

In this task, I use Google BigQuery to run SQL queries and create dataset named vidio and create tables from the file given.

--- Step 1. Left join student table with classroom table, and save this new table as joined\_table ---

```
SELECT
s.student_id,
c.classroom_id,
s.name,
s.age
FROM `vidio.student` AS s
LEFT JOIN `vidio.classroom` AS c
ON s.student_id = c.student_id
```

--- 1a. Student who doesn't have classroom ---

```
SELECT student_id, classroom_id, name
FROM `vidio.joined_table`
WHERE classroom_id IS NULL
```

### Answer: Based on the two tables, every classroom\_id has students, but there is 1 student\_id which has no class and his name is Chip with student\_id = 3.

--- 1b. Display classroom\_id for each student ---

```
SELECT student_id, classroom_id, name
FROM `vidio.joined_table`
ORDER BY student_id
```

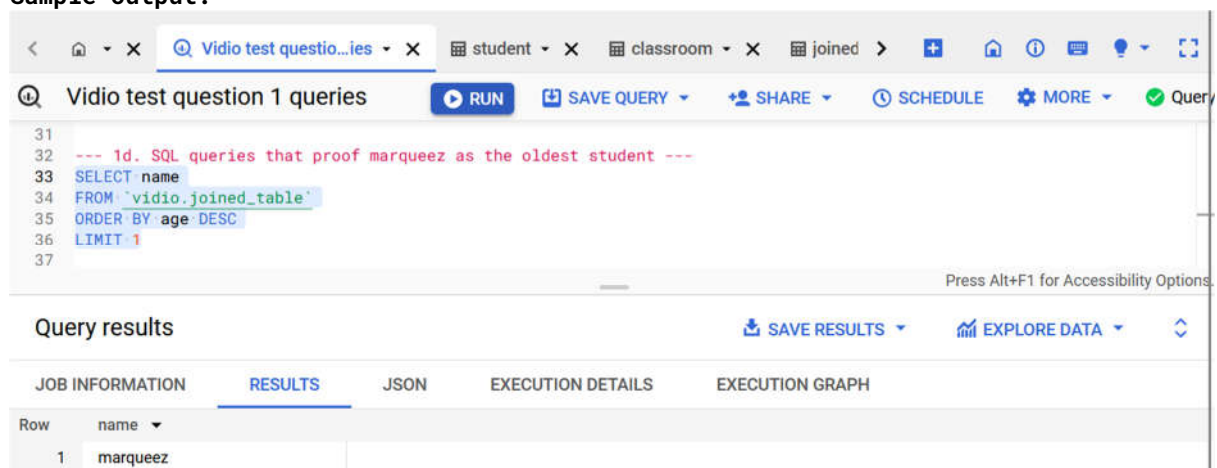
---1c. Create age segmentation for each student ---

```
SELECT
CASE
    WHEN age > 15 THEN 'high school'
    WHEN age < 16 THEN 'middle school'
END AS school_group,
COUNT(*) AS num_of_students
FROM `vidio.joined_table`
GROUP BY 1
```

--- 1d. SQL queries that proof marqueez as the oldest student ---

```
SELECT name
FROM `vidio.joined_table`
ORDER BY age DESC
LIMIT 1
```

Sample output:



The screenshot shows the Google BigQuery web interface. At the top, there's a navigation bar with tabs for 'student', 'classroom', and 'joined'. Below this, the query editor displays the SQL query for finding the oldest student (Query 1d). The query is: 

```
SELECT name
FROM `vidio.joined_table`
ORDER BY age DESC
LIMIT 1
```

 Below the query editor, the 'Query results' section is visible, showing a table with one row: 

| Row | name     |
|-----|----------|
| 1   | marqueez |

 The interface also includes buttons for 'RUN', 'SAVE QUERY', 'SHARE', 'SCHEDULE', and 'MORE'.

```

--- 1e. Find out the cumulative age from the students table for each student record ---
SELECT SUM(age) AS cumulative_age
FROM `vidio.joined_table`

```

**Sample output:**

The screenshot shows a SQL query editor interface. The query is as follows:

```

--- 1e. Find out the cumulative age from the students table for each student record ---
SELECT SUM(age) AS cumulative_age
FROM `vidio.joined_table`

```

Below the query editor, the 'Query results' section is visible. It has tabs for 'JOB INFORMATION', 'RESULTS', 'JSON', 'EXECUTION DETAILS', and 'EXECUTION GRAPH'. The 'RESULTS' tab is selected, showing a table with one row:

| Row | cumulative_age |
|-----|----------------|
| 1   | 59             |

--- 1f. Summary of the tables (classroom and student tables) ---

### From the classroom and student tables, we can find that there is a student who doesn't belong to any classroom, this can be related to missing data, so we can re-check again to find the correct data.

### These tables also provide information about the number of students enrolling in each class along with the names and age of the students.

### Questions that could be created from the tables are:

### 1. What is the age average of all students in the table?

### 2. How many students enrolled in each class and which class has the highest number of students?

### 3. Who is the youngest student and which class does he/she belong to?