ADVANCED DATABASE SYSTEM

INNOVATIVE ASSIGNMENT

SUBMITTED BY:
SHAILESH ARYA 20MCED02
FALGUNI PATEL 20MCEC10

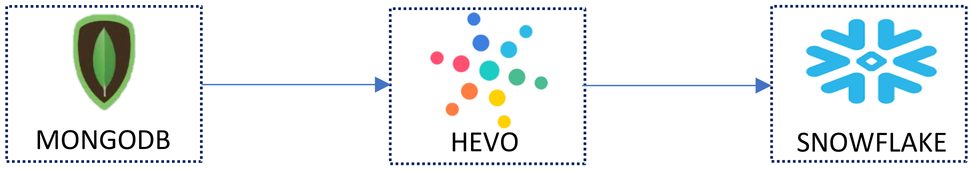
Aim: Create Change Data Capture scenario for streaming data changes to various destination DB nodes as sink.

- As its name suggests, Change Data Capture (CDC) techniques are used to identify changes. CDC can be the basis to synchronize another system with the same incremental changes, or to store an audit trail of changes.

- Source Database: MongoDB
- Sink Datawarehouse: Snowflake
  - For the experiment we have used MongoDB as source database and snowflake as sink data warehouse.
  - MongoDB is a cross-platform document-oriented database that belongs in the class of NoSQL databases. NoSQL databases differ from the traditional SQL databases in the sense that they do not enforce a strict table structure. This flexibility and a mature querying layer make MongoDB a popular choice for modern application development.
  - Snowflake is the only data platform built for the cloud for all your data and all your users. Mobilize your data to advance your business.
- To Implement CDC in MongoDB, there are 2 ways:
  1. Using OpLog in MongoDB: It gives users the power to track changes without having to continuously monitor the operations log (OpLog).
  2. Using Change in Stream:
  - To perform CDC in MongoDB we required to have replica set.
  - For 3rd possible way is to use any platform which does the mentioned approaches for you.
  - For this practical, we have used **HEVO – A Cloud based Automated ETL platform.**
  - Change data capture for MongoDB can be achieved in just 3 steps:
    I. Authenticate and connect your MongoDB data source
    II. Select Change Data Capture as your replication mode
    III. Point to the destination where you want to move data
  - HEVO will take care of the following things:

1. Automatic Schema Handling: MongoDB is schema-less database. So HEVO will dynamically detect and map the schema.
2. Splitting Nested JSON: If you are looking to move data from MongoDB to a data warehouse for analytics, naturally, you might want to split nested JSON and flatten them out. HEVO provides a way for you to handle this easily using a data transformation layer.
3. Detailed Logging and Monitoring
4. Timely Notifications:

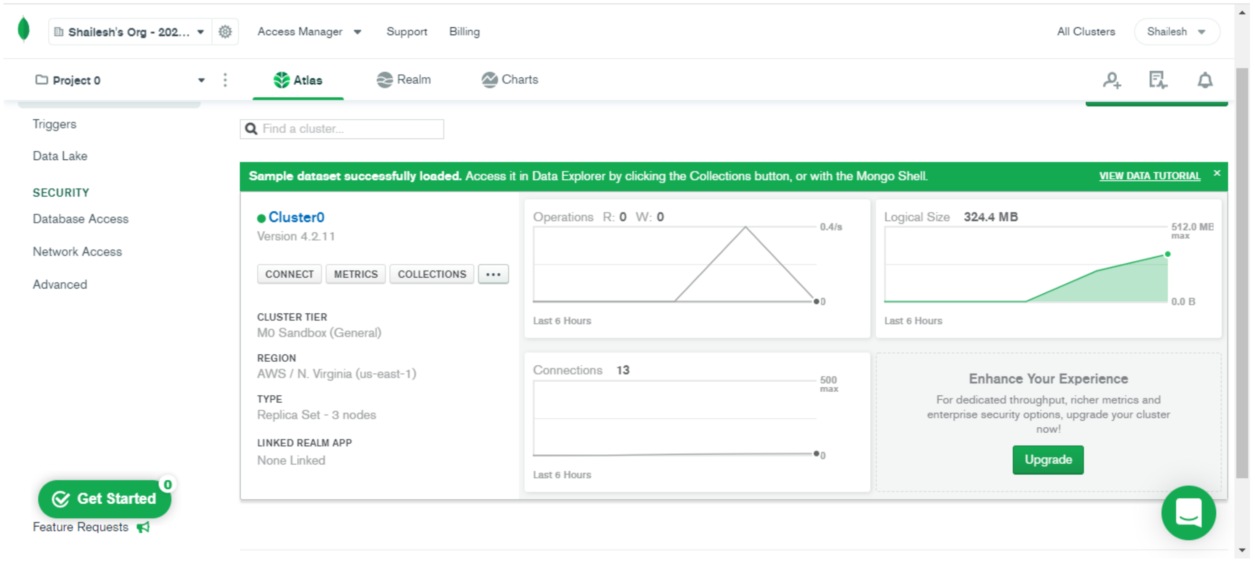
Overall Architecture:



- Here MongoDB will be configured as Source database. We can use either MongoDB installed on local computer or MongoDB Atlas cluster.
- There will be pipeline between HEVO and MongoDB database which captures any change happen to MongoDB in real time.
- HEVO also maps the object of MongoDB data with suitable schema.
- There will be another pipeline between HEVO and snowflake. HEVO will dump the data comes from MongoDB to Snowflake.

Configuring MongoDB on Atlas:

- There are predefined steps available on MongoDB atlas official webpage at: https://docs.atlas.mongodb.com/getting-started.
- Once the Cluster is setup you will have MongoDB database deployed at cloud.
- For CDC, we required to have replica nodes for our MongoDB database.
- On MongoDB Atlas, we have 3 replicas set of our dataset. One is primary node and other 2 are secondary nodes.


Shailesh's Org - 202… Access Manager Support Billing All Clusters Shailesh

Project 0 Atlas Realm Charts

Triggers
Data Lake
SECURITY
Database Access
Network Access
Advanced

Find a cluster...

Sample dataset successfully loaded. Access it in Data Explorer by clicking the Collections button, or with the Mongo Shell. VIEW DATA TUTORIAL

Cluster0
Version 4.2.11

CONNECT METRICS COLLECTIONS …

CLUSTER TIER
M0 Sandbox (General)

REGION
AWS / N. Virginia (us-east-1)

TYPE
Replica Set - 3 nodes

LINKED REALM APP
None Linked

Operations R: 0 W: 0 — 0.4/s — 0 — Last 6 Hours

Logical Size 324.4 MB — 512.0 MB max — 0.0 B — Last 6 Hours

Connections 13 — 500 max — 0 — Last 6 Hours

Enhance Your Experience
For dedicated throughput, richer metrics and enterprise security options, upgrade your cluster now!
Upgrade

