**Hive DDL:**

**Task 1:**

-- i. Create a database named Employee\_DB

CREATE DATABASE IF NOT EXISTS Employee\_DB;

-- ii. Create a table employee with columns (empid, ename, designation, department) partitioned by department

CREATE TABLE IF NOT EXISTS Employee\_DB.employee (

empid INT,

emp\_name STRING,

designation STRING,

department STRING

)

PARTITIONED BY (department STRING);

-- iii. Show the structure of employee table

DESCRIBE Employee\_DB.employee;

-- iv. Modify the name of column ename to emp\_name

ALTER TABLE Employee\_DB.employee CHANGE COLUMN ename emp\_name STRING;

-- v. Add the column salary

ALTER TABLE Employee\_DB.employee ADD COLUMNS (salary FLOAT);

-- vi. Rename table to employee\_new

ALTER TABLE Employee\_DB.employee RENAME TO Employee\_DB.employee\_new;

-- vii. Delete a table employee\_new

DROP TABLE IF EXISTS Employee\_DB.employee\_new;

Task 2:

-- i. Create a database named Student\_DB

CREATE DATABASE IF NOT EXISTS Student\_DB;

-- ii. Create a table student with columns (class, division, roll\_no, sname) partitioned by class

CREATE TABLE IF NOT EXISTS Student\_DB.student (

class STRING,

division STRING,

roll\_no INT,

student\_name STRING

)

PARTITIONED BY (class STRING);

-- iii. Show the structure of student table

DESCRIBE Student\_DB.student;

-- iv. Modify the name of column sname to student\_name

ALTER TABLE Student\_DB.student CHANGE COLUMN sname student\_name STRING;

-- v. Add the column percentage

ALTER TABLE Student\_DB.student ADD COLUMNS (percentage FLOAT);

-- vi. Rename table to student\_new

ALTER TABLE Student\_DB.student RENAME TO Student\_DB.student\_new;

-- vii. Delete a table student\_new

DROP TABLE IF EXISTS Student\_DB.student\_new;

```

**Hive DML:**

**Task 3:**

-- i. Create a database named Employee\_DB

CREATE DATABASE IF NOT EXISTS Employee\_DB;

-- ii. Create a table employee with columns (empid, ename, designation, department, salary)

CREATE TABLE IF NOT EXISTS Employee\_DB.employee (

empid INT,

ename STRING,

designation STRING,

department STRING,

salary FLOAT

);

-- iii. Insert the five records into employee table.

INSERT INTO Employee\_DB.employee VALUES

(101, 'John', 'Manager', 'HR', 50000),

(102, 'Alice', 'Developer', 'IT', 60000),

(103, 'Bob', 'Analyst', 'Finance', 45000),

(104, 'Emma', 'Designer', 'Marketing', 55000),

(105, 'Mike', 'Manager', 'Operations', 48000);

-- iv. Update the salary of employee to 40000 whose empid is 101

UPDATE Employee\_DB.employee SET salary = 40000 WHERE empid = 101;

-- v. Delete the record of employee whose empid is 101

DELETE FROM Employee\_DB.employee WHERE empid = 101;

**Task 4:**

-- i. Create a database named Student\_DB

CREATE DATABASE IF NOT EXISTS Student\_DB;

-- ii. Create a table student with columns (class, division, roll\_no, sname, percentage) partitioned by class

CREATE TABLE IF NOT EXISTS Student\_DB.student (

class STRING,

division STRING,

roll\_no INT,

student\_name STRING,

percentage FLOAT

)

PARTITIONED BY (class STRING);

-- iii. Insert the five records into student table.

INSERT INTO Student\_DB.student VALUES

('10th', 'A', 1, 'John', 85.5),

('12th', 'B', 2, 'Alice', 78.2),

('11th', 'C', 3, 'Bob', 67.8),

('10th', 'B', 4, 'Emma', 92.1),

('12th', 'A', 5, 'Mike', 75.9);

-- vi. Update the percentage of student to 80 whose roll\_no is 4

UPDATE Student\_DB.student SET percentage = 80 WHERE roll\_no = 4;

-- vii. Delete the record of employee whose roll\_no is 4

DELETE FROM Student\_DB.student WHERE roll\_no = 4;

```

**Task 5:**

-- i. Create a database named Employee\_DB

CREATE DATABASE IF NOT EXISTS Employee\_DB;

-- ii. Create a table employee with columns (empid, ename, designation, department, salary)

CREATE TABLE IF NOT EXISTS Employee\_DB.employee (

empid INT,

ename STRING,

designation STRING,

department STRING,

salary FLOAT

);

-- iii. Insert the ten records into employee table.

