

HEVO Customer Experience (CXE)

Major project report submitted in fulfilment of the requirement for the degree of
Bachelor of Technology

in

Computer Science and Engineering

By

Diwakar Srivastava (181330)

UNDER THE SUPERVISION OF

Prof. Praveen Modi

&

Mr. Arun Sunderraj



Department of Computer Science & Engineering and Information
Technology

**Jaypee University of Information Technology, Wahnaghat,
173234, Himachal Pradesh, INDIA**

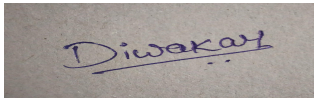
TABLE OF CONTENT

Content	Page No.
Declaration by Candidate	I
Certificate by Supervisor	II
Acknowledgment	III
Abstract	IV-V
TABLE OF FIGURES	VI
1. Introduction	9-17
2. Literature Survey	18-21
3. System Development	22-39
4. Performance Analysis	40-42
5. Conclusions	43
References	44

Candidate's Declaration

I hereby declare that the work presented in this report entitled “**Hevo Customer Experience**” in partial fulfilment of the requirements for the award of the degree of **Bachelor of Technology in Computer Science and Engineering/Information Technology** submitted in the department of Computer Science & Engineering and Information Technology, Jaypee University of Information Technology Waknaghat is an authentic record of my own work carried out over a period from February 2022 to May 30 under the supervision of (**Prof. Praveen Modi**) (Assistant Professor, Department of Computer Science and Engineering).

The matter embodied in the report has not been submitted for the award of any other degree or diploma.



(Student Signature)

DIWAKAR SRIVASTAVA

(181330)

This is to certify that the above statement made by the candidate is true to the best of my knowledge.

(College Supervisor Signature)

Supervisor Name:

Prof. Praveen Modi

Designation: **Asst. Professor**

Departmentname: **CSE**

Dated: **25/05/2022**



(Company Supervisor Signature)

Supervisor Name:

Mr. Arun Sunderraj

Designation: **Manager**

Department name: **CXE**

Dated: **25/05/2022**

CERTIFICATE

This is to certify that the work which is being presented in the project report titled “**HEVO Customer Experience**” in fulfilment of the requirements for the award of the degree of B.Tech in Computer Science And Engineering and submitted to the Department of Computer Science And Engineering, Jaypee University of Information Technology, Wagnaghat is an authentic record of work carried out by “**Diwakar Srivastava (181330)**” during the period from January 2021 to May 2021 under the supervision of **Prof. Praveen Modi**, Department of Computer Science and Engineering, Jaypee University of Information Technology, Wagnaghat and **Mr. Arun Sunderraj**, Manager(CXE) at Hevodata.

Diwakar Srivastava (181330)

The above statement made is correct to the best of my knowledge.

Prof. Praveen Modi

Assistant Professor (SG)

Department of Computer Science & Engineering and Information Technology

Jaypee University of Information Technology

&

Mr. Arun Sunderraj

Manager(CXE)

Hevodata

ACKNOWLEDGEMENT

Firstly, I express my heartiest thanks and gratefulness to almighty God for His divine blessing makes it possible to complete the project work successfully.

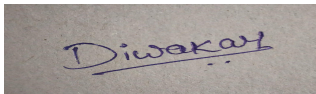
I am really grateful and wish my profound indebtedness to Supervisor **Prof. Praveen Modi** Department of CSE Jaypee University of Information Technology, Wakhnaghat and **Mr. Arun Sunderraj**, Manager(CXE) at Hevodata. Deep Knowledge & keen interest of my supervisor in the field of “ETL” to carry out this project. His endless patience, scholarly guidance, continual encouragement, constant and energetic supervision, constructive criticism, valuable advice, reading many inferior drafts and correcting them at all stages have made it possible to complete this project.

I would like to express my heartiest gratitude to Prof. Praveen Modi Department of CSE and Mr. Arun Sunderraj, Manager(CXE) at Hevodata, for his kind help to finish my project.

I would also generously welcome each one of those individuals who have helped me straightforwardly or in a roundabout way in making this project a win. In this unique situation, I might want to thank the various staff individuals, both educating and non-instructing, which have developed their convenient help and facilitated my undertaking.

Finally, I must acknowledge with due respect the constant support and patients of my parents.

Diwakar Srivastava(181330)

A rectangular box containing a handwritten signature in blue ink that reads "Diwakar".

ABSTRACT

Hevo is a no-code data mobility platform that may be used by both technical and non-technical users and business users.

In every data analytics project, HEVO's near-real time data integration technology minimises data integration issues. With the Hevo platform, you may quickly and easily set up your database or data warehouse for analytics. Simply said, Hevo allows you to concentrate on data analytics without having to worry about data integration.

Hevo's multi-tenant platform is built on Amazon's AWS cloud, which includes various components. The platform is built to process billions of records and can scale up or down automatically based on workload demands. Hevo's architecture guarantees that system resources are used to their full potential, giving you the highest return on your investment.

Hevo's software is supplied via an exceedingly user-friendly interface that removes the need for expert staff to set up and manage your data Pipelines. Hevo's approach to design extends beyond data Pipelines. Its data transformation tools are nicely integrated within the platform, making analytics chores much easier to do.

Hevo offers more than 100 pre-built connectors for databases, SaaS applications, cloud storage, SDKs, and streaming services. You may duplicate data from any of these Sources to a database or data warehouse of your choosing in only five minutes.

Models and Workflows can help you get the data into an analysis-ready state, where the powerful Python-based and drag-and-drop Transformations can help you cleanse and prepare the data to be imported to your Destination.

Activate, Hevo's no-code reverse ETL (Extract-Transform-Load) solution, loads data from your data warehouse into multiple SaaS apps without requiring any coding. This improves the data in your departments' respective software, such as Salesforce for sales, allowing for data synchronisation and access between departments and team members. If you're utilising Hevo

Pipelines to load data from diverse sources into the Destination warehouse, Hevo and Hevo Activate work together to provide a bi-directional Pipeline for your data-driven businesses.

TABLE OF FIGURES

Content	Page No.
1. HEVO	09
2. Hevo Overview	11
3. Process of Data Loading	13
4. Customer Remarks About Hevo	14
5. Customer Remarks About Hevo	15
6. Customer Remarks About Hevo	15
7. Customer Remarks About Hevo	15
8. Workbench	19
9. SQL query	19
10. Difference between Python and Jython	20
11. UI to use Jython for pre-load Data Transformation	21
12. Types of Issues	23
13. Issue-> Source tag	23
14. Issue-> Destination tag	24
15. Tag Generator	25

Chapter 01

INTRODUCTION

1.1 Introduction

1.2 Problem Statement

1.3 Objectives

1.4 Motivation

1.5 Methodology

1.1. INTRODUCTION

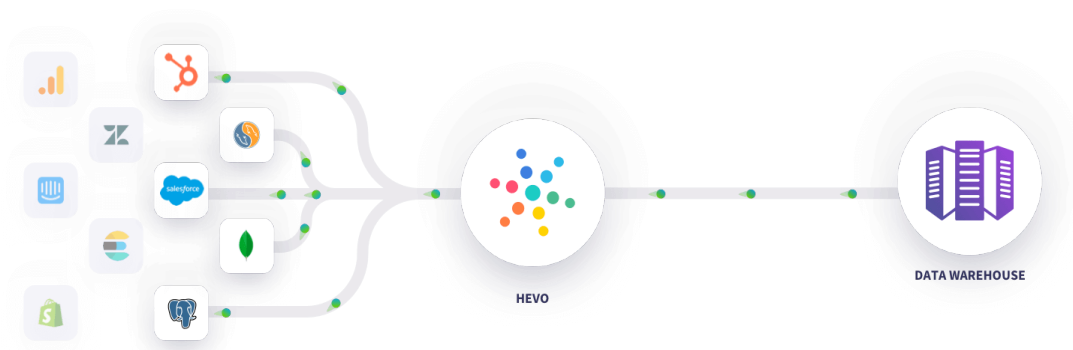


Figure 1. Hevo

Hevo is a no-code data replication platform that can be used by both technical and non-technical users and business users.

In every data analytics project, Hevo's near-real time data integration technology minimises data integration issues. With the Hevo platform, you can quickly and easily set up your database or data warehouse for analytics.

Simply said, Hevo allows you to concentrate on data analytics without having to worry about data integration.

Hevo offers more than 100 pre-built connectors for databases, SaaS applications, cloud storage, SDKs, and streaming services. You can duplicate data from any of

these Sources to a database or data warehouse of your choosing in only five minutes.

Hevo Supports over 100 sources from where users can replicate their data, some of those data sources are:

- Analytics: Magento, PrestaShop, Shopify, WooCommerce, WordPress
- Collaboration: Asana, Github, Jira
- CRM: Freshdesk, Intercom, Salesforce, Zendesk
- Data Warehouses: Amazon Redshift, Google BigQuery
- Databases: Local Database, Amazon DynamoDB, Elasticsearch, MongoDB, MySQL, Oracle, PostgreSQL, SQL server
- E-commerce: Magento, PrestaShop, Shopify, WooCommerce, WordPress
- File Storage: File Log, Amazon S3, FTP/SFTP, Google Cloud Storage, Google Drive, Google Sheets
- Finance and Accounting: QuickBooks Online, Stripe, Xero
- Marketing: AdRoll, Apple Search Ads, AppsFlyer, Criteo, Facebook Ads, Facebook Page Insights, Google Ads API, Google AdWords, Google Campaign Manager, Google Play Console, Google Search Console, Hubspot, Instagram Business, Klaviyo, LinkedIn Ads, Mailchimp, Microsoft Advertising, OutBrain, Pardot, Pinterest Ads, Segment, SendGrid, SendGrid Webhook, Salesforce Marketing Cloud, Snapchat Ads, Taboola, Twitter Ads
- Streaming: Android SDK, Kafka, REST API, Webhook
,etc.

