

به نام آفریننده بیت‌ها



دانشکده‌ی برق و کامپیوتر  
دانشگاه صنعتی اصفهان  
نیم‌سال اول ۱۴۰۱ - ۱۴۰۲

# پایگاه داده ۱

## تمرین دوم

---

نام:

دانیال خراسانی‌زاده

شماره دانشجویی:

۹۹۲۲۳۹۳

استاد درس:

دکتر بصیری





۱

| نوع متغیر        | داده  |
|------------------|---|
| CHAR(3)          | سه حرف مخفف شده ماه های میلادی                |
| DECIMAL(12,2)    | قیمت دلاری محصولات که همگی دو رقم اعشار دارند |
| VARCHAR(MAX)     | نام و نام خانوادگی کاربر                      |
| CHAR(10)         | کد ملی کاربر                                  |
| DOUBLE PRECISION | ذخیره قیمت لحظه ای ارزهای دیجیتال             |
| INT              | تعداد بازدید یک ویدیو                         |

۲

داده VARCHAR(n) برای رشته‌هایی که طول متغیر دارند به کار می‌رود و رشته‌ای با حداکثر طول n را با همان طول اصلی خود رشته ذخیره می‌کند از این جهت در زمان‌هایی که رشته‌ها طول متغیر دارند این نوع داده حجم کمتری اشغال می‌کند. داده CHAR(n) رشته‌ای به طول حداکثر n را بدون توجه به طول اصلی رشته با طول n ذخیره می‌کند. (اگر طول رشته کمتر باشد طول آن را افزایش می‌دهد تا به n برسد.) این نوع داده فضای بیشتری اشغال می‌کند ولی سرعت کارکرد بیشتری دارد و از این جهت اگر طول رشته‌های ما همیشه ثابت باشد انتخاب بهتری است.

۳

۱.۳

```
1 WHERE SUBSTR(first_name, 1, 2) = 'me' AND SUBSTR(lastname, LENGTH(lastname) - 2, 3) = 'avi'
```

۲.۳

```
1 SELECT CONCAT(first_name, last_name) from STUDENT
```



۴

۱.۴

```
1 SELECT id,  
2     name  
3 FROM student  
4 WHERE name LIKE 'M%a';
```

The screenshot shows the pgAdmin 4 web interface. The top navigation bar includes 'File', 'Object', 'Tools', and 'Help'. The main toolbar has icons for various database actions. The 'Query' tab is active, displaying the following SQL query:

```
1 SELECT id,  
2     name  
3 FROM student  
4 WHERE name LIKE 'M%a';
```

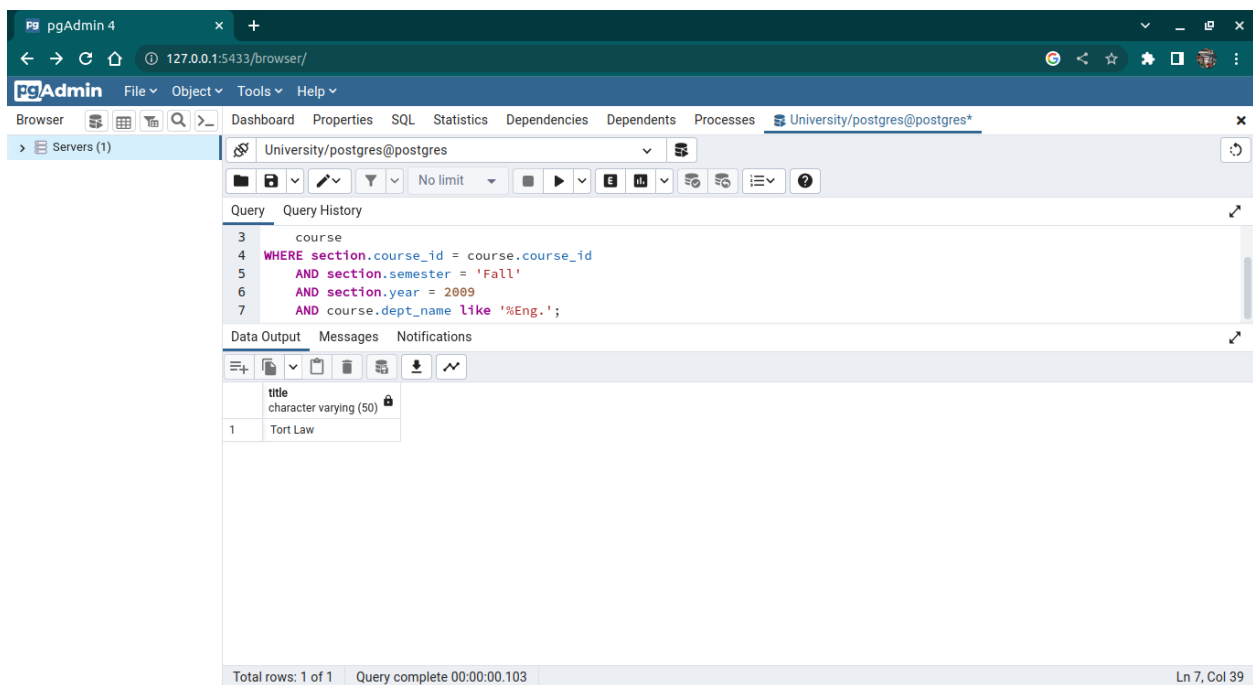
Below the query, the 'Data Output' tab shows the results of the query. The results are displayed in a table with two columns: 'id' and 'name'.

