Session 1 - Introduction to SQL

21 August 2024 19:06

Batch DSML Aug24 Beginner 2,DSML July24 Beginner w/o Python

1. Module Importance & Overview
2. Dashboard Walkthrough
3. Problem Statement
4. What is a Database?
5. Database Management System
6. Database vs. Excel
7. DB Schema Design
8. Data Types
9. Concept of Keys
10. Data Warehouses
11. Setting up BigQuery

1. Introduction., Datatypes, Keys.
2. Data Extraction.
3. Data Aggregation
4. Combine Data (JOINS)
5. Adv. data analysis (RANK, ROW_NUMBER)
6. Adv. Construct (CTE)
7. MySQL setup
8. Beyond basics =>
9. Case study
10. Module test /re-test

PSP score (85+)

↘ 8 assignment Questions

→ 4 H.w questions

Case studies -

Module test / Mock interviews ✓

✓ = 70%

AFM (Amazon farmers Market)

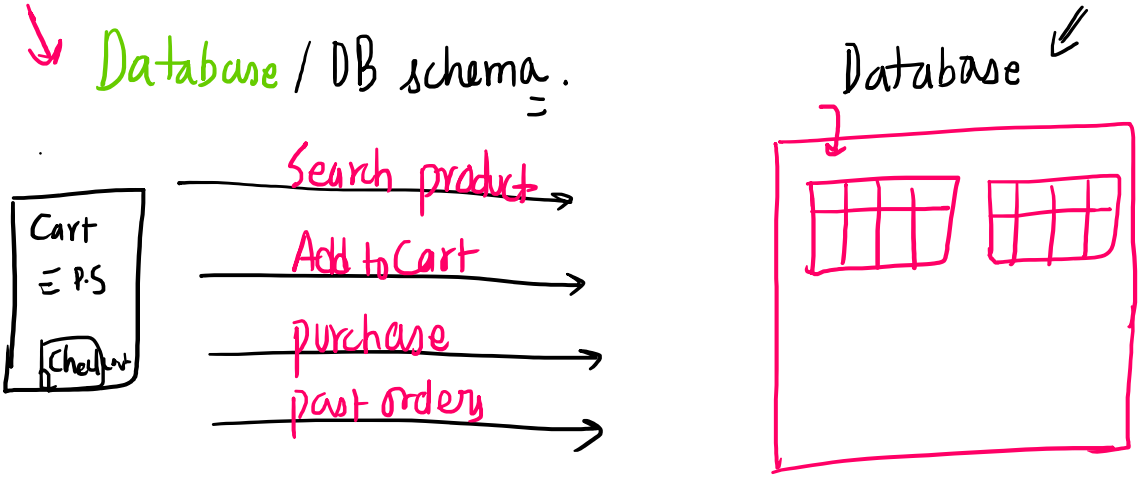
- ⇒ Cust. behaviour.
- ⇒ market operations.
- ⇒ inventory management.

→ Reliance fresh.

→ Big Bazaar.

→ Big basket

↓ Database / DB schema.



↓

DBMS

{ Software to manage large data }

Data ✓

Database ✓ } { store data }

Database

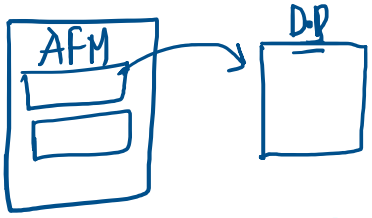
DBMS

Operations to manage it.

↘ CRUD ⇒ Create, read, Update, delete.

Data → Database → RDBMS

Operations – read
– create
– Updata
– Delete.



+ Oracle
+ MySQL
+ PostgeSQL
+ SQLite

(Entities)

table (User)

Purchase

Columns

Rows

id	Name	address
1	Prakash	Delhi
2	Sai	Bangalore
3	Bhavesh	U.K

id	dt	amount	u_id
101	21-08-24	100	1
102	21-0-24	5000	3
103	21-08-24	500	2

P.K

F.K

1. AFM App.
New Account.

2. purchased 5 Brocolli.