Along with these sources Hevo supports multiple destinations where user can replicate their data, some of the major Destinations that are supported by Hevo are:

- Databases: Amazon Aurora, Microsoft SQL Server, MySQL, PostgreSQL, Local Database
- Data Warehouses: Amazon Redshift, Databricks, Firebolt, Hevo Managed Google BigQuery, Snowflake

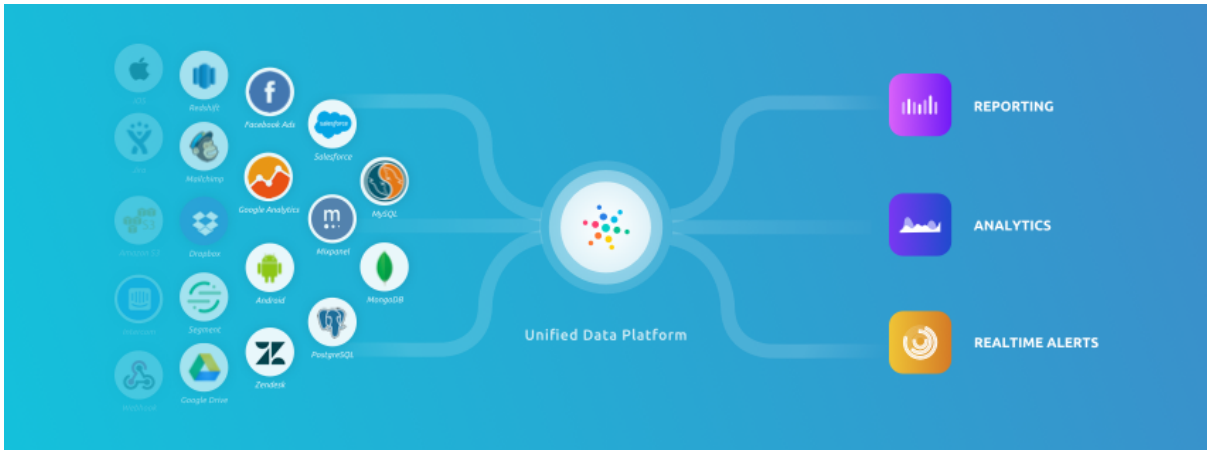


Figure 2. Hevo overview

1.1.1. DATA INGESTION:

Hevo supports three different types of data ingestion modes: log based, Table and Custom SQL. Each mode offers different configurations and settings. These include control over type and amount of data to be ingested.

Pipeline Mode	Description
Log-Based	Data is read from the logs maintained by the Source for each transaction, such as, addition, deletion or update of records. This mode applies only to database sources.
Table	Data is read from tables.
Custom SQL	Data is ingested based on the SQL queries provided by the user.

Table 1.1 Ingestion Modes

1.1.2. DATA LOADING:

At each run of your Pipeline, the data ingested from the Source is placed into the Destination warehouse. If your Events quota is used up, the Events in the Pipeline are held in a data warehouse until you buy more Events, at which point they are replayed. To learn about your Source's replication approach, see the data replication section.

Hevo keeps any primary keys defined in the Source data in the Destination tables by default.

We can load both types of data:

- Data without primary keys
- Data with primary keys

1.1.2.1 Data With Primary Keys:

If primary keys are present in the Source data but not enforceable on the Destination warehouse, as in the case of Google BigQuery, Amazon Redshift, and Snowflake, then, ensuring uniqueness of data is not possible by default. Hevo circumvents this lack of primary key enforcement and guarantees that no duplicate data is loaded to or exists in the Destination tables by:

- Adding temporary Hevo-internal meta columns to the tables to identify eligible Events,
- Using specific queries to cleanse the data of any duplicate and stale Events,
- Adding metadata information to each Event to uniquely identify its ingestion and loading time

Note: These steps utilise your Destination system's resources in terms of CPU usage for running the queries and additional storage utilisation for the duration of processing of the data.

1.1.2.2 Data Without Primary Keys:

If primary keys are not present in the Destination tables, Hevo directly appends the data into the target tables. While this can result in duplicate Events occurring in the Destination, there is no resource overhead stemming from the data loading process.

Loading Data to a Database

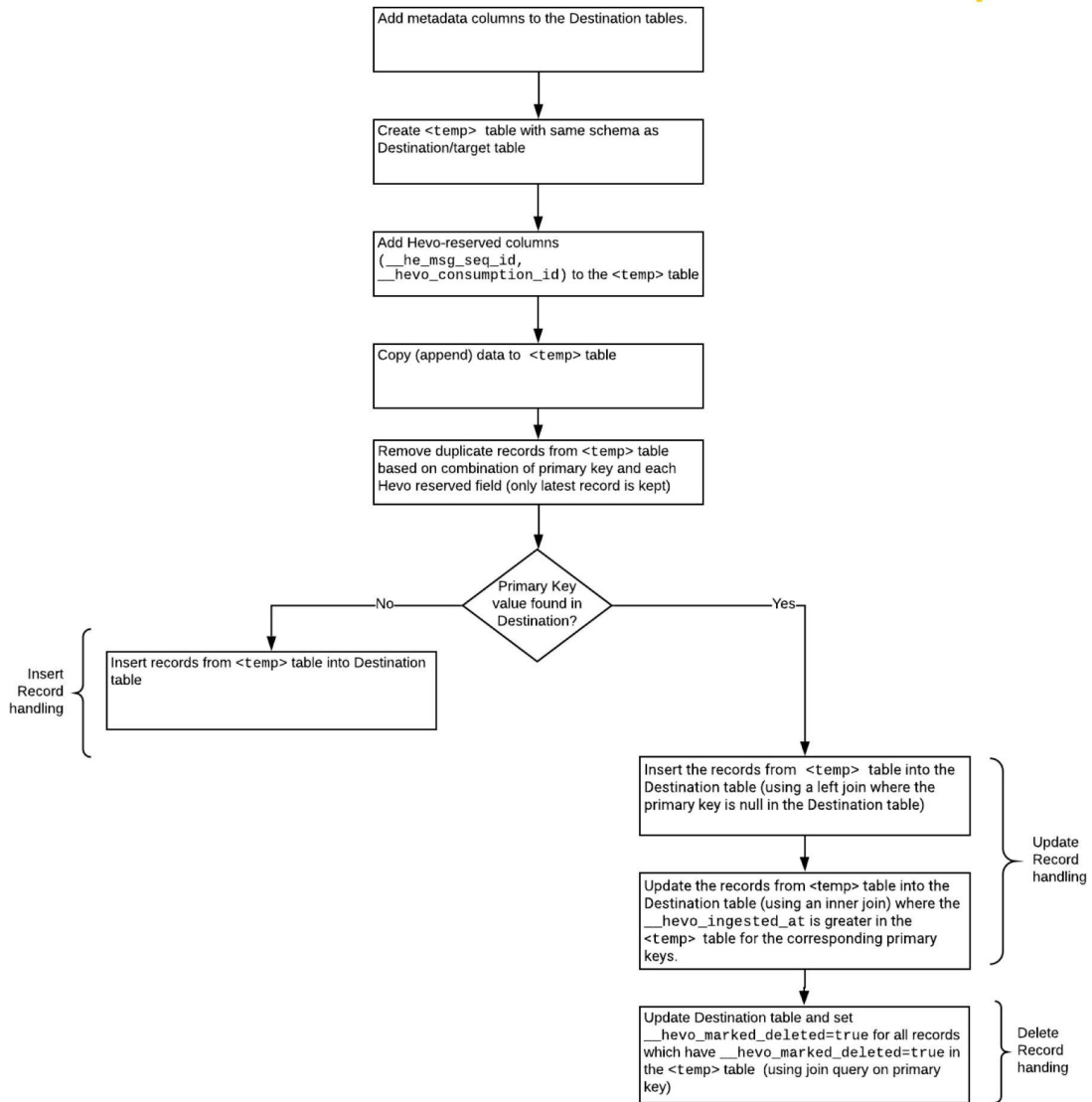


Figure 3. Process of Data Loading

1.2. PROBLEM STATEMENT:

While Hevo handles 100+ sources and dozens of destinations along with Hundreds of clients, These perks come with a major issue that is ‘Resolving User Queries, provide user’s a seamless and comfortable experience’, to mainly facilitate this aspect of product hevo have a designated team of talented individuals which works 24*7 to provide a world class experience to our users and resolve their queries related to any aspect of our product as quickly as we can.

The name of that team is ‘Hevo Customer Experience Team’.

1.3. OBJECTIVE:

- To provide quality product support to the client
- Improve user experience
- Solve user queries
- Handle product Issues

1.4. MOTIVATION:

The motivation to take Hevo to greater heights comes from the response we get from our customers. Their valuable comments have been a source of inspiration for everyone at hevo and works as a driving force for us to achieve the impossible.

Here are some of the examples what drives us forward and keeps us motivated:

“We reviewed tens of ETL tools and Hevo had everything we were looking for. Their support is top-notch and the way all their tools work together is seamless. We are so happy to have found them.”



David Goodman
Manager, Data & Analytics

“Hevo is very flexible compared to other tools. It allows us to handle all exceptions and custom use cases effortlessly. This ensures our data moves seamlessly from all sources to Redshift, enabling us to do so much more with it.”



Chushul Suri
Head Of Data Analytics

“We love Hevo! We tried many tools for migrating our data from MongoDB into Snowflake, but none were as reliable, flexible, and cost-effective as Hevo. Plus, their customer service is excellent.”



Matt Thomas
CTO

“The combination of Hevo and Snowflake has worked best for us. One of the biggest reasons why I would recommend Hevo is because of its lowest price-performance ratio as compared to the competition. It is definitely one of the best solutions if we take into consideration 3 major aspects - scalability, productivity, and reliability.”



Emmet Murphy
Staff Software Engineer

Figure 4,5,6,7. Customer Remarks about Hevo

1.5. METHODOLOGY

Whenever a user reports any issue or puts up a query, we at Hevo Customer Experience team have to follow a certain procedure while solving that ticket, here is a brief description on how we handle a ticket when it comes in our L1 queue:

- **Identify request type**

The first step is always to identify what the client's request is and under which category it falls.

