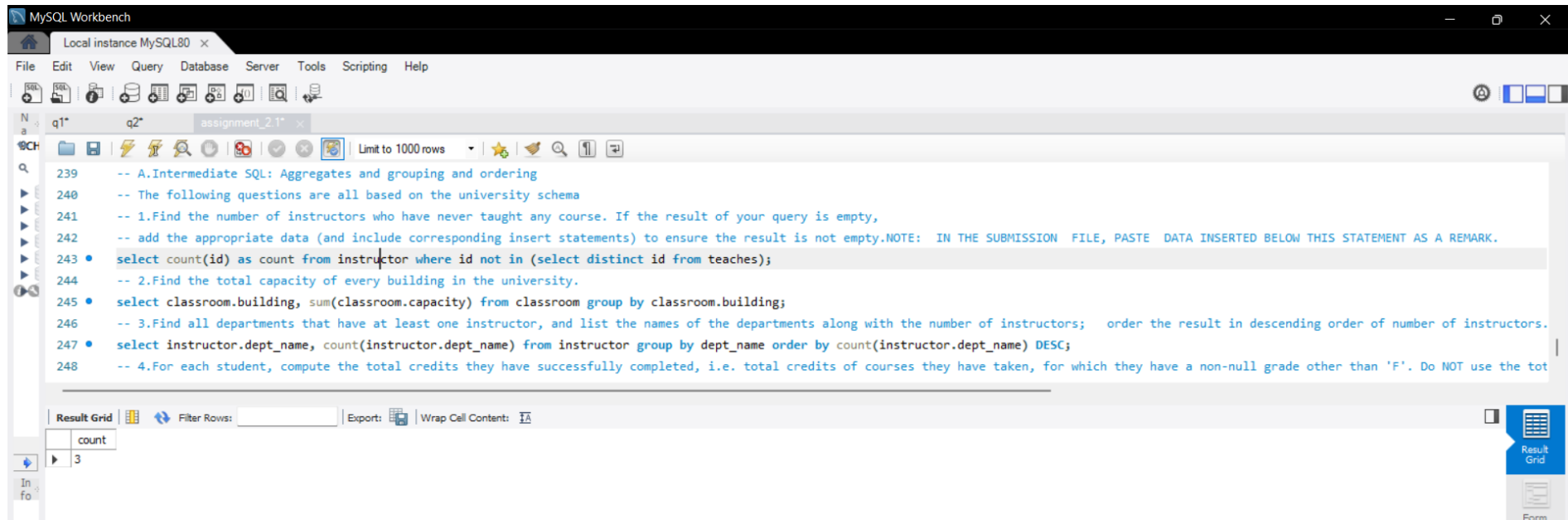


-- A.Intermediate SQL: Aggregates and grouping and ordering  
-- The following questions are all based on the university schema

-- 1.Find the number of instructors who have never taught any course. If the result of your query is empty,  
-- add the appropriate data (and include corresponding insert statements) to ensure the result is not empty.

