BDA Suggested Practical List:

1. **To create Employee data with salary in a single table using R.**

```
# 1 To create Employee data with salary in a single table using R.
# Create a data frame with employee data
employee_data <- data.frame(
  EmployeeID = c(1, 2, 3, 4, 5),
  FirstName = c("John", "Jane", "Bob", "Alice", "Charlie"),
  LastName = c("Doe", "Smith", "Johnson", "Brown", "Wilson"),
  Department = c("HR", "IT", "Sales", "Finance", "Marketing"),
  Salary = c(55000, 60000, 48000, 70000, 62000)
)
# Print the employee data
print(employee_data)
write.csv(employee_data, "employee_data.csv", row.names = FALSE)
```

2. **Define a function in a script, source it to the RStudio and print the multiples of a number.**

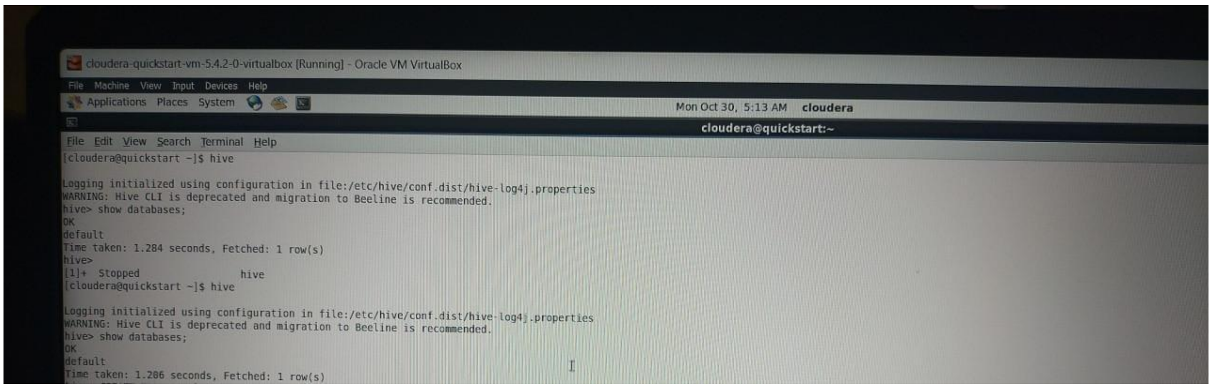
```
# Define a function to print multiples of a number
print_multiples <- function(number, n) {
  multiples <- number * 1:n
  cat("Multiples of", number, "up to", n, ":\n")
  cat(multiples, sep = ", ")
}
# Save this script as "print_multiples.R" in your working directory.
# Source the script in RStudio to make the function available in your current R
session.
# Replace "D:/R_Function.txt" with the actual file path using forward slashes.
source("D:/R_Function.txt")
# Use the function to print multiples.
# For example, print multiples of 4 up to 8.
print_multiples(4, 8)
```

3. **Create two data frames roll no, name, class and other data frame as roll no, subject, marks. Merge this two data frame and print the o/p.**
# 3 Create two data frames roll no, name, class and other data frame as roll no, subject, marks.

```
# Merge this two data frame and print the o/p.
# Create the first data frame with student information
student_df <- data.frame(
  RollNo = c(1, 2, 3, 4, 5),
  Name = c("Alice", "Bob", "Charlie", "David", "Eve"),
  Class = c("A", "B", "A", "C", "B")
)
# Create the second data frame with subject and marks information
marks_df <- data.frame(
  RollNo = c(2, 3, 1, 4, 5),
  Subject = c("Math", "Science", "Math", "History", "English"),
  Marks = c(95, 88, 92, 75, 80)
)
# Merge the two data frames based on the common column "RollNo"
merged_df <- merge(student_df, marks_df, by = "RollNo")
# Print the merged data frame
print(student_df)
print(marks_df)
print(merged_df)
```

4. **Create a table in Hive, display the employees record after loading the data into table.**

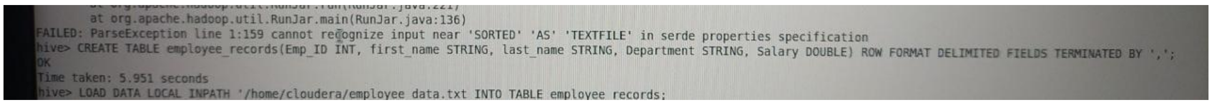
```
hive
show databases;
```



cloudera-quickstart-vm-5.4.2-0-virtualbox [Running] - Oracle VM VirtualBox
File Machine View Input Devices Help
Applications Places System
Mon Oct 30, 5:13 AM cloudera
cloudera@quickstart:~
File Edit View Search Terminal Help

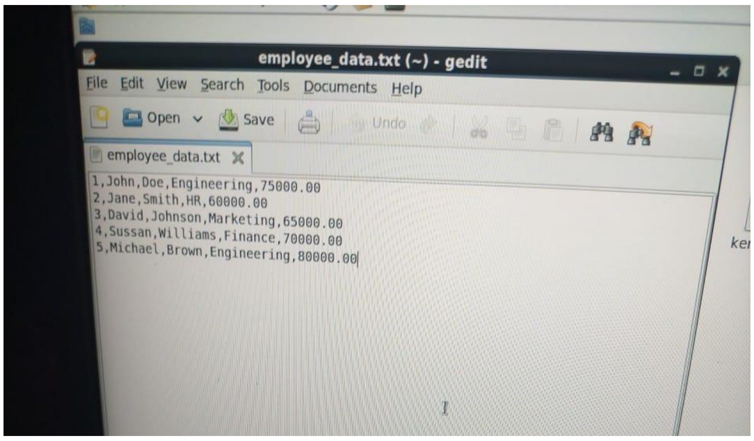
```
[cloudera@quickstart ~]$ hive

Logging initialized using configuration in file:/etc/hive/conf.dist/hive-log4j.properties
WARNING: Hive CLI is deprecated and migration to Beeline is recommended.
hive> show databases;
OK
default
Time taken: 1.284 seconds, Fetched: 1 row(s)
hive>
[1]+  Stopped                 hive
[cloudera@quickstart ~]$ hive

Logging initialized using configuration in file:/etc/hive/conf.dist/hive-log4j.properties
WARNING: Hive CLI is deprecated and migration to Beeline is recommended.
hive> show databases;
OK
default
Time taken: 1.286 seconds, Fetched: 1 row(s)
```

```
CREATE TABLE employee_records (
    employee_id INT,
    first_name STRING,
    last_name STRING,
    department STRING,
    salary DOUBLE
)
ROW FORMAT DELIMITED
FIELDS TERMINATED BY ',';
```