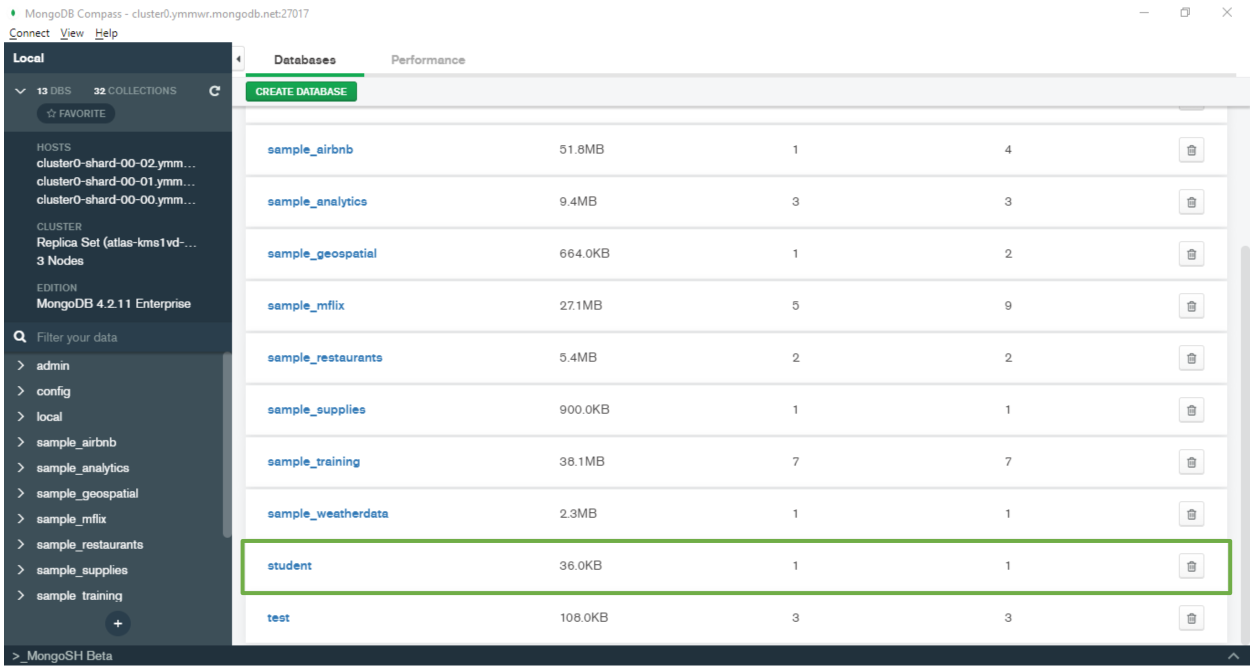
Get Started
Feature Requests

Fig.1. MongoDB Atlas Cluster

- You can access the MongoDB shell using the following command (Replace <dbname> with database name and <username> with your username):

```
mongo "mongodb+srv://cluster0.ymmwr.mongodb.net/<dbname>" --
username <username>
```

- Use can also use the MongoDB Compass Application which provides good user interface.


MongoDB Compass - cluster0.ymmwr.mongodb.net:27017
Connect View Help

Local
13 DBS 32 COLLECTIONS
☆ FAVORITE

HOSTS
cluster0-shard-00-02.ymm…
cluster0-shard-00-01.ymm…
cluster0-shard-00-00.ymm…

CLUSTER
Replica Set (atlas-kms1vd-…
3 Nodes

EDITION
MongoDB 4.2.11 Enterprise

Filter your data
admin
config
local
sample_airbnb
sample_analytics
sample_geospatial
sample_mflix
sample_restaurants
sample_supplies
sample_training

Databases Performance
CREATE DATABASE

sample_airbnb	51.8MB	1	4
sample_analytics	9.4MB	3	3
sample_geospatial	664.0KB	1	2
sample_mflix	27.1MB	5	9
sample_restaurants	5.4MB	2	2
sample_supplies	900.0KB	1	1
sample_training	38.1MB	7	7
sample_weatherdata	2.3MB	1	1
student	36.0KB	1	1
test	108.0KB	3	3

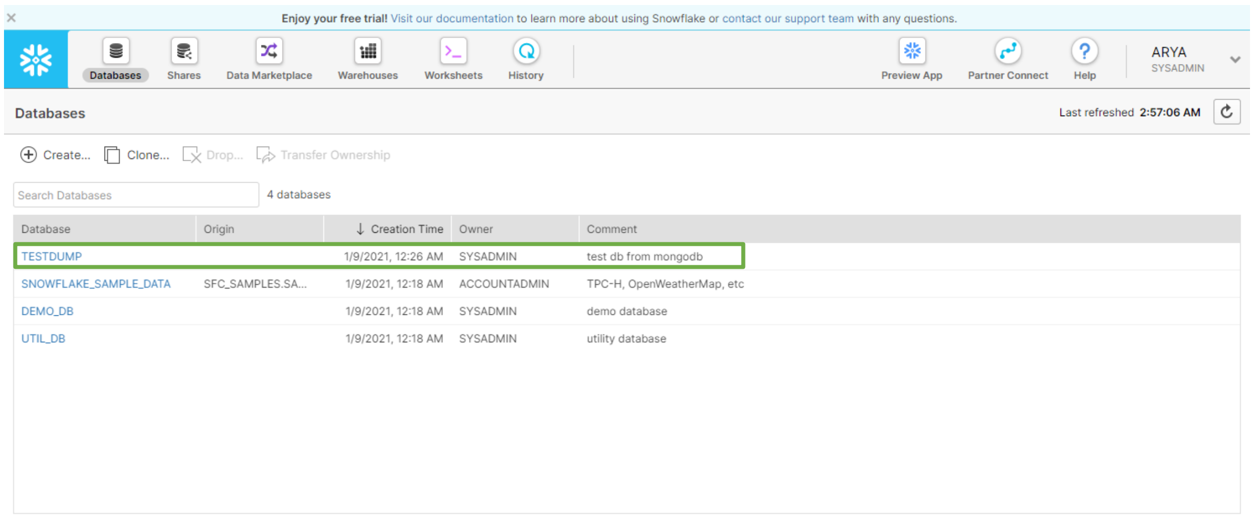
>_MongoSH Beta

Fig.2. MongoDB Compass

- For this practical, we have created the database named as 'test' which has the 'student' collection to store student data.

Configuring Snowflake Datawarehouse:

- You can start with free trial by registering on their official website: https://bit.ly/3s3KfKC
- Once you have registered you need to create a database to store whatever the data come from MongoDB.
- Snowflake provides nice GUI to query your data and perform some analytics as well.
- Here we have created TESTDUMP database in snowflake to dump the data coming from MongoDB.