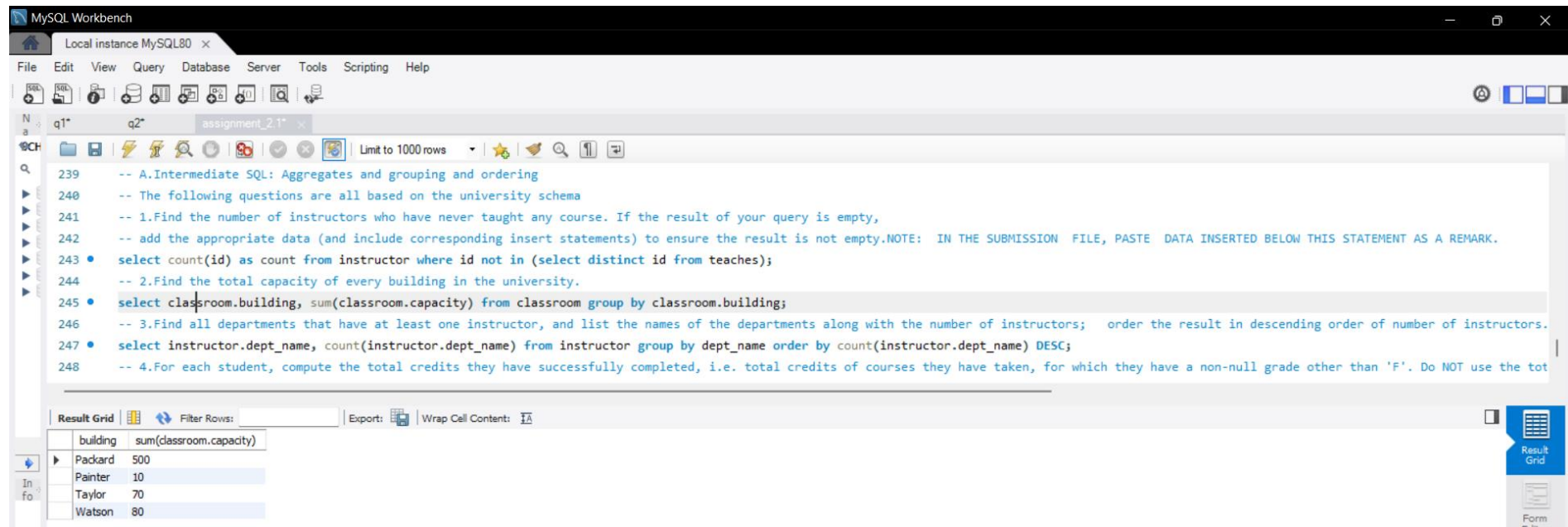
NOTE: IN THE SUBMISSION FILE, PASTE DATA INSERTED BELOW THIS STATEMENT AS A REMARK.

select count(id) as count from instructor where id not in (select distinct id from teaches);



-- 2.Find the total capacity of every building in the university.

select classroom.building, sum(classroom.capacity) from classroom group by classroom.building;



