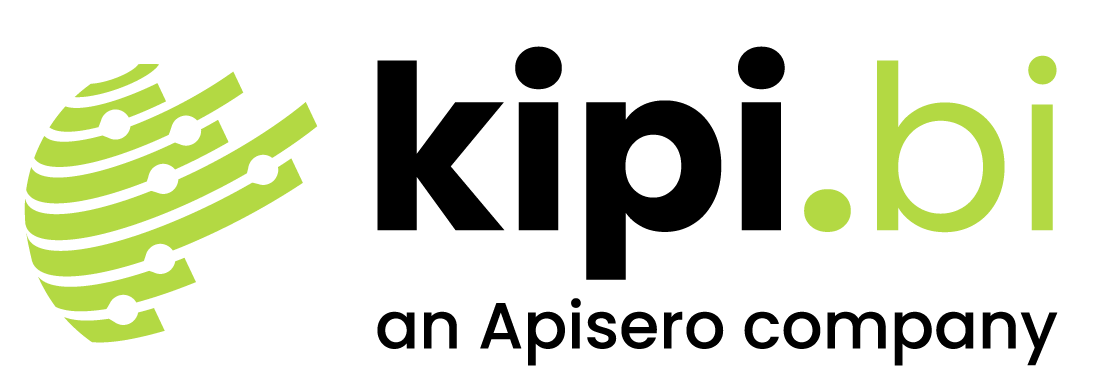
Final Project Documentation for “Trusted Advisor”

Prepared for



|  |  |
| --- | --- |
|  | *Submitted by*  TEAM MEMBERS  Lisa Jia  Yogender Singh  Pranjali Misra  Shweta Pachauri  Bora Sai Prasanna  Pushpender Singh  TEAM CAPTAIN  Akhil Bajpai  Adwitiya Das  PROJECT MENTOR  Manisha Jaiswal |

Table of Contents

[Introduction](#_heading=h.30j0zll) **3**

[Why Trusted Advisor?](#_heading=h.1fob9te) 3

[Who are our end-users?](#_heading=h.3znysh7) 4

[What are our data requirements?](#_heading=h.2et92p0) 5

[Architecture](#_heading=h.tyjcwt) **6**

[Snowflake](#_heading=h.3dy6vkm) 6

[Tableau](#_heading=h.1t3h5sf) 7

[Matillion ETL](#_heading=h.4d34og8) 7

[Dataiku](#_heading=h.2s8eyo1) 8

[Data Model](#_heading=h.17dp8vu) **9**

[**Data Pipeline**](#_heading=h.3rdcrjn) **17**

[Master pipeline for monthly data load to Snowflake Landing](#_heading=h.26in1rg) 17

[Data driven approach for Amazon RDS to Snowflake Landing](#_heading=h.lnxbz9) 18

[Student Attendance Sheet Data into Snowflake](#_heading=h.44sinio) 19

[Student Feedback Data for Mentor](#_heading=h.2jxsxqh) 20

[Master pipeline for yearly data load from Salesforce to Snowflake Landing](#_heading=h.z337ya) 21

[Master pipeline to load Twitter Feed to Snowflake (ad-hoc)](#_heading=h.3j2qqm3) 22

[Snowpipe with error notification](#_heading=h.4i7ojhp) 23

[Master pipeline to load scraped Job requirement details from Ambitionbox.com to Snowflake (ad-hoc)](#_heading=h.2xcytpi) 25

[**Security Features**](#_heading=h.1ci93xb) **27**

[Okta Integration with Snowflake and Tableau](#_heading=h.3whwml4) 27

[Account Login Attempts shared to Account Admin](#_heading=h.qsh70q) 28

[Role-based Access Control using Kipi’s RBAC Accelerator](#_heading=h.3as4poj) 31

[Resource Monitoring](#_heading=h.1pxezwc) **32**

[Performance Monitoring](#_heading=h.49x2ik5) 32

[Cost Monitoring](#_heading=h.2p2csry) 32

[Storage Monitoring](#_heading=h.147n2zr) 33

[**Data Quality Check and Audit Features**](#_heading=h.23ckvvd) **34**

[Type check, Range check and Null check before loading data](#_heading=h.ihv636) 34

[Job Logging](#_heading=h.32hioqz) 35

[Data Masking](#_heading=h.1hmsyys) 36

[Object Tagging](#_heading=h.41mghml) 37

[Data Cataloging and Lineage](#_heading=h.2grqrue) **38**

[Data Sharing](#_heading=h.vx1227) **39**

[Testing](#_heading=h.3fwokq0) **40**

[Manual Unit Testing. Link to the unit testing document is provided below:](#_heading=h.1v1yuxt) **41**

[**Challenges and Accomplishments**](#_heading=h.2u6wntf) **42**

[Challenges](#_heading=h.19c6y18) 42

[Accomplishments](#_heading=h.3tbugp1) 42

[**Future Scope**](#_heading=h.28h4qwu) **43**

# 

# Introduction

The Project “Trusted Advisor” aims to correlate education with employment and translate academic performance to better placement rates. The product aims to enhance the placement rates of colleges/universities, by analyzing the performance of their students and suggesting on how it can be improved. While doing so, it also takes into account the expenses of the college, the skills and expertise of the mentor entrusted with the responsibility to teach the students and gain better insights on how it can impact the student performance and placement.

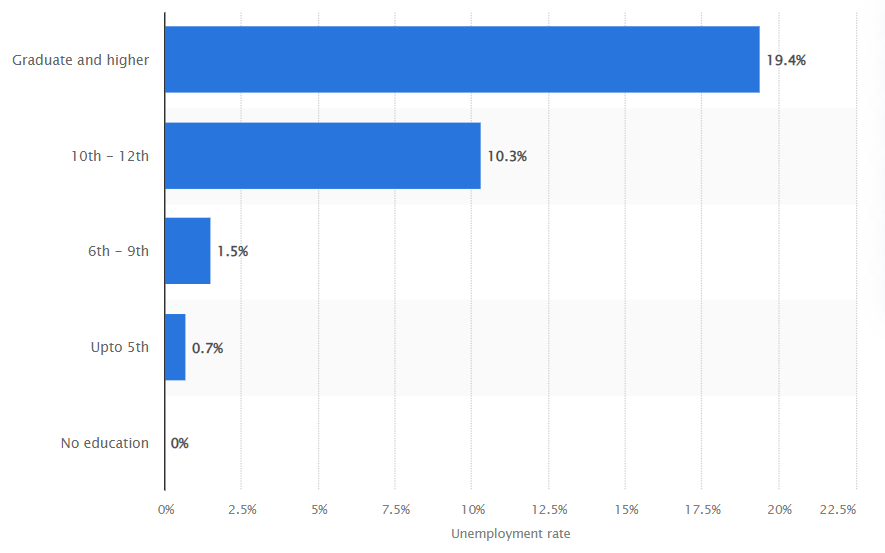
### Why Trusted Advisor?

“The overall unemployment rate in India has increased to 7.83 per cent in April 2022 as compared to 7.60 per cent in the previous month, CMIE said in its report….

In April 2022, Haryana registered the highest unemployment rate at 34.5 per cent, followed by Rajasthan at 28.8 per cent. The unemployment rate in West Bengal during April 2022 has increased to 6.2 per cent as compared to 5.6 per cent in March 2022.”

* From Article named as “[Experts sceptical about CMIE's unemployment data - The Economic Times](https://economictimes.indiatimes.com/news/economy/policy/experts-sceptical-about-cmies-unemployment-data/articleshow/91289669.cms?utm_source=contentofinterest&utm_medium=text&utm_campaign=cppst)”

The following graph shows the unemployment rate for different education qualifications in India:

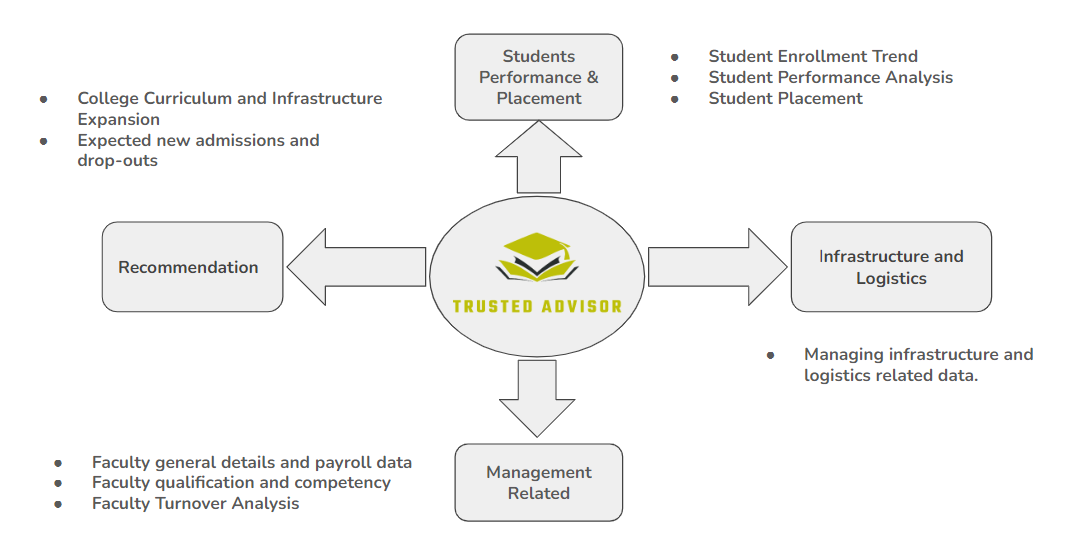
[](https://www.statista.com/statistics/1001039/india-unemployment-rate-by-education-level/)

* From [statista.com](https://www.statista.com/statistics/1001039/india-unemployment-rate-by-education-level/)

The graph above clearly shows how colleges are failing to get their Graduates employed. Trusted Advisor aims to help colleges bridge the gap between education and employment.

The main objective of Trusted Advisor is to provide a holistic approach to improve the performance of a college by focusing on the following four factors:

1. Analyze the student performance, placement rate, enrollment trend.
2. Analyze the infrastructure and logistics expenses of the college to help the college keep a track of the amount spent and if that is considered as a good investment.
3. Analyze the details about the mentor, such as the education qualification, payroll, designation, attrition etc. and derive insights about how these impact the student performance.
4. Recommend ways to expand the curriculum and infrastructure by analyzing the factors above.



### Who are our end-users?

Our product is focused on improving the performance of colleges, and hence College Administrator group is our key Market Segment.

### What are our data requirements?

To be able to produce meaningful insights about the college’s performance and recommend measures to expand and improve it, we need the following datasets:

1. Data from the college regarding:

* Student’s general data such as Student’s Name, Age, Gender, Location, Email, Whatsapp Contact No. etc.
* Student’s academic data such as tenth & twelveth standard score, department in college they belong to, semester scores and subjects they are weak in, internship records, attendance records, placement records etc.
* Mentor general and employment related data such as name, age, years of experience, qualification, designation, department in college they work in, their salary etc.
* Student and Mentor mapping data, and feedback from students about their mentors.

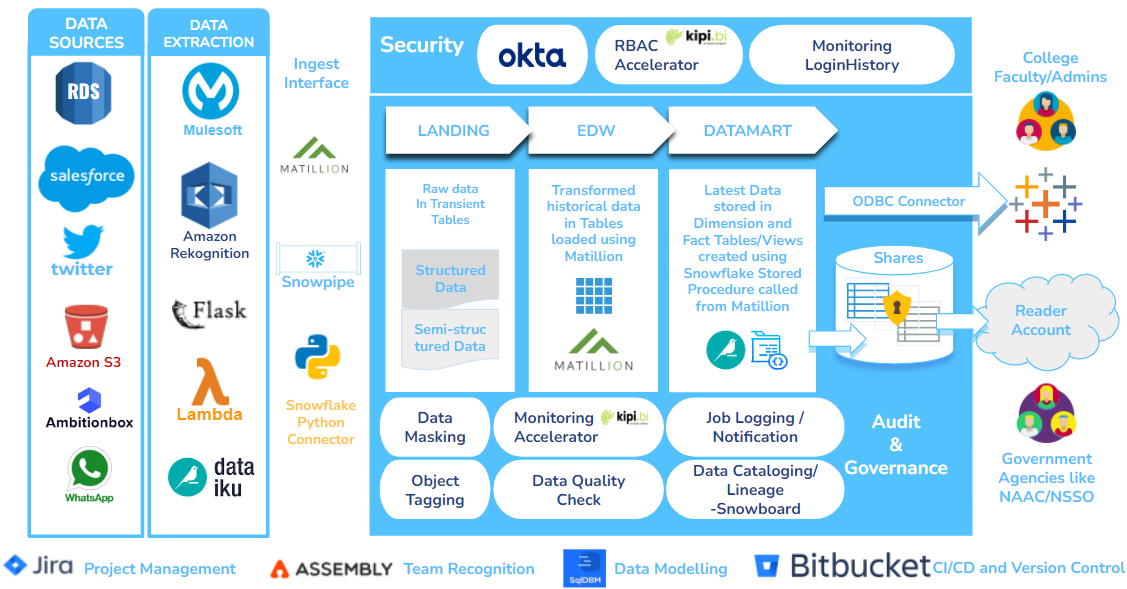
1. Data from Twitter Api to analyze trends in various technologies and suggest a few of them to colleges, so that they can start extra courses and programs to keep their students updated in these.
2. Data from Job sites like Ambitionbox, to get skills required in entry level jobs for different companies that come for campus placements to our partner colleges.

# Architecture

The architecture stack of the product includes 5 main products:

1. Snowflake as a cloud data-warehouse platform
2. Tableau as a BI tool
3. Matillion as an ETL tool
4. Dataiku as a Data Science tool
5. Amazon Web Services (AWS) as a cloud platform

The following diagram represents how the tools have been used and what is the flow of data from left to right, starting from data sources to data consumers:



Let’s quickly have a look at the main tools at a glance:

### Snowflake

Snowflake Inc. is a cloud computing-based data warehousing company . The firm offers a cloud-based data storage and analytics service, generally termed "data warehouse-as-a-service". It allows corporate users to store and analyze data using cloud-based hardware and software. You can sign up for a free trial using the self-service form (on the Snowflake website). When you sign up for a trial account, you select your cloud platform, region, and Snowflake Edition, which determines the number of free credits you receive and the features you can use during the trial. We are using Snowflake Enterprise edition in AWS Asia Pacific Singapore Region. According to [Snowflake documentation](https://docs.snowflake.com/en/user-guide/admin-trial-account.html#:~:text=The%20trial%20continues%20for%2030,trial%2C%20the%20account%20is%20suspended.), the trial period lasts for 30 days and offers 300$ worth of free credits. Snowflake offers an array of out-of-the-box features, we picked and used a few of them as follows:

* Temporary, transient and permanent tables.
* Views, materialized and non-materialized, depending on the data modification frequency.
* Api and storage integrations.
* AWS external stages.
* Snowpipe and Snowflake python connector for data ingestion
* Snowflake ODBC connector to Tableau
* Snowflake Login with Okta SSO
* Snowflake Stored Procedures with Javascript executed with scheduled tasks
* Snowflake External Functions
* Snowflake dynamic data masking and object tagging

### Tableau

Tableau has been used to analyze data and represent it in the form of dashboards. The tableau dashboards are hosted on Tableau Online that comes with a 14 day trial period, with all the features of tableau available. The dashboards have been prepared with kipi’s Row-level security (RLS) Accelerator and workbook permissions set for each user group.

### Matillion **ETL**

Matillion has been used exclusively for data extraction, transformation and loading. It offers a 14 day trial period with limited features. Also, Matillion Account can only be created using an official email id or via Snowflake Partner connect.

Since the project duration is of 7 weeks, of which Build & Test phase starts from 2nd week and Evaluation is from the 3rd week to the 7th week, we created one Matillion account in the 3rd week which lasted till 4th week, and again the second trial account at the end of the 5th week, which lasted till the end of the project. For each Matillion Account, we created one Snowflake account to connect to the Matillion Account from Partner Portal.

According to [Matillion Website](https://www.matillion.com/technology/cloud-data-warehouse/snowflake/):

*“Snowflake architecture helps you build out an entire data analytics platform that takes full advantage of the power and economics of the cloud by providing the speed, performance, and scalability required to handle the exponential growth in data. Matillion ETL for Snowflake helps you get there by making it easy to load all your data into Snowflake and transform it to make it analytics-ready in no time.”*

Also,

* Matillion integrates with Snowflake-specific functionality and best practices
* Matillion provides consumption-based pricing similar to Snowflake

Though the trial version comes with limited features, we made the best use of them. Some of the components used are:

|  |  |  |
| --- | --- | --- |
| 1. Query Result To Grid | 1. S3 Unload | 1. Table Input |
| 1. Grid Iterator | 1. Send Email | 1. Truncate Table |
| 1. SQL Script | 1. Detect Changes | 1. RDS Query |
| 1. Run Orchestration | 1. Filter | 1. Salesforce Query |
| 1. Run Transformation | 1. Distinct | 1. API Query |
| 1. Query Result To Scalar | 1. Create View | 1. Orchestration |
| 1. If | 1. Table Output | 1. Transformation |

### Dataiku

Dataiku has been used as a data science/machine learning tool in our project, to run python scripts. The trial account for Dataiku lasts for 14 days and comes with limitations. The jobs created in dataiku have been hosted on the dataiku cloud.

