SPAM DETECTION SYSTEM

Name: Saiesha Shivayogi Gowdar

Student ID number:23270833

Email address: saiesha.gowdar2@mail.dcu.ie

Program of study: MSc in Computing (Data Analytics)

Module code: CA675

Date of submission: 16/11/2023

Github link : **https://github.com/SaieshaGowdar/YoutubedataSpamHam**

Task 1.1: Install Hadoop and create a Hadoop cluster on AWS using EMR & Task 1.2: Install MapReduce, Pig and Hive to use the cluster created in Task 1.1

1. Accessed AWS Console:
   a. Logged in to AWS Management Console.
2. Navigated to EMR:
   a. Searched for EMR service.
3. Created Cluster:
   a. Clicked on "Create Cluster."
4. Configuration:
   a. Chose "Hadoop," "Hive," "Pig," and "Spark" under Software Configuration.
   b. Selected the number of instances for the main node and data nodes (e.g., 1 main node, 2 data nodes).
   c. Chose instance types (e.g., m4.large).
   d. Configured additional options as needed.
5. Security Configuration:
   a. Specified a key pair for secure access (e.g., "vockey").
6. Created Cluster:
   a. Reviewed configurations and created the cluster.**(Image1)(Image 4)**
7. Created Cloud9 Environment:
   a. Searched for Cloud service & clicked on create environment.(**Image2)**
   b. Clicked on Open cloud environment.
8. Added SSH Key:
   a. Added the SSH key associated with the EMR cluster to the Cloud9 environment.**(Image 3)**
9. Downloaded PEM Key:
   a. Downloaded the PEM key provided during the EMR cluster creation (e.g., labuser.pem).
10. Uploaded Key to Cloud9:**(Image 5)**
    a. Uploaded the labuser.pem key to the Cloud9 environment.
11. Set Permissions:**(Image 5)**
    a. In Cloud9 terminal, ran: `chmod 400 labuser.pem` to set the correct permissions on the key.
12. SSH to EMR Cluster:**(Image 5)**
    a. Connected to the EMR cluster using the DNS link:

    ```
    ssh -i labuser.pem hadoop@<EMR_CLUSTER_DNS>
    ```

13. Checked for Hive Installation:**(Image 7)**
    a. Ran hive in the terminal to check if Hive is installed successfully.
14. Checked Pig Installation:**(Image 9)**
    a. Ran pig to check if Pig is installed successfully.
15. Checked Spark Installation**:(Image 10)**
    a. Ran spark-shell to check if Spark is installed successfully.

Task 2.1: Choose a relevant dataset (should be justified) ,Task 2.2: Get data from any public dataset repository&Task 2.3: Load data into chosen cloud technology (AWS, GCP, Azure, …)

Justification:

1. **Selection Criteria**:
    a. Chose the Youtube spam dataset due to its diverse comments from well-known individuals (Psy, Katy Perry, LMFAO, Eminem, Shakira).
    b. 7800 rows were consolidated, combining 5 datasets to create a "newdataset" .
    c. Date column was removed to facilitate data loading.
2. **Dataset Contains Columns:**
    a. Comment_Id
    b. Author
    c. Content
    d. Class
3. **Dataset Origin:**
    a. Obtained from Kaggle, a prominent platform for data science and machine learning datasets.
    b. Dataset Link: Youtube Spam Dataset.
4. **Cloud Storage:(Image 6)**
    a. Loaded the Youtube spam dataset into an Amazon S3 bucket.
    b. Utilised Amazon S3 for efficient storage and accessibility during subsequent cloud-based analyses.
    c. Used the URI of S3 bucket (**Image 7)**

Task 3: Clean and process the data using Pig and/or Hive

1. **Create External Table:(Image 11)(Image 12)**
   a. **Query:**
      CREATE EXTERNAL TABLE youtubespam(comment_id STRING, author STRING, content STRING, class STRING) ROW FORMAT DELIMITED FIELDS TERMINATED BY ',' LOCATION 's3://mycloudassginmentbucketnew/path/';
   b. **Explanation:**
      i. Created an external table named youtubespam with columns comment_id, author, content, and class.
      ii. Specified the row format as delimited with fields terminated by a comma.
      iii. Set the location to the corresponding path in the S3 bucket.

2. **Load Data into the External Table:**
   a. **Query:**
      LOAD DATA INPATH 's3://mycloudassginmentbucketnew/newdata.csv' INTO TABLE youtubespam;
   b. **Explanation:**
      i. Loaded data from the specified CSV file (newdata.csv) located in the S3 bucket into the youtubespam external table.
      ii. This step effectively incorporates the dataset into the Hive external table for further processing.

3. **Data Cleaning Operations:**
   a. **Performed Operations:**
      Handling Null Values:
      i. Executed queries to handle null values, ensuring data completeness.
      Trimming Data:
      ii. Applied trimming operations to remove unnecessary whitespaces from the dataset.
   b. **Queries:**
      Null Query:**(Image 13)**
         CREATE TABLE youtubespam_nonull AS SELECT * FROM youtubespam WHERE comment_id IS NOT NULL AND author IS NOT NULL AND content IS NOT NULL AND class IS NOT NULL;

      Trimming query:**(Image 14)**
         CREATE TABLE youtubespam_nonull AS SELECT * FROM youtubespam WHERE comment_id IS NOT NULL AND author IS NOT NULL AND content IS NOT NULL AND class IS NOT NULL;

c. **Explanation:**
    i. Used Hive queries to handle null values and trim unnecessary whitespaces in the dataset.
    ii. Ensured data quality and prepared the dataset for further analysis.

Task 4: Ham and Spam using Pig and/or Hive ,Task 4.1: Query processed data to differentiate ham and spam part of the dataset, Task 4.2: Find the top 10 spam accountsTask 4.3: Find the top 10 ham accounts

1. **Querying processed data to differentiate ham and spam part of the dataset:**
    a. CREATE TABLE youtubespam_classification AS SELECT *, CASE WHEN content RLIKE '(Check|subscribe|views|trading)' THEN 'spam' ELSE 'ham' END AS classification FROM youtubespam_trimmed;
    b. **Explanation:**
        i. Created a new table youtubespam_classification.
        ii. Used a CASE statement to classify comments into 'spam' or 'ham' based on a simple regular expression. Comments containing certain keywords ('Check,' 'subscribe,' 'views,' 'trading') are classified as 'spam,' and the rest are classified as 'ham.'

2. **Display Sample Records:**
    a. Select * from youtubespam_classification LIMIT 10;**(Image 15)**
    b. **Explanation:**
        iii. Checked the first 10 rows of the youtubespam_classification table to inspect the newly added 'classification' column.
3. **Find the top 10 spam accounts(Image 16)**
    a. SELECT author, COUNT(*) AS spam_count FROM youtubespam_classification WHERE classification = 'spam' GROUP BY author ORDER BY spam_count DESC LIMIT 10;
    b. **Explanation:**
        iii. Executed a query to find the top 10 accounts associated with spam comments.
        iv. Used the GROUP BY clause to group the results by the author.
        v. The COUNT(*) function is used to count the number of spam comments for each author.
        vi. Ordered the results in descending order based on the spam count and limited the output to the top 10.

4. **Find the top 10 ham accounts(Image 17)**
    a. SELECT author, COUNT(*) AS ham_count FROM youtubespam_classification WHERE classification = 'ham' GROUP BY author ORDER BY ham_count DESC LIMIT 10;
    b. **Explanation:**
        i. Executed a query to find the top 10 accounts associated with ham (non-spam) comments.
        ii. Used the GROUP BY clause to group the results by the author.
        iii. The COUNT(*) function is used to count the number of ham comments for each author.
        iv. Ordered the results in descending order based on the ham count and limited the output to the top 10.

Task 5: TF-IDF using MapReduce

Query1:

1. **Step 1: Tokenization and Word Count(Image18)**
    a. CREATE TABLE word_counts AS SELECT comment_id, word, COUNT(1) AS word_count FROM (SELECT comment_id, EXPLODE(SPLIT(LOWER(content), '\\s+')) AS word FROM youtubespam_classification) t WHERE word IS NOT NULL GROUP BY comment_id, word;
    b. **Explanation:**
        i. The query takes the content column from the youtubespam_classification table, tokenizes it into individual words, and then counts the occurrences of each word for each comment_id. The final result is stored in a new table named word_counts. This kind of operation is commonly used in natural language processing and text analysis to understand the frequency of words in a dataset.

2. **Step 2: Calculate Term Frequency (TF)(Image 19)**
    a. CREATE TABLE term_frequency AS SELECT wc.comment_id, wc.word, wc.word_count, wc.word_count / MAX(wc.word_count) OVER (PARTITION BY wc.comment_id) AS term_frequency FROM word_counts wc;
    b. **Explanation:**
        i. The SQL query creates a table named term_frequency by calculating term frequency for each word in word_counts. It divides the word count by the maximum word count within the corresponding comment, aiding in contextual importance analysis.

3. **Step 3: Calculate Inverse Document Frequency (IDF)(Image 20)**
    a. CREATE TABLE inverse_document_frequency AS SELECT word, COUNT(DISTINCT comment_id) AS document_count, LOG(COUNT(DISTINCT comment_id) / COUNT(DISTINCT comment_id) OVER ()) AS inverse_document_frequency FROM word_counts GROUP BY word;
    b. **Explanation:**
        i. The query generates a table named inverse_document_frequency by computing the inverse document frequency for each word in word_counts. It calculates document count and inverse document frequency, providing insights into word significance across comments.