↘ RDBMS

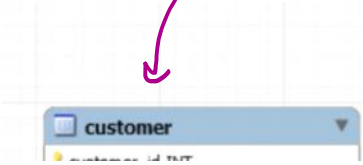
Columns ~ properties / attribute

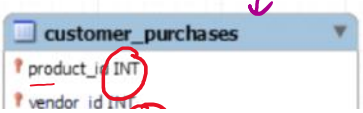
DB schema (structure)

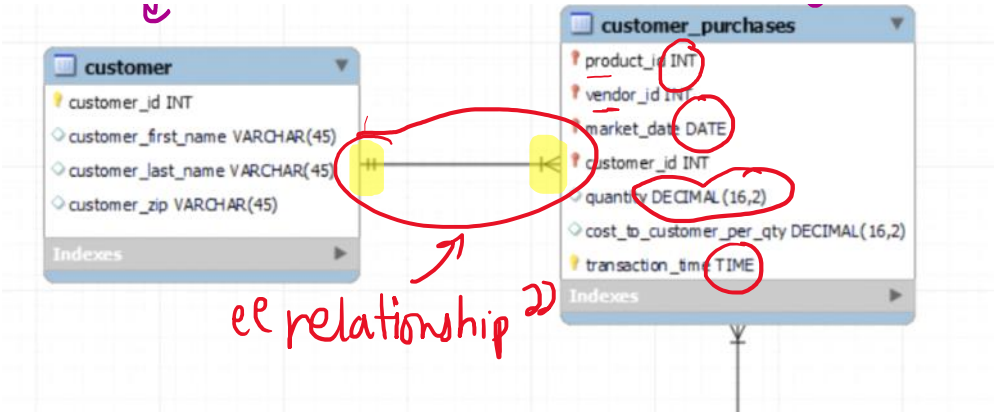
(ER diagram)

table

table / Entity / relation







↳ "prakash" } text / string

+ phone } 0123456789 // Number

+ 2010.10 } // Number but with decimal digit.

+ 21-08-2024 22:51:00 } Datetime

} Data Type

↳ String

↳ Numeric → Whole No. (Integer)
→ Decimal.

↳ Date & time

A, B, C, D

Fixed

"M/F" char(1)

IND / AUS / USA
fixed
char(3)

text (String) → Char (fixed length)
→ varchar (Variable length)

↳ varchar(100)

Country : IND

Numeric ⇒ int, float

int ⇒ 10, 20, 100, -10] ↓ 24
(integer)

float ⇒ 5.4 6.1 ✓ Accuracy

float ⇒ 5.4 6.1 ✓ Accuracy

Date & Time →

↘ Date = 2023-05-17] date

↘ time = 03:07:00 (hour / min / sec.)

↘ datetime =

↘ timestamp → '1970-01-01 00:00:00' UTC (Unix Epoch)

8912264810222

(Primary Key)

User

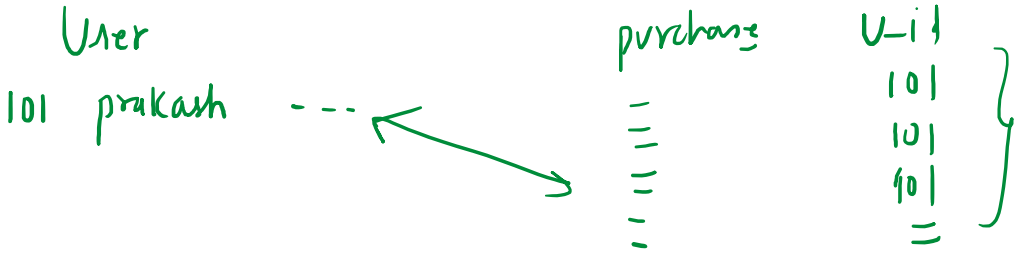
id	Name	phone	add.
1001	Sai	1234	
1002	Sai	5678	

