PreRequisite:

Creating multiple node cluster setup in AWS EMR & loading data to the S3 bucket.

1)Creating Table & loading data from S3 bucket to Hive Table:

CREATE EXTERNAL TABLE youtubespam(comment_id STRING, author STRING, content STRING, class STRING) ROW FORMAT DELIMITED FIELDS TERMINATED BY ',' LOCATION 's3://mycloudbucketass/path/';

LOAD DATA INPATH 's3://mycloudbucketass/newdata.csv' INTO TABLE youtubespam;

2)Checking if the data loaded is correct:

Select * from youtubespam LIMIT 10;

3)Cleaning Dataset:

Removing null values:

- 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 values:

- CREATE TABLE youtubespam_trimmed AS SELECT TRIM(comment_id) AS comment_id, TRIM(author) AS author, TRIM(content) AS content, TRIM(class) AS class FROM youtubespam_nonull;

4)A basic regular expression-based classification rule is used in SQL queries.

CREATE TABLE youtubespam_classification AS SELECT *, CASE WHEN content RLIKE '(Check|subscribe|views|trading)' THEN 'spam' ELSE 'ham' END AS classification FROM youtubespam_trimmed;

Select * from youtubespam_classification LIMIT 10;

5)Query to find the Top 10 SPAM accounts:

SELECT author, COUNT(*) AS spam_count FROM youtubespam_classification WHERE classification = 'spam' GROUP BY author ORDER BY spam_count DESC LIMIT 10;

6)Query to find the Top 10 HAM accounts:

SELECT author, COUNT(*) AS ham_count FROM youtubespam_classification WHERE classification = 'ham' GROUP BY author ORDER BY ham_count DESC LIMIT 10;

7)Tokenization and Word Count

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;

8)Calculate Term Frequency (TF)

```
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;
```

9)Calculate Inverse Document Frequency (IDF)

```
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;
```

10)Calculate TF-IDF

```
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;
```

11)Filter for Top 10 Spam Accounts

```
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;
```

12)Filter for Top 10 Spam Keywords for Each Top 10 Spam Account

```
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;
```

13)Filter for Top 10 Ham Accounts

```
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;
```