1. Amazon **Web Services**

Amazon Web Services, Inc. (AWS) is a subsidiary of [Amazon](https://en.wikipedia.org/wiki/Amazon.com) that provides [on-demand](https://en.wikipedia.org/wiki/Software_as_a_service) [cloud computing](https://en.wikipedia.org/wiki/Cloud_computing) [platforms](https://en.wikipedia.org/wiki/Computing_platform) and [APIs](https://en.wikipedia.org/wiki/Application_programming_interface) to individuals, companies, and governments, on a metered pay-as-you-go basis. These cloud computing [web services](https://en.wikipedia.org/wiki/Web_services) provide [distributed computing](https://en.wikipedia.org/wiki/Distributed_computing) processing capacity and software tools via AWS [server farms](https://en.wikipedia.org/wiki/Server_farms). We have used the following services from AWS:

* AWS Simple Storage Service (S3) - as cloud storage service
* AWS RDS - as cloud database
* AWS Lambda - as cloud computing platform
* AWS Rekognition - for image processing

# Data Model

Database

The main development database is called the “TRUSTED\_ADVISOR\_DEV” and the main production environment database is called “TRUSTED\_ADVISOR\_PROD”

Schema

For both these databases, three schemas are present, as per the data load type:

* LANDING schema - This schema plays the role of a “data lake”, where data from all sources are first brought in from various sources, in structured and semi structured format. All tables in this schema are transient. Data in this layer is loaded using various ingestion interfaces.
* EDW schema - This schema acts as an intermediate layer between landing and datamart schemas, where data from landing schema is checked for quality and then transformed to load into datamart. This schema contains historical and latest data.
* DATAMART schema - This schema contains the latest data loaded from edw schema. It is also the schema where all the views for reporting are present.

Tables

Given below is a list of tables created for different schemas:

*For Landing Schema:*

| **Table Name** | **Source** | **Loading Process** | **Load Frequency** | **Description** |
| --- | --- | --- | --- | --- |
| STG\_MENTOR | RDS.STG\_MENTOR | Matillion Job | Monthly | Contains general data about mentors, like Id, Name, Designation etc. |
| STG\_STUDENT | RDS.STG\_STUDENT | Matillion Job | Monthly | Contains general data about students like Id, Name, Address, College Department. |
| STG\_STUDENT\_EVENT\_ATTENDANCE | AWS S3 bucket containing event attendance data of student scanned from attendance sheet images | Snowpipe | Monthly | Contains data about students' attendance in various job-skill training events |
| STG\_STUDENT\_INTERNSHIP | RDS.STG\_STUDENT\_INTERNSHIP | Matillion Job | Monthly | Contains students' internship records |
| STG\_STUDENT\_MENTOR\_MAPPING | RDS.STG\_STUDENT\_MENTOR\_MAPPING | Matillion Job | Monthly | Contains mapping data between students and mentor |
| STG\_STUDENT\_PLACEMENT | RDS.STG\_STUDENT\_PLACEMENT | Matillion Job | Monthly | Contains students' placement records |
| STG\_STUDENT\_SCORES | RDS.STG\_STUDENT\_SCORES | Matillion Job | Monthly | Contains students' semester scores, attendance and subjects they have failed in |
| STG\_INFRASTRUCTURE\_EXPENSE | Salesforce.STG\_INFRASTRUCTURE\_EXPENSE | Matillion Job | Yearly | Contains data about college expenses on infrastructure development |
| STG\_MENTOR\_PAYROLL | Salesforce.STG\_MENTOR\_PAYROLL | Matillion Job | Yearly | Contains mentor's payroll data |
| STG\_STUDENT\_FEEDBACK | Whatsapp communication with students | Python Connector | Monthly | Contains data about students' feedback scores of their respective mentors |
| STG\_TWITTER\_FEED | AWS S3 bucket containing twitter feed json | Snowpipe | Monthly | Contains twitter feed data gathered from Twitter API |
| MAPPING | NA | Manual Insert | Monthly | Contains static data, which is the mapping between RDS source tables and Landing Schema Target tables, used for metadata driven load of Landing tables |
| API\_LOGS | Matillion ETL API | Matillion Job | Monthly | Contains job logs of Matillion jobs executed used for audit purpose |
| AUTOMATED\_TESTING\_MAPPING | NA | Manual Insert | Monthly | Contains static data, which is the mapping between any source and target, and a query to check the data load between them. |
| AUTOMATED\_TESTING\_RESULT | TRUSTED\_ADVISOR\_DEV.LANDING.AUTOMATED\_TESTING\_MAPPING | Matillion Job | Monthly | Contains results of data load test performed on source-target mappings acquired from TRUSTED\_ADVISOR\_DEV.LANDING.AUTOMATED\_TESTING\_MAPPING |

*For EDW schema:*

| **Table Name** | **Source** | **Loading Process** | **Load Frequency** | **Description** |
| --- | --- | --- | --- | --- |
| INFRASTRUCTURE\_EXPENSE | LANDING.STG\_INFRASTRUCTURE\_EXPENSE | Matillion Job | Yearly | Contains VALID data about college expenses on infrastructure development |
| INFRASTRUCTURE\_EXPENSE\_INVALID | LANDING.STG\_INFRASTRUCTURE\_EXPENSE | Matillion Job | Yearly | Contains INVALID data about college expenses on infrastructure development |
| MENTOR | LANDING.STG\_MENTOR | Matillion Job | Monthly | Contains VALID general data about mentors, like Id, Name, Designation etc. |
| MENTOR\_INVALID | LANDING.STG\_MENTOR | Matillion Job | Monthly | Contains INVALID general data about mentors, like Id, Name, Designation etc. |
| MENTOR\_PAYROLL | LANDING.STG\_MENTOR\_PAYROLL | Matillion Job | Yearly | Contains VALID mentor's payroll data |
| MENTOR\_PAYROLL\_INVALID | LANDING.STG\_MENTOR\_PAYROLL | Matillion Job | Yearly | Contains INVALID mentor's payroll data |
| STG\_TWITTER\_FEED\_MV | LANDING.STG\_TWITTER\_FEED | Matillion Job | Monthly | View that contains twitter feed data gathered from Twitter API |
| STUDENT | LANDING.STG\_STUDENT | Matillion Job | Monthly | Contains VALID general data about students like Id, Name, Address, College Department etc. |
| STUDENT\_INVALID | LANDING.STG\_STUDENT | Matillion Job | Monthly | Contains INVALID general data about students like Id, Name, Address, College Department etc. |
| STUDENT\_EVENT\_ATTENDANCE | LANDING.STG\_EVENT\_ATTENDANCE | Matillion Job | Monthly | Contains VALID data about students' attendance in various job-skill training events |
| STUDENT\_EVENT\_ATTENDANCE\_INVALID | LANDING.STG\_EVENT\_ATTENDANCE | Matillion Job | Monthly | Contains INVALID data about students' attendance in various job-skill training events |
| STUDENT\_FEEDBACK | LANDING.STG\_STUDENT\_FEEDBACK | Matillion Job | Monthly | Contains VALID data about students' feedback scores of their respective mentors |
| STUDENT\_FEEDBACK\_INVALID | LANDING.STG\_STUDENT\_FEEDBACK | Matillion Job | Monthly | Contains INVALID data about students' feedback scores of their respective mentors |
| STUDENT\_INTERNSHIP | LANDING.STG\_STUDENT\_INTERNSHIP | Matillion Job | Monthly | Contains VALID students' internship records |
| STUDENT\_INTERNSHIP\_INVALID | LANDING.STG\_STUDENT\_INTERNSHIP | Matillion Job | Monthly | Contains INVALID students' internship records |
| STUDENT\_MENTOR\_MAPPING | LANDING.STG\_STUDENT\_MENTOR\_MAPPING | Matillion Job | Monthly | Contains VALID mapping data between students and mentor |
| STUDENT\_MENTOR\_MAPPING\_INVALID | LANDING.STG\_STUDENT\_MENTOR\_MAPPING | Matillion Job | Monthly | Contains INVALID mapping data between students and mentor |
| STUDENT\_PLACEMENT | LANDING.STG\_STUDENT\_PLACEMENT | Matillion Job | Monthly | Contains VALID students' placement records |
| STUDENT\_PLACEMENT\_INVALID | LANDING.STG\_STUDENT\_PLACEMENT | Matillion Job | Monthly | Contains INVALID students' placement records |
| STUDENT\_SCORES | LANDING.STG\_STUDENT\_SCORES | Matillion Job | Monthly | Contains VALID students' semester scores, attendance and subjects they have failed in |
| STUDENT\_SCORES\_INVALID | LANDING.STG\_STUDENT\_SCORES | Matillion Job | Monthly | Contains INVALID students' semester scores, attendance and subjects they have failed in |

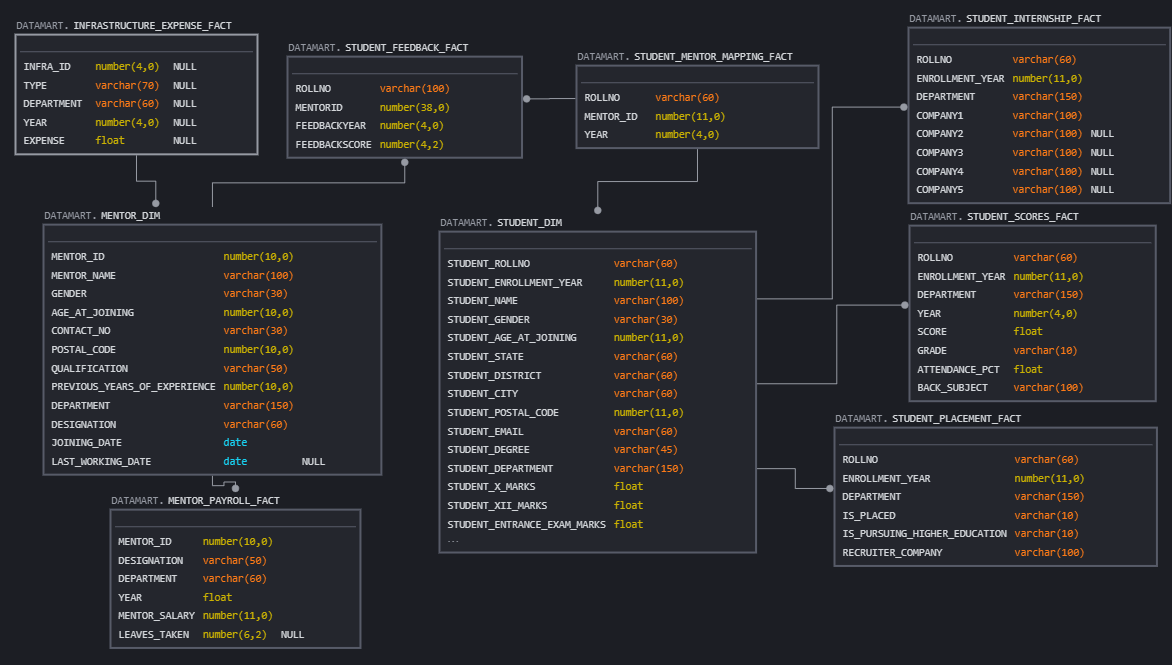
*For DATAMART schema:*

| **Table Name** | **Source** | **Loading Process** | **Load Frequency** | **Description** |
| --- | --- | --- | --- | --- |
| COMPANY\_JOBS\_SKILLS | Web scraped data from Dataiku | Dataiku Job | Ad-hoc | Contains skills requirements for jobs posted by companies on AmbitionBox, which come for campus recruitments |
| INFRASTRUCTURE\_EXPENSE\_FACT | EDW.INFRASTRUCTURE\_EXPENSE | Snowflake SP+Matillion Job | Yearly | Contains latest data about college expenses on infrastructure development |
| MENTOR\_DIM | EDW.MENTOR | Snowflake SP+Matillion Job | Monthly | Contains latest general data about mentors, like Id, Name, Designation etc. |
| MENTOR\_PAYROLL\_FACT | EDW.MENTOR\_PAYROLL | Snowflake SP+Matillion Job | Yearly | Contains latest mentor's payroll data |
| STUDENT\_DIM | EDW.STUDENT | Snowflake SP+Matillion Job | Monthly | Contains latest general data about students like Id, Name, Address, College Department etc. |
| STUDENT\_FEEDBACK\_FACT | EDW.STUDENT\_FEEDBACK | Snowflake SP+Matillion Job | Monthly | Contains latest data about students' feedback scores of their respective mentors |
| STUDENT\_INTERNSHIP\_FACT | EDW.STUDENT\_INTERNSHIP | Snowflake SP+Matillion Job | Monthly | Contains latest students' internship records |
| STUDENT\_MENTOR\_MAPPING\_FACT | EDW.STUDENT\_MENTOR\_MAPPING | Snowflake SP+Matillion Job | Monthly | Contains latest mapping data between students and mentor |
| STUDENT\_PLACEMENT\_FACT | EDW.STUDENT\_PLACEMENT | Snowflake SP+Matillion Job | Monthly | Contains latest students' placement records |
| STUDENT\_SCORES\_FACT | EDW.STUDENT\_SCORES | Snowflake SP+Matillion Job | Monthly | Contains latest students' semester scores, attendance and subjects they have failed in |
| TWEETS\_ANALYSIS\_BY\_KEYWORD | EDW.STG\_TWITTER\_FEED\_MV | Dataiku Job | Monthly | Contains sentiment analysis data on twitter feed |
| STUDENT\_EVENT\_ATTENDANCE\_VW | EDW.STUDENT\_EVENT\_ATTENDANCE | Snowflake SP+Matillion Job | Monthly | Contains latest data about students' attendance in various job-skill training events |

Further details regarding the data mapping can be found the link mentioned below:

<https://docs.google.com/spreadsheets/d/1r8LGTYPotKDbRlKxr6oX0Pxq_oC-dYkb/edit?usp=sharing&ouid=109023705375788511168&rtpof=true&sd=true>

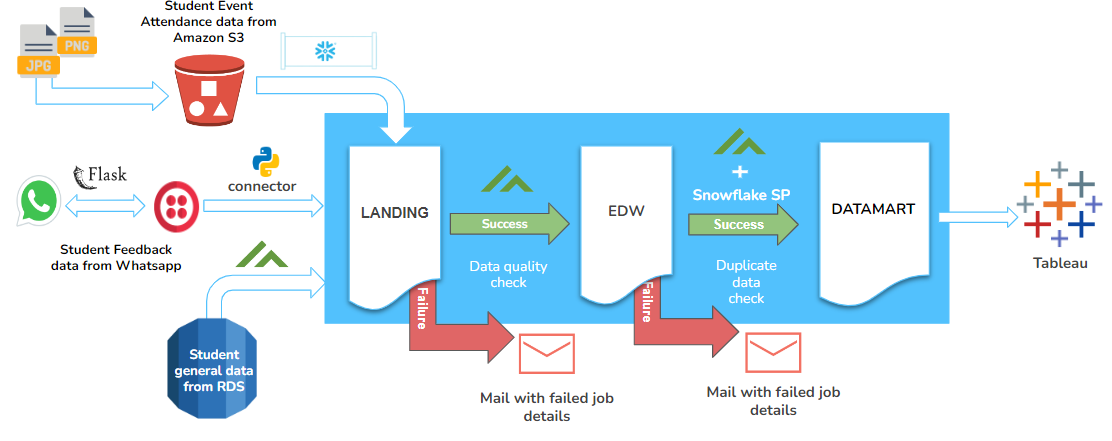
Given below is the detailed data model screenshot implemented in SQLdbm:



# Data Pipeline

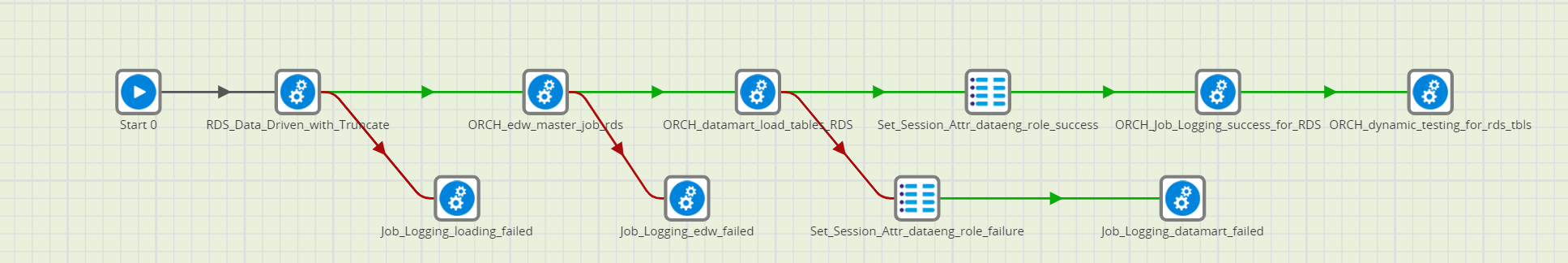
### Master pipeline for monthly data load to Snowflake Landing

Master pipeline to be scheduled as per requirement to load data from monthly data from various sources to Snowflake, with Truncate Load for Landing Layer tables, Incremental load for EDW table and UPSERT for Datamart tables. Student and mentor general data provided by the college is loaded using this pipeline.