| id | name      |
|----|-----------|
| 1  | Moreira   |
| 2  | Mediratta |
| 3  | Mejia     |
| 4  | Mehra     |
| 5  | Montilla  |
| 6  | Matsukawa |
| 7  | Matsuda   |
| 8  | Masamura  |
| 9  | Ma        |

At the bottom of the interface, a status bar indicates 'Total rows: 9 of 9' and 'Query complete 00:00:00.071'.



```
1 SELECT course.title
2 FROM section,
3      course
4 WHERE section.course_id = course.course_id
5        AND section.semester = 'Fall'
6        AND section.year = 2009
7        AND course.dept_name like '%Eng.';
```





```
1 WITH take_more_than_3 AS (  
2     SELECT id,  
3         course_id  
4     FROM takes  
5     GROUP BY id,  
6         course_id  
7     HAVING count(*) >= 3  
8 )  
9 SELECT student.name,  
10    course.title  
11 FROM student,  
12    take_more_than_3,  
13    course  
14 WHERE student.id = take_more_than_3.id  
15    AND course.course_id = take_more_than_3.course_id;
```

The screenshot shows the pgAdmin 4 interface. The query window displays the SQL query, and the Data Output window shows the results of the query. The results are as follows:

|    | name        | title            |
|----|-------------|------------------|
| 1  | Preuss      | Embedded Systems |
| 2  | Date        | Embedded Systems |
| 3  | Aiken       | Embedded Systems |
| 4  | Noga        | Embedded Systems |
| 5  | Karande     | Embedded Systems |
| 6  | Prabhakaran | Embedded Systems |
| 7  | Hakkinen    | Embedded Systems |
| 8  | Drig        | Embedded Systems |
| 9  | Harshman    | Embedded Systems |
| 10 | Won         | Embedded Systems |

Total rows: 10 of 10    Query complete 00:00:00.104    Ln 15, Col 55



```
1 SELECT prereq_id AS course_id,  
2       sum(credits) AS sum_credits  
3 FROM prereq,  
4       course  
5 WHERE prereq.course_id = course.course_id  
6 GROUP BY prereq_id  
7 HAVING sum(credits) > 4  
8 ORDER BY sum_credits DESC;
```

The screenshot shows the pgAdmin 4 web interface. The query editor on the left contains the SQL query. The 'Data Output' tab on the right displays the results of the query execution. The results are shown in a table with two columns: 'course\_id' and 'sum\_credits'. The table contains 11 rows of data, sorted by 'sum\_credits' in descending order. The status bar at the bottom indicates 'Total rows: 20 of 20' and 'Query complete 00:00:00.076'.

|    | course_id<br>character varying (8) | sum_credits<br>numeric |
|----|------------------------------------|------------------------|
| 1  | 852                                | 11                     |
| 2  | 130                                | 10                     |
| 3  | 818                                | 8                      |
| 4  | 403                                | 8                      |
| 5  | 696                                | 8                      |
| 6  | 947                                | 7                      |
| 7  | 416                                | 7                      |
| 8  | 774                                | 7                      |
| 9  | 352                                | 7                      |
| 10 | 694                                | 7                      |
| 11 | 864                                | 7                      |



```
1 SELECT section.room_number
2 FROM section,
3     time_slot
4 WHERE section.time_slot_id = time_slot.time_slot_id
5     AND section.semester = 'Spring'
6     AND section.year = 2008
7 GROUP BY section.room_number
8 HAVING sum(time_slot.end_hr - time_slot.start_hr) >= 2;
```