4. **Step 4: Calculate TF-IDF**
    a. CREATE TABLE tfidf AS SELECT tf.comment_id, tf.word, tf.term_frequency * idf.inverse_document_frequency AS tfidf FROM term_frequency tf JOIN inverse_document_frequency idf ON tf.word = idf.word;
    b. **Explanation:**
        i. The SQL query creates a table named tfidf by computing TF-IDF (Term Frequency-Inverse Document Frequency) for each word in term_frequency and inverse_document_frequency. It joins the two tables based on the word and calculates the product of term frequency and inverse document frequency.

5. **Filter for Top 10 Spam Accounts(Image 21)(Image 22)**
    a. CREATE TABLE top_spam_accounts AS SELECT author, COUNT(DISTINCT comment_id) AS spam_count FROM youtubespam_classification WHERE classification = 'spam' GROUP BY author ORDER BY spam_count DESC LIMIT 10;
    b. **Explanation:**
        i. The query creates a table named top_spam_accounts by counting the distinct spam comments for each author in the youtubespam_classification table. It selects the top 10 authors with the highest spam comment counts.

6. **Filter for Top 10 Spam Keywords for Each Top 10 Spam Account(Image 25)**
    a. CREATE TABLE top_spam_keywords AS SELECT ts.author, t.word, SUM(tfidf) AS total_tfidf FROM top_spam_accounts ts JOIN tfidf t ON ts.comment_id = t.comment_id GROUP BY ts.author, t.word ORDER BY ts.author, total_tfidf DESC LIMIT 10;
    b. **Explanation:**
        i. This query creates a table named top_spam_keywords by joining the top_spam_accounts and tfidf tables, calculating the total TF-IDF for

each word associated with the top spam authors. It then selects the top 10 results based on author and total TF-IDF.

7. **Filter for Top 10 Ham Accounts(Image 23)(Image 24)**
    a. CREATE TABLE top_ham_accounts AS SELECT author, COUNT(DISTINCT comment_id) AS ham_count FROM youtubespam_classification WHERE classification = 'ham' GROUP BY author ORDER BY spam_count DESC LIMIT 10;
    b. **Explanation:**
        i. This query creates a table named top_ham_accounts by counting the distinct non-spam comments for each author in the youtubespam_classification table. It selects the top 10 authors with the highest non-spam comment counts.

Image1:

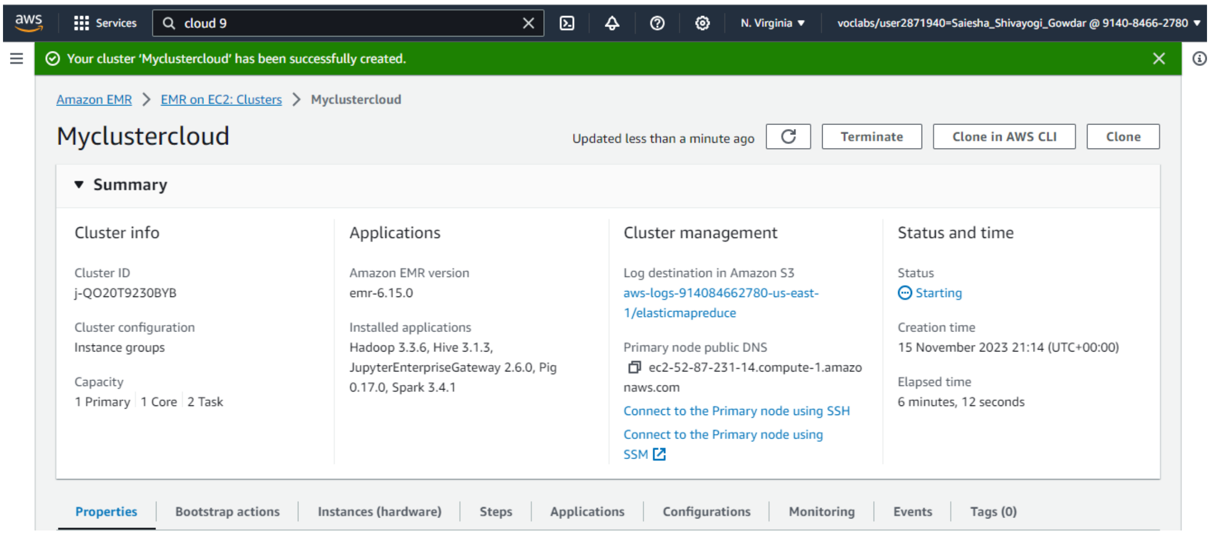


Image2:

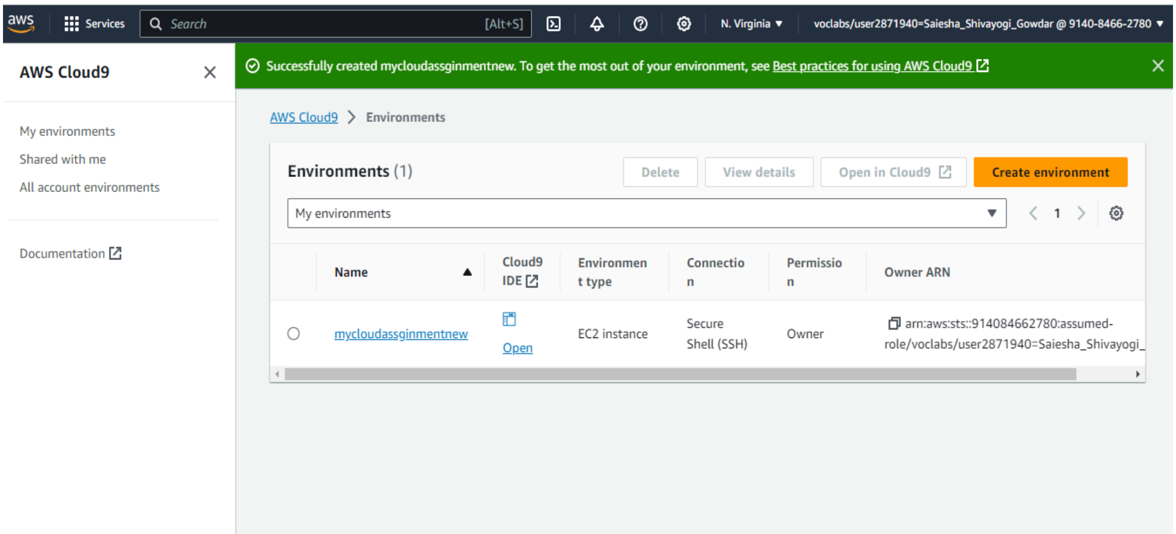


Image 3:

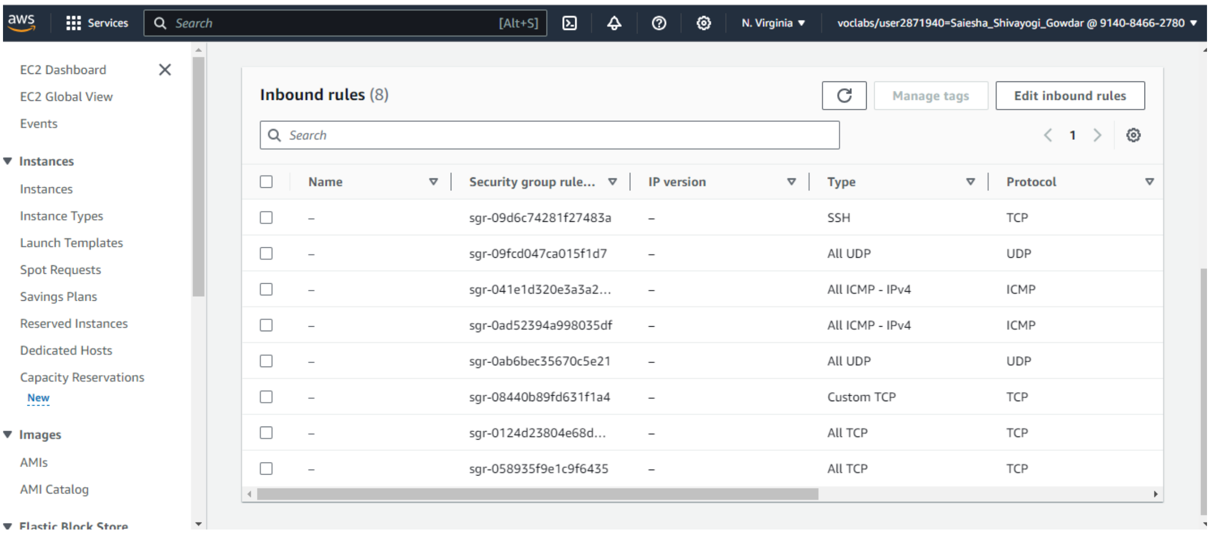


Image 4:

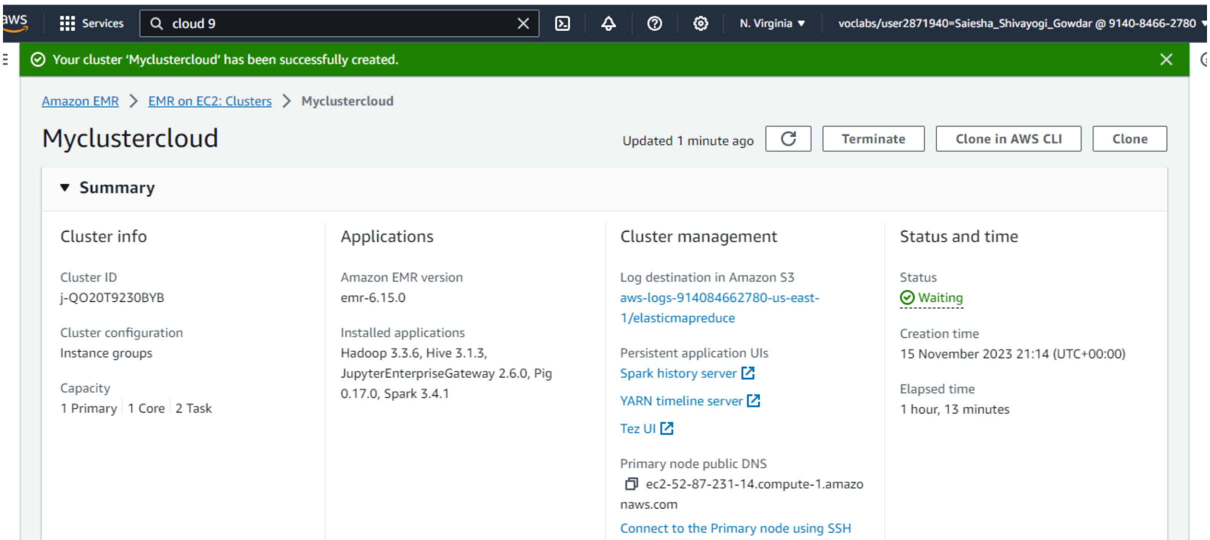


Image 5:

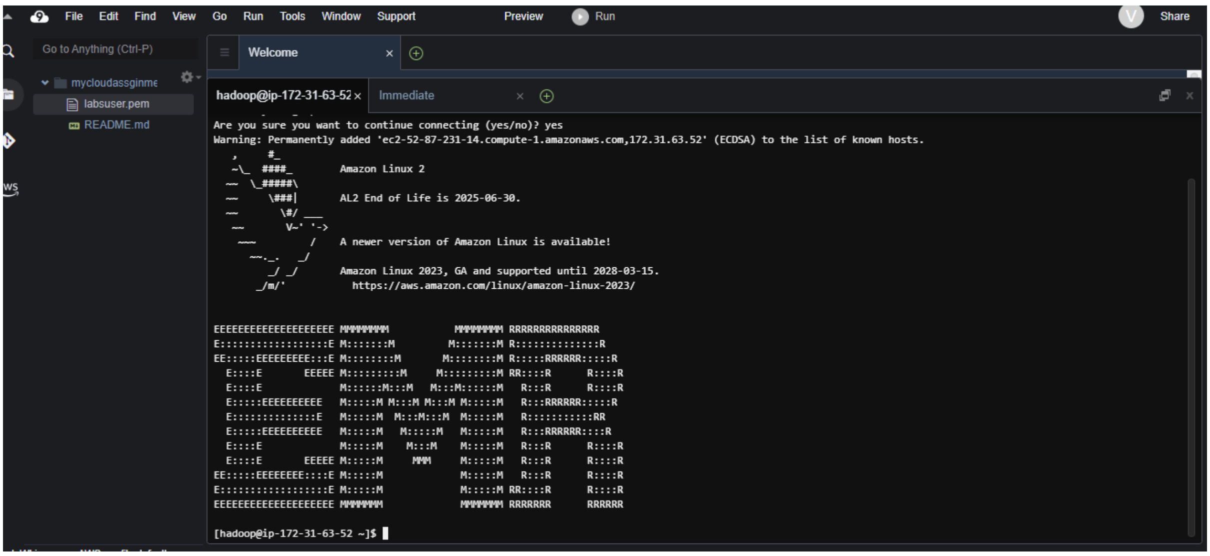


Image 6:

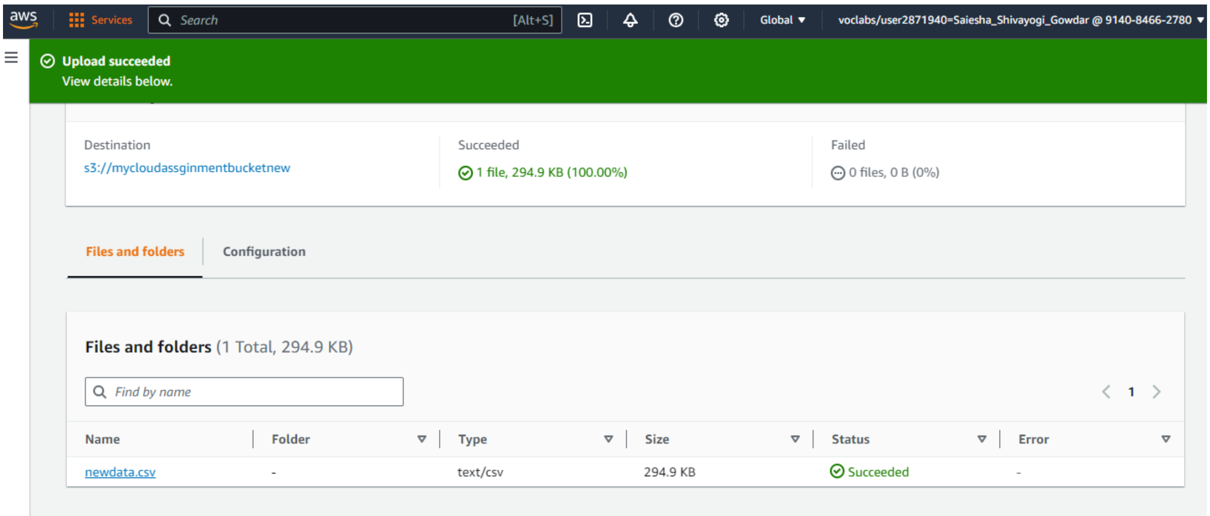


Image 7:

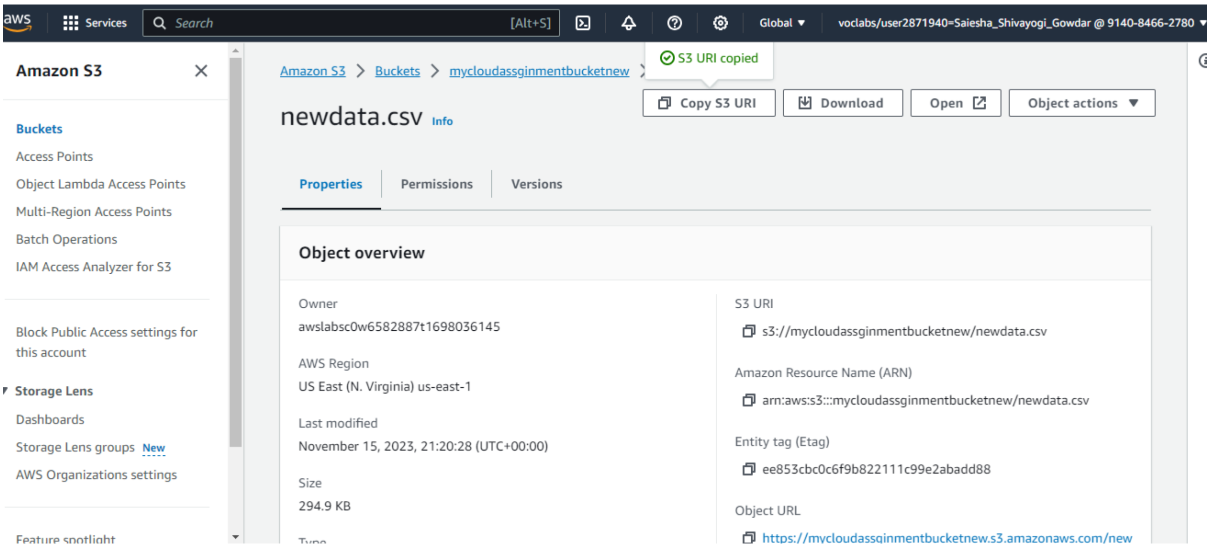


Image 8:

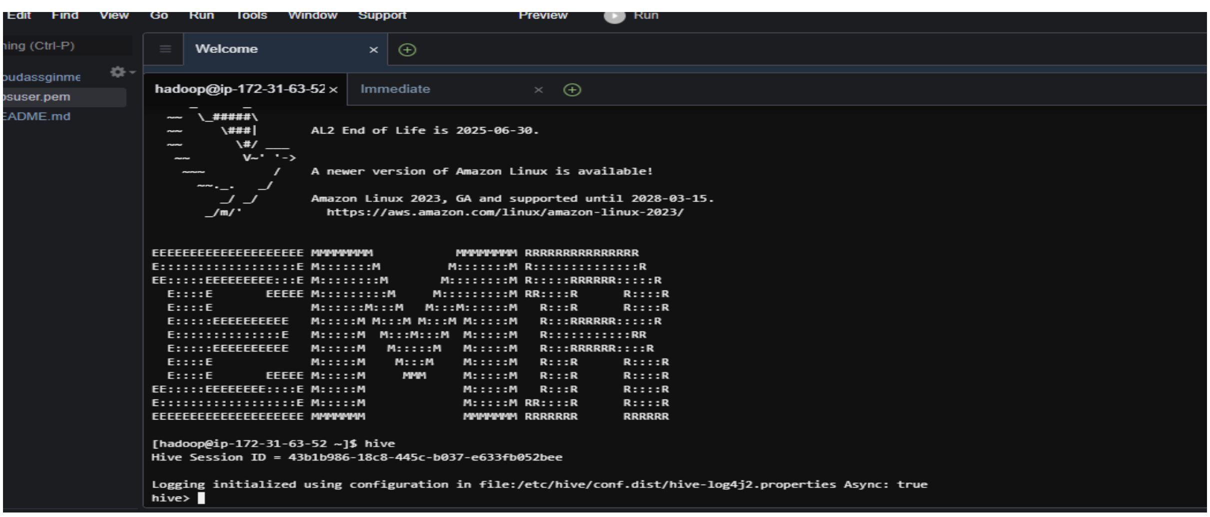


Image 9:

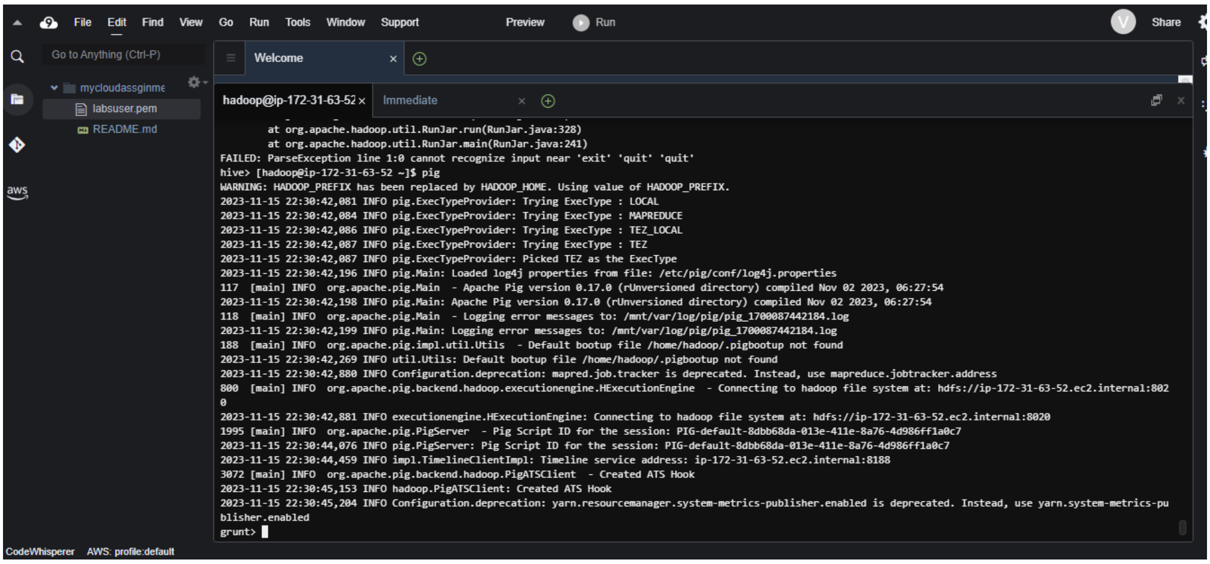


Image 10:

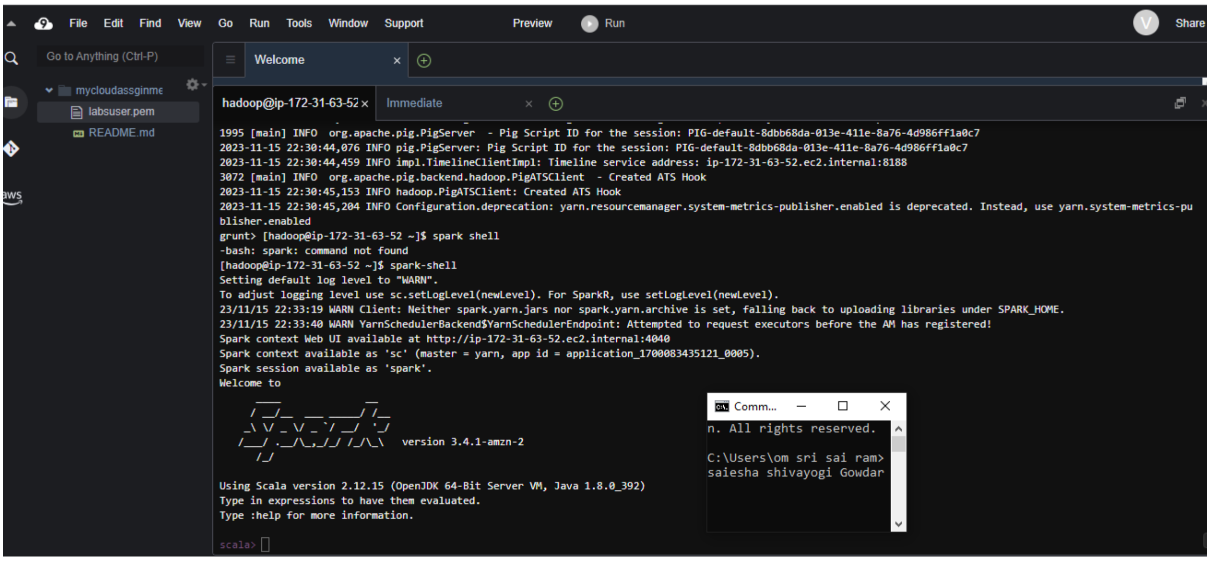


Image 11:

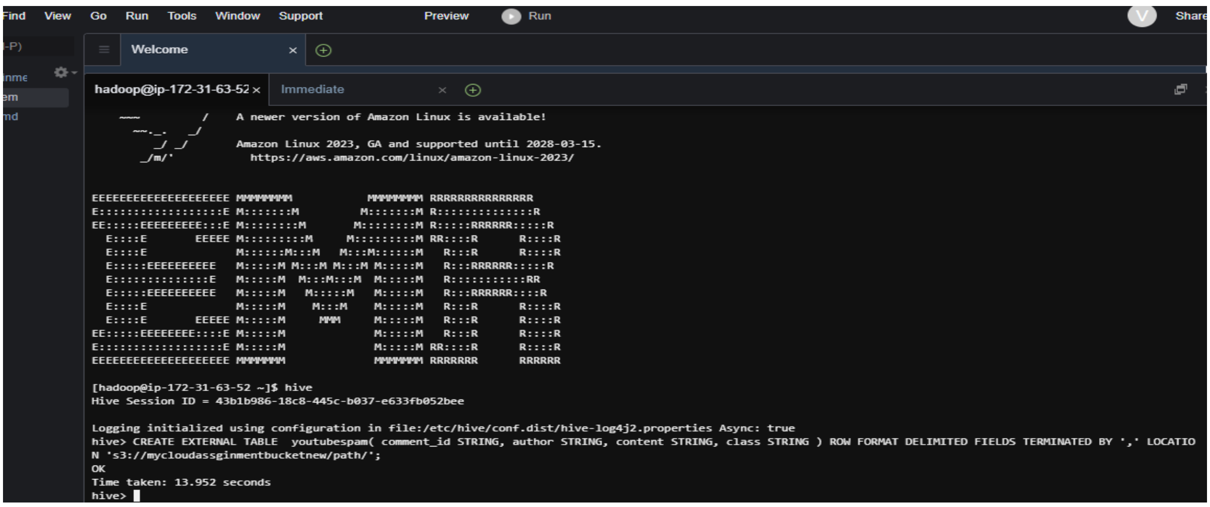


Image 12:

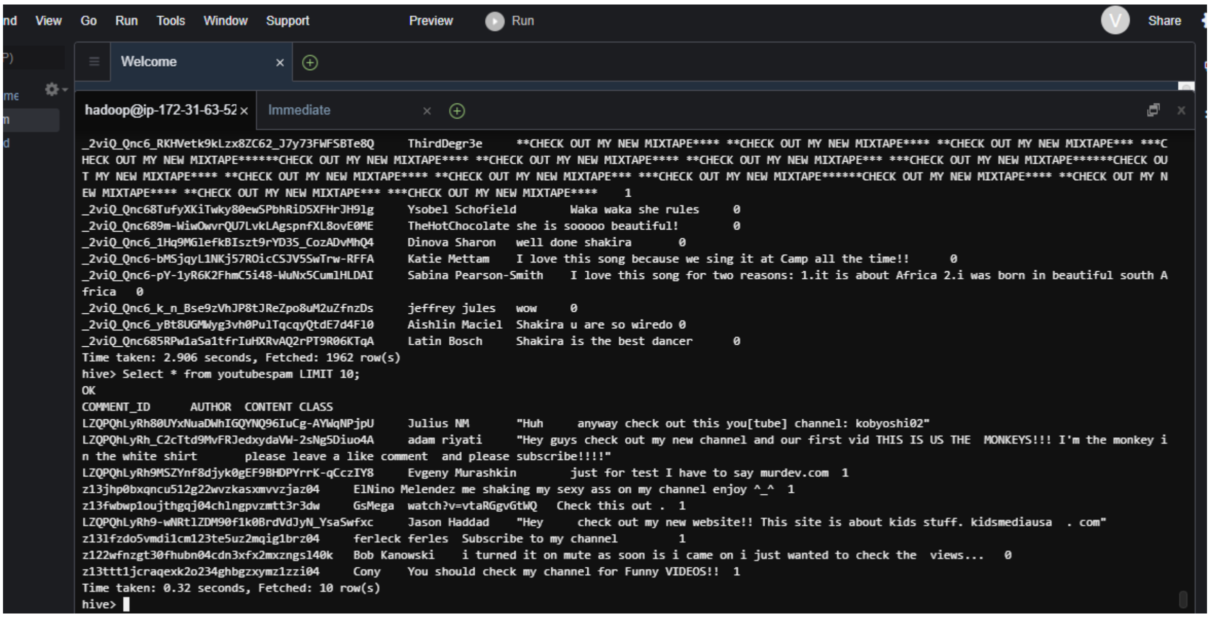


Image 13:

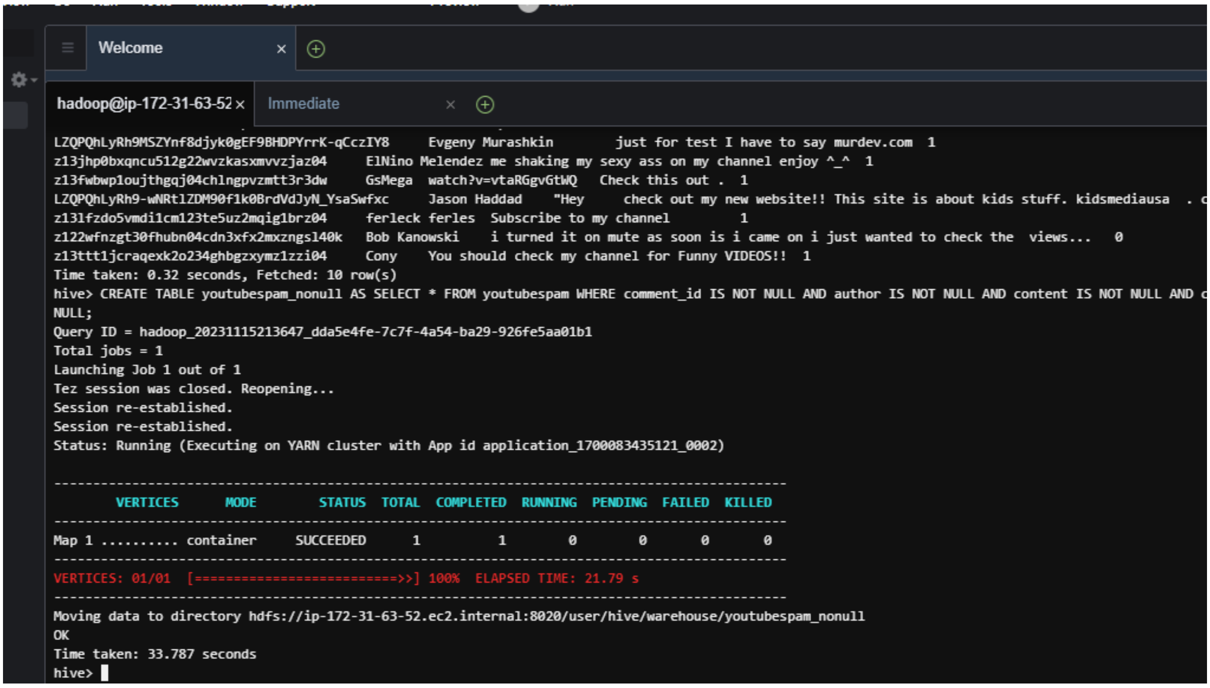


Image 14:

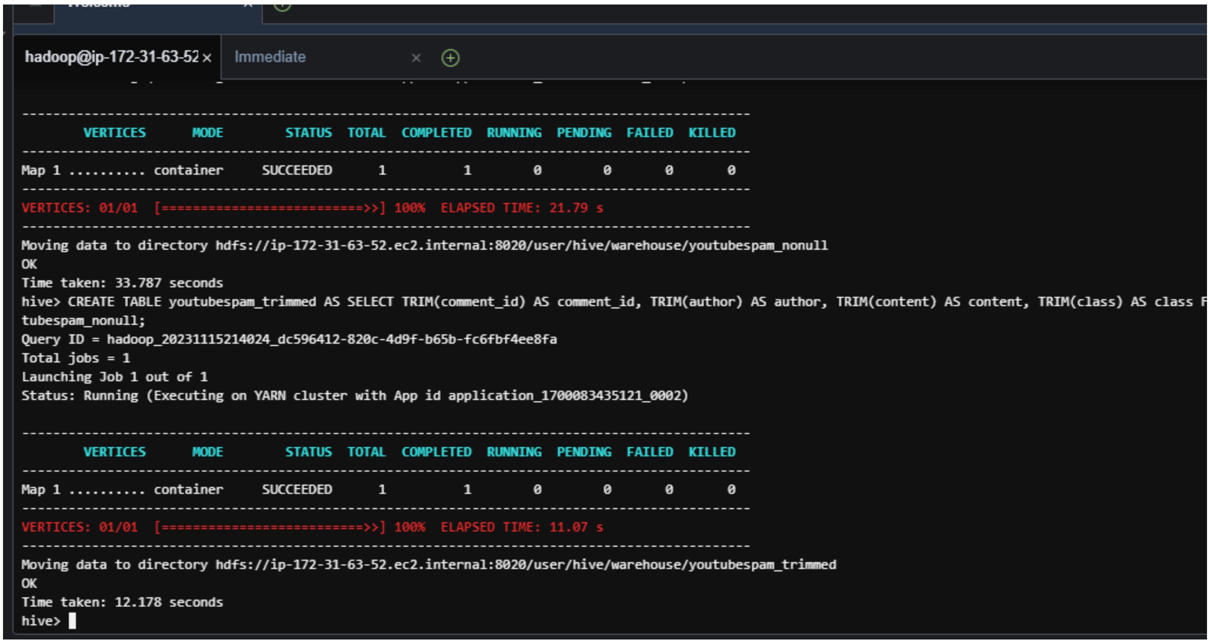


Image 15:

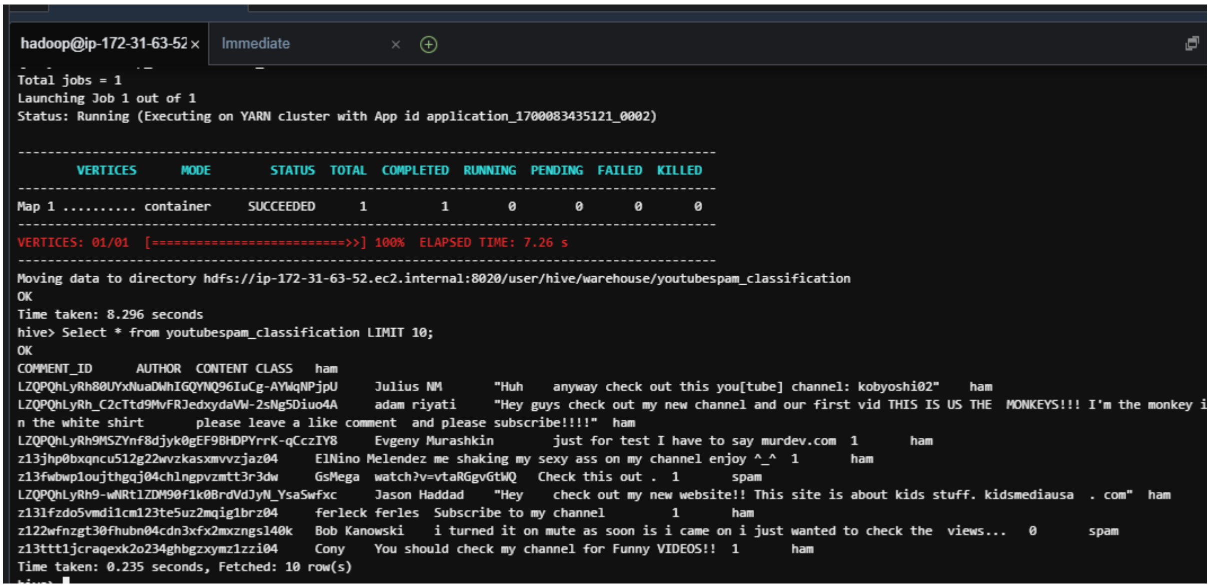
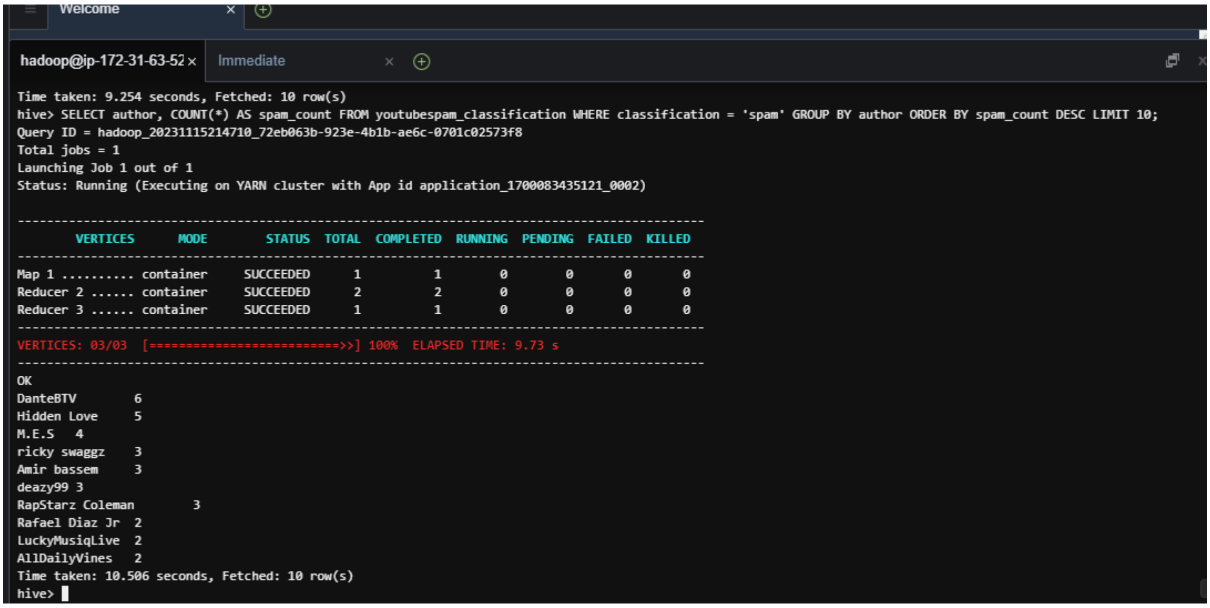