It can be an Issue or a Query, If it's an issue it can be further classified into six different categories (which I'll explain in Chapter-3). if its Query, it can be related to pipeline or just normal product related information.

- **Ask for required informations**

After identifying the request type of client we ask for related information about the request, Example: If request is regarding Delay in Data Load in one of the pipeline of users, we ask the user about the pipeline details so that we can do further RCA.

- **Verify the user request if its correct**

The next step is to verify what the client claims is true or not, many of our clients are people who are from a non-technical background and sometimes what may seem to them as an error or bug might just be a normal product behaviour, so we need to verify the client request.

Taking the example from the last point, if its Delay is Data load we verify it from UI, Load Status Page and we cross-check it using our Internal tool Grafana.

- **Search hevo docs**

After verifying client requests, we search in our product documentation for the steps prescribed to solve the problem. Hevo has one of the best product documentation in the market and most of the user queries can actually be solved by just going through Hevo Docs.

- **Search confluence for similar documentation**

Confluence has our Knowledge Base where teams keep records of different cases that

have arrived over any particular error or user query. Using KB we can give clients a tested and trusted solution for their problem.

- **Search Intercom for previous similar tickets**

Intercom is our ticketing solution which is our first contact point with users, when solving user queries any L1 executive can search Intercom using keywords for any previous cases and use the solution provided there to solve user queries faster.

- **Google Search**

For solving user query this is mainly last step and not usually performed, but at Hevo support more than 100+ sources sometimes the issue user faces is not exactly because of our product but can be a limitation of source also, for those cases we need to do a google search and provide our findings to user after verifying it with internal team.

- **Escalation**

Many times the issue the user is facing can be out of the expertise of the L1 executive and for those cases, Hevo has a full fledged mechanism for escalation of tickets to designated teams or executives who are well experienced in handling those cases.

Example: If what the user faces is a problem related to Sign-up, we escalate it to Sales team using the Hubspot task.

If what the user faces is a problem related to pre-load transformation of data, such tickets can be escalated to the L2 queue after getting verification for L2 executives in the Slack channel designated for Escalation.

Chapter 02

LITERATURE SURVEY

2.1. SQL

SQL is Structured Query Language, which is a computer language for storing, manipulating and retrieving data stored in a relational database.

SQL is the standard language for Relational Database Systems. All the Relational Database Management Systems (RDMS) like MySQL, MS Access, Oracle, Sybase, Informix, Postgres and SQL Server use SQL as their standard database language.

Also, they are using different dialects, such as –

- MS SQL Server using T-SQL,
- Oracle using PL/SQL,
- The MS Access version of SQL is called JET SQL (native format) etc .

1.5.1.1.Applications of SQL

As mentioned before, SQL is one of the most widely used query languages over databases.

I'm going to list few of them here:

- Allows users to access data in the relational database management systems.
- Allows users to describe the data.
- Allows users to define the data in a database and manipulate that data.
- Allows embedding within other languages using SQL modules, libraries & pre-compilers.
- Allows users to create and drop databases and tables.
- Allows users to create views, stored procedures, and functions in a database.
- Allows users to set permissions on tables, procedures and views.

2.1.1. How HEVO uses SQL:

Hevo uses SQL to fetch data from different sources as well as uses it to help users retrieve data in the workbench.

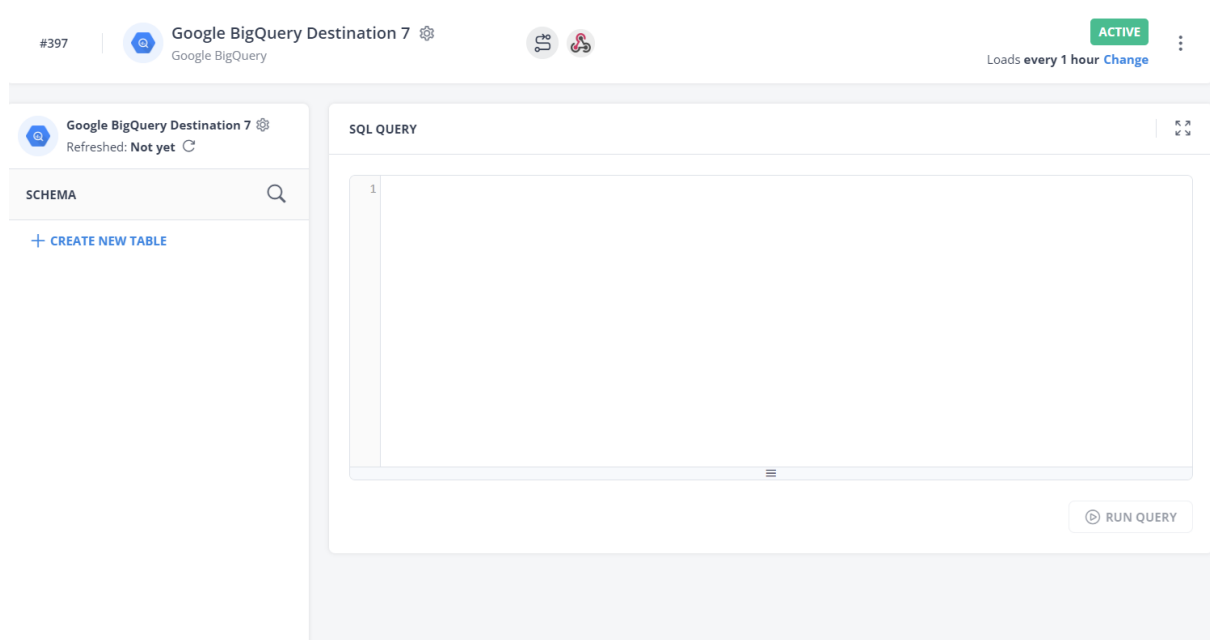


Figure 8. Workbench

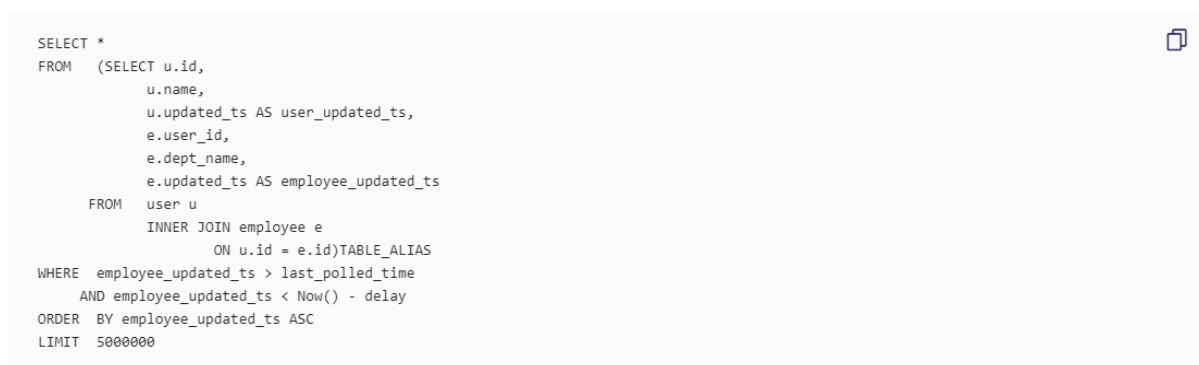


Figure 9. SQL query

2.2. JYTHON

Jython is a Java implementation of Python that combines expressive power with clarity. Jython is freely available for both commercial and non-commercial use and is distributed with source code under the PSF Licence v2. Jython is complementary to Java and is especially suited for the following tasks:

Embedded scripting - Java programmers can add the Jython libraries to their system to allow end users to write simple or complicated scripts that add functionality to the application.

Interactive experimentation - Jython provides an interactive interpreter that can be used to interact with Java packages or with running Java applications. This allows programmers to experiment and debug any Java system using Jython.

Rapid application development - Python programs are typically 2-10x shorter than the equivalent Java program. This translates directly to increased programmer productivity. The seamless interaction between Python and Java allows developers to freely mix the two languages both during development and in shipping products.

Here is an example of running Python code inside a simple Java application

```
import org.python.util.PythonInterpreter;

public class JythonHelloWorld {
    public static void main(String[] args) {
        try(PythonInterpreter pyInterp = new PythonInterpreter()) {
            pyInterp.exec("print('Hello Python World!')");
        }
    }
}
```

