2022 Intern Project Data Design and Pipeline Runbook- ML/Analytics Workstream

August 2022

Prepared by:

**Sean Slavich**

Contents

[1) Functionality Overview 3](#_Toc111127102)

[2) Workstream Overview 3](#_Toc111127103)

[3) Architecture 4](#_Toc111127104)

[4) Modeling/Inference Pipeline 4](#_Toc111127105)

[5) SageMaker Model Specifics 7](#_Toc111127106)

[6) Current Inference Pipeline 8](#_Toc111127107)

[7) Historical Data Pipeline 10](#_Toc111127108)

[8) QuickSight 11](#_Toc111127109)

[9) Integration with Data Lake 15](#_Toc111127110)

[10) How I Mocked the Data 15](#_Toc111127111)

[11) Outcomes 17](#_Toc111127112)

[12) Appendix 17](#_Toc111127113)

[a) List of Useful Links 17](#_Toc111127114)

# Functionality Overview

The overall purpose of the ML/Analytics workstream is to predict whether an employee will accept or decline to interview a candidate with SageMaker and to display the trends/predictions outputted from this model and from historical data onto QuickSight dashboards. Hiring will be able to use this data to improve hiring efficiency (total cycle time) and improve employee interview acceptance rates across AWS Professional Services. Hiring will also be able to schedule more efficiently and balance the disparity between employees who interview frequently and those who do not.

# Workstream Overview

The Machine Learning/Analytics workflow starts where the data lake workflow ends. After the Glue job is complete that stitches data together from different sources, this new master dataset is uploaded to an S3 curated bucket. From this bucket, the data will be exported into two places: Athena and SageMaker.

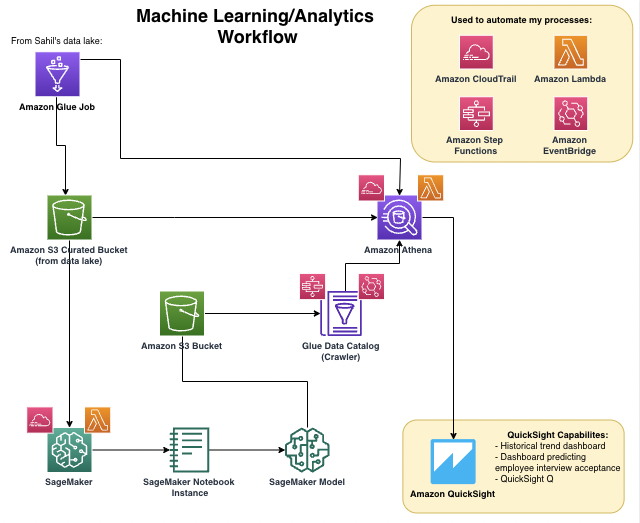
For the path directly to Athena (the historical trends pipeline), a query on the master dataset is scheduled to execute via Lambda based on the successful completion of the Glue job denoted in CloudWatch logs. This dataset is then imported into SPICE on QuickSight which powers the historical trend dashboard. Since the Athena table is automatically updated with the Lambda function, the QuickSight dashboard is updated as well, but scheduled to refresh on a daily basis.

For the path directed toward SageMaker, there are currently three notebooks that power the logistic regression model: a pre-processing notebook, a training notebook, and a deployment notebook. The pre-processing notebook pulls the most recent dataset from the S3 curated bucket and performs any necessary feature engineering to get the model ready for training and deployment. The training notebook trains the data and only needs to be run occasionally as I have saved the model via [joblib](https://scikit-learn.org/stable/model_persistence.html) so no unnecessary cost is incurred. The deployment notebook takes our trained model and makes inferences on new data.

These inferences are exported to an S3 bucket. The metadata of this file is read into a Glue data catalog which is triggered to run every 12 hours via a Step function and EventBridge rule. As a result of the Glue job, Athena can now query this dataset, which is then imported to SPICE in QuickSight from Athena. The outcome of this dataset in QuickSight is a prediction dashboard where we can infer trends with our predictions. This dashboard is also on a daily refresh schedule, meaning it will be automatically updated as well.

An additional piece of QuickSight functionality is QuickSight Q. QuickSight Q allows the user to ask any question related to the associated data topic in plain English and receive a KPI/ visualization in return. This is ideal for a user that has a specific question that needs to be answered and only wants one visualization/KPI to quickly export out of QuickSight.

# Architecture



# Modeling/Inference Pipeline

The modeling pipeline is still open-ended, and multiple routes can be taken:

1. **SageMaker Data Wrangler to SageMaker Pipelines** – this is a viable option because the Data Wrangler flow is already completed (with mock data). Once real data access is gained, steps for creating a Data Wrangler flow include the following:
   1. Import the correct dataset from Athena
   2. Feature engineer the target column (employee interview acceptance with a binary yes or no attribute) until we can pull this from real data
   3. Drop any post-interview columns (Voop data), and one-hot encode all of the categorical data
   4. Create an executable script from the Data Wrangler flow and integrate that with SageMaker pipelines as the pre-processing step.

Using Python SDK, one script can be written for the training step and one script for the inference step which would both be connected to the pre-processing step. These steps can all be displayed as nodes by using SageMaker Pipeline Projects. A batch transform job is recommended to be ran within the inference script so that users can fetch inferences ad-hoc. This saves AWS extra and unneeded costs associated with real-time predictions.

Figure 1: An image of a Data Wrangler flow:

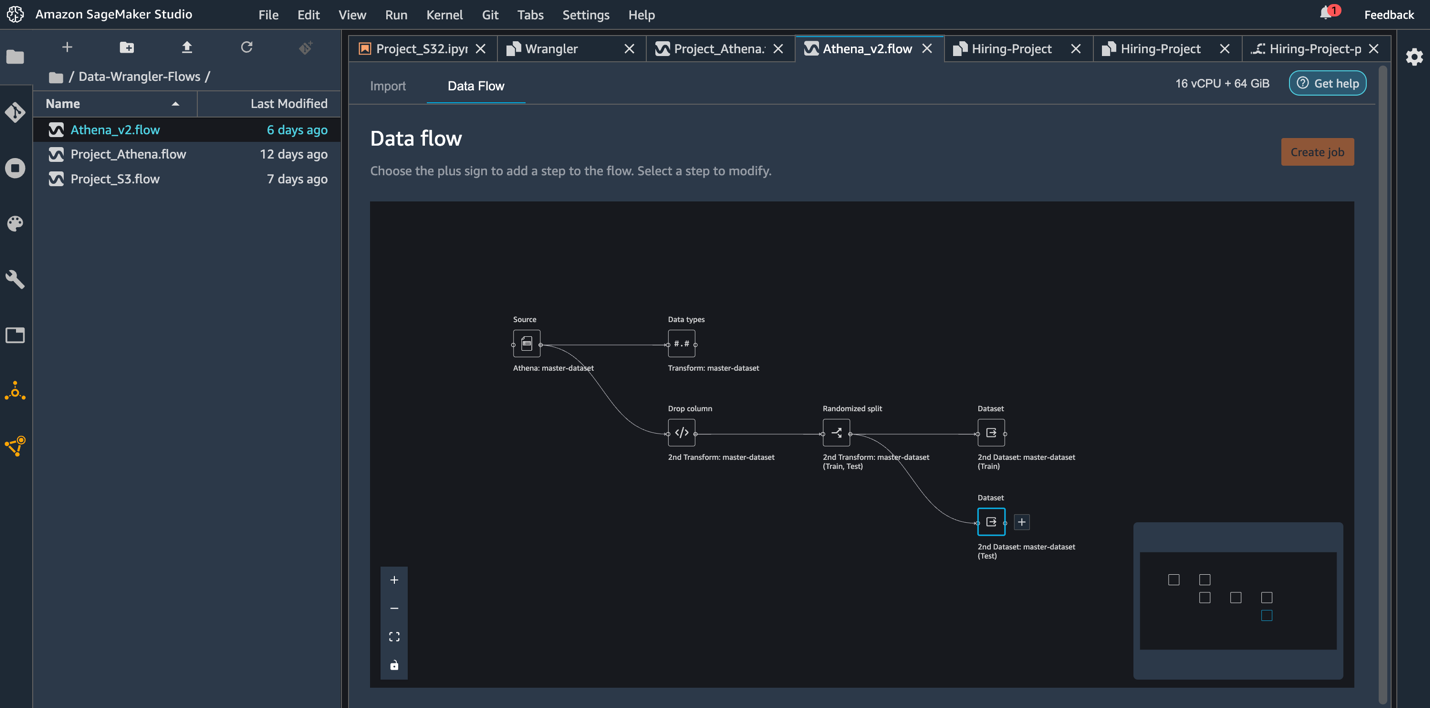
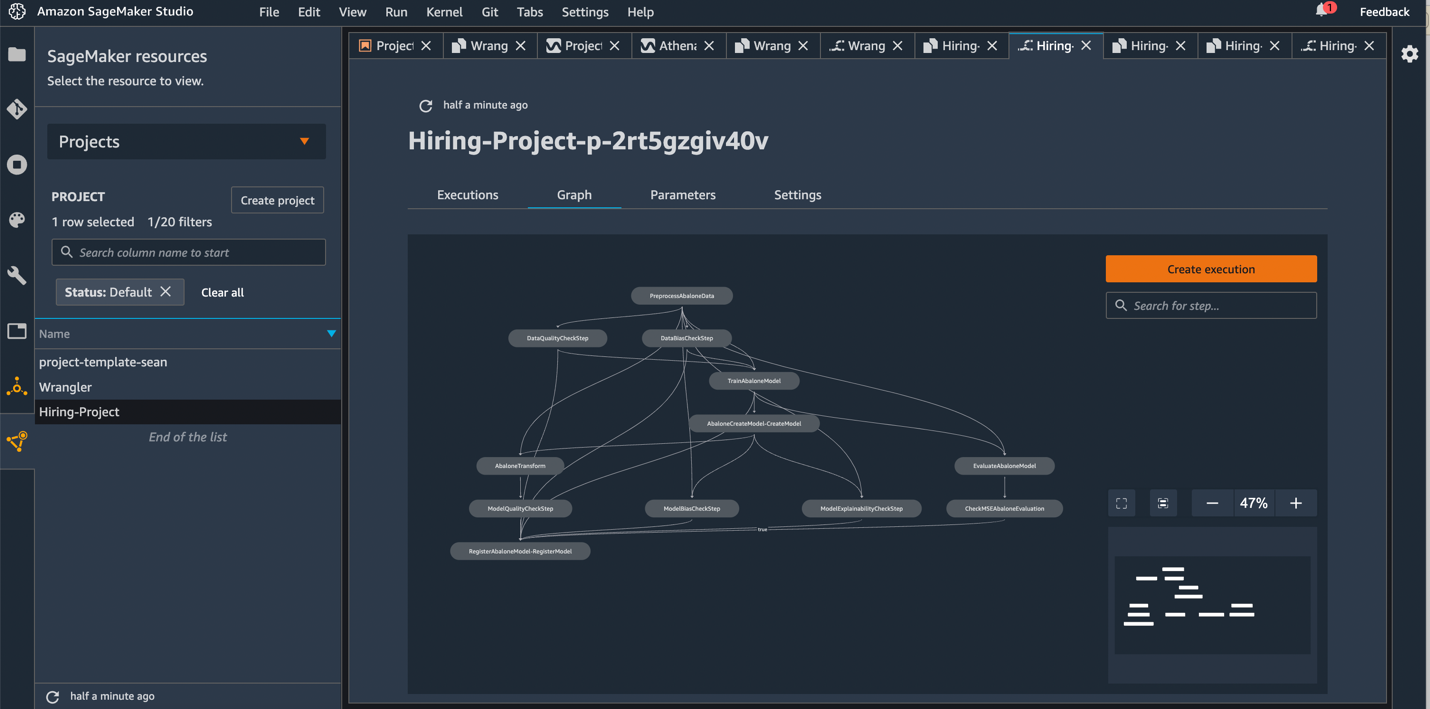


Figure 2: An image of a SageMaker Pipelines flow:



**Pros**:

* Data insights from Data Wrangler are very detailed and can provide so much more clarity on the data over a traditional model ran in Jupyter Notebook. Examples of useful insights that are auto-generated are the prediction power of each feature of the model, a confusion matrix, and an ROC curve.
* SageMaker Pipelines can be executed with one click, with the directed acyclic graph (DAG) graph showing exactly where the model goes wrong if it does fail.
* Best option for scalability.

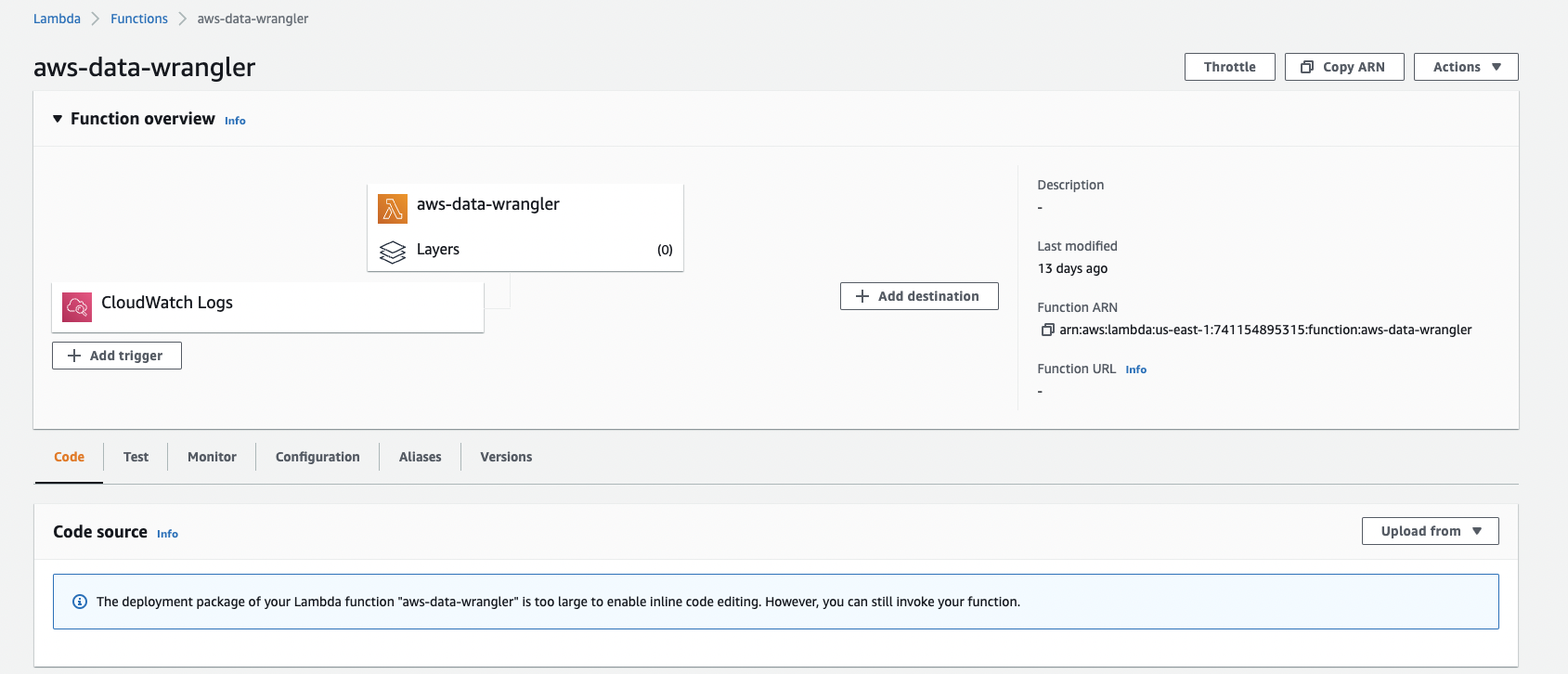
**Cons**:

* AutoPilot is harder to troubleshoot than a model written by the user. If the model does fail, it is hard to figure out what to fix.
* AutoPilot model currently is outputting incorrect values in the batch transform job.