1. The pipeline is triggered by an orchestration job.
2. Data in each layer is loaded using one orchestration job.
3. Data in the landing layer is loaded using one generic metadata driven job.
4. After loading data in each layer, the load is checked for failure logs using Matillion API Query. If found, the details of job failure are sent to an email address(configurable) and the pipeline is stopped.
5. If data is completely loaded in each layer without failure, one success-status email at the end of the data load process.
6. After the data load is complete, one reconcile job checks if all the data is loaded from each source layer to the target layer.

A screenshot for this master job is attached below:



#### **Data driven approach for Amazon RDS to Snowflake Landing**

Metadata-driven load implemented for loading data from RDS source to Snowflake, wherein source and target mapping between RDS tables and Snowflake Landing tables are stored in a Snowflake Mapping Table. Using information from this table, the Matillion ETL job loads the data for all RDS tables in one go. Following image shows a snapshot of the mapping table.

#### 

The following image shows the matillion transformation job used to load the data from RDS tables to Snowflake. The Grid Iterator component is used to iterate over the rows of the mapping table shown above, and truncate-load the data from tables mentioned under the “RDS\_SOURCE” column to the corresponding tables mentioned under the “LANDING\_TARGET” column.

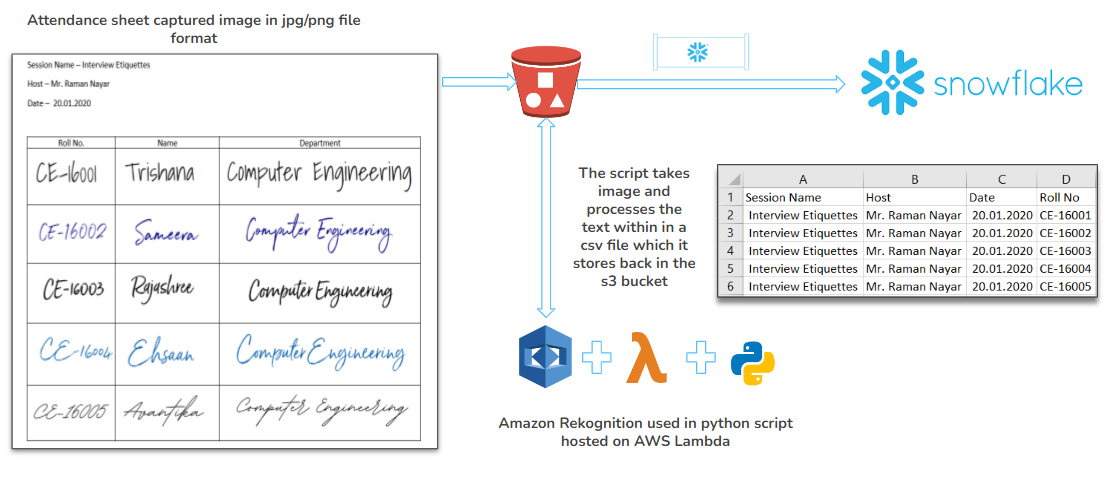
#### 

#### Student Attendance Sheet Data into Snowflake

The student event attendance sheet data is used to analyze the impact of student’s attendance in technical/non-technical sessions arranged by college on their placement. The steps for the process are as described below:

1. Attendance sheet images (.jpg or .png) are uploaded to an AWS S3 bucket.
2. A code written in python and hosted on lambda uses Amazon Rekognition to read the attendance sheet image and convert the text inside it into a csv file, which is stored back into the S3 bucket.
3. This csv file is ingested into the snowflake environment using Snowpipe integrated with error notification.

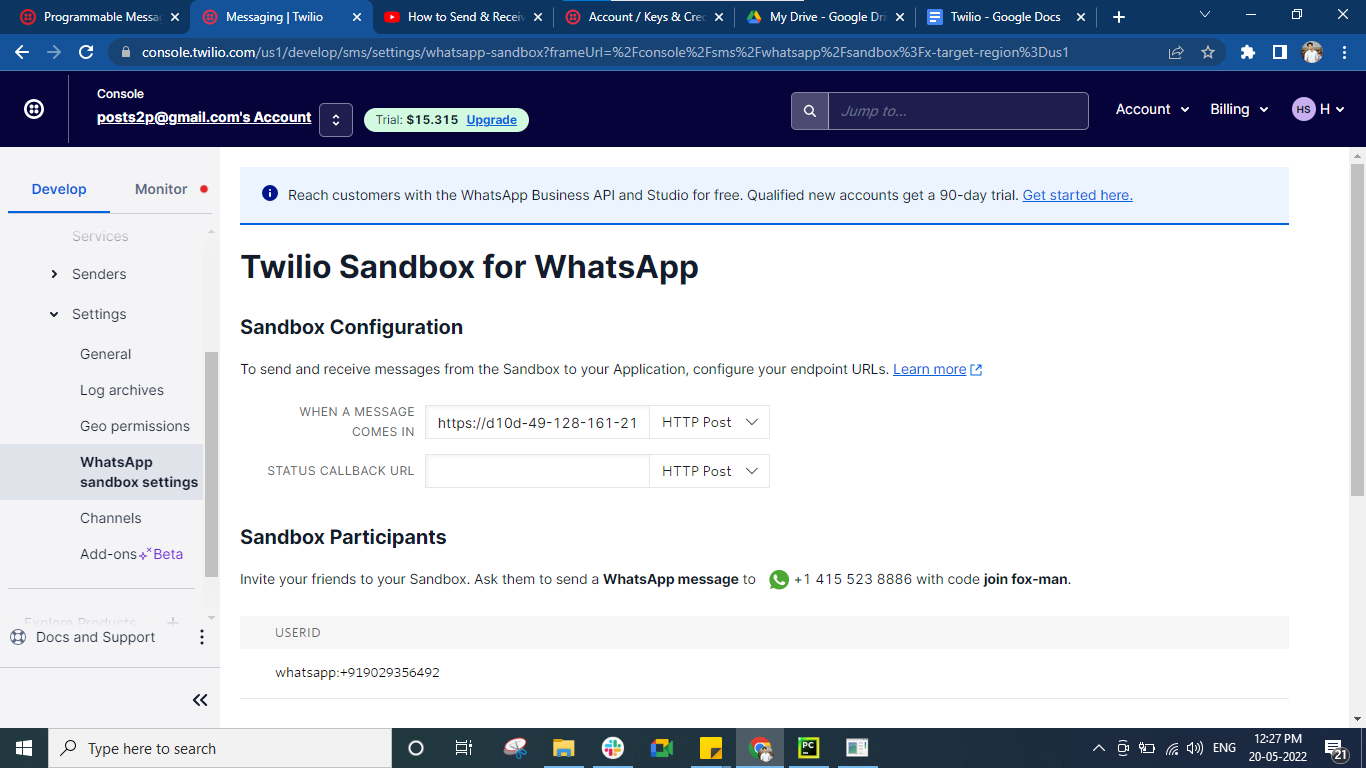
The steps mentioned above has been shown in the data flow diagram below:



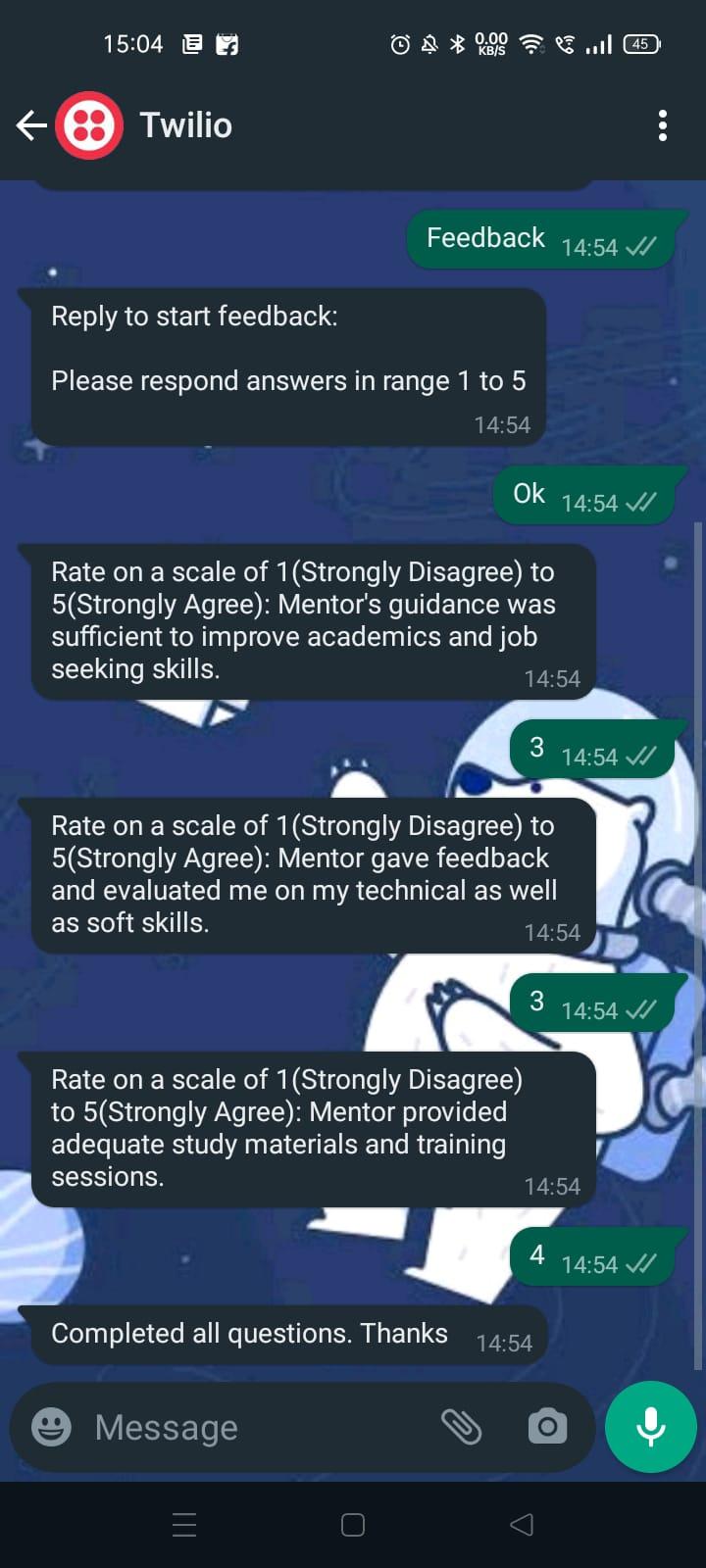
#### Student Feedback Data for Mentor

The code is written in python and hosted on heroku. The students first join the sandbox with a code provided by twilio. They can initiate the feedback flow, but are not allowed to submit feedback more than once every quarter. The steps for the same are mentioned below:

1. Students need to sign up into Twilio sandbox using a unique code. As seen in the image below the special code in our case was “join fox-man”.



1. Once a month, a feedback process is initiated from a python code using twilio library to communicate with the students on their whatsapp numbers, and the feedback scores for their mentors are received by asking a couple of questions as shown in the conversation screenshot given below:



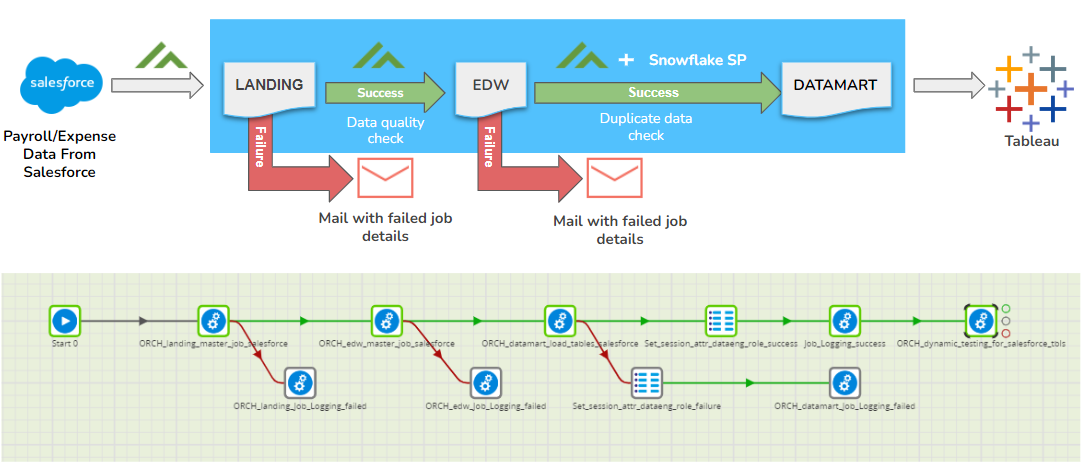
1. Once the feedback is received, the feedback score is sent to the Snowflake Landing schema using Snowflake’s python connector.

The following image shows the process explained above:



### Master pipeline for yearly data load from Salesforce to Snowflake Landing

Master pipeline to be scheduled as per requirement to load data from Salesforce to Snowflake, with Truncate Load for Landing Layer tables, Incremental load for EDW table and UPSERT for Datamart tables.



* The pipeline is triggered by an orchestration job.
* Data in each layer is loaded using one orchestration job.
* After loading data in each layer, the load is checked for failure logs using Matillion API Query. If found, the details of job failure are sent to an email address(configurable) and the pipeline is stopped.
* If data is completely loaded in each layer, one success-status email at the end of the data load process.
* After the data load is complete, one reconcile job checks if all the data is loaded from each source layer to the target layer.

### Master pipeline to load Twitter Feed to Snowflake (ad-hoc)

* Twitter Data is being used to do sentiment analysis of tweets and find out what technologies are currently hot topics and can be recommended to the college in order to include them as extra courses.
* We have picked a few technologies and then did a “quote”/keyword search on them from Twitter API using mulesoft.
* The tweets collected are stored in a json file in AWS S3, from where it is ingested using Snowpipe into Snowflake Landing Variant Column table. A materialized view is created on top of this table.
* Data from the materialized view is fetched and analyzed in a Dataiku job, and the technologies are stored along with the polarity back into Snowflake Datamart schema.
* From Datamart schema the analyzed data is consumed by Tableau for further analysis.