Here is an example of using Java from Python code

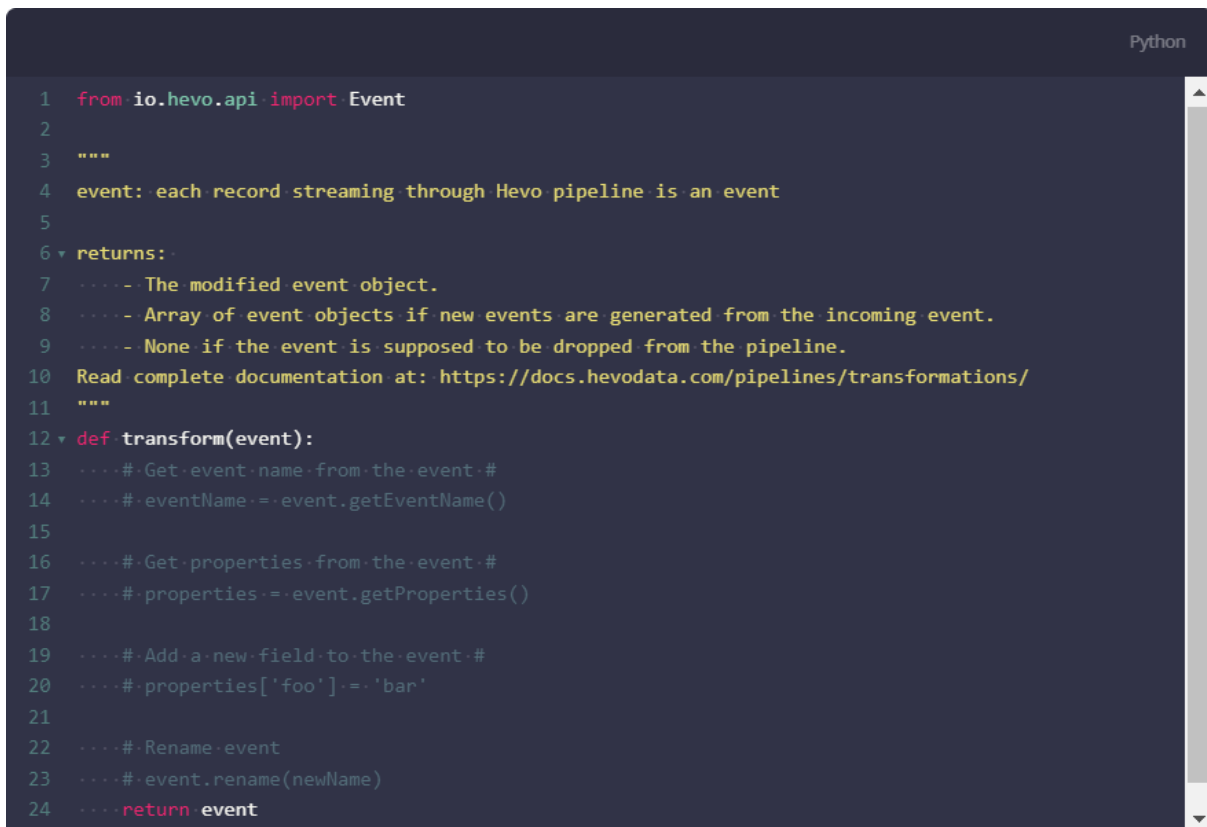
```
from java.lang import System # Java import

print('Running on Java version: ' + System.getProperty('java.version'))
print('Unix time from Java: ' + str(System.currentTimeMillis()))
```

Figure 10. Difference between Python and Jython

2.2.1. HISTORY OF JYTHON

Jython was created in 1997 to replace C with Java for performance intensive code accessed by python programs. It was given a grant by Python Software Foundation in January 2005 and Jython 2.5 was released in June 2009.

A screenshot of a Python IDE window titled "Python". The window contains a Python script with the following code:

```
1 from io.hevo.api import Event
2
3 """
4 event: each record streaming through Hevo pipeline is an event
5
6 returns:
7     - The modified event object.
8     - Array of event objects if new events are generated from the incoming event.
9     - None if the event is supposed to be dropped from the pipeline.
10 Read complete documentation at: https://docs.hevodata.com/pipelines/transformations/
11 """
12 def transform(event):
13     # Get event name from the event #
14     # eventName = event.getEventName()
15
16     # Get properties from the event #
17     # properties = event.getProperties()
18
19     # Add a new field to the event #
20     # properties['foo'] = 'bar'
21
22     # Rename event
23     # event.rename(newName)
24     return event
```

Figure 11. UI to use Jython for pre-load Data Transformation

Chapter 03

SYSTEM DEVELOPMENT

3.1. PROCEDURE OF TICKET HANDLING

Intercom is our first point of contact with our client and whenever a ticket comes into the L1 queue, it is assigned to L1 executives in round-robin order, which ensures that everyone gets a similar amount of workload.

Once a ticket has been assigned to an L1 executive, he/she follows these certain steps to solve a ticket while making use of our internal tools.

- **Identify request type**

The first step is always to identify what the client's request is and under which category it falls. It can be an Issue or a Query, If it's an issue it can be further classified into six different categories (which I'll explain in Chapter-3). if its Query, it can be related to pipeline or just normal product related information.

- **Ask for required informations**

After identifying the request type of client we ask for related information about the request, Example: If request is regarding Delay in Data Load in one of the pipeline of users, we ask the user about the pipeline details so that we can do further RCA.

- **Putting related tags to the conversation**

Once required information is collected, conversation needs to be tagged accordingly. If it's an Issue, Issue tag should be applied along with the Issue tag, it must specify what type of issue the client is facing and should be tagged accordingly.

Hevo classifies issues in mainly six categories:

- Delay in Data Load
- Volume in Data mismatch
- Data Load not started
- Incorrect Structure of data loaded
- Issue with transformation
- Others

Primary

Issue

Secondary

Issue

Tertiary

--

--

- Delay in data load
- Volume of data mismatch
- Data load not started
- Incorrect structure of data loaded
- Issue with transformation
- Others

Figure 12. Types of Issues

Along with these tags, Tags related to Source and Destination must be applied.

--

- Ad Roll
- Amazon S3
- Amplitude
- Apple Search Ads
- Appsflyer
- Asana
- BigQuery
- Campaign Manager
- Clevertap
- Criteo
- DataBricks
- Dynamo DB
- Elastic Search
- FB Ads
- FB_Pages
- FireBase Analytics
- FireBolt
- Freshdesk

--

Issue - Source -

COPY

Figure 13. Issue->Source tag

Primary

Issue ▼

Secondary

Destination ▼

Teritary

-- ▼

-
- MySQL
- MS-SQL
- Postgres
- Redshift
- Snowflake
- BigQuery
- Databricks
- FireBolt
- Amazon S3
- Others

Figure 14.Issue->Destination tag

Similar to Issue, if a user's ticket comes under the category of Query, required and appropriate tags should be applied accordingly.

To apply tickets to user conversation on Intercom we use Tag-Generator chrome extension.

