

students 1 departments 2 students 3 Results 4 ×

```
SQL> SELECT CONCAT(first_name, ' ', last_name) AS full_name FROM Students WHERE
```

```
*<SMS_AZIZ> Script × SMS_AZIZ.erd
SELECT
    s.student,
    s.first_name,
    s.last_name,
    d.department_name,
    d.budget
FROM
    Students s
JOIN
    Departments d ON s.department_id = d.department_id
WHERE
    d.budget > 1000000;
```

```
● UPDATE Students  
SET DecadeStatus = '2000s'  
WHERE admission_year BETWEEN 2000 AND 2010;
```

<SMS\_AZIZ> Script ×

SMS\_AZIZ.erd

• DELETE FROM  
Students  
WHERE  
student NOT IN (SELECT DISTINCT student\_id FROM Enrollments);

Statistics 1

Statistics 2 ×

| Name         | Value  |
|--------------|--|
| Updated Rows | 15   |
| Query        | DELETE FROM<br>Students<br>WHERE<br>student NOT IN (SELECT DISTINCT student_id FROM Enrollments) |
| Start time   | Tue Nov 19 05:35:32 BDT 2024   |
| Finish time  | Tue Nov 19 05:35:32 BDT 2024   |

\*<SMS\_AZIZ> Script × SMS\_AZIZ.erd

Statistics 1

students 2 ×

```
SQL> SELECT s1.student AS student_id_1, CONCAT(s1.first_name, ' ', s1.last_name) AS full_name FROM students s1
```









```

● SELECT
    student,
    first_name,
    last_name,
    admission_year
FROM
    Students
WHERE
    admission_year % 2 = 0;

```

Results 1   students 2   students 3   **students 4**

Sheet 1 student first\_name last\_name admission\_year Filter students by SQL Enter a SQL expression to filter results (use Ctrl+Space)

|    | ID# student | A-Z first_name | A-Z last_name | ID# admission_year |
|----|-------------|----------------|---------------|--------------------|
| 1  | 1           | John           | Smith         | 2,022              |
| 2  | 2           | Alice          | Turner        | 2,024              |
| 3  | 3           | Emily          | Davis         | 2,024              |
| 4  | 4           | Michael        | Brown         | 2,022              |
| 5  | 6           | David          | Martinez      | 2,022              |
| 6  | 7           | Jessica        | Anderson      | 2,024              |
| 7  | 9           | Olivia         | Taylor        | 2,022              |
| 8  | 10          | Ethan          | White         | 2,024              |
| 9  | 11          | Emma           | Lee           | 2,022              |
| 10 | 13          | Madison        | Clark         | 2,024              |
| 11 | 14          | Benjamin       | King          | 2,022              |
| 12 | 16          | Ethan          | Scott         | 2,024              |
| 13 | 17          | Mia            | Young         | 2,022              |
| 14 | 19          | Ava            | Hernandez     | 2,024              |
| 15 | 20          | William        | Adams         | 2,022              |

Results 1 students 2 students 3 students 4 courses 5

SQL: `SELECT c.course_id, c.course_name, COUNT(r.student_id) AS student_count FROM courses c JOIN students r ON c.course_id = r.course_id` Enter a SQL expression to filter results

|   | course_id | course_name                 | student_count |
|---|-----------|-----------------------------|---------------|
| 1 | 101       | Introduction to Programming | 1             |
| 2 | 102       | Digital Logic Design        | 1             |
| 3 | 103       | Calculus I                  | 1             |
| 4 | 104       | Shakespearean Tragedies     | 1             |
| 5 | 105       | American Revolution         | 1             |

Script SMS\_AZIZ.erd

```
SELECT
  s.first_name,
  s.last_name,
  CASE
    WHEN COUNT(e.course_id) > 3 THEN 'Heavy'
    ELSE 'Light'
  END AS CourseStatus
FROM
  Students s
LEFT JOIN
  Enrollments e ON s.student = e.student_id
GROUP BY
  s.student, s.first_name, s.last_name;
```

