Master Data Analysis

First of all, I have tried to read the master file, 'PBJ_Daily_Nurse_Staffing_Q2_2024.csv', using Pandas library, but I got a file integrity or structure based problem, which is formal in large files because of windows-based characters, mostly. This is a classic data-engineering problem. The error is:

```
UnicodeDecodeError: 'utf-8' codec can't decode byte 0x92 in position 173163: invalid start byte
```

To fix this problem, Read the file with the correct encoding, most likely with **'cp1252'**, otherwise with **'latin1'**:

Option 1:

```
df = pd.read_csv(
    r'.\data\master_data.csv',
    encoding='cp1252'
)
```

Option 2:

```
df = pd.read_csv(
    r'.\data\master_data.csv',
    encoding='latin1'
)
```

After solving this problem, then I have tried to analyze deeply this master data, like duplication rows or columns, missing NA values and histogram of all columns.

Data Duplication:

Check for duplicate rows and columns in master data:

```
No duplicate rows or columns found
```

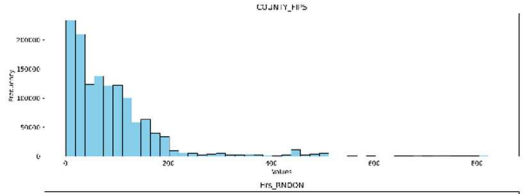
Missing Values:

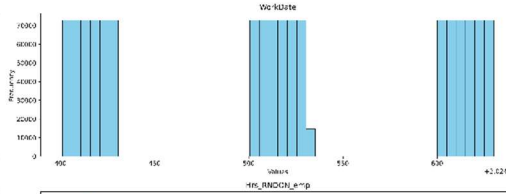
Check for missing values in master data:

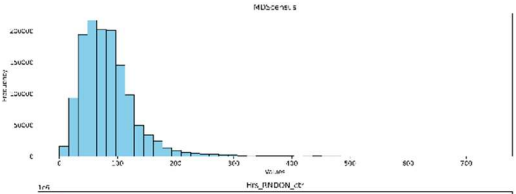
```
Is there missing values in the master_dataset?  False
```

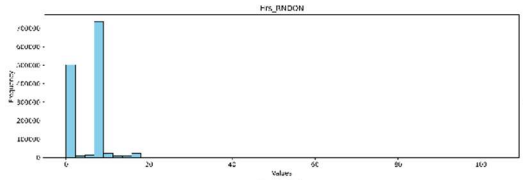
Histogram Analysis:

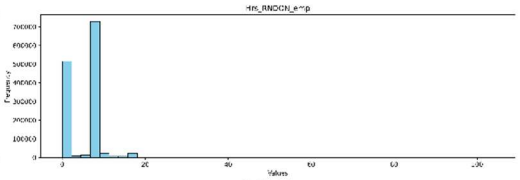
Here is the histogram for only numerical columns:

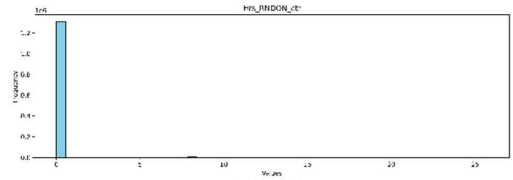

COUNTY_HRS
Frequency
Values

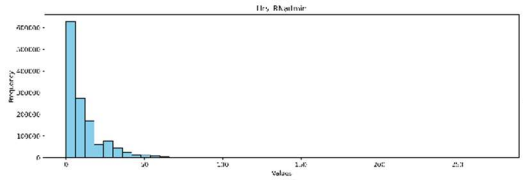

WorkDate
Frequency
Values

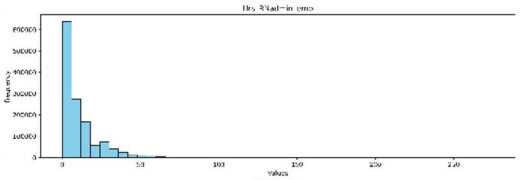

MDScensus
Frequency
Values

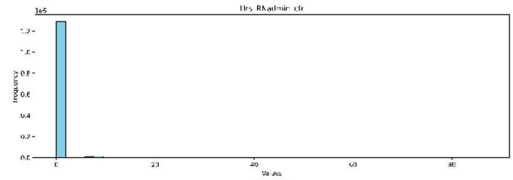

Hrs_RNDON
Frequency
Values

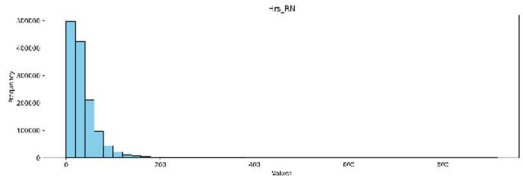

Hrs_RNDON_emp
Frequency
Values

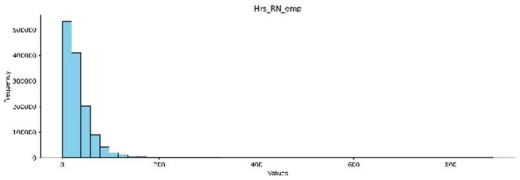

Hrs_RNDON_ctr
Frequency
Values

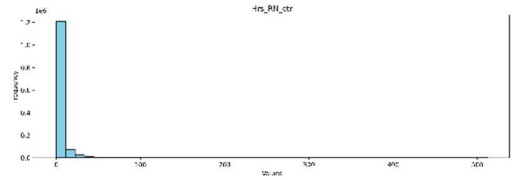

Hrs_RNadmin
Frequency
Values

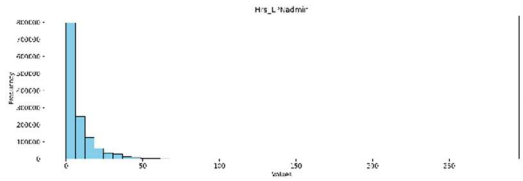

Hrs_RNadmin_emp
Frequency
Values

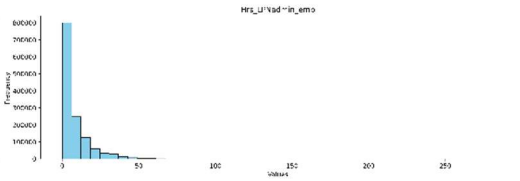

Hrs_RNadmin_ctr
Frequency
Values

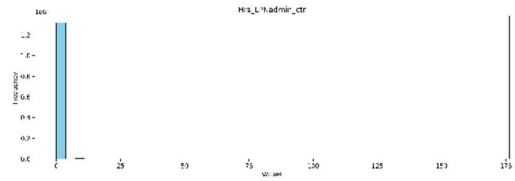

Hrs_RN
Frequency
Values

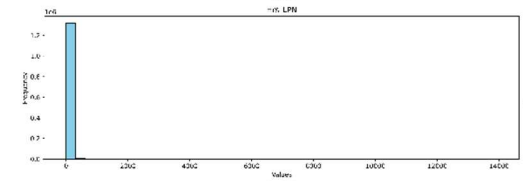

Hrs_RN_emp
Frequency
Values

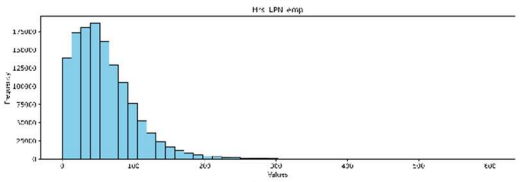

Hrs_RN_ctr
Frequency
Values

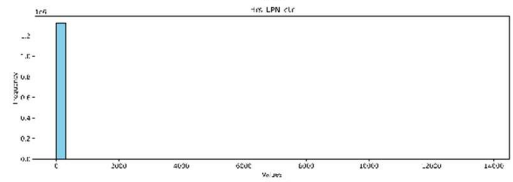

Hrs_LPNadmin
Frequency
Values

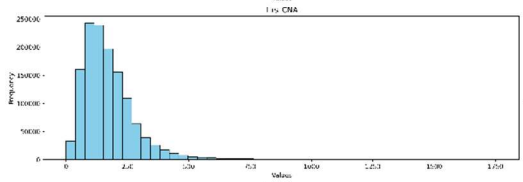

Hrs_LPNadmin_emp
Frequency
Values

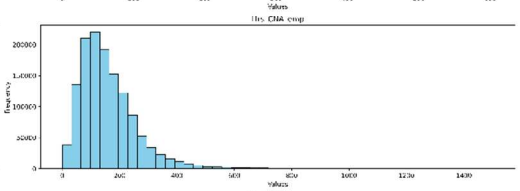

Hrs_LPNadmin_ctr
Frequency
Values

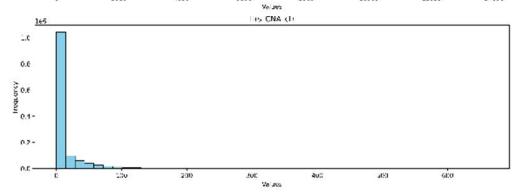

Hrs_LPN
Frequency
Values

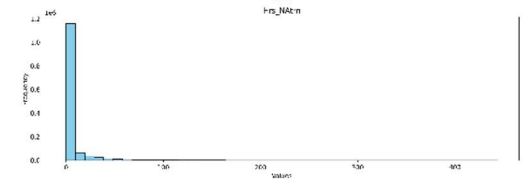

Hrs_LPN_emp
Frequency
Values

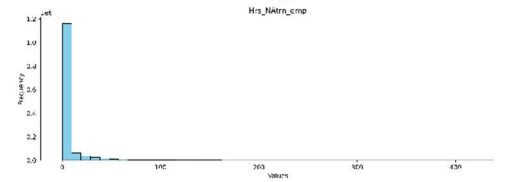

Hrs_LPN_ctr
Frequency
Values

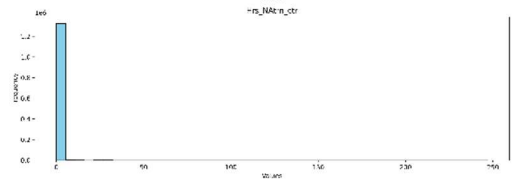

Hrs_CNA
Frequency
Values

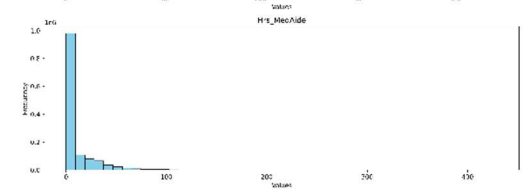

Hrs_CNA_emp
Frequency
Values

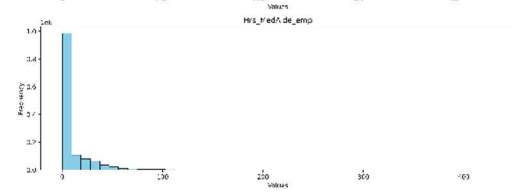

Hrs_CNA_ctr
Frequency
Values

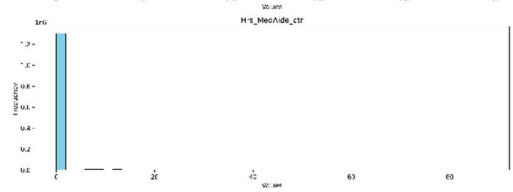

Hrs_NAtrn
Frequency
Values


Hrs_NAtrn_emp
Frequency
Values

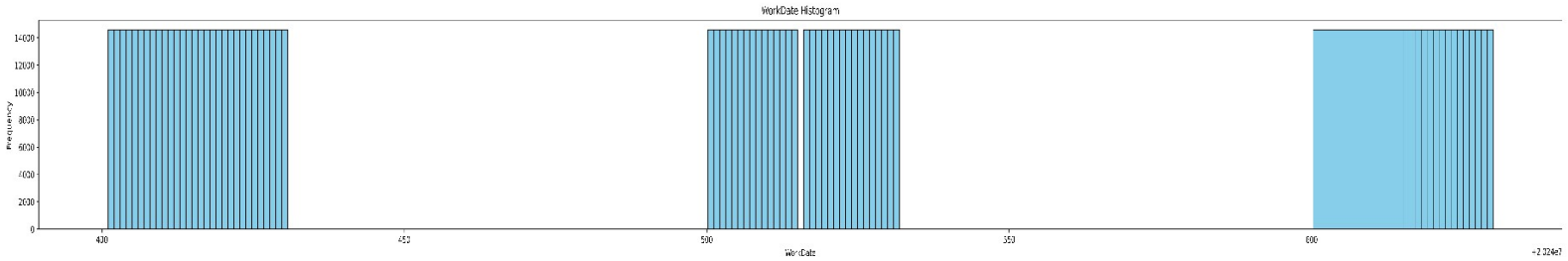

Hrs_NAtrn_ctr
Frequency
Values


Hrs_MedAide
Frequency
Values


Hrs_MedAide_emp
Frequency
Values


Hrs_MedAide_ctr
Frequency
Values

Some columns have a semi-normal distribution, but some are highly skewed. But for ‘WorkDate’ column, we have unified distribution.



Then, I decided to select a primary key for this dataset, which will be unique. I found that the combination of `('PROVNUM', 'WorkDate')` will be a good choice for unique key.

```
unique_key_dups = df[['PROVNUM', 'WorkDate']].duplicated().any()
False
```

I chosed all fileds, except three of them, as I need them in downstream analysis.

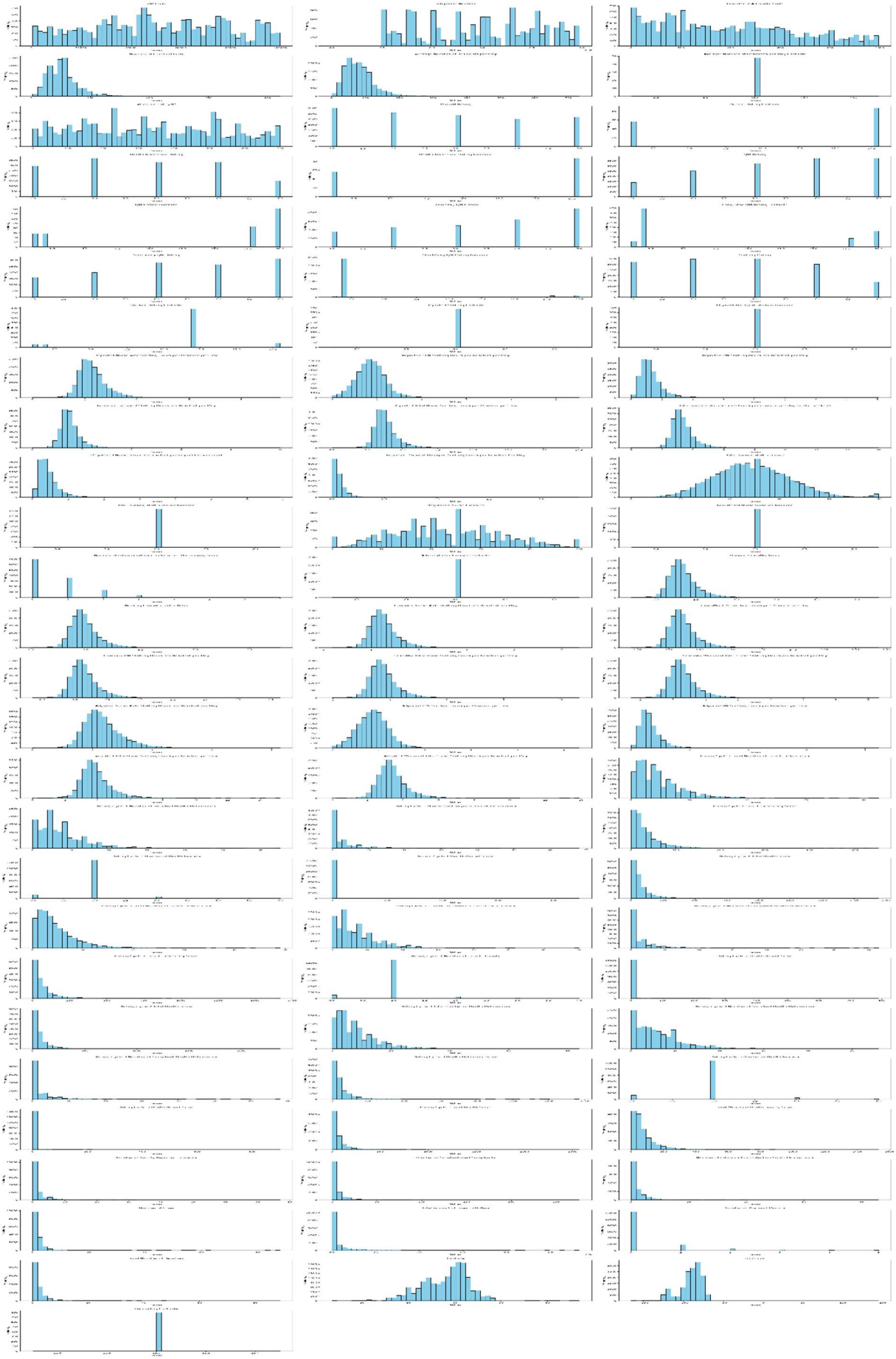
['PROVNUM', 'PROVNAME', 'CITY', 'STATE', ~~'COUNTY_NAME', 'COUNTY_FIPS', 'CY_Qtr',~~ 'WorkDate', 'MDScensus', 'Hrs_RNDON', 'Hrs_RNDON_emp', 'Hrs_RNDON_ctr', 'Hrs_RNadmin', 'Hrs_RNadmin_emp', 'Hrs_RNadmin_ctr', 'Hrs_RN', 'Hrs_RN_emp', 'Hrs_RN_ctr', 'Hrs_LPNadmin', 'Hrs_LPNadmin_emp', 'Hrs_LPNadmin_ctr', 'Hrs_LPN', 'Hrs_LPN_emp', 'Hrs_LPN_ctr', 'Hrs_CNA', 'Hrs_CNA_emp', 'Hrs_CNA_ctr', 'Hrs_NAtrn', 'Hrs_NAtrn_emp', 'Hrs_NAtrn_ctr', 'Hrs_MedAide', 'Hrs_MedAide_emp', 'Hrs_MedAide_ctr']

Then, I did some statistical analysis for this dataset:

```
This dataset contains information reagarding quarter of ['2024Q2']
There are 14378 providers in the dataset
There are 5086 cities in the dataset
There are 52 states in the dataset
There are 1672 counties in the dataset
There are 631 MD-Censsus in the dataset
There are 91 different work dates in the dataset starting from 20240401 until 20240630
```

Additional Data Analysis

Next, I tried to analyze 20 additional data. All of them working well. However, based on analysis questions, afrer exploring them, I chose only one additional data as my dim table: **‘NH_ProviderInfo_Oct2024.csv’**. This table has a lot of information, which I need for the downstream analysis. For this dataset, I have repeated all above data analysis. There is no duplication in rows or files, the data is not corrupted, and there is no missing value. Here is the histogoram of its columns:



Then, I decided to select a primary key for this dataset too, which will be unique. I found that the `'CMS Certification Number (CCN)'`, which is fitted with `'PROVNUM'` in master data, will be a good choice for unique key.

```
unique_key_dups = df ['CMS Certification Number (CCN)'].duplicated().any()
False
```

Finally, I chosed only 23 usefull fileds from this dataset, which I need them in downstream analysis.

Here is my dim and fact tables for Silver layer!

dim_providere	
Source: NH_ProviderInfo_Oct2024.csv	
#	**Fields**
1	**CMS Certification Number (CCN) [PK]**
2	Provider Name
3	Provider Type
4	State
5	City/Town
6	ZIP Code
7	Number of Certified Beds
8	Average Number of Residents per Day
9	Average Number of Residents per Day Footnote
10	Reported Nurse Aide Staffing Hours per Resident per Day
11	Reported LPN Staffing Hours per Resident per Day
12	Reported RN Staffing Hours per Resident per Day
13	Reported Licensed Staffing Hours per Resident per Day
14	Reported Total Nurse Staffing Hours per Resident per Day
15	Total number of nurse staff hours per resident per day on the weekend
16	Registered Nurse hours per resident per day on the weekend
17	Reported Physical Therapist Staffing Hours per Resident Per Day
18	Total nursing staff turnover
19	Total nursing staff turnover footnote
20	Registered Nurse turnover
21	Registered Nurse turnover footnote
22	Number of administrators who have left the nursing home
23	Administrator turnover footnote

fact_table	
Source: PBJ_Daily_Nurse_Staffing_Q2_2024.csv	
#	**Fields**
1	**PROVNUM [PK]**
2	PROVNAME
3	CITY
4	STATE
5	**WorkDate [PK]**
6	MDScensus
7	Hrs_RNDON
8	Hrs_RNDON_emp
9	Hrs_RNDON_ctr
10	Hrs_RNadmin
11	Hrs_RNadmin_emp
12	Hrs_RNadmin_ctr
13	Hrs_RN
14	Hrs_RN_emp
15	Hrs_RN_ctr
16	Hrs_LPNadmin
17	Hrs_LPNadmin_emp
18	Hrs_LPNadmin_ctr
19	Hrs_LPN
20	Hrs_LPN_emp
21	Hrs_LPN_ctr
22	Hrs_CNA
23	Hrs_CNA_emp
24	Hrs_CNA_ctr
25	Hrs_NAtrn
26	Hrs_NAtrn_emp
27	Hrs_NAtrn_ctr
28	Hrs_MedAide
29	Hrs_MedAide_emp
30	Hrs_MedAide_ctr

Proposal:

Now, after preanalysis of the master and additional data, here is my proposal for this project:

Step 1 – Data Ingestion (S3 Bronze Layer)

The first goal is to pull raw master file and additional CSV files from Google Drive into the S3 Bronze bucket named **project2-healthcare-bronze-bucket**.

Implementation:

- Use an AWS Glue Python Shell Job to fetch files from Google Drive.
- Store Google Drive credentials securely in AWS Secrets Manager.
- Grant the Glue job IAM permissions to write to the S3 bucket.
- The Glue job can be run manually or scheduled via Glue Scheduler (daily or weekly).
- Enable S3 bucket versioning to retain all versions of the raw data.

Note: The Bronze layer preserves the raw, original CSV files for auditing and traceability.

Step 2 – Transformation (S3 Silver Layer)

In this step, the goal is to clean and standardize the raw data, organizing it into dimensional and fact tables for downstream analysis.

Implementation:

- Use another S3 bucket: **project2-healthcare-silver-bucket**.
- Run an AWS Glue PySpark Job to perform the following tasks:
    - Enforce schemas for dimensional and fact tables
    - Correct data types (if needed)
    - Remove duplicates (if needed)
    - Handle missing values and outliers (if needed)
    - Normalize column names (if needed)

Note: Since the source data is moderate in size, partitioning or converting CSV to Parquet is not necessary at this stage.

Step 3 – Analytics & Aggregation (S3 Gold Layer)

In this step, the goal is to create aggregated, dashboard-ready datasets by joining the dimensional and fact tables from the Silver layer.

Implementation:

- Store the outputs in a new S3 bucket: **project2-healthcare-gold-bucket**.
- Use another Glue PySpark Job to generate metrics based on the analytics questions, such as:
    - Nurse-to-patient ratio

- Staffing hours at different levels
- Provider- and State-level aggregated metrics
- Other relevant KPI metrics

Note: The Gold layer is optimized for Streamlit consumption and easy visualization.

Step 4 – Visualization (Streamlit)

In the final step, Streamlit dashboards will query data directly from the Gold S3 bucket.

Implementation:

- Use pandas for data handling (sufficient for moderate-sized CSV files).
- Build interactive dashboards for stakeholders with filters, charts, and KPI cards.

Note: No data warehouse or query engine is required; this approach is fully serverless and cost-efficient.

Conclusion

This pipeline is a cost-efficient batch processing solution using AWS Glue and S3, following a Bronze-Silver-Gold lakehouse pattern. Streamlit consumes cleaned CSVs directly from S3, eliminating the need for Athena or Redshift, while ensuring traceability, flexibility, and simplicity.