**Automation**:

If the Data Wrangler approach is chosen, I have written a Lambda function to trigger the Data Wrangler flow based on when the CloudWatch logs deem the data lake workstream Glue job successful. The code is packaged in a zip file and can be viewed in the GitLab repository shared with the team.

Figure 3: An image of the ‘aws-data-wrangler’ Lambda function shown below:



1. **SageMaker Notebooks running in Notebook Instance** – this is the current option that is powering the QuickSight prediction dashboard. Three notebooks (pre-processing, training, and deployment) are in one notebook instance in SageMaker. These notebooks were converted into executable scripts. Next steps include exporting these scripts into individual Docker containers, deploying the Docker containers to ECR, and having a Lambda function for each container that are triggered together by a Step Function.

**Pros**:

* This process is not new and has been done before. With the current setup of the model, this would not be hard to implement, but would take about a month.
* Model is working well in these notebooks. No troubleshooting needed.

**Cons**:

* Lambda functions are not as scalable of an option as doing all model pipelining in SageMaker.
* Loss of data insights generated in Data Wrangler.

**Final Recommendation**: To account for scalability, choosing Data Wrangler and SageMaker Pipelines would be the best option. Not only is this scalable, but the addition of the data insights report provides high business value to stakeholders. Stakeholders would be able to see exactly what is impacting employee interview acceptance rate the most and how this impact varies by employee level or role as an example.

**Backlog**: Choosing which model pipeline will be the best option for this scalable use-case

# SageMaker Model Specifics

The logistic regression model is split into three notebook steps (Pre-processing, training, and deployment) under the Prediction-model notebook instance in the SageMaker console. All three steps will be described in depth below:

1. **Pre-Processing** – After importing all necessary libraries we first define a function to pull the most recent Glue job completed from the data lake. This provides a master dataset with past data and the addition of the newest data. The csv is read using pandas and then feature engineers the “Accepted\_or\_Not?” column (Please note that feature engineering this column is temporary; “Accepted\_or\_Not?” will be a column that we receive from hiring data so this step will not be needed once access is gained). Next step is to select only the columns we need for the model (pre-interview data) and to one-hot encode all categorical data. This 101-feature dataset is exported into S3 to be used in the training step.
2. **Training** – The training step imports the encoded data we created from the pre-processing step. Next, we define our training and testing data. My input data (x\_train and x\_test sets) are the whole dataset with the target column (Accepted\_or\_Not?) dropped. My output data (y\_train and y\_test sets) are only the target column.

**Note**: For now, splitting the data is not ideal. We lose out on the training data which is the majority, to be displayed in QuickSight. Instead, we are training our model on a static dataset that contains the first iteration of the data lake. As we get more data, we can train the model as needed, but without the split we can display predictions for all employees. If a train test split is needed, I have commented out the code so it is still usable.

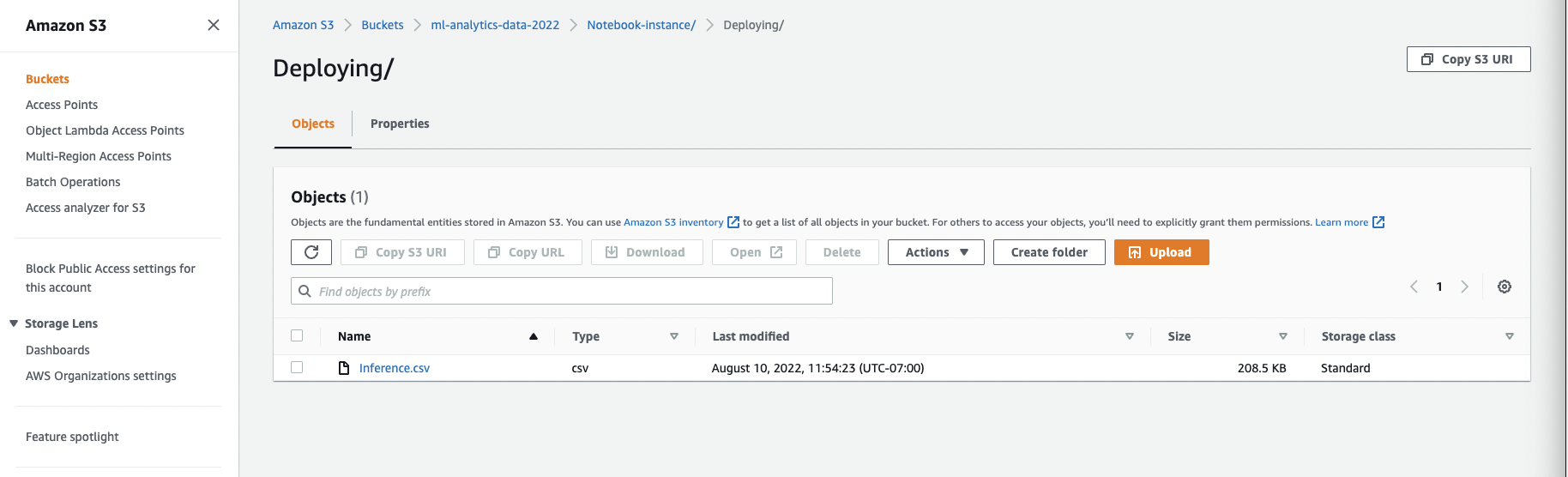
Now, we define and train our logistic regression model using sklearn.linear\_model and then find the accuracy of the model. Using joblib we can save our model, which is crucial for lowering cost. This allows us to skip the training step in the future and perform predictions on a saved model. This model is saved as ‘Inference\_log\_reg.joblib’. We will only need to perform the training step if we want to update the trained model.

1. **Deployment** – The deployment step starts by running the max\_partition function. This pulls the most recent iteration of the data lake to be use for adding prediction and probabilities after inference. Once we read the csv using pandas we can feature engineer our target column (this is temporary). Then, we select the columns that are needed for inference, excluding all post-interview columns (this dataset is solely used to add the predictions that are made on the encoded data to data that is not encoded). Next, we read in our pre-processed dataset used for testing. We define our inputs and outputs and load our saved model using dump and load from joblib. We make predictions on the test set, output a confusion matrix, and display the test accuracy. We then list the prediction probabilities and add the prediction (yes or no) along with its associated probability to the previous dataset that is not encoded. This gives us the dataset that will be used in QuickSight. We export this data to an S3 path that will be ran through an inference pipeline to get to QuickSight.

