Technical Test Solution: Data Engineering, Data Science, and Data Analytics

Introduction

This document outlines the steps taken to complete the technical test for Data Engineering, Data Science, and Data Analytics using AWS services. Each section describes the methodology, tools, and AWS services employed, as well as the results obtained.

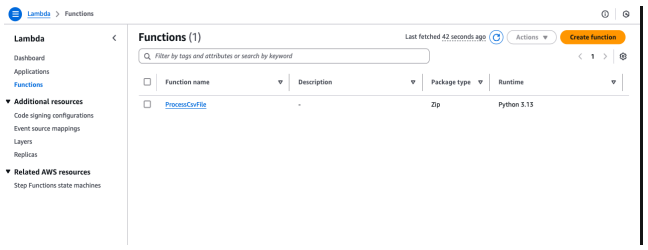
Data Engineering

1. Inspected the and Documented the Data:

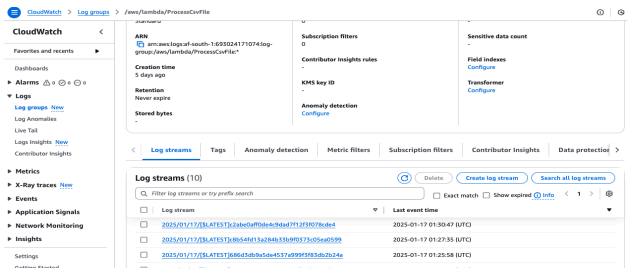
    - The CSV contained one column that had one multiple column name.
    - Missing values in some columns.
    - Data types were Incorrect.
    - Data had rows that were empty throughout the columns.
    - Some values were aligned in incorrect columns.

2. Actions Taken:

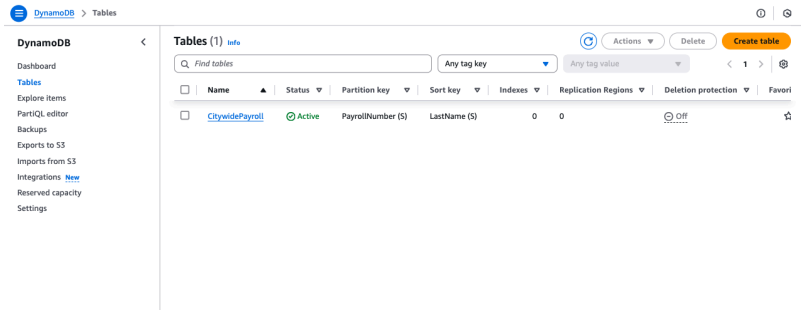
    - I used AWS S3(**citywide-payroll-data-lake**) to store the raw dataset. This will mimic a Data Lake for unstructured data.
    - I used AWS Lambda to monitor the S3 bucket for changes and trigger further processes. Created a Function Called **ProcessCsvFile.**
    - Successfully set up an S3 bucket event notification ensuring data is protected with **IAM policies** and encryption.



    - I used CloudWatch to check for updates.



3. ETL to Data Warehouse:
    - I used ETL to Data Warehouse and transform the data into a structured format.
    - Successfully deployed and tested the solution.
    - Stored the structured data in DynamoDB. The table represents my Database named “CityWidePayroll”



4. Challenges encountered in this section:

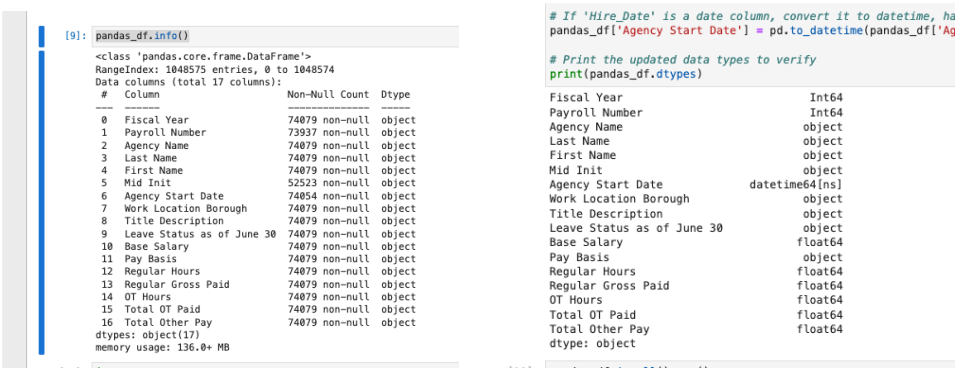
    - I did not experience many problems in this section. I encountered some problems in the following sections.

