BIG DATA ANALYTICS LAB

1. To implement a simple map-reduce job that builds an inverted index on the set of input documents in Hadoop, we can follow these steps:

1. Input data preparation:
   We need to prepare the input data in a format that can be processed by MapReduce. In this case, we can assume that the input data is a set of text documents, where each document is stored as a separate file in a directory. The contents of each file should be in plain text format.

2. Mapper function:
   The mapper function takes a key-value pair as input, where the key is the file name and the value is the contents of the file. The mapper function then tokenizes the contents into individual words and emits a key-value pair for each word in the document. The key is the word and the value is a pair consisting of the file name and the frequency of the word in that file.

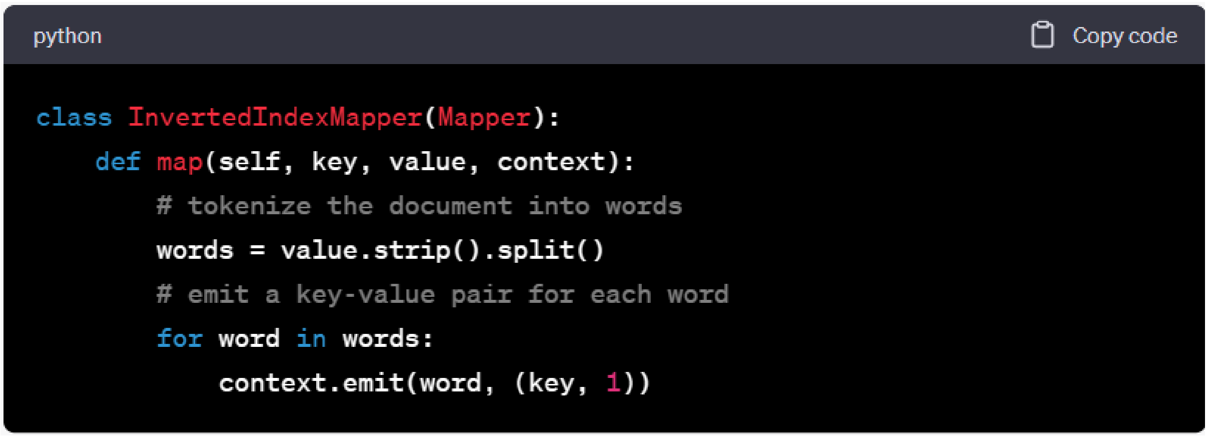
3. Reducer function:
   The reducer function takes a key-value pair as input, where the key is a word and the value is a list of pairs (file name, frequency). The reducer function then aggregates the pairs for each file, producing a list of documents where the word occurs and the frequency of the word in each document.

4. Output format:
   The output of the reducer function is the inverted index, which is a mapping from words to a list of documents where the word occurs and the frequency of the word in each document.

Here is the implementation of the above steps in Hadoop:

1. Input data preparation:
   Assume that the input data is stored in a directory called "input". Each file in the directory contains the contents of a single document.

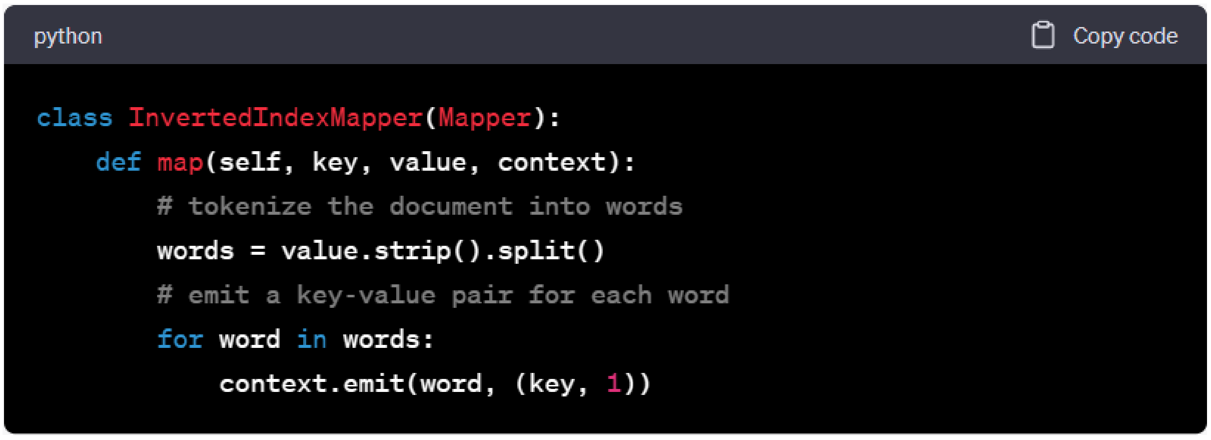
2. Mapper function:
   The following mapper function tokenizes the input documents and emits a key-value pair for each word in the document:



```python
class InvertedIndexMapper(Mapper):
    def map(self, key, value, context):
        # tokenize the document into words
        words = value.strip().split()
        # emit a key-value pair for each word
        for word in words:
            context.emit(word, (key, 1))
```

3. Reducer function:

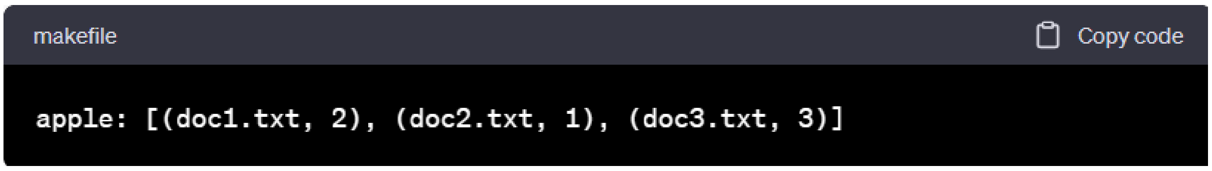
The following reducer function aggregates the pairs for each file and produces a list of documents where the word occurs and the frequency of the word in each document:



```python
class InvertedIndexMapper(Mapper):
    def map(self, key, value, context):
        # tokenize the document into words
        words = value.strip().split()
        # emit a key-value pair for each word
        for word in words:
            context.emit(word, (key, 1))
```

4. Output format:

The output of the reducer function is the inverted index, which is a mapping from words to a list of documents where the word occurs and the frequency of the word in each document. The output can be written to a file or printed to the console. Here is an example of the output for the word "apple":



```makefile
apple: [(doc1.txt, 2), (doc2.txt, 1), (doc3.txt, 3)]
```

This output indicates that the word "apple" occurs twice in "doc1.txt", once in "doc2.txt", and three times in "doc3.txt".

