683.819259643555      5
Time taken: 468.832 seconds, Fetched: 25 row(s)
hive>
```

Figure 17 : Question 5

- **6-Afficher le pourcentage de chaque catégorie par revenue dans chaque département. Afficher les résultats par department_name et pourcentage par ordre décroissant.**

```
SELECT
   cr.department_name,
   cr.category_name,
   (cr.category_revenue / dr.total_department_revenue) * 100 AS percentage
FROM (
   -- Revenu total par catégorie dans chaque département
   SELECT
      d.department_name,
      c.category_name,
      SUM(oi.order_item_quantity * oi.order_item_product_price) AS
category_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 ('COMPLETE', 'CLOSED')
    GROUP BY
        d.department_name,
        c.category_name
) AS cr
JOIN (
    -- Revenu total pour chaque département
    SELECT
        d.department_name,
        SUM(oi.order_item_quantity * oi.order_item_product_price) AS
total_department_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 ('COMPLETE', 'CLOSED')
    GROUP BY
        d.department_name
) AS dr ON cr.department_name = dr.department_name
ORDER BY
    cr.department_name,
    percentage DESC;
```

Sous-requête cr :

- Calcule le revenu total pour chaque catégorie dans chaque département.

Sous-requête dr :

- Calcule le revenu total pour chaque département sans utiliser de fonction d'agrégation imbriquée.

Calcul du pourcentage :

- Le revenu de chaque catégorie est divisé par le revenu total du département et multiplié par 100 pour obtenir le pourcentage.

Ordre de tri :

Tri des résultats par department_name et percentage décroissant

```
OK
Apparel Cleats  60.19818255984804
Apparel Men's Footwear  39.80181744015196
Fan Shop        Fishing 40.567586908881815
Fan Shop        Camping & Hiking        24.20002009975048
Fan Shop        Water Sports    18.024149230042607
Fan Shop        Indoor/Outdoor Games    16.881366685118262
Fan Shop        Hunting & Shooting      0.3268770762068359
Fitness Baseball & Softball     34.95769958924147
Fitness Hockey  16.7749719379731
Fitness Tennis & Racquet        15.85976499120934
Fitness Lacrosse        13.762439190178494
Fitness Soccer  10.668655941039608
Fitness Basketball      7.97646835035799
Footwear        Cardio Equipment        91.97043062462296