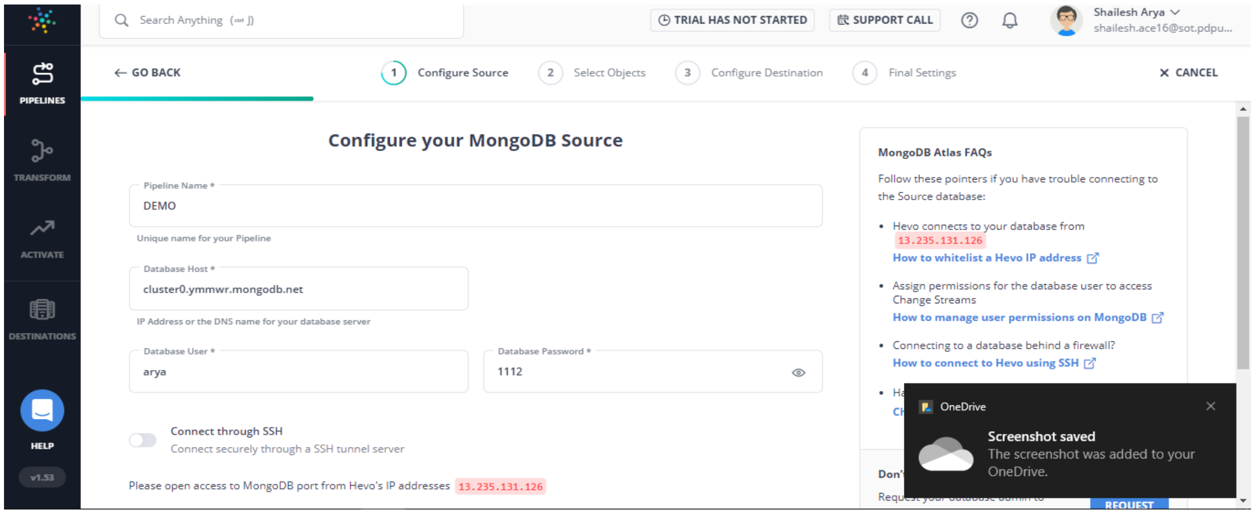


Database	Origin	Creation Time	Owner	Comment
TESTDUMP		1/9/2021, 12:26 AM	SYSADMIN	test db from mongodb
SNOWFLAKE_SAMPLE_DATA	SFC_SAMPLES.SA...	1/9/2021, 12:18 AM	ACCOUNTADMIN	TPC-H, OpenWeatherMap, etc
DEMO_DB		1/9/2021, 12:18 AM	SYSADMIN	demo database
UTIL_DB		1/9/2021, 12:18 AM	SYSADMIN	utility database

Fig. 3. Snowflake User Interface

Creating Pipeline Between MongoDB and HEVO:

- To crate pipeline, choose MongoDB as source database. This will lead to the following screen where you need to add the MongoDB Atlas URL as database host, username and password.



Configure your MongoDB Source

Pipeline Name *
DEMO
Unique name for your Pipeline

Database Host *
cluster0.ymmwr.mongodb.net
IP Address or the DNS name for your database server

Database User *
arya

Database Password *
1112

Connect through SSH
Connect securely through a SSH tunnel server

Please open access to MongoDB port from Hevo's IP addresses 13.235.131.126

MongoDB Atlas FAQs

Follow these pointers if you have trouble connecting to the Source database:

- Hevo connects to your database from 13.235.131.126
  How to whitelist a Hevo IP address
- Assign permissions for the database user to access Change Streams
  How to manage user permissions on MongoDB
- Connecting to a database behind a firewall?
  How to connect to Hevo using SSH

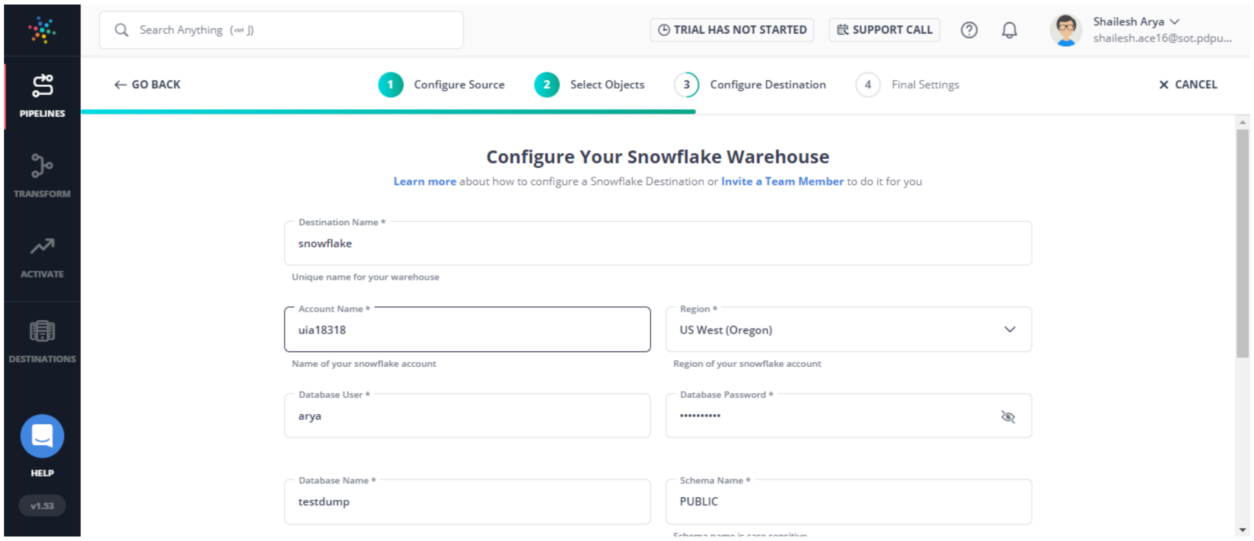
Screenshot saved
The screenshot was added to your OneDrive.

Fig. 4. Configuring MongoDB as Source

- After testing successful connection, you can go to next step.

Creating Pipeline Between HEVO and Snowflake:

- For this select Snowflake as your sink database in HEVO. HEVO also provides another database integration such as MySQL, MySQL Server, Amazon Redshift, etc.



Configure Your Snowflake Warehouse

Learn more about how to configure a Snowflake Destination or Invite a Team Member to do it for you

Destination Name *	snowflake
Account Name *	uia18318
Region *	US West (Oregon)
Database User *	arya
Database Password *	••••••••••
Database Name *	testdump
Schema Name *	PUBLIC

Unique name for your warehouse

Name of your snowflake account

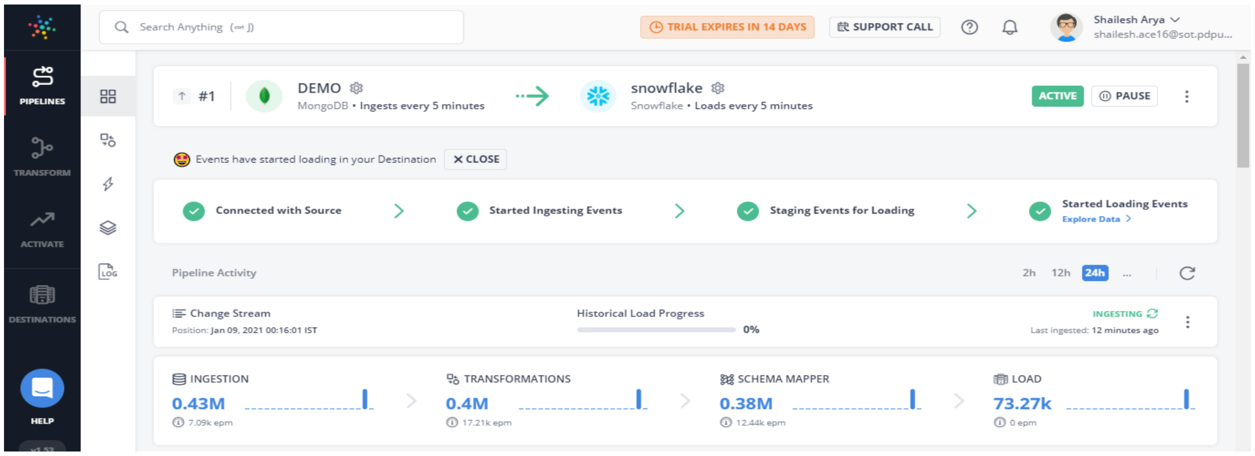
Region of your snowflake account

Fig. 5. Configuring Snowflake as sink

- You need to give your account name and database password to connect Snowflake with HEVO.
- After Successful testing we have working pipeline between MongoDB and Snowflake.