2. Process big data in HBase

To process big data in HBase in Hadoop, we can follow these steps:

1. Set up a Hadoop cluster:

First, we need to set up a Hadoop cluster with HBase installed. This can be done using a distribution like Apache Hadoop or a cloud service like Amazon EMR or Google Cloud Dataproc.

2. Define the data model:

We need to define the data model for our HBase tables, including the row key, column families, and columns. This will depend on the structure of the data and the requirements of the application.

3. Ingest data into HBase:

We can ingest data into HBase using Hadoop MapReduce or Spark. The input data can be stored in a distributed file system like HDFS or S3, and can be processed in parallel by the map tasks.

4. MapReduce processing:

We can use MapReduce to process the data in HBase tables. MapReduce can be used to perform complex aggregations, filtering, and transformations on the data. The output of the MapReduce job can be stored back into HBase or written to a file in a distributed file system.

5. Analyze data with HBase API:

We can use the HBase API to analyze the data in HBase tables. The API provides several methods for querying and manipulating the data, including get, put, scan, delete, and increment. We can use these methods to perform data analysis, and can also use Apache Phoenix to run SQL-like queries on HBase tables.

6. Visualize data:

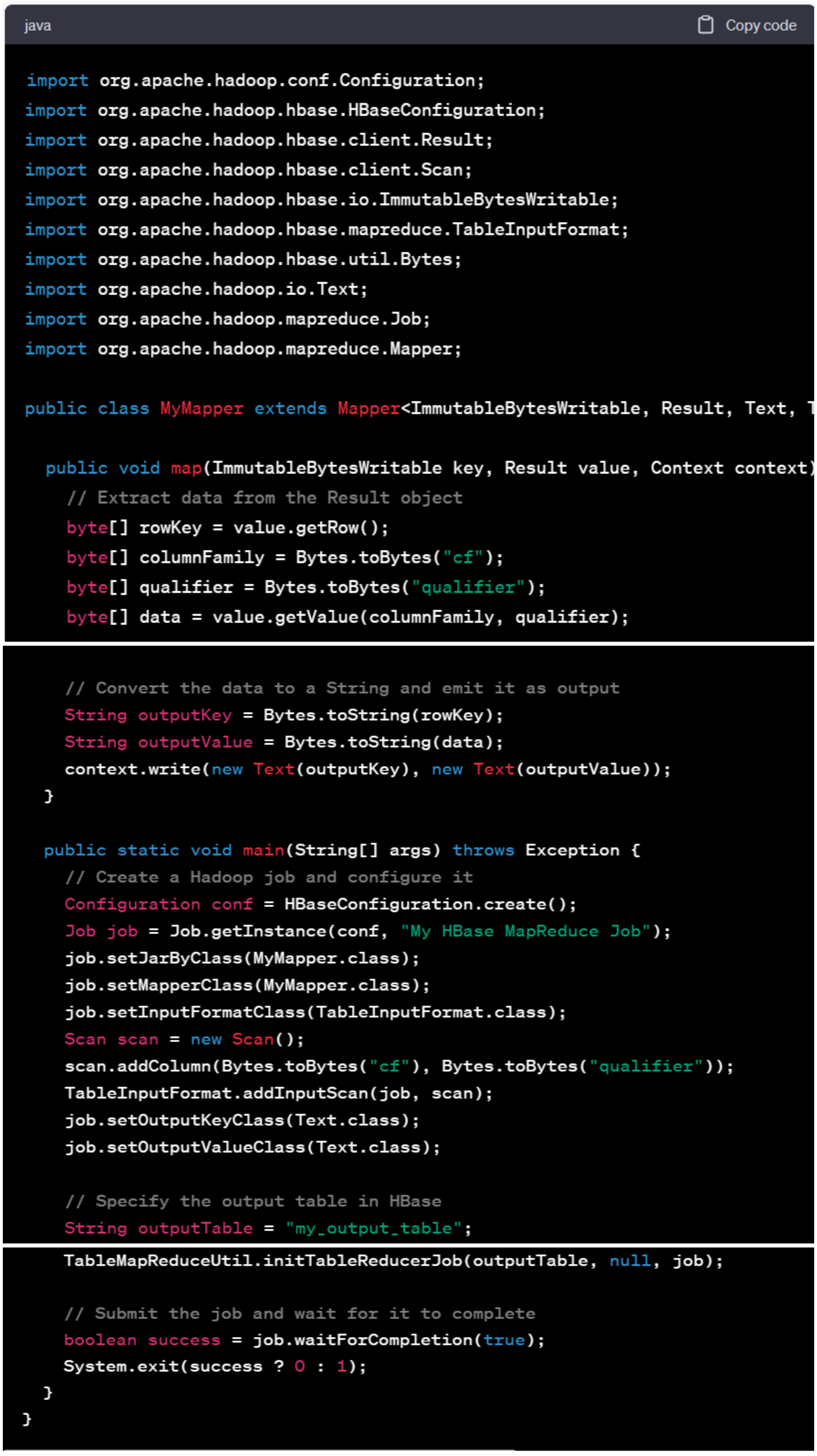
We can visualize the results of data analysis using tools like Apache Zeppelin or Tableau. Apache Zeppelin is a web-based notebook that supports several languages, including SQL, Python, and R, and can be used to visualize data in real-time. Tableau is a popular data visualization tool that can be used to create interactive dashboards and reports.

Overall, Hadoop and HBase provide a powerful platform for storing and processing big data, and can be used in a variety of applications, including real-time analytics, machine learning, and IoT data processing.

To process big data in HBase in Hadoop code, we can follow these steps:

To process big data in HBase using Hadoop, we can use the HBase MapReduce API to read data from HBase tables, process it using Hadoop's MapReduce framework, and then write the results back to HBase.