# Current Inference Pipeline

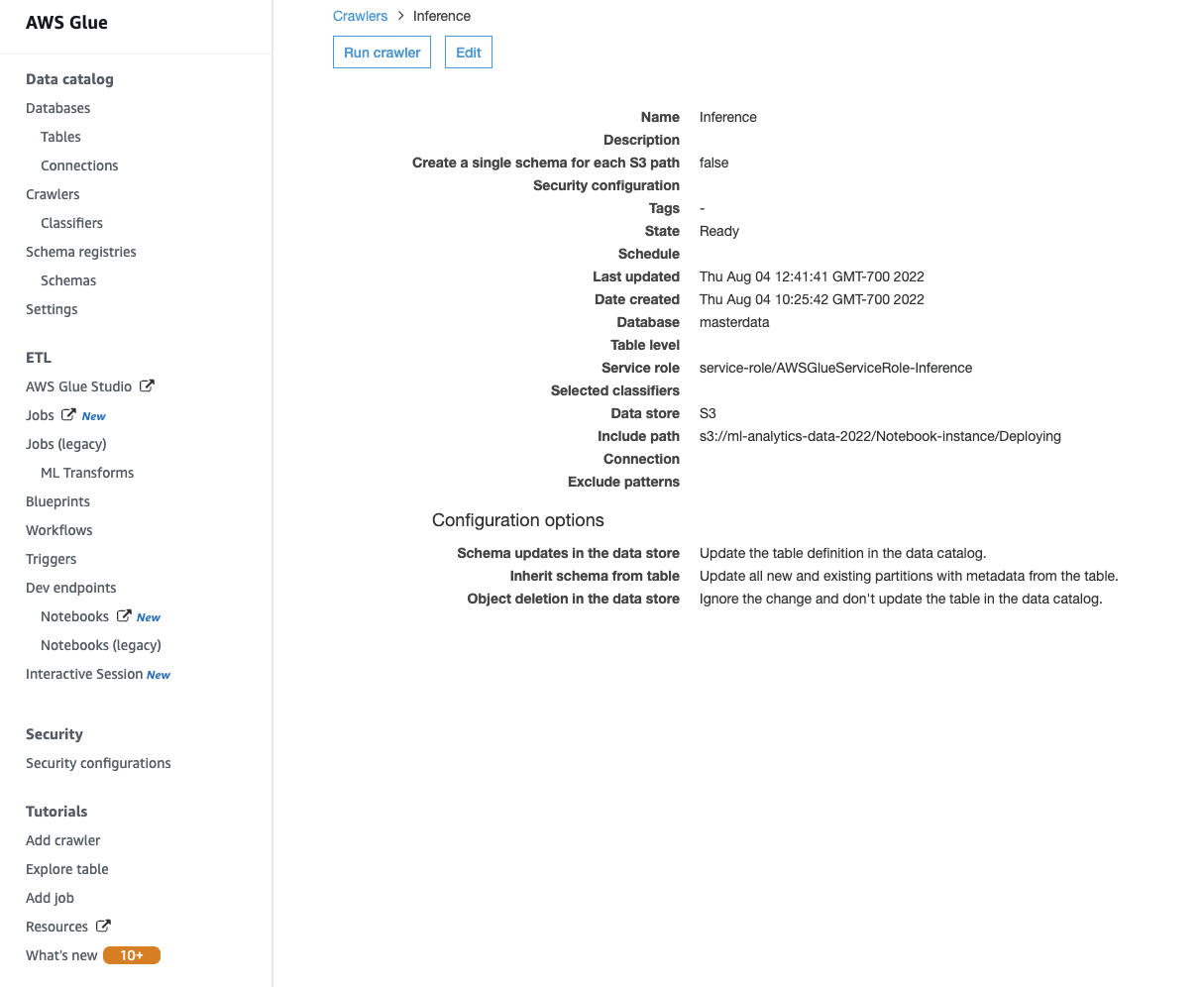
After the model is deployed, the data is exported to this S3 path: ml-analytics-2022/Notebook-Instance/Deploying/Inference.csv.

Figure 4: S3 path for inference data:



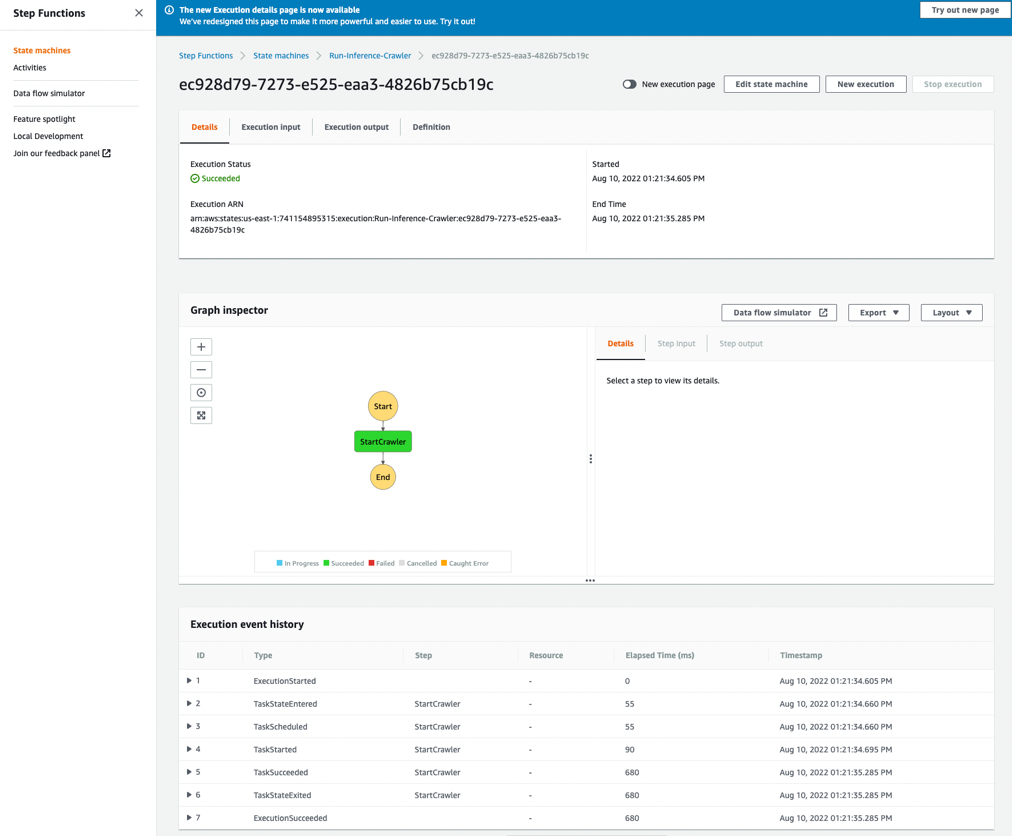
This path is then read into AWS Glue as a crawler. The purpose of using a crawler is to determine the format/schema of the data and to write this metadata to a Glue Data Catalog. In turn, this inference crawler populates the AWS Data Catalog with the table that we see in the SageMaker notebook.