Results 1students 2students 3students 4courses 5students 6

SELECT s.first\_name, s.last\_name, CASE WHEN COUNT(e.course\_id) > 3 THEN 'Heavy' ELSE 'Light' END AS CourseStatus

|    | first_name | last_name | CourseStatus |
|----|------------|-----------|--------------|
| 1  | John       | Smith     | Light        |
| 2  | Alice      | Turner    | Light        |
| 3  | Emily      | Davis     | Light        |
| 4  | Michael    | Brown     | Light        |
| 5  | Sarah      | Wilson    | Light        |
| 6  | David      | Martinez  | Light        |
| 7  | Jessica    | Anderson  | Light        |
| 8  | Ryan       | Thomas    | Light        |
| 9  | Olivia     | Taylor    | Light        |
| 10 | Ethan      | White     | Light        |
| 11 | Emma       | Lee       | Light        |
| 12 | Alexander  | Harris    | Light        |
| 13 | Madison    | Clark     | Light        |
| 14 | Benjamin   | King      | Light        |
| 15 | Sophia     | Rodriguez | Light        |
| 16 | Ethan      | Scott     | Light        |
| 17 | Mia        | Young     | Light        |
| 18 | Jacob      | Hall      | Light        |
| 19 | Ava        | Hernandez | Light        |
| 20 | William    | Adams     | Light        |



students 1

departments 2

SQL

SELECT department\_name, MIN(earliest\_admission\_year) AS earliest\_admission\_year

Enter a SQL expression to filter results (use Ctrl+Space)