HEVO Dashboard:

- After configuration of MongoDB as source database and Snowflake as sink database we are ready to capture any change in data in MongoDB and send it to Snowflake.



#1 DEMO — MongoDB • Ingests every 5 minutes → snowflake — Snowflake • Loads every 5 minutes

Events have started loading in your Destination

Connected with Source › Started Ingesting Events › Staging Events for Loading › Started Loading Events

Pipeline Activity

Change Stream — Position: Jan 09, 2021 00:16:01 IST

Historical Load Progress 0%

INGESTING — Last ingested: 12 minutes ago

INGESTION	TRANSFORMATIONS	SCHEMA MAPPER	LOAD
0.43M	0.4M	0.38M	73.27k
7.09k epm	17.21k epm	12.44k epm	0 epm

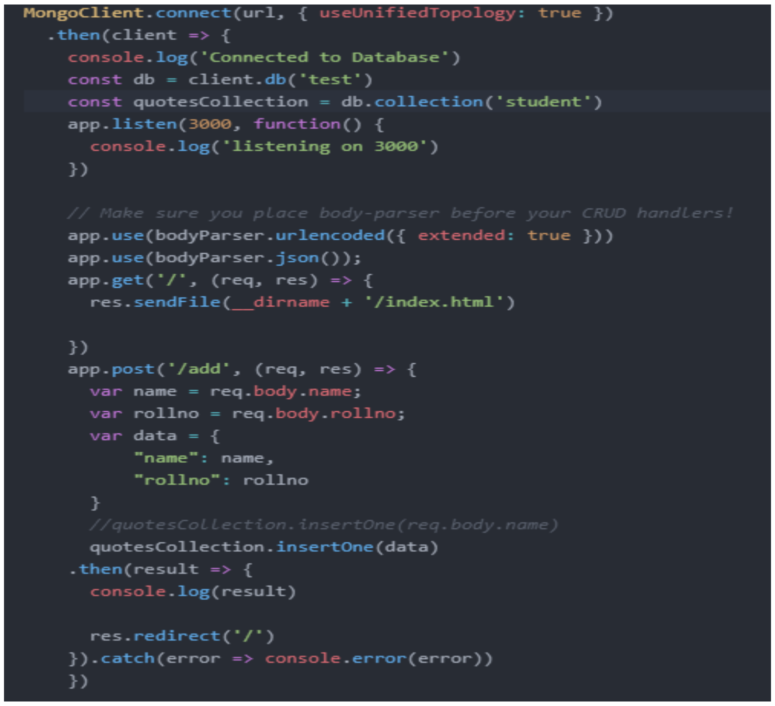
- Fig. 6. HEVO Dashboard

CDC in action:

- The following shows the data insertion in source database and how it is being reflected in sink database.
- Components:
    a) Webpage for Data Insertion in MongoDB
    b) HEVO Dashboard for Realtime Data capture
    c) Webpage for showing and Querying Data on Webpage
    d) Snowflake database (Sink datastore) Interactive GUI
    e) Webpage for Showing snowflake (sink database) data

a) Webpage for Data Insertion in MongoDB:

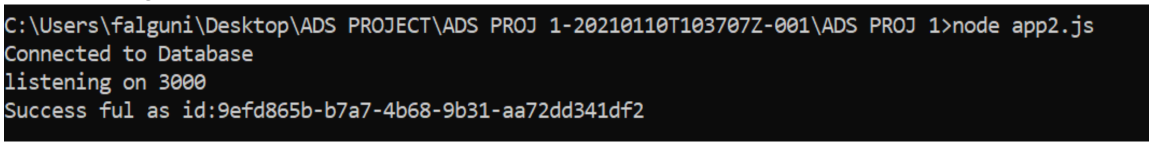
- We have created a web page using bootstrap which has a simple form to take student roll number and name and user can submit details which is then stored in MongoDB.
- The webpage is hosted on node JS server on "localhost:3000/"
- The following shows the snippets to retrieve data from HTML form and store it to MongoDB.



```
MongoClient.connect(url, { useUnifiedTopology: true })
  .then(client => {
    console.log('Connected to Database')
    const db = client.db('test')
    const quotesCollection = db.collection('student')
    app.listen(3000, function() {
      console.log('listening on 3000')
    })

    // Make sure you place body-parser before your CRUD handlers!
    app.use(bodyParser.urlencoded({ extended: true }))
    app.use(bodyParser.json());
    app.get('/', (req, res) => {
      res.sendFile(__dirname + '/index.html')

    })
    app.post('/add', (req, res) => {
      var name = req.body.name;
      var rollno = req.body.rollno;
      var data = {
          "name": name,
          "rollno": rollno
      }
      //quotesCollection.insertOne(req.body.name)
      quotesCollection.insertOne(data)
    .then(result => {
      console.log(result)

      res.redirect('/')
    }).catch(error => console.error(error))
    })
```

Fig.7. Node JS Snippet for Data Insertion in MongoDB

- Fig- 8. Shows shows the NodeJS script running at localhost:3000 which hosts the webpage for data insertion.



```
C:\Users\falguni\Desktop\ADS PROJECT\ADS PROJ 1-20210110T103707Z-001\ADS PROJ 1>node app2.js
Connected to Database
listening on 3000
Success ful as id:9efd865b-b7a7-4b68-9b31-aa72dd341df2
```