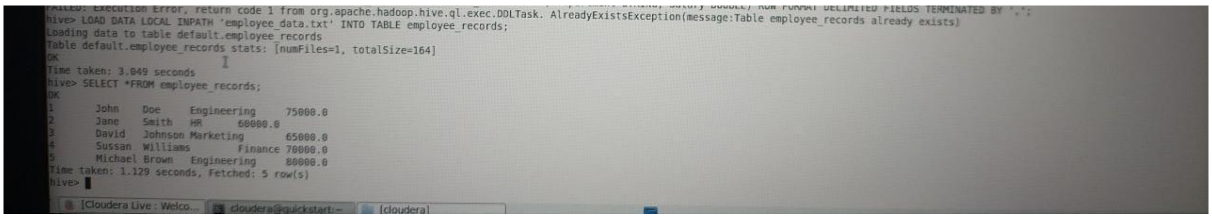
```
        at org.apache.hadoop.util.RunJar.main(RunJar.java:136)
FAILED: ParseException line 1:159 cannot recognize input near 'SORTED' 'AS' 'TEXTFILE' in serde properties specification
hive> CREATE TABLE employee_records(Emp_ID INT, first_name STRING, last_name STRING, Department STRING, Salary DOUBLE) ROW FORMAT DELIMITED FIELDS TERMINATED BY ',';
OK
Time taken: 5.951 seconds
hive> LOAD DATA LOCAL INPATH '/home/cloudera/employee data.txt INTO TABLE employee records;
```

employee_data.txt


employee_data.txt (~) - gedit
File Edit View Search Tools Documents Help
Open
Save
Undo
employee_data.txt

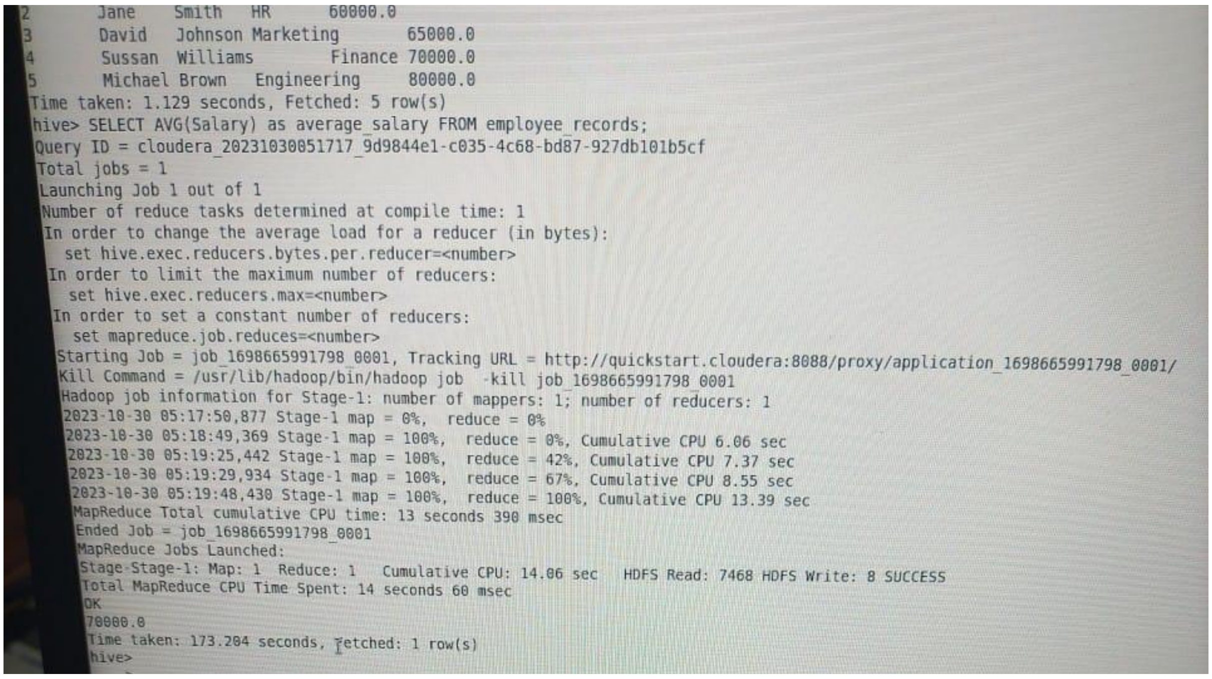
```
1,John,Doe,Engineering,75000.00
2,Jane,Smith,HR,60000.00
3,David,Johnson,Marketing,65000.00
4,Sussan,Williams,Finance,70000.00
5,Michael,Brown,Engineering,80000.00
```

LOAD DATA LOCAL INPATH 'employee_data.txt' INTO TABLE employee_records;
SELECT *FROM employee_records;

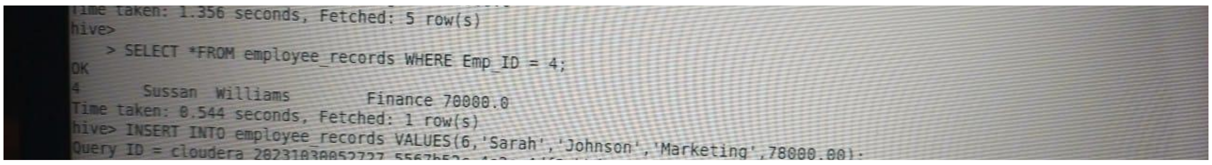


```
FAILED: Execution Error, return code 1 from org.apache.hadoop.hive.ql.exec.DDLTask. AlreadyExistsException(message:Table employee_records already exists)
hive> LOAD DATA LOCAL INPATH 'employee_data.txt' INTO TABLE employee_records;
Loading data to table default.employee_records
Table default.employee_records stats: [numFiles=1, totalSize=164]
OK
Time taken: 3.049 seconds
hive> SELECT *FROM employee_records;
OK
1       John    Doe     Engineering     75000.0
2       Jane    Smith   HR      60000.0
3       David   Johnson Marketing       65000.0
4       Sussan  Williams        Finance 70000.0
5       Michael Brown   Engineering     80000.0
Time taken: 1.129 seconds, Fetched: 5 row(s)
hive>
```

extra query



```
2       Jane    Smith   HR      60000.0
3       David   Johnson Marketing       65000.0
4       Sussan  Williams        Finance 70000.0
5       Michael Brown   Engineering     80000.0
Time taken: 1.129 seconds, Fetched: 5 row(s)
hive> SELECT AVG(Salary) as average_salary FROM employee_records;
Query ID = cloudera_20231030051717_9d9844e1-c035-4c68-bd87-927db101b5cf
Total jobs = 1
Launching Job 1 out of 1
Number of reduce tasks determined at compile time: 1
In order to change the average load for a reducer (in bytes):
  set hive.exec.reducers.bytes.per.reducer=<number>
In order to limit the maximum number of reducers:
  set hive.exec.reducers.max=<number>
In order to set a constant number of reducers:
  set mapreduce.job.reduces=<number>
Starting Job = job_1698665991798_0001, Tracking URL = http://quickstart.cloudera:8088/proxy/application_1698665991798_0001/
Kill Command = /usr/lib/hadoop/bin/hadoop job  -kill job_1698665991798_0001
Hadoop job information for Stage-1: number of mappers: 1; number of reducers: 1
2023-10-30 05:17:50,877 Stage-1 map = 0%,  reduce = 0%
2023-10-30 05:18:49,369 Stage-1 map = 100%,  reduce = 0%, Cumulative CPU 6.06 sec
2023-10-30 05:19:25,442 Stage-1 map = 100%,  reduce = 42%, Cumulative CPU 7.37 sec
2023-10-30 05:19:29,934 Stage-1 map = 100%,  reduce = 67%, Cumulative CPU 8.55 sec
2023-10-30 05:19:48,430 Stage-1 map = 100%,  reduce = 100%, Cumulative CPU 13.39 sec
MapReduce Total cumulative CPU time: 13 seconds 390 msec
Ended Job = job_1698665991798_0001
MapReduce Jobs Launched:
Stage-Stage-1: Map: 1  Reduce: 1   Cumulative CPU: 14.06 sec   HDFS Read: 7468 HDFS Write: 8 SUCCESS
Total MapReduce CPU Time Spent: 14 seconds 60 msec
OK
70000.0
Time taken: 173.204 seconds, Fetched: 1 row(s)
hive>
```



```
Time taken: 1.356 seconds, Fetched: 5 row(s)
hive>
    > SELECT *FROM employee_records WHERE Emp_ID = 4;
OK
4       Sussan  Williams        Finance 70000.0
Time taken: 0.544 seconds, Fetched: 1 row(s)
hive> INSERT INTO employee_records VALUES(6,'Sarah','Johnson','Marketing',78000.00);
Query ID = cloudera_20231030052727_5567b5
```

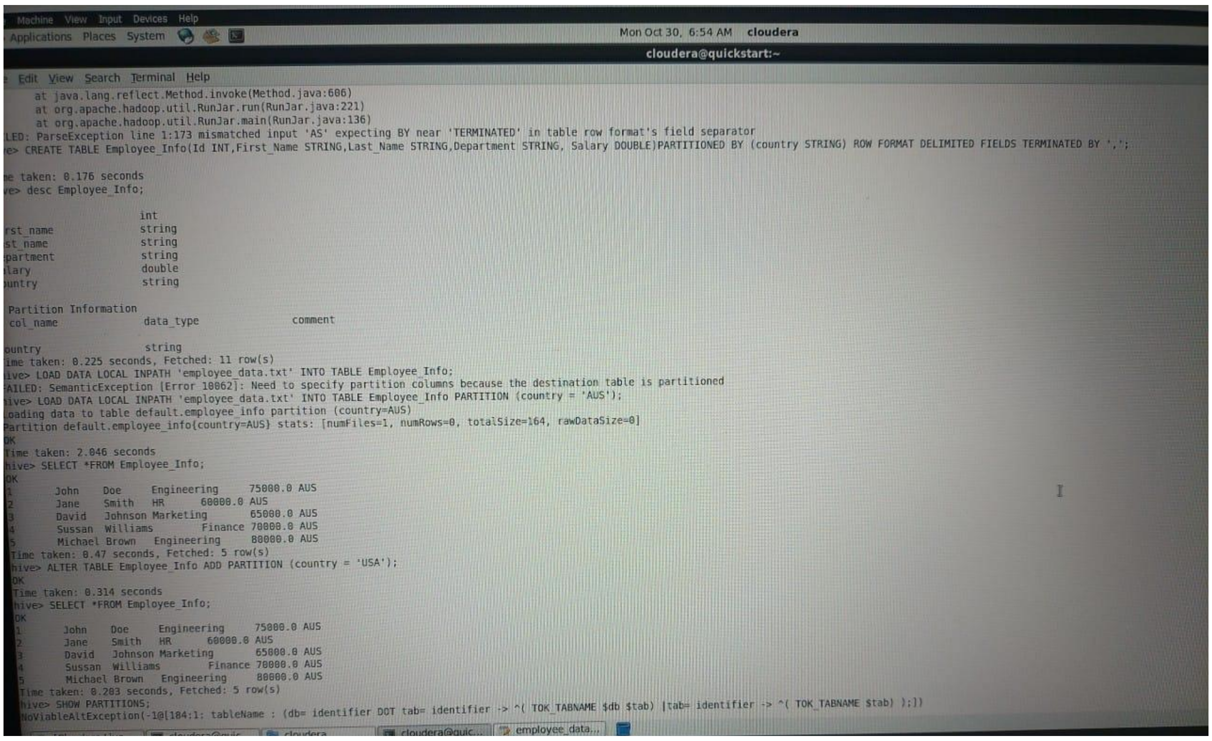
5. Create a table in HIVE and add a partition as country='AUS'.

METHOD 1

```
CREATE TABLE employee_records (
    employee_id INT,
    first_name STRING,
    last_name STRING,
    department STRING,
    salary DOUBLE
)PARTITIONED BY (country STRING)
ROW FORMAT DELIMITED
FIELDS TERMINATED BY ',';


desc employee_records;

LOAD DATA LOCAL INPATH 'employee_data_AUS.txt' OVERWRITE INTO TABLE Employee_Info
PARTITION(country = ‘AUS’);

LOAD DATA LOCAL INPATH 'employee_data_US.txt' OVERWRITE INTO TABLE Employee_Info
PARTITION(country = ‘US’);

SHOW PARTITIONS employee_records;
```

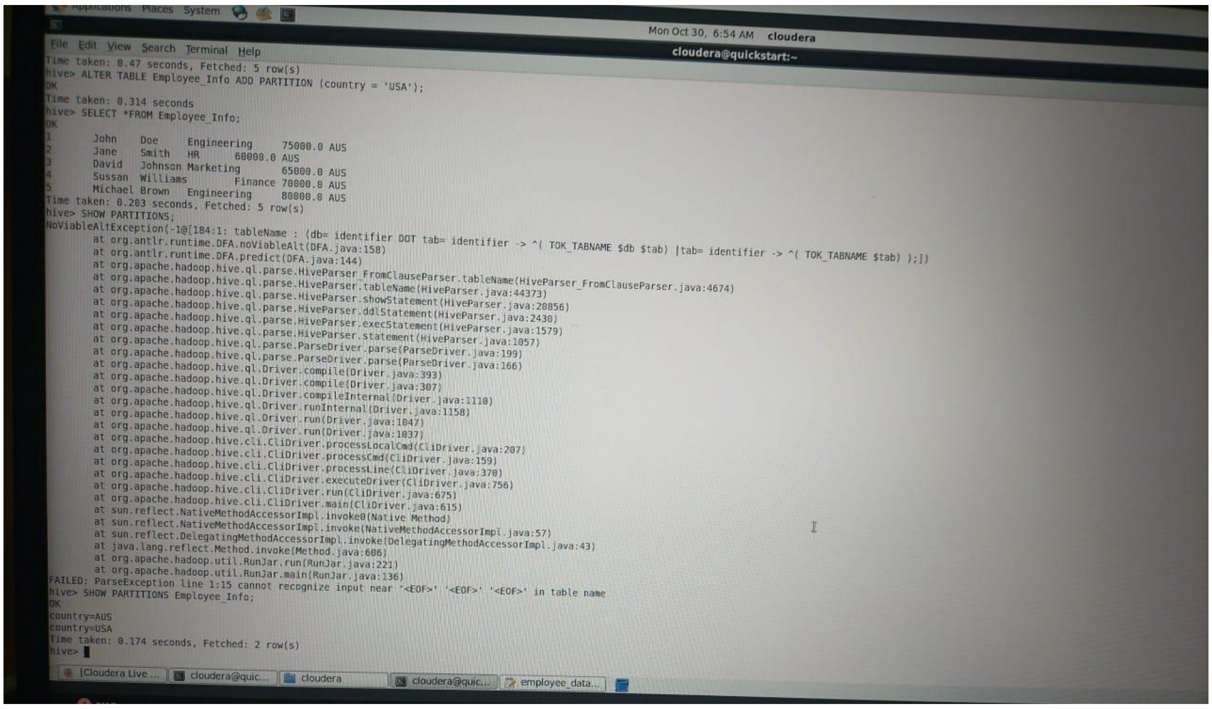
METHOD 2

```
CREATE TABLE employee_records (
    employee_id INT,
    first_name STRING,
    last_name STRING,
    department STRING,
    salary DOUBLE
)PARTITIONED BY (country STRING)
ROW FORMAT DELIMITED
FIELDS TERMINATED BY ',';

desc Employee_Info;


LOAD DATA LOCAL INPATH 'employee_data.txt' INTO TABLE Employee_Info
PARTITION(country = ‘AUS’);

SELECT  *FROM Employee_Info;

ALTER TABLE employee_info ADD PARTITION (country='US');
```



```
    at java.lang.reflect.Method.invoke(Method.java:606)
    at org.apache.hadoop.util.RunJar.run(RunJar.java:221)
    at org.apache.hadoop.util.RunJar.main(RunJar.java:136)
LED: ParseException line 1:173 mismatched input 'AS' expecting BY near 'TERMINATED' in table row format's field separator
e> CREATE TABLE Employee_Info(Id INT,First_Name STRING,Last_Name STRING,Department STRING, Salary DOUBLE)PARTITIONED BY (country STRING) ROW FORMAT DELIMITED FIELDS TERMINATED BY ',';

e taken: 0.176 seconds
ve> desc Employee_Info;

                    int
rst_name            string
st_name             string
partment            string
lary                double
untry               string

 Partition Information
 col_name           data_type           comment

ountry              string
ime taken: 0.225 seconds, Fetched: 11 row(s)
ive> LOAD DATA LOCAL INPATH 'employee_data.txt' INTO TABLE Employee_Info;
AILED: SemanticException [Error 10062]: Need to specify partition columns because the destination table is partitioned
ive> LOAD DATA LOCAL INPATH 'employee_data.txt' INTO TABLE Employee_Info PARTITION (country = 'AUS');
oading data to table default.employee_info partition (country=AUS)
Partition default.employee_info{country=AUS} stats: [numFiles=1, numRows=0, totalSize=164, rawDataSize=0]
OK
Time taken: 2.046 seconds
hive> SELECT *FROM Employee_Info;
OK
1       John    Doe     Engineering     75000.0 AUS
2       Jane    Smith   HR      60000.0 AUS
3       David   Johnson Marketing       65000.0 AUS
4       Sussan  Williams        Finance 70000.0 AUS
5       Michael Brown   Engineering     80000.0 AUS
Time taken: 0.47 seconds, Fetched: 5 row(s)
hive> ALTER TABLE Employee_Info ADD PARTITION (country = 'USA');
OK
Time taken: 0.314 seconds
hive> SELECT *FROM Employee_Info;
OK
1       John    Doe     Engineering     75000.0 AUS
2       Jane    Smith   HR      60000.0 AUS
3       David   Johnson Marketing       65000.0 AUS
4       Sussan  Williams        Finance 70000.0 AUS
5       Michael Brown   Engineering     80000.0 AUS
Time taken: 0.203 seconds, Fetched: 5 row(s)
hive> SHOW PARTITIONS;
NoViableAltException(-1@[184:1: tableName : (db= identifier DOT tab= identifier -> ^( TOK_TABNAME $db $tab) |tab= identifier -> ^( TOK_TABNAME $tab) );])
```

SHOW PARTITIONS employee_info;



```
Time taken: 0.47 seconds, Fetched: 5 row(s)
hive> ALTER TABLE Employee_Info ADD PARTITION (country = 'USA');
OK
Time taken: 0.314 seconds
hive> SELECT *FROM Employee_Info;
OK
1       John    Doe     Engineering     75000.0 AUS
2       Jane    Smith   HR      60000.0 AUS
3       David   Johnson Marketing       65000.0 AUS
4       Sussan  Williams        Finance 70000.0 AUS
5       Michael Brown   Engineering     80000.0 AUS
Time taken: 0.203 seconds, Fetched: 5 row(s)
hive> SHOW PARTITIONS;
NoViableAltException(-1@[184:1: tableName : (db= identifier DOT tab= identifier -> ^( TOK_TABNAME $db $tab) |tab= identifier -> ^( TOK_TABNAME $tab) );])
        at org.antlr.runtime.DFA.noViableAlt(DFA.java:158)
        at org.antlr.runtime.DFA.predict(DFA.java:144)
        at org.apache.hadoop.hive.ql.parse.HiveParser_FromClauseParser.tableName(HiveParser_FromClauseParser.java:4674)
        at org.apache.hadoop.hive.ql.parse.HiveParser.tableName(HiveParser.java:44373)
        at org.apache.hadoop.hive.ql.parse.HiveParser.showStatement(HiveParser.java:20856)
        at org.apache.hadoop.hive.ql.parse.HiveParser.ddlStatement(HiveParser.java:2430)
        at org.apache.hadoop.hive.ql.parse.HiveParser.execStatement(HiveParser.java:1579)
        at org.apache.hadoop.hive.ql.parse.HiveParser.statement(HiveParser.java:1057)
        at org.apache.hadoop.hive.ql.parse.ParseDriver.parse(ParseDriver.java:199)
        at org.apache.hadoop.hive.ql.parse.ParseDriver.parse(ParseDriver.java:166)
        at org.apache.hadoop.hive.ql.Driver.compile(Driver.java:393)
        at org.apache.hadoop.hive.ql.Driver.compile(Driver.java:307)
        at org.apache.hadoop.hive.ql.Driver.compileInternal(Driver.java:1110)
        at org.apache.hadoop.hive.ql.Driver.runInternal(Driver.java:1158)
        at org.apache.hadoop.hive.ql.Driver.run(Driver.java:1047)
        at org.apache.hadoop.hive.ql.Driver.run(Driver.java:1037)
        at org.apache.hadoop.hive.cli.CliDriver.processLocalCmd(CliDriver.java:207)
        at org.apache.hadoop.hive.cli.CliDriver.processCmd(CliDriver.java:159)
        at org.apache.hadoop.hive.cli.CliDriver.processLine(CliDriver.java:370)
        at org.apache.hadoop.hive.cli.CliDriver.executeDriver(CliDriver.java:756)
        at org.apache.hadoop.hive.cli.CliDriver.run(CliDriver.java:675)
        at org.apache.hadoop.hive.cli.CliDriver.main(CliDriver.java:615)
        at sun.reflect.NativeMethodAccessorImpl.invoke0(Native Method)
        at sun.reflect.NativeMethodAccessorImpl.invoke(NativeMethodAccessorImpl.java:57)
        at sun.reflect.DelegatingMethodAccessorImpl.invoke(DelegatingMethodAccessorImpl.java:43)
        at java.lang.reflect.Method.invoke(Method.java:606)
        at org.apache.hadoop.util.RunJar.run(RunJar.java:221)
        at org.apache.hadoop.util.RunJar.main(RunJar.java:136)
FAILED: ParseException line 1:15 cannot recognize input near '<EOF>' '<EOF>' '<EOF>' in table name
hive> SHOW PARTITIONS Employee_Info;
OK
country=AUS
country=USA
Time taken: 0.174 seconds, Fetched: 2 row(s)
hive>
```

6. Hadoop Basic commands –

Hdfs dfs -ls /
Hdfs dfs -mkdir /user/cloudera/swarali
Hdfs dfs -touchz /home/cloudera/sample.txt
Hdfs dfs -mkdir /user/cloudera/swarali/

```
 Hdfs dfs -copyFromLocal  /home/cloudera/a1.txt  /user/cloudera/swarali
 Hdfs dfs -copyToLocal   /home/cloudera/swarali/sample.txt  /home/cloudera
 Hdfs dfs -cat  /user/cloudera/swarali/a1.txt
Hdfs dfs -rm /user/cloudera/swarali/a1.txt
 Hdfs dfs -rmdir  /user/cloudera/swarali
 Hdfs dfs -cp /user/cloudera/a1.txt  /user/cloudera/swarali/
 Hdfs dfs -mv /user/cloudera/a1.txt  /user/cloudera/swarali/
 Hdfs dfs -chmod 750 /user/cloudera/swarali/a1.txt
 Hdfs dfs -chown swarali  /user/cloudera/swarali/a1.txt
 Hdfs dfs -chgrp swarali  /user/cloudera/swarali/a1.txt
 Hdfs dfs -getmerge /user/cloudera/swarali  /home/cloudera/merged.txt
 Hdfs dfs -put /home/cloudera/merged.txt  /user/cloudera/swarali1
  Hdfs dfs -get  /user/cloudera/sample.txt  /home/cloudera
 Hdfs dfs -du /user/cloudera/swarali/sample.txt
 Hdfs dfsadmin -safemode enter
 Hdfs dfsadmin -safemode leave
 Hdfs dfsadmin -refreshNodes
```

7. **Create and insert a post in NOSQL database for a social media website using MongoDB.**

```
use your_database_name


db.createCollection("your_collection_name")


db.your_collection_name.insertOne({
  title: "My First Post",
  content: "This is the content of my first post.",
  author: "John Doe",
  timestamp: new Date()
})
```

```
db.your_collection_name.find()
```

8. Create and update a post in NOSQL database for a social media website using MongoDB

```
use your_database_name


db.createCollection("your_collection_name")


db.your_collection_name.insertOne({
  title: "My First Post",
  content: "This is the content of my first post.",
  author: "John Doe",
  timestamp: new Date()
})


db.your_collection_name.find().pretty()

db.your_collection_name.updateOne(
  { title: "My First Post" },
  { $set: { content: "Updated content" } }
)

db.your_collection_name.find().pretty()
```

9. Implement FM algorithm

```python
import time
#BE3_22_SwaraliJanaskar_Expt-6 import time
def flajolet_martin(stream):
   maxnum = 0
   for i in range(0, len(stream)):
       val = bin((1 * stream[i] + 6) % 32)[2:]
       sum = 0
       for j in range(len(val) - 1, 0, -1):
           if val[j] == '0':
               sum += 1
           else:
               break
       if sum > maxnum:
           maxnum = sum
   return 2 ** maxnum
if __name__ == "__main__":
   stream = [1, 2, 3, 4, 5, 6, 4, 2, 5, 9, 1, 6, 3, 7, 1, 2, 2, 4, 2,
1]
   print('Using Flajolet Martin Algorithm:')
   start_time = time.time()
```

```
distinct_elements = flajolet_martin(stream)
print('Distinct elements:', distinct_elements)
print('Execution time:', time.time() - start_time, 'seconds')
```

The code you've provided is implementing the Flajolet-Martin algorithm to estimate the number of distinct elements in a stream of data. The algorithm is a probabilistic algorithm used for cardinality estimation and is particularly useful when you have a large dataset and you want to estimate the number of unique elements without storing all of them in memory.