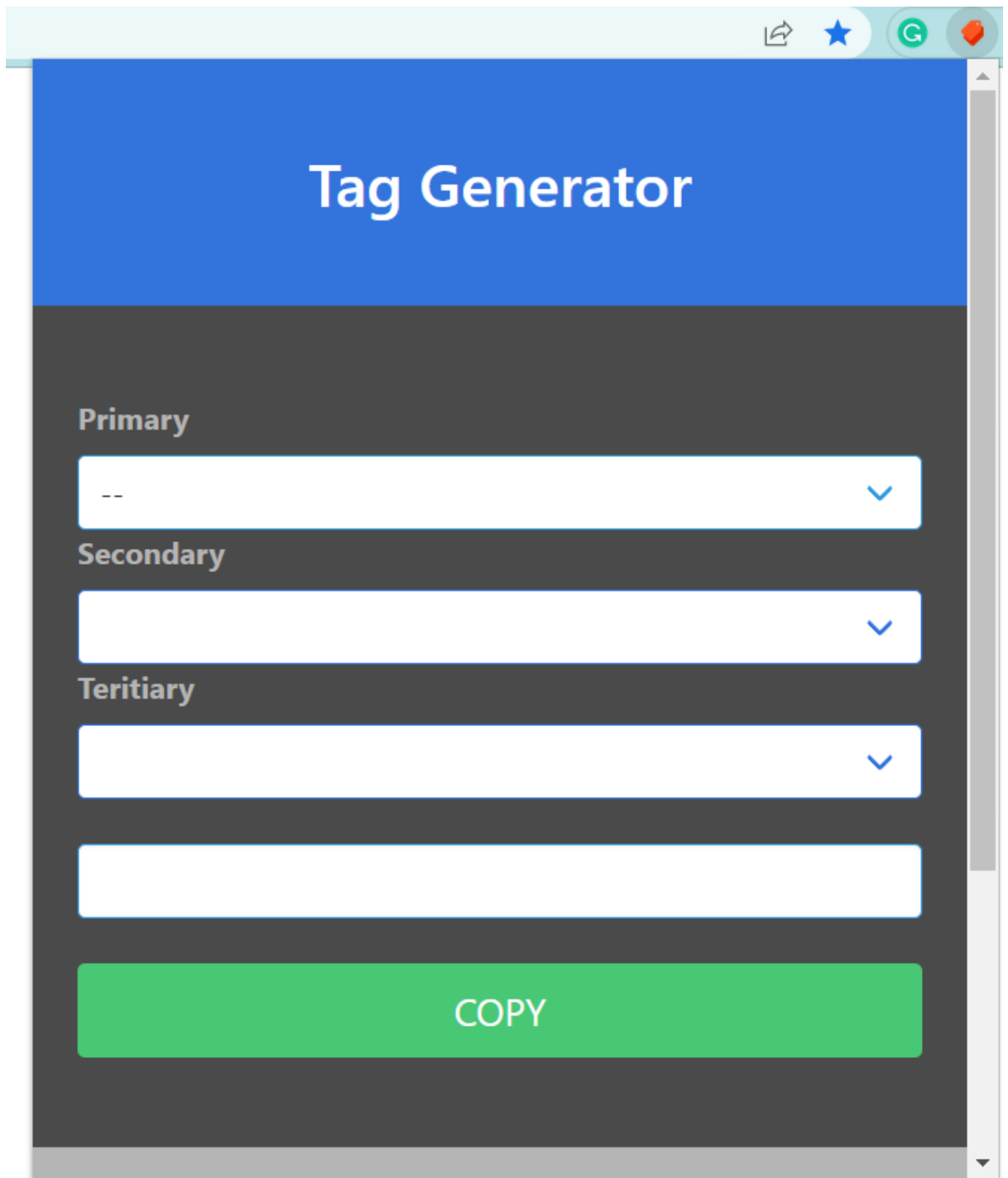


Figure 15. Tag Generator

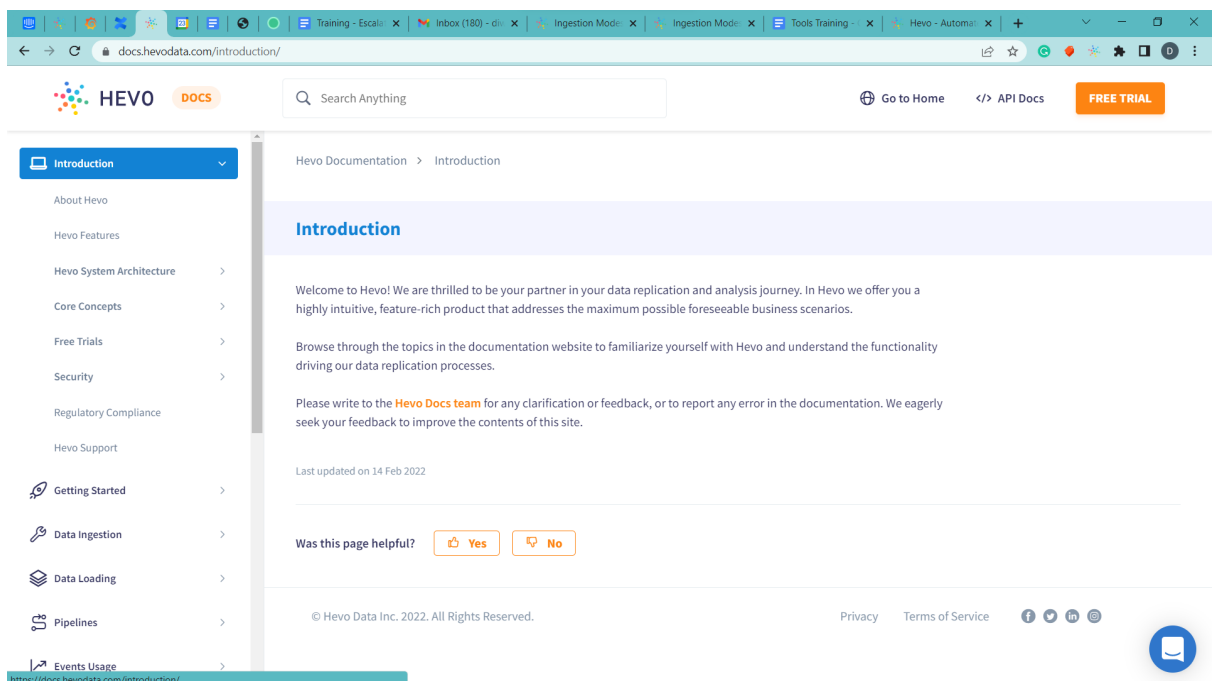
- **Verify the user request if its correct**

The next step is to verify what the client claims is true or not, many of our clients are people who are from a non-technical background and sometimes what may seem to them as an error or bug might just be a normal product behaviour, so we need to verify the client request.

Taking the example from the last point, if its Delay is Data load we verify it from UI, Load Status Page and we cross-check it using our Internal tool Grafana.

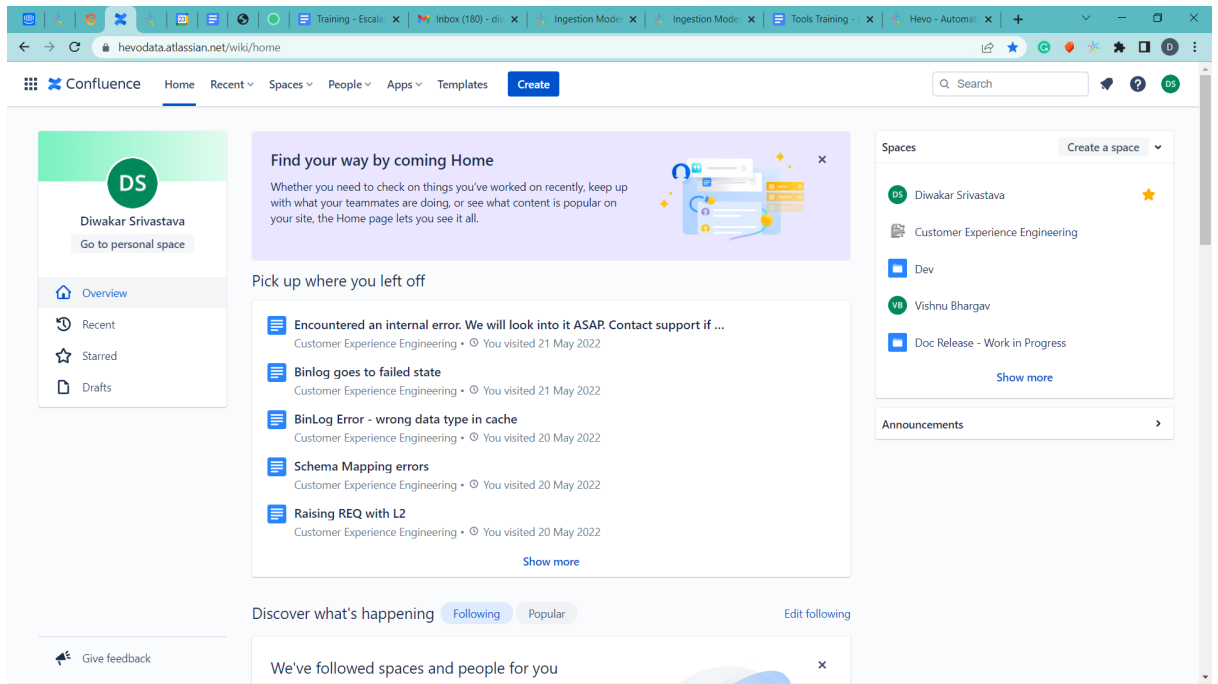
- **Search hevo docs**

After verifying client requests, we search in our product documentation for the steps prescribed to solve the problem. Hevo has one of the best product documentation in the market and most of the user queries can actually be solved by just going through Hevo Docs.



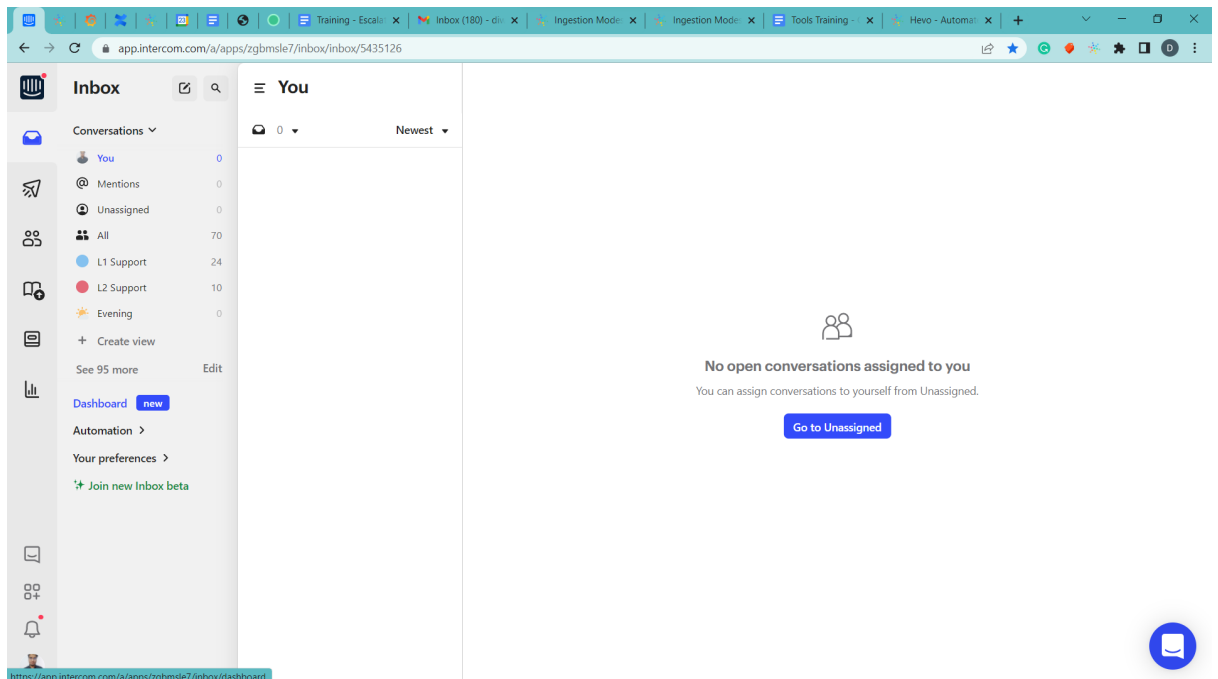
- **Search confluence for similar documentation**

Confluence has our Knowledge Base where teams keep records of different cases that have arrived over any particular error or user query. Using KB we can give clients a tested and trusted solution for their problem.



- **Search Intercom for previous similar tickets**

Intercom is our ticketing solution which is our first contact point with users, when solving user queries any L1 executive can search Intercom using keywords for any previous cases and use the solution provided there to solve user queries faster.



- **Google Search**

For solving user query this is mainly last step and not usually performed, but at Hevo support more than 100+ sources sometimes the issue user faces is not exactly because of our product but can be a limitation of source also, for those cases we need to do a google search and provide our findings to user after verifying it with internal team.

- **Escalation**

Many times the issue the user is facing can be out of the expertise of the L1 executive and for those cases, Hevo has a full fledged mechanism for escalation of tickets to designated teams or executives who are well experienced in handling those cases.

Example: If what the user faces is a problem related to Sign-up, we escalate it to Sales team using the Hubspot task.

If what the user faces is a problem related to pre-load transformation of data, such tickets can be escalated to the L2 queue after getting verification for L2 executives in the Slack channel designated for Escalation.

To get confirmation in Escalation group we need to put a request, format of the request is as follows;



L2 Support Request



Account Name

Account Env

Is that Premium Customer? (Monthly Spend > 999\$)

Conversation Type

If Issue, Issue Type?

Query/Issue Brief

Any Preliminary findings?

Close

Submit

Once this request is approved, we can use L2 escalation macro in Intercom and escalate the ticket to the L2 queue.