The screenshot shows the pgAdmin 4 web interface. The top navigation bar includes 'File', 'Object', 'Tools', and 'Help'. The main toolbar contains icons for various database actions. The 'Query' tab is active, displaying the following SQL query:

```
1 SELECT section.room_number
2 FROM section,
3     time_slot
4 WHERE section.time_slot_id = time_slot.time_slot_id
5     AND section.semester = 'Spring'
```

The 'Data Output' tab shows the results of the query in a table with two rows:

| room_number |
|-------------|
| 707         |
| 812         |

The status bar at the bottom indicates 'Total rows: 2 of 2' and 'Query complete 00:00:00.127'.





```
1 WITH avg_count AS (  
2     SELECT avg(count) AS avg  
3     FROM (  
4         SELECT count(*)  
5         FROM teaches  
6         WHERE year = 2003  
7         GROUP BY id  
8     ) AS course_count  
9 )  
10 SELECT teaches.id,  
11     count(*)  
12 FROM teaches,  
13     avg_count  
14 WHERE teaches.year = 2003  
15 GROUP BY teaches.id  
16 HAVING count(*) < avg(avg_count.avg);
```

The screenshot shows the pgAdmin 4 interface. The top bar indicates the connection is to 'University/postgres@postgres'. The left sidebar shows the 'Servers' tree. The main pane displays the 'Query' tab with the following SQL code:

```
1 WITH avg_count AS (  
2     SELECT avg(count) as avg  
3     FROM (  
4         SELECT count(*)  
5         FROM teaches
```

Below the query editor, the 'Data Output' tab shows the results of the query. The results are displayed in a table with two columns: 'id' (character varying (5)) and 'count' (bigint). There are 5 rows of data.

|   | id    | count |
|---|-------|-------|
| 1 | 77346 | 1     |
| 2 | 28097 | 1     |
| 3 | 41930 | 1     |
| 4 | 6569  | 1     |
| 5 | 28400 | 1     |

The status bar at the bottom indicates 'Total rows: 5 of 5' and 'Query complete 00:00:00.128'.



```
1 SELECT distinct course_id
2 FROM section,
3      time_slot
4 WHERE section.building = 'Taylor'
5        AND section.year = 2007
6        AND time_slot.time_slot_id = section.time_slot_id
7        AND time_slot.start_hr BETWEEN 8 and 12;
```

The screenshot shows the pgAdmin 4 web interface. The query editor contains the following SQL query:

```
3      time_slot
4 WHERE section.building = 'Taylor'
5        AND section.year = 2007
6        AND time_slot.time_slot_id = section.time_slot_id
7        AND time_slot.start_hr BETWEEN 8 and 12;
```

The query results are displayed in the Data Output tab, showing two rows of data:

| course_id |
|-----------|
| 408       |
| 631       |

The status bar at the bottom indicates "Total rows: 2 of 2" and "Query complete 00:00:00.063".



```
1 WITH passed_course AS (  
2     SELECT takes.id,  
3           sum(course.credits) as credits  
4     FROM takes,  
5           course  
6     WHERE takes.course_id = course.course_id  
7           AND (  
8               grade like 'A%'  
9               OR grade like 'B%'  
10          )  
11     GROUP BY takes.id  
12 )  
13 SELECT student.name,  
14        passed_course.credits  
15 FROM student,  
16        passed_course  
17 WHERE passed_course.id = student.id;
```

The screenshot shows the pgAdmin 4 interface with the query executed. The 'Data Output' tab is active, displaying a table with two columns: 'name' and 'credits'. The table contains 11 rows of data. The status bar at the bottom indicates 'Total rows: 1000 of 2000' and 'Query complete 00:00:00.116'.

|    | name        | credits |
|----|-------------|---------|
| 1  | Scibili     | 21      |
| 2  | Towsey      | 34      |
| 3  | Hameed      | 25      |
| 4  | Kocsis      | 30      |
| 5  | Frolova     | 40      |
| 6  | Coppens     | 44      |
| 7  | Ralev       | 53      |
| 8  | Katehakis   | 16      |
| 9  | Hayrapetyan | 50      |
| 10 | Richter     | 28      |
| 11 | Legendijk   | 35      |