INSERT INTO Employee\_DB.employee VALUES

(101, 'John', 'Manager', 'Sales', 25000),

(102, 'Alice', 'Developer', 'IT', 30000),

(103, 'Bob', 'Analyst', 'Finance', 28000),

(104, 'Emma', 'Designer', 'Sales', 22000),

(105, 'Mike', 'Manager', 'HR', 35000),

(106, 'Sarah', 'HR', 'HR', 28000),

(107, 'David', 'Sales', 'Sales', 26000),

(108, 'Mary', 'Finance', 'Finance', 29000),

(109, 'James', 'IT', 'IT', 32000),

(110, 'Laura', 'Developer', 'IT', 27000);

-- iv. Retrieve the employee records that have salary greater than 20000 and from “sales” department

SELECT \*

FROM Employee\_DB.employee

WHERE salary > 20000 AND department = 'Sales';

-- v. Calculate the average salary obtained by employees from all department. Result should display the department and average salary.

SELECT department, AVG(salary) AS avg\_salary

FROM Employee\_DB.employee

GROUP BY department;

-- vi. Retrieve the name of department having average salary greater than 20000.

SELECT department

FROM (

SELECT department, AVG(salary) AS avg\_salary

FROM Employee\_DB.employee

GROUP BY department

) sub

WHERE avg\_salary > 20000;

```

**Task 6:**

```sql

-- i. Create a database named Student\_DB

CREATE DATABASE IF NOT EXISTS Student\_DB;

-- ii. Create a table student with columns (class, division, roll\_no, sname, percentage) partitioned by class

CREATE TABLE IF NOT EXISTS Student\_DB.student (

class STRING,

division STRING,

roll\_no INT,

sname STRING,

percentage FLOAT

)

PARTITIONED BY (class STRING);

-- iii. Insert the ten records into student table.

INSERT INTO Student\_DB.student PARTITION (class)

VALUES

('10th', 'A', 101, 'John', 85.5),

('12th', 'B', 102, 'Alice', 78.2),

('11th', 'C', 103, 'Bob', 67.8),

('10th', 'A', 104, 'Emma', 92.1),

('12th', 'B', 105, 'Mike', 75.9),

('10th', 'A', 106, 'Sarah', 88.3),

('12th', 'B', 107, 'David', 82.6),

('11th', 'C', 108, 'Mary', 91.7),

('10th', 'A', 109, 'James', 79.4),

('12th', 'B', 110, 'Laura', 70.5);

-- iv. Retrieve the student records that have percentage greater than 60 and from “TY” class

SELECT \*

FROM Student\_DB.student

WHERE percentage > 60 AND class = 'TY';

-- v. Calculate the average percentage obtained by students from “TY” Class. Result should display the class and average percentage.

SELECT class, AVG(percentage) AS avg\_percentage

FROM Student\_DB.student

WHERE class = 'TY'

GROUP BY class;

-- vi. Retrieve the name of class having average percentage greater than 60.

SELECT class

FROM (

SELECT class, AVG(percentage) AS avg\_percentage

FROM Student\_DB.student

GROUP BY class

) sub

WHERE avg\_percentage > 60;

PIG -7

-- i. Create a database named Employee\_DB

CREATE DATABASE Employee\_DB;

-- ii. Create a table employee with columns (empid, ename, department, salary)

CREATE TABLE Employee\_DB.employee (empid:int, ename:chararray, department:chararray, salary:float);

-- iii. Insert the ten records into employee table.

