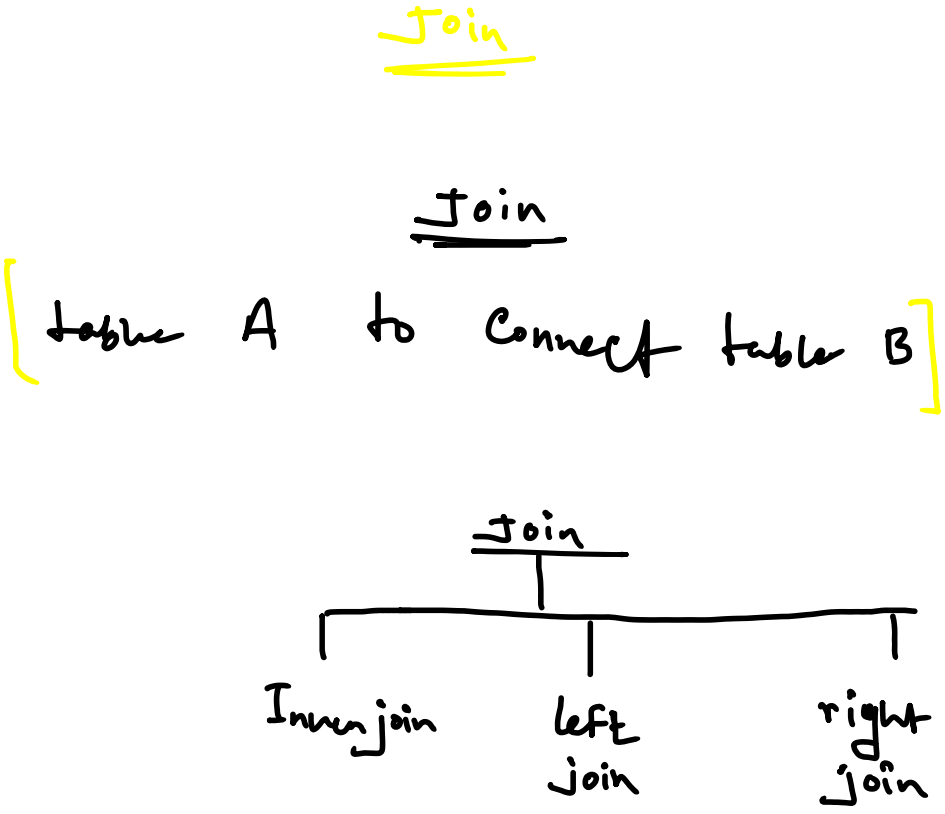
19-09-2024

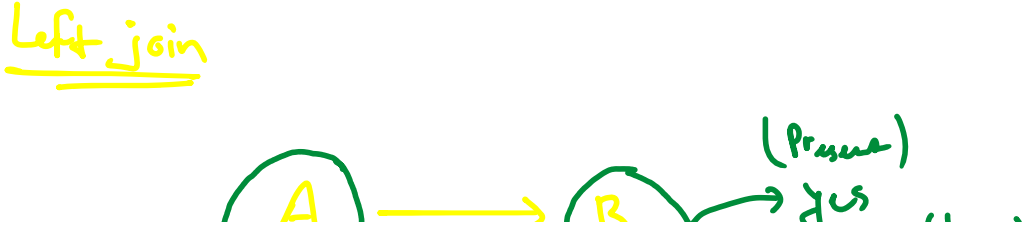
19 September 2024 05:49

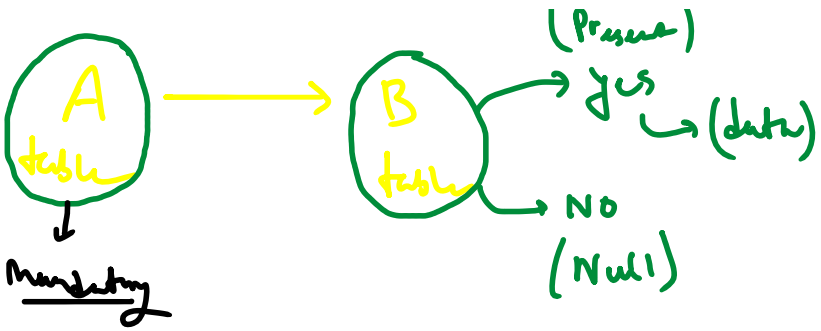
SQL Basic Queries

```
-- CREATE DATABASE store_product;

-- SELECT * FROM employees;

-- SELECT e1.employeeNumber, e1.reportsTo,
-- concat(e1.firstName, " ", e1.lastName) AS EmployeeName,
-- concat(e2.firstName, " ", e2.lastName) AS ManagerName, e2.jobTitle AS
ManagerJobTitle, e1.jobTitle AS EmployeeJobTitle
-- FROM employees e1
-- JOIN employees e2 ON e1.reportsTo = e2.employeeNumber

-- SELECT * FROM film;

-- SELECT f1.title, f1.release_year, f2.release_year, f2.rental_rate FROM film f1
-- JOIN film f2 ON f2.rental_rate > f1.rental_rate
-- AND f2.release_year BETWEEN f1.release_year - 2 AND f1.release_year + 2;

-- SELECT c.customerName, c.country, c.addressLine1, c.phone, c.creditLimit FROM
customers c
-- JOIN orders o ON c.country = 'France' AND c.customerNumber =
o.customerNumber
-- ORDER BY o.orderDate DESC;


-- Show each actor with film(s) they have done
-- SELECT f.title, f.description, concat(a.first_name, " ",a.last_name)
-- FROM film_actor fa
-- JOIN film f ON fa.film_id=f.film_id
-- JOIN actor a ON fa.actor_id = a.actor_id;

-- Agregator

-- SELECT COUNT(1) FROM actor;

-- SELECT * FROM city ORDER BY country_id DESC LIMIT 1 OFFSET 2;

-- SELECT batch_id, AVG(marks) FROM student GROUP BY batch_id;

 SELECT * FROM film;

 select COUNT(*), rating
 from film
 group by rating
 ORDER BY rating;

 select * from rental;
```

```
select COUNT(rental_id), r.customer_id, c.first_name
from rental r