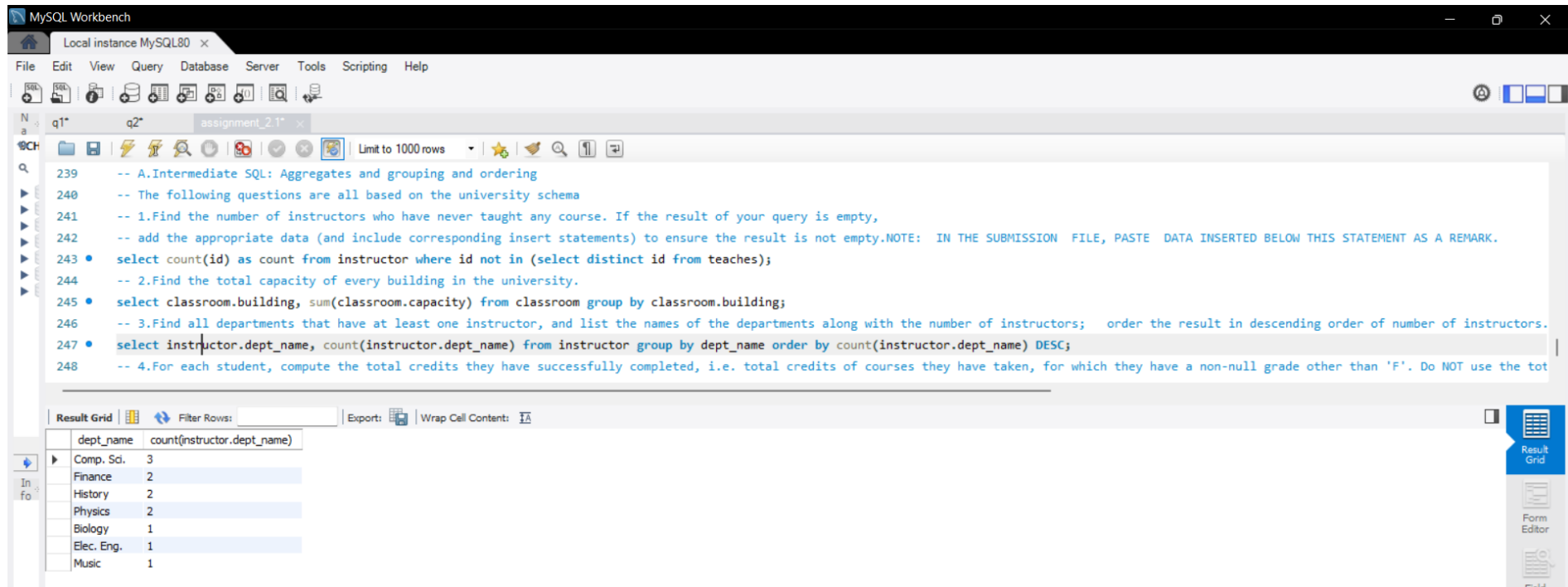
The screenshot shows the MySQL Workbench interface. The SQL editor contains a query for finding the total capacity of every building in the university. The result grid below the editor shows the output of the query.

```
-- A.Intermediate SQL: Aggregates and grouping and ordering
-- The following questions are all based on the university schema
-- 1.Find the number of instructors who have never taught any course. If the result of your query is empty,
-- add the appropriate data (and include corresponding insert statements) to ensure the result is not empty.NOTE: IN THE SUBMISSION FILE, PASTE DATA INSERTED BELOW THIS STATEMENT AS A REMARK.
239 -- 2.Find the total capacity of every building in the university.
240
241
242
243 • select count(id) as count from instructor where id not in (select distinct id from teaches);
244
245 • select classroom.building, sum(classroom.capacity) from classroom group by classroom.building;
246
247 -- 3.Find all departments that have at least one instructor, and list the names of the departments along with the number of instructors; order the result in descending order of number of instructors.
248 • select instructor.dept_name, count(instructor.dept_name) from instructor group by dept_name order by count(instructor.dept_name) DESC;
-- 4.For each student, compute the total credits they have successfully completed, i.e. total credits of courses they have taken, for which they have a non-null grade other than 'F'. Do NOT use the tot
```

building	sum(classroom.capacity)
Packard	500
Painter	10
Taylor	70
Watson	80

-- 3. Find all departments that have at least one instructor, and list the names of the departments along with the number of instructors; order the result in descending order of number of instructors.

```
select instructor.dept_name, count(instructor.dept_name) from instructor group by dept_name order by count(instructor.dept_name) DESC;
```



The screenshot shows the MySQL Workbench interface. The main editor displays a SQL query for question 3. The query is:

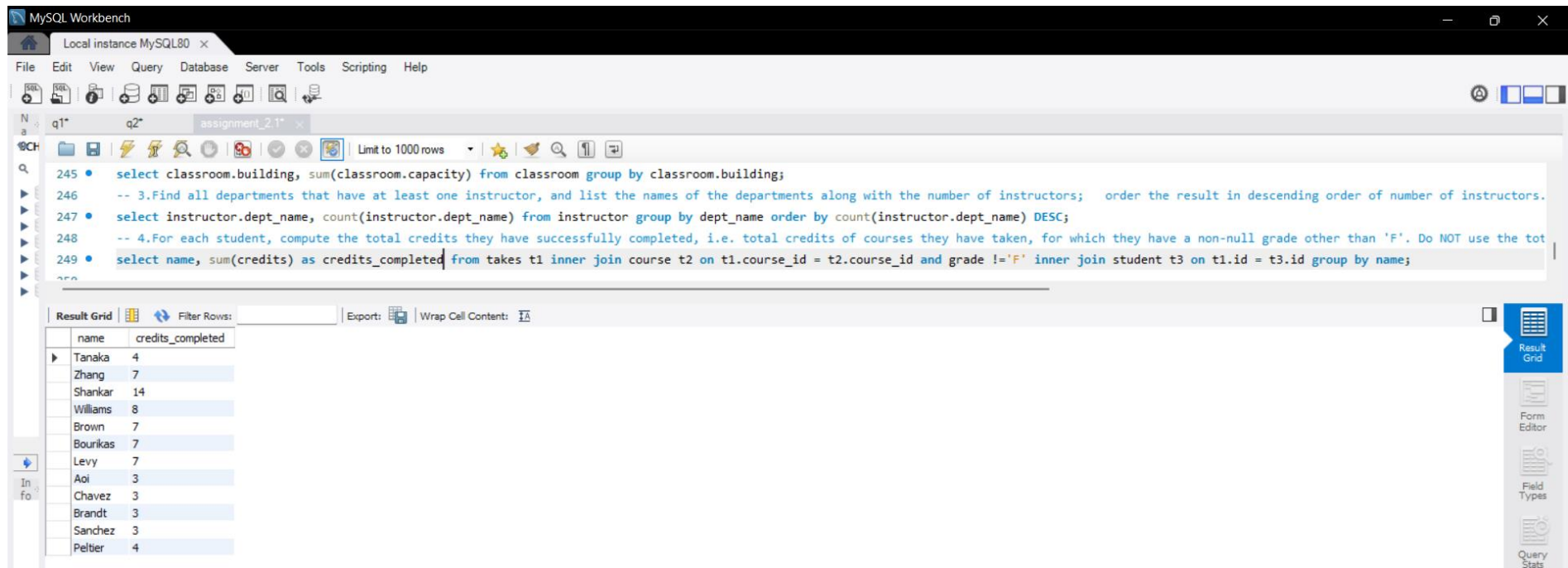
```
-- 3. Find all departments that have at least one instructor, and list the names of the departments along with the number of instructors; order the result in descending order of number of instructors.
select instructor.dept_name, count(instructor.dept_name) from instructor group by dept_name order by count(instructor.dept_name) DESC;
```

The results are shown in the 'Result Grid' at the bottom. The grid has two columns: 'dept\_name' and 'count(instructor.dept\_name)'. The data is as follows:

dept_name	count(instructor.dept_name)
Comp. Sci.	3
Finance	2
History	2
Physics	2
Biology	1
Elec. Eng.	1
Music	1

-- 4. For each student, compute the total credits they have successfully completed, i.e. total credits of courses they have taken, for which they have a non-null grade other than 'F'. Do NOT use the tot\_credits attribute of the student.

select name, sum(credits) as credits\_completed from takes t1 inner join course t2 on t1.course\_id = t2.course\_id and grade != 'F' inner join student t3 on t1.id = t3.id group by name;



The screenshot shows the MySQL Workbench interface. The SQL editor contains the following queries:

```
245 • select classroom.building, sum(classroom.capacity) from classroom group by classroom.building;
246 -- 3. Find all departments that have at least one instructor, and list the names of the departments along with the number of instructors; order the result in descending order of number of instructors.
247 • select instructor.dept_name, count(instructor.dept_name) from instructor group by dept_name order by count(instructor.dept_name) DESC;
248 -- 4. For each student, compute the total credits they have successfully completed, i.e. total credits of courses they have taken, for which they have a non-null grade other than 'F'. Do NOT use the tot
249 • select name, sum(credits) as credits_completed from takes t1 inner join course t2 on t1.course_id = t2.course_id and grade != 'F' inner join student t3 on t1.id = t3.id group by name;
```