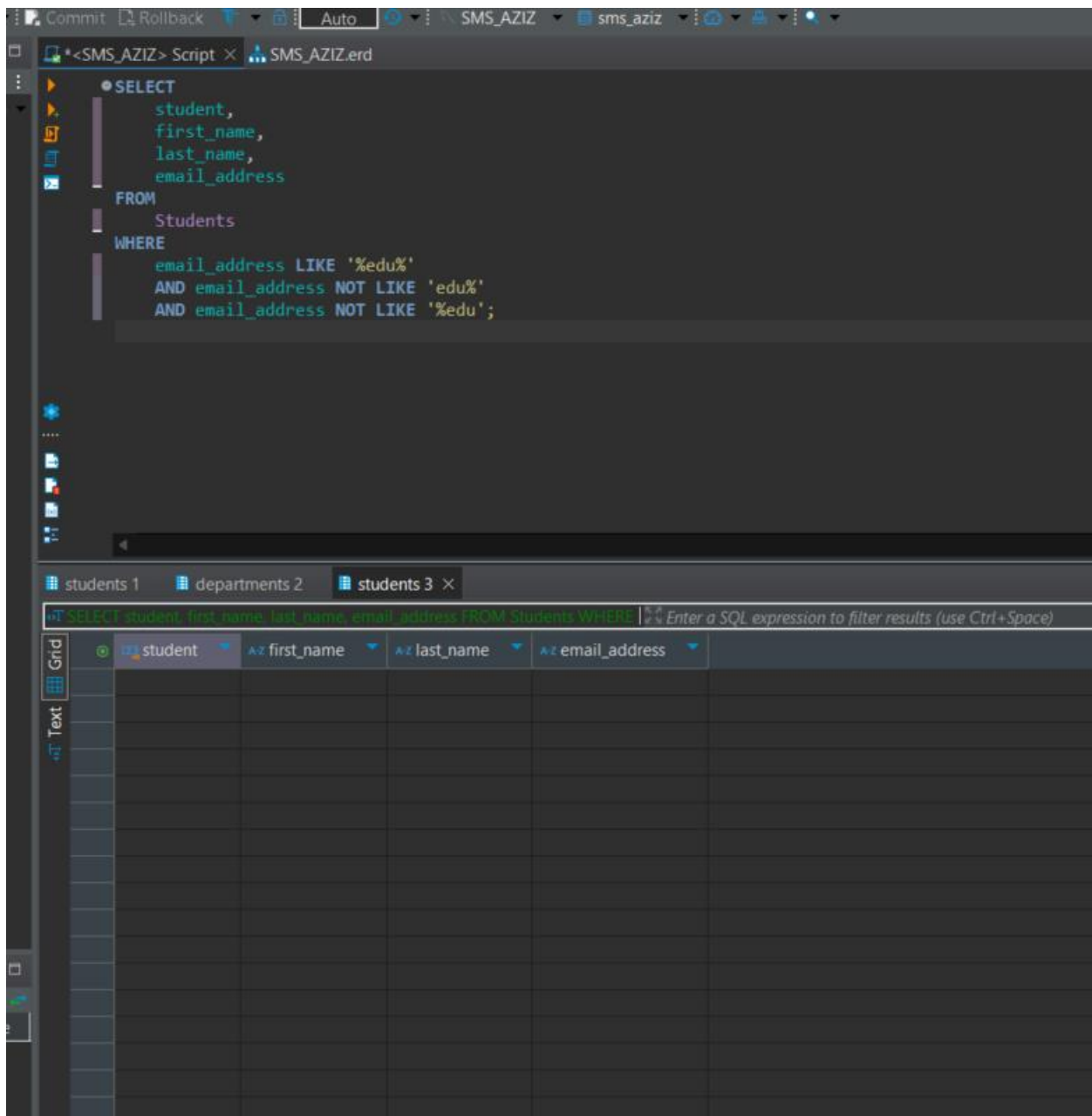
Grid

Text

department\_name

earliest\_admission\_year

|   |                        |       |
|---|------------------------|-------|
| 1 | Computer Science       | 2,022 |
| 2 | Electrical Engineering | 2,022 |
| 3 | Mathematics            | 2,022 |



The screenshot shows a database IDE with a dark theme. The top pane displays a SQL query:

```

SELECT
    c.course_id,
    c.course_name,
    COUNT(e.student_id) AS student_count
FROM
    Courses c
LEFT JOIN
    Enrollments e ON c.course_id = e.course_id
GROUP BY
    c.course_id, c.course_name
ORDER BY
    student_count DESC;

```

The bottom pane shows the results of the query in a table view. The table has three columns: course\_id, course\_name, and student\_count. The results are ordered by student\_count in descending order.

|   | course_id | course_name                 | student_count |
|---|-----------|-----------------------------|---------------|
| 1 | 101       | Introduction to Programming | 1             |
| 2 | 102       | Digital Logic Design        | 1             |
| 3 | 103       | Calculus I                  | 1             |
| 4 | 104       | Shakespearean Tragedies     | 1             |
| 5 | 105       | American Revolution         | 1             |





The screenshot shows a database IDE with a SQL query editor and a results pane. The query is as follows:

```
SELECT
    d.department_name,
    COUNT(s.student) AS student_count
FROM
    Departments d
JOIN
    Students s ON d.department_id = s.department_id
GROUP BY
    d.department_name
ORDER BY
    student_count DESC
LIMIT 5;
```

The results pane displays the following data:

|   | department_name        | student_count |
|---|------------------------|---------------|
| 1 | Computer Science       | 7             |
| 2 | Electrical Engineering | 7             |
| 3 | Mathematics            | 6             |

[illegible]

Script SMS\_AZIZ.erd

```
SELECT
  s.first_name,
  s.last_name,
  CASE
    WHEN COUNT(e.course_id) > 3 THEN 'Heavy'
    ELSE 'Light'
  END AS CourseStatus
FROM
  Students s
LEFT JOIN
  Enrollments e ON s.student = e.student_id
GROUP BY
  s.student, s.first_name, s.last_name;
```

