

School Ranking Analysis.

DESCRIPTION

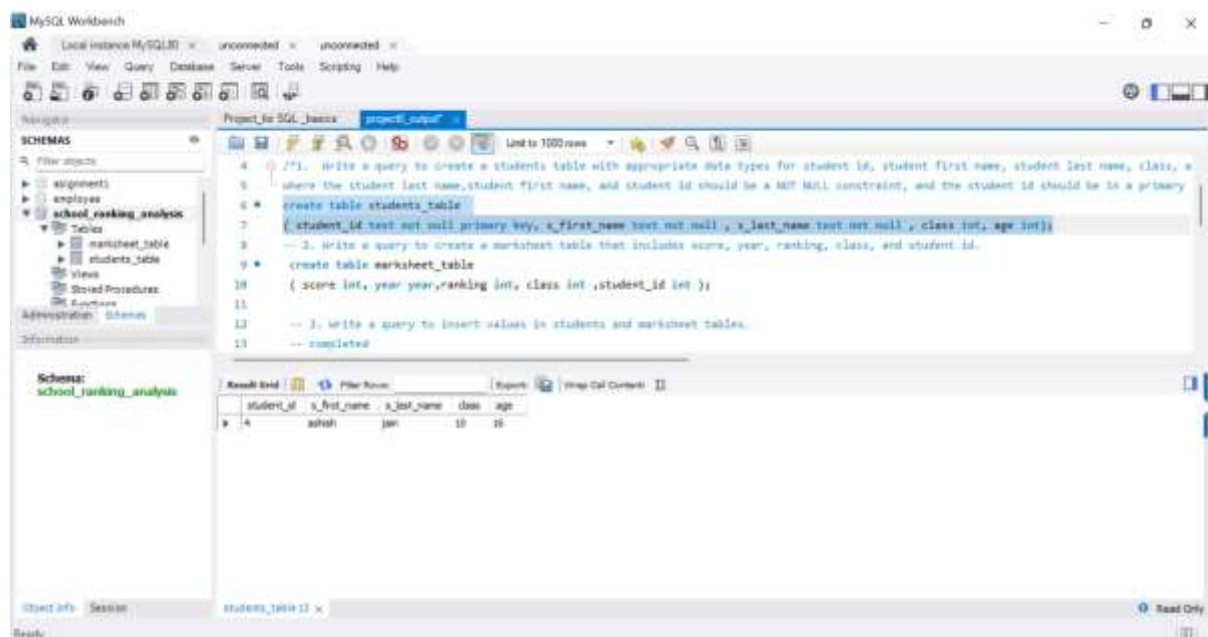
Consider an institution that wants to store the students' details and their marks records to track their progress. The database would contain the students' information, marks of the students with the rank that can be viewed, updated, and evaluated for the performance evaluation.

Objective:

The design of the database helps to easily retrieve thousands of student records.

Task to be performed:

- Write a query to create a **students** table with appropriate data types for student id, student first name, student last name, class, and age where the student last name, student first name, and student id should be a **NOT NULL constraint**, and the student id should be in a **primary key**.



2. Write a query to create a **marksheet** table that includes score, year, ranking, class, and student id.

The screenshot shows the MySQL Workbench interface. The left sidebar displays the 'Schemas' panel with a tree view containing 'school_ranking_analysis' and its sub-tables 'marksheet_table' and 'students_table'. The main editor window shows a SQL script with the following content:

```
4 -- 1. Write a query to create a students table with appropriate data types for student id, student first name, student last name, class, &
5 where the student last name, student first name, and student id should be a NOT NULL constraint, and the student id should be in a primary
6
7 create table students_table
8 ( student_id text not null, s_first_name text not null, s_last_name text not null, class int, age int);
9
10 -- 2. Write a query to create a marksheet table that includes score, year, ranking, class, and student id.
11
12 create table marksheet_table
13 ( score int, year year, ranking int, class int, student_id int );
14
15 -- 3. Write a query to insert values in students and marksheet tables.
16 -- completed
17
18 -- 4. Write a query to display student id and student first name from the student table if the age is greater than or
19 equal to 16 and the student's last name is Kumar.
20
21 select student_id, s_first_name
22 from students_table
```

The 'Result Grid' at the bottom shows the output of the last query, displaying student details:

student_id	s_first_name	s_last_name	class	age
4	akash	pat	10	11

4. . Write a query to display student id and student first name from the student table if the age is greater than or equal to 16 and the student's last name is Kumar.

