# Milestone -1

What is Data wrangling?

Data wrangling is the process of converting and mapping raw data and getting it ready for analysis.

The POC is to demonstrate effective data storage and interactive data analysis, thus data wrangling.

## Reasoning :

***1. Identify the storage system which you would like to use as a part of POC → HIVE***

Why am I selecting HIVE for storage?

1. Interactive data analysis is the priority so the storage system should be a data warehouse rather than a database.
2. Hive is open-sourced and has a good active community working on it.
3. Hive stores data in HDFS, where it creates replicas for fault tolerance and also Hive supports partitioning, bucketing, ORC,Parquet file types for faster query times.
4. Data is well structured, So no need of NoSql databases, Hence options to pick from → Hive, Bigquery, Snowflake, Redshift, Teradata

As Blackburn.Co serves European clients, It’s familiar with strict GDPR practices. And also no matter how good the security with cloud providers, they will always be a common target for hackers. Since the energy data is sensitive I wanted utmost security so opted for a non-cloud based solution.

***2. Ensure you increase the availability of data in data storage***

I think here you are implying to select multi-region mode in any cloud services to make the data available from any region. Since I am not using cloud I can’t make it available from any region.

But if it’s availability in general perspective,

I am making sure the replication of the data is at least 3, so datanode failovers can be easily accepted making it highly available.

hadoop fs -setrep -R 3 /user/cloudera/electricity\_data

***3. Data stored should be accessible to write scripts and create interactive queries for ad-hoc requests***

Here to write scripts we are using spark(in part-3) and for ad-hoc queries we have decided to use Hive. So the same data should suffice both conditions, I cannot create a duplicate set of the data(mentioned in part-3).

So this is the classic use case of External Table. Where the same data can be used by multiple tools as it won’t have the danger of accidentally being deleted from Hive.

## Loading Logic :

## The schema of all the buildings is not the same. By going through the schema of every distinct building(16), I was able to separate them into 3 sets based on their schema.

The details of those are available in file → *different-schema's.csv*

Steps :

1. Firstly load the data into an external table named → **stage**.
2. Create 3 different views on top of the stage table based on the 3 different schemas we identified.
3. Create a final view by using union, named → **vw\_energy\_new**
4. Create a managed table with partition in place named → **managed\_stage**
5. Load the data into a managed\_stage table using the final view. This loads the data that is partitioned dynamically.
6. Copy this uniformly structured and partitioned into the home directory.
7. Now create the final external table using partitions in place and add all the 14 place partitions by using ALTER TABLE ADD PARTITION command. → **energy.**
8. **ENERGY →** Is the final table that customers are gonna use.

## Schema :

External Table Name : energy

| **Column Name** | **Data Type** |
| --- | --- |
| time | timestamp |
| month | int |
| day | int |
| hour | int |
| building\_name | string |
| facility\_hourly | struct<electricity:double,gas:double> |
| electricity | struct<fans:double,cooling:double,heating:double,interiorlights:double,interiorequipment:double> |
| gas | struct<heating:double,interiorequipment:double,waterheater:double> |
| facility\_monthly | struct<electricity:double,gas:double> |
| place | string |

Other Details:

PARTITIONED BY place(because in the part-2 ad-hoc query it would be necessary to efficiently filter the data),

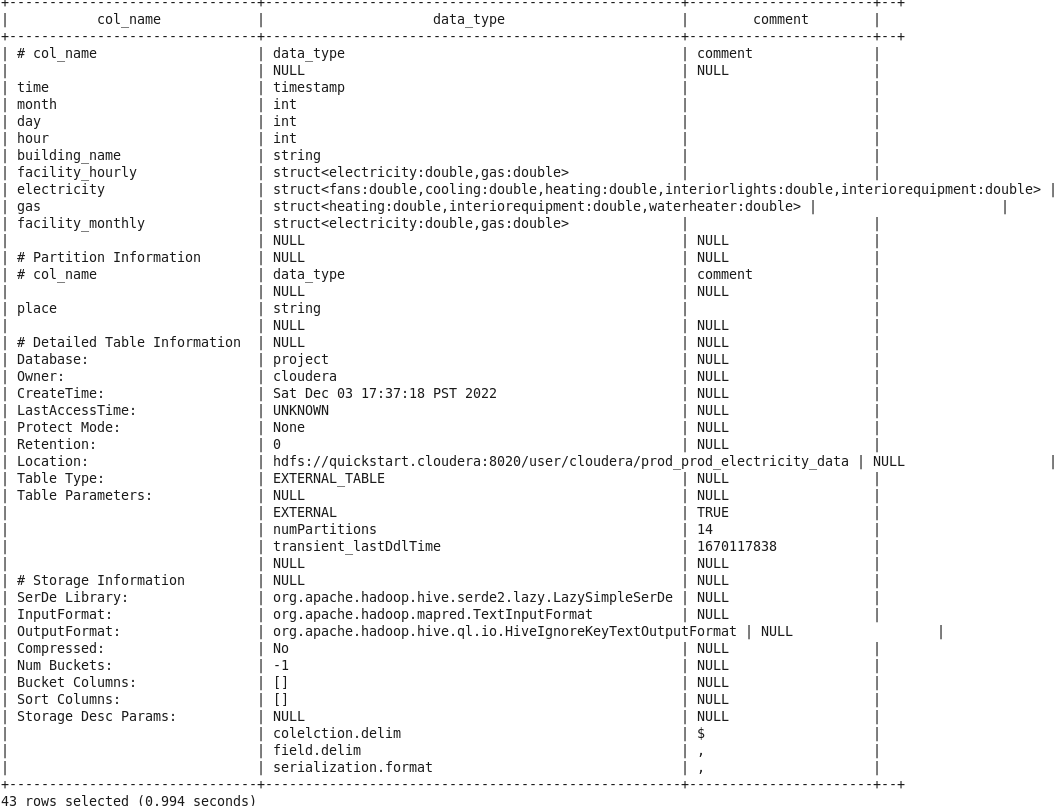
stored AS TEXTFILE

## Loading Scripts :

Scripts used are mentioned in this file → scripts.sql

## Screen Shots :

describe formatted energy;



show partitions stage;