↘ P.K → Null X

⇒ One P.K

⇒ Unchangeable ✓

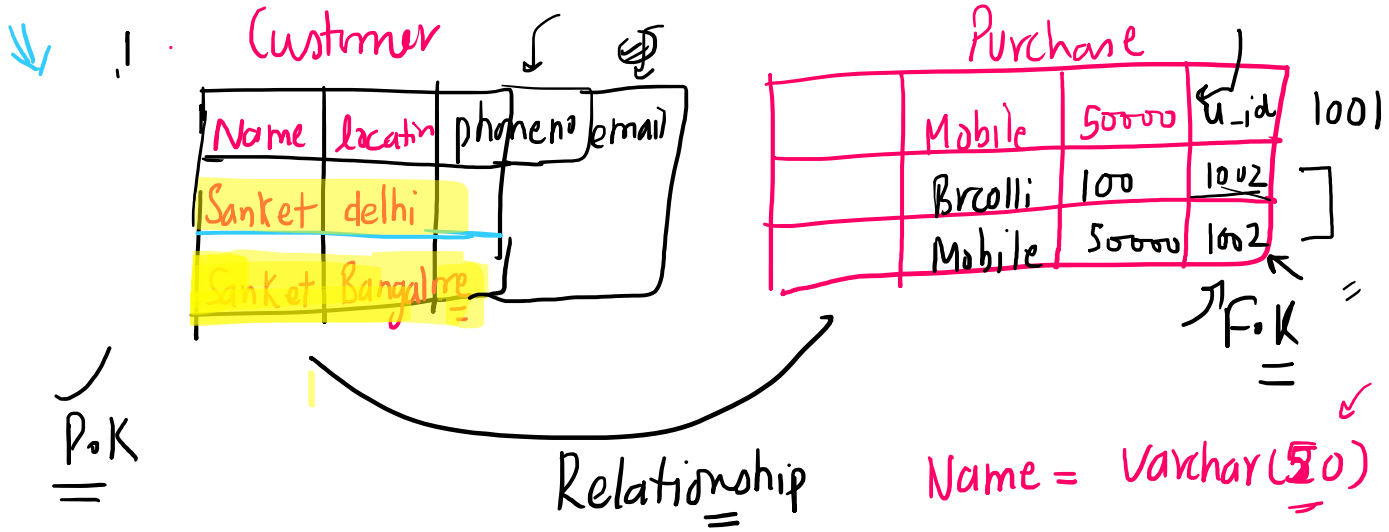
customer_id	aadhaar_number
1	4738 2956 7103
2	9123 7456 8521
3	1357 8964 2053
4	8765 4321 0987
5	



⇓ Characters ⇒ prakash, email, gender

⇒ char (fixed length) ↙ Gender (M/F) : M

⇒ varchar(Variable length) ↙ Name : (variable)



Session 2 - Extracting Data using SQL

21 August 2024 20:45

Database vs Data Warehouse?

⇒ create Account in AFM.
→ search fruits.
→ Added to Cart
→ purchase

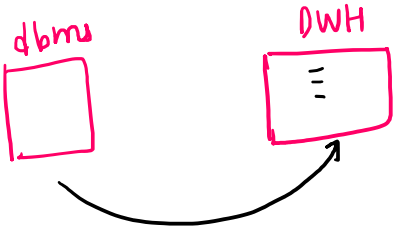
Database = Online transaction processing

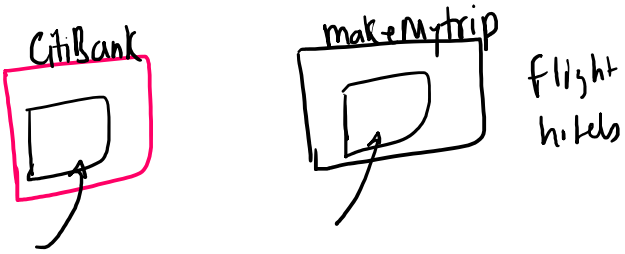
OLTP

OLAP (Online Analytical processing)

⇒ large repository of historical data.