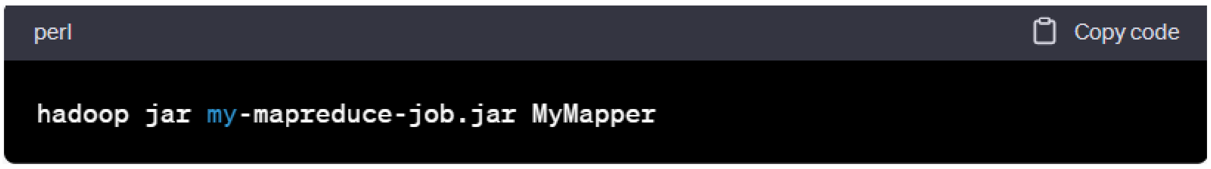
Here's an example of how to read data from an HBase table using Hadoop's MapReduce API:



```java
import org.apache.hadoop.conf.Configuration;
import org.apache.hadoop.hbase.HBaseConfiguration;
import org.apache.hadoop.hbase.client.Result;
import org.apache.hadoop.hbase.client.Scan;
import org.apache.hadoop.hbase.io.ImmutableBytesWritable;
import org.apache.hadoop.hbase.mapreduce.TableInputFormat;
import org.apache.hadoop.hbase.util.Bytes;
import org.apache.hadoop.io.Text;
import org.apache.hadoop.mapreduce.Job;
import org.apache.hadoop.mapreduce.Mapper;

public class MyMapper extends Mapper<ImmutableBytesWritable, Result, Text, T

  public void map(ImmutableBytesWritable key, Result value, Context context)
    // Extract data from the Result object
    byte[] rowKey = value.getRow();
    byte[] columnFamily = Bytes.toBytes("cf");
    byte[] qualifier = Bytes.toBytes("qualifier");
    byte[] data = value.getValue(columnFamily, qualifier);

    // Convert the data to a String and emit it as output
    String outputKey = Bytes.toString(rowKey);
    String outputValue = Bytes.toString(data);
    context.write(new Text(outputKey), new Text(outputValue));
  }

  public static void main(String[] args) throws Exception {
    // Create a Hadoop job and configure it
    Configuration conf = HBaseConfiguration.create();
    Job job = Job.getInstance(conf, "My HBase MapReduce Job");
    job.setJarByClass(MyMapper.class);
    job.setMapperClass(MyMapper.class);
    job.setInputFormatClass(TableInputFormat.class);
    Scan scan = new Scan();
    scan.addColumn(Bytes.toBytes("cf"), Bytes.toBytes("qualifier"));
    TableInputFormat.addInputScan(job, scan);
    job.setOutputKeyClass(Text.class);
    job.setOutputValueClass(Text.class);

    // Specify the output table in HBase
    String outputTable = "my_output_table";
    TableMapReduceUtil.initTableReducerJob(outputTable, null, job);

    // Submit the job and wait for it to complete
    boolean success = job.waitForCompletion(true);
    System.exit(success ? 0 : 1);
  }
}
```

In this example, we define a Mapper class that reads data from an HBase table and emits it as output. We also configure a Hadoop job to use the TableInputFormat to read data from the table, and to use the TableMapReduceUtil class to write the output back to another HBase table.

Once we have our Mapper class and Hadoop job configured, we can submit the job using the following command:

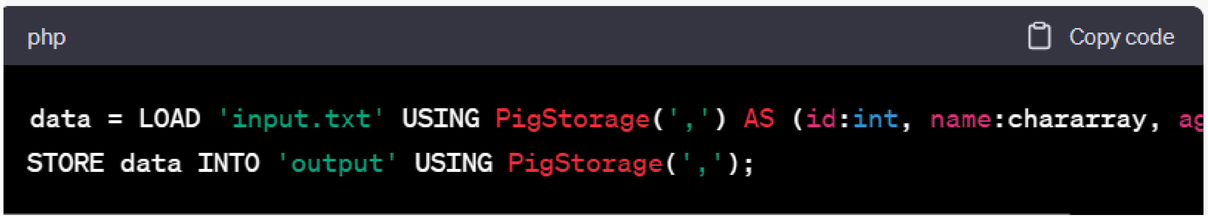


```
hadoop jar my-mapreduce-job.jar MyMapper
```

This will run the job and process the data in the HBase table using Hadoop's MapReduce framework.

3. Store and retrieve data in Pig
Pig is a high-level platform for creating MapReduce programs used to process large datasets. It provides a simple language called Pig Latin, which allows users to write data transformation queries using a SQL-like syntax.

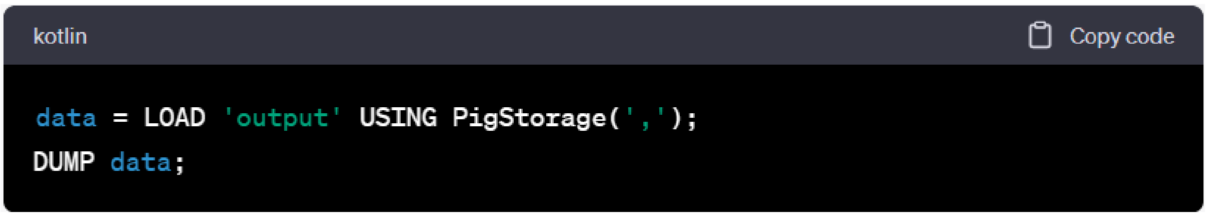
To store data in Pig, you can use the STORE operator, which allows you to write data to a file or Hadoop Distributed File System (HDFS). Here's an example of how to store data in Pig:



```
data = LOAD 'input.txt' USING PigStorage(',') AS (id:int, name:chararray, ag
STORE data INTO 'output' USING PigStorage(',');
```

In this example, we are loading data from a file called "input.txt" and storing it in a relation called "data". We are then using the STORE operator to write the data to a file called "output" using the PigStorage function to specify the delimiter as a comma (',').

To retrieve data in Pig, you can use the LOAD operator, which allows you to read data from a file or HDFS. Here's an example of how to retrieve data in Pig:



```
data = LOAD 'output' USING PigStorage(',');
DUMP data;
```

In this example, we are loading data from the "output" file using the LOAD operator and storing it in a relation called "data". We are then using the DUMP operator to output the data to the console. The PigStorage function is used again to specify the delimiter as a comma (',').

4. Perform Social media analysis using Cassandra

Performing social media analysis using Cassandra involves storing social media data in Cassandra and then analyzing the data using CQL (Cassandra Query Language) queries. Cassandra is a distributed NoSQL database that can be used to store and analyze large amounts of data. To perform social media analysis using Cassandra, you can follow the steps outlined below:

1. Data collection: The first step is to collect data from social media platforms like Twitter, Facebook, or LinkedIn. This can be done using APIs provided by the respective platforms or using third-party tools.