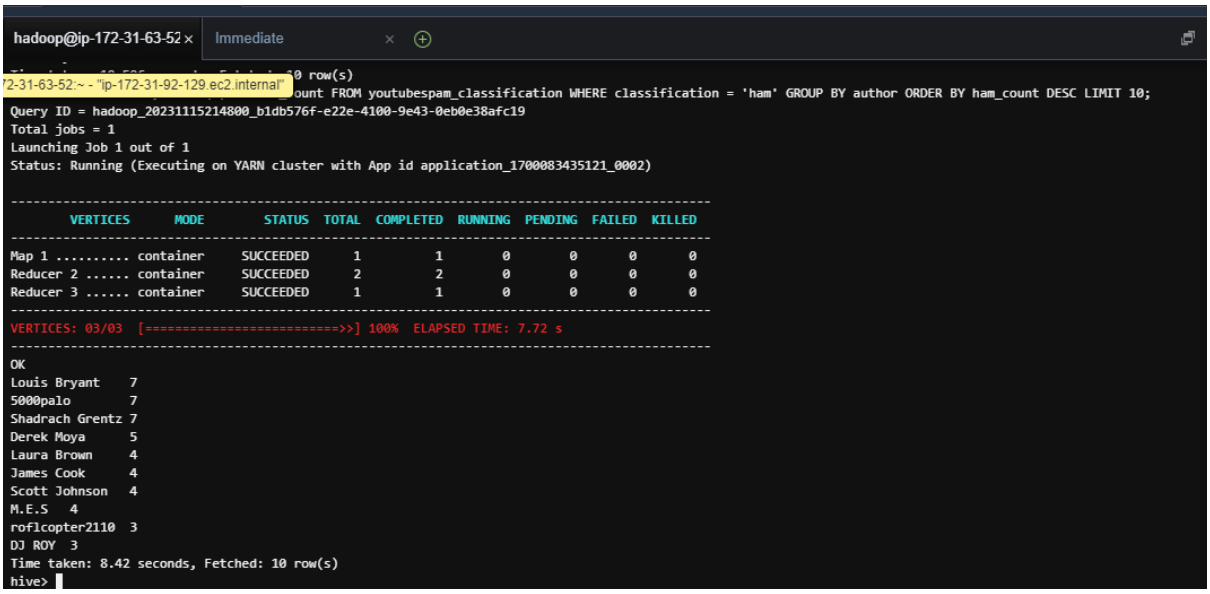
Image 16:



```
hadoop@ip-172-31-63-52    Immediate
Time taken: 9.254 seconds, Fetched: 10 row(s)
hive> SELECT author, COUNT(*) AS spam_count FROM youtubespam_classification WHERE classification = 'spam' GROUP BY author ORDER BY spam_count DESC LIMIT 10;
Query ID = hadoop_20231115214710_72eb063b-923e-4b1b-ae6c-0701c02573f8
Total jobs = 1
Launching Job 1 out of 1
Status: Running (Executing on YARN cluster with App id application_1700083435121_0002)

----------------------------------------------------------------------------------------------
        VERTICES      MODE        STATUS  TOTAL  COMPLETED  RUNNING  PENDING  FAILED  KILLED
----------------------------------------------------------------------------------------------
Map 1 .......... container     SUCCEEDED      1          1        0        0       0       0
Reducer 2 ...... container     SUCCEEDED      2          2        0        0       0       0
Reducer 3 ...... container     SUCCEEDED      1          1        0        0       0       0
----------------------------------------------------------------------------------------------
VERTICES: 03/03  [==========================>>] 100%  ELAPSED TIME: 9.73 s
----------------------------------------------------------------------------------------------
OK
DanteBTV        6
Hidden Love     5
M.E.S   4
ricky swaggz    3
Amir bassem     3
deazy99 3
RapStarz Coleman        3
Rafael Diaz Jr  2
LuckyMusiqLive  2
AllDailyVines   2
Time taken: 10.506 seconds, Fetched: 10 row(s)
hive>
```

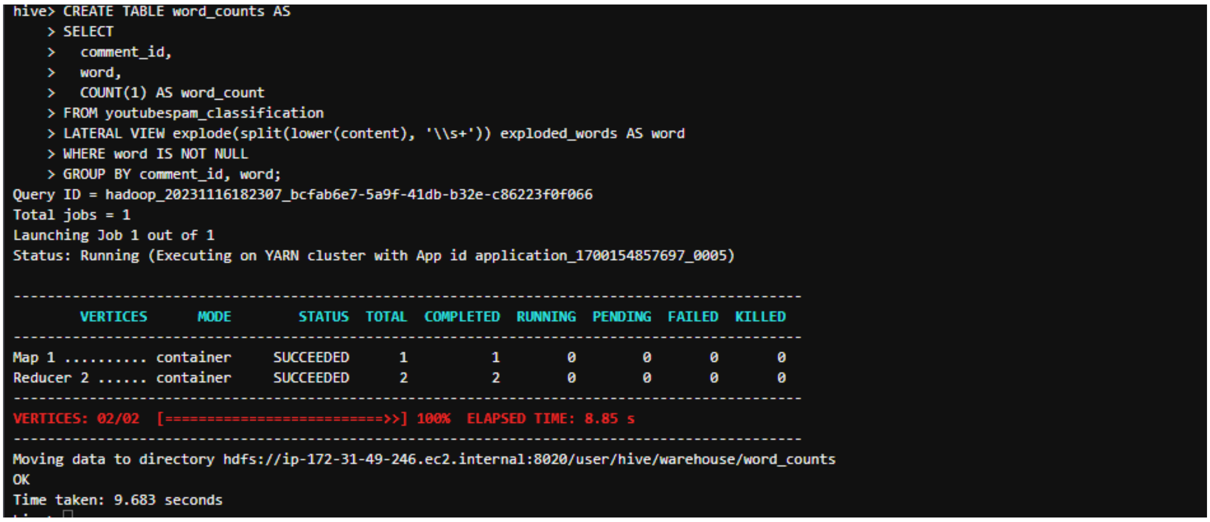
Image 17:



```
hadoop@ip-172-31-63-52    Immediate
72-31-63-52:~ - "ip-172-31-92-129.ec2.internal" 0 row(s)
ount FROM youtubespam_classification WHERE classification = 'ham' GROUP BY author ORDER BY ham_count DESC LIMIT 10;
Query ID = hadoop_20231115214800_b1db576f-e22e-4100-9e43-0eb0e38afc19
Total jobs = 1
Launching Job 1 out of 1
Status: Running (Executing on YARN cluster with App id application_1700083435121_0002)

----------------------------------------------------------------------------------------------
        VERTICES      MODE        STATUS  TOTAL  COMPLETED  RUNNING  PENDING  FAILED  KILLED
----------------------------------------------------------------------------------------------
Map 1 .......... container     SUCCEEDED      1          1        0        0       0       0
Reducer 2 ...... container     SUCCEEDED      2          2        0        0       0       0
Reducer 3 ...... container     SUCCEEDED      1          1        0        0       0       0
----------------------------------------------------------------------------------------------
VERTICES: 03/03  [==========================>>] 100%  ELAPSED TIME: 7.72 s
----------------------------------------------------------------------------------------------
OK
Louis Bryant    7
5000palo        7
Shadrach Grentz 7
Derek Moya      5
Laura Brown     4
James Cook      4
Scott Johnson   4
M.E.S   4
roflcopter2110  3
DJ ROY  3
Time taken: 8.42 seconds, Fetched: 10 row(s)
hive>
```