Results 1students 2students 3students 4courses 5students 6

SELECT s.first\_name, s.last\_name, CASE WHEN COUNT(e.course\_id) > 3 THEN 'Heavy' ELSE 'Light' END AS CourseStatus

|    | first_name | last_name | CourseStatus |
|----|------------|-----------|--------------|
| 1  | John       | Smith     | Light        |
| 2  | Alice      | Turner    | Light        |
| 3  | Emily      | Davis     | Light        |
| 4  | Michael    | Brown     | Light        |
| 5  | Sarah      | Wilson    | Light        |
| 6  | David      | Martinez  | Light        |
| 7  | Jessica    | Anderson  | Light        |
| 8  | Ryan       | Thomas    | Light        |
| 9  | Olivia     | Taylor    | Light        |
| 10 | Ethan      | White     | Light        |
| 11 | Emma       | Lee       | Light        |
| 12 | Alexander  | Harris    | Light        |
| 13 | Madison    | Clark     | Light        |
| 14 | Benjamin   | King      | Light        |
| 15 | Sophia     | Rodriguez | Light        |
| 16 | Ethan      | Scott     | Light        |
| 17 | Mia        | Young     | Light        |
| 18 | Jacob      | Hall      | Light        |
| 19 | Ava        | Hernandez | Light        |
| 20 | William    | Adams     | Light        |

•<SMS\_AZIZ> Script × SMS\_AZIZ.erd

SELECT

d.department\_name,

MIN(s.admission\_year) AS earliest\_admission\_year

FROM

Departments d

JOIN

Students s ON d.department\_id = s.department\_id

GROUP BY

d.department\_name;

students 1 departments 2 ×

SELECT d.department\_name, MIN(s.admission\_year) AS earliest\_admission\_year | Enter a SQL expression to filter results (use Ctrl+Space)