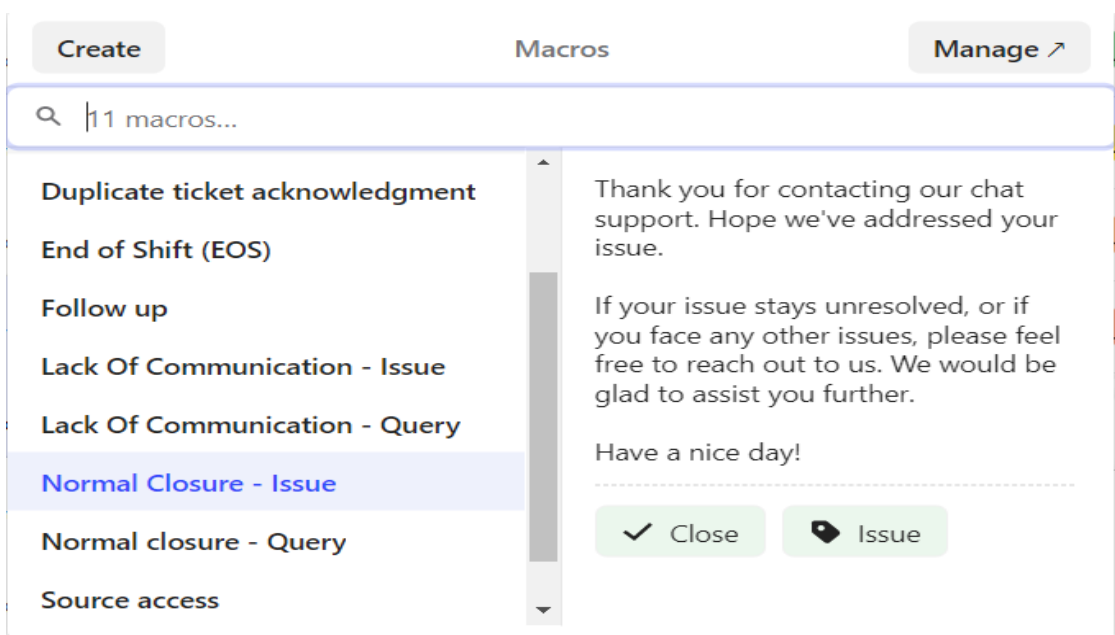


- **Successful solution of Ticket**

If the user query/ issue has been successfully solved, we inform the user and tag the message where we have informed the user or have provided the solution and as ‘Resolution Provided’.

Later on we ask the user if he/she still has any other query or issue or we are good to close this ticket. If the user confirms that he/she doesn’t have any further queries/issues we close the ticket with related macros.

For the Issue, we use the Normal Closure Issue.



For Query, we use Normal Closure Query.

Create Macros Manage ↗

🔍 11 macros...

<p>Duplicate ticket acknowledgment</p> <p>End of Shift (EOS)</p> <p>Follow up</p> <p>Lack Of Communication - Issue</p> <p>Lack Of Communication - Query</p> <p>Normal Closure - Issue</p> <p>Normal closure - Query</p> <p>Source access</p>	<p>Thank you for contacting our chat support. I am now closing this chat. If you have any more issues, please don't hesitate to re-open or start a new one.</p> <p>Have a great day! 😊</p> <hr style="border-top: 1px dashed #ccc;"/> <p>✓ Close 🏷️ Query</p>
---	--

3.2. INTERNAL TOOLS

As we saw earlier, as a first check, we investigate the issue in the product thoroughly. There are several issues that can be solved or investigated well at the Product-level itself. We find the root cause and we guide the user accordingly. For example, if a high amount of events is getting loaded into the destination, we can simply compare sample destination values with the source values (provided by the user) or directly check duplicate values (using the count function of unique values in Workbench).

If duplicate data is showing in the destination, it may happen due to a missing primary key in the object (viewable in Schema Mapper of the object) or Append Rows on Update option enabled in the destination. We can guide the user accordingly.

However, there will be some cases for which we would need to further deep-dive into the issue and would require the use of some backend tools to pinpoint the root cause or the error for the issue that is happening in the pipeline.

In this training, we will understand each of these tools and their significance, how these tools can be used, and in which scenarios or issues a certain tool will be required.

Some of these tools are created in-house (like Alfred, Backend DB, Braavos, Hevo Team Switcher, etc.), while others are third-party tools (like Coralogix, Grafana, Postman, Atlassian Jira/Confluence, etc.)

3.2.1. Hevo Team Switcher

Use Case - This is a very essential tool that helps us to quickly switch between user accounts within a cluster from the same window. This skips the need to repeatedly log out and again log in to a new account every time. Please note that you can only switch between accounts belonging to the same cluster (us, us2, eu, in, asia, au) and cannot jump across accounts on a different cluster. For that, you will still need to manually go to the cluster website (xy.hevodata.com), log in through the demo account/user's company name account (reach out to your buddy on how to access the user's pipeline), and then use the switcher.

- View important parameters like **Team ID**, **Integration ID**, etc. which are relevant when using tools like Alfred, Coralogix, etc. as we will see below.
- This is a Google extension developed in-house by the dev team.

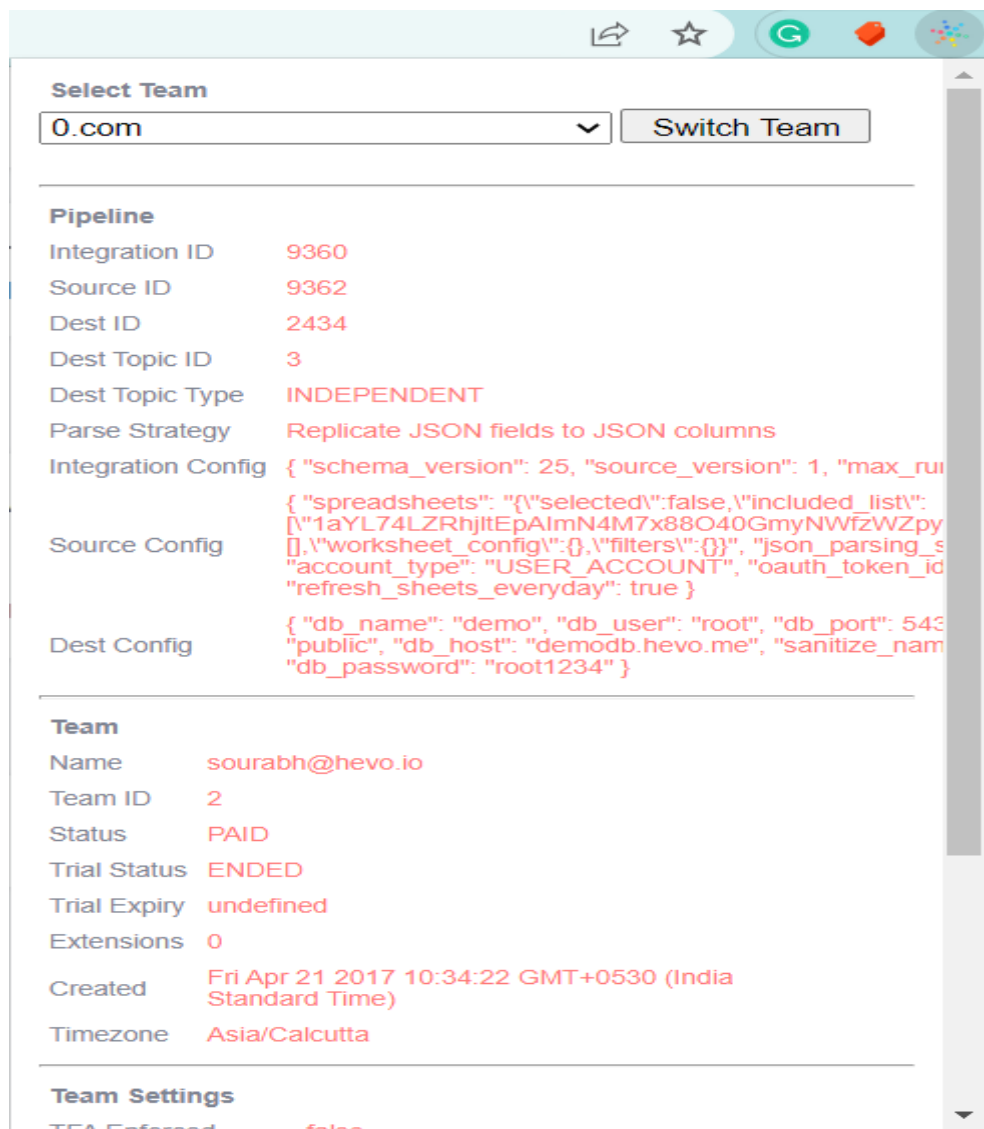


Figure . Hevo Team Switcher

3.2.2. Alfred

- **Use Case** - This tool is essential to access all the essential backend features related to a user's environment. This is used in multiple scenarios, some of which include -
 - Querying source and destination data - Use SQL queries similar to how we do with Destination Workbench. For Source querying, we need to get consent from the user to provide temporary access. This is required on a need basis (only if authorised by L2 or the on-call team) whereas the destination can be queried at any time.
 - Billing-related functions - To extend the Free trial period for a trial user, waive event quota consumed by Hevo due to an internal bug, etc.
 - Account-related actions - Checking and verifying the email address of the user, checking detailed pipeline usage summary (by events and pipelines)
 - Investigating issues in a pipeline using Backend databases (eg, checking connector's tasks, sidelined events tables)
 - Changing pipeline settings from the backend (eg, enable disabled parsing strategies, increase the model timeout, increase WAL timeout, etc.) - This function is allowed only under the authorization of the L2 team.
 - Test Database and SSH connection - Self-explanatory - We put the source credentials and check the health of the connection.

→ Query Services in Alfred

- In Hevo, we maintain Backend DBs that capture crucial backend information of pipelines on their operations and functionalities. There is a lot of information that we can get here, a few examples are mentioned below -
- Timestamps when the data was polled, ingested, processed, or loaded by pipeline
- Object status (Active, paused, scheduled, skipped, included, etc.)
- Internal error messages for any object
- We refer to these tables in Query Services mainly to investigate the issue

happening in the pipeline from the backend. We check the behaviour of the pipeline at the object level - when was the last time the object was processed, what was the last offset, is there any error message that has come up for the object. Accordingly, we take any one of the following routes -

- Search for the error on Google if we feel that the error is something due to source behaviours (eg, Data per call limit exceeded in API, connection error, etc.)
- Check the product (eg, event type mismatch - check Schema mapper, transformation error, etc.)
- Escalate the issue to the L2 team - If we find an error that is not getting solved or might need Dev intervention, we escalate the ticket to L2 with all our findings (Product check findings, a screenshot of the Query Services table, and the error showing in the backend, discrepancy in backend data, etc.)
- Mentioned below are some of the common Backend DBs that we check in Query Services of Alfred -

1. **Connectors Tasks** -

This is a common table that we use to check the object behaviour for the pipeline that is having the issue or throwing the error. To use this table, you will need to search for the connectors_tasks table and then search the respective pipeline ID as an SQL query.