2. Data modeling: Once the data is collected, you need to model it in Cassandra. You can create a keyspace to store your data and define one or more tables within the keyspace to hold your data. You can also create secondary indexes on certain columns to enable faster querying.

3. Data ingestion: After modeling the data, you need to ingest it into Cassandra. You can use the DataStax Bulk Loader to load large amounts of data into Cassandra quickly.

4. Data analysis: Once the data is ingested into Cassandra, you can perform social media analysis by running queries on the data. For example, you can use CQL (Cassandra Query Language) to find the top hashtags used in tweets or the most frequently mentioned users.

Here's an example CQL query to find the top hashtags used in tweets:

vbnet Copy code

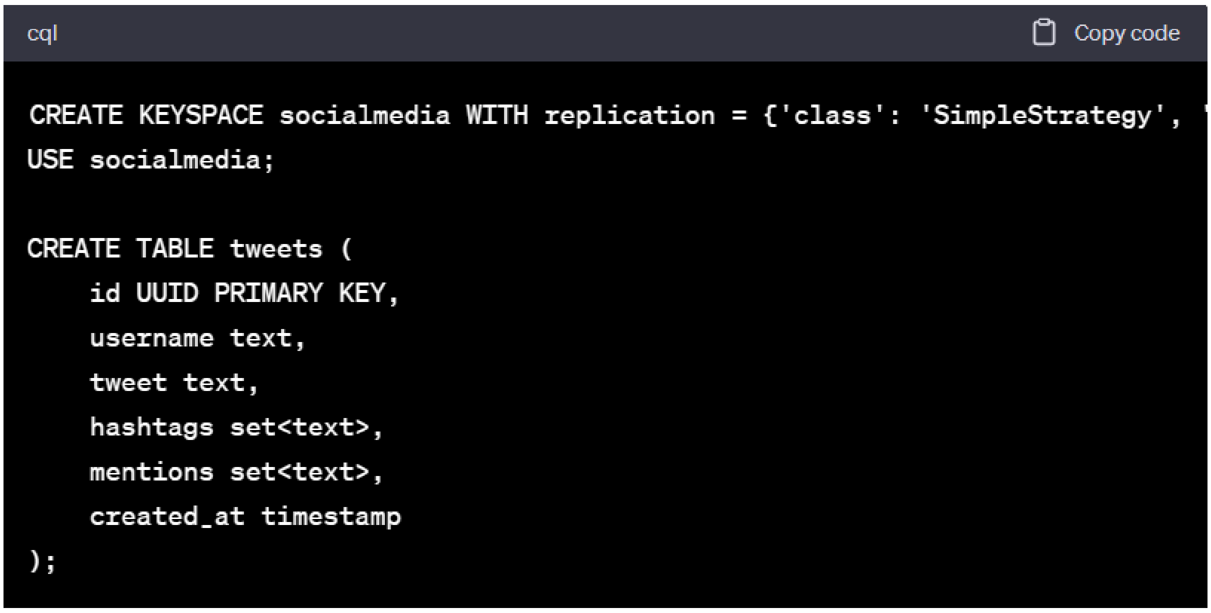
```
SELECT hashtag, COUNT(*) as count FROM tweets WHERE hashtag IS NOT NULL GROU
```

In this example, we are selecting the hashtag column from the tweets table, filtering out any null values, grouping the results by hashtag, counting the number of occurrences of each hashtag, ordering the results in descending order by count, and limiting the output to the top 10 results.

By using Cassandra for social media analysis, you can store and analyze large amounts of data quickly and easily, enabling you to gain insights into social media trends and user behavior.

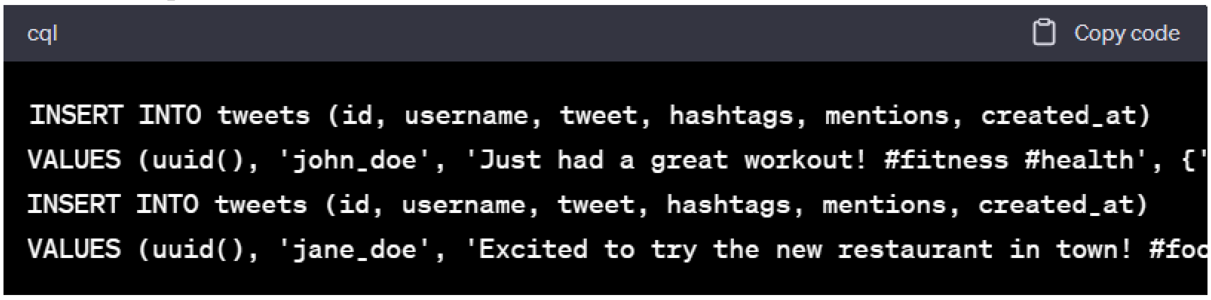
Here's an example of how to perform social media analysis using Cassandra:

1. Create a keyspace and table to store social media data:



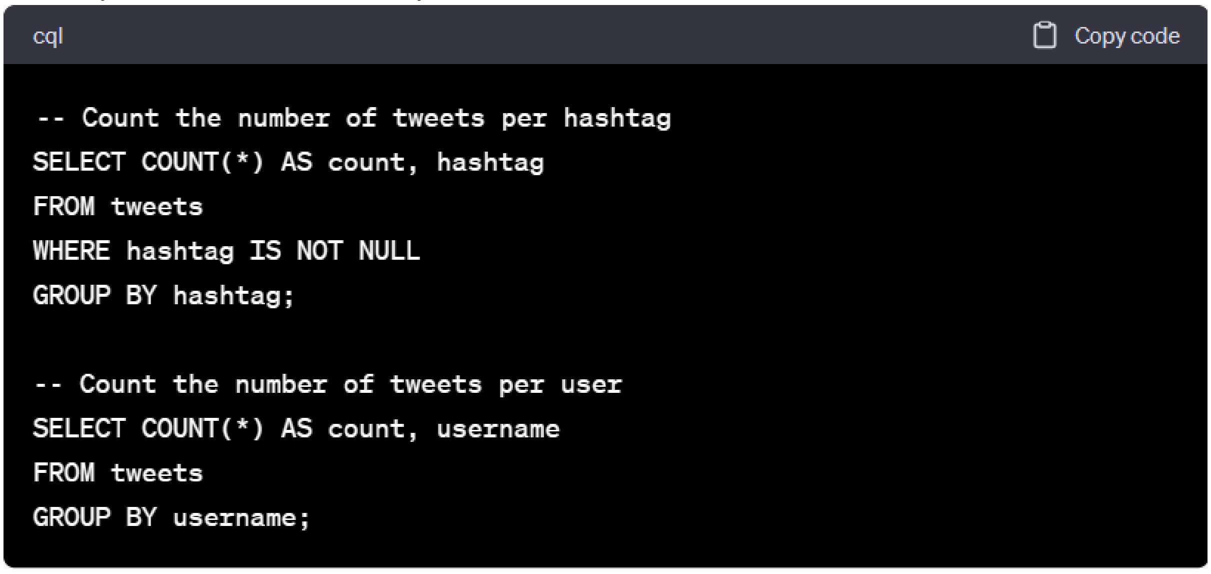
```
CREATE KEYSPACE socialmedia WITH replication = {'class': 'SimpleStrategy', '
USE socialmedia;

CREATE TABLE tweets (
    id UUID PRIMARY KEY,
    username text,
    tweet text,
    hashtags set<text>,
    mentions set<text>,
    created_at timestamp
);
```

2. Insert sample data into the tweets table:



```
INSERT INTO tweets (id, username, tweet, hashtags, mentions, created_at)
VALUES (uuid(), 'john_doe', 'Just had a great workout! #fitness #health', {'
INSERT INTO tweets (id, username, tweet, hashtags, mentions, created_at)
VALUES (uuid(), 'jane_doe', 'Excited to try the new restaurant in town! #foo
```

3. Query the tweets table to analyze the social media data:



```
-- Count the number of tweets per hashtag
SELECT COUNT(*) AS count, hashtag
FROM tweets
WHERE hashtag IS NOT NULL
GROUP BY hashtag;

-- Count the number of tweets per user
SELECT COUNT(*) AS count, username
FROM tweets
GROUP BY username;
```

These queries demonstrate how you can analyze social media data stored in Cassandra by counting the number of tweets per hashtag and per user.

5. Buyer event analytics using Cassandra on suitable product sales data.

By using Cassandra for buyer event analytics, you can store and analyze large amounts of sales data quickly and easily, enabling you to gain insights into buyer behavior and product trends.

To perform buyer event analytics using Cassandra on suitable product sales data, you can follow these steps:

1. Data modeling: The first step is to model the data in Cassandra. You can create a keyspace to store your data and define one or more tables within the keyspace to hold your data. In this case, you will want to create a table to store the sales data, with columns for the buyer ID, product ID, date of purchase, and any other relevant information.

2. Data ingestion: After modeling the data, you need to ingest it into Cassandra. You can use the DataStax Bulk Loader to load large amounts of data into Cassandra quickly.

3. Data analysis: Once the data is ingested into Cassandra, you can perform buyer event analytics by running queries on the data. For example, you can use CQL (Cassandra Query Language) to find the top-selling products by date range, the total sales for each buyer, and the products most frequently purchased by each buyer.

Here's an example CQL query to find the top-selling products by date range:

sql Copy code

```sql
SELECT product_id, COUNT(*) as count FROM sales WHERE purchase_date >= '2022
```

In this example, we are selecting the product_id column from the sales table, filtering the results by purchase date between January 1, 2022, and March 31, 2022, grouping the results by product_id, counting the number of occurrences of each product_id, ordering the results in descending order by count, and limiting the output to the top 10 results.

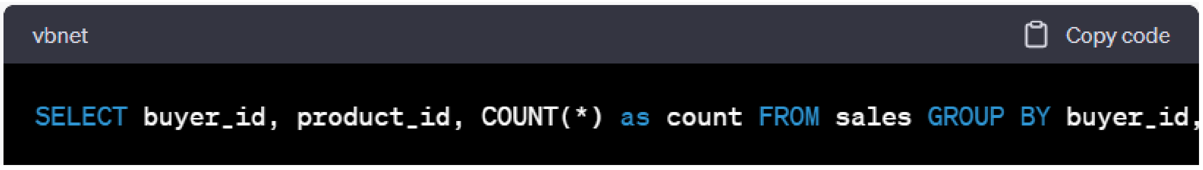
You can also use CQL to find the total sales for each buyer:

vbnet Copy code

```vbnet
SELECT buyer_id, SUM(price) as total_sales FROM sales GROUP BY buyer_id;
```

In this example, we are selecting the buyer_id column and the sum of the price column from the sales table, grouping the results by buyer_id to calculate the total sales for each buyer.

Finally, you can use CQL to find the products most frequently purchased by each buyer:



```
SELECT buyer_id, product_id, COUNT(*) as count FROM sales GROUP BY buyer_id,
```

In this example, we are selecting the buyer_id and product_id columns from the sales table, grouping the results by both buyer_id and product_id, counting the number of occurrences of each buyer_id and product_id combination, and ordering the results by buyer_id and count in descending order.

6. Using Power Pivot (Excel) Perform the following on any dataset

Power Pivot in Excel is a powerful tool for performing big data analytics on large datasets. Here are some examples of what you can do with Power Pivot:

1. Importing and combining data: With Power Pivot, you can easily import data from multiple sources such as CSV, Excel, SQL Server, Access, and others. Once the data is imported, you can combine it to create a single data model for analysis.

2. Data modeling: Power Pivot allows you to create complex data models to organize and analyze your data. You can define relationships between tables, create calculated columns, and add measures for aggregating data.

3. Data analysis: Once the data is modeled, you can use Power Pivot to perform complex data analysis. For example, you can create pivot tables and pivot charts to summarize and visualize data, or use DAX (Data Analysis Expressions) formulas to calculate advanced metrics.

4. Big data analysis: Power Pivot is designed to handle large datasets with ease. You can load millions of rows of data and still perform complex calculations and analysis quickly.

Here's an example of how to use Power Pivot to perform big data analytics:

1. Import data: Start by importing your data into Power Pivot. You can do this by selecting the "From Other Sources" option in the Power Pivot tab in Excel and selecting your data source.

2. Create data model: Once the data is imported, create a data model by defining relationships between tables, creating calculated columns, and adding measures for aggregating data.