Figure 5: ‘Inference’ crawler setup in AWS Glue:



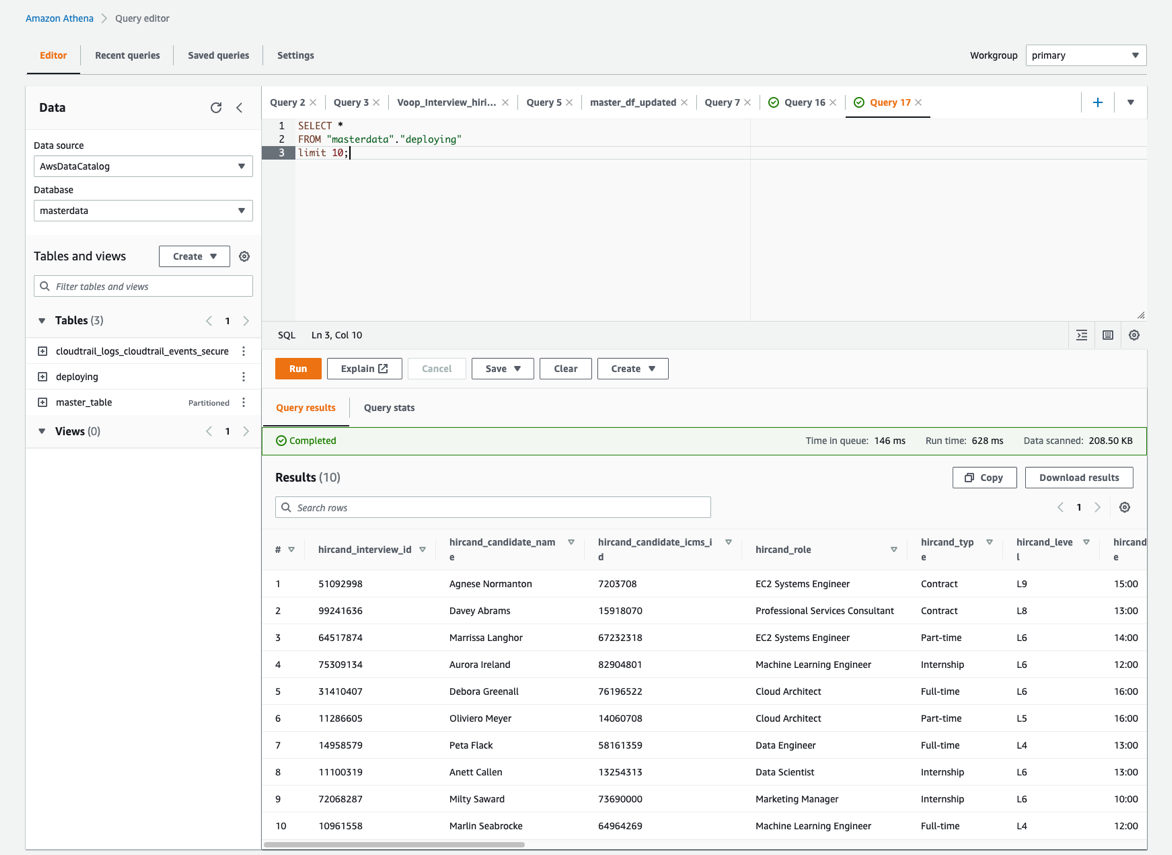
To reproduce a crawler like this, follow the settings shown in the image above with the correct S3 path to a new dataset. Automation of this crawler is taken care of via a Step Function state machine and a 12-hour EventBridge rule.

Figure 6: The step function ‘Run-Inference-Crawler’ triggers the inference crawler to run every 12 hours:



The execution of this inference crawler allows Athena to view this table. Users may make any appropriate queries to the table needed.

Figure 7: Athena showing the ‘deploying’ table from SageMaker due to the connection of the Glue crawler:



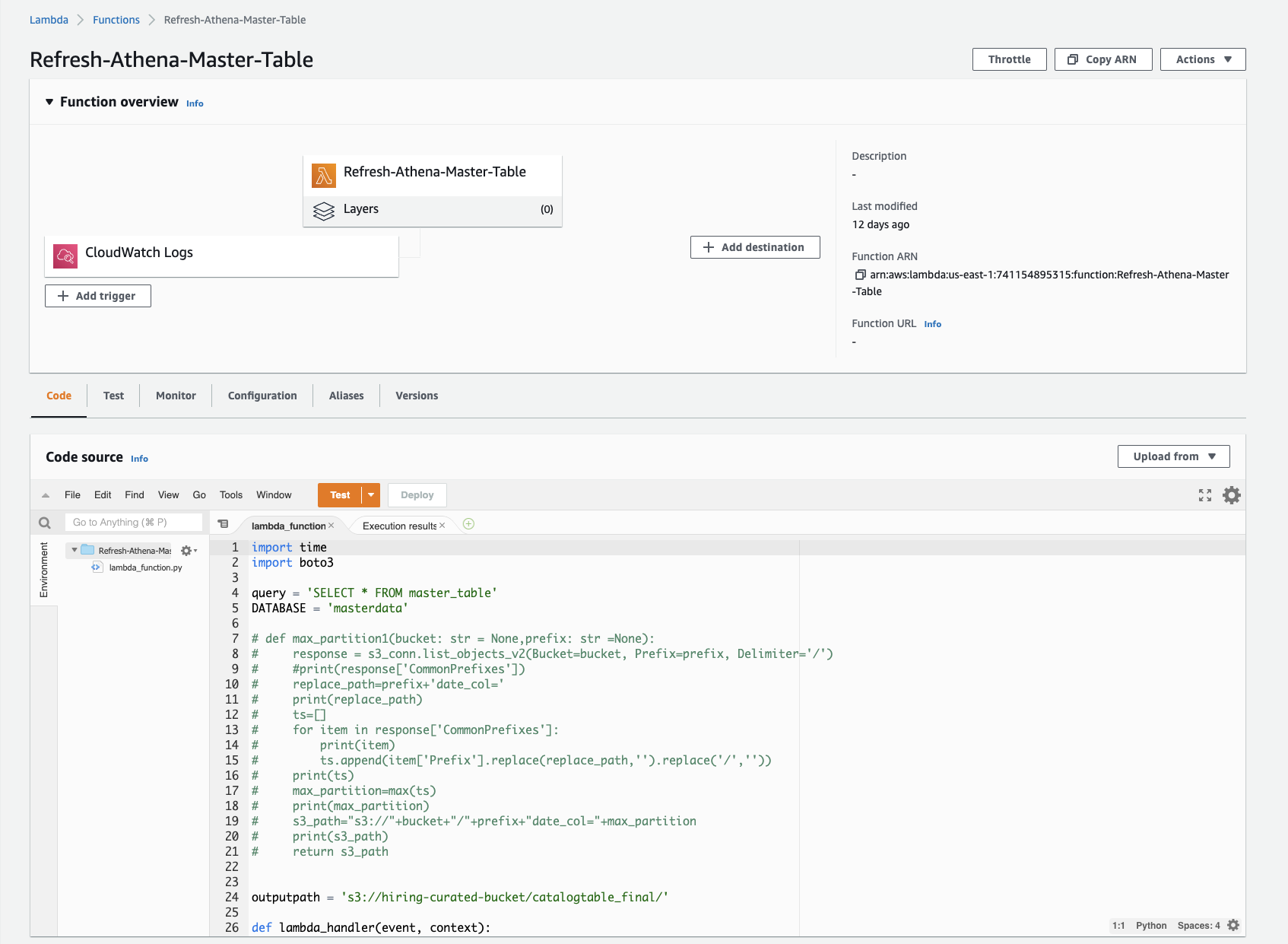
To learn how to import this data into QuickSight, go to the **importing data into SPICE from Athena** within the QuickSight section where there is a step-by-step process on how to do this.

**Backlog**: N/A

# Historical Data Pipeline

Historical data is defined as data that is stitched together in the Glue job, read directly into Athena, and then imported into SPICE in QuickSight. Once a Glue job is deemed successful via CloudTrail logs, the ‘Refresh-Athena-Master-Table’ Lambda function is triggered that executes a query of the additional data in Athena.

Figure 8: The Refresh-Athena-Master-Table’ Lambda function:



With the QuickSight refresh set to daily, the historical trends dashboard in QuickSight will constantly be updated, meaning this historical data pipeline is self-sustaining.

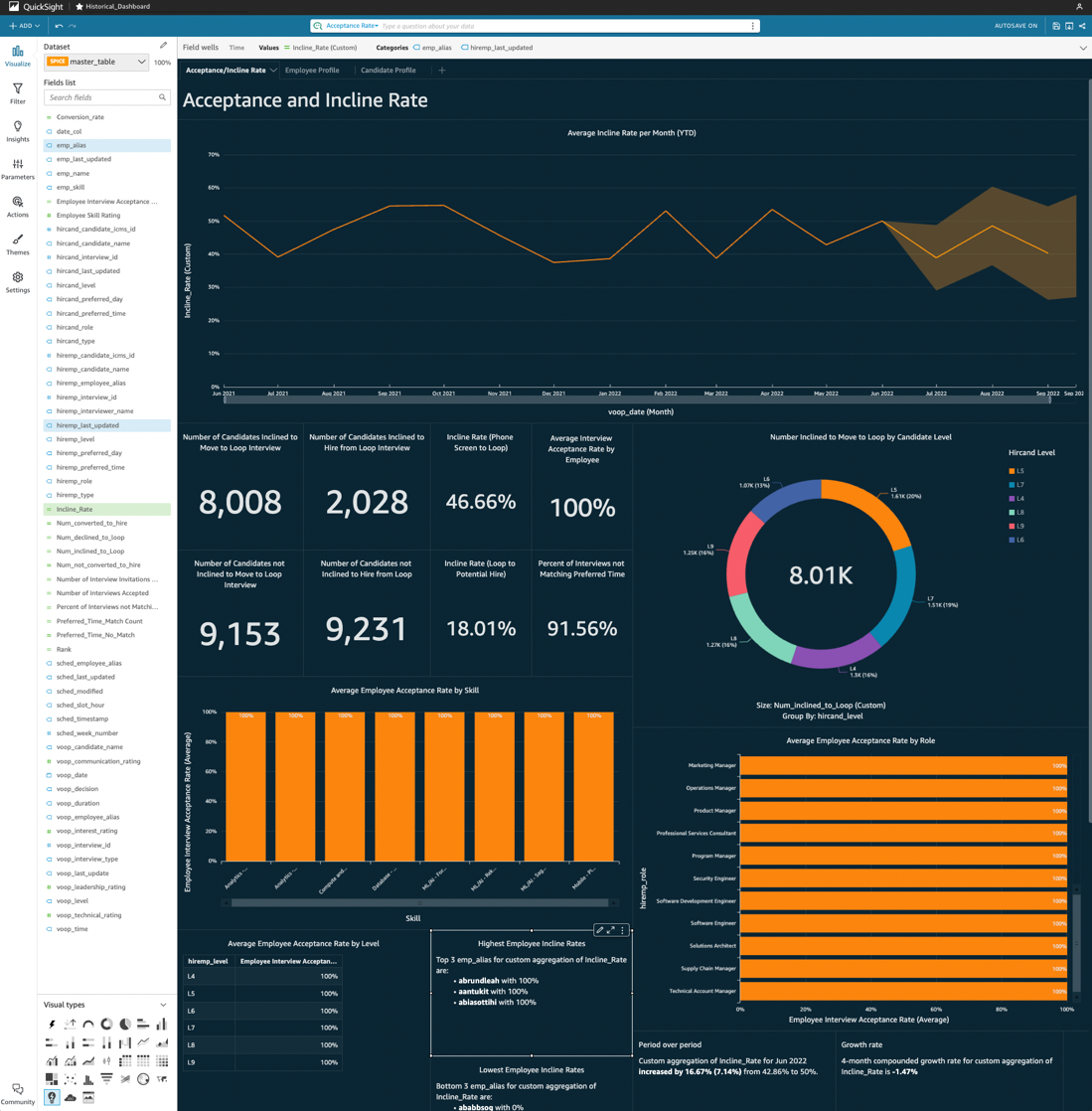
**Backlog**: N/A

# QuickSight

QuickSight provides and easy-to-use dashboard to identify hiring trends that the hiring team can use to make adjustments accordingly. This dashboard empowers us to find outliers in the data and figure out why these outliers are occurring.

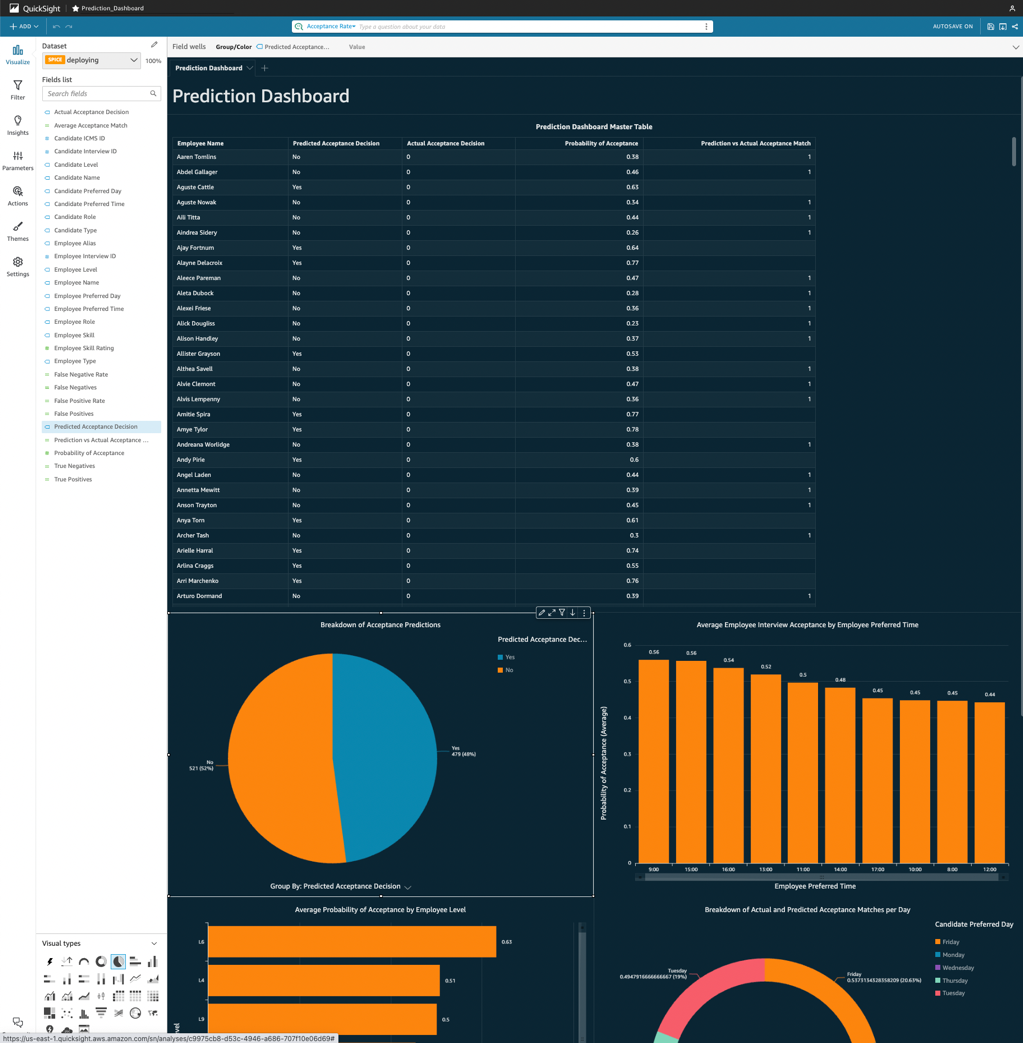
1. **QuickSight Historical Trends Dashboard** – this dashboard provides us with an analysis of all the data from the data lake. This dashboard is separated into three different sheets. The Acceptance/Incline Rate sheet focuses on how these rates compare to role, level, and skill. The two other sheets, Employee Profile and Candidate Profile, show us more specifically who our interviewers and interviewees are. This dashboard currently uses the mock data I created via Mockaroo that was pumped through the historical data pipeline. In QuickSight, this dataset is ‘master\_table’.

