**Step 1: Create the department table**

This table contains department information.

sql

CopyEdit

CREATE TABLE department (

id BIGSERIAL PRIMARY KEY,

department\_name VARCHAR(50) NOT NULL

);

**Step 2: Create the student table**

This table contains student details and a foreign key to the department table.

sql

CopyEdit

CREATE TABLE student (

std\_id BIGSERIAL PRIMARY KEY,

f\_name VARCHAR(50) NOT NULL,

l\_name VARCHAR(50) NOT NULL,

gender VARCHAR(50) NOT NULL,

marks INT NOT NULL,

department\_id INT REFERENCES department(id) -- Foreign Key to department

);

**Step 3: Insert sample data into the department table**

sql

CopyEdit

INSERT INTO department (department\_name)

VALUES

('IT'),

('Science'),

('Commerce');

**Step 4: Insert sample data into the student table**

sql

CopyEdit

INSERT INTO student (f\_name, l\_name, gender, marks, department\_id)

VALUES

('muhd', 'Yaseen', 'male', 50, 1),

('thanvi', 'thasni', 'female', 90, 2),

('muhd', 'shamil', 'male', 30, 1),

('risvana', 'sherin', 'female', 98, 2),

('rahul', 'iswar', 'male', 84, 3),

('muhammed', 'shahin', 'male', 99, 2),

('Eid', 'bilal', 'male', 72, 3);

**Step 5: Query to count total marks per department**

Now, here’s the query to count the total marks of students in each department.

sql

CopyEdit

SELECT

d.department\_name,

SUM(s.marks) AS total\_marks

FROM

student s

JOIN

department d

ON

s.department\_id = d.id

GROUP BY

d.department\_name;

**Expected Output:**

| **department\_name** | **total\_marks** |
| --- | --- |
| IT | 80 |
| Science | 188 |
| Commerce | 156 |

**Explanation:**

* **JOIN**: This combines the student table with the department table using the department\_id field.
* **SUM(s.marks)**: Adds up the marks for each student within each department.
* **GROUP BY d.department\_name**: Groups the result by department, so you get the total marks per department.

You can run the entire set of commands to see the result in your PostgreSQL database. Let me know if you need further assistance!

To enforce a constraint on the gender column so that it only allows values 'male' or 'female', you can use the CHECK constraint.

### Alter Existing Table:

sql

CopyEdit

ALTER TABLE student

ADD CONSTRAINT gender\_check

CHECK (gender IN ('male', 'female'));

This will ensure that only 'male' or 'female' can be inserted into the gender column.

You can concatenate f\_name and l\_name to create a full name in your query using CONCAT() or the || operator. Here's how you can modify your query:

### Using CONCAT():

sql

CopyEdit

SELECT

CONCAT(s.f\_name, ' ', s.l\_name) AS full\_name,

d.dep\_name,

s.marks

FROM student s

JOIN dep d ON s.dep\_id = d.id;

### Using || (PostgreSQL string concatenation operator):

sql

CopyEdit

SELECT

s.f\_name || ' ' || s.l\_name AS full\_name,

d.dep\_name,

s.marks

FROM student s

JOIN dep d ON s.dep\_id = d.id;

### If you want the total marks grouped by department:

sql

CopyEdit

SELECT

d.dep\_name,

STRING\_AGG(s.f\_name || ' ' || s.l\_name, ', ') AS full\_names,

SUM(s.marks) AS total\_marks

FROM student s

JOIN dep d ON s.dep\_id = d.id

GROUP BY d.dep\_name;

This will list all student names in each department along with their total marks.

To find the student with the highest marks, you can use the ORDER BY clause with LIMIT 1 or the MAX() function.

### Using ORDER BY and LIMIT:

sql

CopyEdit

SELECT

f\_name || ' ' || l\_name AS full\_name,

marks

FROM student

ORDER BY marks DESC

LIMIT 1;

### Using MAX() with JOIN:

sql

CopyEdit

SELECT

f\_name || ' ' || l\_name AS full\_name,

marks

FROM student

WHERE marks = (SELECT MAX(marks) FROM student);

The OFFSET clause is used in SQL queries to **skip** a specified number of rows before returning the result. It is commonly used with LIMIT for **pagination** or **skipping specific rows**.

### 🔹 ****Why Use**** OFFSET****?****

1. **Pagination** – Retrieve results page by page (useful for large datasets).
2. **Skipping Initial Rows** – Ignore a certain number of top results.
3. **Fetching Specific Records** – Get results starting from a certain row.

### 🔹 ****Example Usage****

#### ✅ Get the second-highest marks (skip the highest)

sql

CopyEdit

SELECT f\_name || ' ' || l\_name AS full\_name, marks

FROM student

ORDER BY marks DESC

LIMIT 1 OFFSET 1;

* ORDER BY marks DESC → Sorts students by marks (highest first).
* LIMIT 1 → Retrieves only **one** result.
* OFFSET 1 → Skips the **top** student, giving the **second-highest**.

#### ✅ Fetch students ranked **3rd to 5th**

sql

CopyEdit

SELECT f\_name || ' ' || l\_name AS full\_name, marks

FROM student

ORDER BY marks DESC

LIMIT 3 OFFSET 2;

* Skips **top 2** students and fetches the **next 3**.

If you want to add a new column place to the student table with a default value of "Kochi", use the following SQL query:

ALTER TABLE student ADD COLUMN place VARCHAR(50) DEFAULT 'Kochi';

**Explanation:**

* ALTER TABLE student → Modifies the student table.
* ADD COLUMN place VARCHAR(50) → Adds a new column named place with a VARCHAR(50) data type.
* DEFAULT 'Kochi' → Sets the default value of the column to "Kochi".

**Behavior:**

* If you **insert a new row** without specifying place, it will automatically be set to "Kochi".
* If you **update existing rows**, the new column will have NULL unless modified.

**If you want to set "Kochi" for existing records:**

UPDATE student SET place = 'Kochi' WHERE place IS NULL;

Let me know if you need any changes! 🚀

std=# UPDATE student SET place='Calicut' WHERE f\_name='muhd';

UPDATE 2

std=# SELECT \* FROM student;

f\_name | l\_name | gender | marks | dep\_id | place

----------+--------+--------+-------+--------+---------

thanvi | thasni | female | 90 | 2 | Kochi

risvana | sherin | female | 98 | 2 | Kochi

rahul | iswar | male | 84 | 3 | Kochi

muhammed | shahin | male | 99 | 2 | Kochi

Eid | bilal | male | 72 | 3 | Kochi

muhd | Yaseen | male | 50 | 1 | Calicut

muhd | shamil | male | 30 | 1 | Calicut