3. Analyze data: Use the data model to perform analysis on your data. For example, you can create a pivot table to summarize sales data by region and product, and use DAX formulas to calculate metrics such as average sales per product.

4. Visualize data: Use pivot charts to visualize your data and gain insights into trends and patterns. For example, you can create a pivot chart that shows sales trends by product over time.

a) Big Data Analytics
With Power Pivot, you can easily handle and analyze large datasets, allowing you to gain insights into your data and make informed decisions.

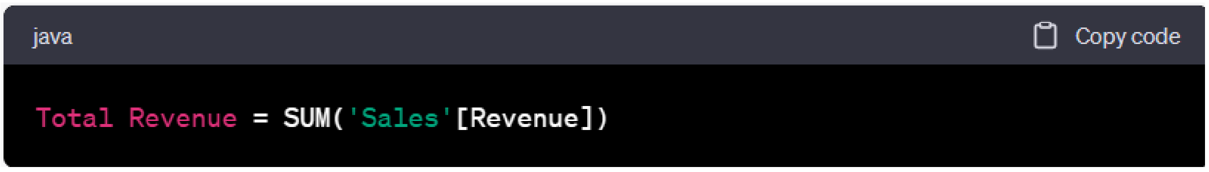
Power Pivot is a data modeling and analysis tool in Microsoft Excel that allows you to work with large datasets and perform complex analysis using formulas and functions. To perform big data analytics with code in Power Pivot, you can follow the steps below:

1. Import Data: First, you need to import the dataset that you want to analyze into Power Pivot. You can do this by clicking on the "Manage Data Model" button in the Power Pivot tab and then selecting "From Other Sources". You can choose to import data from a file, database, or other sources.

2. Data Modeling: Once you have imported the data, you need to create a data model that represents the relationships between the data. You can do this by creating tables, defining relationships between tables, and adding calculated columns using DAX (Data Analysis Expressions) formulas.

3. Data Analysis: After creating the data model, you can use DAX formulas to perform data analysis. DAX formulas can be used to create measures, calculated columns, and calculated tables. Measures are calculations based on data in the model, while calculated columns and tables are created based on DAX formulas.

Here's an example of a DAX formula to create a measure that calculates the total revenue for each product category:



```java
Total Revenue = SUM('Sales'[Revenue])
```

In this example, 'Sales' is the name of the table that contains the sales data, and 'Revenue' is the name of the column that contains the revenue data. The SUM function is used to add up the revenue for each row in the table, resulting in the total revenue for each product category.

4. Visualizations: Once you have created measures and calculated columns, you can use them to create visualizations such as charts, graphs, and tables in Excel. You can use the "Insert" tab to add visualizations to your worksheet.

6. Using Power Pivot (Excel) Perform the following on any dataset
b) Big Data Charting

By using Power Pivot for big data analytics with code, you can analyze large datasets and gain insights into trends and patterns that would be difficult to uncover using traditional Excel tools.

To perform big data charting in Power Pivot, you can follow the steps below:

1. Import Data: First, you need to import the dataset that you want to chart into Power Pivot. You can do this by clicking on the "Manage Data Model" button in the Power Pivot tab and then selecting "From Other Sources". You can choose to import data from a file, database, or other sources.

2. Data Modeling: Once you have imported the data, you need to create a data model that represents the relationships between the data. You can do this by creating tables, defining relationships between tables, and adding calculated columns using DAX formulas.

3. Data Analysis: After creating the data model, you can use DAX formulas to perform data analysis. DAX formulas can be used to create measures, calculated columns, and calculated tables. Measures are calculations based on data in the model, while calculated columns and tables are created based on DAX formulas.

4. Chart Creation: Once you have created measures and calculated columns, you can use them to create charts in Excel. You can use the "Insert" tab to add charts to your worksheet.

Here's an example of a chart that displays the total revenue for each product category:

1. Click on the Insert tab and select the chart type you want to create.
2. Select the data range you want to use for the chart.
3. Click on the Power Pivot Field List button to select the fields you want to use in the chart.
4. Drag the "Product Category" field to the Axis (Categories) section of the chart, and drag the "Total Revenue" measure to the Values section of the chart.
5. Customize the chart as desired using the Chart Tools ribbon.

By using Power Pivot for big data charting, you can create charts that visualize large datasets and highlight trends and patterns that would be difficult to see using traditional Excel tools. You can also customize the charts to your liking and easily update them as new data becomes available.

7. Use R-Project to carry out statistical analysis of big data

To carry out statistical analysis of big data in R, you can follow the steps below:

1. Install Required Packages: First, you need to install and load the required R packages for big data analysis. Some popular packages for big data analysis include dplyr, data.table, and sqldf. You can install these packages using the following command:

less Copy code

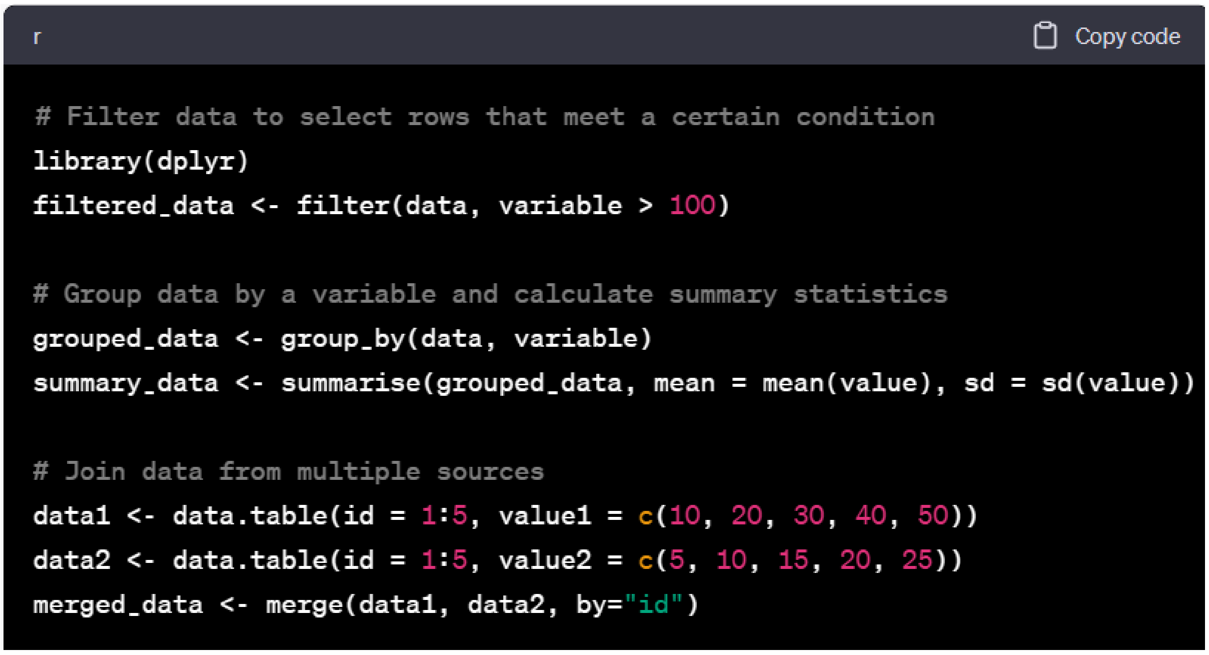
```
install.packages(c("dplyr", "data.table", "sqldf"))
```