Figure 9: Image of the historical trend dashboard:



1. **QuickSight Prediction Dashboard** – this dashboard provides us with an analysis from the data outputted from the logistic regression model. Here you can find a specific employee’s probability of accepting an interview with its predicted acceptance and actual acceptance. We also compare these statistics with an employee’s role, level, skill, etc, to identify model-based trends. This dashboard currently uses the mock data I created via Mockaroo that was pumped through the inference pipeline. In QuickSight, this dataset is called ‘deploying’.

Figure 10: Image of the prediction dashboard:



1. **QuickSight Q** – QuickSight Q allows the user to ask any question in plain English on topic that contains specialized data and expect to receive a visualization or KPI in return. There is a bar at the top of the screen to use this feature where you can choose the topic and then type the question you need an answer to. A ‘topic’ allows the user to focus on subsets of the dataset for questions. This makes the natural language processing more effective as it is more likely to answer your question correctly if it is only including a fraction of the columns that encompass that subset of the dataset. If a question is answered incorrectly, feedback can be left for the administrator to improve its answering capabilities. QuickSight Q relies heavily on synonyms, words that an administrator can enter for any column of data that might be asked in a question. For example, a synonym that could be useful for the column “Employee Name” would be “Interviewer Name”. As time goes on, QuickSight Q will become a robust feature because of the addition of synonyms.

**Importing data into SPICE from Athena**: Using the current architecture, all data will be coming from Athena. To import the real data that will be available in Athena, follow these steps once logged into QuickSight:

1. Click on ‘Datasets’ on the left-hand side of your screen.
2. Click on ‘New Dataset’ on the top-right corner of your screen.
3. Choose Athena as data source and enter an appropriate data source name.
4. Identify which database your and data catalog your dataset is in. Once you select a database, a list of tables from Athena will appear: select one and then click ‘Select’.
5. Click ‘Import to SPICE for quicker analytics’ and then ‘Edit/Preview data’.
6. Make any necessary edits to column names or column data types and then select ‘Save and Publish’.
7. Congrats! You now have a functioning dataset in QuickSight that can be used to create an analysis.

**Creating an analysis**:

1. Click on ‘Datasets’ on the left-hand side of your screen.
2. Select the dataset you created.
3. Click on ‘Create analysis’.
4. Here you can add visuals and KPIs to create a dashboard!

**Setting your dataset to refresh:**

Setting the dataset to refresh daily is essential to having dashboards that update automatically. Here is how to do it:

1. Click on your dataset of choice in the ‘Datasets’ tab.
2. Click ‘Schedule a refresh’.
3. Select daily as an option.

**Navigating QuickSight Q:**

1. Click on ‘Topics’ on the left-hand side of your screen.
2. Select ‘New Topic’ to create a topic based on a subset of the dataset.
3. ‘Summary’ shows the feedback statistics over time
4. Click on ‘Data’
5. Exclude any columns that would not be utilized when asking a question about this topic.
6. Convert column names to friendly names that natural language processing will identify easier
7. Add synonyms if certain column names are common to improve Q performance.
8. Add calculated fields that are topic-specific (ex: I added only acceptance rate calculated fields for the acceptance rate topic).
9. Add a filter if certain attributes within a column need to be explicitly included or excluded.
10. Click on ‘User Activity’ – this shows us the percentage of marked positive or negative questions by users depending on if they were answered correctly

**Augmenting QuickSight dashboard with SageMaker AutoPilot model (OPTIONAL)**:

If an AutoPilot model is preferred to be used, augmenting with SageMaker is a possible way to connect our predictions from our SageMaker model to our data in QuickSight. This route led to multiple errors, but documenting this will provide do’s and do not’s moving forward.

What is needed:

* A working AutoPilot model
* A JSON schema file that contains metadata to process the model
* A test dataset with the target column that matches the model schema

What went wrong:

* My model’s batch transform job done through QuickSight produced an error file

Fixing this would allow you to use AutoPilot instead of the manual model. This is another option for the future if utilizing AutoPilot’s functionality for a model is desired.

**Backlog**:

N/A

# Integration with Data Lake

The Reporting/Analytics workflow connects with the data lake workflow through the successful completion of the Glue job. The Glue job stitches together all of the data from various data sources and removes the necessity for the Reporting/Analytics workflow to do any joins. Within S3, you can visit the hiring-curated bucket and view all of the successful Glue jobs that have been run. The most recent file will have the latest updated data, along with all of the data from previous Glue job stitching. Our prediction model takes the most recent file and uses that for inferences. Our historical data pipeline will grow to immense amounts as time goes on since new data will be appended onto existing data.

# How I Mocked the Data

In order for the UI, data lake, and ML/analytics workstreams to start working they all needed data. Unfortunately, data access was never granted. As a result, I initiated an effort to mock the data. This took place using a third-party tool called Mockaroo. Schemas of the real data sources are nearly identical with attributes emulating what real data would look like. All of the datasets mocked with associated column names are the following:

* **Employee\_Skills** – from Salesforce showcasing our employee data
  + Alias, Name, Skill, Skill Rating, Last Updated
* **Interviewee\_Hiring** – from Hiring showcasing candidate data
  + Interview ID, Candidate Name, Candidate ICMS\_ID, Role, Type, Level, Preferred Time, Preferred Day, Last Updated
* **Interviewer\_Hiring** – from Hiring showcasing employee/interviewer data
  + Interview ID, Interviewer Name, Candidate Name, Candidate ICMS\_ID, Role, Type, Level, Preferred Time, Preferred Day, Last Updated
* **Scheduling** – from front-end showcasing employee schedule preferences
  + Employee\_Alias, Timestamp, Week #, Slot (Hour), Modified, Last Updated
* **Voop\_Data** – from Voop showcasing post-interview data
  + Interview ID, Employee\_Alias, Date, Time, Candidate Name, Decision, Duration, Level, Interview Type, Communication Rating, Leadership Rating, Technical Rating, Interest Rating, Last Updated