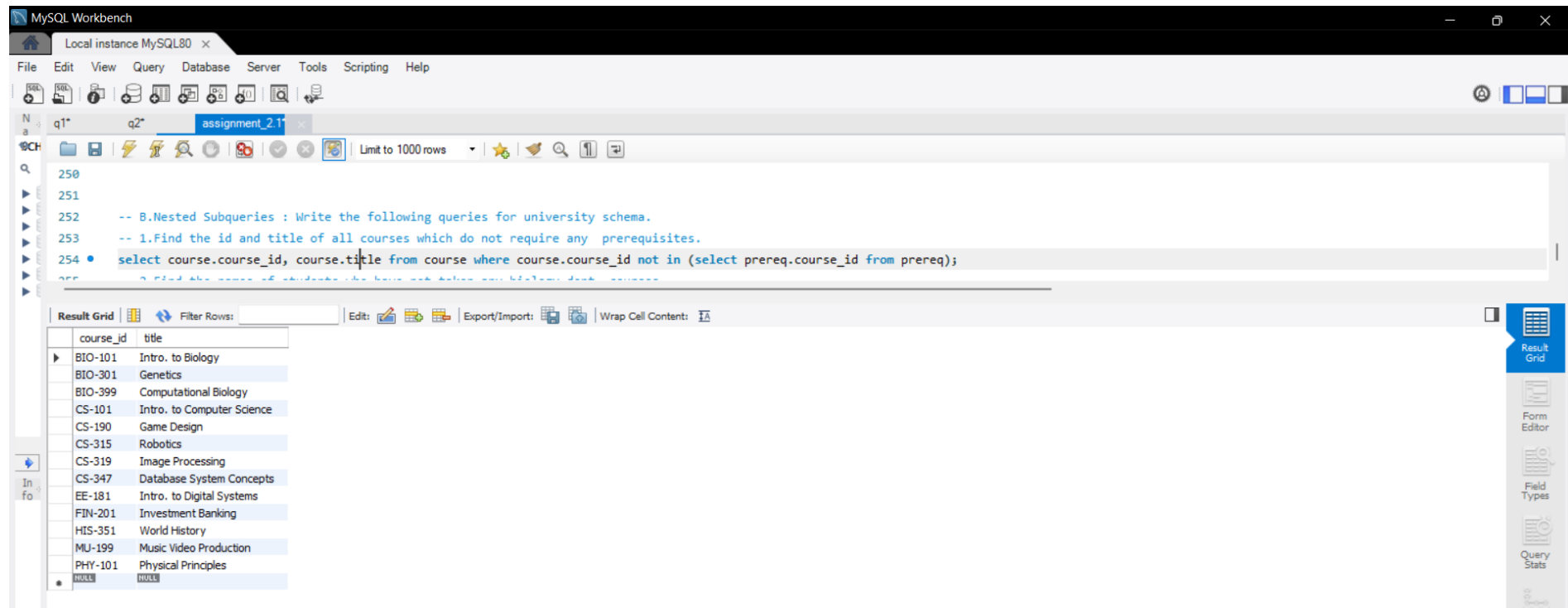
The results are displayed in the Result Grid below the query editor:

name	credits_completed
Tanaka	4
Zhang	7
Shankar	14
Williams	8
Brown	7
Bourikas	7
Levy	7
Aoi	3
Chavez	3
Brandt	3
Sanchez	3
Peltier	4

-- B.Nested Subqueries : Write the following queries for university schema.

-- 1.Find the id and title of all courses which do not require any prerequisites.

`select course.course_id, course.title from course where course.course_id not in (select prereq.course_id from prereq);`



The screenshot shows the MySQL Workbench interface. The query editor displays the following SQL query:

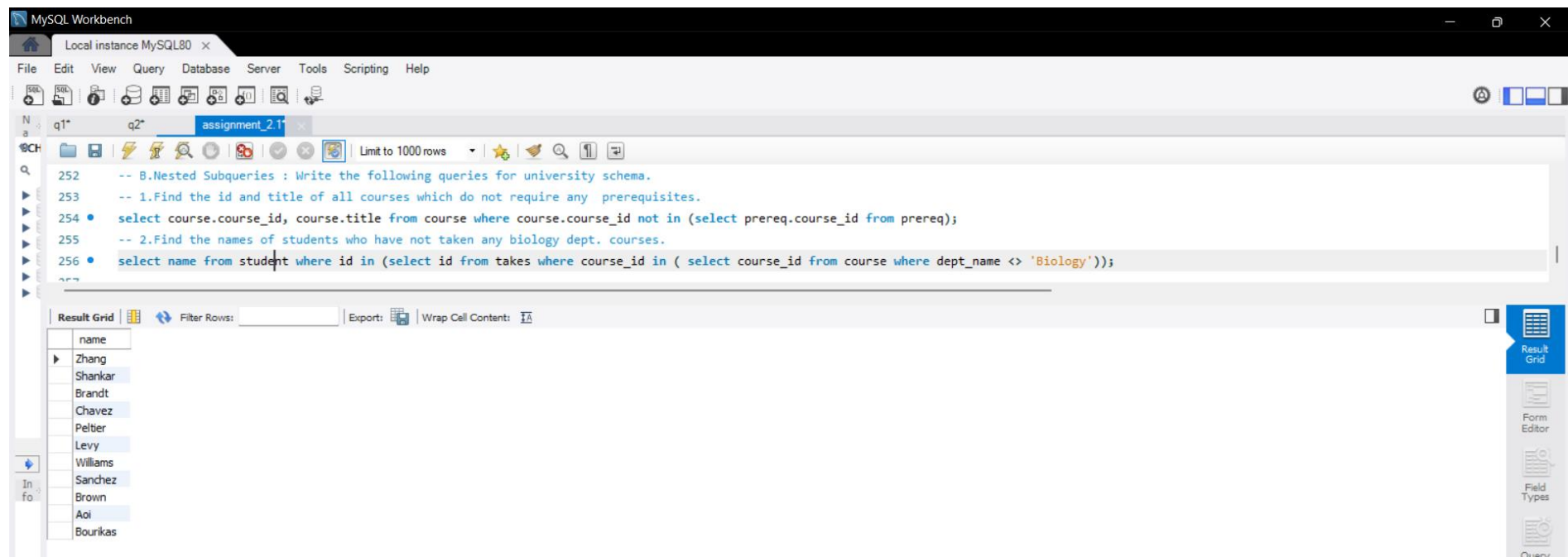
```
-- B.Nested Subqueries : Write the following queries for university schema.  
-- 1.Find the id and title of all courses which do not require any prerequisites.  
select course.course_id, course.title from course where course.course_id not in (select prereq.course_id from prereq);
```

The query results are displayed in the Result Grid below the editor. The results show a list of courses with their IDs and titles, excluding those that have prerequisites.

course_id	title
BIO-101	Intro. to Biology
BIO-301	Genetics
BIO-399	Computational Biology
CS-101	Intro. to Computer Science
CS-190	Game Design
CS-315	Robotics
CS-319	Image Processing
CS-347	Database System Concepts
EE-181	Intro. to Digital Systems
FIN-201	Investment Banking
HIS-351	World History
MU-199	Music Video Production
PHY-101	Physical Principles
NULL	NULL

-- 2.Find the names of students who have not taken any biology dept. courses.

select name from student where id in (select id from takes where course\_id in ( select course\_id from course where dept\_name <> 'Biology'));



The screenshot shows the MySQL Workbench interface. The main window displays a SQL query in the editor, which is a nested subquery designed to find students who have not taken any biology department courses. The query is as follows:

```
-- 8.Nested Subqueries : Write the following queries for university schema.
-- 1.Find the id and title of all courses which do not require any prerequisites.
254 • select course.course_id, course.title from course where course.course_id not in (select prereq.course_id from prereq);
-- 2.Find the names of students who have not taken any biology dept. courses.
256 • select name from student where id in (select id from takes where course_id in ( select course_id from course where dept_name <> 'Biology'));
```

Below the query editor, the 'Result Grid' is visible, showing the results of the query. The results are a list of student names:

name
Zhang
Shankar
Brandt
Chavez
Peltier
Levy
Williams
Sanchez
Brown
Aoi
Bourikas