The following image shows the process described in a diagram:

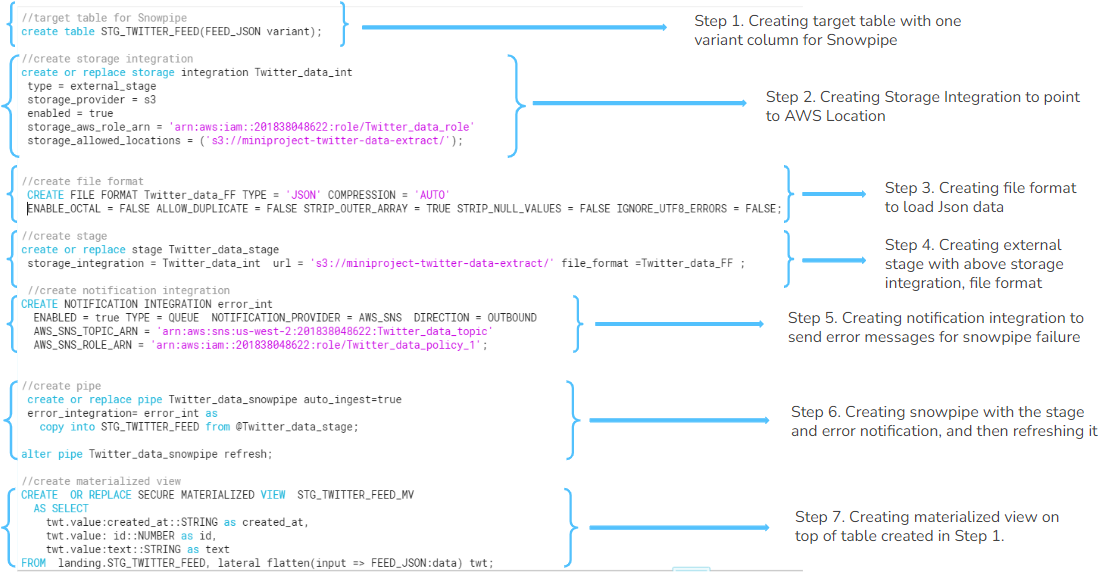
### 

#### Snowpipe with error notification

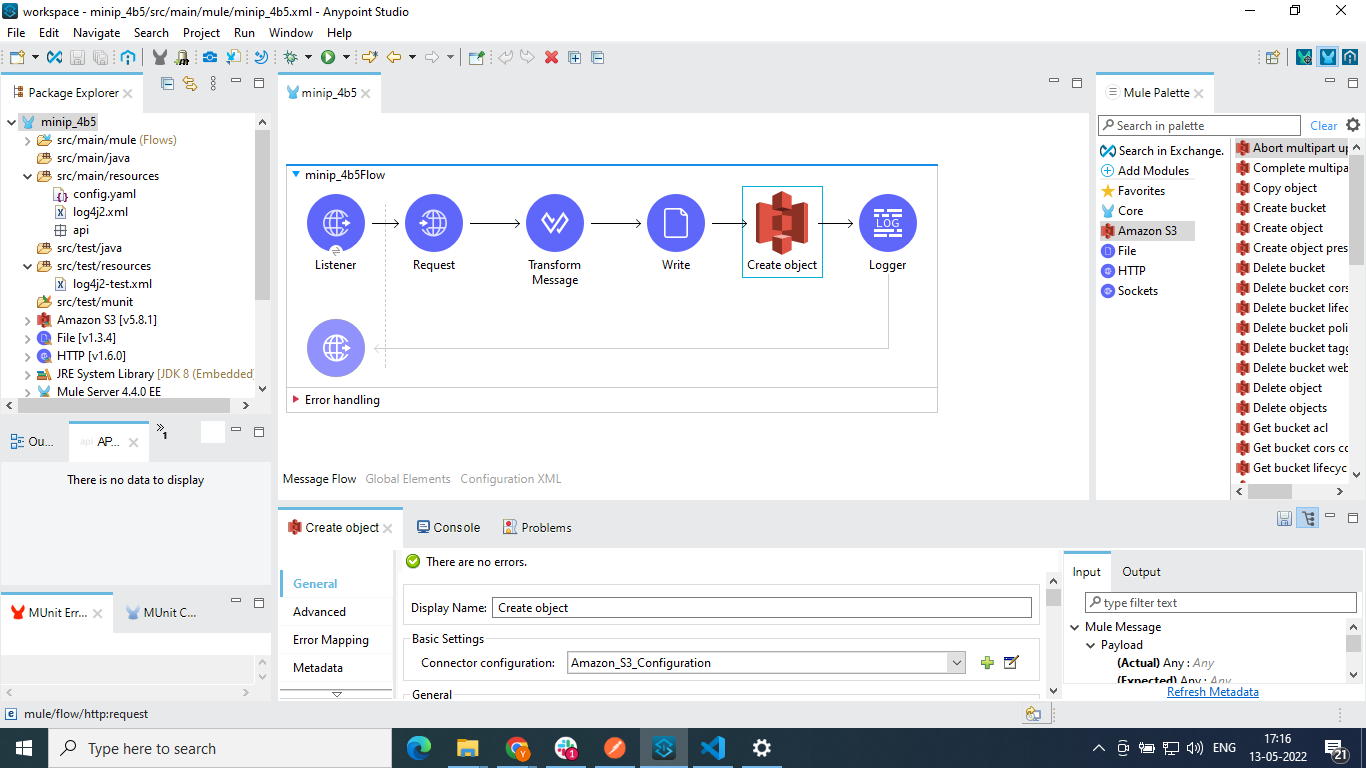
Till now, we have shown two instances where we have used Snowpipe:

* To Ingest Student Event Attendance data from AWS S3
* To Ingest Twitter Feed data from AWS S3

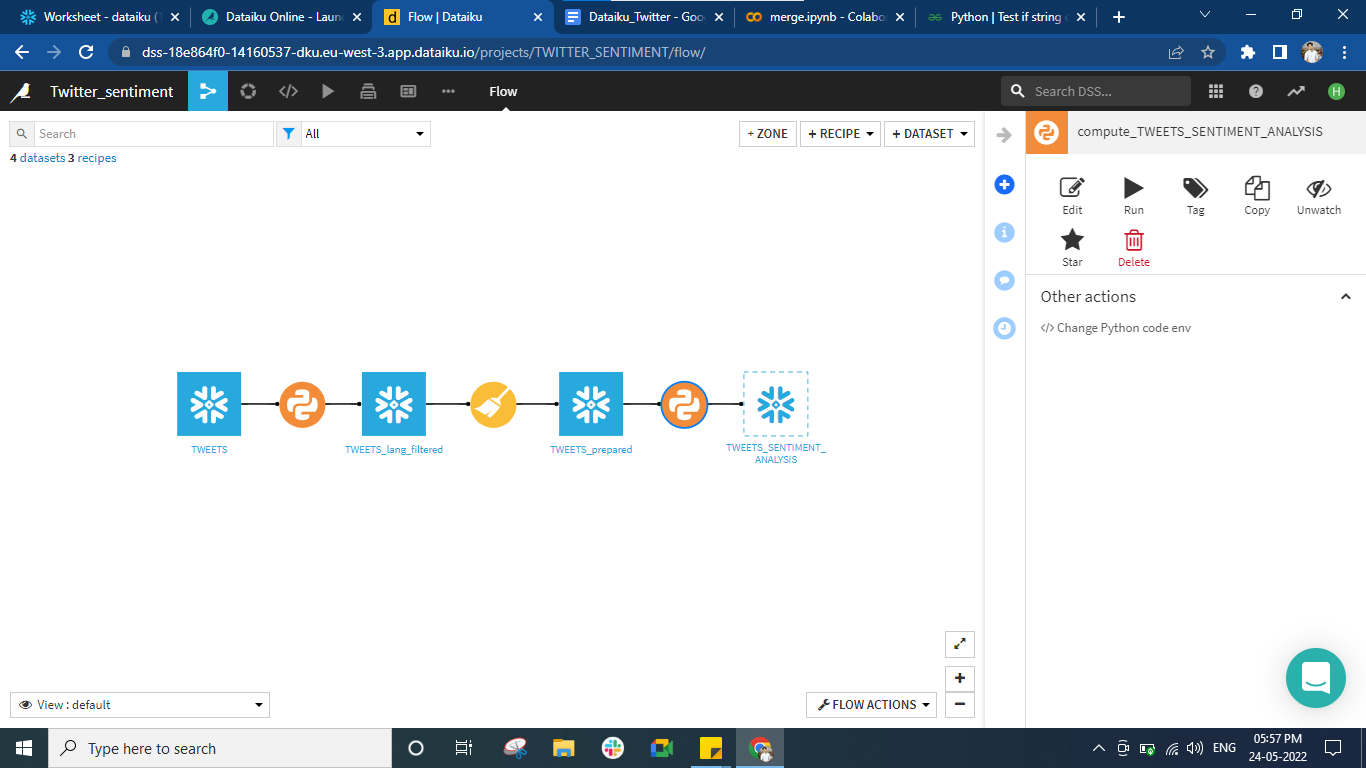
A sample code for the Twitter feed snowpipe has been shown below:

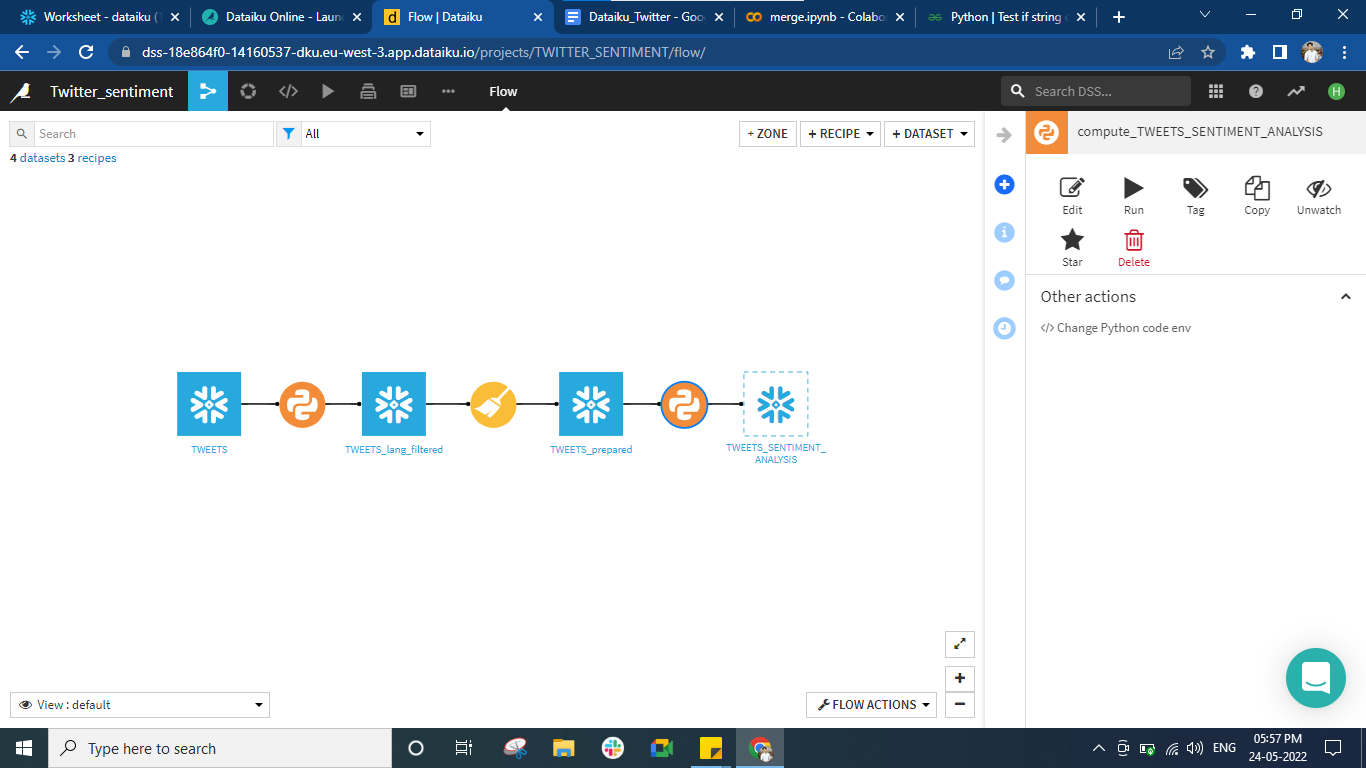


The following image shows the snapshot of the mulesoft job used to fetch data from twitter:

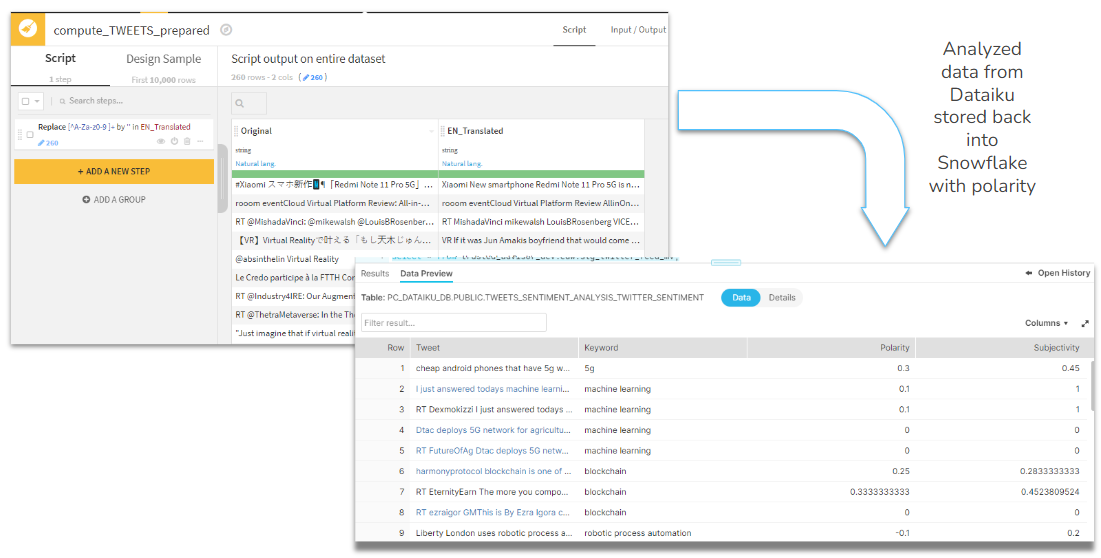


The following image shows the snapshot of the mulesoft job used to fetch data from twitter:





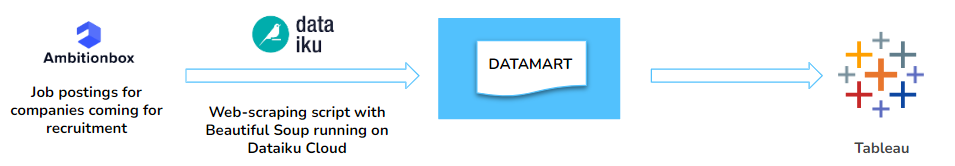
The following image shows the input for the dataiku job (the twitter data) and the output of the same stored in a snowflake table (the tweet text, the keyword found, the polarity and subjectivity around the text).



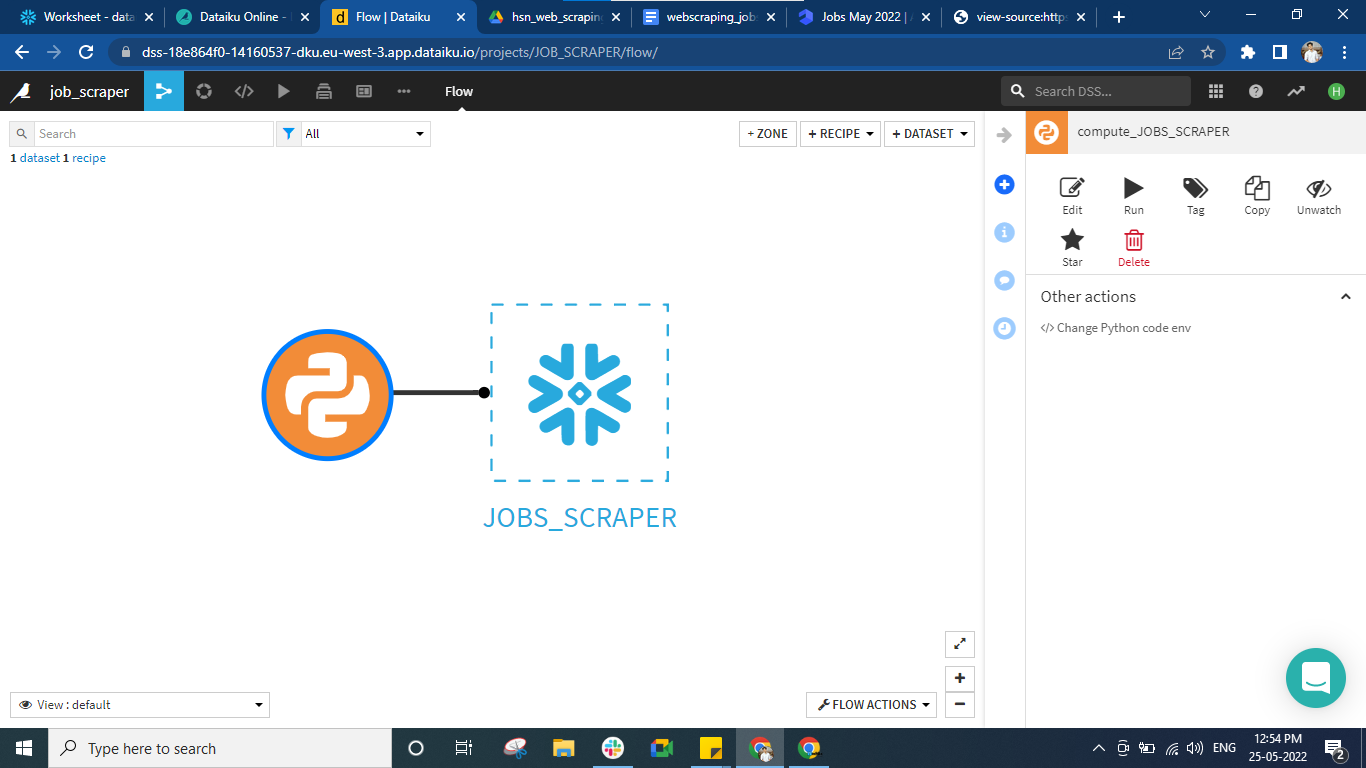
### Master pipeline to load scraped Job requirement details from Ambitionbox.com to Snowflake (ad-hoc)

* A lot of times, students are not aware about the technologies the companies are looking for, when they come for campus recruitments.
* We have found a way to scrape job-sites like AmbitionBox, where such companies post job requirements for freshers and use that scraped data to make students more aware of technologies to focus on before going in for the interviews.
* The scrapper script uses Beautiful soup python library and is hosted on Dataiku. The list of Companies with technologies scrapped is stored back in Snowflake and then processed by Tableau.

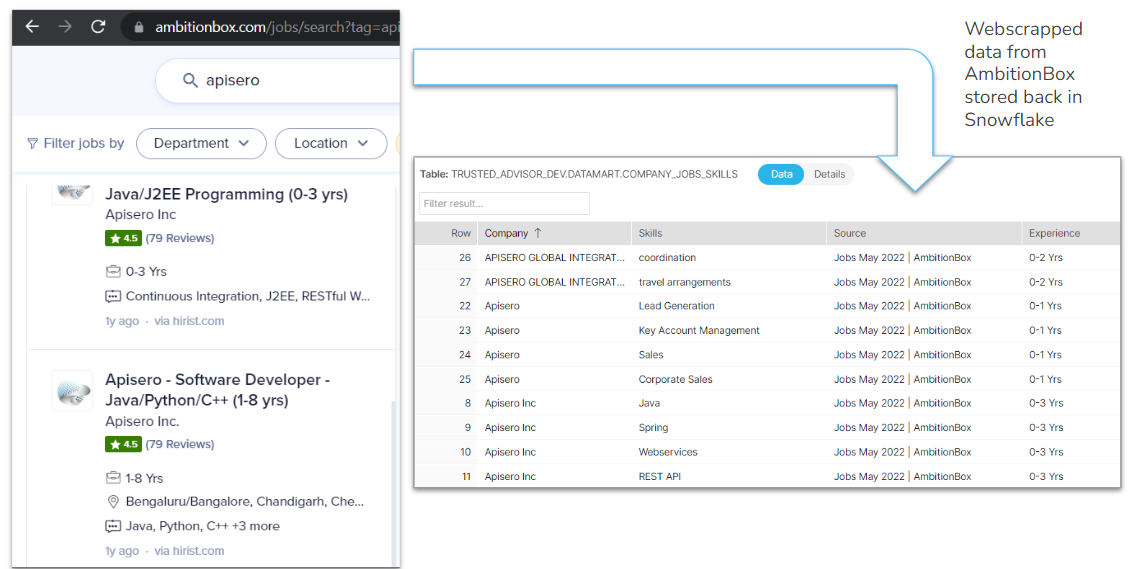
The image below shows the data flow process for the same:



The image below shows the dataiku job that hosts the web-scraper script:



The image below shows the input of the scraper script (the job details from ambition box) and the output (the job skill requirement with the company name) stored back into the snowflake table.

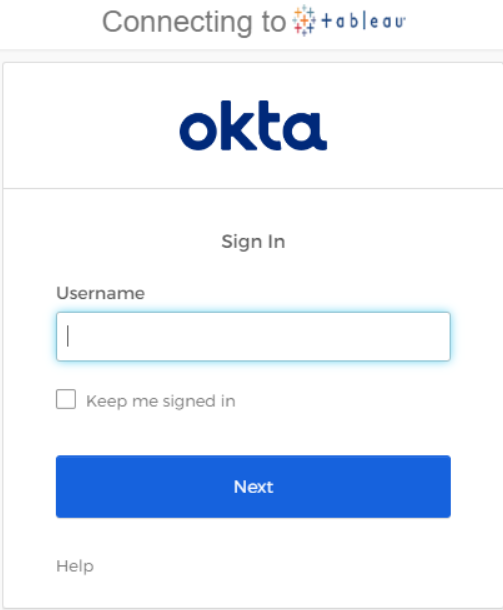
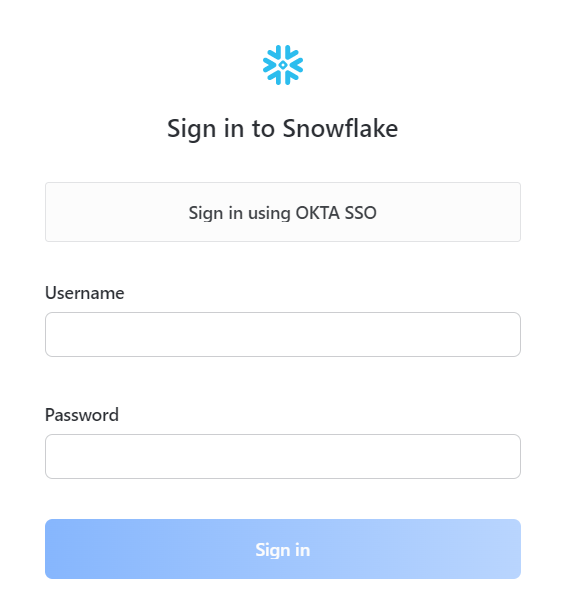


# Security Features

Like in any data warehouse project, security is of utmost importance because your data needs to be protected against tampering and unauthorized access/viewing. For this purpose, we have implemented the following security measures:

### Okta Integration with Snowflake and Tableau

Federated authentication is implemented in our snowflake and tableau environment using Okta. Federated authentication enables users to connect to Snowflake using secure SSO (single sign-on). With SSO enabled, your users authenticate through an external, SAML 2.0-compliant identity provider (IdP). App integration is created with snowflake and all the users are assigned to it. With this users can directly login to their snowflake environment using option “Sign In using Okta” without providing credentials.

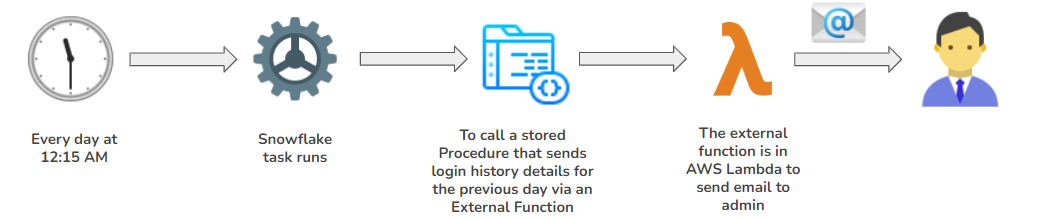
 

### 

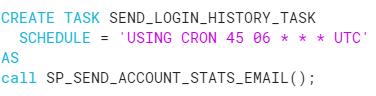
### Account Login Attempts shared to Account Admin

As an account admin of a snowflake account, it is important to know which users have logged in on a daily basis. For this purpose, we have created the following process, to notify the account admin which users have logged in to the snowflake account every day.

Following is a pictorial representation of the process:

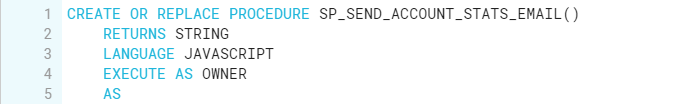


1. At 12:15 AM IST every day, a task will run. The SQL for the task is as follows:

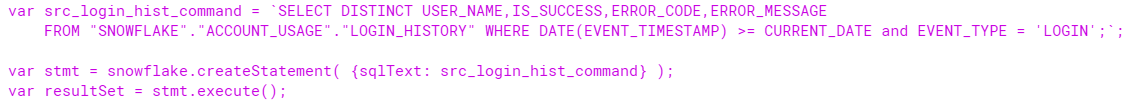


1. This task calls a Snowflake Stored Procedure written using Javascript. The following is the SQL of the stored procedure:

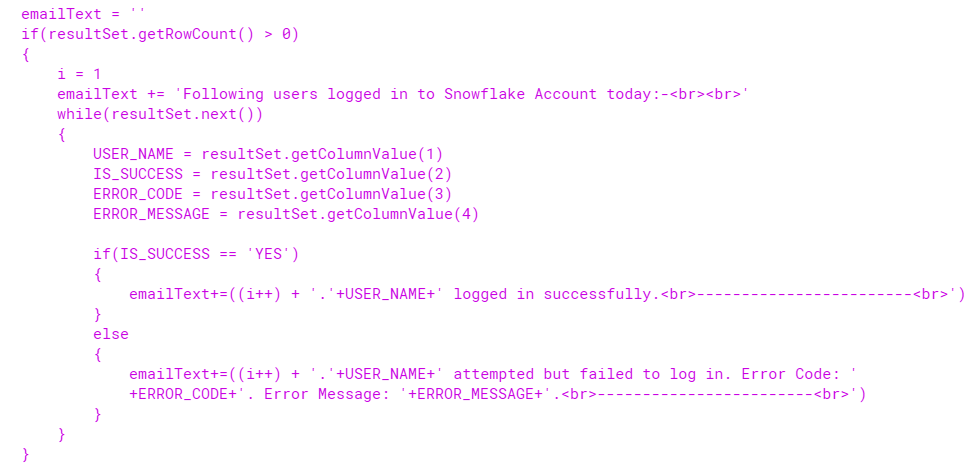
Declaration:



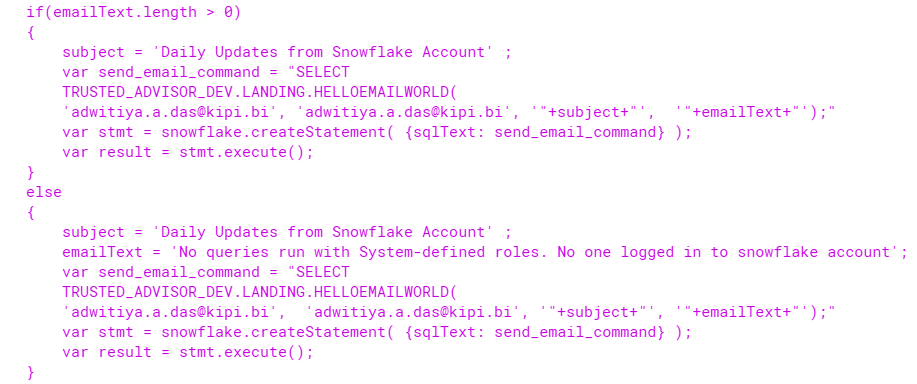
1. Fetching the previous day’s login history from "SNOWFLAKE"."ACCOUNT\_USAGE"."LOGIN\_HISTORY":



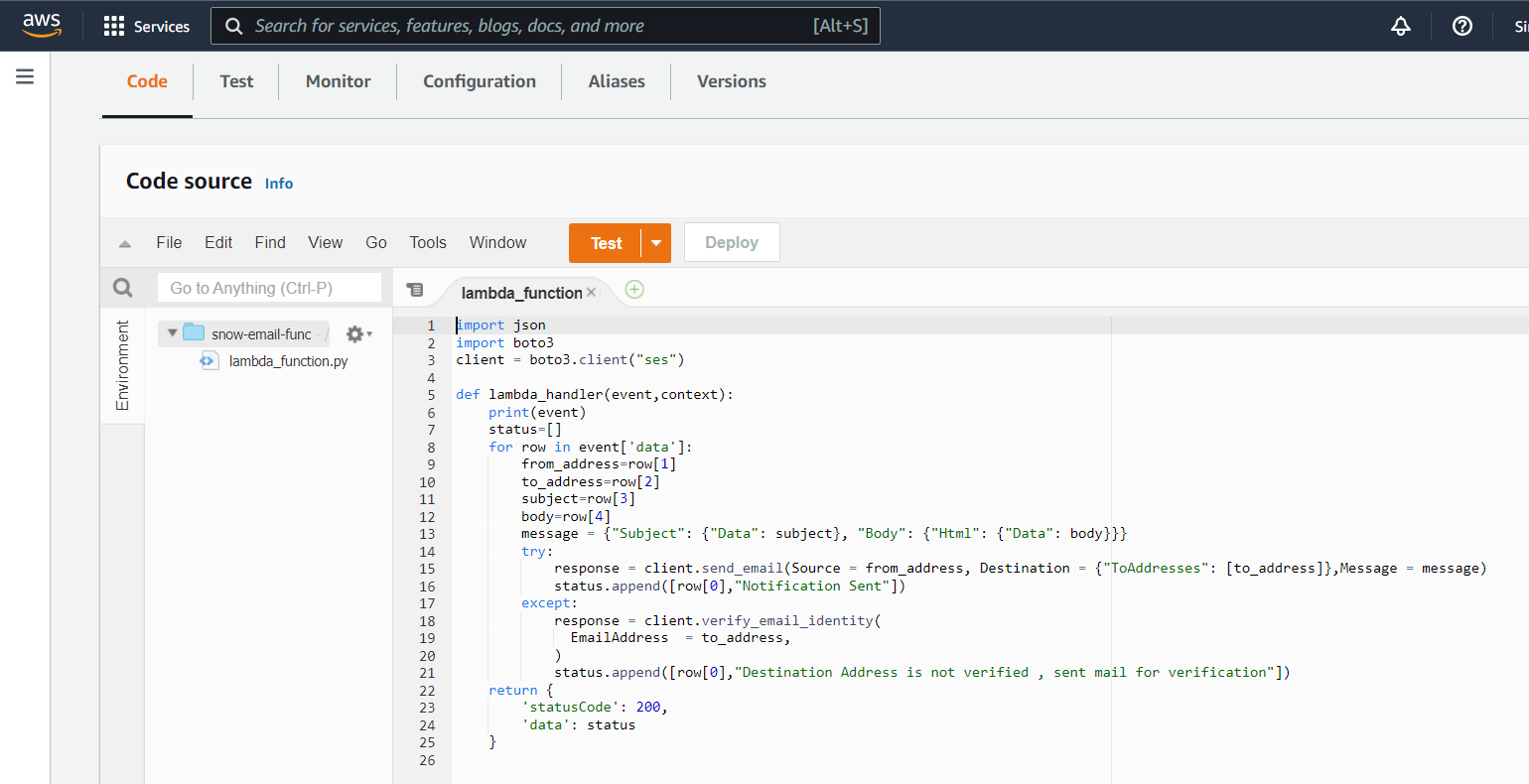
1. Forming Email Text:



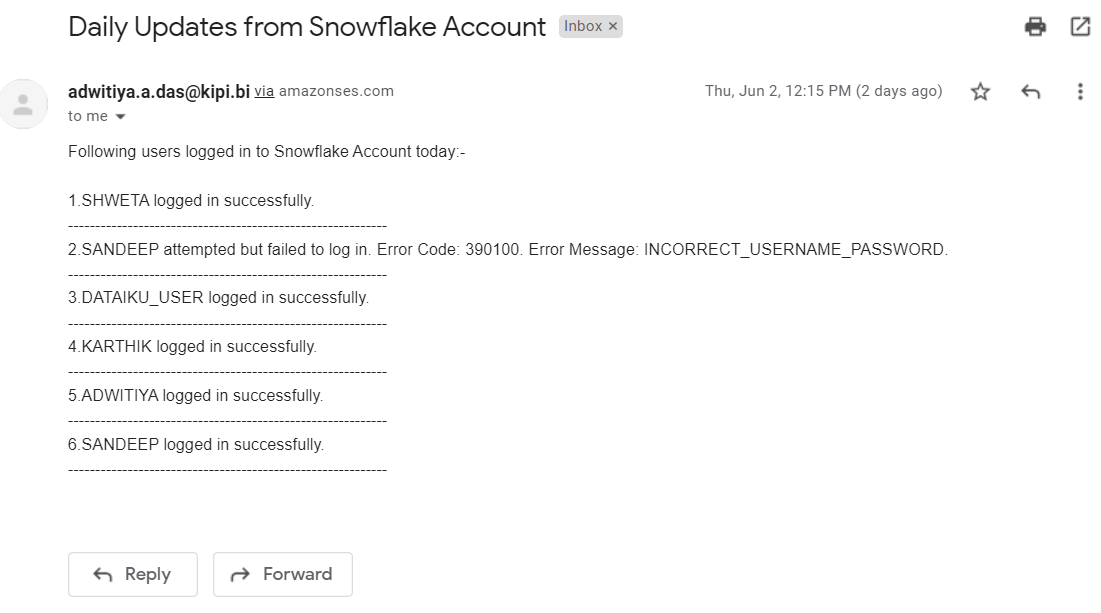
1. Sending email:



1. The external function to send the email to the account admin is present in AWS Lambda. This email function can be used to send email to any email address from any email address, as long as the address is verified in AWS. If the email address is not verified, then on first calling this function, a verification email is sent to the sender and the receiver. After the verification link is clicked and the email is verified, the receiver will get the email with login history details. The code for the same is shown below:

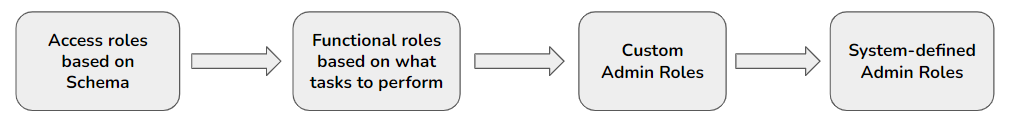


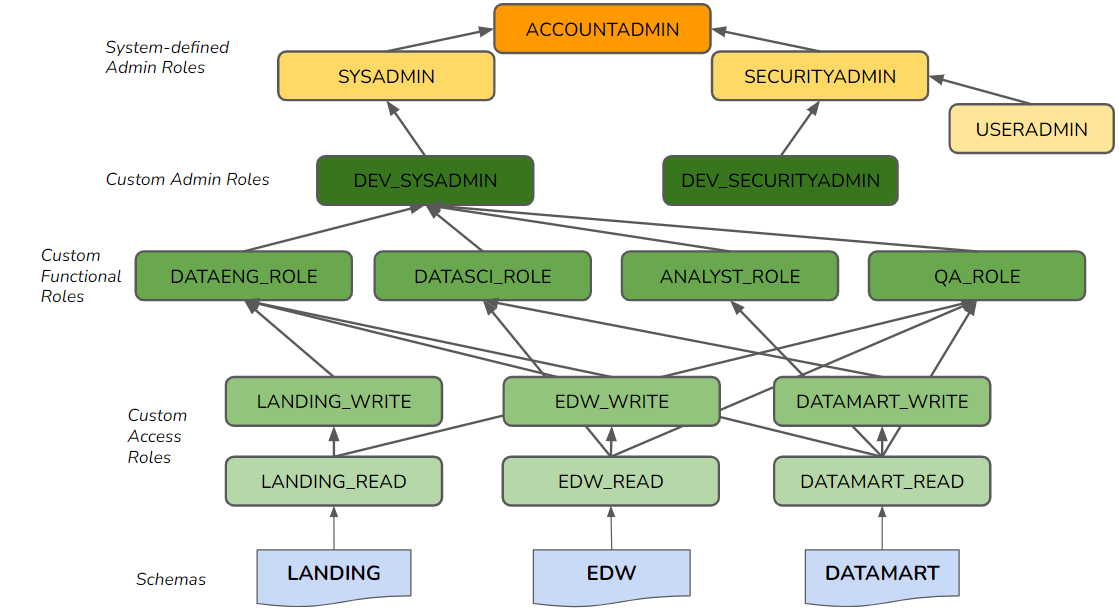
1. The following image shows the email received:



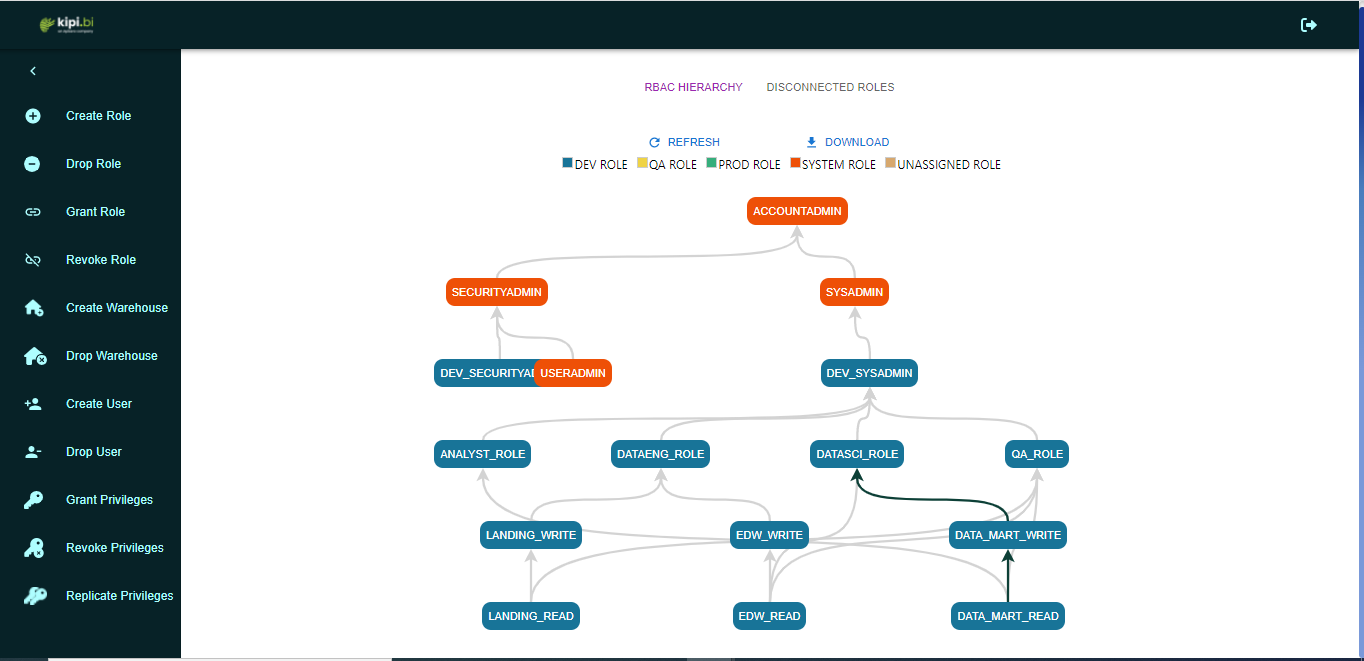
### Role-based Access Control using Kipi’s RBAC Accelerator

The RBAC hierarchy consists of 4 types of roles. Each role type is granted to the next role type as shown below:





The above RBAC hierarchy has been implemented using the RBAC Accelerator. The following screenshot show the RBAC hierarchy, as seen in the accelerator interface:



# Resource Monitoring

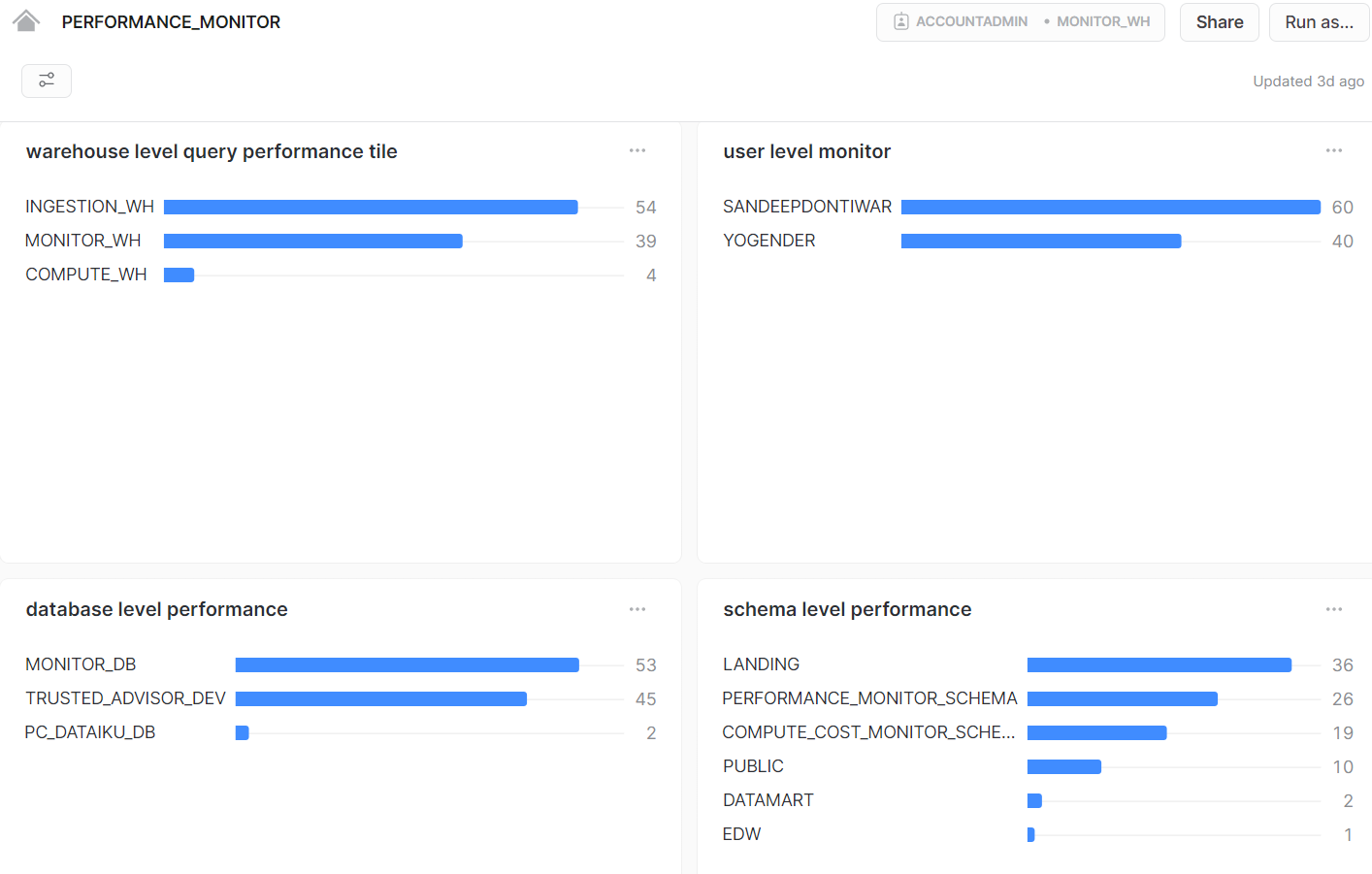
Resource Monitoring has been done using kipi’s Resource monitor. The Monitoring accelerator is used to manage the resources we are using like credits, storage. In order to do effective monitoring we have used 3 monitoring techniques used to monitor cost, performance, storage.

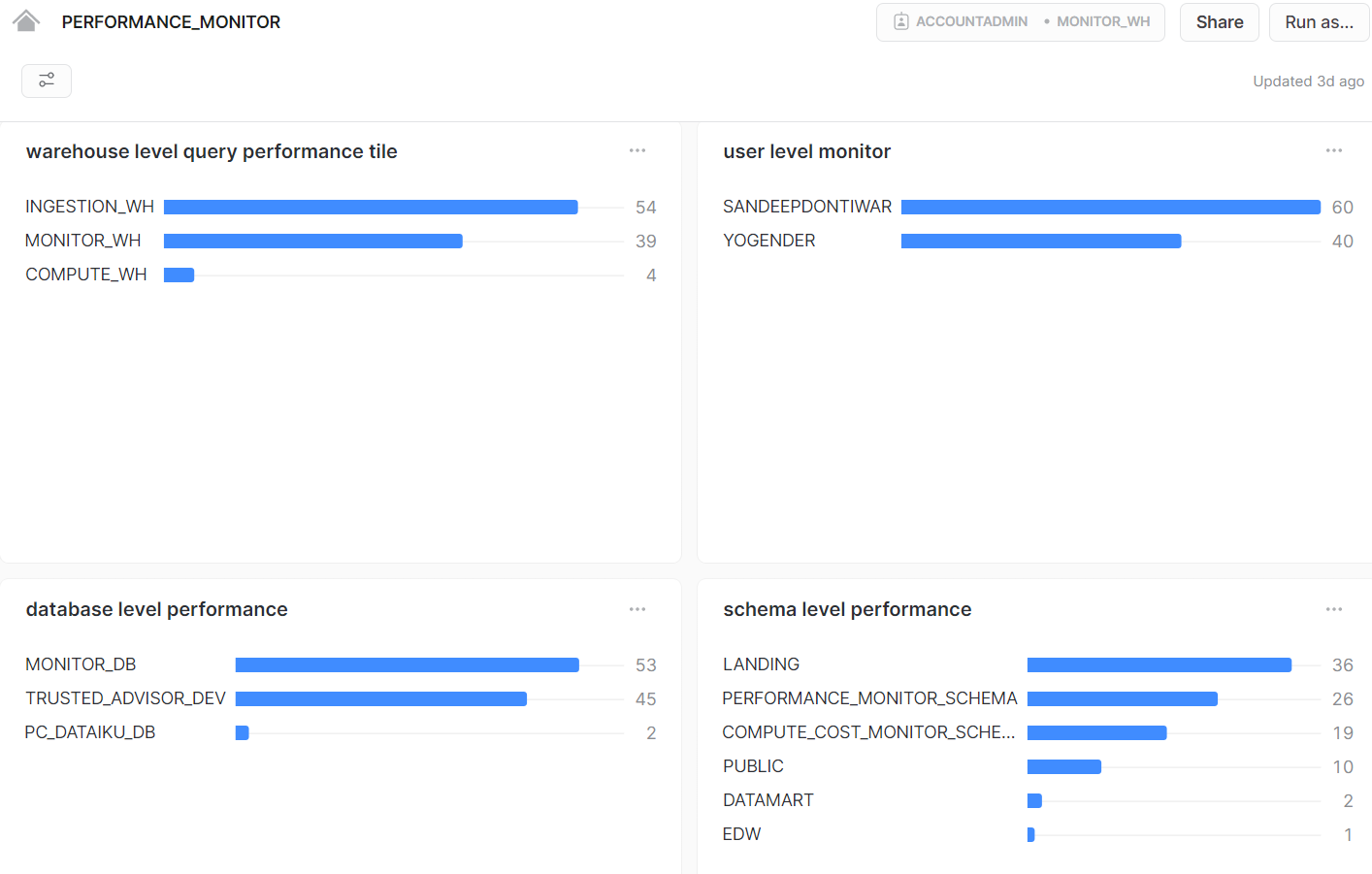
### Performance Monitoring

Performance monitor is used to monitor the performance of the account by shortlisting queries as good or bad. If any query exceeds 20 min of running time, it will be considered as a bad query.

Performance is monitored on 4 levels : warehouse, user, database and schema levels.

The screenshot below shows the performance monitoring dashboards on each level:



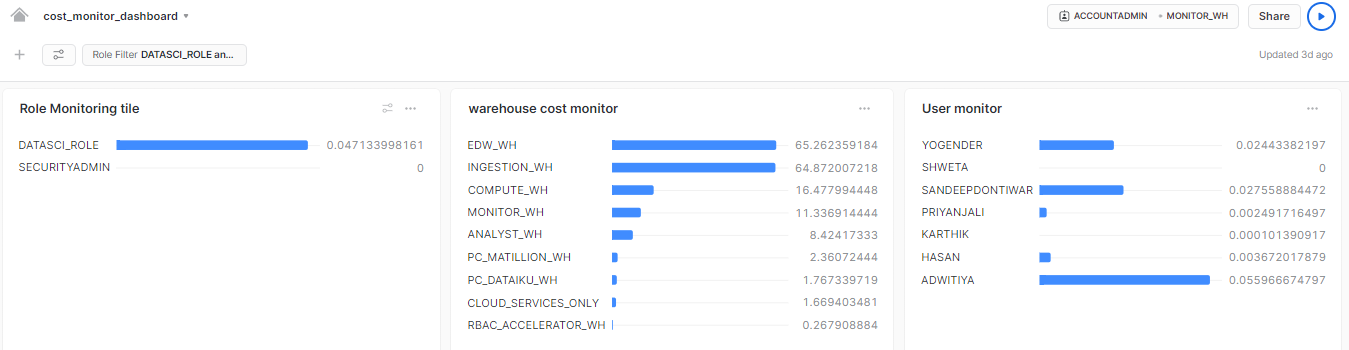


### Cost Monitoring

Cost monitoring is used for maintaining the number of credits used in our account on different levels and thus updating users on the cost used.

Cost monitor works on 3 levels here that is role, warehouse, user level.

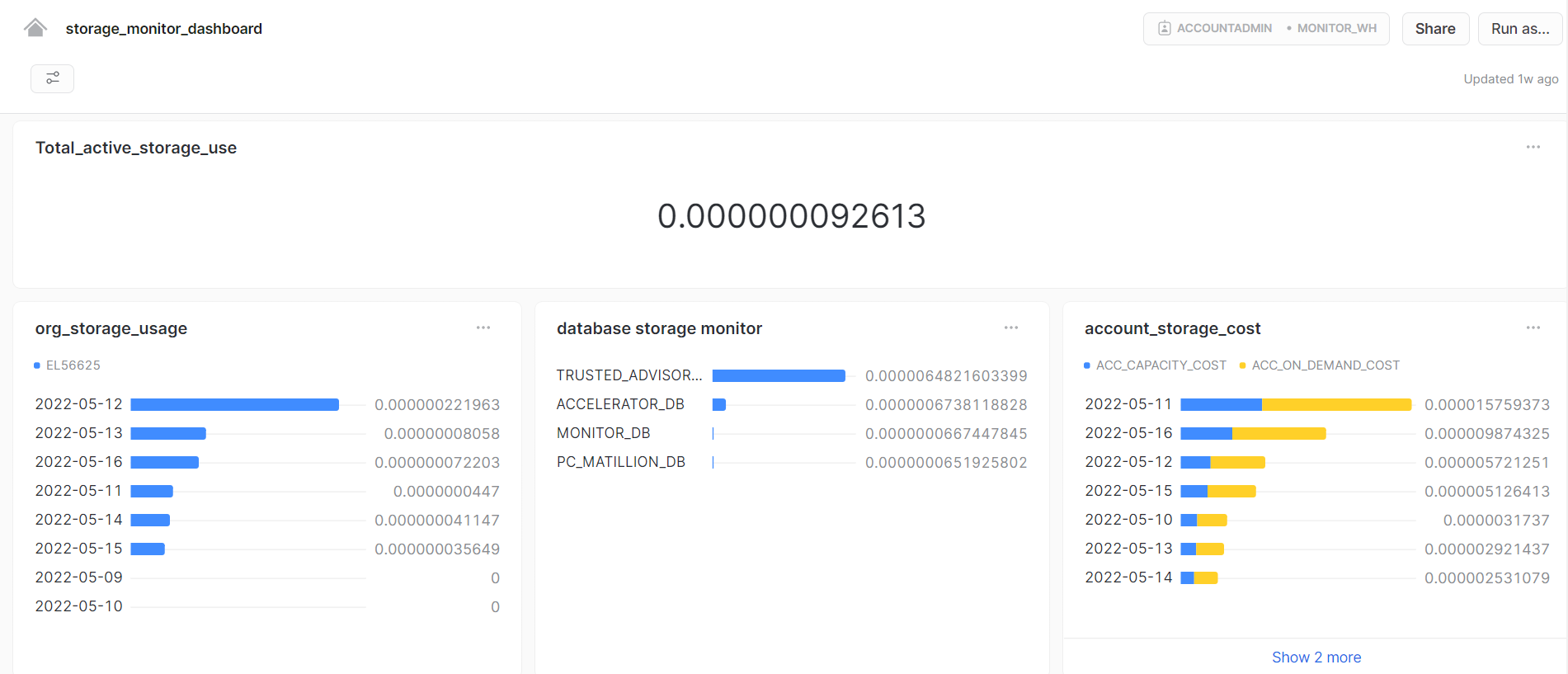
The screenshot below shows the cost monitoring dashboards on each level:



### Storage Monitoring

Storage monitor is used to effectively use the amount of total storage available and for this purpose we have used managed storage on 3 levels, which are account level, organization level, database level. Also, we can track the storage used by active data.

The screenshot below shows the storage monitoring dashboards on each level:



# 

# Data Quality Check and Audit Features

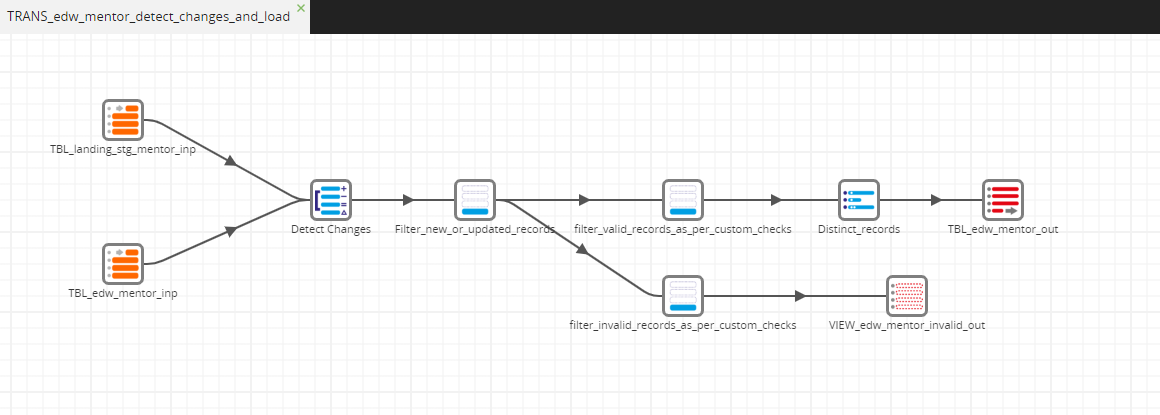
Data quality checks and audits are important to ensure data consistency and integrity. It also helps to reduce rework by catching issues in data early on in the pipeline. We are checking for data quality before data is loaded from Landing to EDW schema. Mentioned below are a few data quality and audit measures implemented:

### Type check, Range check and Null check before loading data

The following checks are done before data is loaded from Landing schema to EDW schema:

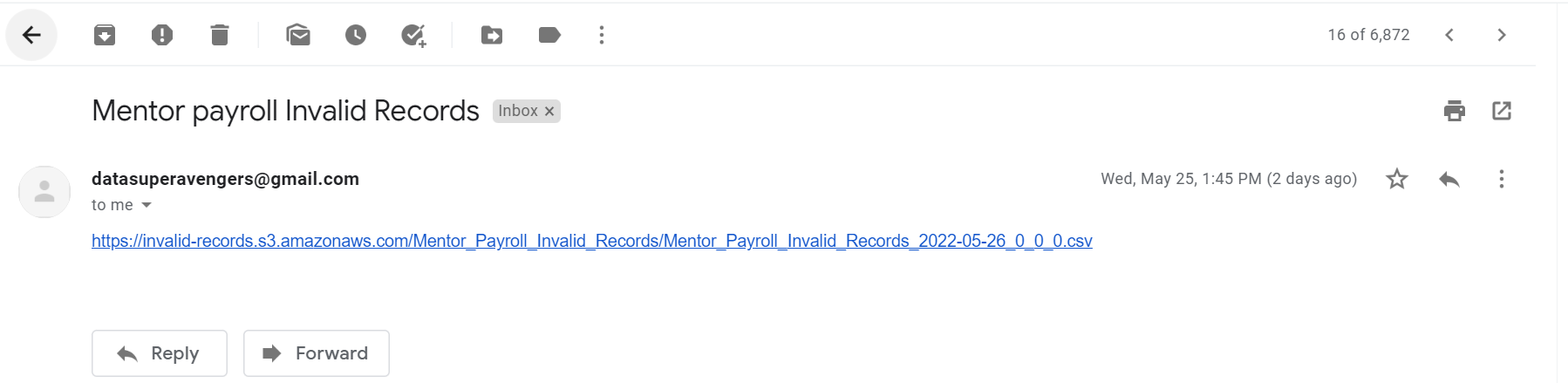
1. If the data type of the value is correct
2. If the value is null (allows if the target field is nullable)
3. If the value falls within the expected range. For example, the postal code in our case should be between 100000 and 999999.

The following image shows a sample matillion job where data is checked for quality and then allowed to pass:



If there are any records found which do not pass the data quality check, the records are stored in a view created in snowflake. After the pipeline has executed for the remaining records, the invalid records are fetched from the views and stored into a csv file stored in S3 bucket. The S3 link of this file is sent in an email to the concerned addresses.

The following image shows an email received with the link of S3 location of a file containing invalid records:



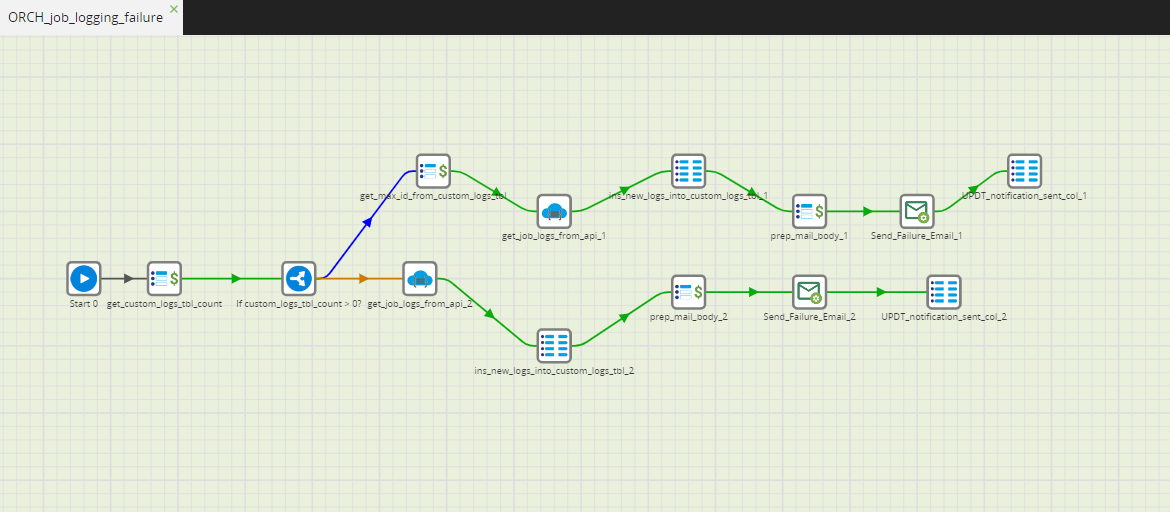
On clicking the link, the file is downloaded. The contents of the file are shown below:



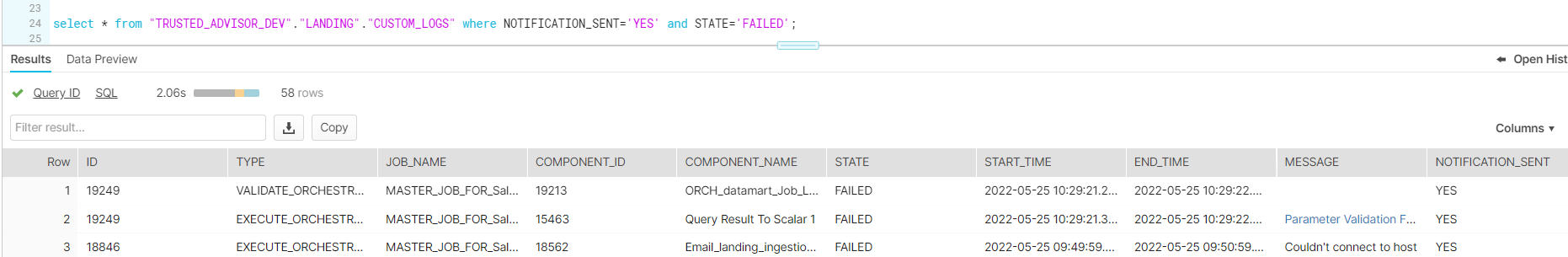
### Job Logging

Logging job execution status in "TRUSTED\_ADVISOR\_DEV"."LANDING"."CUSTOM\_LOGS" table, using Matillion Api. Notification sent for jobs that failed in the master pipeline with specific details like job name, component name, failure reason etc. Notification is also sent at the end of each master pipeline, when the data is loaded successfully, but without any specific detail about the jobs.

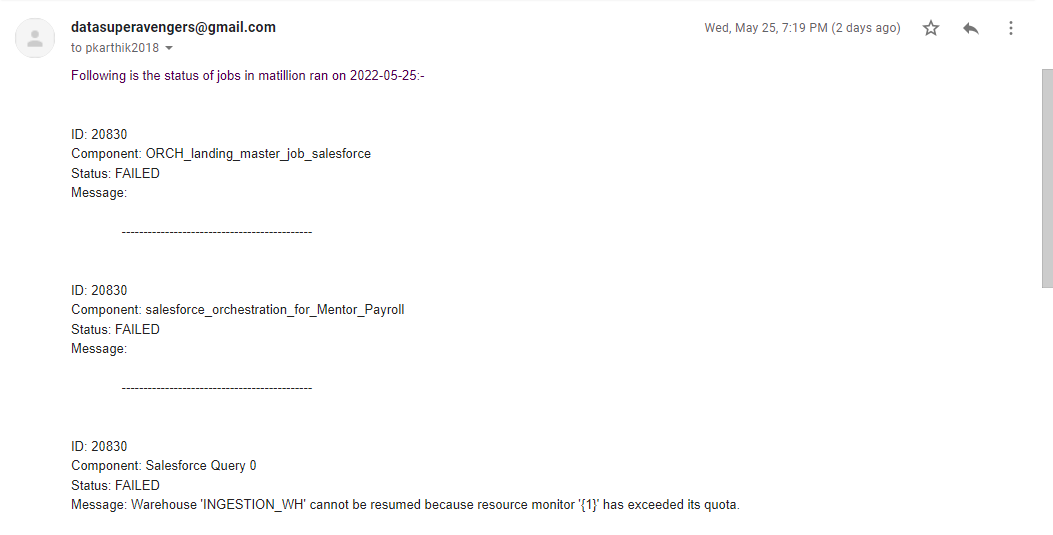
The following image shows the Matillion orchestration job used to call Matillion Api and get details of the jobs executed for the current date. Once the details are received, the data is loaded in the CUSTOM\_LOGS table in LANDING\_SCHEMA. Between the two flows of the job (as seen in the image), the top flow is executed when the target table has zero records, while the bottom flow is executed when the target table already has one or more records.



The following image shows the CUSTOM\_LOGS table in LANDING\_SCHEMA, where the logs of all the jobs that have been executed are stored at the end of each pipeline.

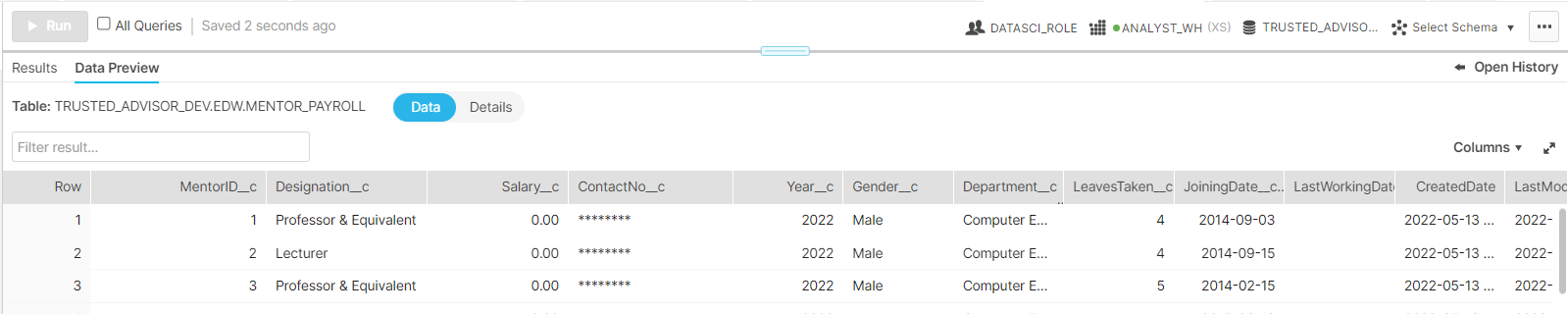
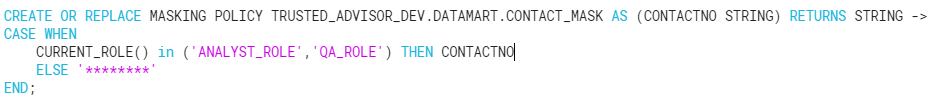


The following image shows the email received when any of the jobs in any pipeline has failed. The email received contains details about the component that failed and the reason for failure:



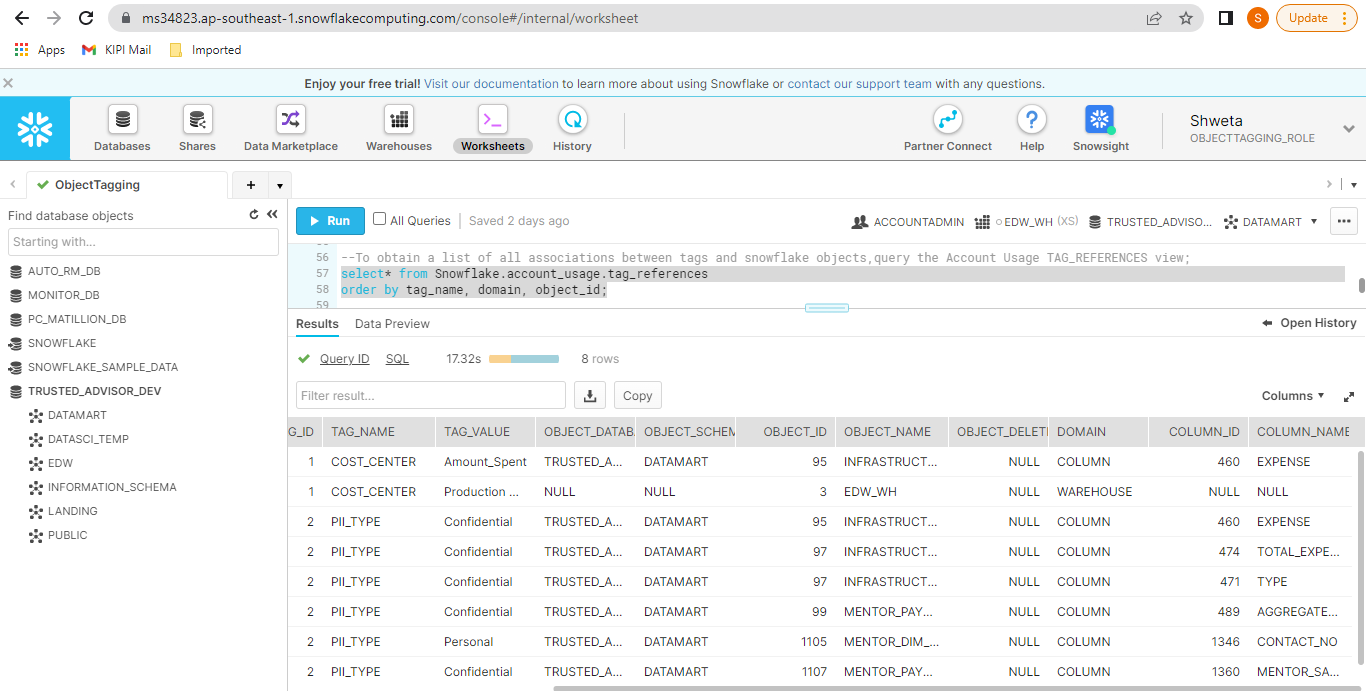
### Data Masking

Dynamic Data Masking has been used to protect sensitive data from unauthorized viewing. Salary, Contact No., Email columns are masked for all roles but the QA role and Analyst Role.



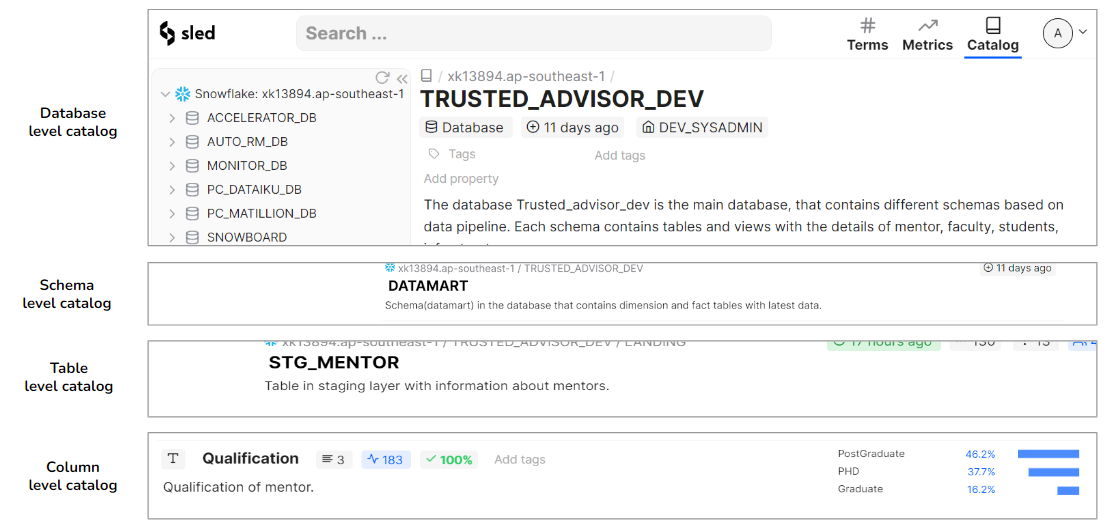
### Object Tagging

Object Tagging enables data stewards to track sensitive data for compliance, discovery, protection, and resource usage use cases through either a centralized or decentralized data governance management approach. Salary, Amount, Contact No., Email columns are tagged according to the sensitivity type like “PII\_TYPE” and assigned values according to the sensitivity level and categorized for tables and views.

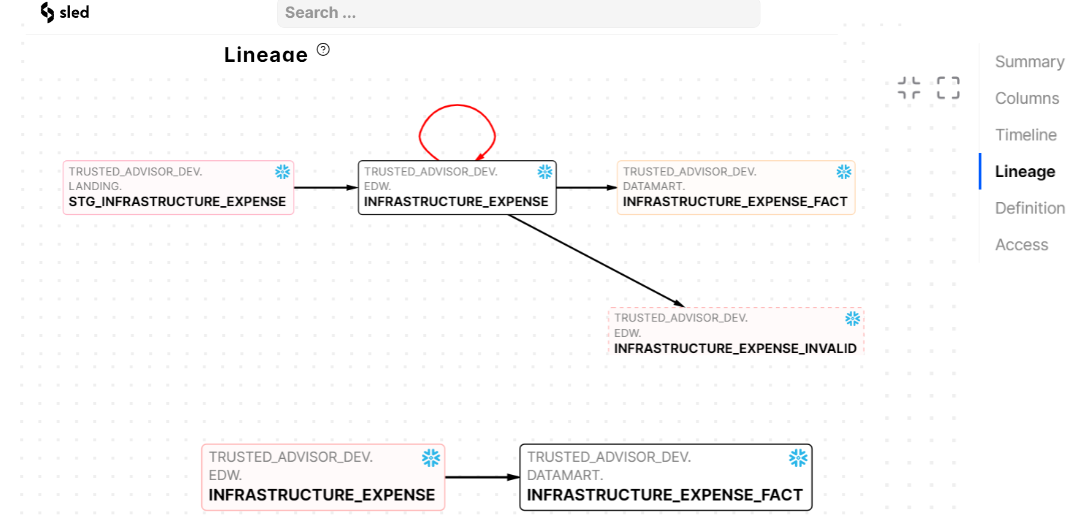


# Data Cataloging and Lineage

Snowboard has been used as the data cataloging and lineage tool. Account has been created using personal email. Cataloging has been done on all levels as shown below:

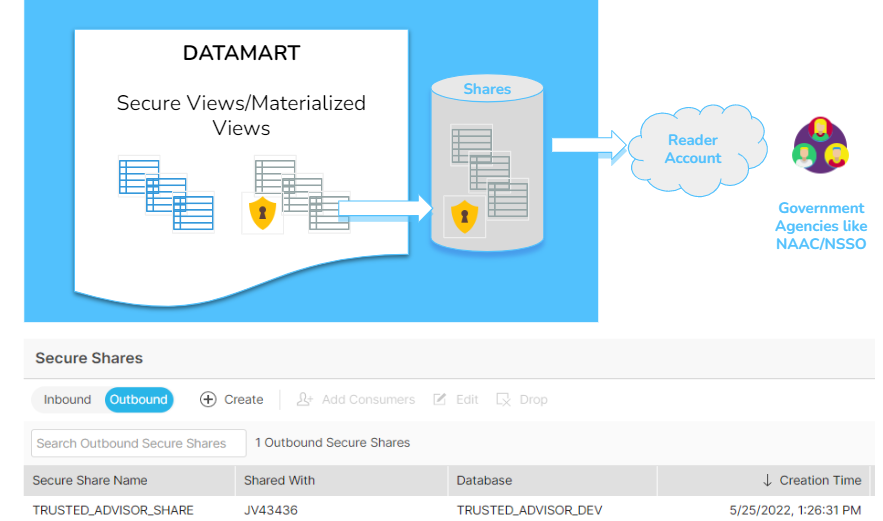


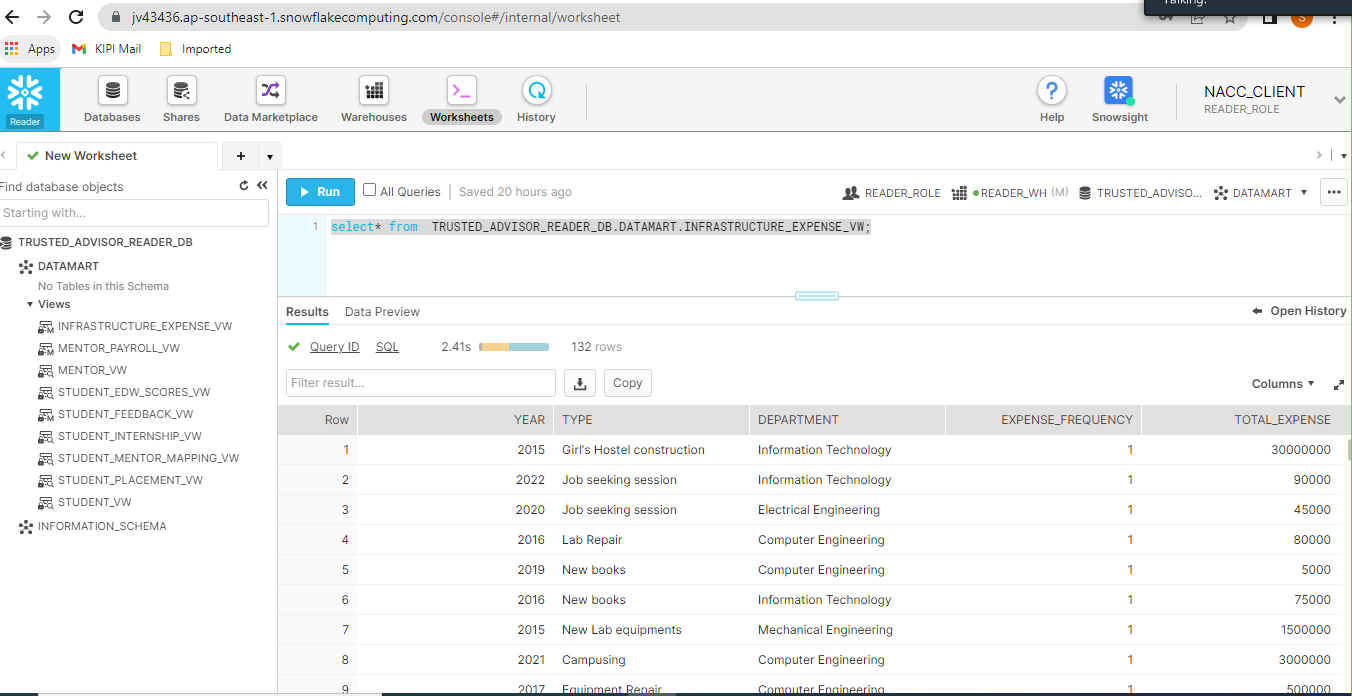
The following image shows the lineage recorded in Snowboard for one of the tables from Landing to Datamart schema.



# Data Sharing

Data sharing has been used to share monthly/yearly aggregate data in secure views with reader accounts. These reader accounts are accessed by third parties like government agencies such as NAAC and NSSO, that maintain college reports for such data.





# Testing

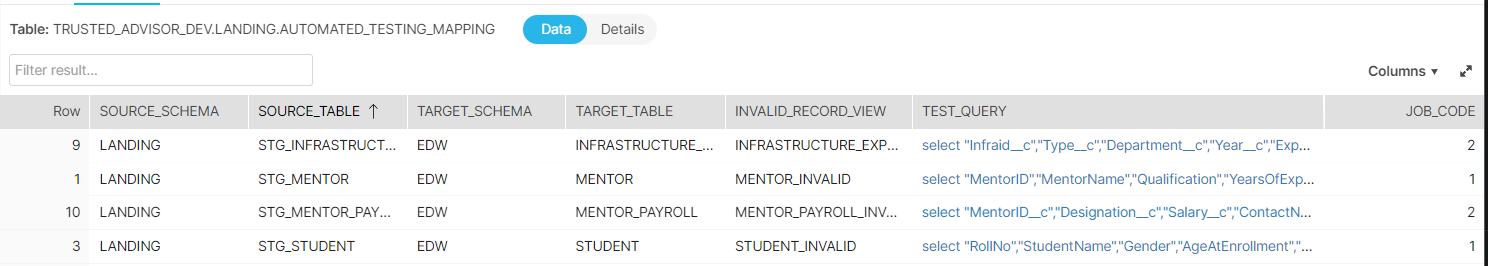
Two types of tests have been conducted:

1. Automated testing using reconcile job in Matillion:

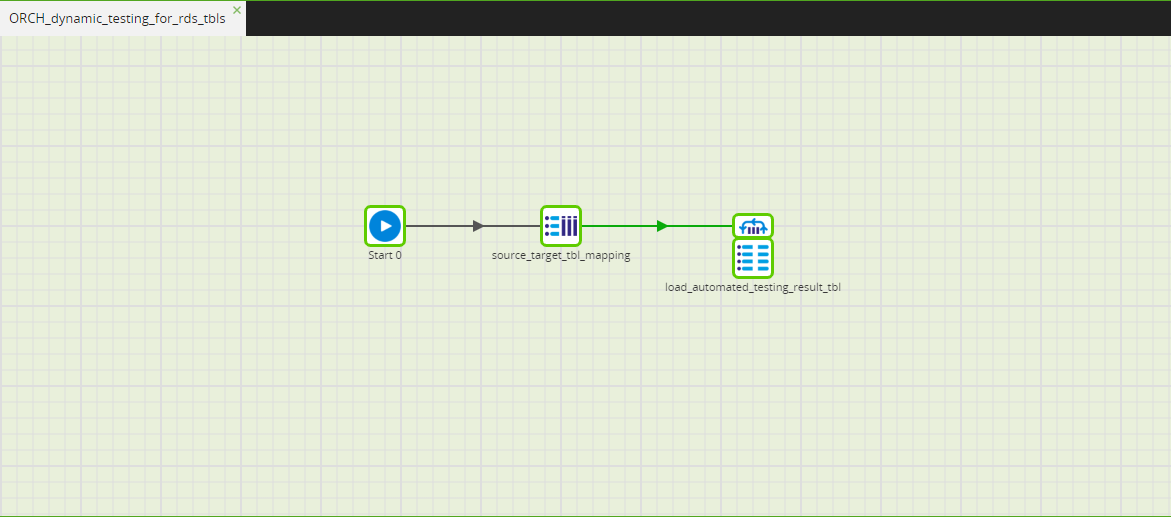
Data load from each source to target is checked using one metadata driven reconcile job.

* For any Source A mapped to be loaded to Target B, it is checked if there is any record present in A which is not loaded in B. Such scenario is only expected to happen in case there is any invalid record in A that did not pass the data quality check.
* The source to target mapping is stored in a mapping table in Snowflake, along with a query to check the data load, and the view name that stores invalid records of Source.
* Every time this reconcile job is ran, the source information, target information, query to check the load from source to target, and the record count of the view containing invalid records are fetched and stored in another Snowflake Table

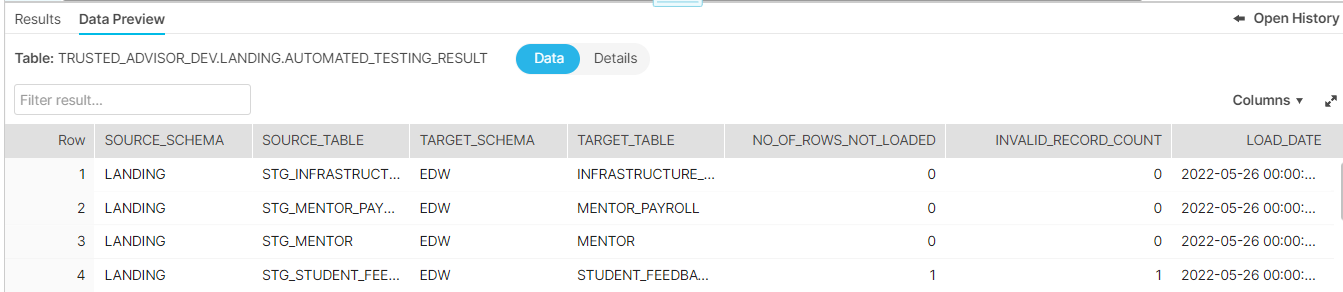
The Source Mapping Table:



Reconcile job in Matillion:



Target Results Table:



# Manual Unit Testing. Link to the unit testing document is provided below:

[Unit Testing Document -Trusted Advisor](https://docs.google.com/spreadsheets/d/1Nqdtt_SqPm6wWd-13gVw-5gGClKYk4aa/edit?usp=sharing&ouid=109023705375788511168&rtpof=true&sd=true)

# 

# Challenges and Accomplishments

Listed below are the challenges and accomplishments made during the course of the project.

### Challenges

Challenge 1: Resource crunch due to members’ unavailability.

Solution: We pulled through with the efforts of everyone.

Challenge 2: Matillion Job Logging Email was not showing the job logs properly.

Solution: Figured out the issue and was able to complete the task.

Challenge 3: Initial approach to get Twitter feed using user handles wasn’t returning much valuable tweet data to analyze.

Solution: Turned the idea around and used keywords instead.

Challenge 4: Twitter plugin for Mulesoft was not working properly.

Solution: Connected with various members of Kipi and found a solution.

### Accomplishments

*Functional -*

1. Improving on our use-case every week to add something new.
2. Generated our own data, with coding logic in Python.
3. Followed processes that are implemented in real projects, such as:
   1. Data quality checks
   2. Job logging and error notification
4. Automated data load testing.
5. Implemented two of kipi’s accelerators : RBAC accelerator and Monitoring Accelerator.
6. Cross-KT sessions to make other team-members aware of our work.
7. Used a new project management tool for team recognition called “Assembly”.

*Technical -*

1. Implemented a bunch features of Snowflake and learned to use them very well:
   1. Okta Integration for Snowflake and Tableau
   2. Snowpipe with error handling
   3. Snowflake dynamic data masking for columns
   4. Snowflake Stored Procedures and tasks
   5. External Functions with Lambda
   6. Data sharing to Reader Accounts
   7. Snowboard cataloging/lineage
2. Used Matillion to the full extent for:
   1. Data ingestion
   2. Data transformation
   3. Calling scripts
   4. Job logging
   5. Sending Mails
3. Learnt how to use new tools like AWS Rekognition, Mulesoft, Dataiku and many more

# Future Scope

1. Get google rating and implement sentiment analysis on google reviews on college.
2. Design a pricing plan depending on the features implemented
3. Implement email notification feature for drop-out students, and get reasons for drop-outs, which can be used to analyze what are the main reasons for which students are leaving.