Data Science

Part 1

1. Additional Cleaning and Structuring:

    - Made use of the Notebook in AWS Glue Studio. Instead of using Spark DataFrame to further clean the data. I converted the uploaded DF to a pandas DataFrame. It made it easier for me to look through the data and clean it.
    - The data consisted of extra rows and an extra column.
    - The data types needed to be converted into correct data types.



```
[9]: pandas_df.info()

<class 'pandas.core.frame.DataFrame'>
RangeIndex: 1048575 entries, 0 to 1048574
Data columns (total 17 columns):
 #   Column                        Non-Null Count    Dtype
---  ------                        --------------    -----
 0   Fiscal Year                   74079 non-null    object
 1   Payroll Number                73937 non-null    object
 2   Agency Name                   74079 non-null    object
 3   Last Name                     74079 non-null    object
 4   First Name                    74079 non-null    object
 5   Mid Init                      52523 non-null    object
 6   Agency Start Date             74054 non-null    object
 7   Work Location Borough         74079 non-null    object
 8   Title Description             74079 non-null    object
 9   Leave Status as of June 30    74079 non-null    object
 10  Base Salary                   74079 non-null    object
 11  Pay Basis                     74079 non-null    object
 12  Regular Hours                 74079 non-null    object
 13  Regular Gross Paid            74079 non-null    object
 14  OT Hours                      74079 non-null    object
 15  Total OT Paid                 74079 non-null    object
 16  Total Other Pay               74079 non-null    object
dtypes: object(17)
memory usage: 136.0+ MB
```

```
# If 'Hire_Date' is a date column, convert it to datetime, ha
pandas_df['Agency Start Date'] = pd.to_datetime(pandas_df['Ag

# Print the updated data types to verify
print(pandas_df.dtypes)

Fiscal Year                      Int64
Payroll Number                   Int64
Agency Name                     object
Last Name                       object
First Name                      object
Mid Init                        object
Agency Start Date       datetime64[ns]
Work Location Borough           object
Title Description               object
Leave Status as of June 30      object
Base Salary                    float64
Pay Basis                       object
Regular Hours                  float64
Regular Gross Paid             float64
OT Hours                       float64
Total OT Paid                  float64
Total Other Pay                float64
dtype: object
```

- Some Columns had incorrect values that belonged to other columns. Alignment was needed.

```
array(['BROOKLYN', 'MANHATTAN', 'BRONX', 'RICHMOND', 'QUEENS',
       '11/20/2000', '01/07/2019', 'WASHINGTON DC', 'ULSTER', 'OTHER',
       '12/07/1992', '12/02/2013', 'WESTCHESTER', '07/11/2005',
       '08/31/1998', '03/01/2000', '06/30/1995', '06/30/1992',
       '07/01/2004', '07/22/2002', '07/01/2003', '07/01/1998',
       '01/09/2006', '03/04/2019', '01/14/2009', '01/17/2008',
       '07/06/2010', '07/01/2002', '07/09/2007', '07/12/2006',
       '01/10/2007', '12/19/2005', '12/08/1997', '01/23/2007',
       '01/09/2013', '01/09/2012', '01/08/2014', '09/29/2000',
       '12/26/2001', '07/10/2006', '01/25/2010', '01/23/2006',
       '05/29/2007', '04/06/2016', '09/24/2013', '11/29/1999',
       '12/21/2006', '09/22/2008', '07/18/2005', '07/05/2017',
       '09/26/2011', '08/30/2013', '08/22/2005', '02/28/2012',
       '08/30/1993', '02/25/1994', '07/28/1987', '04/11/2013',
       '05/29/2020'], dtype=object)
```

```
24]: # Define a function to check if a string is a valid date
```

2. Actions Taken:

- Removed extra rows and an extra column.
- Filled missing rows.
- Converted columns to correct data types.
- Aligned some values to the correct column.

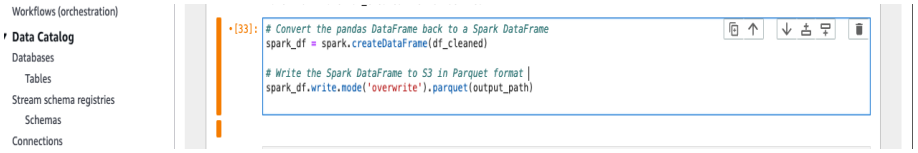
```
[29]: df_cleaned['Pay Basis'].unique()

array(['per Annum', 'per Hour', 'Prorated Annual', 'Unknown', 'per Day'],
      dtype=object)

[30]: df_cleaned.isnull().sum()

Fiscal Year                  0
Payroll Number               0
Agency Name                  0
Last Name                    0
First Name                   0
Mid Init                     0
Agency Start Date            0
Work Location Borough        0
Title Description            0
Leave Status as of June 30   0
Base Salary                  0
Pay Basis                    0
Regular Hours                0
Regular Gross Paid           0
OT Hours                     0
Total OT Paid                0
Total Other Pay              0
dtype: int64
```

- Converted the pandas DataFrame back to a Spark DataFrame.
- Wrote the Spark DataFrame to S3 in Parquet format.



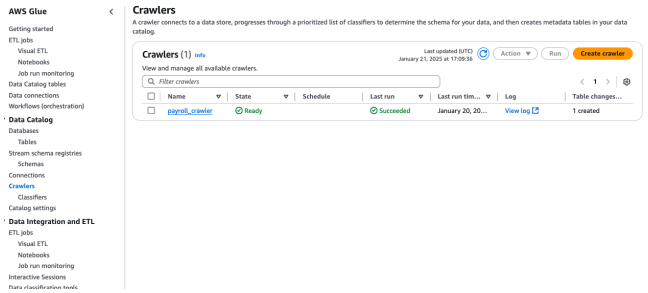
3. Challenges encountered:

- Later when I was visualising I realised that I had to clean.
- I noticed in AWS Athena that some columns had important value located in the wrong column.
- I had to go back to the Notebook in AWS Glue.
- It took me a bit of time to rectify what I had noticed.
- I still believe that I need to go over the data again to make sure everything is cleaned accordingly.

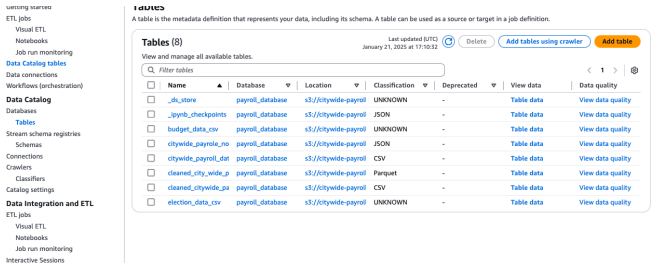
Part 2

1. Moving on to AWS Crawler:

    - Security: Ensured sensitive data is protected with proper IAM policies and encryption from the beginning.
    - The AWS Crawler was needed to scan the S3 Bucket for the data and extra metadata from CSV files.

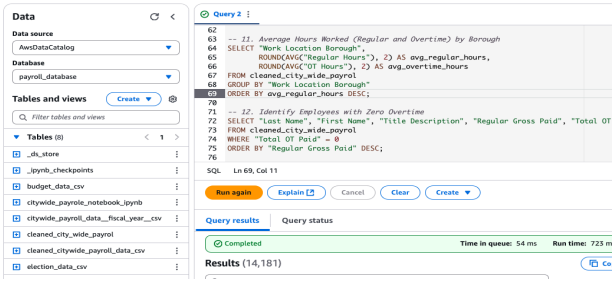


    - Set up and ran an AWS Glue ETL Job and named it , “” and it successfully transformed the data.
    - Table was also successfully created under the, “payroll_database”.



2. Moving on to AWS Athena:

    - Amazon Athena: Enabled SQL-based querying of data.
    - Advanced SQL queries in Athena were used to clean, standardize, and aggregate data, such as calculating average working hours or filtering employees with zero overtime.
    - Athena's SQL views act as the data source for Amazon QuickSight, providing the cleaned and structured datasets needed to build insightful dashboards.



3. Challenges Encountered:

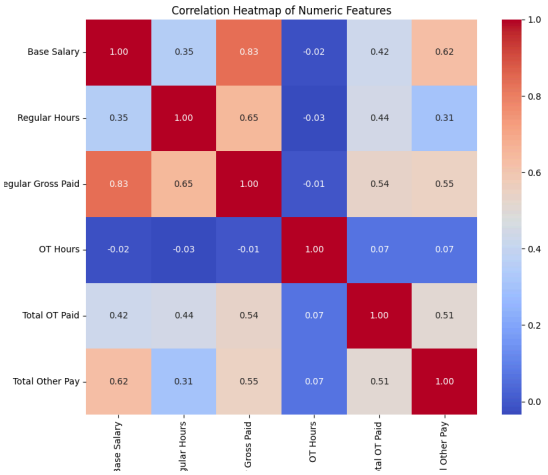
    - Having to update policies and permissions was a bit straining at some point.