Fig. 8. NodeJS script running for Data Insertion

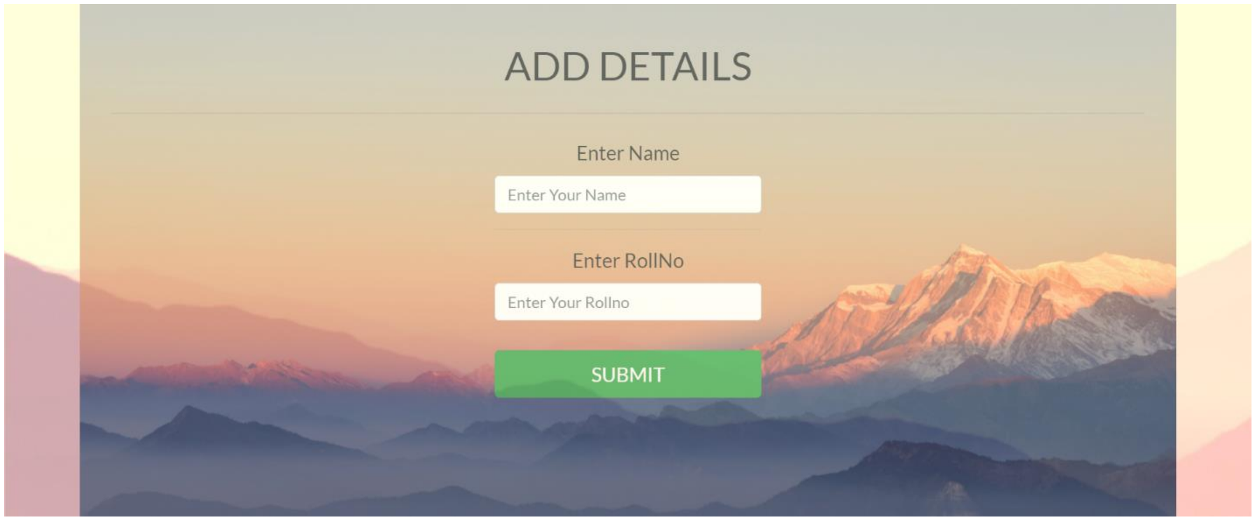


Fig.9. Webpage for Data Insertion in MongoDB

- The following picture shows the webpage UI for data insertion.
  - We have inserted record: {"name": "Falguni Patel", "rollno":"29"}

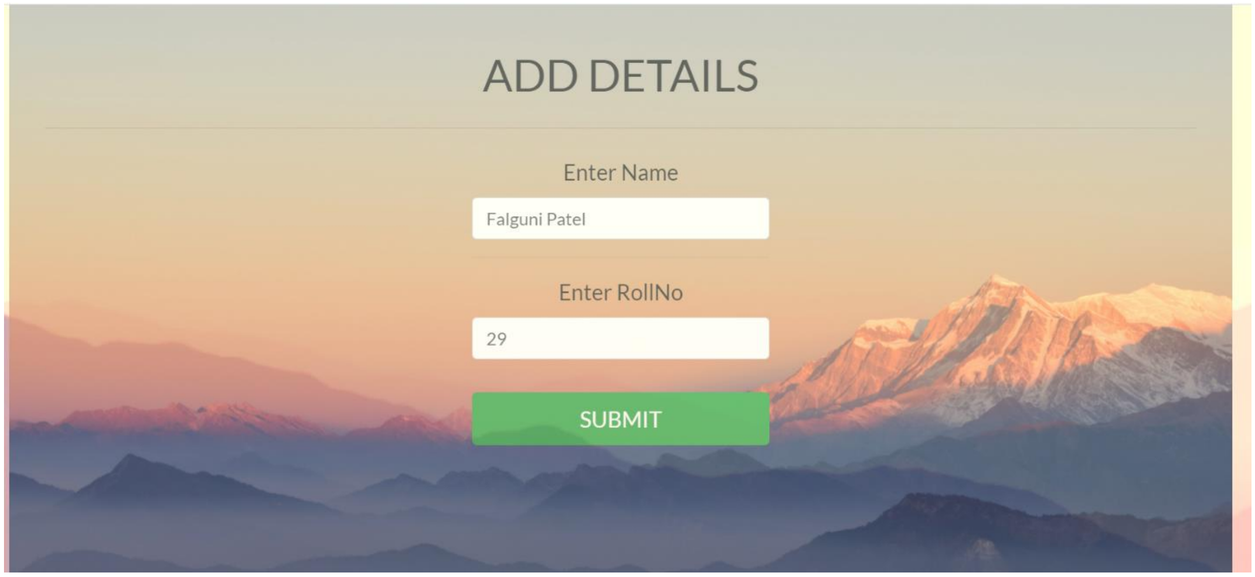


Fig.10. Adding Data through User Input

b) HEVO Dashboard for Realtime Data capture

- Figure 11. Shows the data update event in source database captured by HEVO in real time.

- HEVO will map the no-schema architecture of MongoDB to structured architecture with adding few more fields before sending it to sink database.

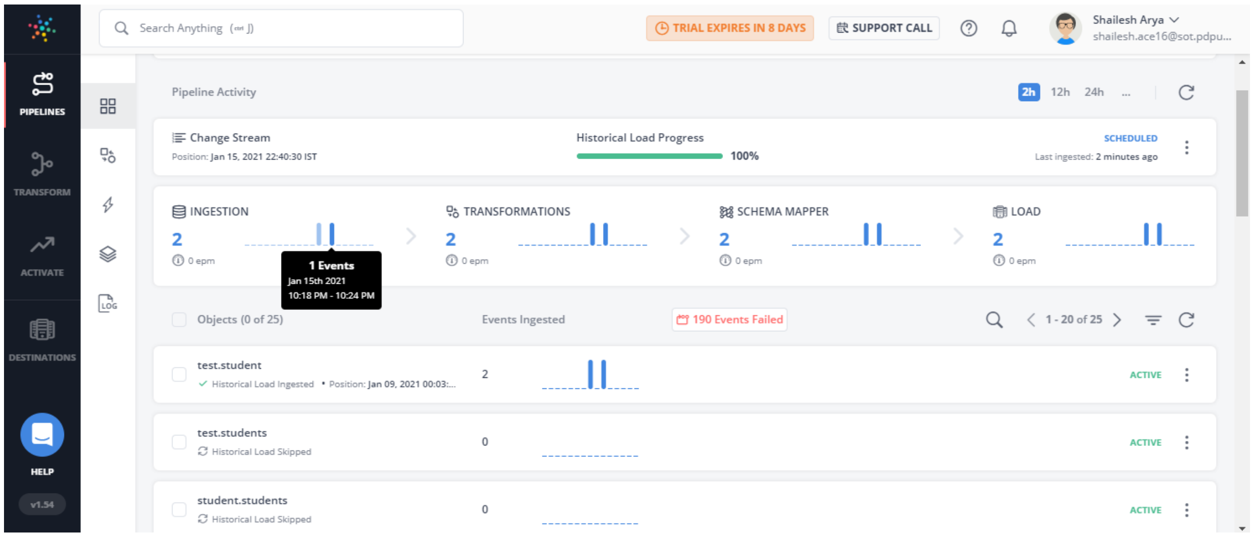
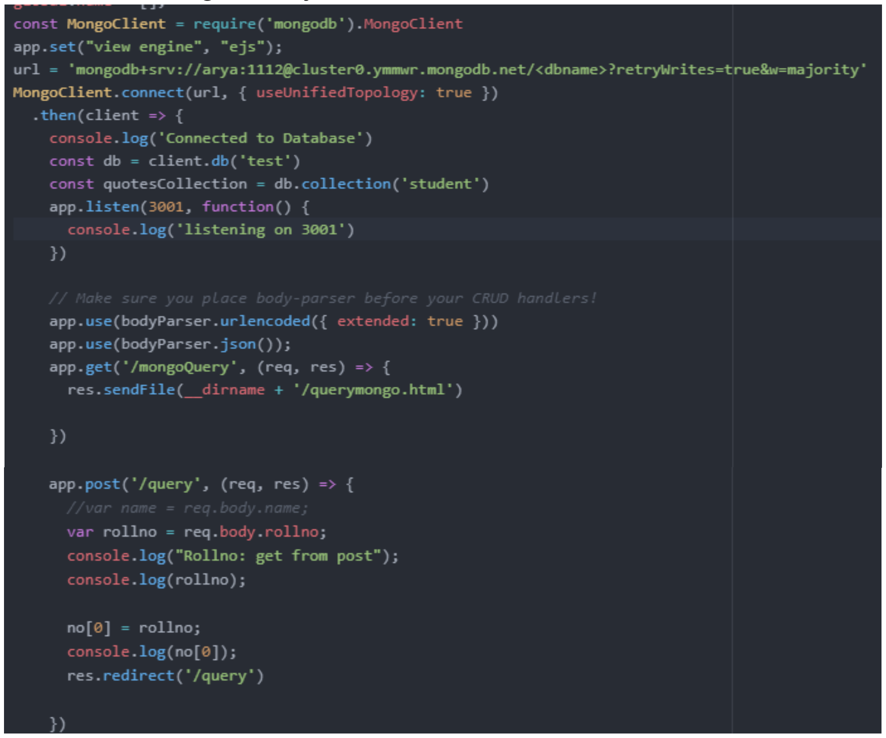