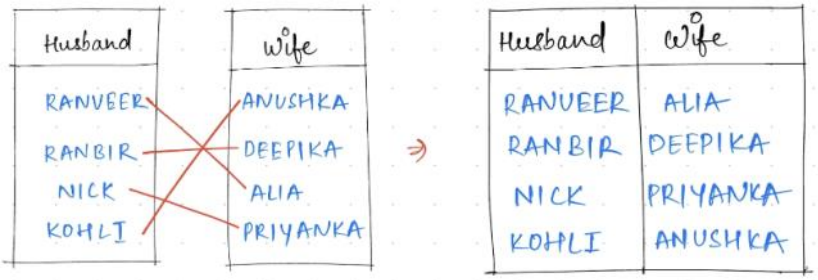
(Source)





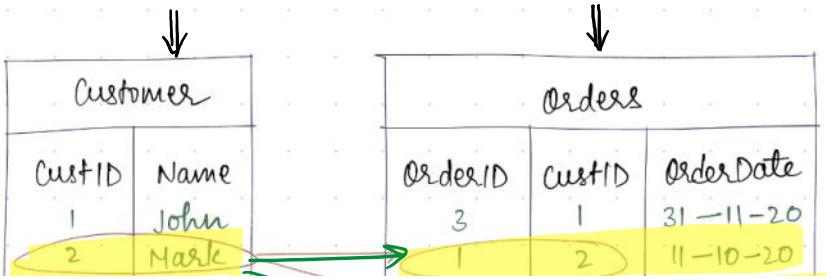
Relationships & Types:

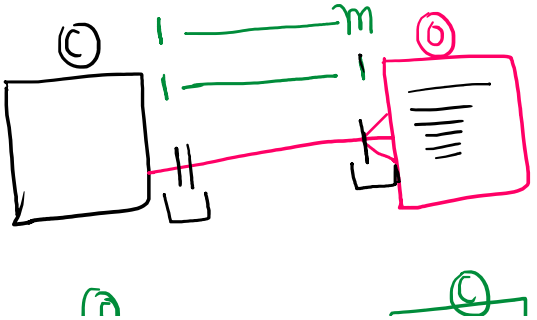
1. One to one

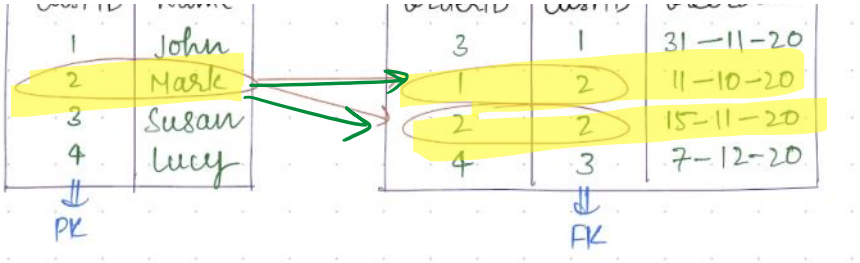


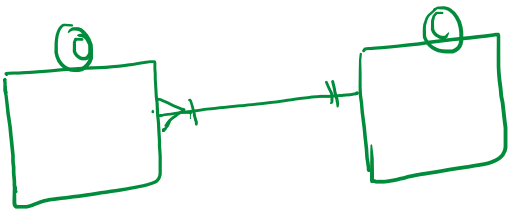
Person - VoterID ✓

2. One to Many (AFM)









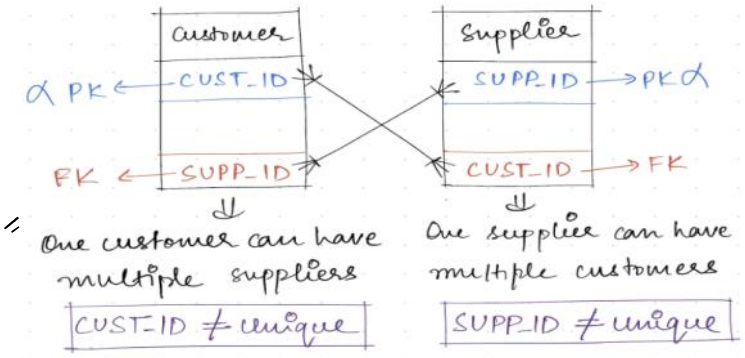
Author → Books ?

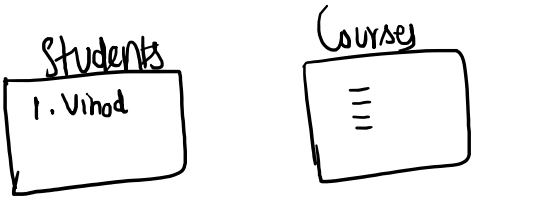
Student → Teacher ?