Machine Learning (ML) / Artificial Intelligence (AI)

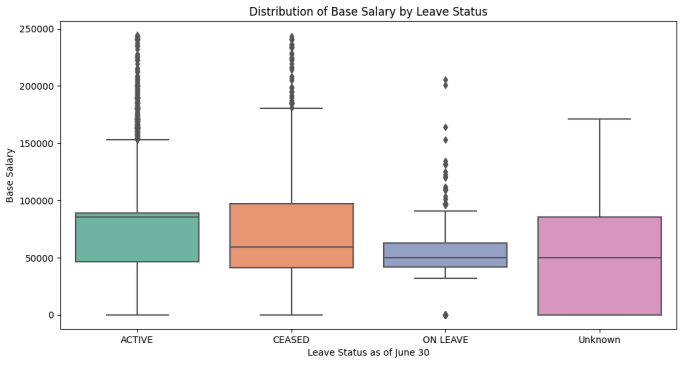
1. Exploratory Data Analysis (EDA):

    Below are images of the EDA process. This process helps us see the relation between features to help us get our target feature(Column) for our Model. The visuals were created on AWS Glue Notebook.

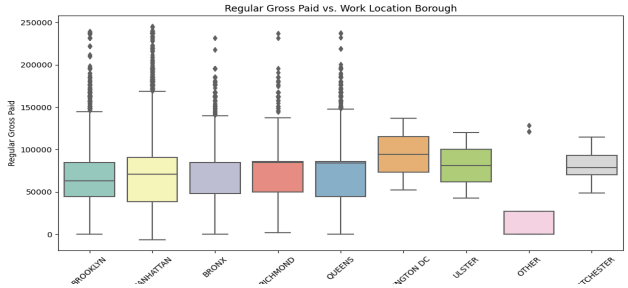
    - Correlation Heatmap



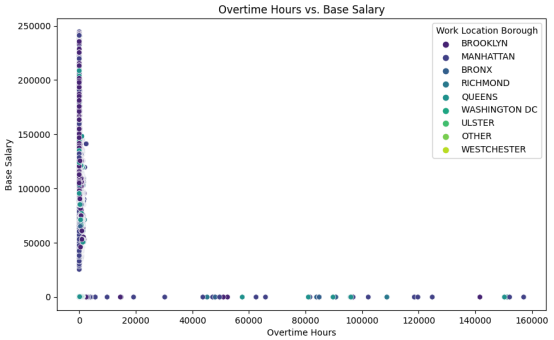
    - Distribution of Base Salary by Leave Status



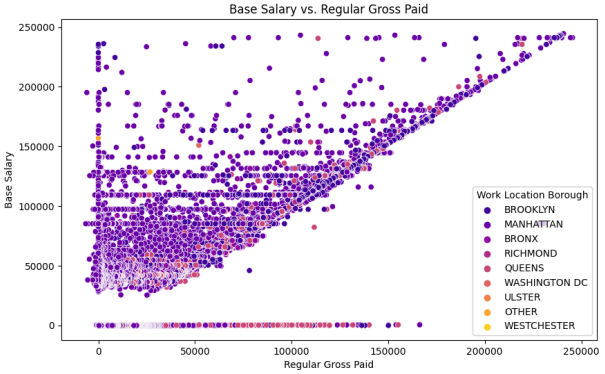
- Distribution of Base Salary by Leave Status



- Overtime Hours vs. Base Salary



- Base Salary vs. Regular Gross Paid



2. Feature Engineering

Steps I followed in AWS Glue to prepare the data for machine learning

- **Splitting the Data**: I used train_test_split from sklearn.model_selection to split the df_cleaned dataset into three parts: 70% for training, 15% for validation, and 15% for testing. I set the random_state to 42 to ensure the splits are reproducible. After splitting, I printed the shapes of the resulting datasets to verify the distribution of the data.

- **Uploading Data to S3**: After splitting the data, I used boto3 to upload the training, validation, and testing data to an S3 bucket called, “mymodeltraining”. I first saved the data locally as CSV files and then uploaded them to S3, which makes the data accessible for future model training.