2. Import Data: Next, you need to import the big data set into R. You can use the read.csv() function to import data from a CSV file or the read.table() function to import data from a tab-delimited file. If the data is too large to fit in memory, you can use the data.table() function to read the data in chunks.

graphql Copy code

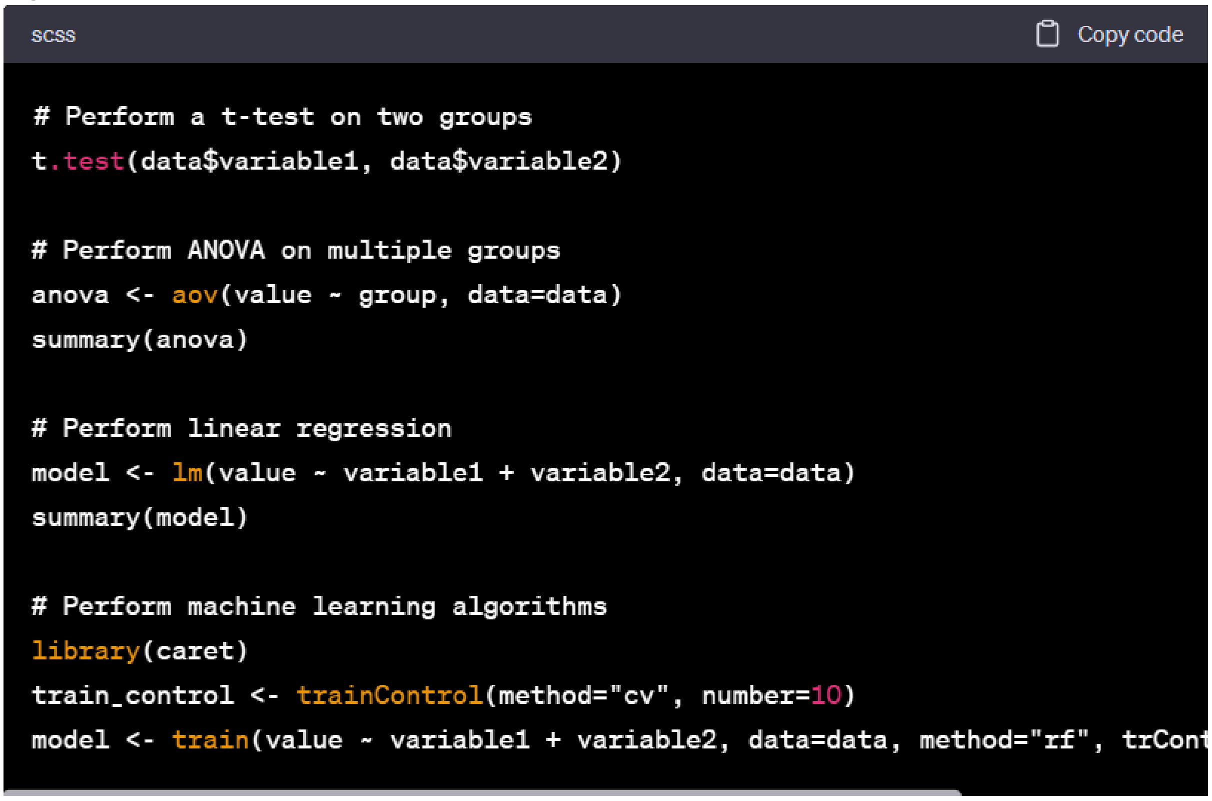
```
# Import data from a CSV file
data <- read.csv("big_data.csv")

# Import data from a tab-delimited file
data <- read.table("big_data.txt", sep="\t", header=TRUE)

# Import data in chunks using data.table
library(data.table)
data <- fread("big_data.csv")
```

3. Data Manipulation: Once you have imported the data, you can use the dplyr package to manipulate the data. The dplyr package provides a set of functions that make it easy to filter, group, and summarize data.



```r
# Filter data to select rows that meet a certain condition
library(dplyr)
filtered_data <- filter(data, variable > 100)

# Group data by a variable and calculate summary statistics
grouped_data <- group_by(data, variable)
summary_data <- summarise(grouped_data, mean = mean(value), sd = sd(value))

# Join data from multiple sources
data1 <- data.table(id = 1:5, value1 = c(10, 20, 30, 40, 50))
data2 <- data.table(id = 1:5, value2 = c(5, 10, 15, 20, 25))
merged_data <- merge(data1, data2, by="id")
```

4. Statistical Analysis: Once you have manipulated the data, you can use R to perform statistical analysis. R provides a wide range of statistical functions that can be used for big data analysis, including t-tests, ANOVA, regression analysis, and machine learning algorithms.



```scss
# Perform a t-test on two groups
t.test(data$variable1, data$variable2)

# Perform ANOVA on multiple groups
anova <- aov(value ~ group, data=data)
summary(anova)

# Perform linear regression
model <- lm(value ~ variable1 + variable2, data=data)
summary(model)

# Perform machine learning algorithms
library(caret)
train_control <- trainControl(method="cv", number=10)
model <- train(value ~ variable1 + variable2, data=data, method="rf", trCont
```

By using R for statistical analysis of big data, you can manipulate and analyze large datasets and gain insights into trends and patterns that would be difficult to uncover using traditional statistical analysis tools.