1 — m ⇒ m:m

m — 1

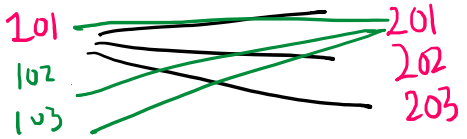
3. Many to Many





1 — m ⇒ m:m ✓

m — 1

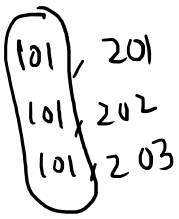


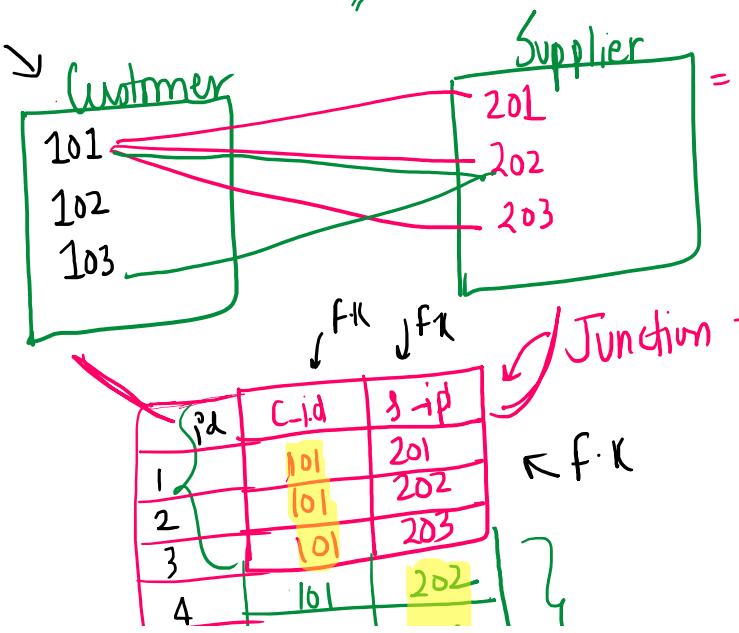
Customers — Suppliers

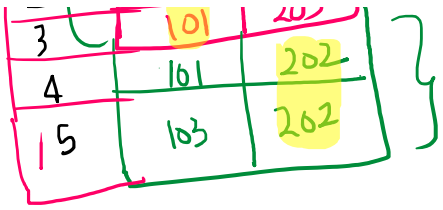
1 — m

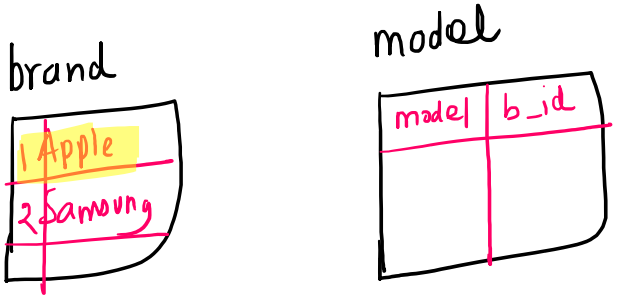
m — 1

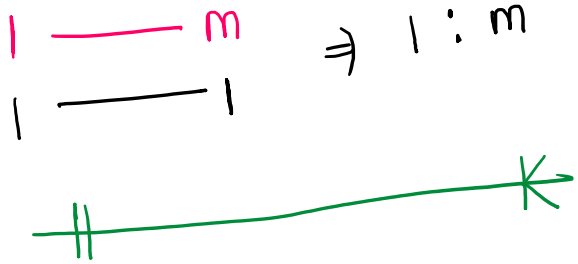
} = m:m











Question: How will you extract data from each of these tables?

Language
↳ SQL (Structured Query Language)
↑
Syntax

1. DDL → create table
Drop
Alter

2. DML = CRUD

3. TCL = S1
S2
↓ transaction →

Amazon
↳ Gift Card (1000)
≥ 2000

Order
Gift Card balance
1000 = 0

4. DQL = (Select)

5. DCL = (Grant, Revoke)

> = SELECT <Columns>

>= SELECT <columns>
FROM <table>
WHERE
GROUP BY
HAVING
ORDER BY
LIMIT

date	price
23-08-2024	50
23-08-2024	60
22-08-2024	100

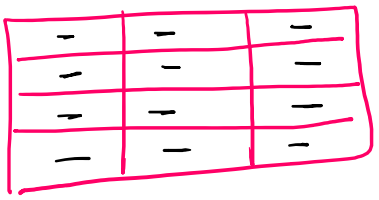
23-08-24	60
23-08-2024	50
22-08-2024	100

order by order_date desc

offset

Which of these is right? (N, M could be any number)

1. **Offset skips top 'N' rows**
2. **Limit skips to top 'N' rows**
3. **Offset decides 'M' rows to display**
4. **Limit decides 'M' rows to display**



limit 10 ⇒ 10 rows
limit 3 ⇒ 3 rows

```
select
  market_date,
  transaction_time,
  quantity,
  cost_to_customer_per_qty
from `farmers_market.customer_purchases`
order by  market_date desc, transaction_time
limit 3
offset 1;
```

① ⇒ FROM (tableName)
② ⇒ SELECT
③ ⇒ ORDER BY
④ ⇒ OFFSET → LIMIT ⑤

⇒ OFFSET → LIMIT

Limit
↓
offset 1

round (100.123, 1) ⇒ 100.1

ceil (5.4) ⇒ 6

floor (5.7) ⇒ 5

Concat (_ , _) ⇒

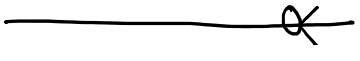
Problem Statement:
Which of the following scenarios represents a **one-to-one relationship** in a database?
a) Parent - Child
b) Person - Aadhar Card
c) Vehicle Identification Number - Vehicle
d) Employee - Manager
e) Student - Course
f) Person - Fingerprints

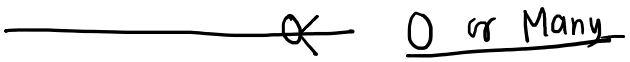
⑤

usecase_api_business_scenario_1

A price B price ⇒ A.price


1 or Many


0 or Many



Cust_id	Cust_email	Cust_phone	...