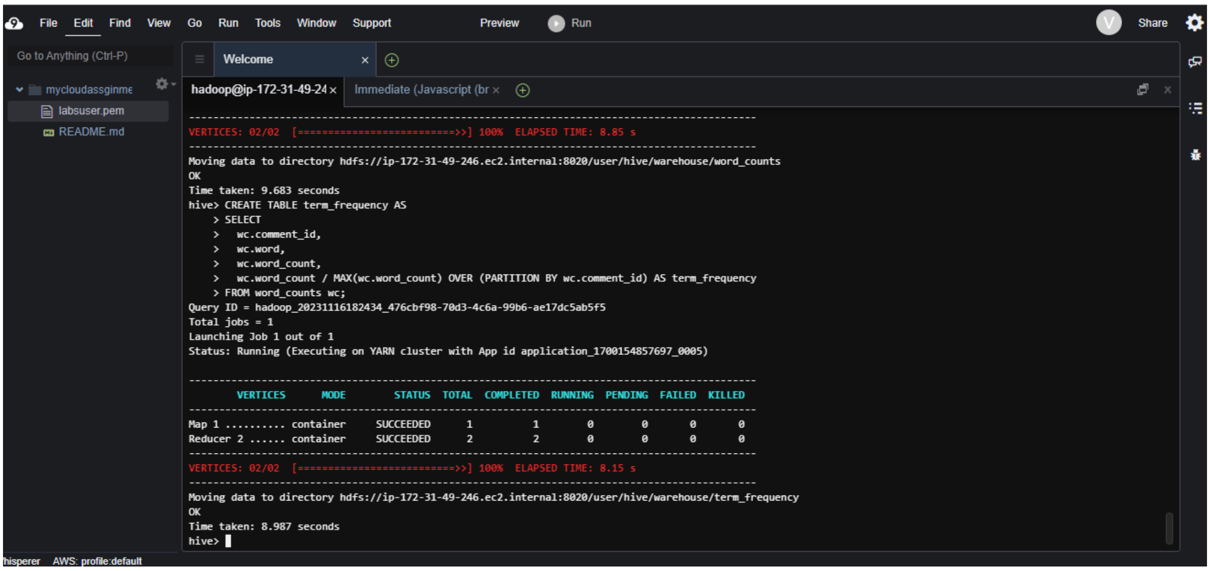
Image18:



```
hive> CREATE TABLE word_counts AS
    > SELECT
    >   comment_id,
    >   word,
    >   COUNT(1) AS word_count
    > FROM youtubespam_classification
    > LATERAL VIEW explode(split(lower(content), '\\s+')) exploded_words AS word
    > WHERE word IS NOT NULL
    > GROUP BY comment_id, word;
Query ID = hadoop_20231116182307_bcfab6e7-5a9f-41db-b32e-c86223f0f066
Total jobs = 1
Launching Job 1 out of 1
Status: Running (Executing on YARN cluster with App id application_1700154857697_0005)

----------------------------------------------------------------------------------------------
        VERTICES      MODE        STATUS  TOTAL  COMPLETED  RUNNING  PENDING  FAILED  KILLED
----------------------------------------------------------------------------------------------
Map 1 .......... container     SUCCEEDED      1          1        0        0       0       0
Reducer 2 ...... container     SUCCEEDED      2          2        0        0       0       0
----------------------------------------------------------------------------------------------
VERTICES: 02/02  [==========================>>] 100%  ELAPSED TIME: 8.85 s
----------------------------------------------------------------------------------------------
Moving data to directory hdfs://ip-172-31-49-246.ec2.internal:8020/user/hive/warehouse/word_counts
OK
Time taken: 9.683 seconds
```

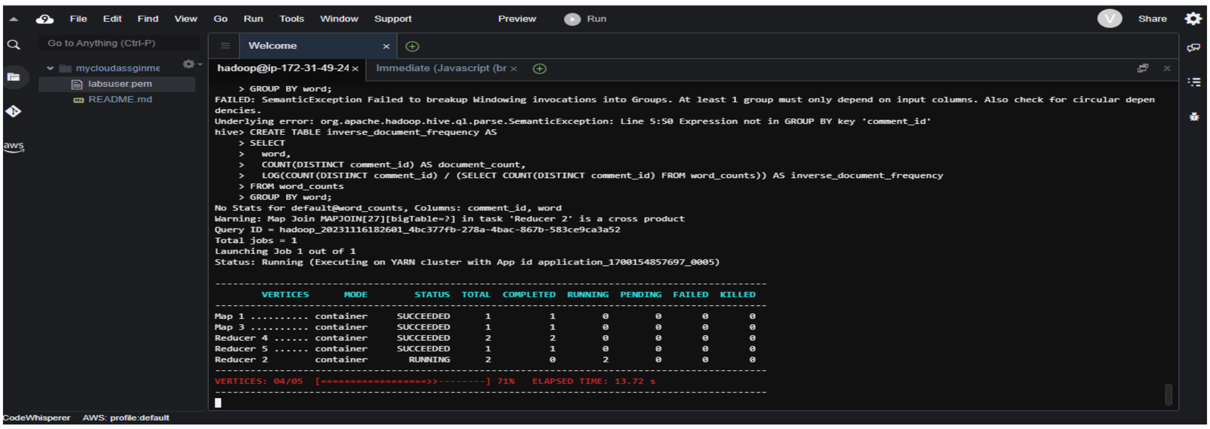
Image 19:



```
File  Edit  Find  View  Go  Run  Tools  Window  Support     Preview   Run     Share
Go to Anything (Ctrl-P)     Welcome
mycloudassginme
labsuser.pem
README.md
hadoop@ip-172-31-49-24   Immediate (Javascript (br
----------------------------------------------------------------------------------------------
VERTICES: 02/02  [==========================>>] 100%  ELAPSED TIME: 8.85 s
----------------------------------------------------------------------------------------------
Moving data to directory hdfs://ip-172-31-49-246.ec2.internal:8020/user/hive/warehouse/word_counts
OK
Time taken: 9.683 seconds
hive> CREATE TABLE term_frequency AS
    > SELECT
    >   wc.comment_id,
    >   wc.word,
    >   wc.word_count,
    >   wc.word_count / MAX(wc.word_count) OVER (PARTITION BY wc.comment_id) AS term_frequency
    > FROM word_counts wc;
Query ID = hadoop_20231116182434_476cbf98-70d3-4c6a-99b6-ae17dc5ab5f5
Total jobs = 1
Launching Job 1 out of 1
Status: Running (Executing on YARN cluster with App id application_1700154857697_0005)

----------------------------------------------------------------------------------------------
        VERTICES      MODE        STATUS  TOTAL  COMPLETED  RUNNING  PENDING  FAILED  KILLED
----------------------------------------------------------------------------------------------
Map 1 .......... container     SUCCEEDED      1          1        0        0       0       0
Reducer 2 ...... container     SUCCEEDED      2          2        0        0       0       0
----------------------------------------------------------------------------------------------
VERTICES: 02/02  [==========================>>] 100%  ELAPSED TIME: 8.15 s
----------------------------------------------------------------------------------------------
Moving data to directory hdfs://ip-172-31-49-246.ec2.internal:8020/user/hive/warehouse/term_frequency
OK
Time taken: 8.987 seconds
hive>
hisperer   AWS: profile:default
```

Image 20:



```
File  Edit  Find  View  Go  Run  Tools  Window  Support     Preview   Run     Share
Go to Anything (Ctrl-P)     Welcome
mycloudassginme
labsuser.pem
README.md
hadoop@ip-172-31-49-24   Immediate (Javascript (br
    > GROUP BY word;
FAILED: SemanticException Failed to breakup Windowing invocations into Groups. At least 1 group must only depend on input columns. Also check for circular depen
dencies.
Underlying error: org.apache.hadoop.hive.ql.parse.SemanticException: Line 5:50 Expression not in GROUP BY key 'comment_id'
hive> CREATE TABLE inverse_document_frequency AS
    > SELECT
    >   word,
    >   COUNT(DISTINCT comment_id) AS document_count,
    >   LOG(COUNT(DISTINCT comment_id) / (SELECT COUNT(DISTINCT comment_id) FROM word_counts)) AS inverse_document_frequency
    > FROM word_counts
    > GROUP BY word;
No Stats for default@word_counts, Columns: comment_id, word
Warning: Map Join MAPJOIN[27][bigTable=?] in task 'Reducer 2' is a cross product
Query ID = hadoop_20231116182601_4bc377fb-278a-4bac-867b-583ce9ca3a52
Total jobs = 1
Launching Job 1 out of 1
Status: Running (Executing on YARN cluster with App id application_1700154857697_0005)

----------------------------------------------------------------------------------------------
        VERTICES      MODE        STATUS  TOTAL  COMPLETED  RUNNING  PENDING  FAILED  KILLED
----------------------------------------------------------------------------------------------
Map 1 .......... container     SUCCEEDED      1          1        0        0       0       0
Map 3 .......... container     SUCCEEDED      1          1        0        0       0       0
Reducer 4 ...... container     SUCCEEDED      2          2        0        0       0       0
Reducer 5 ...... container     SUCCEEDED      1          1        0        0       0       0
Reducer 2        container       RUNNING      2          0        2        0       0       0
----------------------------------------------------------------------------------------------
VERTICES: 04/05  [==================>>--------] 71%   ELAPSED TIME: 13.72 s
----------------------------------------------------------------------------------------------
CodeWhisperer   AWS: profile:default
```