8. Use R-Project for data visualization of social media data

To visualize social media data in R, you can follow the steps below:

1. Install Required Packages: First, you need to install and load the required R packages for data visualization. Some popular packages for data visualization include ggplot2, plotly, and wordcloud2. You can install these packages using the following command:

less Copy code

```less
install.packages(c("ggplot2", "plotly", "wordcloud2"))
```

2. Import Data: Next, you need to import the social media data into R. You can use the read.csv() function to import data from a CSV file or the read.table() function to import data from a tab-delimited file.

graphql Copy code

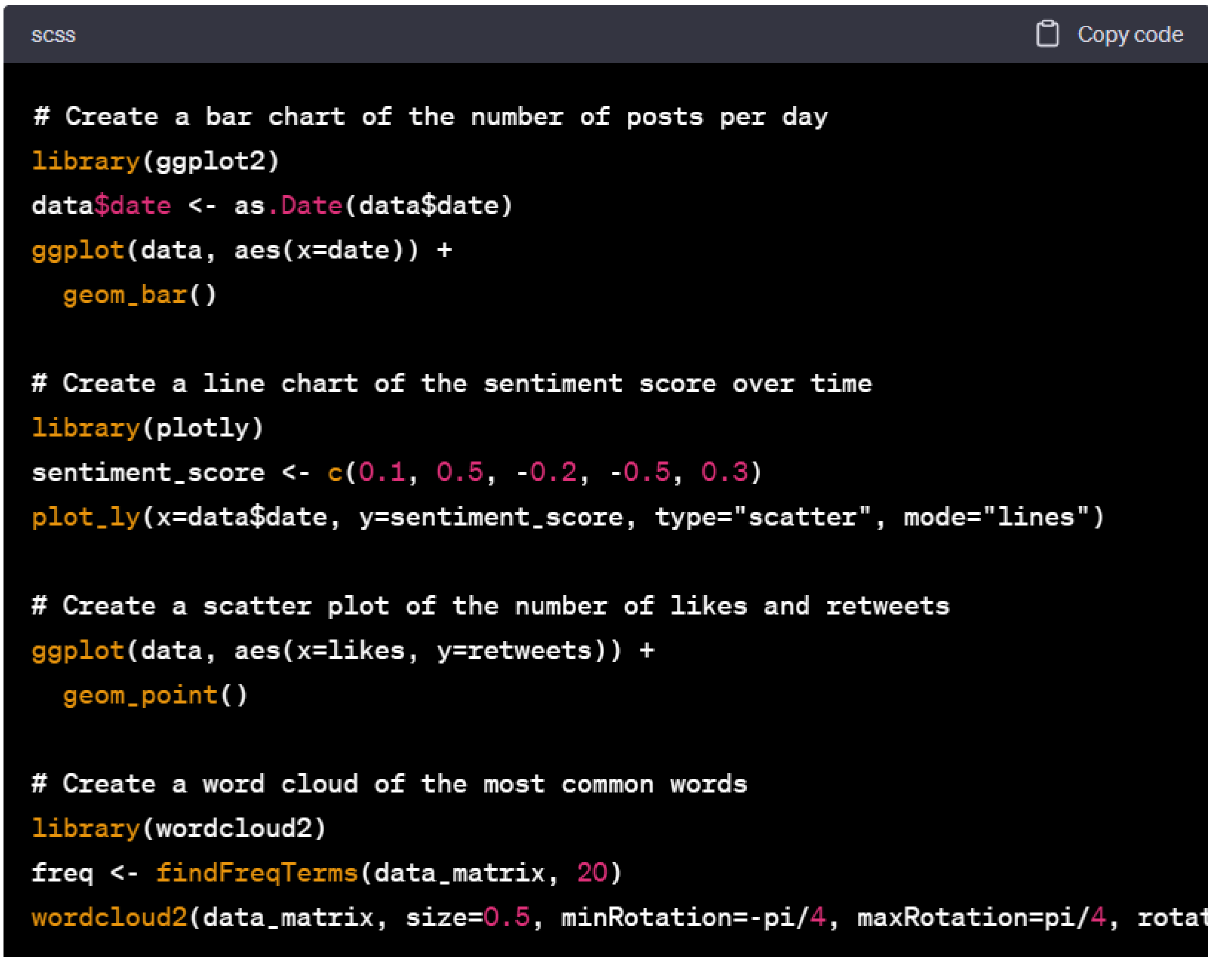
```graphql
# Import data from a CSV file
data <- read.csv("social_media_data.csv")

# Import data from a tab-delimited file
data <- read.table("social_media_data.txt", sep="\t", header=TRUE)
```

3. Data Cleaning: Once you have imported the data, you need to clean and transform it. Social media data often contains unstructured text data, which needs to be processed before it can be visualized. You can use regular expressions and text mining techniques to clean and transform the data.

scss Copy code

```scss
# Clean and transform text data
library(tm)
data_corpus <- VCorpus(VectorSource(data$text))
data_corpus <- tm_map(data_corpus, removePunctuation)
data_corpus <- tm_map(data_corpus, removeNumbers)
data_corpus <- tm_map(data_corpus, content_transformer(tolower))
data_corpus <- tm_map(data_corpus, removeWords, stopwords("english"))
data_matrix <- DocumentTermMatrix(data_corpus)
```

4. Data Visualization: Once you have cleaned and transformed the data, you can use R to create visualizations. R provides a wide range of visualization functions that can be used for social media data, including bar charts, line charts, scatter plots, and word clouds.



```
# Create a bar chart of the number of posts per day
library(ggplot2)
data$date <- as.Date(data$date)
ggplot(data, aes(x=date)) +
  geom_bar()

# Create a line chart of the sentiment score over time
library(plotly)
sentiment_score <- c(0.1, 0.5, -0.2, -0.5, 0.3)
plot_ly(x=data$date, y=sentiment_score, type="scatter", mode="lines")

# Create a scatter plot of the number of likes and retweets
ggplot(data, aes(x=likes, y=retweets)) +
  geom_point()

# Create a word cloud of the most common words
library(wordcloud2)
freq <- findFreqTerms(data_matrix, 20)
wordcloud2(data_matrix, size=0.5, minRotation=-pi/4, maxRotation=pi/4, rotat
```

By using R for data visualization of social media data, you can create visualizations that highlight trends and patterns in the data and gain insights into the sentiment, topics, and engagement levels of social media posts.