Fig. 11. HEVO Capturing Events from source database

c) Webpage for showing and Querying Data on Webpage

- We have also created a webpage which shows either all the records stored in MongoDB database or user can query a particular roll number and web page displays the student name along with roll number.
- This web page is also created using Node JS script running on url: “localhost:3001/mongoQuery”.



```
const MongoClient = require('mongodb').MongoClient
app.set("view engine", "ejs");
url = 'mongodb+srv://arya:1112@cluster0.ymmwr.mongodb.net/<dbname>?retryWrites=true&w=majority'
MongoClient.connect(url, { useUnifiedTopology: true })
  .then(client => {
    console.log('Connected to Database')
    const db = client.db('test')
    const quotesCollection = db.collection('student')
    app.listen(3001, function() {
      console.log('listening on 3001')
    })

    // Make sure you place body-parser before your CRUD handlers!
    app.use(bodyParser.urlencoded({ extended: true }))
    app.use(bodyParser.json());
    app.get('/mongoQuery', (req, res) => {
      res.sendFile(__dirname + '/querymongo.html')

    })

    app.post('/query', (req, res) => {
      //var name = req.body.name;
      var rollno = req.body.rollno;
      console.log("Rollno: get from post");
      console.log(rollno);

      no[0] = rollno;
      console.log(no[0]);
      res.redirect('/query')

    })
```

Fig.12. Node JS script for Querying MongoDB Data

- Figure 13. Shows the we page UI to search MongoDB data using student's roll number.
- User can either enter particular roll number or can click on "View all Records" button to display all the records available.

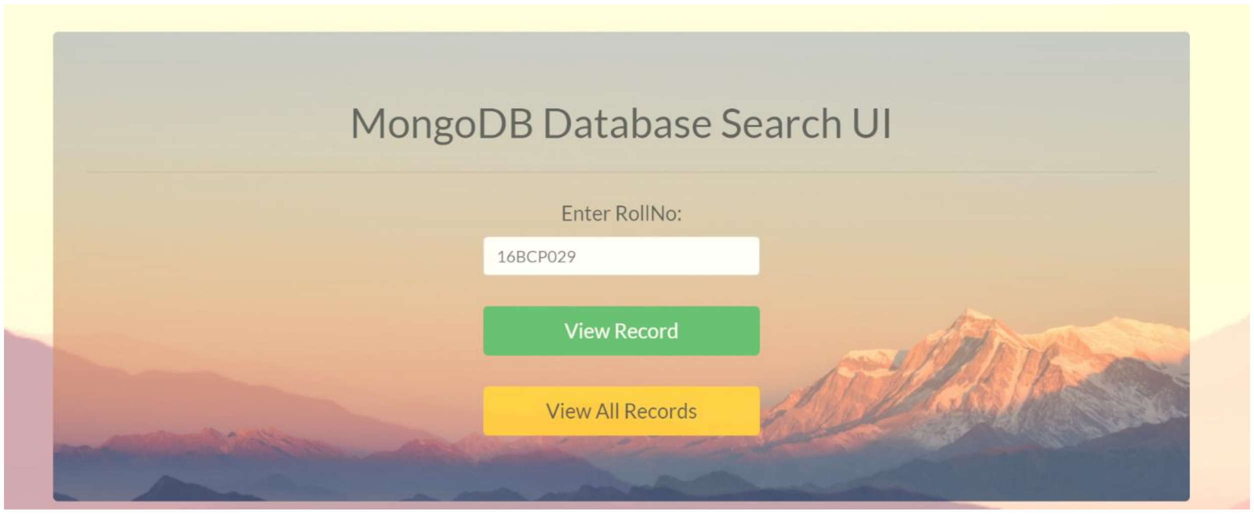
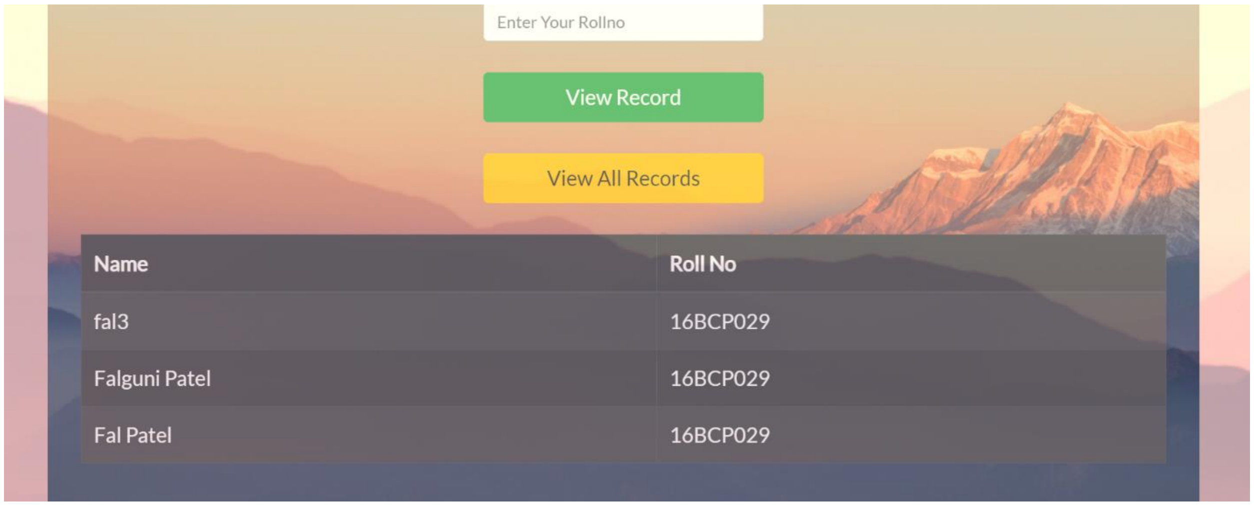


