01 - Distributed Analysis of Energy Data with Hive and PostgreSQL

By Adrián Romero Flores. Source code available at this link.

Overview

1. Introduction
2. Setup and Configuration
3. Data Ingestion and Storage
4. Data Exploration and Analysis
5. Conclusions

Introduction

This project demonstrates how to run a Hive server and a PostgreSQL database as a metastore in order to perform distributed analysis of energy data. This includes setting up the environment, ingesting data, exploring and analyzing the data, and drawing conclusions from the analysis.

The document structure is as follows, in section Setup and Configuration we will explore the changes made to the environment to support PostgreSQL as the metastore. In the Data Ingestion and Storage section we will how we populated the datasets into Hive. In the Data Exploration and Analysis section we will answer the given questions about the data. In the Conclusions section we will draw conclusions from the analysis and reflexions asked in the deliverable. In the Screenshots section we will provide screenshots of the results proving that all the SQL queries were executed successfully.

Setup and Configuration

We used the sample code provided in S2 - Hive y Trino. The following changes were made:

- Changed MySQL docker image in `docker-compose.yml` to a PostgreSQL.
- Changed the DB_DRIVER from `mysql` to `postgres` in Hive images.
- Changed the connection driver name and connection url to the corresponding ones.
- In `hive/conf/hive-site.xml` removed MySQL metastore configuration to:

```xml
<property>
  <name>javax.jdo.option.ConnectionURL</name>
  <value>jdbc:postgresql://postgres:5432/metastore_db</value>
</property>
<property>
```

```xml
  <name>javax.jdo.option.ConnectionDriverName</name>
  <value>org.postgresql.Driver</value>
</property>
<property>
  <name>javax.jdo.option.ConnectionUserName</name>
  <value>hive</value>
</property>
<property>
  <name>javax.jdo.option.ConnectionPassword</name>
  <value>password</value>
</property>
```

- Changed the `init.sql` file to:

```sql
CREATE DATABASE metastore_db;
CREATE USER hive WITH PASSWORD 'password';
GRANT ALL PRIVILEGES ON DATABASE metastore_db TO hive;
```

To start the environment, run the following command:

```bash
docker-compose up -d
```

Data Ingestion and Storage

To ingest the datasets into the PostgreSQL database, we first need to upload the files to the Hive server. We had deleted the first row of each csv file cause they contained headers.

```bash
# copy the datasets to the hive server
docker cp datasets/informations_households.csv hiveserver2:/opt/hive/data/warehouse/
docker cp datasets/daily_dataset.csv hiveserver2:/opt/hive/data/warehouse/
# verify the files are uploaded
docker exec -it hiveserver2 ls /opt/hive/data/warehouse/
```

We can access the Hive CLI by running the following command:

```bash
docker exec -it hiveserver2 beeline -u jdbc:hive2://hiveserver2:10000
```

Now, we can create the tables and load the data into them.

```sql
CREATE TABLE clientes (
    LCLid STRING,
    stdorToU STRING,
    Acorn STRING,
    Acorn_grouped STRING,
    file STRING
)
ROW FORMAT DELIMITED
FIELDS TERMINATED BY ','
STORED AS TEXTFILE
```

```sql
TBLPROPERTIES ("skip.header.line.count"="1"); -- this did not work, i dont know why

CREATE TABLE consumos (
    LCLid STRING,
    day DATE,
    energy_median FLOAT,
    energy_mean FLOAT,
    energy_max FLOAT,
    energy_count INT,
    energy_std FLOAT,
    energy_sum FLOAT,
    energy_min FLOAT
)
ROW FORMAT DELIMITED
FIELDS TERMINATED BY ','
STORED AS TEXTFILE
TBLPROPERTIES ("skip.header.line.count"="1"); -- this did not work, i dont know why

LOAD DATA INPATH '/opt/hive/data/warehouse/informations_households.csv' INTO TABLE clientes;
LOAD DATA INPATH '/opt/hive/data/warehouse/daily_dataset.csv' INTO TABLE consumos;
```

Once the data is loaded, we can start exploring and analyzing it.