Footwear        Electronics     2.750084343731172
Footwear        Boxing & MMA    2.0045979128704015
Footwear        Strength Training       1.9246337178130675
Footwear        Fitness Accessories     0.9126599400412017
Footwear        As Seen on  TV! 0.43759346092118945
Golf    Women's Apparel 68.71598666188814
Golf    Shop By Sport   28.14887391236501
Golf    Girls' Apparel  3.135139425746847
Outdoors        Electronics     25.93584976386162
Outdoors        Accessories     13.680370120588679
Outdoors        Golf Shoes      11.371450891294227
Outdoors        Golf Gloves     10.925124767291818
Outdoors        Kids' Golf Clubs        9.87180594215578
Outdoors        Golf Balls      7.970129283764583
Outdoors        Trade-In        6.925654842270711
Outdoors        Women's Golf Clubs      4.421604342558532
Outdoors        Men's Golf Clubs        4.238943781232408
Outdoors        Golf Apparel    3.636879924549147
Outdoors        Golf Bags & Carts       1.0221863404324993
Time taken: 1279.133 seconds, Fetched: 33 row(s)
hive> m
```

Figure 18 : Question 6

- **7-Afficher tous les clients qui ont passé une commande d'un montant supérieur à 200 $.**

SELECT

customer_id,

customer_fname,

customer_lname,

customer_email,

customer_street,

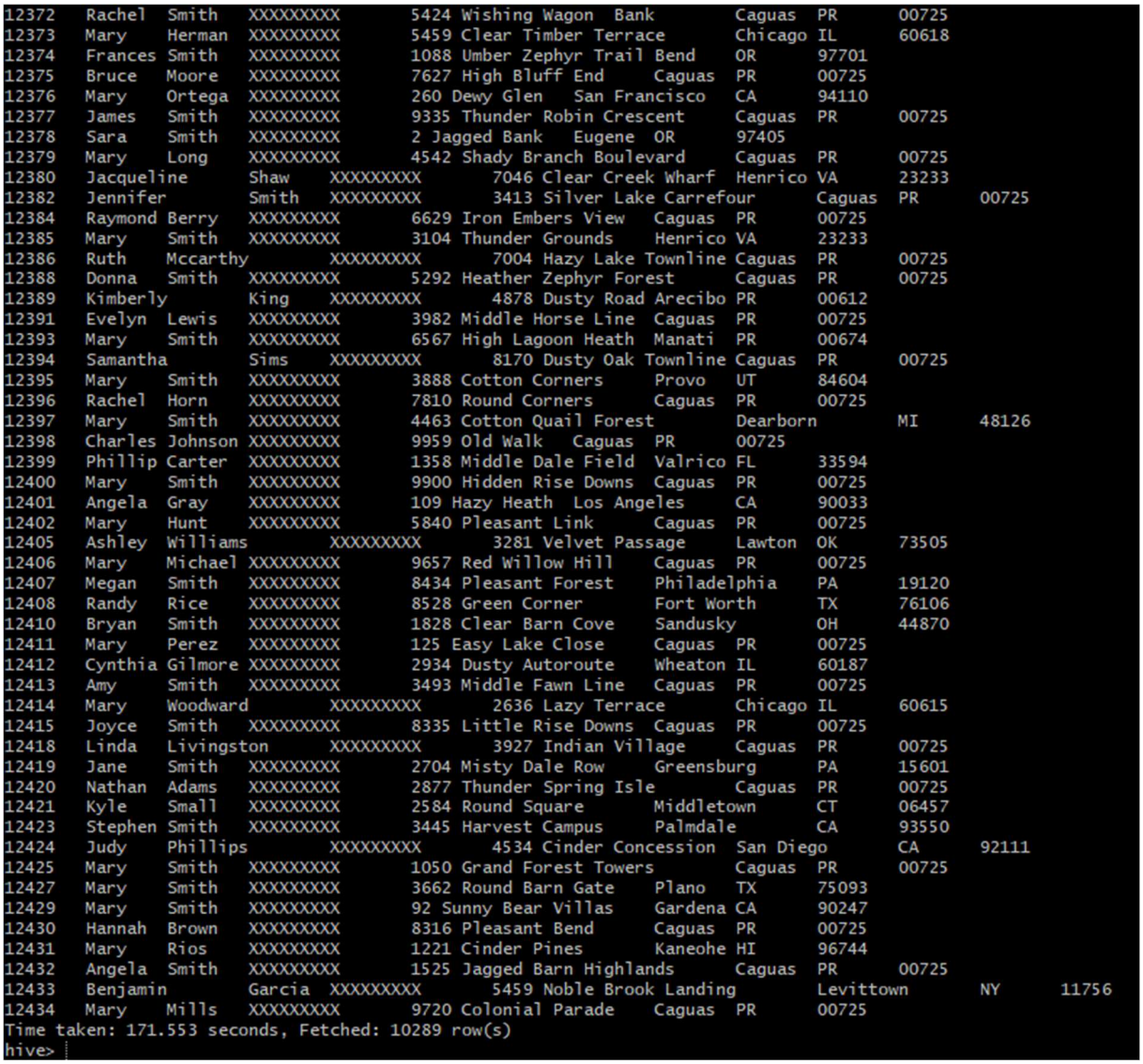
```
    customer_city,
    customer_state,
    customer_zipcode
FROM (
    SELECT
        c.customer_id,
        c.customer_fname,
        c.customer_lname,
        c.customer_email,
        c.customer_street,
        c.customer_city,
        c.customer_state,
        c.customer_zipcode,
        SUM(oi.order_item_quantity * oi.order_item_product_price) AS total_spent
    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
        customers c ON o.order_customer_id = c.customer_id
    WHERE
        o.order_status IN ('COMPLETE', 'CLOSED')
    GROUP BY
        c.customer_id,
        c.customer_fname,
        c.customer_lname,
        c.customer_email,
        c.customer_street,
        c.customer_city,
        c.customer_state,
        c.customer_zipcode
) tmp
WHERE
    tmp.total_spent > 200
```

La sous-requête **(FROM (...) tmp)** calcule le total dépensé pour chaque client, en regroupant par toutes les colonnes de clients nécessaires.