Fig.13. Web-UI to query particular student data

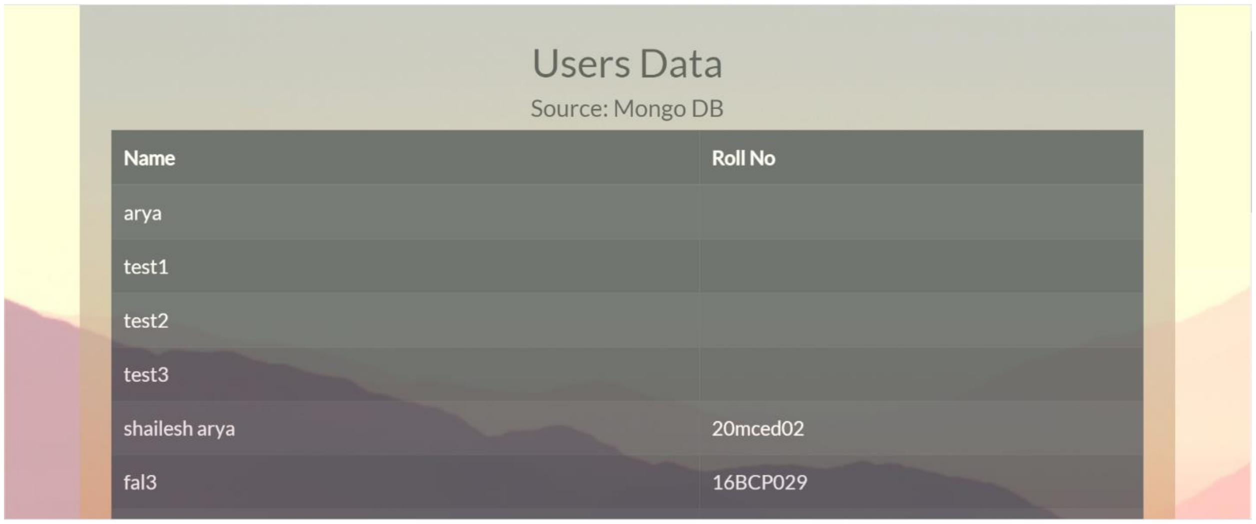
- Figure 14. Shows the results retrieved for "roll no: 16BCP029".



Name	Roll No
fal3	16BCP029
Falguni Patel	16BCP029
Fal Patel	16BCP029

Fig. 14 Retrieved Results

- Figure 15. Shows the results retrieved using View All Records button.



Users Data

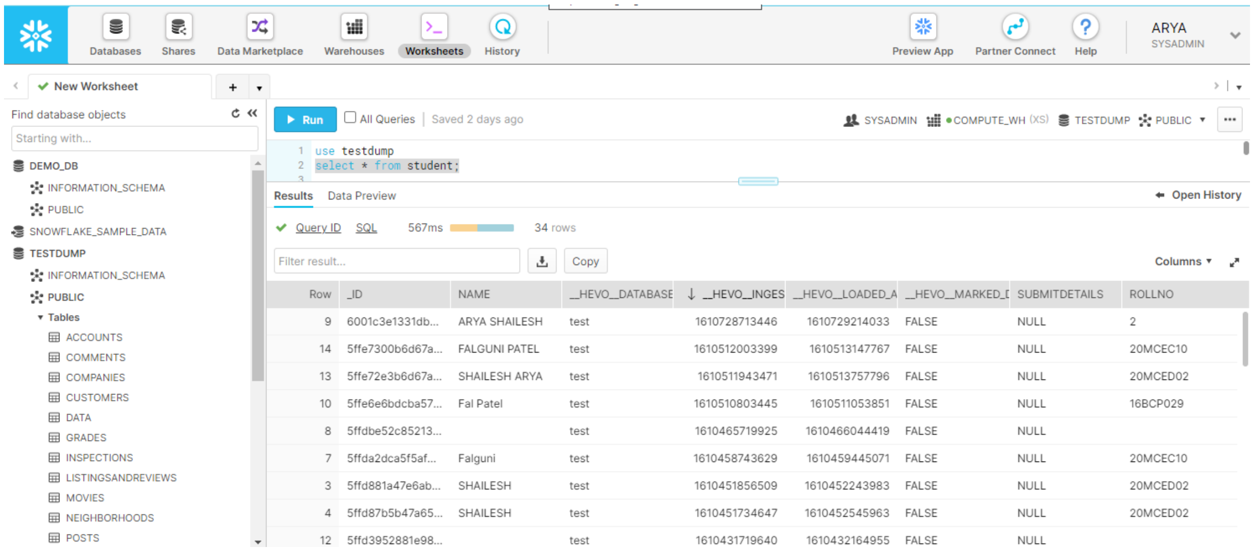
Source: Mongo DB

Name	Roll No
arya	
test1	
test2	
test3	
shailesh arya	20mced02
fal3	16BCP029

Fig. 15 Retrieved all data from MongoDB Database

d) Snowflake database (Sink datastore) Interactive GUI:

- As we have inserted data in section (a), the data change will be captured by HEVO shown in section (b), once the data is transformed to schema formatted, it'll be moved to the snowflake datastore.
- The following figure shows the data stored in Snowflake UI and user can write SQL queries to retrieve data from datastore.