Here's how the Flajolet-Martin algorithm works:

- ➔ It processes the elements in the stream one by one.
- ➔ For each element in the stream, it computes a binary representation of the element. In this code, the binary representation is computed as `bin((1 * stream[i] + 6) % 32)[2:]`.
- ➔ It counts the number of trailing zeros in the binary representation (i.e., the number of consecutive '0' bits from the right).
- ➔ It keeps track of the maximum count of trailing zeros encountered in the binary representations of the elements.
- ➔ The algorithm returns an estimate of the number of distinct elements as 2 raised to the power of the maximum count of trailing zeros.

Certainly, let's break down the code step by step:

```python
def flajolet_martin(stream):
    maxnum = 0  # Initialize the maximum number of trailing zeros to 0
    for i in range(0, len(stream)):
        val = bin((1 * stream[i] + 6) % 32)[2:]
```

1. `maxnum` is initialized to 0, which will keep track of the maximum count of trailing zeros in the binary representations of elements in the stream.

2. The code iterates through the `stream`, which is a list of numbers.

3. For each element in the stream, it computes a binary representation. It does this by taking the element, multiplying it by 1, adding 6, and taking the result modulo 32. This ensures that the number is in the range [0, 31]. Then, `bin()` is used to obtain the binary representation of the number, and `[2:]` is used to remove the '0b' prefix from the binary string.

```python
        sum = 0
        for j in range(len(val) - 1, 0, -1):
            if val[j] == '0':
                sum += 1
            else:
                break
```

~~~
```
~~~

4. For each binary representation, the code initializes `sum` to 0 and starts iterating through the binary string from right to left (from the least significant bit to the most significant bit).

5. It counts the number of trailing zeros (consecutive '0' bits) in the binary representation. If it encounters a '1', the loop breaks because it's no longer counting trailing zeros.

~~~
```python
    if sum > maxnum:
        maxnum = sum
```
~~~

6. It compares the `sum` (the count of trailing zeros) with the current maximum count (`maxnum`). If the new `sum` is greater than the current `maxnum`, it updates `maxnum` with the new value.

~~~
```python
  return 2 ** maxnum
```
~~~

7. After processing all elements in the stream, the function returns an estimate of the number of distinct elements. It does this by taking 2 raised to the power of `maxnum`. The idea is that the maximum count of trailing zeros in the binary representations can be used to estimate the number of distinct elements probabilistically.

8. In the main part of the code, a sample `stream` is provided, and the Flajolet-Martin algorithm is applied to estimate the number of distinct elements in the stream. The execution time is also measured and printed.

The Flajolet-Martin algorithm is a probabilistic algorithm and provides an approximate estimate of the number of distinct elements in a large dataset, which can be useful in various applications like estimating unique website visitors or counting unique items in a dataset without having to store all the unique items explicitly. The accuracy of the estimate depends on the characteristics of the data and the chosen hash functions.

10. Implement Blooms Filter and verify the contents are present or not.

```python
import mmh3
from bitarray import bitarray

class BloomFilter:
   def __init__(self, size, hash_functions):
       self.size = size
       self.bit_array = bitarray(size)
       self.bit_array.setall(0)
       self.hash_functions = hash_functions

   def add(self, element):
       for i in range(self.hash_functions):
           index = mmh3.hash(element, i) % self.size
           self.bit_array[index] = 1
```

```python
    def contains(self, element):
        for i in range(self.hash_functions):
            index = mmh3.hash(element, i) % self.size
            if not self.bit_array[index]:
                return False
        return True

if __name__ == "__main__":
    size = 1000
    hash_functions = 5
    bloom_filter = BloomFilter(size, hash_functions)

    # Add elements to the Bloom Filter
    elements_to_add = ["apple", "banana", "cherry", "date",
"elderberry"]
    for element in elements_to_add:
        bloom_filter.add(element)

    # Verify if elements are present
    elements_to_verify = ["apple", "banana", "grape", "fig"]
    for element in elements_to_verify:
        if bloom_filter.contains(element):
            print(f"'{element}' may be in the set.")
        else:
            print(f"'{element}' is definitely not in the set.")
```
Certainly, let's go through each line of the provided code and explain its purpose:

```python
import mmh3
from bitarray import bitarray
```
- These lines import the necessary libraries. `mmh3` is used for hash functions, and `bitarray` is used for working with a bit array.

```python
class BloomFilter:
    def __init__(self, size, hash_functions):
        self.size = size
        self.bit_array = bitarray(size)
        self.bit_array.setall(0)
        self.hash_functions = hash_functions
```
- This code defines a class named `BloomFilter`. The class has an `__init__` method that initializes a new instance of the BloomFilter with the specified size and number of hash functions.
- `self.size` stores the size of the bit array, which is passed as a parameter to the constructor.
- `self.bit_array` is a bit array of the specified size created using the `bitarray` library. It's initialized with all bits set to 0 using `setall(0)`.
- `self.hash_functions` stores the number of hash functions to be used.

```python
    def add(self, element):
        for i in range(self.hash_functions):
            index = mmh3.hash(element, i) % self.size
            self.bit_array[index] = 1
```
- The `add` method is used to add an element to the Bloom Filter.
- It iterates through each of the specified hash functions (from 0 to `hash_functions - 1`).
- For each hash function, it calculates an index within the bit array by taking the result of `mmh3.hash(element, i)` modulo the size of the bit array (`self.size`).
- It then sets the corresponding bit at that index to 1, indicating that the element is present in the set.

```python
    def contains(self, element):
        for i in range(self.hash_functions):
            index = mmh3.hash(element, i) % self.size
            if not self.bit_array[index]:
                return False
        return True
```
- The `contains` method is used to check if an element may be in the set (with a possibility of false positives).
- It also iterates through each of the specified hash functions.
- For each hash function, it calculates the index within the bit array as before.
- If any of the bits at the calculated indexes are not set (equal to 0), it returns `False` because the element is definitely not in the set.
- If all bits at the calculated indexes are set (equal to 1), it returns `True`, indicating that the element may be in the set.