۵

۱.۵

لیست دپارتمان‌هایی که مجموع حقوق اساتیدشان از میانگین مجموع حقوق اساتید همه دپارتمان‌ها بیشتر است.

```
1 SELECT dept_name
2 FROM instructor
3 GROUP BY dept_name
4 HAVING sum(salary) >(
5     SELECT avg(total_salary_sum)
6     FROM (
7         SELECT sum(salary) AS total_salary_sum
8         FROM instructor
9         GROUP BY dept_name
10    ) AS dept_total
11 );
```

The screenshot shows the pgAdmin 4 web interface. The query editor contains the following SQL code:

```
7 SELECT sum(salary) AS total_salary_sum
8 FROM instructor
9 GROUP BY dept_name
10 ) AS dept_total
11 );
```

The 'Data Output' tab shows the results of the query, which are the department names:

| dept_name     |
|---------------|
| 1 Pol. Sci.   |
| 2 Marketing   |
| 3 English     |
| 4 Cybernetics |
| 5 Physics     |
| 6 Athletics   |
| 7 Statistics  |
| 8 Elec. Eng.  |

The status bar at the bottom indicates 'Total rows: 8 of 8' and 'Query complete 00:00:00.107'.



## ۲.۵

لیست اساتید و تعداد دروس ارائه شده آن‌ها در سال ۲۰۰۳ که در این سال بیشتر از میانگین درس ارائه داده‌اند.

```
1 WITH
2 instructor_teach_count(name, teach_count) AS (
3     SELECT instructor.name, COUNT(*)
4     FROM teaches,
5          instructor
6     WHERE teaches.id = instructor.id
7           AND year = 2003
8     GROUP BY instructor.id
9 ),
10 instructor_teach_count_avg(avg_value) AS (
11     SELECT avg(teach_count)
12     FROM instructor_teach_count
13 )
14 SELECT name, teach_count
15 FROM instructor_teach_count, instructor_teach_count_avg
16 WHERE teach_count > instructor_teach_count_avg.avg_value;
```

The screenshot shows the pgAdmin 4 interface. The query editor contains the following SQL code:

```
12     FROM instructor_teach_count
13 )
14 SELECT name, teach_count
15 FROM instructor_teach_count, instructor_teach_count_avg
16 WHERE teach_count > instructor_teach_count_avg.avg_value;
```

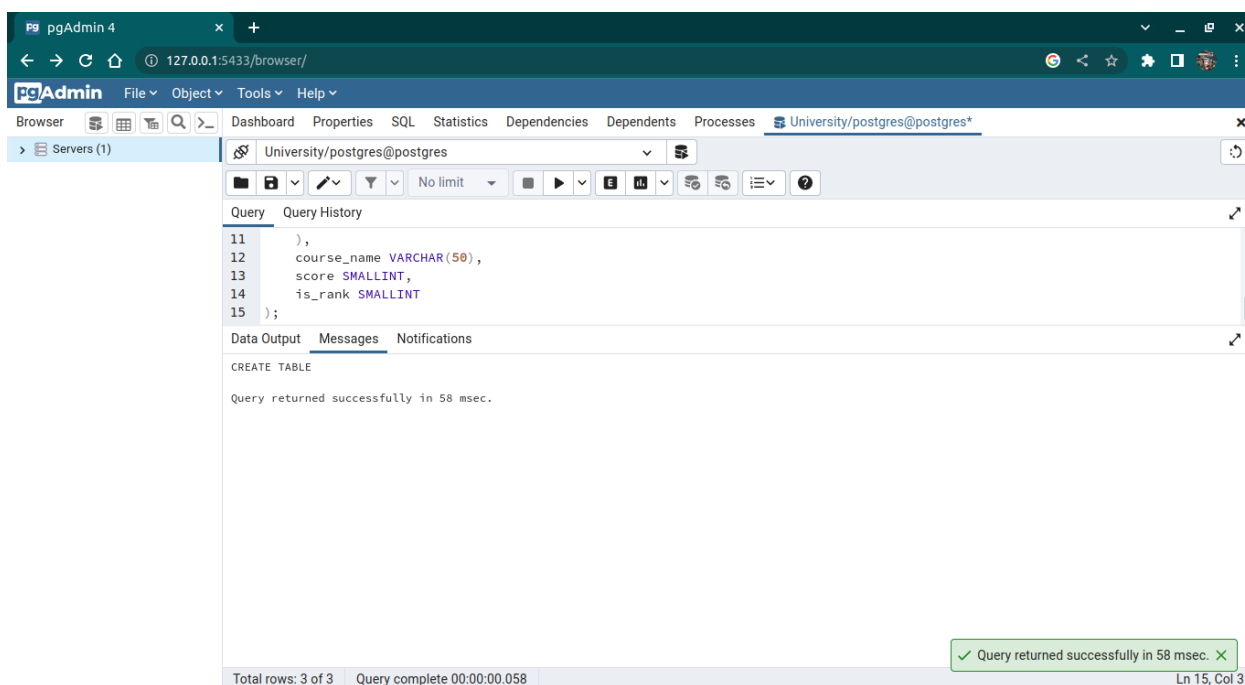
The Data Output tab shows the following results:

|   | name<br>character varying (20) | teach_count<br>bigint |
|---|--------------------------------|-----------------------|
| 1 | Ullman                         | 2                     |
| 2 | Dale                           | 3                     |
| 3 | DAgostino                      | 2                     |

Total rows: 3 of 3    Query complete 00:00:00.062    Ln 16, Col 56

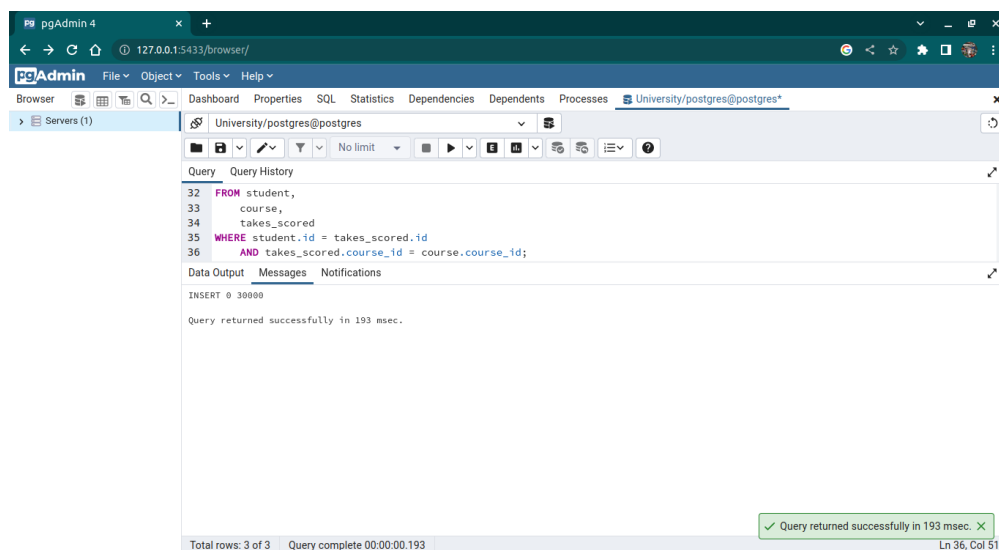


```
1 CREATE TABLE uni_data (  
2     stu_id VARCHAR(5),  
3     stu_name VARCHAR(20) NOT NULL,  
4     stu_dept_name VARCHAR(20),  
5     year NUMERIC(4, 0) CHECK (  
6         year > 1701  
7         AND year < 2100  
8     ),  
9     semester VARCHAR(6) CHECK (  
10        semester IN ('Fall', 'Winter', 'Spring', 'Summer')  
11    ),  
12    course_name VARCHAR(50),  
13    score SMALLINT,  
14    is_rank SMALLINT  
15 );
```