For example, select * from connectors_tasks where integration_id = 4341;

2. **Sidelined file details** -

There may be issues when some events have been ingested by Hevo, but they were not loaded. This would mean that Hevo had sidelined these events due to some error. To check the same, we refer to the sideline_file_details table. For the particular object, we check the status and compare the status to the stage where it got sidelined.

3. **Apart from these two tables**, there are other important tables as follows -

- a. Security_users - Check account-related details of the user (Email verified or not, etc.)
- b. Handyman tasks - Check the status of individual objects (Processed, Failed,

Picked, etc.)

c. History_v2

d. Integration_source_objects - Check source ID for Event Tracker tool

These tables will need to be investigated whenever an issue from the associated area arises.

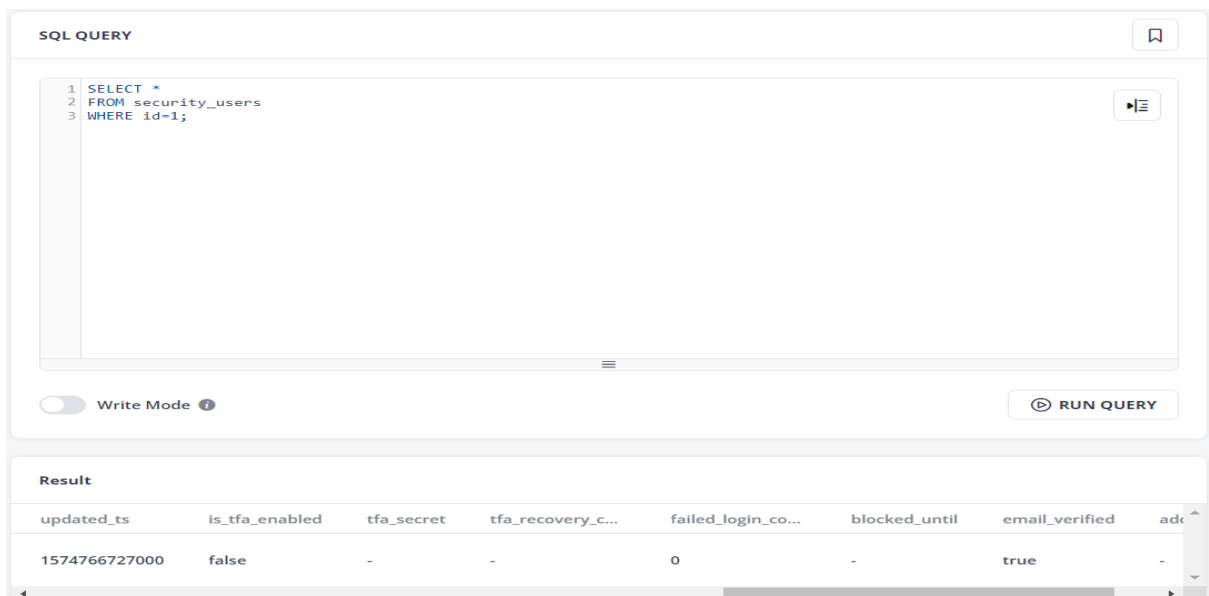
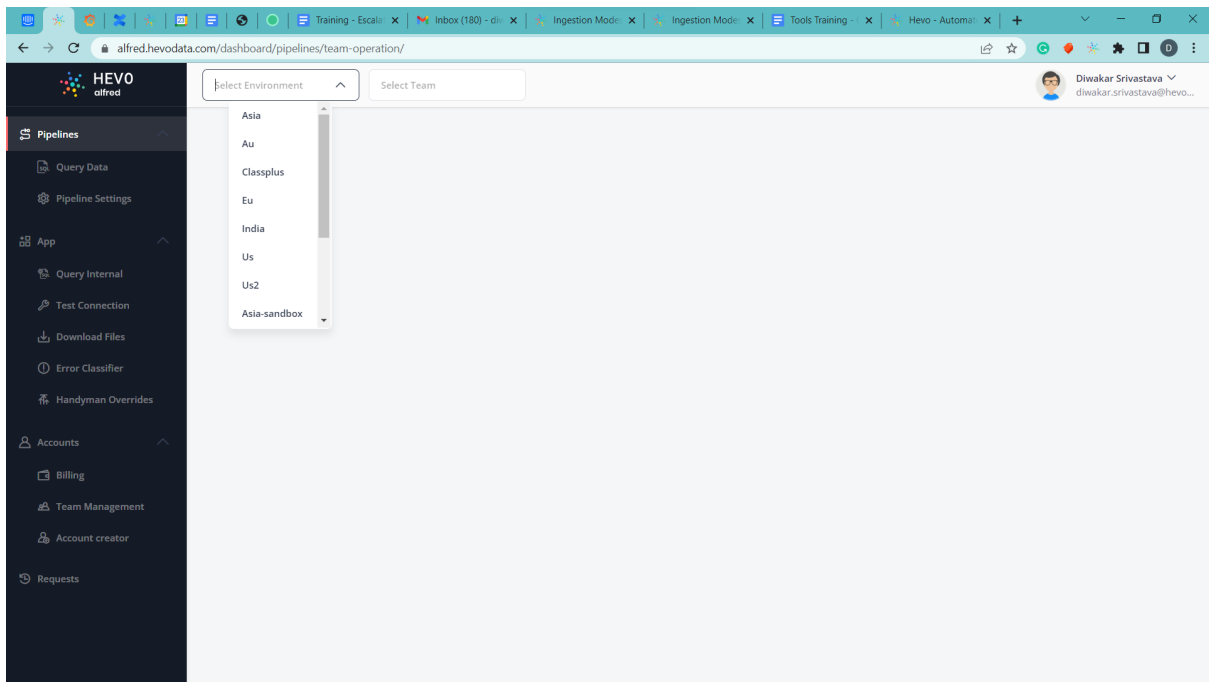


Figure. Alfred

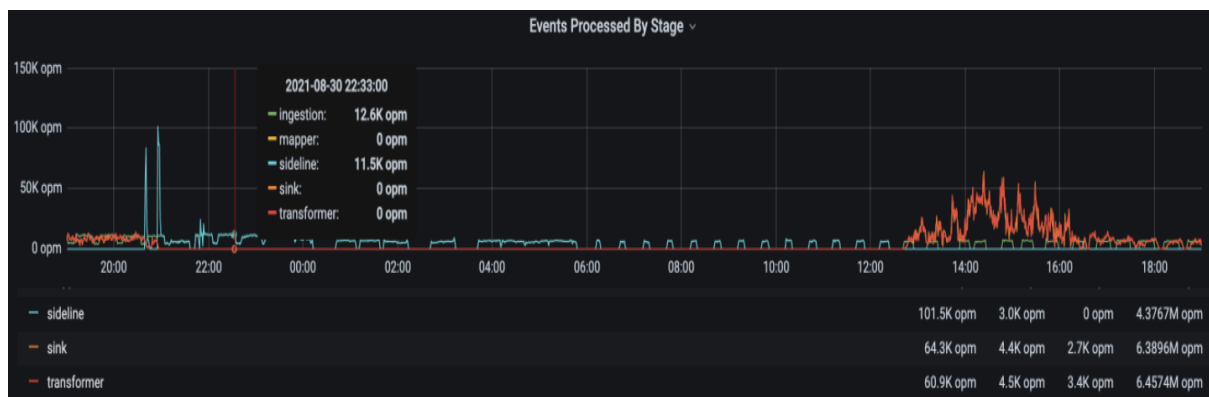
3.2.3. Coralogix

- **Use Case** - This tool is used to investigate any error logs associated with the issue. This is a very straightforward tool and is mostly used in conjunction with Alfred (when issues need to be further investigated to identify the root cause). Using this tool is very easy. You just need to search the integration_id of the pipeline in Coralogix, set the cluster and error filters, set the time filter to a wider timeframe, and check the result.
- **Resolution Approaches once the root cause is found** -
 - Share all your findings (entire error log as above and the coralogix link) with the L2 team during escalation.
 - Check the error on the internet using Google search and share a probable solution with the user (eg. connection error in databases, SaaS sources, etc.).
 - For example, in the above screenshot, the error is “Database connection failed when ending copy”. We can either search this error directly on Google and find a possible solution and share it with the user, or share the log and the coralogix link with the L2 team during escalation.

```
ERROR {
  exception: org.postgresql.util.PSQLException
  error_message: Database connection failed when ending copy
  hostname: us-dev-service-5
  level: ERROR
  release: v1.79.37
  logger: io.hevo.connectors.jdbc.postgres.logicalreplication.PostgresLogicalReplicationTaskV1
  throwable: org.postgresql.util.PSQLException: Database connection failed when ending copy
  at org.postgresql.core.v3.QueryExecutorImpl.endCopy(QueryExecutorImpl.java:1021)
  at org.postgresql.core.v3.CopyDualImpl.endCopy(CopyDualImpl.java:34)
  at org.postgresql.core.v3.replication.V3PGReplicationStream.close(V3PGReplicationStream.java:286)
  at io.hevo.connectors.jdbc.postgres.logicalreplication.PostgresLogicalReplicationTaskV1$PostgresStreamReader.proces
  ssStreamRead(PostgresLogicalReplicationTaskV1.java:488)
  at io.hevo.connectors.jdbc.postgres.logicalreplication.PostgresLogicalReplicationTaskV1$PostgresStreamReader.run(P
  ostgresLogicalReplicationTaskV1.java:385)
  at java.util.concurrent.Executors$RunnableAdapter.call(Executors.java:511)
  at java.util.concurrent.FutureTask.run(FutureTask.java:266)
  at java.util.concurrent.ThreadPoolExecutor.runWorker(ThreadPoolExecutor.java:1149)
  at java.util.concurrent.ThreadPoolExecutor$Worker.run(ThreadPoolExecutor.java:624)
  at java.lang.Thread.run(Thread.java:748)
  Caused by: java.net.SocketException: Connection or outbound has closed
  at sun.security.ssl.SSLSocketImpl$AppOutputStream.write(SSLSocketImpl.java:967)
  at java.io.BufferedOutputStream.flushBuffer(BufferedOutputStream.java:82)
  at java.io.BufferedOutputStream.flush(BufferedOutputStream.java:140)
  at org.postgresql.core.PGStream.flush(PGStream.java:663)
  at org.postgresql.core.v3.QueryExecutorImpl.endCopy(QueryExecutorImpl.java:1014)
  ... 9 common frames omitted
  client: us
  thread: PostgresStreamReader-02acc4d5d-bdf8-4656-bc36-26408fa7ab61-0
  message: Error while closing the stream for integration: 2961
}
```