Figure 11:Image of Mockaroo project with all schemas and datasets

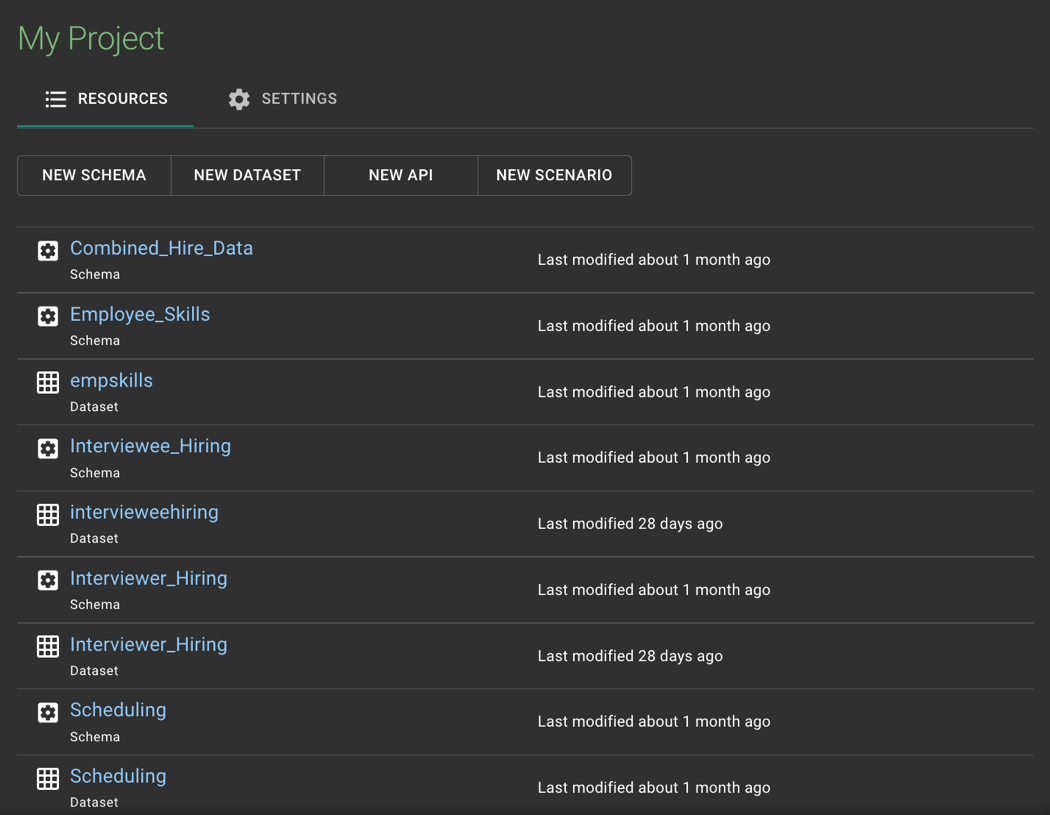
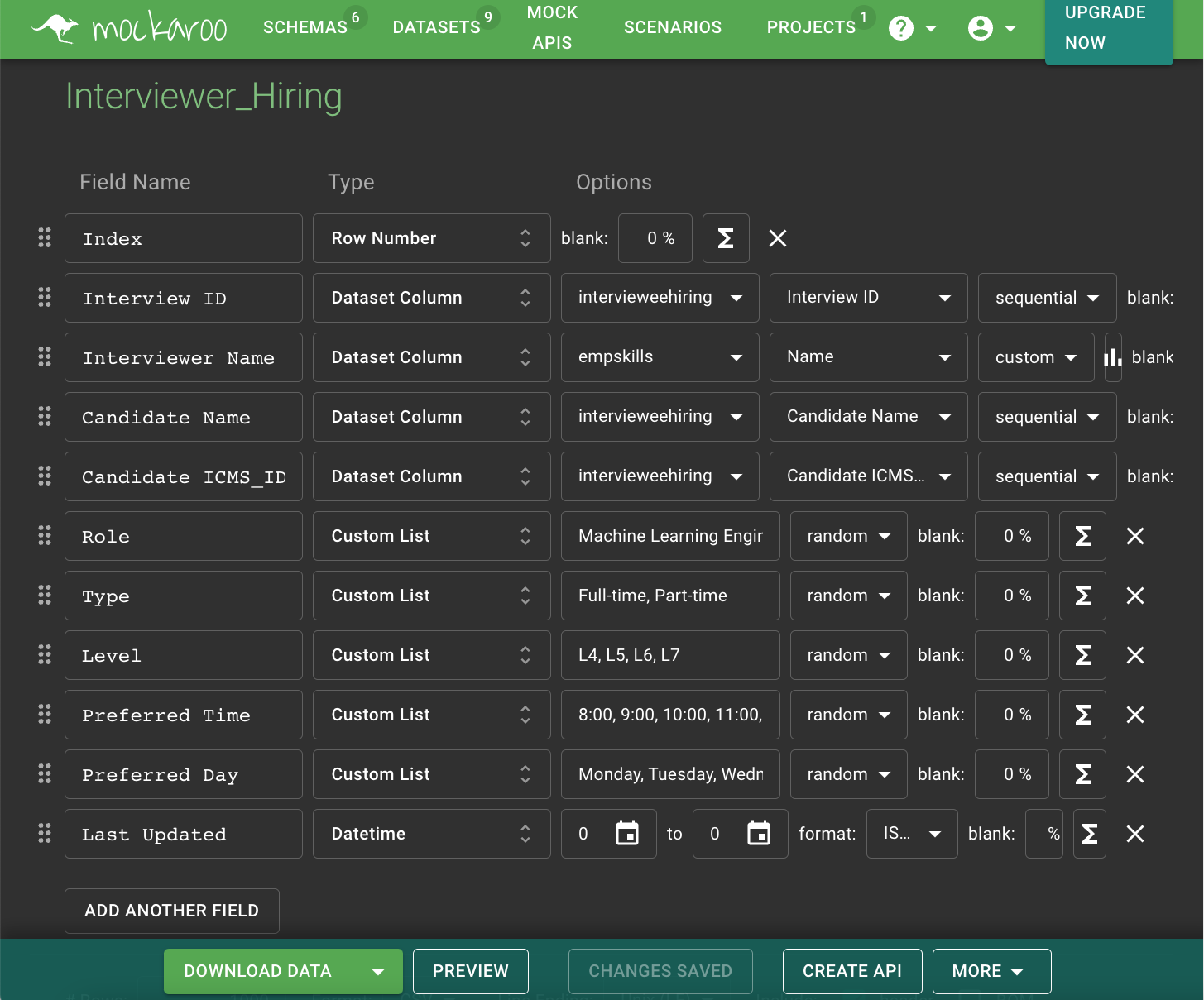


Figure 12: Image of one schema within Mockaroo project



Our five individual datasets with 1000 randomized rows for the project were created with this tool. This enabled the data lake workflow to stitch these datasets together providing the ML/Analytics workflow with one master dataset. If data restriction continues Mockaroo could be a great tool to modify this data as needed until access is granted.

# Outcomes

Here are the outcomes of this workflow that will provide business value for AWS:

* A QuickSight historical trends dashboard that will enable hiring to analyze past hiring trends for acceptance and incline rate in relation to employee and candidate profiles on a granular level.
  + Hiring will have access to multiple time series plots to isolate extremes on the basis of months, years or quarters.
    - Ex: Number of interviews conducted last year by employee role
  + Hiring can group by an employee or candidate’s role, skill, level, preferred day to interview, or preferred time to interview to capture the gaps in our hiring efforts and ultimately improve hiring efficiency.
* A QuickSight prediction dashboard that will enable hiring to showcase the outputs of the logistic regression model in SageMaker.
  + Hiring will be able to improve upon this model as time goes on and as more real data comes in
* A clear pipeline from the central data repository to the QuickSight dashboards
  + The data will be going through the model pipeline and historical trends pipeline to get from point A to point B.
    - Most of these pipelines are automated as long as new data comes in from the data lake.

# Appendix

# List of Useful Links

**SageMaker Studio:**

<https://docs.aws.amazon.com/sagemaker/latest/dg/data-wrangler-data-flow.html>

<https://docs.aws.amazon.com/sagemaker/latest/dg/define-pipeline.html>

<https://docs.aws.amazon.com/sagemaker/latest/dg/run-pipeline.html>

<https://docs.aws.amazon.com/sagemaker/latest/dg/batch-transform.html>

**SageMaker Notebook Instance:**

<https://sagemaker-workshop.com/introduction/notebook.html>

<https://docs.aws.amazon.com/sagemaker/latest/dg/nbi.html>

**Glue:**

<https://docs.aws.amazon.com/glue/latest/dg/crawler-configuration.html>

**Athena:**

<https://docs.aws.amazon.com/athena/latest/ug/data-sources-glue.html>

**QuickSight:**

<https://aws.amazon.com/quicksight/getting-started/>

<https://docs.aws.amazon.com/quicksight/latest/user/spice.html>

<https://docs.aws.amazon.com/quicksight/latest/user/example-create-an-analysis.html>

<https://docs.aws.amazon.com/quicksight/latest/user/sagemaker-integration.html>

<https://www.youtube.com/watch?v=VN_4nuCVzeo>

**QuickSight Q:**

<https://docs.aws.amazon.com/quicksight/latest/user/quicksight-q-get-started.html>

<https://docs.aws.amazon.com/quicksight/latest/user/quicksight-q-ask.html>

<https://docs.aws.amazon.com/quicksight/latest/user/quicksight-q-wrong-answers.html>