Class #12: Introduction to SQL - Part 2

1. Understanding different types of joins in SQL.

PostgreSQL Joins and Unions

1. PostgreSQL Joins

2. PostgreSQL INNER JOIN

3. PostgreSQL LEFT JOIN

4. PostgreSQL RIGHT JOIN

Advanced Queries

1. PostgreSQL GROUP BY

PostgreSQL Joins and Unions

PostgreSQL Joins এবং Unions হল দুটি গুরুত্বপূর্ণ SQL অপারেশন, যা একাধিক টেবিলের তথ্য একত্রিত করতে ব্যবহার করা হয়। তবে তাদের কাজের ধরন এবং ব্যবহার ভিন্ন।

2. PostgreSQL Unions:

SET এর Union এর মতন কাজ করে । unique value and sum of all value of two set.

Set A = {1, 2, 3, 4, 5}

Set B = {4, 5, 6, 7, 8}

Set A ∪ Set B = {1, 2, 3, 4, 5, 6, 7, 8}

UNION উদাহরণ:

ধরা যাক, দুটি টেবিল রয়েছে:

table1:

id	name
1	Alice
2	Bob

table2:

id	name
3	Charlie
4	David

```
SELECT name FROM table1 UNION SELECT name FROM table2;
```

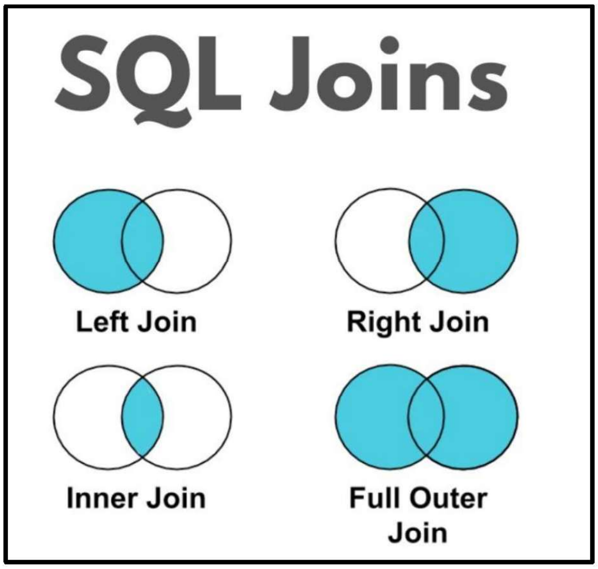
এখানে, UNION ব্যবহারের ফলে Alice, Bob, Charlie, এবং David এই চারটি ইউনিক নাম দেখানো হবে। যদি কোন নাম দুইটি টেবিলেই থাকে, তবে তা একবারই দেখানো হবে।

1. PostgreSQL Joins

1. PostgreSQL Joins:

PostgreSQL Joins হল এমন একটি SQL অপারেশন যা একাধিক টেবিলের মধ্যে সম্পর্ক স্থাপন করে এবং সেগুলির মধ্যে থাকা তথ্য একত্রিত করে। এই অপারেশনটির মাধ্যমে আমরা একাধিক টেবিলের মধ্যে সম্পর্কিত ডেটা একসাথে দেখতে বা বিশ্লেষণ করতে পারি।

Joins-এর প্রকার:



2. PostgreSQL INNER JOIN

PostgreSQL INNER JOIN হল এমন একটি SQL অপারেশন, যা দুটি টেবিলের মধ্যে মিল থাকা রেকর্ডগুলোকে একত্রিত করে।

Set A = {1, 2, 3, 4, 5}

Set B = {4, 5, 6, 7, 8}

Set A ∩ Set B = {4, 5}

উদাহরণ দিয়ে বুঝাই:

ধরা যাক, আমাদের কাছে দুটি টেবিল আছে:

1. employees টেবিল:

employee_id	name	department_id
1	John	101
2	Alice	102
3	Bob	103
4	Charlie	101

2. departments টেবিল:

department_id	department_name
101	HR
102	IT
104	Marketing

এখানে, employees টেবিলের department_id কলামটি এবং departments টেবিলের department_id কলামটি সম্পর্কিত।

INNER JOIN উদাহরণ:

আমরা যদি উভয় টেবিলের department_name এবং employee_name একসাথে দেখতে চাই, তাহলে আমরা INNER JOIN ব্যবহার করতে পারি।

```
SELECT employees.name, departments.department_name
FROM employees
INNER JOIN departments
ON employees.department_id = departments.department_id;
```