```python
if __name__ == "__main__":
    size = 1000
    hash_functions = 5
    bloom_filter = BloomFilter(size, hash_functions)

    # Add elements to the Bloom Filter
    elements_to_add = ["apple", "banana", "cherry", "date", "elderberry"]
    for element in elements_to_add:
        bloom_filter.add(element)

    # Verify if elements are present
    elements_to_verify = ["apple", "banana", "grape", "fig"]
    for element in elements_to_verify:
        if bloom_filter.contains(element):
            print(f"'{element}' may be in the set.")
        else:
            print(f"'{element}' is definitely not in the set.")
```

- This section is the main block of the program.
- It initializes a Bloom Filter instance `bloom_filter` with a bit array of size 1000 and 5 hash functions.
- It adds several elements to the Bloom Filter using the `add` method.
- It then verifies if certain elements are present in the Bloom Filter using the `contains` method. For each element, it prints whether the element may be in the set or is definitely not in the set based on the results of the Bloom Filter.

Overall, this code demonstrates the use of a Bloom Filter to test for potential set membership with possible false positives.

11. Implement Word count using map reduce.

Step 1 : New->Java project - > Project_name
Add external libraries
Libraies->Add External jar files -> file system -> user ->select all jar files-> finish
Step 2 : Right click on src-> new-> class->class_name(WordCount)->next
Step 3 : copy code from the above link and save the file
https://hadoop.apache.org/docs/current/hadoop-mapreduce-client/hadoop-mapreduce-client-core/MapReduceTutorial.html
Code:

```
import java.io.IOException;
import java.util.StringTokenizer;

import org.apache.hadoop.conf.Configuration;
import org.apache.hadoop.fs.Path;
import org.apache.hadoop.io.IntWritable;
import org.apache.hadoop.io.Text;
import org.apache.hadoop.mapreduce.Job;
import org.apache.hadoop.mapreduce.Mapper;
import org.apache.hadoop.mapreduce.Reducer;
import org.apache.hadoop.mapreduce.lib.input.FileInputFormat;
import org.apache.hadoop.mapreduce.lib.output.FileOutputFormat;

public class WordCount {

  public static class TokenizerMapper
       extends Mapper<Object, Text, Text, IntWritable>{

    private final static IntWritable one = new IntWritable(1);
    private Text word = new Text();

    public void map(Object key, Text value, Context context
                    ) throws IOException, InterruptedException {
```

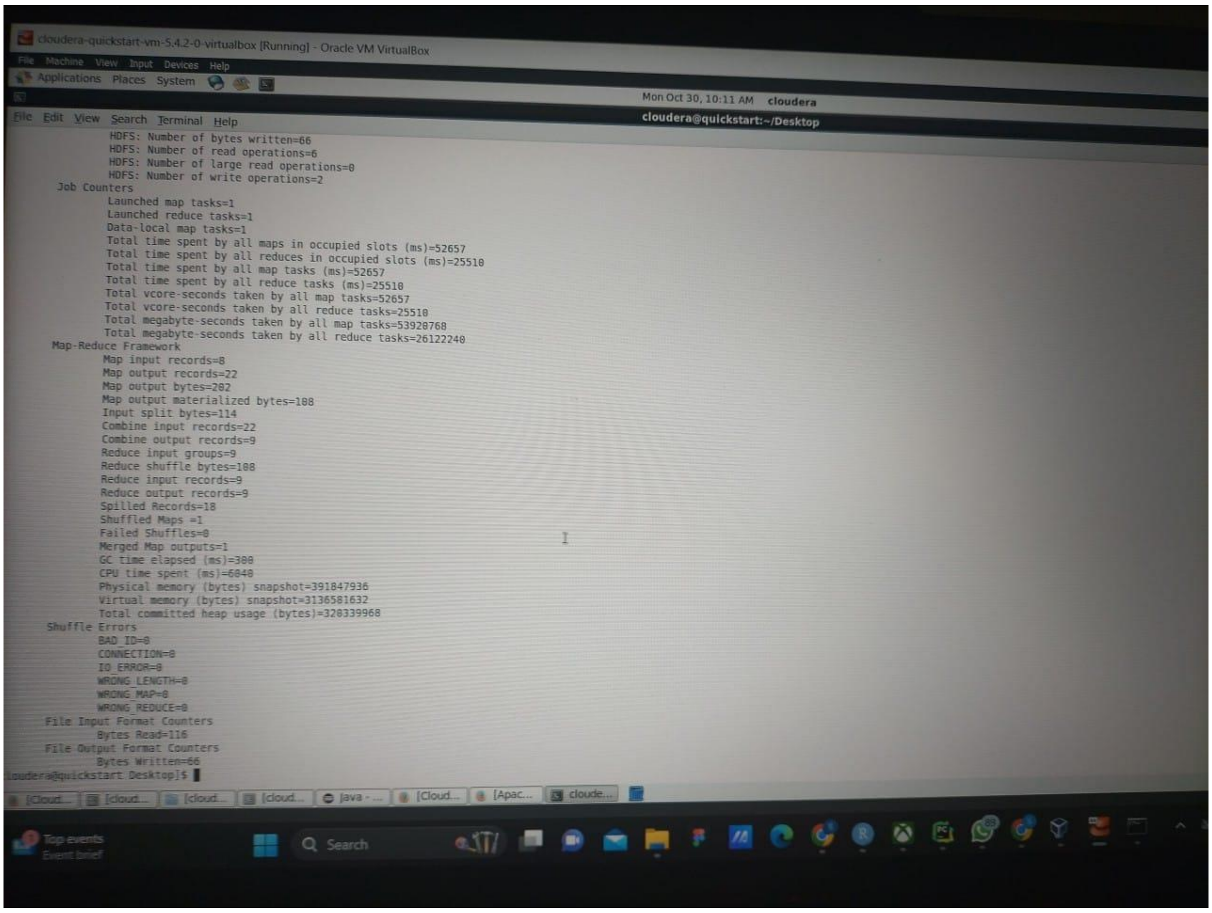
```
      StringTokenizer itr = new StringTokenizer(value.toString());
      while (itr.hasMoreTokens()) {
        word.set(itr.nextToken());
        context.write(word, one);
      }
    }
  }

  public static class IntSumReducer
       extends Reducer<Text,IntWritable,Text,IntWritable> {
    private IntWritable result = new IntWritable();

    public void reduce(Text key, Iterable<IntWritable> values,
                       Context context
                       ) throws IOException, InterruptedException {
      int sum = 0;
      for (IntWritable val : values) {
        sum += val.get();
      }
      result.set(sum);
      context.write(key, result);
    }
  }

  public static void main(String[] args) throws Exception {
    Configuration conf = new Configuration();
    Job job = Job.getInstance(conf, "word count");
    job.setJarByClass(WordCount.class);
    job.setMapperClass(TokenizerMapper.class);
    job.setCombinerClass(IntSumReducer.class);
    job.setReducerClass(IntSumReducer.class);
    job.setOutputKeyClass(Text.class);
    job.setOutputValueClass(IntWritable.class);
    FileInputFormat.addInputPath(job, new Path(args[0]));
    FileOutputFormat.setOutputPath(job, new Path(args[1]));
    System.exit(job.waitForCompletion(true) ? 0 : 1);
  }
}
```

Step 4 : click on export->Java->jar file->next->browse->save jar file as WordCount.jar->location choose desktop->next->finish

Step 5 : Create Sample.txt and save it on desktop

Step 6 : Create directory