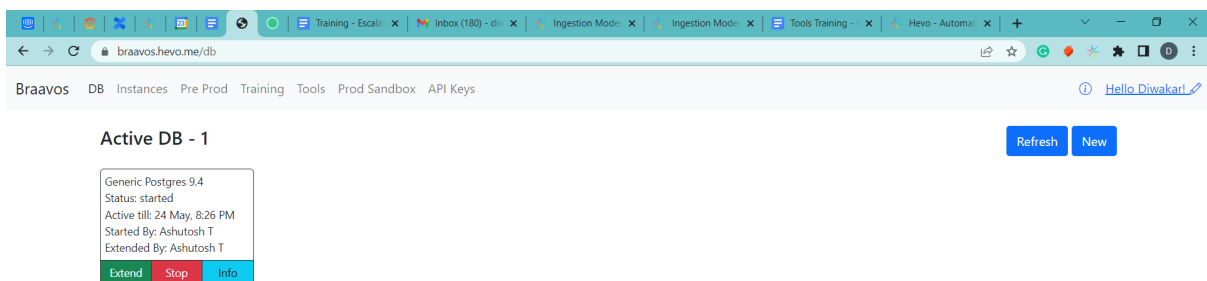
3.2.4. Grafana

- **Use Case** - Grafana is used to check the throughput (performance) of the pipeline in different stages. With the help of the graphs shown here, we can isolate the part which is causing the disturbance in the normal flow of events.
- This is an investigation tool (just like Alfred and Coralogix) and the finding from this tool will be needed to be shared with the L2 team while escalating the ticket.
- Some of the common issues where we need to check Grafana are as follows -
 - High latency in data movement in the pipeline - There is a lot of delay in data getting loaded
 - There is a high count of data getting loaded to the destination. We can check the spike in the pipeline throughput and share it with the L2 team during escalation. Sample as follows -



3.2.4. Braavos

- **Use Case** - Braavos is an internally built tool (just like Alfred) with the help of which we can create demo simulations of pipeline environments, databases, and SSH instances.
- This tool is useful when we want to simulate a demo pipeline with demo DB and validate a product behaviour. For example, if a user is facing duplication issues, we want to validate the working solution by recreating the same scenario in a demo pipeline and following the solution steps. Accordingly, you will be in a better position to guide the user on the working solution.



3.2.5. BackEnd DB Check

- We can access the Backend DB tables (same as those mentioned under *Query Services* in *Alfred*) using a tool such as Data Grip or dbVisualizer and query the information from there.
- Using a tool like Data Grip provides more intuitive and easier functionalities to query any data and hence, is sometimes preferable to using Alfred. You can still access these tables from *Alfred - Query Services*.

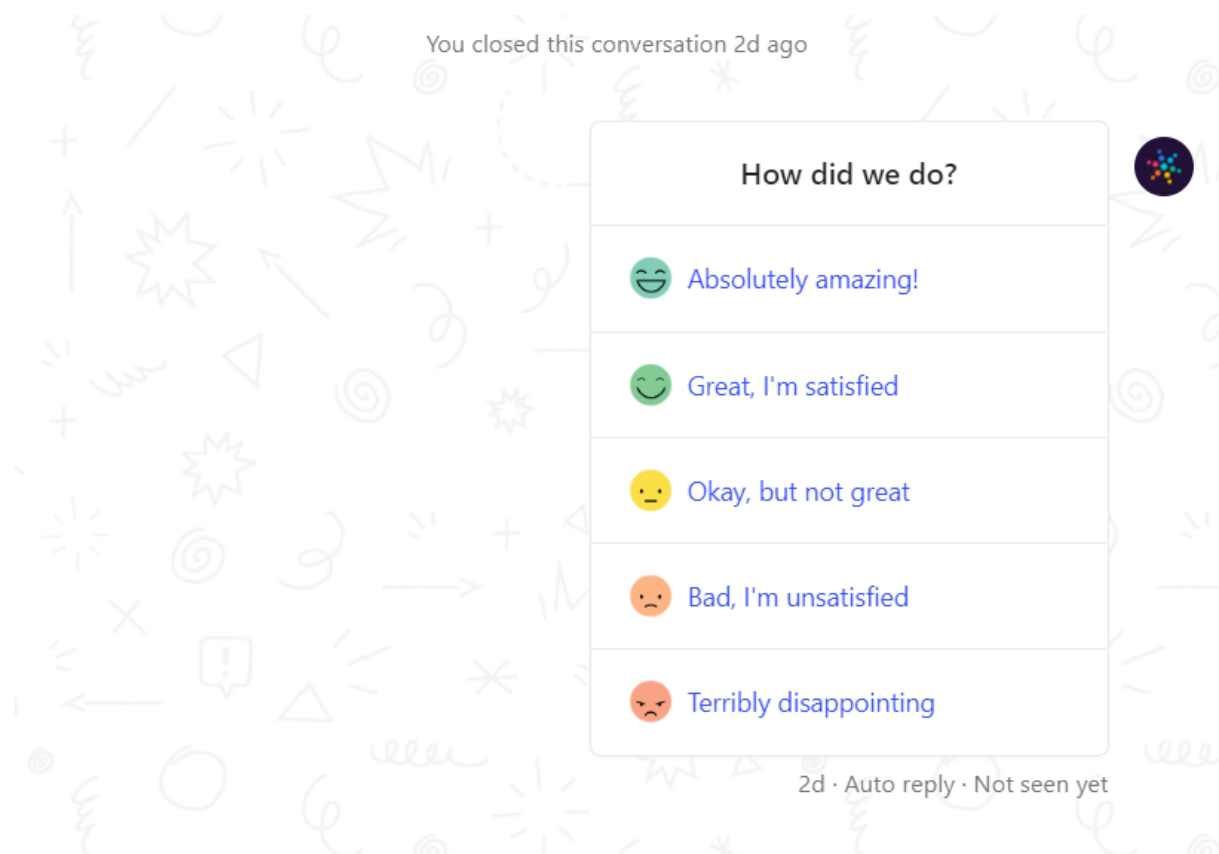
Chapter 04

PERFORMANCE ANALYSIS

4.1. EVALUATION MATRIX

There are various matrix on which a customer experience engineer is evaluated:

- CSAT: Also known as Customer Satisfaction rating, this is the rating that customer gives us after successful closure of a ticket, customer gives this rating based on his satisfaction with the service.



- Lead - Validated RP - 1: This is not a rating but a tag given to each closed conversation by the team lead, this tag depends on the factor that the resolution provided by the customer experience engineer is correct or not. By this tag the resolution provided in the conversation is considered correct and up-to the mark.



- Number of ticket closure: Another matrix on which CXE engineers are evaluated is number of tickets closed, 3.5 tickets per day or above is considered to be the minimum.
- Time taken to handle Issues and Queries: The allotted time to solve a Query is 1hour and to solve an issue is 2h, if issue is out of the scope of L1 engineer it needs to be escalated to L2 within the first 45 minutes.

Apart from these, the best performance analysis matrix for us is our satisfied customers and their comments about our product as well as our support team.

Some of the precious comments are:

“We reviewed tens of ETL tools and Hevo had everything we were looking for. Their support is top-notch and the way all their tools work together is seamless. We are so happy to have found them.”



David Goodman
Manager, Data & Analytics

“We love Hevo! We tried many tools for migrating our data from MongoDB into Snowflake, but none were as reliable, flexible, and cost-effective as Hevo. Plus, their customer service is excellent.”



Matt Thomas
CTO

4.2. FUTURE WORK

- **24*7 Customer Support :**

Currently Hevo, is trying to reach a point where it can provide world quality customer support 24*7, this is not a far-fetched dream and soon going to be a reality. Hevo management has been focusing on this for a longer period of time and very soon this will not be a future work but a reality.

- **Audio/Video Support :**

Currently Hevo provides customer support in the chat-based medium or over an e-mail conversation, sooner or later Hevo will be implementing a 24*7 world class Audio/Video support to its customers world-wide.

- **Adding more sources and improving product :**

Along with the support team, our sales and marketing as well as Dev team are working hard to make our product better, add more features, include more sources and reach out to more customers. The goal for this year is to reach 2000 paid customers and we still have a long way to go.

Chapter 05

CONCLUSION

The past three months have been a great learning experience for me and I'm learning something new about the product every day. The concepts I have learned in the training program were from basics and covered every bit of the product to give interns a complete understanding of the product. The training is so well designed that it keeps you occupied but still makes you more curious about the product. Hevo is on a mission to make life easier for people who are spending their time thinking about how to move their data rather than investing their time in analysing that data.

Hevo as a No-code ETL platform has taken the initiative to make life of people easier not by just replicating data but also has introduced Reverse-ETL, Activate to provide a complete data replication as well as analysis platform.

Working so far at Hevo has been a great journey and I'm looking forward to continuing on this path.

REFERENCES :

- <https://docs.hevo.me/introduction/>
- <https://en.wikipedia.org/wiki/SQL>
- <https://en.wikipedia.org/wiki/Jython>