Data Swan Inc: Medicaid and Medicare evaluation of sponsorship

and ratings

Initial ETL Pipeline  
  
Team #11

Lucius Anderson

Anita Pinto

Henry Rivera

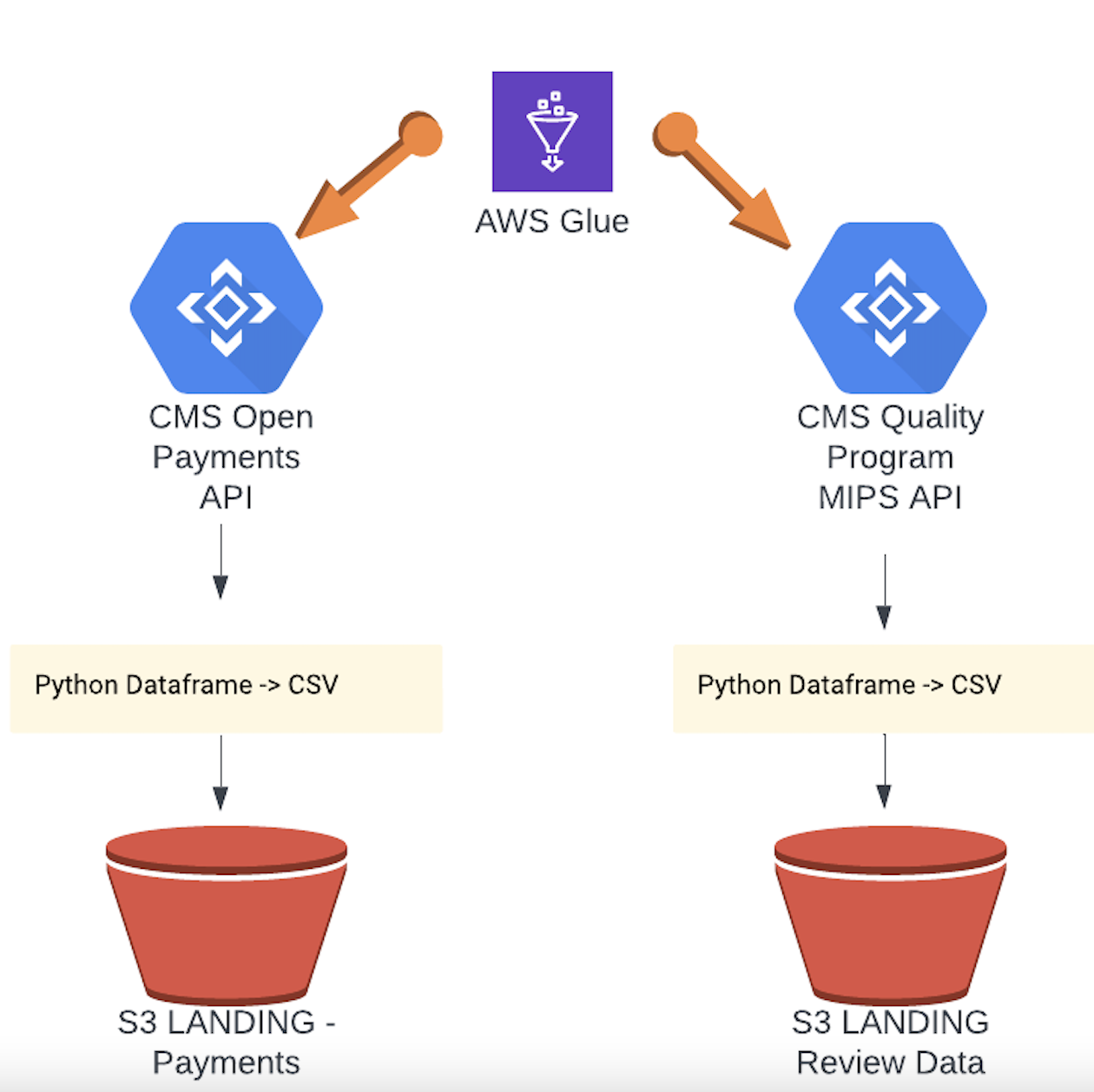
Anthony Robnett

## ELT Step 1: Extract

Our extraction process involves two python scripts and AWS Glue jobs. Our first python script leverages the OpenPaymentsData API to acquire raw Research, Ownership, and General payment data for 2020 and 2021. The data for each of these types is read into a pandas dataframe which is then converted to a CSV that is placed in the landing section of our S3 bucket. The automation of this script is handled via an AWS Glue job. No cadence has been established for this Glue job to avoid incurring charges. If we had an unlimited budget the Glue job would be set to run on a monthly basis. We do not foresee data from 2020 or 2021 changing so there would not be a need to be running more frequently, but we would need to be checking in case of changes to the data. On the topic of looking to avoid unwanted charges, we are only reading in one million rows of each data file because of space restrictions to the free tier of S3.

For the review data, we used CMS’s Quality Payment Program Dataset which has datasets from 2017. We used CMS’s API endpoints to download the 2020 and 2021 MIPS review data. Each dataset is around 600K-900K rows and 92 columns. For the reviews, we only fetched the individual final quality scores and the final MIPS score. The review dataset extraction is also run using a python script and the fetched pandas dataframe is loaded as a CSV in the landing section of S3 bucket via AWS Glue Job.

The NPI ( Physician ID) would serve as a common field to link both the payments and review dataset.



## ELT Step 2: Transform (data cleansing, data deduplication and data format revision)

Tools Used: Pyspark, AWS Glue Visual Transform

**Data Cleansing:**

The code reads the Payment Data from S3 Landing Folder and applies cleansing transformations.

The ApplyMapping function is used to map the input columns of the data to their corresponding output columns. The mapping includes specifying the data types and format revisions.

**Data Deduplication:**

The code reads the Review data from the LANDING S3 bucket and applies deduplication transformations.

Similar to the General Payment data, the ApplyMapping function is used to map the input columns of the Review data to their corresponding output columns, specifying the data types and format revisions.

The transformed Review data is written to a new location in the S3 bucket

**Data Format Revision:**

Both the General Payment and Review data have columns that are converted to the desired data types and formats during the mapping process.

For example, PAYMENT\_DATE column is converted from string to date format, and COST\_CATEGEORY\_SCORE column is renamed to COST\_CATEGORY\_SCORE.

The transformed data is stored in the specified locations as Glue Parquet format.

**Data Quality Check:**

After the transformation steps for General Payment and Review data, data quality checks are performed to ensure the data is not empty.

The SelectFields function is used to select a specific field (PAYMENT\_KEY and REVIEW\_KEY respectively) from the transformed data frames.

The count of selected fields is obtained, and if the count is 0, an exception is raised indicating that the data quality check failed.

**Joining Payment and Review Data:**

Finally, the transformed General Payment and Review data frames are joined using the Join.apply function with the CR\_PHYS\_NPI column as the join key.

The joined data frame is written to a new location in the S3 bucket (PaymentReviewSchemaS3bucket\_node4), storing the integrated payment and review data in Glue Parquet format.

This code successfully performs data cleansing, deduplication, and format revision for the payment and review data, ensuring data quality and integrating them based on the common CR\_PHYS\_NPI column.