```
        hadoop dfs -mkdir /WordIn
         cd Desktop/
        hadoop dfs -copyFromLocal Sample.txt /WordIn
        hadoop jar WordCount.jar WordCount /WordIn /WordOut
```



```
                HDFS: Number of bytes written=66
                HDFS: Number of read operations=6
                HDFS: Number of large read operations=0
                HDFS: Number of write operations=2
        Job Counters
                Launched map tasks=1
                Launched reduce tasks=1
                Data-local map tasks=1
                Total time spent by all maps in occupied slots (ms)=52657
                Total time spent by all reduces in occupied slots (ms)=25510
                Total time spent by all map tasks (ms)=52657
                Total time spent by all reduce tasks (ms)=25510
                Total vcore-seconds taken by all map tasks=52657
                Total vcore-seconds taken by all reduce tasks=25510
                Total megabyte-seconds taken by all map tasks=53920768
                Total megabyte-seconds taken by all reduce tasks=26122240
        Map-Reduce Framework
                Map input records=8
                Map output records=22
                Map output bytes=202
                Map output materialized bytes=188
                Input split bytes=114
                Combine input records=22
                Combine output records=9
                Reduce input groups=9
                Reduce shuffle bytes=188
                Reduce input records=9
                Reduce output records=9
                Spilled Records=18
                Shuffled Maps =1
                Failed Shuffles=0
                Merged Map outputs=1
                GC time elapsed (ms)=388
                CPU time spent (ms)=6040
                Physical memory (bytes) snapshot=391847936
                Virtual memory (bytes) snapshot=3136581632
                Total committed heap usage (bytes)=320339968
        Shuffle Errors
                BAD_ID=0
                CONNECTION=0
                IO_ERROR=0
                WRONG_LENGTH=0
                WRONG_MAP=0
                WRONG_REDUCE=0
        File Input Format Counters
                Bytes Read=116
        File Output Format Counters
                Bytes Written=66
loudera@quickstart Desktop]$
```

12. Write a R program to create a Data frames which contain details of 5 employees and display the details.

```r
# 12 Write a R program to create a Data frames
# which contain details of 5 employees and display the details.
# Create a data frame with employee details
employee_df <- data.frame(
  EmployeeID = c(1, 2, 3, 4, 5),
  FirstName = c("John", "Jane", "Bob", "Alice", "Charlie"),
  LastName = c("Doe", "Smith", "Johnson", "Brown", "Wilson"),
  Department = c("HR", "IT", "Sales", "Finance", "Marketing"),
  Salary = c(55000, 60000, 48000, 70000, 62000)
)
# Display the employee details
print(employee_df)
```

13. Write a R program to create a list of heterogeneous data, which include character, numeric and logical vectors. Print the lists.

```r
# 13 Write a R program to create a list of heterogeneous data,
# which include character, numeric and logical vectors. Print the lists.
# Create a list of heterogeneous data
my_list <- list(
  character_vector = c("Alice", "Bob", "Charlie"),
  numeric_vector = c(25, 30, 35),
  logical_vector = c(TRUE, FALSE, TRUE)
)
```

```
# Print the list
print(my_list)
```

14. Write a R program to create bell curve of a random normal distribution.

```
# 14 Write a R program to create bell curve of a random normal distribution.
# Set the seed for reproducibility (optional)
set.seed(123)
# Generate random data from a normal distribution
data <- rnorm(1000, mean = 0, sd = 1)
# Create a histogram to visualize the distribution
hist(data, main = "Random Normal Distribution", xlab = "Value", ylab = "Frequency", col =
"lightblue", border = "black", breaks = 20)
# when data is given
# Create a vector with your existing data
your_data <- c(0.5, 1.2, 1.8, 2.4, 2.9, 3.2, 3.5, 4.0, 4.5, 5.0, 5.5, 5.8, 6.2, 6.5, 6.8)
# Create a histogram to visualize the data
hist(your_data, main = "Your Data Distribution", xlab = "Value", ylab = "Frequency", col =
"lightblue", border = "black", breaks = 10)
```

15. Write a R program to create a simple bar plot of five subjects marks.

```
# 15 Write a R program to create a simple bar plot of five subjects marks.
# Create a vector of subject names
subjects <- c("Math", "Science", "English", "History", "Art")
# Create a vector of corresponding marks
marks <- c(90, 78, 85, 92, 70)
# Create a bar plot
barplot(marks, names.arg = subjects, col = "skyblue", main = "Subject Marks", xlab =
"Subjects", ylab = "Marks")
```

16. Write a R program to create a 5 × 4 matrix , 3 × 3 matrix with labels and fill the matrix by rows and 2 × 2 matrix with labels and fill the matrix by columns.

```
# 16 Write a R program to create a 5 × 4 matrix , 3 × 3 matrix with labels and fill the matrix by

rows

# and 2 × 2 matrix with labels and fill the matrix by columns.

# Create a 5x4 matrix with labels and fill it by rows

matrix1 <- matrix(1:20, nrow = 5, ncol = 4, byrow = TRUE, dimnames = list(c("Row1",

"Row2", "Row3", "Row4", "Row5"), c("Col1", "Col2", "Col3", "Col4")))

# Create a 3x3 matrix with labels and fill it by rows

matrix2 <- matrix(21:29, nrow = 3, ncol = 3, byrow = TRUE, dimnames = list(c("RowA",

"RowB", "RowC"), c("ColX", "ColY", "ColZ")))
```

```
# Create a 2x2 matrix with labels and fill it by columns
matrix3 <- matrix(30:33, nrow = 2, ncol = 2, byrow = FALSE, dimnames = list(c("RowI",
"RowII"), c("ColM", "ColN")))
# Print the matrices
print(matrix1)
print(matrix2)
print(matrix3)
```

17. Write a R program to create a sequence of numbers from 20 to 50 and find the mean of numbers from 20 to 60 and sum of numbers from 51 to 91.

```
# 17 Write a R program to create a sequence of numbers from 20 to 50
# and find the mean of numbers from 20 to 60 and sum of numbers from 51 to 91.
# Create a sequence of numbers from 20 to 50
numbers_seq <- seq(20, 50)
# Calculate the mean of numbers from 20 to 60
mean_20_to_60 <- mean(seq(20, 60))
# Calculate the sum of numbers from 51 to 91
sum_51_to_91 <- sum(seq(51, 91))
# Print the results
cat("Sequence from 20 to 50:", numbers_seq, "\n")
cat("Mean of numbers from 20 to 60:", mean_20_to_60, "\n")
cat("Sum of numbers from 51 to 91:", sum_51_to_91, "\n")
```