INSERT INTO Employee\_DB.employee VALUES

(1, 'John', 'IT', 25000),

(2, 'Alice', 'HR', 22000),

(3, 'Bob', 'Finance', 28000),

(4, 'Emma', 'IT', 30000),

(5, 'Mike', 'Finance', 27000),

(6, 'Sarah', 'HR', 23000),

(7, 'David', 'IT', 26000),

(8, 'Mary', 'Finance', 29000),

(9, 'James', 'IT', 31000),

(10, 'Laura', 'HR', 24000);

-- iv. Load the employee data from the HDFS. Utilize the FOREACH operator to transform the employee table to extract the fields ‘empid’, ‘ename’,’salary’ and display the output.

employee\_data = LOAD 'hdfs://path/to/employee\_data' USING PigStorage(',') AS (empid:int, ename:chararray, department:chararray, salary:float);

transformed\_data = FOREACH employee\_data GENERATE empid, ename, salary;

-- Display the output

DUMP transformed\_data;

-- v. Filter the employee data to retrieve records of employees earning a salary greater than 20,000.

high\_paid\_employees = FILTER employee\_data BY salary > 20000;

-- Display the output

DUMP high\_paid\_employees;

**Task 8**

-- i. Create a database named Employee\_DB

CREATE DATABASE Employee\_DB;

-- ii. Create a table employee with columns (empid, ename, department, salary)

CREATE TABLE Employee\_DB.employee (empid:int, ename:chararray, department:chararray, salary:float);

-- iii. Insert the ten records into employee table.

INSERT INTO Employee\_DB.employee VALUES

(1, 'John', 'IT', 25000),

(2, 'Alice', 'HR', 22000),

(3, 'Bob', 'Finance', 28000),

(4, 'Emma', 'IT', 30000),

(5, 'Mike', 'Finance', 27000),

(6, 'Sarah', 'HR', 23000),

(7, 'David', 'IT', 26000),

(8, 'Mary', 'Finance', 29000),

(9, 'James', 'IT', 31000),

(10, 'Laura', 'HR', 24000);

-- iv. Group the employee dataset by department and display the output

grouped\_data = GROUP Employee\_DB.employee BY department;

-- Display the output

DUMP grouped\_data;

-- v. Sort the employee dataset based on the salary field in descending order to identify the highest-paid employees first.

sorted\_data = ORDER Employee\_DB.employee BY salary DESC;

-- Display the output

DUMP sorted\_data;

Task 9

`

-- i. Create a database named Student\_DB

CREATE DATABASE Student\_DB;

-- ii. Create a table student\_info with columns (exam\_no, name, class, department)

CREATE TABLE Student\_DB.student\_info (exam\_no:int, name:chararray, class:chararray, department:chararray);

-- iii. Insert the five records into student table.

INSERT INTO Student\_DB.student\_info VALUES

(1, 'John', '10th', 'Science'),

(2, 'Alice', '12th', 'Commerce'),

(3, 'Bob', '11th', 'Arts'),

(4, 'Emma', '10th', 'Science'),

(5, 'Mike', '12th', 'Commerce');

-- iv. Create a table result\_info with columns (exam\_no, percentage, grade)

CREATE TABLE Student\_DB.result\_info (exam\_no:int, percentage:float, grade:chararray);

-- v. Insert the five records into result\_info table.

INSERT INTO Student\_DB.result\_info VALUES

(1, 85.5, 'A'),

(2, 78.2, 'B'),

(3, 67.8, 'B'),

(4, 92.1, 'A'),

(5, 75.9, 'B');

-- vi. Perform an inner join between the student\_info and result\_info and display output.

joined\_data = JOIN Student\_DB.student\_info BY exam\_no, Student\_DB.result\_info BY exam\_no;

-- Display the output

DUMP joined\_data;

```

Task 10

-- i. Create a database named Student\_DB

CREATE DATABASE Student\_DB;

-- ii. Create a table student\_info with columns (exam\_no, name, department, class, division)

CREATE TABLE Student\_DB.student\_info (exam\_no:int, name:chararray, department:chararray, class:chararray, division:chararray);

-- iii. Insert the five records into student table.

INSERT INTO Student\_DB.student\_info VALUES

(1, 'John', 'Science', '10th', 'A'),

(2, 'Alice', 'Commerce', '12th', 'B'),

(3, 'Bob', 'Arts', '11th', 'C'),

(4, 'Emma', 'Science', '10th', 'A'),

(5, 'Mike', 'Commerce', '12th', 'B');

-- iv. Create a table result\_info with columns (exam\_no, percentage, grade)

CREATE TABLE Student\_DB.result\_info (exam\_no:int, percentage:float, grade:chararray);

-- v. Insert the five records into student table.

INSERT INTO Student\_DB.result\_info VALUES

(1, 85.5, 'A'),

(2, 78.2, 'B'),

(3, 67.8, 'B'),

(4, 92.1, 'A'),

(5, 75.9, 'B');

-- vi. Perform the left, right and full outer join between the student\_info and result\_info and display output.

left\_join\_data = JOIN Student\_DB.student\_info BY exam\_no LEFT OUTER, Student\_DB.result\_info BY exam\_no;

right\_join\_data = JOIN Student\_DB.student\_info BY exam\_no RIGHT OUTER, Student\_DB.result\_info BY exam\_no;

full\_outer\_join\_data = JOIN Student\_DB.student\_info BY exam\_no FULL OUTER, Student\_DB.result\_info BY exam\_no;

-- Display the output

DUMP left\_join\_data;

DUMP right\_join\_data;

DUMP full\_outer\_join\_data;

```

Task 11

-- i. Create a database named Student\_DB

CREATE DATABASE Student\_DB;

-- ii. Create a table result\_info with columns (exam\_no, name, percentage)

CREATE TABLE Student\_DB.result\_info (exam\_no:int, name:chararray, percentage:float);

-- iii. Insert the ten records into result\_info table.

INSERT INTO Student\_DB.result\_info VALUES

(1, 'John', 85.5),

(2, 'Alice', 78.2),

(3, 'Bob', 67.8),

(4, 'Emma', 92.1),

(5, 'Mike', 75.9),

(6, 'Sarah', 88.3),

(7, 'David', 82.6),

(8, 'Mary', 91.7),

(9, 'James', 79.4),

(10, 'Laura', 70.5);

-- iv. Find the top three rankers and display their details.

ranked\_data = RANK result\_info BY percentage DESC;

top\_three = LIMIT ranked\_data 3;

-- Display the output

DUMP top\_three;

```

Task 12

1-- i. Create a database named Student\_DB

CREATE DATABASE Student\_DB;

-- ii. Create a table result\_info with columns (exam\_no, name, percentage)

CREATE TABLE Student\_DB.result\_info (exam\_no:int, name:chararray, percentage:float);

-- iii. Insert the ten records into result\_info table.

INSERT INTO Student\_DB.result\_info VALUES

(1, 'John', 85.5),

(2, 'Alice', 78.2),

(3, 'Bob', 67.8),

(4, 'Emma', 92.1),

(5, 'Mike', 75.9),

(6, 'Sarah', 88.3),

(7, 'David', 82.6),

(8, 'Mary', 91.7),

(9, 'James', 79.4),

(10, 'Laura', 70.5);

-- iv. Randomly sample 20% of records from the result\_info dataset to conduct statistical analysis and display the output

sampled\_data = SAMPLE result\_info 0.2;

-- Display the output

DUMP sampled\_data;

-- v. Divide the result\_info dataset into two separate datasets based on a condition on their percentage being greater than equal to 60% or below the 60%.

above\_sixty = FILTER result\_info BY percentage >= 60;

below\_sixty = FILTER result\_info BY percentage < 60;

-- Display the outputs

DUMP above\_sixty;

DUMP below\_sixty;

```

problem statements in R:

**Task 13 - 1**

# a) ls() - List objects in the current R workspace

ls()

# b) rm() - Remove objects from the current R workspace

# Example: Remove object named "my\_data"

rm(my\_data)

# c) getwd() - Get the current working directory

getwd()

# d) save() - Save objects to a file

# Example: Save objects "my\_data" and "my\_results" to a file named "data\_results.RData"

save(my\_data, my\_results, file = "data\_results.RData")

# e) load() - Load objects from a file

# Example: Load objects from the file "data\_results.RData"

load("data\_results.RData")

```

**2-**

# Collect student information

roll\_no <- 1:5

name <- c("John", "Alice", "Bob", "Emma", "Mike")

marks\_subject1 <- c(80, 75, 85, 90, 78)

marks\_subject2 <- c(85, 72, 88, 92, 80)

marks\_subject3 <- c(78, 70, 90, 85, 75)

marks\_subject4 <- c(90, 85, 92, 88, 82)

marks\_subject5 <- c(82, 78, 86, 90, 79)

# Calculate total marks and percentage

total\_marks <- marks\_subject1 + marks\_subject2 + marks\_subject3 + marks\_subject4 + marks\_subject5

percentage <- (total\_marks / 500) \* 100

# Create a data frame to store student information

student\_info <- data.frame(roll\_no, name, marks\_subject1, marks\_subject2, marks\_subject3, marks\_subject4, marks\_subject5, total\_marks, percentage)

# Display all student information

print(student\_info)

```

**3**

# Display the list of built-in datasets in R

data()

# Choose a dataset, for example, "iris"

# Summarize the dataset

summary(iris)

# Plot the graph of the dataset, for example, a boxplot of Sepal Length by Species

boxplot(Sepal.Length ~ Species, data = iris)

```

the tasks in R: Task 14

1. First, create a CSV file containing student information. You can use any text editor to create the file. Let's name it "student\_info.csv" and place it in your working directory. The file should look like this:

```

roll\_no,name,percentage

1,John,85.5

2,Alice,78.2

3,Bob,67.8

4,Emma,92.1

5,Mike,75.9

```

2. Now, in RStudio, you can read this CSV file and display the data:

```R

# Read the CSV file

student\_data <- read.csv("student\_info.csv")

# Display the data

print(student\_data)

```

14 - 2

You can create the data in R Console and then store it into a CSV file.

```R

# Create student information data

roll\_no <- c(1, 2, 3, 4, 5)

name <- c("John", "Alice", "Bob", "Emma", "Mike")

percentage <- c(85.5, 78.2, 67.8, 92.1, 75.9)

# Combine data into a data frame

student\_info <- data.frame(roll\_no, name, percentage)

# Display the data frame

print(student\_info)

# Store data into a CSV file

write.csv(student\_info, file = "student\_info\_new.csv", row.names = FALSE)

```

This script creates the student information data in R Console, displays it, and then stores it into a new CSV file named "student\_info\_new.csv" in your working directory.

Ensure you have proper permissions to read from and write to the directory where you are working. Adjust the file paths accordingly if your files are located in a different directory.

Here are the solutions to the tasks using R:

Task 15

```R

# i. Create the data frame for Student Information

student\_info <- data.frame(

roll\_no = 1:5,

student\_name = c("John", "Alice", "Bob", "Emma", "Mike"),

percentage = c(85.5, 78.2, 67.8, 92.1, 75.9)

)

# Display students who have failed, pass, second class, first class, and distinction using subset function

failed <- subset(student\_info, percentage < 40)

pass <- subset(student\_info, percentage >= 40 & percentage < 60)

second\_class <- subset(student\_info, percentage >= 60 & percentage < 75)

first\_class <- subset(student\_info, percentage >= 75 & percentage < 90)

distinction <- subset(student\_info, percentage >= 90)

print("Failed:")

print(failed)

print("Pass:")

print(pass)

print("Second Class:")

print(second\_class)

print("First Class:")

print(first\_class)

print("Distinction:")

print(distinction)

# ii. Create the data frame for Employee Information

employee\_info <- data.frame(

emp\_id = 1:5,

emp\_name = c("John", "Alice", "Bob", "Emma", "Mike"),

designation = c("Manager", "Assistant", "Clerk", "Manager", "Assistant"),

salary = c(35000, 30000, 25000, 40000, 28000)

)

# Display the list of employees having salary greater than 30,000 using subset function

employees\_gt\_30000 <- subset(employee\_info, salary > 30000, select = -designation)

print(employees\_gt\_30000)

```

Task 16```R

# i. Create two data frames containing Student Information

student\_details <- data.frame(

exam\_no = 1:5,

name = c("John", "Alice", "Bob", "Emma", "Mike"),

class = c("10th", "12th", "11th", "10th", "12th")

)

student\_marks <- data.frame(

exam\_no = 1:5,

S1 = c(85, 78, 67, 92, 75),

S2 = c(80, 75, 68, 90, 79),

S3 = c(75, 70, 65, 88, 72),

S4 = c(90, 85, 70, 95, 80),

S5 = c(82, 78, 72, 89, 76)

)

# Merge these two data frames on the basis of exam no and display the result

merged\_data <- merge(student\_details, student\_marks, by = "exam\_no")

print(merged\_data)

# ii. Create two data frames containing Student Information

# Merge these two data frames on the basis of (exam no, name) and display the result

merged\_data <- merge(student\_details, student\_marks, by = c("exam\_no", "name"))

print(merged\_data)