- **One-Hot Encoding**: I used OneHotEncoder from sklearn.preprocessing to encode categorical variables into a machine-readable format. I applied the encoder to the categorical columns in the training, validation, and testing datasets. This transformation converts categorical values into binary columns, making them suitable for machine learning models.

- **Feature Scaling (Standardization)**: I applied StandardScaler from sklearn.preprocessing to scale the numeric features like salary and hours. This step ensures that all features contribute equally to the model by normalizing them to have a mean of 0 and a standard deviation of 1. Scaling is essential for many machine learning algorithms to perform well.

- Training the RandomForest Model: I initialized a RandomForestClassifier from sklearn.ensemble and trained it on the training data.

Challenges encountered:

- I encountered errors while training my model and took several steps to resolve them.

- To troubleshoot, I decided to inspect how the data was cleaned.

- In my experience, such errors often occur when the data hasn't been cleaned thoroughly or properly.

- This prompted me to check for any potential issues with the data cleaning process.

Data Analytics

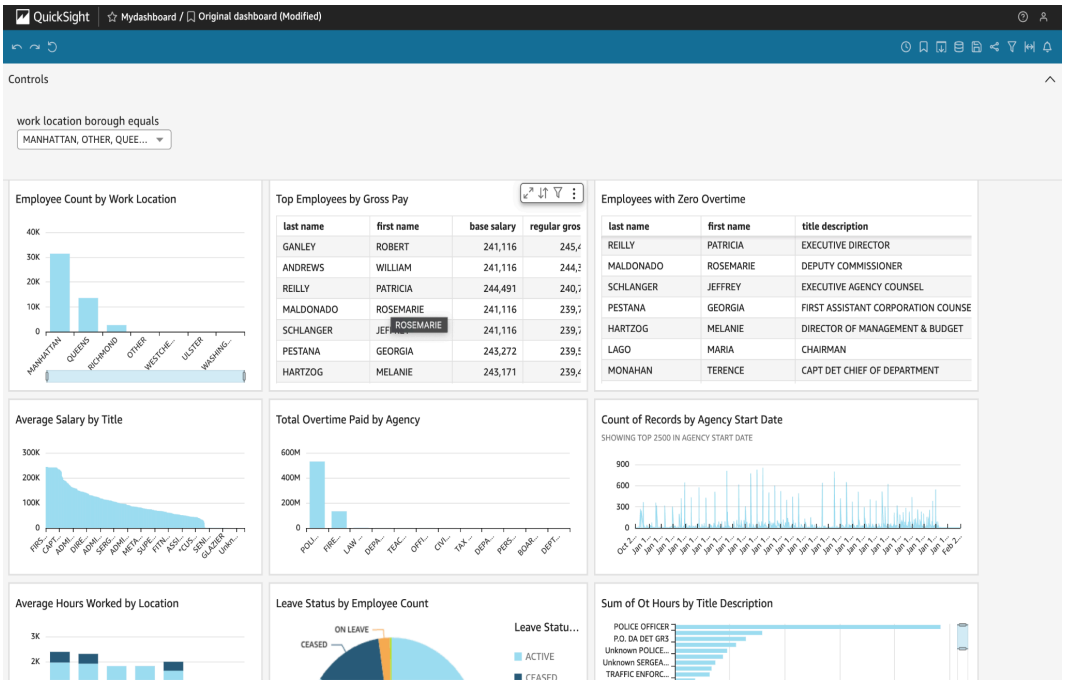
1. Visualizing using Amazon QuickSight:

    - Athena sql views acted as the data source for Amazon QuickSight, providing the cleaned and structured datasets needed to build insightful dashboards.
    - QuickSight used SPICE to optimize dashboard performance.

2. Challenges Encountered:
    - I had some problems with signing in to Amazon QuickSight but it was later resolved.

Below is a Dashboard made on Amazon Quicksight:

1. Dashboard:



2. Some of the Insights from the Visuals:

    - Top 3 work location boroughs for total count of first name are:
        1. MANHATTAN with 31,348
        2. BROOKLYN with 17,478
        3. QUEENS with 13,537

    - Top 3 title descriptions for average base salary are:
        1. FIRST ASSISTANT CORPORATION COUNSEL with 243,272
        2. CHAIRMAN with 243,171
        3. DIRECTOR OF MANAGEMENT & BUDGET with 243,171

Conclusion

This project demonstrates the ability to leverage AWS services to build a complete data pipeline from raw data to insights. Automation, AI/ML, and visualization tools were integrated to meet the test requirements.