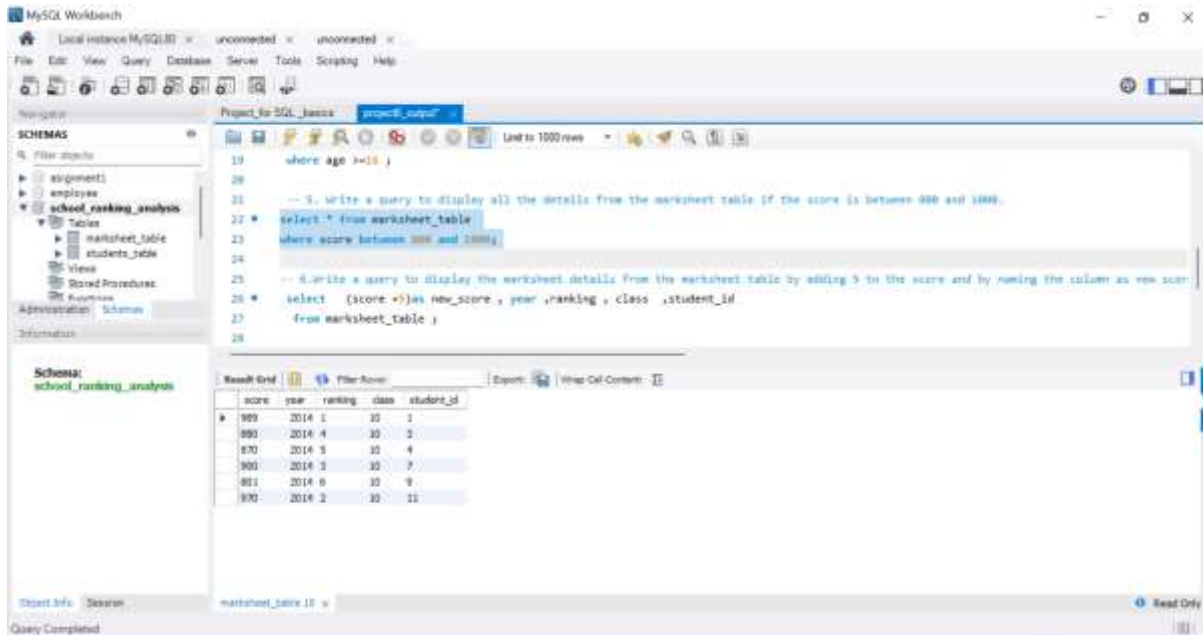
The screenshot shows the MySQL Workbench interface. The left sidebar displays the 'Schemas' panel with a tree view containing 'school_ranking_analysis' and its sub-tables 'marksheet_table' and 'students_table'. The main editor window shows a SQL script with the following content:

```
14 -- completed
15
16 -- 4. write a query to display student id and student first name from the student table if the age is greater than or
17 equal to 16 and the student's last name is Kumar.
18
19 select student_id, s_first_name
20 from students_table
21 where age >= 16 and s_last_name = 'Kumar';
22
23 -- 5. Write a query to display all the details from the marksheet table if the score is between 800 and 1000.
24
25 select * from marksheet_table
```

The 'Result Grid' at the bottom shows the output of the last query, displaying student details:

student_id	s_first_name
3	akash
11	raaj

5. Write a query to display all the details from the marksheet table **if the score is between 800 and 1000**.



The screenshot shows the MySQL Workbench interface. The left sidebar displays the 'Schemas' tree with 'school_ranking_analysis' selected, showing tables 'marksheet_table' and 'students_table'. The main editor contains the following SQL query:

```

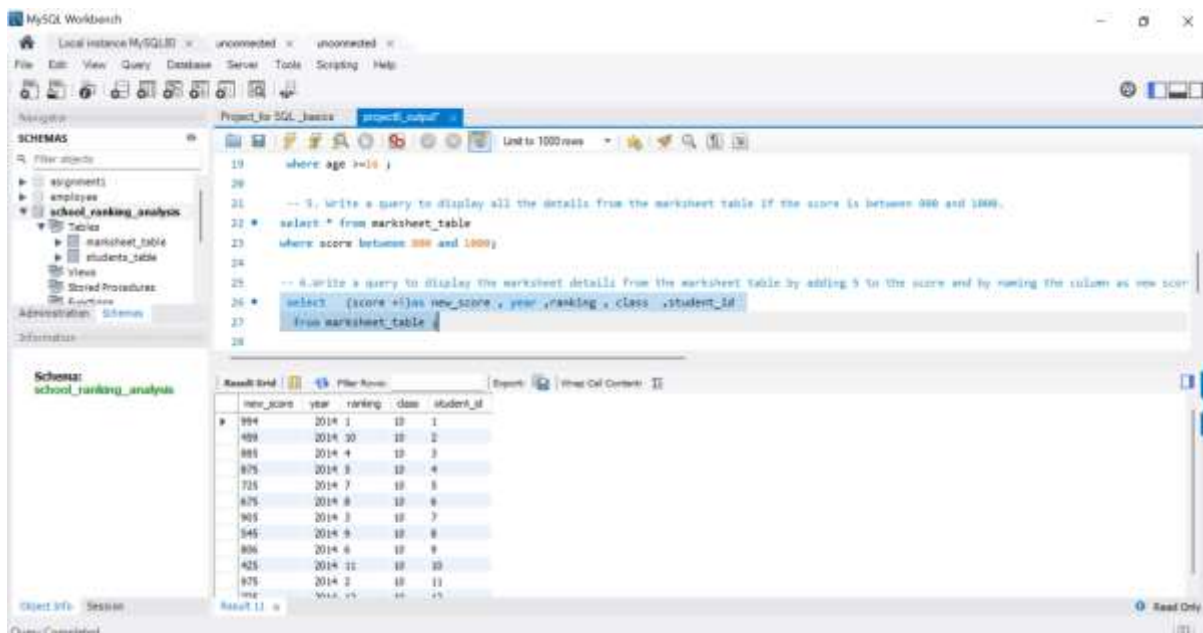
19 where age >=16 )
20
21 -- 3. Write a query to display all the details from the marksheet table if the score is between 800 and 1000.
22 * select * from marksheet_table
23 where score between 800 and 1000;
24
25 -- 4. write a query to display the worksheet details from the marksheet table by adding 5 to the score and by naming the column as new score
26 * select (score+5) as new_score , year , ranking , class , student_id
27 from marksheet_table ;
28

```

The 'Result Grid' at the bottom shows the output of the query:

score	year	ranking	class	student_id
989	2014	1	10	1
880	2014	4	10	3
870	2014	5	10	4
900	2014	3	10	7
881	2014	6	10	8
870	2014	2	10	11

- 6. Write a query to display the marksheet details from the marksheet table by **adding 5 to the score** and by naming the **column as new score**.



The screenshot shows the MySQL Workbench interface. The left sidebar displays the 'Schemas' tree with 'school_ranking_analysis' selected, showing tables 'marksheet_table' and 'students_table'. The main editor contains the following SQL query:

```

19 where age >=16 )
20
21 -- 3. Write a query to display all the details from the marksheet table if the score is between 800 and 1000.
22 * select * from marksheet_table
23 where score between 800 and 1000;
24
25 -- 4. write a query to display the worksheet details from the marksheet table by adding 5 to the score and by naming the column as new score
26 * select (score+5) as new_score , year , ranking , class , student_id
27 from marksheet_table ;
28

```

The 'Result Grid' at the bottom shows the output of the query:

new_score	year	ranking	class	student_id
994	2014	1	10	1
885	2014	30	10	2
885	2014	4	10	3
875	2014	5	10	4
725	2014	7	10	5
875	2014	8	10	6
905	2014	3	10	7
945	2014	9	10	8
886	2014	6	10	9
925	2014	11	10	10
875	2014	2	10	11
avg	2014	15	10	15

7. Write a query to display the marksheet table in **descending order of the score**.

The screenshot shows the MySQL Workbench interface. The left sidebar displays the 'Schemas' tree with 'school_ranking_analysis' selected. The main editor window contains the following SQL query:

```
23 where score between 800 and 1000;
24
25 -- 6. write a query to display the marksheet details from the marksheet table by adding 5 to the score and by naming the column as new_score
26 * select (score + 5) as new_score, year, ranking, class, student_id
27 from marksheet_table;
28
29 -- 7. write a query to display the marksheet table in descending order of the score.
30 * select * from marksheet_table
31 order by score desc;
32
```

The 'Result Grid' at the bottom shows the output of the query, displaying 12 rows of data from the 'marksheet_table' ordered by score in descending order.

score	year	ranking	class	student_id
989	2014	1	10	1
970	2014	2	10	11
960	2014	3	10	7
880	2014	4	10	3
870	2014	5	10	4
861	2014	6	10	9
720	2014	7	10	5
720	2014	12	10	12
670	2014	8	10	6
540	2014	9	10	8
454	2014	10	10	2

- 8. Write a query to display details of the students whose **first name starts with a**.

The screenshot shows the MySQL Workbench interface. The left sidebar displays the 'Schemas' tree with 'school_ranking_analysis' selected. The main editor window contains the following SQL query:

```
29 -- 7. write a query to display the marksheet table in descending order of the score.
30 * select * from marksheet_table
31 order by score desc;
32
33 -- 8. Write a query to display details of the students whose first name starts with a.
34
35 * select * from students_table
36 where s_first_name like 'a%';

```

The 'Result Grid' at the bottom shows the output of the query, displaying 1 row of data from the 'students_table' where the first name starts with 'a'.

student_id	s_first_name	s_last_name	class	age
4	ashish	jan	10	10