JOIN customer c on r.customer_id = c.customer_id
group by r.customer_id
ORDER BY r.customer_id;


-- HW Problem
-- Challenging Query (HOMEWORK)
-- Query 4: Retrieve the total revenue earned by each film category
-- but include categories where the total revenue is greater than 1000 USD
-- Show categories and revenue, order the results in desc order by revenue -- HINT:
Map each payment ==(joins)=> film_category


-- SELECT
-- year, AVG(rent)
-- FROM film
-- GROUP BY year
-- HAVING AVG(rent) >= (SELECT AVG(rent) FROM film);
```

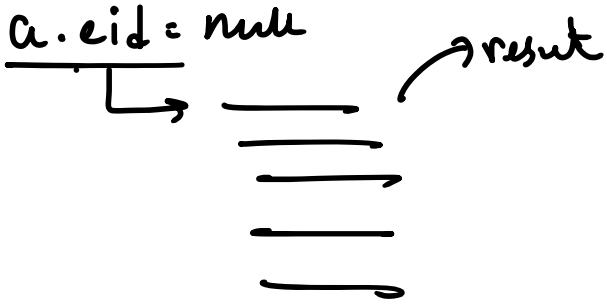






Example

```
select * from Actor a
left join Employee e on a.eid = e.id
```



Right join

→ Same as left join but opposite of the left join

A ⟶ B (Left)

A ⟵ B (Right)

Inner join

⇒ Both side should be present.

A ⟷ B (both side mandatory)

A ⟷ B (both side mandatory)

```
SELECT
c.customer_id, f.film_id
FROM customer c
LEFT JOIN film f ON c.customer_id = f.film_id;

SELECT
c.customer_id, f.film_id
FROM customer c
RIGHT JOIN film f ON c.customer_id = f.film_id;

SELECT
c.customer_id, f.film_id
FROM customer c
JOIN film f ON c.customer_id = f.film_id;
```

Group By, having

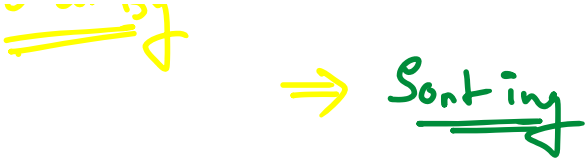
group By → remove duplicates depends on column.

having → having work with groupby based on condition.

```
-- GROUP BY

SELECT
release_year, COUNT(release_year)
FROM film
GROUP BY release_year;

-- HAVING
SELECT
release_year, MAX(rental_duration)
FROM film
GROUP BY release_year
HAVING release_year > 2009;
```

order By

⇒ Sorting




```
-- ORDER BY (SORTING)
SELECT * FROM film ORDER BY rental_duration DESC;
```