Grid

1 Computer Science 2,022

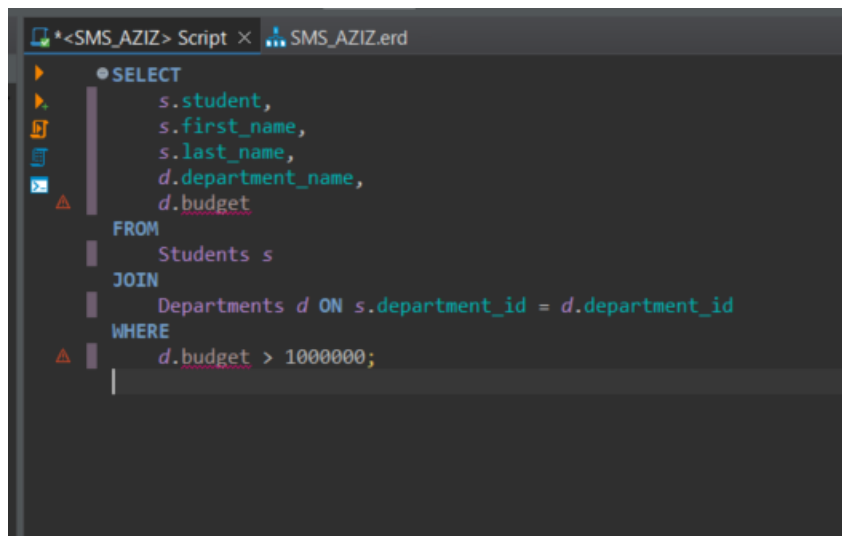
2 Electrical Engineering 2,022

3 Mathematics 2,022



students 1 departments 2 students 3 Results 4 ×

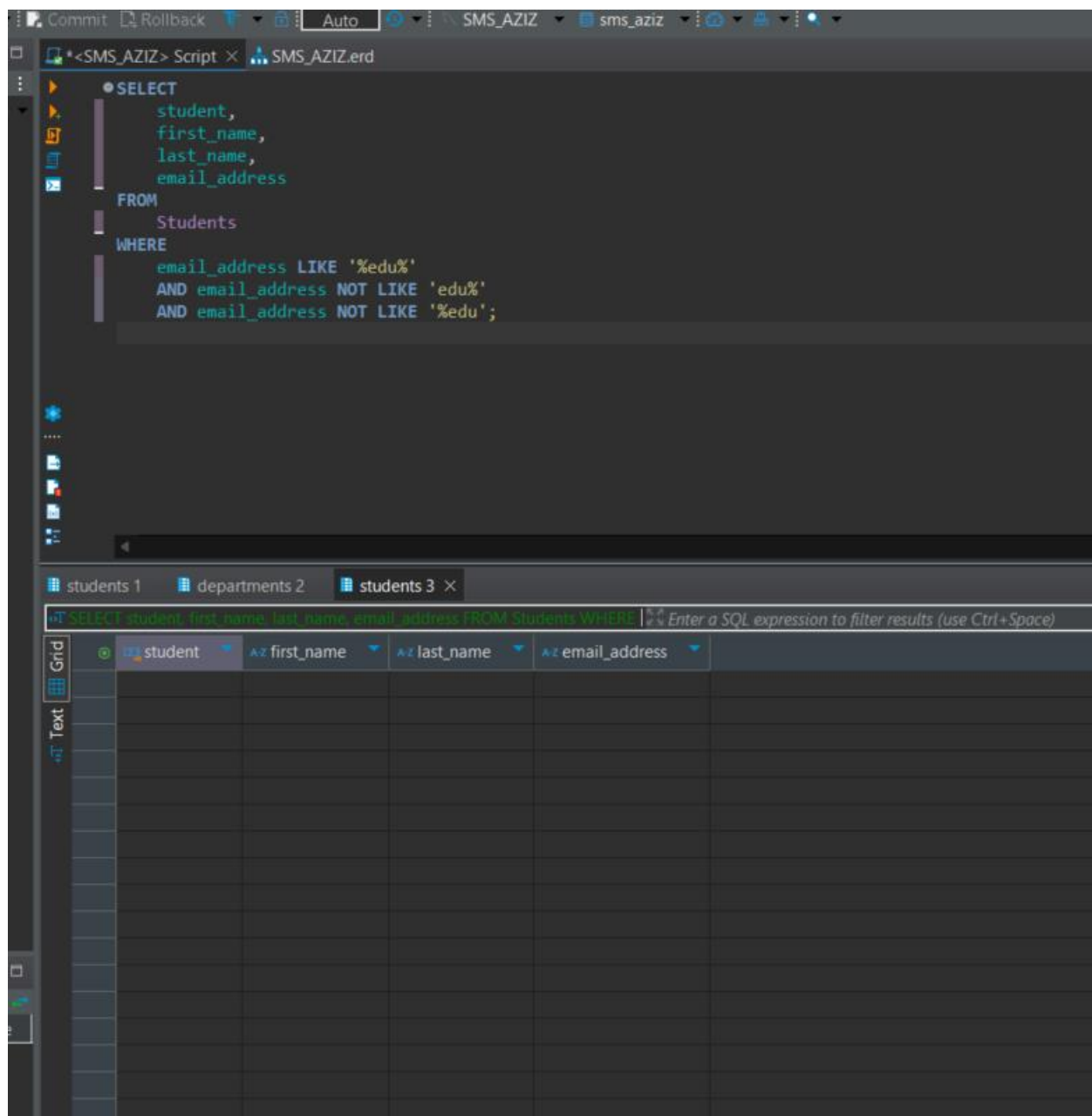
```
SQL> SELECT CONCAT(first_name, ' ', last_name) AS full_name FROM Students WHERE
```



The screenshot shows a SQL script editor with two tabs: "<SMS\_AZIZ> Script" and "SMS\_AZIZ.erd". The script contains a SQL query with the following structure:

```
● SELECT
  s.student,
  s.first_name,
  s.last_name,
  d.department_name,
  d.budget
FROM
  Students s
JOIN
  Departments d ON s.department_id = d.department_id
WHERE
  d.budget > 1000000;
```

The query selects columns from the 'Students' table (s) and the 'Departments' table (d). The columns selected are s.student, s.first\_name, s.last\_name, d.department\_name, and d.budget. The tables are joined on the condition s.department\_id = d.department\_id. The WHERE clause filters for departments with a budget greater than 1000000.