```

Task 17

# i. Create two data frames containing Student Information

details <- data.frame(

roll\_no = 1:5,

name = c("John", "Alice", "Bob", "Emma", "Mike"),

class = c("10th", "12th", "11th", "10th", "12th")

)

marks <- data.frame(

roll\_no = 1:5,

name = c("John", "Alice", "Bob", "Emma", "Mike"),

total\_marks = c(450, 430, 410, 470, 420),

percentage = c(90, 86, 82, 94, 84)

)

# Display the data of these two data frames and combine the columns using cbind() function

combined\_data <- cbind(details[, c("roll\_no", "name", "class")], marks[, c("total\_marks", "percentage")])

print(combined\_data)

# ii. Create two data frames containing employee information

sales\_dept <- data.frame(

emp\_id = 1:3,

name = c("John", "Alice", "Bob")

)

finance\_dept <- data.frame(

emp\_id = 4:5,

name = c("Emma", "Mike")

)

# Combine the rows of these two data frames using rbind() and display the result

combined\_employees <- rbind(sales\_dept, finance\_dept)

print(combined\_employees)

```

These scripts should work in RStudio. Adjust them according to your specific requirements or dataset structure if needed.

Here's how you can complete the tasks in RStudio using R:

18

# i. Create data frame containing employee information

Employee <- data.frame(

empid = c(1, 2, 3, 4, 5),

name = c("John", "Alice", "Bob", "Emma", "Mike"),

salary = c(50000, 60000, 45000, 55000, 48000)

)

# Display the data in data frame

print(Employee)

# Sort the data in data frame on the basis of salary in increasing order

Employee\_sorted <- Employee[order(Employee$salary), ]

# Display the sorted result

print(Employee\_sorted)

# ii. Create data frame containing student information

Student <- data.frame(

rollno = c(101, 102, 103, 104, 105),

name = c("John", "Alice", "Bob", "Emma", "Mike"),

division = c("A", "B", "C", "B", "A"),

percentage = c(85.5, 78.2, 67.8, 92.1, 75.9)

)

# Display the data in data frame

print(Student)

# Sort the data in data frame first on the basis of division in increasing order and then on the basis of percentage in decreasing order

Student\_sorted <- Student[order(Student$division, -Student$percentage), ]

# Display the sorted result

print(Student\_sorted)

```

19

# i. Create data frame containing student information

Student <- data.frame(

rollno = c(101, 102, 103, 104, 105),

name = c("John", "Alice", "Bob", "Emma", "Mike"),

S1 = c(85, 78, 67, 92, 75),

S2 = c(80, 75, 68, 90, 79),

S3 = c(75, 70, 65, 88, 72)

)

# Display the data in data frame

print(Student)

# ii. Apply melt() function on this student data frame

library(reshape2)

Student\_long <- melt(Student, id.vars = c("rollno", "name"), measure.vars = c("S1", "S2", "S3"))

# Display the result

print(Student\_long)

# iii. Apply dcast() function on the Student\_long

Student\_wide <- dcast(Student\_long, rollno + name ~ variable, value.var = "value")

# Display the result

print(Student\_wide)

```

20

# Create data frame containing employee information

Employee <- data.frame(

empid = 1:10,

name = paste("Employee", 1:10, sep = "\_"),

salary = sample(40000:80000, 10)

)

# i. All the records in data frame

print(Employee)

# ii. First 3 rows in data frame

print(Employee[1:3, ])

# iii. Last 3 rows in data frame

print(Employee[(nrow(Employee) - 2):nrow(Employee), ])

# iv. Only second column of data frame

print(Employee[, 2])

```

These scripts should work in RStudio. Make sure to load the required packages, especially for tasks 19ii and 19iii, where you need the `reshape2` package. Adjust them according to your specific requirements or dataset structure if needed.