La requête externe filtre les clients dont le total dépensé dépasse 200.

Cette méthode garantit que chaque client apparaît une seule fois dans les résultats, tout en respectant les contraintes de Hive concernant **SELECT DISTINCT et GROUP BY**



```
12372   Rachel  Smith   XXXXXXXXX       5424 Wishing Wagon  Bank        Caguas  PR      00725
12373   Mary    Herman  XXXXXXXXX       5459 Clear Timber Terrace       Chicago IL      60618
12374   Frances Smith   XXXXXXXXX       1088 Umber Zephyr Trail Bend    OR      97701
12375   Bruce   Moore   XXXXXXXXX       7627 High Bluff End     Caguas  PR      00725
12376   Mary    Ortega  XXXXXXXXX       260 Dewy Glen   San Francisco   CA      94110
12377   James   Smith   XXXXXXXXX       9335 Thunder Robin Crescent     Caguas  PR      00725
12378   Sara    Smith   XXXXXXXXX       2 Jagged Bank   Eugene  OR      97405
12379   Mary    Long    XXXXXXXXX       4542 Shady Branch Boulevard     Caguas  PR      00725
12380   Jacqueline      Shaw    XXXXXXXXX       7046 Clear Creek Wharf  Henrico VA      23233
12382   Jennifer        Smith   XXXXXXXXX       3413 Silver Lake Carrefour      Caguas  PR      00725
12384   Raymond Berry   XXXXXXXXX       6629 Iron Embers View   Caguas  PR      00725
12385   Mary    Smith   XXXXXXXXX       3104 Thunder Grounds    Henrico VA      23233
12386   Ruth    Mccarthy        XXXXXXXXX       7004 Hazy Lake Townline Caguas  PR      00725
12388   Donna   Smith   XXXXXXXXX       5292 Heather Zephyr Forest      Caguas  PR      00725
12389   Kimberly        King    XXXXXXXXX       4878 Dusty Road Arecibo PR      00612
12391   Evelyn  Lewis   XXXXXXXXX       3982 Middle Horse Line  Caguas  PR      00725
12393   Mary    Smith   XXXXXXXXX       6567 High Lagoon Heath  Manati  PR      00674
12394   Samantha        Sims    XXXXXXXXX       8170 Dusty Oak Townline Caguas  PR      00725
12395   Mary    Smith   XXXXXXXXX       3888 Cotton Corners     Provo   UT      84604
12396   Rachel  Horn    XXXXXXXXX       7810 Round Corners      Caguas  PR      00725
12397   Mary    Smith   XXXXXXXXX       4463 Cotton Quail Forest        Dearborn        MI      48126
12398   Charles Johnson XXXXXXXXX       9959 Old Walk   Caguas  PR      00725
12399   Phillip Carter  XXXXXXXXX       1358 Middle Dale Field  Valrico FL      33594
12400   Mary    Smith   XXXXXXXXX       9900 Hidden Rise Downs  Caguas  PR      00725
12401   Angela  Gray    XXXXXXXXX       109 Hazy Heath  Los Angeles     CA      90033
12402   Mary    Hunt    XXXXXXXXX       5840 Pleasant Link      Caguas  PR      00725
12405   Ashley  Williams        XXXXXXXXX       3281 Velvet Passage     Lawton  OK      73505
12406   Mary    Michael XXXXXXXXX       9657 Red Willow Hill    Caguas  PR      00725
12407   Megan   Smith   XXXXXXXXX       8434 Pleasant Forest    Philadelphia    PA      19120
12408   Randy   Rice    XXXXXXXXX       8528 Green Corner       Fort Worth      TX      76106
12410   Bryan   Smith   XXXXXXXXX       1828 Clear Barn Cove    Sandusky        OH      44870
12411   Mary    Perez   XXXXXXXXX       125 Easy Lake Close     Caguas  PR      00725
12412   Cynthia Gilmore XXXXXXXXX       2934 Dusty Autoroute    Wheaton IL      60187
12413   Amy     Smith   XXXXXXXXX       3493 Middle Fawn Line   Caguas  PR      00725
12414   Mary    Woodward        XXXXXXXXX       2636 Lazy Terrace       Chicago IL      60615
12415   Joyce   Smith   XXXXXXXXX       8335 Little Rise Downs  Caguas  PR      00725
12418   Linda   Livingston      XXXXXXXXX       3927 Indian Village     Caguas  PR      00725
12419   Jane    Smith   XXXXXXXXX       2704 Misty Dale Row     Greensburg      PA      15601
12420   Nathan  Adams   XXXXXXXXX       2877 Thunder Spring Isle        Caguas  PR      00725
12421   Kyle    Small   XXXXXXXXX       2584 Round Square       Middletown      CT      06457
12423   Stephen Smith   XXXXXXXXX       3445 Harvest Campus     Palmdale        CA      93550
12424   Judy    Phillips        XXXXXXXXX       4534 Cinder Concession  San Diego       CA      92111
12425   Mary    Smith   XXXXXXXXX       1050 Grand Forest Towers        Caguas  PR      00725
12427   Mary    Smith   XXXXXXXXX       3662 Round Barn Gate    Plano   TX      75093
12429   Mary    Smith   XXXXXXXXX       92 Sunny Bear Villas    Gardena CA      90247
12430   Hannah  Brown   XXXXXXXXX       8316 Pleasant Bend      Caguas  PR      00725
12431   Mary    Rios    XXXXXXXXX       1221 Cinder Pines       Kaneohe HI      96744
12432   Angela  Smith   XXXXXXXXX       1525 Jagged Barn Highlands      Caguas  PR      00725
12433   Benjamin        Garcia  XXXXXXXXX       5459 Noble Brook Landing        Levittown       NY      11756
12434   Mary    Mills   XXXXXXXXX       9720 Colonial Parade    Caguas  PR      00725
Time taken: 171.553 seconds, Fetched: 10289 row(s)
hive>
```

Figure 19 : Question 7

- **8-Afficher les clients de la "customers" dont les noms customer_fname commence par "Rich"**

SELECT

customer_id,

customer_fname,

customer_lname,

customer_email,

customer_street,

```
    customer_city,
    customer_state,
    customer_zipcode
FROM
    customers
WHERE
    customer_fname LIKE 'Rich%';
```

Sélection des colonnes :

- La requête sélectionne toutes les colonnes disponibles dans la table customers.

Filtrage avec LIKE :

customer_fname LIKE 'Rich%' : Cette condition filtre les enregistrements dont la colonne customer_fname commence par "Rich". Le caractère % permet de dire tus ce qui se situe apés est bon.

```
11391   Richard Holland XXXXXXXXX       3867 Velvet Hills Diversion     Philadelphia    PA      19139
11460   Richard Payne   XXXXXXXXX       1156 Silver Sky Line    New York        NY      10128
11530   Richard Gibson  XXXXXXXXX       8275 Broad Nectar Pointe        Caguas  PR      00725
11576   Richard Andrade XXXXXXXXX       1987 Burning Rabbit Crescent    Caguas  PR      00725
11688   Richard Smith   XXXXXXXXX       7641 Emerald Diversion  Germantown      MD      20874
11753   Richard Reid    XXXXXXXXX       9128 Little Cloud Bay   Opa Locka       FL      33055
11867   Richard Ward    XXXXXXXXX       6396 Iron Pony Passage  Chino   CA      91710
11940   Richard Reese   XXXXXXXXX       9761 Broad Crossing     Caguas  PR      00725
11964   Richard Huffman XXXXXXXXX       2293 Thunder Vale       Caguas  PR      00725
11984   Richard Smith   XXXXXXXXX       2275 Indian Embers Range        Caguas  PR      00725
12048   Richard Smith   XXXXXXXXX       8428 Noble Elk Loop     Hamtramck       MI      48212
12100   Richard Bolton  XXXXXXXXX       4675 Sleepy Rise        Chicago IL      60609
12403   Richard Ferguson        XXXXXXXXX       5382 Hazy Pathway       Cypress CA      90630
3210    Richard Welch   XXXXXXXXX       5382 Hazy Pathway       Escondido       CA      92026
3250    Richard Smith   XXXXXXXXX       9706 Honey Alley        Bell Gardens    CA      90201
3301    Richard Davila  XXXXXXXXX       9729 Middle Shadow Run  Caguas  PR      00725
3539    Richard Smith   XXXXXXXXX       6157 Quaking Extension  Washington      DC      20011
3551    Richard Kramer  XXXXXXXXX       762 Clear Harbour       Caguas  PR      00725
3781    Richard Lara    XXXXXXXXX       79 Misty Goose Townline Hamtramck       MI      48212
4388    Richard Moore   XXXXXXXXX       7177 Old Nectar Ridge   Holland MI      49423
4399    Richard Smith   XXXXXXXXX       9627 Little Terrace     Virginia Beach  VA      23455
4891    Richard Smith   XXXXXXXXX       5293 Emerald Concession Philadelphia    PA      19144
4906    Richard Smith   XXXXXXXXX       8977 Sunny Dale Run     San Jose        CA      95124
4934    Richard Warner  XXXXXXXXX       6646 Iron Fox Mews      Caguas  PR      00725
4981    Richard Shepherd        XXXXXXXXX       8684 Golden Point       Bronx   NY      10466
5556    Richard Burns   XXXXXXXXX       2406 Merry Horse Isle   Caguas  PR      00725
5572    Richard Smith   XXXXXXXXX       5284 Emerald Abbey      San Jose        CA      95127
5575    Richard Smith   XXXXXXXXX       7855 Golden Lake Limits Elk Grove       CA      95758
5965    Richard Parker  XXXXXXXXX       1018 Colonial Nectar Subdivision        Caguas  PR      00725
6054    Richard Reilly  XXXXXXXXX       8364 High Road  Los Angeles     CA      90047
6210    Richard Smith   XXXXXXXXX       8852 Iron Port  Caguas  PR      00725
6299    Richard Robinson        XXXXXXXXX       6468 Quiet Street       Pomona  CA      91767
6322    Richard Smith   XXXXXXXXX       398 Emerald Grove       Hanford CA      93230
6336    Richard Maddox  XXXXXXXXX       5351 Blue Treasure Square       Caguas  PR      00725
6431    Richard Smith   XXXXXXXXX       6280 Cozy Embers Island Placentia       CA      92870
6779    Richard Durham  XXXXXXXXX       6292 Hidden Treasure Thicket    Pacoima CA      91331
6946    Richard Sanchez XXXXXXXXX       3610 Quiet Highlands    Caguas  PR      00725
7155    Richard Thompson        XXXXXXXXX       1494 Easy Crossing      Lawrence        MA      01841
7296    Richard Randall XXXXXXXXX       3741 Noble Nook Caguas  PR      00725
7385    Richard Arellano        XXXXXXXXX       7533 Clear Goose Lane   Phoenix AZ      85040
7669    Richard Patton  XXXXXXXXX       8004 Colonial River Square      Caguas  PR      00725
8031    Richard Smith   XXXXXXXXX       6580 Thunder Village    Richmond        VA      23223
8044    Richard Smith   XXXXXXXXX       5387 Iron Nook  Gaithersburg    MD      20878
8064    Richard Rich    XXXXXXXXX       8171 Indian Ramp        Oxnard  CA      93030
8339    Richard Smith   XXXXXXXXX       2210 Cotton Spring Plaza        Alhambra        CA      91801
8387    Richard Fuentes XXXXXXXXX       5237 Rustic Village     Waipahu HI      96797
8535    Richard Quinn   XXXXXXXXX       7994 Easy Estates       Caguas  PR      00725
8853    Richard Ali     XXXXXXXXX       760 Lazy Pines  Littleton       CO      80126
9102    Richard Marshall        XXXXXXXXX       911 Middle Fawn Townline        Aurora  IL      60504
9189    Richard Smith   XXXXXXXXX       8310 Broad Centre       Caguas  PR      00725
Time taken: 0.462 seconds, Fetched: 76 row(s)
hive>
```

Figure 20 : Question 8

- **9-Fournir le nombre total de clients dans chaque état (state) dont le prénom commence par « M »**

```
SELECT
    customer_state,
    COUNT(*) AS total_customers
FROM
    customers
WHERE
```

```
    customer_fname LIKE 'M%'
GROUP BY
    customer_state;
```

COUNT(*) compte le nombre de clients.

WHERE customer_fname LIKE 'M%' filtre les clients dont le prénom commence par "M".

GROUP BY customer_state regroupe les résultats par état.

```
Stage-Stage-1: Map: 1  Reduce: 1   Cumulative CPU: 26.84 sec   HDFS Read: 62223 HDFS Write: 879 SUCCESS
Total MapReduce CPU Time Spent: 26 seconds 840 msec
OK
AL      1
AR      3
AZ      98
CA      850
CO      51
CT      34
DC      17
DE      9
FL      162
GA      86
HI      34
IA      2
ID      4
IL      222
IN      16
KS      11
KY      13
LA      24
MA      43
MD      73
MI      114
MN      14
MO      35
MT      5
NC      74
ND      6
NJ      87
NM      22
NV      43
NY      331
OH      130
OK      8
OR      45
PA      120
PR      2063
RI      8
SC      16
TN      46
TX      267
UT      25
VA      59
WA      32
WI      25
WV      7
Time taken: 107.284 seconds, Fetched: 44 row(s)
hive>
```

Figure 21: Question 9

- **10-Trouver le produit le plus cher dans chaque catégorie**

```
SELECT
    c.category_name,
    p.product_name,
    p.product_price
FROM
    products p
```

```
JOIN
   categories c ON p.product_category_id = c.category_id
WHERE
   p.product_price = (
      SELECT
         MAX(p2.product_price)
      FROM
         products p2
      WHERE
         p2.product_category_id = p.product_category_id
 );
```

```
International Soccer    adidas Men's 2014 MLS All-Star Game Replica B    85.0
World Cup Shop  adidas Brazuca 2014 Official Match Ball 159.99
World Cup Shop  adidas Brazuca Final Rio Official Match Ball     159.99
MLB Players     PUMA Men's evoPOWER 1 Tricks FG Soccer Cleat     189.99
NFL Players     Majestic Men's Authentic Los Angeles Dodgers     241.0
Women's Golf Clubs      TaylorMade SLDR Irons - (Steel) 4-PW, AW         899.99
Golf Apparel    Callaway Women's Solaire Gems 20-Piece Comple    999.99
Golf Apparel    Callaway Women's Solaire Gems 20-Piece Comple    999.99
Golf Shoes      Nike TW 14 Mesh Golf Shoes      169.99
Golf Shoes      Nike TW 14 Mesh Golf Shoes      169.99
Golf Bags & Carts       Ogio Race Golf Shoes    169.99
Golf Bags & Carts       Ogio Race Golf Shoes    169.99
Golf Gloves     Club Glove Last Bag     299.99
Golf Gloves     Titleist Club Glove Travel Cover        299.99
Golf Balls      Hirzl Women's Trust Feel Golf Glove     24.99
Golf Balls      Hirzl Women's Trust Control Golf Glove  24.99
Golf Balls      Glove It Women's Violet Bling Golf Glove        24.99
Electronics     Titleist Pro V1 Personalized Golf Balls 51.99
Electronics     Titleist Pro V1x Personalized Golf Balls        51.99
Electronics     Titleist Pro V1 High Numbers Personalized Gol   51.99
Electronics     Titleist Pro V1x High Numbers Personalized Go   51.99
Electronics     Titleist Pro V1 Double Number Golf Balls        51.99
Kids' Golf Clubs        Bushnell Pro X7 Jolt Slope Rangefinder  599.99
Team Shop       Cobra Junior Kids' Complete Set (Height 53''    199.99
Accessories     Team Golf St. Louis Cardinals Putter Grip       24.99
Accessories     Team Golf San Francisco Giants Putter Grip      24.99
Accessories     Team Golf New York Yankees Putter Grip  24.99
Accessories     Team Golf Detroit Tigers Putter Grip    24.99
Accessories     Team Golf Chicago Cubs Putter Grip      24.99
Accessories     Team Golf Boston Red Sox Putter Grip    24.99
Accessories     Team Golf Washington Redskins Putter Grip       24.99
Accessories     Team Golf San Francisco 49ers Putter Grip       24.99
Accessories     Team Golf Pittsburgh Steelers Putter Grip       24.99
Accessories     Team Golf Dallas Cowboys Putter Grip    24.99
Accessories     Team Golf Oakland Raiders Putter Grip   24.99
Accessories     Team Golf New York Giants Putter Grip   24.99
Accessories     Team Golf New England Patriots Putter Grip      24.99
Accessories     Team Golf Minnesota Vikings Putter Grip 24.99
Accessories     Team Golf Green Bay Packers Putter Grip 24.99
Accessories     Team Golf Chicago Bears Putter Grip     24.99
Accessories     Team Golf Baltimore Ravens Putter Grip  24.99
Accessories     Team Golf West Virginia Mountaineers Putter G   24.99
Accessories     Team Golf Missouri Tigers Putter Grip   24.99
Accessories     Team Golf Wisconsin Badgers Putter Grip 24.99
Accessories     Team Golf Texas Longhorns Putter Grip   24.99
Accessories     Team Golf Tennessee Volunteers Putter Grip      24.99
Accessories     Team Golf South Carolina Gamecocks Putter Gri   24.99
Accessories     Team Golf Penn State Nittany Lions Putter Gri   24.99
Bike & Skate Shop       Nike VR_S Covert Driver 179.99
Camping & Hiking        Diamondback Adult Trace Hybrid Bike 2014        449.99
Hunting & Shooting      YETI Tundra 65 Chest Cooler     399.99
Time taken: 302.3 seconds, Fetched: 117 row(s)
hive>
```

Figure 22 : Question 10

MAX(p2.product_price) trouve le prix maximum pour chaque catégorie.

WHERE p.product_price = assure que seul le produit le plus cher est sélectionné

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

```
SELECT
    p.product_id,
    p.product_name,
    SUM(oi.order_item_quantity * oi.order_item_product_price) AS total_revenue
FROM
    order_items oi
JOIN
    products p ON oi.order_item_product_id = p.product_id
JOIN
    orders o ON oi.order_item_order_id = o.order_id
WHERE
    o.order_status IN ('COMPLETE', 'CLOSED')
GROUP BY
    p.product_id,
    p.product_name
ORDER BY
    total_revenue DESC
LIMIT 10;
```

SUM (oi.order_item_quantity * oi.order_item_product_price) calcule le revenu total pour chaque produit.

ORDER BY total_revenue DESC trie les produits par revenu total, en ordre décroissant.

LIMIT 10 limite les résultats aux 10 premiers produits.

```
MapReduce Jobs Launched:
Stage-Stage-3: Map: 1  Reduce: 1   Cumulative CPU: 35.4 sec   HDFS Read: 863085 HDFS Write: 6544 SUCCESS
Stage-Stage-4: Map: 1  Reduce: 1   Cumulative CPU: 13.03 sec   HDFS Read: 14813 HDFS Write: 798 SUCCESS
Total MapReduce CPU Time Spent: 48 seconds 430 msec
OK
1004    Field & Stream Sportsman 16 Gun Fire Safe       3022248.9630126953
365     Perfect Fitness Perfect Rip Deck        1929578.4039878845
957     Diamondback Women's Serene Classic Comfort Bi   1802879.866027832
191     Nike Men's Free 5.0+ Running Shoe       1627337.2152328491
502     Nike Men's Dri-FIT Victory Golf Polo    1387000.0
1073    Pelican Sunstream 100 Kayak     1337533.1567382812
403     Nike Men's CJ Elite 2 TD Football Cleat 1278971.6640472412
1014    O'Brien Men's Neoprene Life Vest        1257646.7284812927
627     Under Armour Girls' Toddler Spine Surge Runni   551862.0231628418
728     LIJA Women's Eyelet Sleeveless Golf Polo        27820.0
Time taken: 243.377 seconds, Fetched: 10 row(s)
hive>
```

Figure 23 : Question 11

Conclusion :

Le projet "Ingestion de données dans Big Data" est une illustration pratique et approfondie de l'utilisation des technologies Apache Sqoop et Apache Hive pour gérer et analyser de grandes quantités de données. En deux parties distinctes mais complémentaires, ce projet démontre comment les entreprises peuvent tirer parti de l'écosystème Hadoop pour optimiser leurs processus de gestion des données.

La première partie du projet s'est concentrée sur l'ingestion des données depuis des bases de données relationnelles vers le système de fichiers distribué Hadoop (HDFS) en utilisant Apache Sqoop. L'objectif principal de cette phase était de démontrer comment Apache Sqoop peut être utilisé pour transférer efficacement des données depuis des systèmes de gestion de bases de données (SGBD) traditionnels, comme MySQL, vers un environnement de Big Data.

Les étapes détaillées incluaient la configuration de la connexion à la base de données, la liste des bases de données et des tables disponibles, et l'importation des données dans HDFS. En spécifiant des formats de fichiers optimisés comme Parquet, nous avons pu non seulement importer les données mais aussi préparer le terrain pour des analyses plus rapides et plus efficaces dans la deuxième phase du projet.

La deuxième partie du projet s'est concentrée sur le traitement et l'analyse des données importées en utilisant Apache Hive. Hive offre une interface de requête SQL-like qui permet aux utilisateurs de traiter et d'analyser de grandes quantités de données stockées dans HDFS de manière efficace et intuitive. Cette partie du projet visait à montrer comment les données ingérées peuvent être transformées en informations utiles grâce à des requêtes et des analyses sophistiquées.

Nous avons commencé par créer des tables externes dans Hive pour les différentes entités de données. En utilisant des formats de stockage optimisés comme Parquet et en spécifiant des emplacements dans HDFS, nous avons préparé les données pour un accès rapide et efficace.

Ensuite, nous avons chargé les données dans ces tables Hive et exécuté des requêtes HiveQL pour extraire des informations et réaliser des analyses

En conclusion, le projet "Ingestion de données dans Big Data" démontre de manière convaincante comment les technologies Apache Sqoop et Apache Hive peuvent être utilisées conjointement pour créer des solutions de Big Data efficaces et évolutives. En automatisant

l'ingestion des données et en fournissant une plate-forme puissante pour le traitement et l'analyse, ces outils permettent aux entreprises de tirer pleinement parti de leurs données pour obtenir des insights précieux et prendre des décisions basées sur des données solides.

En fin de compte, l'intégration de Sqoop et Hive dans une architecture Big Data offre une solution complète et puissante pour la gestion des données à grande échelle, ouvrant de nouvelles possibilités pour l'analyse des données et la prise de décision stratégique dans un environnement en constante évolution.