Data Exploration and Analysis

We will answer the following questions about the data.

1. List the first 10 records of each table: We will use the following SQL queries to display the 10 first records of each table:

```sql
SELECT * FROM clientes LIMIT 10;
SELECT * FROM consumos LIMIT 10;
```

Their results will be displayed below.

clientes.lclid	clientes.stdortou	clientes.acorn	clientes.acorn_grouped	clientes.file
MAC005492	ToU	ACORN-	ACORN-	block_0
MAC001074	ToU	ACORN-	ACORN-	block_0
MAC000002	Std	ACORN-A	Affluent	block_0
MAC003613	Std	ACORN-A	Affluent	block_0
MAC003597	Std	ACORN-A	Affluent	block_0
MAC003579	Std	ACORN-A	Affluent	block_0
MAC003566	Std	ACORN-A	Affluent	block_0
MAC003557	Std	ACORN-A	Affluent	block_0
MAC003553	Std	ACORN-A	Affluent	block_0
MAC003482	Std	ACORN-A	Affluent	block_0

consumos.lclid	consumos.day	consumos.energy_median	consumos.energy_mean	consumos.energy_max	consumos.energy_count	consumos.energy_std	consumos.energy_sum	consumos.energy_min
[illegible]	2011-12-15	0.485000001	[illegible]	[illegible]	[illegible]	[illegible]	[illegible]	[illegible]
[illegible]	2011-12-16	0.141499996	[illegible]	[illegible]	[illegible]	[illegible]	[illegible]	[illegible]
[illegible]	2011-12-17	0.101499997	[illegible]	[illegible]	[illegible]	[illegible]	[illegible]	[illegible]
[illegible]	2011-12-18	0.114000000	[illegible]	[illegible]	[illegible]	[illegible]	[illegible]	[illegible]
[illegible]	2011-12-19	0.190999999	[illegible]	[illegible]	[illegible]	[illegible]	[illegible]	[illegible]
[illegible]	2011-12-20	0.217999999	[illegible]	[illegible]	[illegible]	[illegible]	[illegible]	[illegible]
[illegible]	2011-12-21	0.130500003	[illegible]	[illegible]	[illegible]	[illegible]	[illegible]	[illegible]
[illegible]	2011-12-22	0.089000001	[illegible]	[illegible]	[illegible]	[illegible]	[illegible]	[illegible]
[illegible]	2011-12-23	0.160500004	[illegible]	[illegible]	[illegible]	[illegible]	[illegible]	[illegible]
[illegible]	2011-12-24	0.107000000	[illegible]	[illegible]	[illegible]	[illegible]	[illegible]	[illegible]

(Apologies for the visualization of last table md to pdf conversion is weird)

2. Count the number of households in each socioeconomic category (Acorn): We will use the `COUNT` and `GROUP BY` functions to count the number of households in each socioeconomic category. The query will look like this:

```sql
SELECT acorn, COUNT(*) AS num_houses
FROM clientes
GROUP BY acorn;
```

The results are:

acorn	num_houses
ACORN-	2
ACORN-A	157
ACORN-B	25
ACORN-C	151
ACORN-D	292
ACORN-E	1567
ACORN-F	684
ACORN-G	205
ACORN-H	455
ACORN-I	51
ACORN-J	112
ACORN-K	165
ACORN-L	342
ACORN-M	113
ACORN-N	152
ACORN-O	103
ACORN-P	110
ACORN-Q	831
ACORN-U	49

3. Show the top 10 households with the most consumption records: As previously, we will use the `COUNT`and `GROUP BY` functions to count the number of consumption records for each household. Then we will use the `ORDER BY` function to sort the results in descending order and limit the output to the top 10 households.

```
SELECT lclid, SUM(energy_count) AS num_records
FROM consumos
GROUP BY lclid
ORDER BY num_records DESC
LIMIT 10;
```