আউটপুট:

name	**department_name**
John	HR
Alice	IT
Charlie	HR

ব্যাখ্যা:

- INNER JOIN দুইটি টেবিলকে department_id কলামের উপর যুক্ত করেছে।
- John এবং Charlie উভয়ই HR বিভাগের সদস্য, তাই তারা উভয়কে আনা হয়েছে।
- Alice হল IT বিভাগের সদস্য, তাই তার তথ্যও অন্তর্ভুক্ত করা হয়েছে।
- কিন্তু Bob এর department_id = 103, কিন্তু সেই department_id departments টেবিলে নেই (এখানে কোনো "department_id 103" নেই), তাই Bob এর তথ্য আনা হয়নি।

অতএব, INNER JOIN কেবলমাত্র সেই রেকর্ডগুলি ফেরত দেয় যেগুলির মধ্যে সম্পর্ক আছে উভয় টেবিলের মধ্যে।

3. PostgreSQL LEFT JOIN

PostgreSQL LEFT JOIN (বা LEFT OUTER JOIN) হল একটি SQL অপারেশন, যা দুটি টেবিলের মধ্যে সম্পর্ক স্থাপন করে এবং বাম (first) টেবিলের সমস্ত রেকর্ড নিয়ে আসে।

Set A = {1, 2, 3, 4, 5}

Set B = {4, 5, 6, 7, 8}

Set A = {1, 2, 3, 4, 5}

LEFT JOIN উদাহরণ:

ধরা যাক, আমাদের দুটি টেবিল রয়েছে:

1. employees টেবিল:

employee_id	name	department_id
1	John	101
2	Alice	102
3	Bob	103
4	Charlie	101

2. departments টেবিল:

department_id	department_name
101	HR
102	IT
104	Marketing

এখানে, employees টেবিলের department_id কলামটি এবং departments টেবিলের department_id কলামটি সম্পর্কিত।

LEFT JOIN এর SQL Query:

আমরা যদি employees (বাম/first) টেবিলের সমস্ত employee_name এবং তাদের department_name দেখতে চাই, তাহলে LEFT JOIN ব্যবহার করবো:

```
SELECT employees.name, departments.department_name
FROM employees
LEFT JOIN departments
ON employees.department_id = departments.department_id;
```

আউটপুট:

name	**department_name**
John	HR
Alice	IT
Bob	NULL
Charlie	HR

ব্যাখ্যা:

- John এবং Charlie উভয়ই HR বিভাগের সদস্য, তাই তাদের নাম এবং বিভাগ প্রদর্শিত হয়েছে।
- Alice IT বিভাগের সদস্য, তাই তার নাম এবং বিভাগও দেখানো হয়েছে।
- কিন্তু Bob এর department_id 103, যা departments টেবিলে নেই (এখানে কোনো "department_id 103" নেই), তাই তার department_name কলামে NULL প্রদর্শিত হয়েছে।

Set A = {1, 2, 3, 4, 5} Set B = {4, 5, 6, 7, 8}	Set A = {1, 2, 3, 4, 5}

A সেটে না থাকলে NULL দেখাবে।

সারাংশ:

LEFT JOIN ব্যবহার করলে, আপনি বাম টেবিলের সমস্ত রেকর্ড পাবেন এবং যদি ডান টেবিলের সাথে সম্পর্কিত কোনো রেকর্ড না থাকে, তবে ডান টেবিলের কলামে NULL মান দেখাবে।

4. PostgreSQL RIGHT JOIN

PostgreSQL RIGHT JOIN (বা RIGHT OUTER JOIN) হল একটি SQL অপারেশন যা দুটি টেবিলের মধ্যে সম্পর্ক স্থাপন করে এবং ডান (second) টেবিলের সমস্ত রেকর্ড নিয়ে আসে, সাথে বাম (first) টেবিলের সম্পর্কিত রেকর্ডগুলোও দেখায়।

Set A = {1, 2, 3, 4, 5}

Set B = {4, 5, 6, 7, 8}

Set B = {4, 5, 6, 7, 8}

RIGHT JOIN উদাহরণ:

ধরা যাক, আমাদের কাছে দুটি টেবিল রয়েছে:

1. employees টেবিল:

employee_id	name	department_id
1	John	101
2	Alice	102
3	Bob	103
4	Charlie	101

2. departments টেবিল:

department_id	department_name
101	HR
102	IT
104	Marketing

এখানে, employees টেবিলের department_id কলামটি এবং departments টেবিলের department_id কলামটি সম্পর্কিত।

RIGHT JOIN এর SQL Query:

আমরা যদি departments (ডান/second) টেবিলের সমস্ত department_name এবং তাদের সাথে সম্পর্কিত employee_name দেখতে চাই, তাহলে RIGHT JOIN ব্যবহার করতে পারি:

```
SELECT employees.name, departments.department_name
FROM employees
RIGHT JOIN departments
ON employees.department_id = departments.department_id;
```

আউটপুট:

name	**department_name**
John	HR
Alice	IT
NULL	Marketing
Charlie	HR

ব্যাখ্যা:

- John এবং Charlie উভয়ই HR বিভাগের সদস্য, তাই তাদের নাম এবং বিভাগ দেখানো হয়েছে।
- Alice IT বিভাগের সদস্য, তাই তার নাম এবং বিভাগও দেখানো হয়েছে।
- কিন্তু Marketing বিভাগ (department_id 104) এর সাথে কোনো employee সম্পর্কিত নেই (কারণ employees টেবিলে department_id 104 নেই), তাই Marketing বিভাগের জন্য employee_name কলামে NULL দেখানো হয়েছে।

Set A = {1, 2, 3, 4, 5} Set B = {4, 5, 6, 7, 8}	Set B = {4, 5, 6, 7, 8}

B সেটে না থাকলে NULL দেখাবে।

সারাংশ:

RIGHT JOIN ব্যবহার করলে ডান (second) টেবিলের সমস্ত রেকর্ড ফেরত পাওয়া যায়, এবং বাম (first) টেবিলের সাথে সম্পর্কিত রেকর্ডও দেখানো হয়। যদি বাম টেবিলে সম্পর্কিত রেকর্ড না থাকে, তবে সেই রেকর্ডের জন্য NULL মান দেখানো হয়।

4. PostgreSQL FULL OUTER JOIN

PostgreSQL FULL OUTER JOIN হল একটি SQL অপারেশন যা দুটি টেবিলের সব রেকর্ডকে একত্রিত করে, এবং উভয় টেবিলের মধ্যে সম্পর্ক থাকা রেকর্ডগুলোকেও দেখায়।

Set A = {1, 2, 3, 4, 5}

Set B = {4, 5, 6, 7, 8}

Set A ∪ Set B = {1, 2, 3, 4, 5, 6, 7, 8}

FULL OUTER JOIN উদাহরণ:

ধরা যাক, আমাদের দুটি টেবিল রয়েছে:

1. employees টেবিল:

employee_id	name	department_id
1	John	101
2	Alice	102
3	Bob	103
4	Charlie	101

2. departments টেবিল:

department_id	department_name
101	HR
102	IT
104	Marketing

এখানে, employees টেবিলের department_id কলামটি এবং departments টেবিলের department_id কলামটি সম্পর্কিত।

FULL OUTER JOIN এর SQL Query:

আমরা যদি employees টেবিলের সমস্ত employee_name এবং তাদের সাথে সম্পর্কিত department_name দেখতে চাই, তাহলে আমরা FULL OUTER JOIN ব্যবহার করতে পারি:

```
SELECT employees.name, departments.department_name
FROM employees
FULL OUTER JOIN departments
ON employees.department_id = departments.department_id;
```

আউটপুট:

name	**department_name**
John	HR
Alice	IT
Bob	NULL
Charlie	HR
NULL	Marketing

ব্যাখ্যা:

- John এবং Charlie উভয়ই HR বিভাগের সদস্য, তাই তাদের নাম এবং বিভাগ দেখানো হয়েছে।
- Alice IT বিভাগের সদস্য, তাই তার নাম এবং বিভাগও দেখানো হয়েছে।
- কিন্তু Bob এর department_id = 103, যা departments টেবিলে নেই (এখানে কোনো "department_id 103" নেই), তাই Bob এর জন্য department_name কলামে NULL দেখানো হয়েছে।
- Marketing বিভাগ (department_id 104) এর সাথে কোনো employee সম্পর্কিত নেই (কারণ employees টেবিলের department_id 104 নেই), তাই department_name কলামে Marketing এসেছে এবং employee_name কলামে NULL দেখানো হয়েছে।

সারাংশ:

FULL OUTER JOIN ব্যবহার করলে, আপনি উভয় টেবিলের সমস্ত রেকর্ড পাবেন। যেখানে সম্পর্ক না থাকে, সেখানে NULL মান দেখানো হয়। এটি LEFT JOIN এবং RIGHT JOIN এর সমন্বিত ফলাফল প্রদান করে, এবং উভয় টেবিলের সমস্ত রেকর্ড অন্তর্ভুক্ত করা হয়।

Advanced Queries

1. PostgreSQL GROUP BY

PostgreSQL GROUP BY একটি SQL অপারেশন যা একই ধরণের ডেটাকে একটি গ্রুপে একত্রিত করতে ব্যবহার করা হয়। এটি সাধারণত ব্যবহার করা হয় যখন আপনি কোনো নির্দিষ্ট কলামের মান অনুসারে ডেটা গ্রুপ করতে চান এবং প্রতি গ্রুপে কিছু aggregate function (যেমন, COUNT(), SUM(), AVG(), MAX(), MIN()) প্রয়োগ করতে চান।

GROUP BY এর উদ্দেশ্য:

- ডেটাকে এক বা একাধিক কলামের ভিত্তিতে গ্রুপ করে, এবং প্রতিটি গ্রুপের জন্য একটি সারাংশ তৈরি করা।
- গ্রুপের উপর বিভিন্ন অ্যাগ্রিগেট ফাংশন (যেমন, গুন, যোগফল, গড়) প্রয়োগ করা।

GROUP BY এর সাধারণ সিনট্যাক্স:

```
SELECT column_name, aggregate_function(column_name)
FROM table_name
GROUP BY column_name;
```

এখানে:

- column_name: যে কলামের উপর গ্রুপিং করতে চান।
- aggregate_function: অ্যাগ্রিগেট ফাংশন (যেমন, SUM(), COUNT(), AVG(), ইত্যাদি)।

GROUP BY উদাহরণ:

ধরা যাক, আমাদের একটি sales টেবিল আছে যা বিক্রয়ের তথ্য ধারণ করে:

sales টেবিল:

sale_id	product_name	amount	sale_date
1	Laptop	500	2025-01-01
2	Phone	300	2025-01-01
3	Laptop	700	2025-01-02
4	Tablet	200	2025-01-02
5	Phone	500	2025-01-03
6	Laptop	600	2025-01-03

GROUP BY এর SQL Query:

ধরা যাক, আমরা প্রতিটি product_name এর জন্য মোট বিক্রয় পরিমাণ (sum) বের করতে চাই।

```
SELECT product_name, SUM(amount) AS total_sales
FROM sales
GROUP BY product_name;
```

SUM(amount) কে alias করলাম total_sales নামে।

আউটপুট:

product_name	total_sales
Laptop	1800
Phone	800
Tablet	200

ব্যাখ্যা:

- এখানে GROUP BY অপারেশনটি product_name কলামের উপর গ্রুপ করেছে।
- তারপর প্রতিটি গ্রুপে amount কলামের উপর SUM() ফাংশন প্রয়োগ করেছে, যার ফলে প্রতিটি পণ্যের জন্য মোট বিক্রয় পরিমাণ পাওয়া গেছে।
- Laptop পণ্যের মোট বিক্রয় পরিমাণ হলো 500 + 700 + 600 = 1800।
- Phone পণ্যের মোট বিক্রয় পরিমাণ হলো 300 + 500 = 800।
- Tablet পণ্যের মোট বিক্রয় পরিমাণ হলো 200।

সারাংশ:

PostgreSQL GROUP BY অপারেশনটি ব্যবহার করা হয় ডেটাকে একটি নির্দিষ্ট কলামের ভিত্তিতে গ্রুপ করার জন্য, এবং তারপর সেই গ্রুপগুলির জন্য বিভিন্ন অ্যাগ্রিগেট ফাংশন প্রয়োগ করতে। এটি সাধারণত বিশ্লেষণাত্মক কাজগুলির জন্য ব্যবহৃত হয়, যেমন মোট যোগফল, গড়, সংখ্যা, সর্বোচ্চ, সর্বনিম্ন ইত্যাদি বের করার জন্য।