✕

<SMS\_AZIZ> Script

✕

SMS\_AZIZ.erd

▶

▶

📄

🔍

🔗

•

DELETE FROM

Students

WHERE

student NOT IN (SELECT DISTINCT student\_id FROM Enrollments);

⚙️

....

📄

🔍

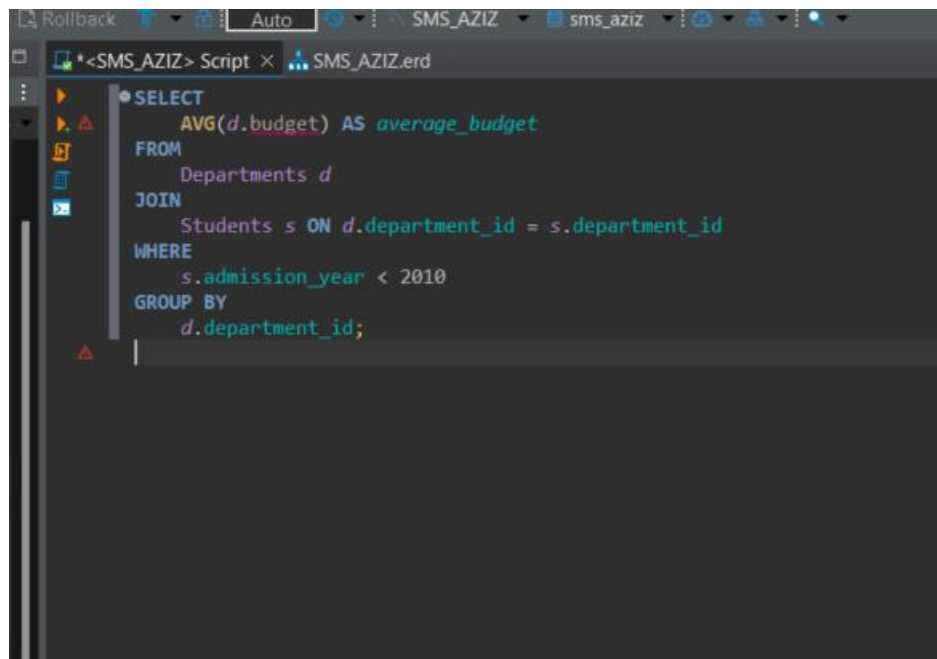
🔗

◀

📊 Statistics 1

📊 Statistics 2 ✕

| Name         | Value  |
|--------------|--|
| Updated Rows | 15   |
| Query        | DELETE FROM<br>Students<br>WHERE<br>student NOT IN (SELECT DISTINCT student_id FROM Enrollments) |
| Start time   | Tue Nov 19 05:35:32 BDT 2024   |
| Finish time  | Tue Nov 19 05:35:32 BDT 2024   |



The screenshot shows a database management tool interface. At the top, there is a toolbar with buttons for 'Rollback', 'Auto', and 'SMS\_AZIZ'. Below the toolbar, there is a tab labeled '\*<SMS\_AZIZ> Script' and another tab labeled 'SMS\_AZIZ.erd'. The main area displays a SQL script with the following text:

```
SELECT
  AVG(d.budget) AS average_budget
FROM
  Departments d
JOIN
  Students s ON d.department_id = s.department_id
WHERE
  s.admission_year < 2010
GROUP BY
  d.department_id;
```

The script is color-coded: 'SELECT' is in blue, 'AVG(d.budget)' is in orange, 'AS average\_budget' is in green, 'FROM' is in blue, 'Departments d' is in purple, 'JOIN' is in blue, 'Students s' is in purple, 'ON d.department\_id = s.department\_id' is in purple, 'WHERE' is in blue, 's.admission\_year < 2010' is in purple, 'GROUP BY' is in blue, and 'd.department\_id;' is in purple.

```
SELECT s1.student AS student_id_1, CONCAT(s1.first_name, ' ', s1.last_name
```

[illegible]