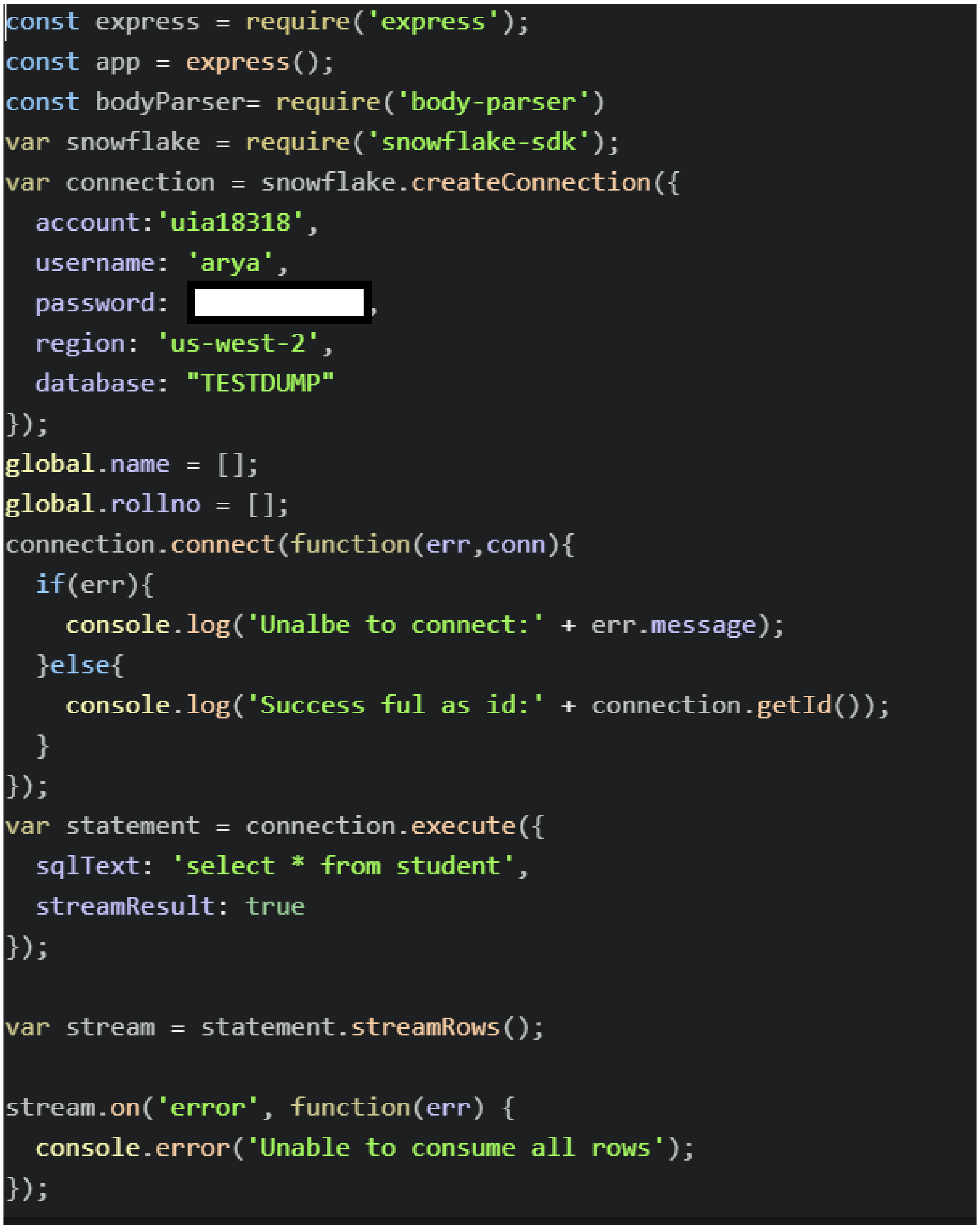
```
use testdump
select * from student;
```

Row	_ID	NAME	__HEVO__DATABASE	↓ __HEVO__INGES	__HEVO__LOADED_A	__HEVO__MARKED_I	SUBMITDETAILS	ROLLNO
9	6001c3e1331db...	ARYA SHAILESH	test	1610728713446	1610729214033	FALSE	NULL	2
14	5ffe7300b6d67a...	FALGUNI PATEL	test	1610512003399	1610513147767	FALSE	NULL	20MCEC10
13	5ffe72e3b6d67a...	SHAILESH ARYA	test	1610511943471	1610513757796	FALSE	NULL	20MCED02
10	5ffe6e6bdcba57...	Fal Patel	test	1610510803445	1610511053851	FALSE	NULL	16BCP029
8	5ffdbe52c85213...		test	1610465719925	1610466044419	FALSE	NULL	
7	5ffda2dca5f5af...	Falguni	test	1610458743629	1610459445071	FALSE	NULL	20MCEC10
3	5ffd881a47e6ab...	SHAILESH	test	1610451856509	1610452243983	FALSE	NULL	20MCED02
4	5ffd87b5b47a65...	SHAILESH	test	1610451734647	1610452545963	FALSE	NULL	20MCED02
12	5ffd3952881e98...		test	1610431719640	1610432164955	FALSE	NULL	

Fig. 16. Snowflake Datastore

e) Webpage for Showing snowflake (sink database) data

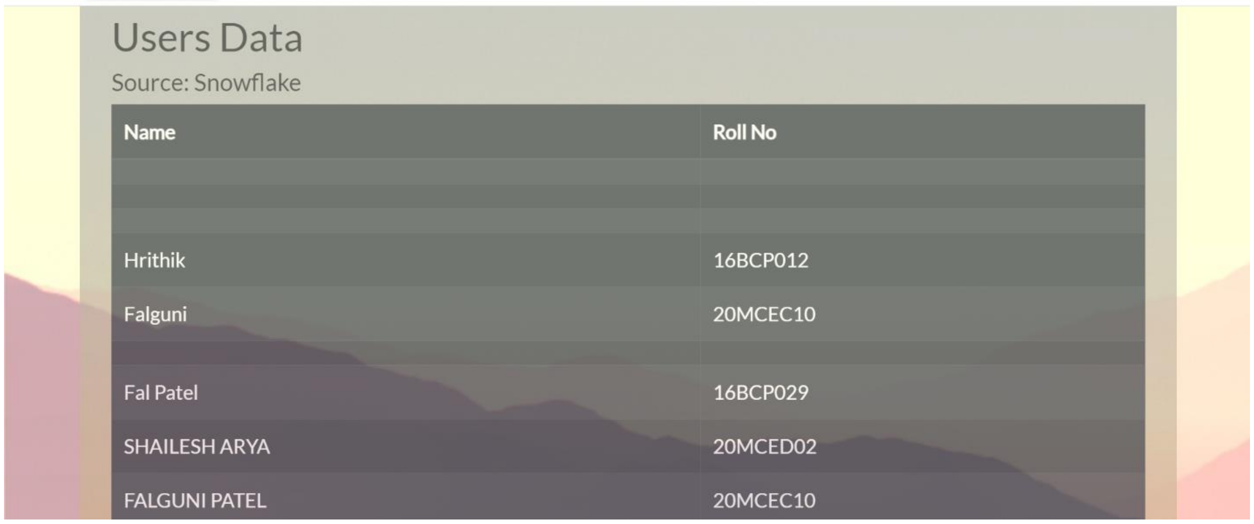
- We have written a Node JS script to retrieve all data being stored at Snowflake datastore. The script runs at url: "localhost:3000/show"



```
const express = require('express');
const app = express();
const bodyParser= require('body-parser')
var snowflake = require('snowflake-sdk');
var connection = snowflake.createConnection({
  account:'uia18318',
  username: 'arya',
  password: [redacted],
  region: 'us-west-2',
  database: "TESTDUMP"
});
global.name = [];
global.rollno = [];
connection.connect(function(err,conn){
  if(err){
    console.log('Unalbe to connect:' + err.message);
  }else{
    console.log('Success ful as id:' + connection.getId());
  }
});
var statement = connection.execute({
  sqlText: 'select * from student',
  streamResult: true
});

var stream = statement.streamRows();

stream.on('error', function(err) {
  console.error('Unable to consume all rows');
});
```

Fig.17. Snippet for snowflake datastore connection using Node JS

- Figure 18 shows the data retrieved from snowflake datastore on webpage using Node JS.



Users Data

Source: Snowflake

Name	Roll No
Hrithik	16BCP012
Falguni	20MCEC10
Fal Patel	16BCP029
SHAILESH ARYA	20MCED02
FALGUNI PATEL	20MCEC10

Fig. 18. Retrieval of all snow flake data

Conclusion:

- CDC is important for resumable database, scalability, Ease of use, Filter capability, and Security.
- WE have successfully demonstrated how the Change in Data works in NoSQL database and how one can use 3rd party application like HEVO to implement CDC with source and sink databases and can monitor how the data moves from source to sink and can also monitor any change, update or delete operation performed in source datastore and that can to can be reflected in sink database.

--- END ---