Image 21:

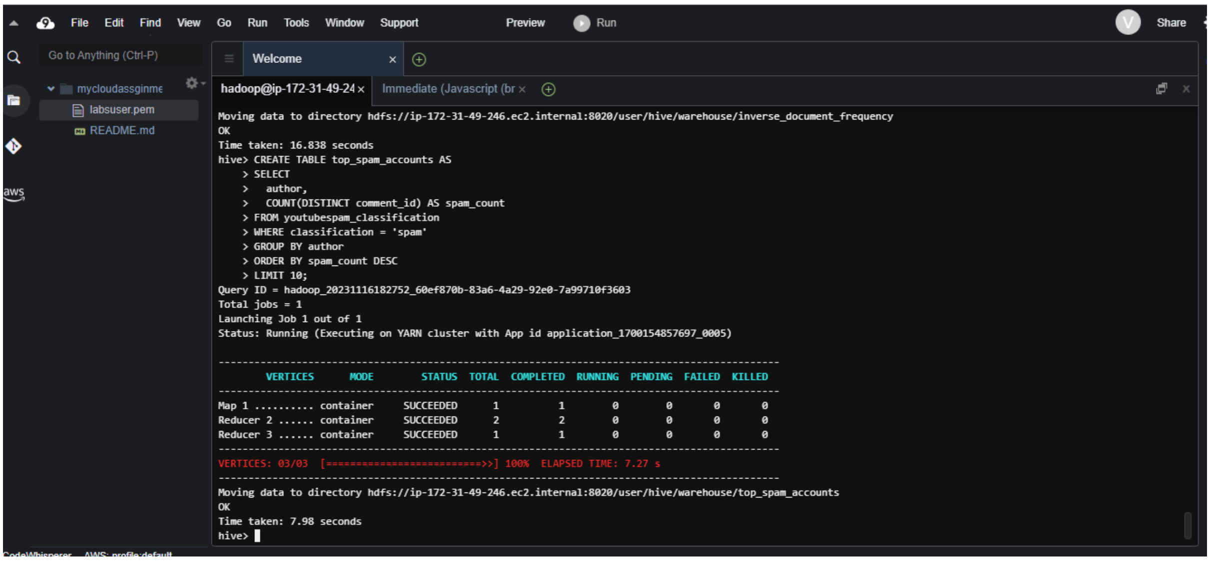


Image 22:

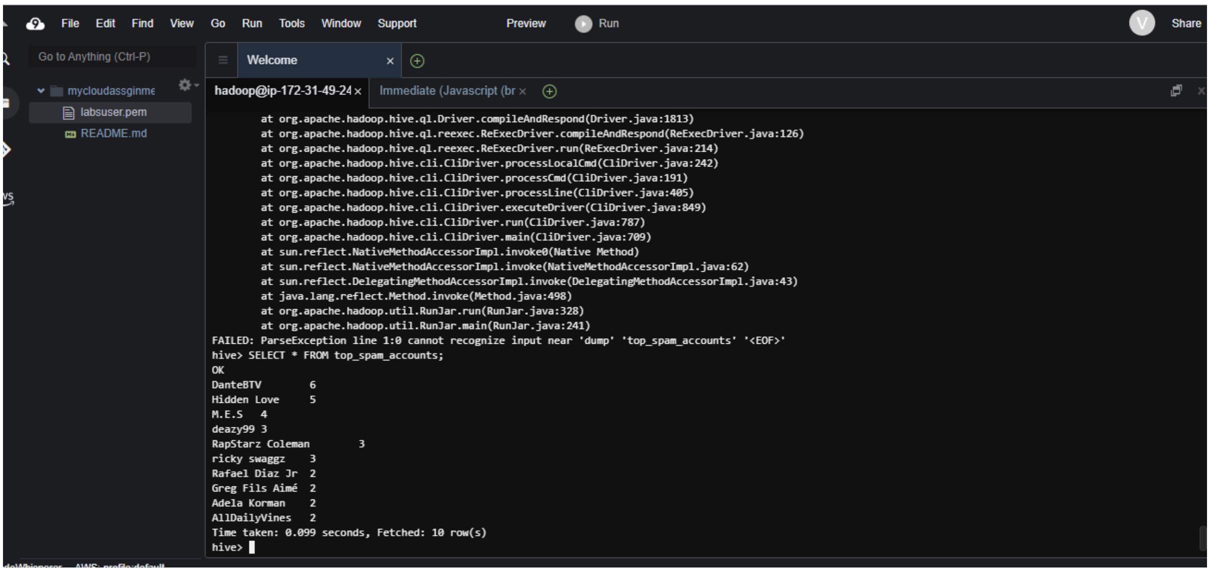


Image 23

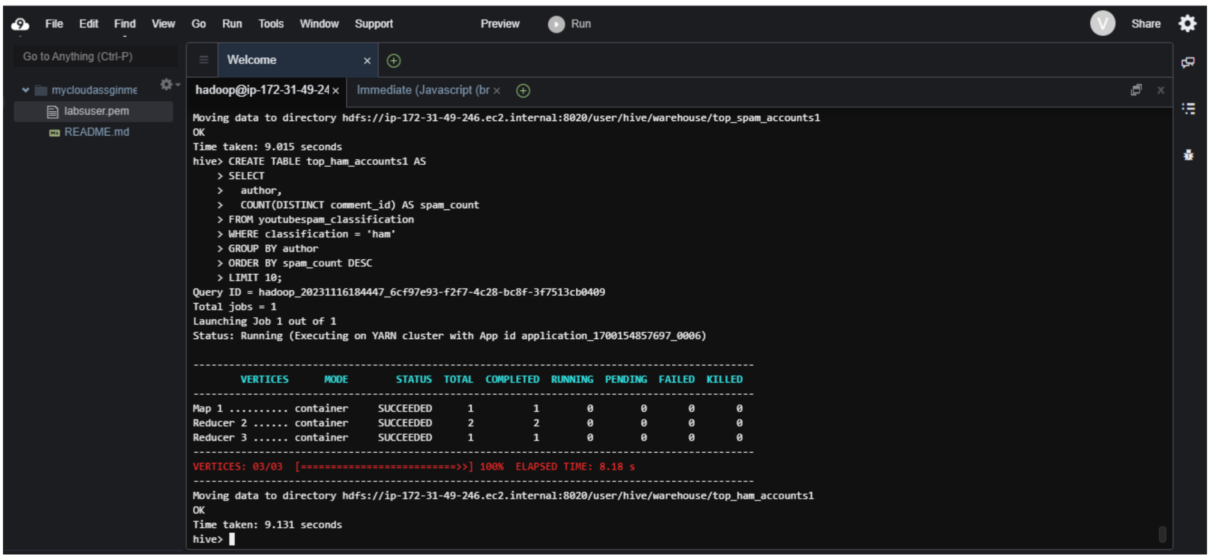


Image 24:

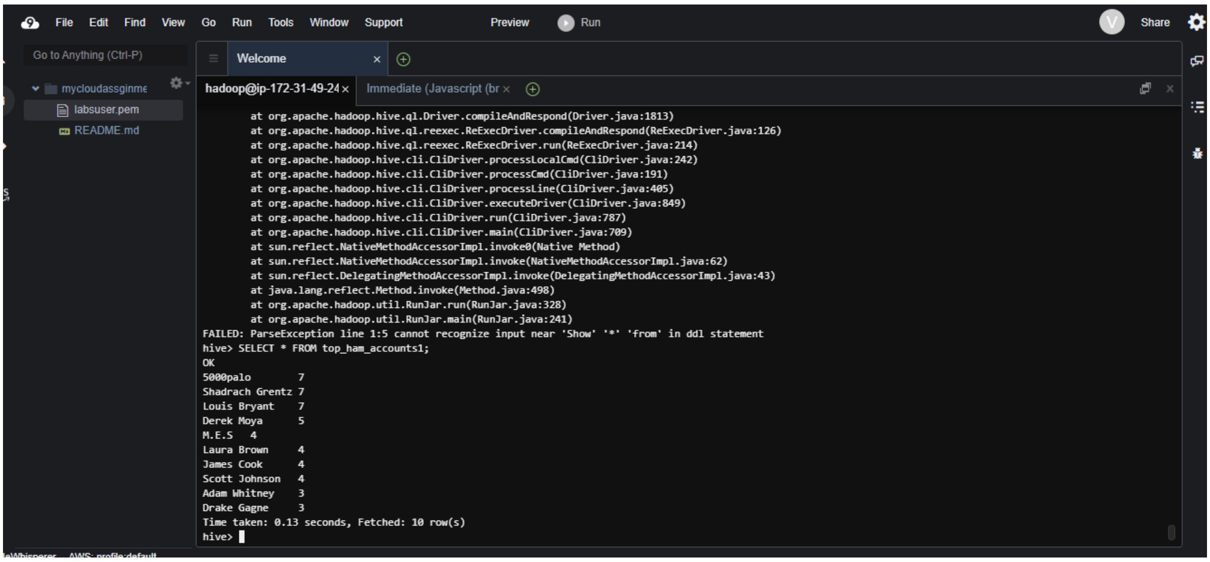


Image 25: