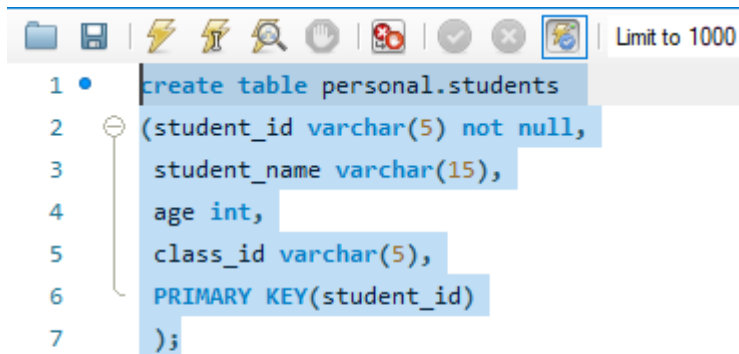
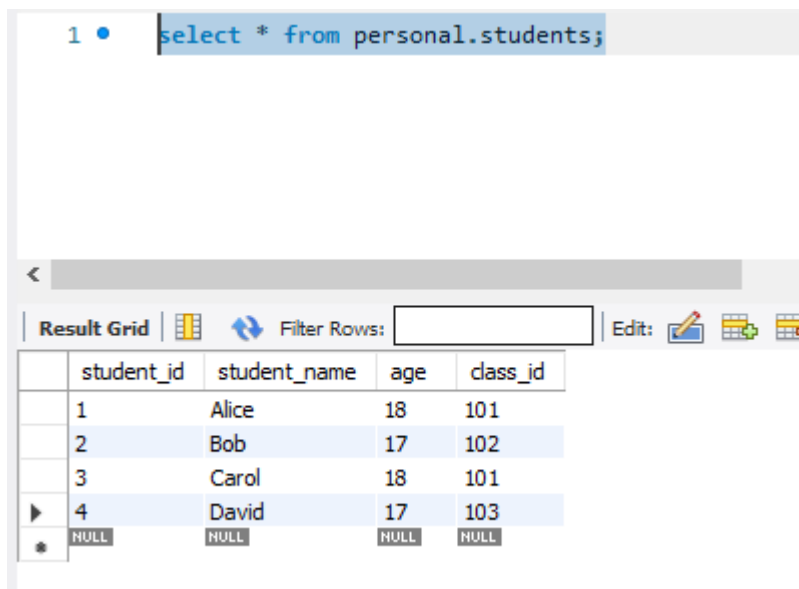


Students Table



The screenshot shows a SQL IDE interface with a toolbar at the top containing icons for file operations, execution, and search. The main area displays a SQL script to create a table named 'personal.students'. The script is as follows:

```
1 • create table personal.students
2   (student_id varchar(5) not null,
3     student_name varchar(15),
4     age int,
5     class_id varchar(5),
6     PRIMARY KEY(student_id)
7   );
```



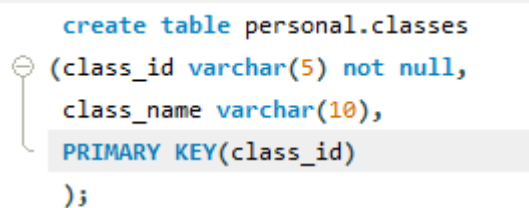
The screenshot shows a SQL IDE interface with a toolbar at the top. The main area displays a SQL query to select all data from the 'personal.students' table:

```
1 • select * from personal.students;
```

Below the query, the 'Result Grid' is displayed, showing the data returned by the query. The grid has columns for 'student_id', 'student_name', 'age', and 'class_id'. The data is as follows:

	student_id	student_name	age	class_id
	1	Alice	18	101
	2	Bob	17	102
	3	Carol	18	101
	4	David	17	103
•	NULL	NULL	NULL	NULL

Classes table



The screenshot shows a SQL IDE interface with a toolbar at the top. The main area displays a SQL script to create a table named 'personal.classes'. The script is as follows:

```
create table personal.classes
(class_id varchar(5) not null,
class_name varchar(10),
PRIMARY KEY(class_id)
);
```

Limit to 1000 rows

```
ALTER TABLE personal.students  
ADD FOREIGN KEY (class_id) REFERENCES personal.classes(class_id);
```

1 • `select * from personal.classes;`

Result Grid

	class_id	class_name
	101	Math
	102	Science
▶	103	History
*	NULL	NULL

Score table

Limit to 1000 rows

```
1 • create table personal.score  
2 ( id int not null,  
3   student_id varchar(5),  
4   subject varchar(10),  
5   score int,  
6   PRIMARY KEY(id)  
7 );
```

```
1 select * from personal.score;
```

Result Grid				
Filter Rows: <input type="text"/>				
Edit:				
	id	student_id	subject	score
	1	1	Math	90
	2	1	Science	85
	3	2	Math	78
	4	2	Science	92
	5	3	Math	88
	6	3	Science	79
	7	4	History	95
*	NULL	NULL	NULL	NULL

Queries :

1. Question: For each class, find the student(s) who scored the highest in Science.

```
select s.student_name ,sc.subject, sc.score from personal.students s
```

```
join personal.score sc on s.student_id = sc.student_id
```

```
where sc.score = (select max(sc1.score)
```

```
from personal.score sc1 where sc1.subject = "Science" group by sc1.subject);
```

```
1 • select s.student_name ,sc.subject, sc.score from personal.students s
2 join personal.score sc on s.student_id = sc.student_id
3 where sc.score = (select max(sc1.score)
4 from personal.score sc1 where sc1.subject = "Science" group by sc1.subject);
```

Result Grid			
Filter Rows: <input type="text"/>			
Export: Wrap Cell Content:			
	student_name	subject	score
▶	Bob	Science	92

2. Question: List the names of students who scored lower in Math than their average Science score.

```
select s.student_name ,sc.subject, sc.score from personal.students s
join personal.score sc on s.student_id = sc.student_id
where sc.subject = "Math" and sc.score < (select avg(sc1.score)
from personal.score sc1 where sc1.subject = "Science" group by sc1.subject);
```

The screenshot shows a SQL query editor with the following code:

```
1 • select s.student_name ,sc.subject, sc.score from personal.students s
2   join personal.score sc on s.student_id = sc.student_id
3   where sc.subject = "Math" and sc.score < (select avg(sc1.score)
4     from personal.score sc1 where sc1.subject = "Science" group by sc1.subject);
```

Below the editor is a toolbar with options: Result Grid, Filter Rows, Export, and Wrap Cell Content. The Result Grid shows the following data:

student_name	subject	score
Bob	Math	78

3. Question: Display the class names with the highest number of students who scored above 80 in any subject.

Query1 : create table personal.temp_score select sc.subject,count(*) as no_of_students_above_80
from personal.score sc where sc.score > 80 group by sc.subject ;

query2 : select subject from personal.temp_score where no_of_students_above_80 =
(select max(no_of_students_above_80) from personal.temp_score tsc);

Query 3: drop table personal.temp_score;

```

1 • create table personal.temp_score select sc.subject,count(*) as no_of_students_above_80
2   from personal.score sc where sc.score > 80 group by sc.subject ;
3
4 • select subject from personal.temp_score where no_of_students_above_80 =
5   (select max(no_of_students_above_80) from personal.temp_score tsc);
6
7 • drop table personal.temp_score;

```

Result Grid		Filter Rows:	Export:	Wrap Cell Content:
subject				
Math				
Science				

4. Question: Find the students who scored the highest in each subject.

```
SELECT m.subject,s.student_name,m.score
```

```
from personal.students s
```

```
inner join personal.score m on s.student_id=m.student_id
```

```
inner join personal.classes su on m.subject=su.class_name
```

```
inner join (select subject
```

```
    ,max(score) as maximum
```

```
    from personal.score ma group by subject
```

```
) highmarks
```

```
ON highmarks.subject=m.subject
```

```
AND highmarks.maximum=m.score
```

```
order by subject,student_name;
```

The screenshot shows a database query editor with a toolbar at the top. The SQL query is as follows:

```

1 • SELECT m.subject,s.student_name,m.score
2     from personal.students s
3         inner join personal.score m on s.student_id=m.student_id
4         inner join personal.classes su on m.subject=su.class_name
5         inner join (select subject
6                     ,max(score) as maximum
7                     from personal.score ma group by subject
8                     ) highmarks
9             ON highmarks.subject=m.subject
10            AND highmarks.maximum=m.score
11     order by subject,student_name;

```

Below the query editor, the 'Result Grid' is displayed with the following data:

	subject	student_name	score
▶	History	David	95
	Math	Alice	90
	Science	Bob	92

5. Question: List the names of students who scored higher than the average of any student's score in their own class.

```
select distinct s.student_name,sc.score,sc.subject from personal.students s
```

```
inner join personal.score sc on s.student_id = sc.student_id
```

```
inner join (select subject,avg(score) a from personal.score group by subject) avgScore
```

```
where sc.score > avgScore.a and sc.subject = avgScore.subject;
```

```
1 • select distinct s.student_name,sc.score,sc.subject from personal.students s
2 inner join personal.score sc on s.student_id = sc.student_id
3 inner join (select subject,avg(score) a from personal.score group by subject) avgScore
4 where sc.score > avgScore.a and sc.subject = avgScore.subject;
```

Result Grid

	student_name	score	subject
▶	Alice	90	Math
	Bob	92	Science
	Carol	88	Math

6. Question: Find the class(es) where the students average age is above the average age of all students.

```
select distinct c.class_name from personal.students s
```

```
inner join personal.classes c on s.class_id = c.class_id
```

```
where age > (select avg(age) from personal.students ) ;
```

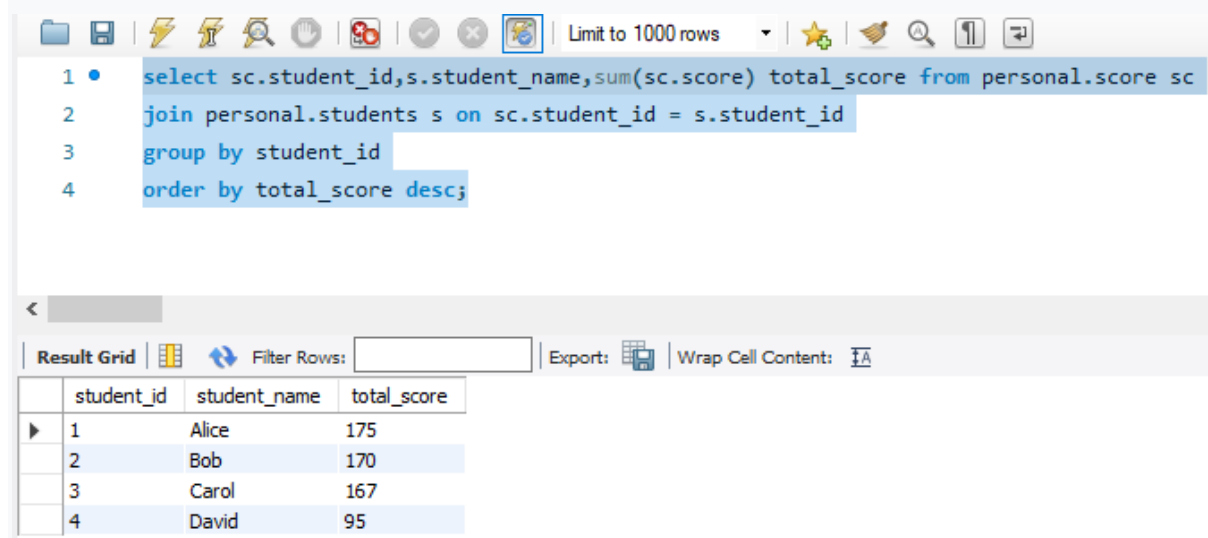
```
1 • select distinct c.class_name from personal.students s
2 inner join personal.classes c on s.class_id = c.class_id
3 where age > (select avg(age) from personal.students ) ;
4
5
```

Result Grid

	class_name
▶	Math

7. Question: Display the student names and their total scores, ordered by the total score in descending order.

```
select sc.student_id,s.student_name,sum(sc.score) total_score from personal.score sc  
join personal.students s on sc.student_id = s.student_id  
group by student_id  
order by total_score desc;
```



The screenshot shows a SQL query editor with a toolbar at the top. The query is entered in a text area and is highlighted in blue. Below the query, a 'Result Grid' is displayed, showing the results of the query in a table format. The table has four columns: 'student_id', 'student_name', and 'total_score'. The results are ordered by 'total_score' in descending order.

	student_id	student_name	total_score
1	1	Alice	175
2	2	Bob	170
3	3	Carol	167
4	4	David	95

8. Question: Find the student(s) who scored the highest in the class with the lowest average score.

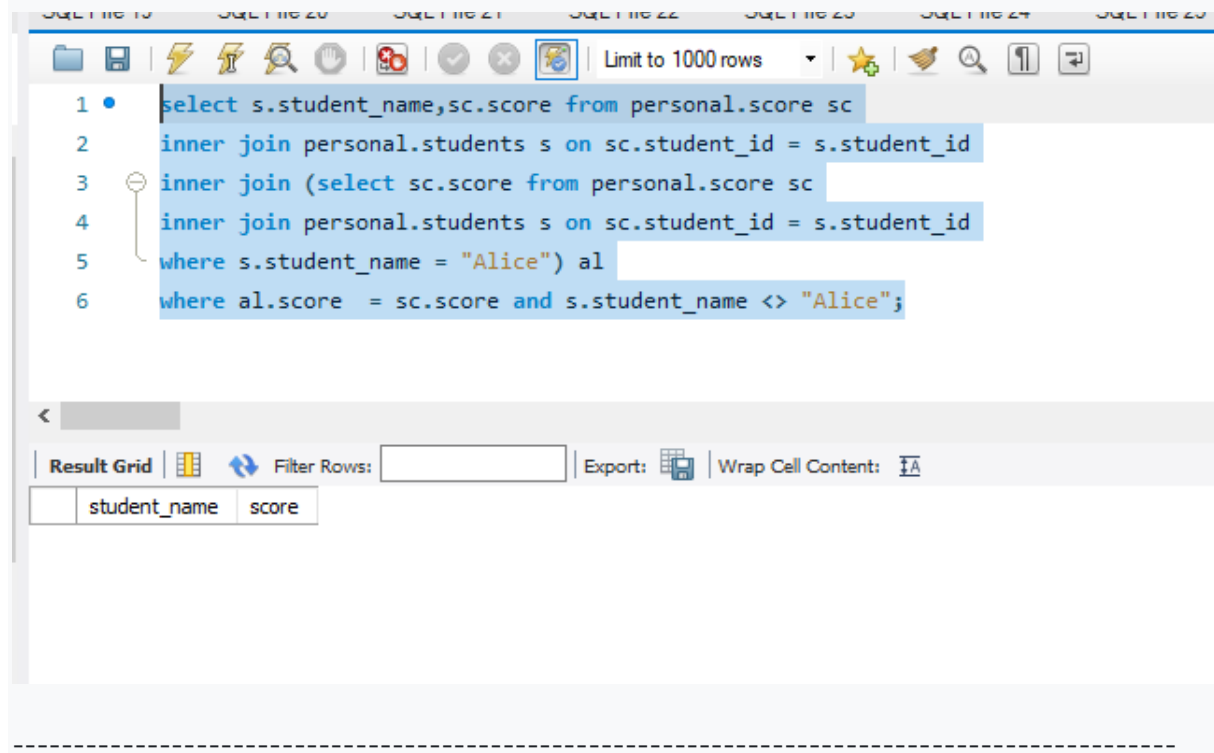
```
select distinct s.student_name , sc.score from personal.students s  
inner join personal.score sc on sc.student_id = s.student_id  
inner join (select max(score) mx from personal.score group by subject) maxScore on  
inner join (select avg(score) avgs from personal.score group by subject) avgScore  
where sc.score = (select max(maxScore.mx) from maxScore) and
```



```
sc.score = (select min(avgScore.avgScore) from avgScore);
```

9. Question: List the names of students who scored the same as Alice in at least one subject.

```
select s.student_name,sc.score from personal.score sc
inner join personal.students s on sc.student_id = s.student_id
inner join (select sc.score from personal.score sc
inner join personal.students s on sc.student_id = s.student_id
where s.student_name = "Alice") al
where al.score = sc.score and s.student_name <> "Alice";
```



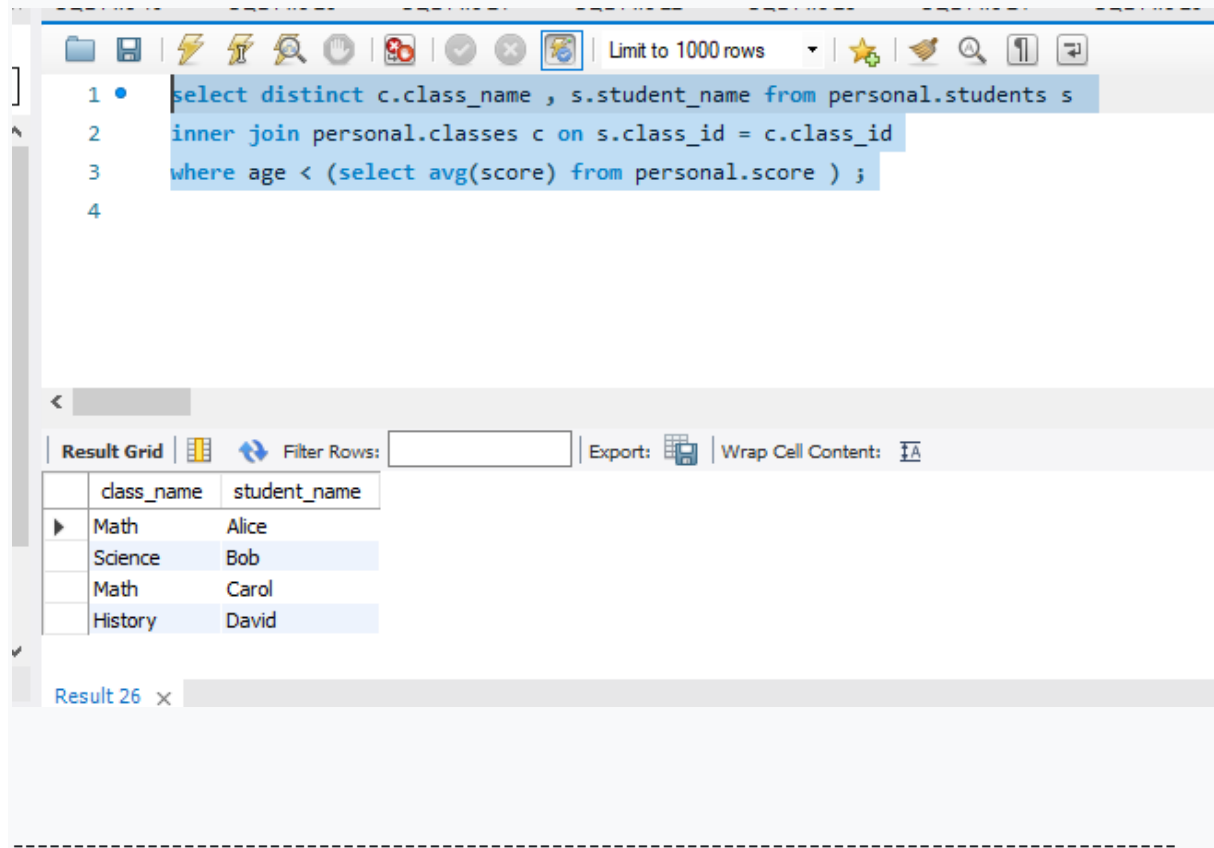
The screenshot shows a SQL IDE interface. The top toolbar includes icons for file operations, execution, and search, along with a 'Limit to 1000 rows' dropdown. The SQL editor contains the following query:

```
1 • select s.student_name,sc.score from personal.score sc
2   inner join personal.students s on sc.student_id = s.student_id
3   inner join (select sc.score from personal.score sc
4   inner join personal.students s on sc.student_id = s.student_id
5   where s.student_name = "Alice") al
6   where al.score = sc.score and s.student_name <> "Alice";
```

Below the editor, the 'Result Grid' tab is active, showing a table with two columns: 'student_name' and 'score'. The table is currently empty.

10. Question: Display the class names along with the number of students who scored below the average score in their class.

```
select distinct c.class_name , s.student_name from personal.students s  
inner join personal.classes c on s.class_id = c.class_id  
where age < (select avg(score) from personal.score ) ;
```



The screenshot shows a database query editor interface. The top toolbar includes icons for file operations, execution, and settings, along with a dropdown menu set to "Limit to 1000 rows". The SQL query is entered in a text area and is highlighted in blue. Below the query editor, the "Result Grid" tab is active, displaying a table with two columns: "class_name" and "student_name". The table contains four rows of data. At the bottom of the interface, there is a tab labeled "Result 26" with a close button (X).

```
1 • select distinct c.class_name , s.student_name from personal.students s  
2 inner join personal.classes c on s.class_id = c.class_id  
3 where age < (select avg(score) from personal.score ) ;  
4
```

class_name	student_name
Math	Alice
Science	Bob
Math	Carol
History	David

Result 26 X