The results are:

lclid	num_records
MAC000147	39724
MAC000145	39724
MAC000150	39719
MAC000152	39718
MAC000148	39717
MAC000149	39717
MAC000153	39713
MAC000156	39712
MAC000151	39710

lclid	num_records
MAC000155	39704

4. Total energy consumption per household: We will assume that the total energy consumption per household is the sum of all `energy_sum` for records for that household. For this, we will use the `SUM` function and `GROUP BY` clause.

```
SELECT lclid, SUM(energy_sum) AS total_energy_consumption
FROM consumos
GROUP BY lclid;
```

As this query shows all households and their total energy consumption, for better understanding and visualization we will show only the first five and the last five households.

```
SELECT lclid, SUM(energy_sum) AS total_energy_consumption
FROM consumos
GROUP BY lclid
LIMIT 5;

SELECT lclid, SUM(energy_sum) AS total_energy_consumption
FROM consumos
GROUP BY lclid
ORDER BY lclid DESC
LIMIT 5;
```

The results are:

lclid	total_energy_consumption
MAC000002	6095.671997562051
MAC000003	14080.862013287842
MAC000004	1119.8390001356602
MAC000005	2911.00600380823
MAC000006	2167.4479979783064
...	...
MAC005567	2266.4009990394115
MAC005566	8942.237986594439
MAC005565	5.789999961853027
MAC005564	2314.1690012402833
MAC005563	NULL

5. Mean average consumption by tariff type (Standard or Time-of-Use): For this, we will need to `JOIN` the `consumos` table with the `clientes` table on the `lclid` column. Then we will calculate the average consumption for each tariff type.

```
SELECT cl.stdortou, AVG(c.energy_mean) AS avg_consumption
FROM consumos c
JOIN clientes cl ON c.lclid = cl.lclid
GROUP BY cl.stdortou;
```

The results are:

cl.stdortou	avg_consumption
Std	0.2150364198457096
ToU	0.19859910474893103

6. Households with consumption greater than 5 kWh in at least one measurement: For this, we will use the `MAX` function to find the records where `energy_max` is greater than 5.

```
SELECT lclid, MAX(energy_max) AS max_consumption
FROM consumos
GROUP BY lclid
HAVING max_consumption > 5;
```

The results are that 172 households consumed more than 5 kWh in at least one measurement.

7. Mean average consumption by Acorn category: For this query, we will need to `JOIN` the `consumos` table with the `clientes` table on the `lclid` column. Then we will calculate the average consumption for each Acorn category.

```
SELECT cl.acorn, AVG(c.energy_mean) AS avg_consumption
FROM consumos c
JOIN clientes cl ON c.lclid = cl.lclid
GROUP BY acorn;
```

The results are:

cl.acorn	avg_consumption
ACORN-	0.25118649260602444
ACORN-A	0.3986920369470763
ACORN-B	0.24879375867435508
ACORN-C	0.24976696618681513
ACORN-D	0.2838815010353751
ACORN-E	0.2164889663286261
ACORN-F	0.1921971820315579
ACORN-G	0.21261190647472208
ACORN-H	0.230077194768691
ACORN-I	0.1973362082329818
ACORN-J	0.2378239282401691

cl.acorn	avg_consumption
ACORN-K	0.20926374434632464
ACORN-L	0.20960142785916674
ACORN-M	0.2087953587404035
ACORN-N	0.19273709538147743
ACORN-O	0.17903307652120706
ACORN-P	0.13878119382347404
ACORN-Q	0.15828054603228647
ACORN-U	0.24315419030061705

8. Compare the consumption of households with different tariff types: For this comparison we will compare the statistical metrics for energy consumption. These metrics are: `count`, `sum`, `mean`, `median`, `std`, `max` and `min`.

```
SELECT
    cl.stdortou,
    AVG(c.energy_count) AS avg_count,
    AVG(c.energy_sum) AS avg_sum,
    AVG(c.energy_mean) AS avg_mean,
    AVG(c.energy_median) AS avg_median,
    AVG(c.energy_std) AS avg_std,
    AVG(c.energy_max) AS avg_max,
    AVG(c.energy_min) AS avg_min
FROM consumos c
JOIN clientes cl ON c.lclid = cl.lclid
GROUP BY cl.stdortou;
```

These are the results of the query:

cl.stdortou	avg_count	avg_sum	avg_mean	avg_median	avg_std	avg_max	avg_min
Std	47.8006	10.2816	0.2150	0.1610	0.1753	0.8451	0.0605
ToU	47.8158	9.4988	0.1986	0.1497	0.1622	0.7926	0.0561

The key findings are the following: (1) we see that the average total and mean consumption of households with standard tariffs is higher than those with time-of-use tariffs, (2) the standard deviation of consumption is higher for households with standard tariffs, and (3) the maximum consumption is higher for households with standard tariffs.

These findings suggest that households with standard tariffs consume more energy on average and have more variability in their consumption patterns compared to households with time-of-use tariffs, which may indicate that households with standard tariffs are less likely to adjust their consumption patterns based on time-of-use pricing.

9. Detect households with inconsistent consumption (below 0.1 kWh for more than 3 consecutive days): For this analysis, we will use window functions `LAG` and `LEAD` to look at consecutive days for each household where energy consumption was below 0.1 kWh. Then we will filter if the number of consecutive days is greater than 3, using the `COUNT` function and the `WHERE` clause.

```
WITH consistencycheck AS (
    SELECT
        lclid,
        day,
        energy_sum,
        LAG(energy_sum, 1) OVER (PARTITION BY lclid ORDER BY day) AS prev_day,
        LEAD(energy_sum, 1) OVER (PARTITION BY lclid ORDER BY day) AS next_day
    FROM consumos
)
SELECT
    lclid,
    COUNT(*) AS consecutive_days_with_low_consumption
FROM consistencycheck
WHERE
    energy_sum < 0.1 AND prev_day < 0.1 AND next_day < 0.1
GROUP BY lclid;
```

The results shows that there has been 213 households that have been inconsistent for at least 3 days in a row.

10. Consumo total de energía por franja horaria (mañana, tarde, noche):

    [!IMPORTANT] This exercise is not possible with the given dataset.

11. Final Boss: How much does the average consumption change between weekdays and weekends?: To analyze how energy consumption patterns differ between weekdays and weekends, we will use the `DAYOFWEEK` function to classify days and calculate average consumption for each type. We first mark each day as either a weekday or weekend, then averages the energy consumption for each category.

```
WITH consumption_by_day_type AS (
    SELECT
        lclid,
        day,
        energy_sum,
        CASE
            WHEN DAYOFWEEK(day) IN (1, 7) THEN 'weekend' -- 1 is sunday 7 is saturday
            ELSE 'weekday'
        END AS day_type
    FROM consumos
)
```

```sql
SELECT
    day_type,
    AVG(energy_sum) AS avg_energy_consumption
FROM consumption_by_day_type
GROUP BY day_type;
```

The results shows that there has been a difference in average consumption between weekdays and weekends, having more energy consumption on weekends.

day_type	avg_energy_consumption
weekday	9.987461653068868
weekend	10.46783693386403

Conclusions

We have successfully answered all questions. Personally, I did not found any significant difficulty regarding some special functions of Hive SQL, which i am not so familiar with.

Regarding the impact of not using HDFS, is that we do not have any data persisted in a distributed file system, which can lead to data loss in case of failures. It can also lead to performance issues, as data is not distributed across multiple nodes, and all operations are performed on a single node. Finally, it is viable to use Hive without HDFS, but it is not recommended for production environments due to the lack of fault tolerance and scalability.