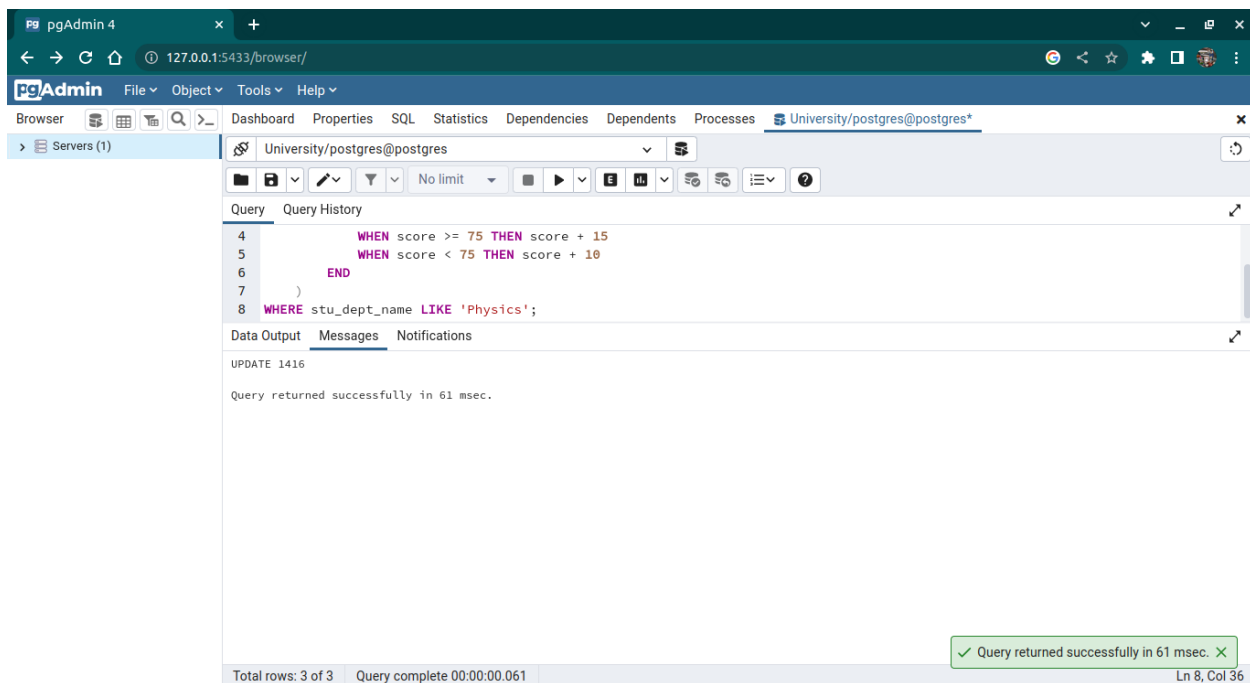


```
1  INSERT INTO uni_data WITH takes_scored AS (  
2      SELECT *,  
3      (  
4          CASE  
5              WHEN takes.grade LIKE 'A+' THEN 100  
6              WHEN takes.grade LIKE 'A' THEN 95  
7              WHEN takes.grade LIKE 'A-' THEN 90  
8              WHEN takes.grade LIKE 'B+' THEN 85  
9              WHEN takes.grade LIKE 'B' THEN 80  
10             WHEN takes.grade LIKE 'B-' THEN 75  
11             WHEN takes.grade LIKE 'C+' THEN 70  
12             WHEN takes.grade LIKE 'C' THEN 65  
13             WHEN takes.grade LIKE 'C-' THEN 60  
14             ELSE 0  
15         END  
16     ) AS score  
17     FROM takes  
18 )  
19 SELECT student.id AS stu_id,  
20        student.name AS stu_name,  
21        student.dept_name AS stu_dept_name,  
22        takes_scored.year AS year,  
23        takes_scored.semester AS semester,  
24        course.title AS course_name,  
25        takes_scored.score AS score,  
26        (  
27            CASE  
28                WHEN takes_scored.score > 70 THEN 1  
29                ELSE 0  
30            END  
31        ) AS is_rank  
32 FROM student,  
33        course,  
34        takes_scored  
35 WHERE student.id = takes_scored.id  
36        AND takes_scored.course_id = course.course_id;
```





```
1 UPDATE uni_data
2 SET score = (
3     CASE
4         WHEN score >= 75 THEN score + 15
5         WHEN score < 75 THEN score + 10
6     END
7 )
8 WHERE stu_dept_name LIKE 'Physics';
```







```
1 DELETE FROM uni_data AS u
2 WHERE u.stu_name LIKE 'T%'
3 AND u.score > (
4     SELECT avg(score)
5     FROM uni_data AS t
6     GROUP BY stu_dept_name
7     HAVING t.stu_dept_name = u.stu_dept_name
8 );
```