PK Candidate

1. Cust_id

2. email

3. phone

(2)

email

phone

Session 3 - Filtering Data in SQL

26 August 2024 15:47

⇒ Concat

⇒ Upper, Lower

⇒ substr, initcap

"pRAKash" ⇓ "Prakash"

⇒ Where

⇒ IN, BETWEEN

Amazon Farmers Market (AFM), a beloved community market, faces a growing threat from large grocery chains that are expanding their local produce offerings. To maintain its customer base and thrive in this competitive environment, AFM needs to leverage data analytics for a deeper understanding of:

- **Customer Behavior**: Analyze purchase history, spending habits, and visit frequency to identify customer segments, preferences, and potential churn risks.
- **Vendor Performance**: Analyze sales data and customer attraction metrics to identify top-performing vendors, evaluate product popularity, and optimize vendor selection.
- **Market Operations**: Analyze traffic trends and seasonal sales patterns to optimize inventory management, staffing levels, and marketing campaigns.
- **Inventory management**: Analyze future demand for specific products based on historical sales data to optimize the right amount of inventory to meet customer needs while minimizing waste and storage costs.

What is the right format if you want to concatenate month (eg. Jan) and year (eg. 2020) columns, separated by a hyphen (-)?

Sample Output: Jan-2020

Seperator

CONCAT (month , "-" , year)

Second Column

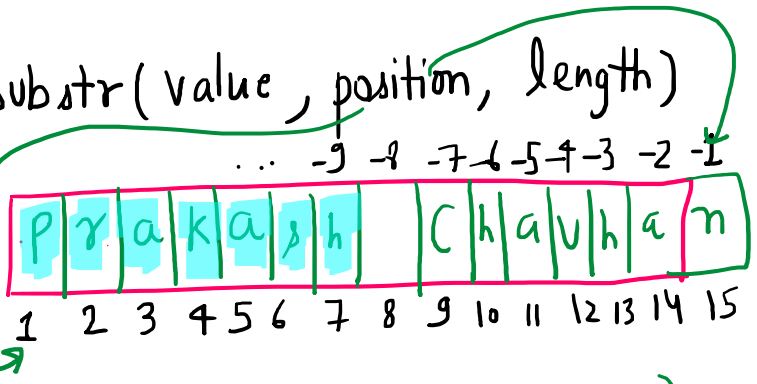
CONCAT (month + "-" + year) X

Prakash Chauhan

⇓

Hello P. Chauhan

substr(value , position , length)



substr("Prakash Chauhan", 1, 7)

Value = "Prakash Chauhan"

position = 1

length = 7

Prakash Chauhan

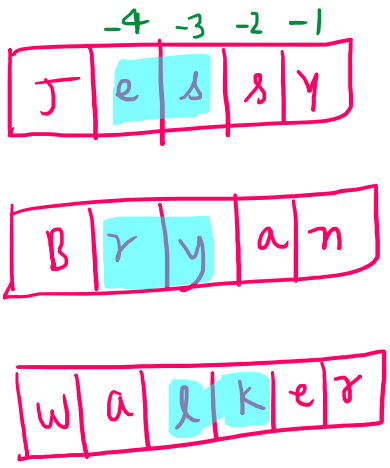
-1 = position

7 = Length

Prakash Chauhan

-7

What is the output of SUBSTR(customer_name, -4, 2) on a customer column having the following names “Jessy”, “Bryan”, and “Walker”?



p = -4

l = 2

"es" "ry" "lk"

⇒ filter the Data → WHERE

Question: Print a report of everything the customer_id 4 has ever purchased at the market, sorted by date. Add total_amt column as well for each purchase.

```
select
  customer_id,
  market_date,
  quantity,
  cost_to_customer_per_qty,
  round(quantity * cost_to_customer_per_qty, 2) as
total_amt
from `farmers_market.customer_purchases`
where customer_id = 4
order by market_date;
```

✓ Offset
✓ Limit

Select *

Which is the right query to get all info about a customer named "Rajesh"? ← where

A
Select * from DB.table where customer_name == "Rajesh" ✗

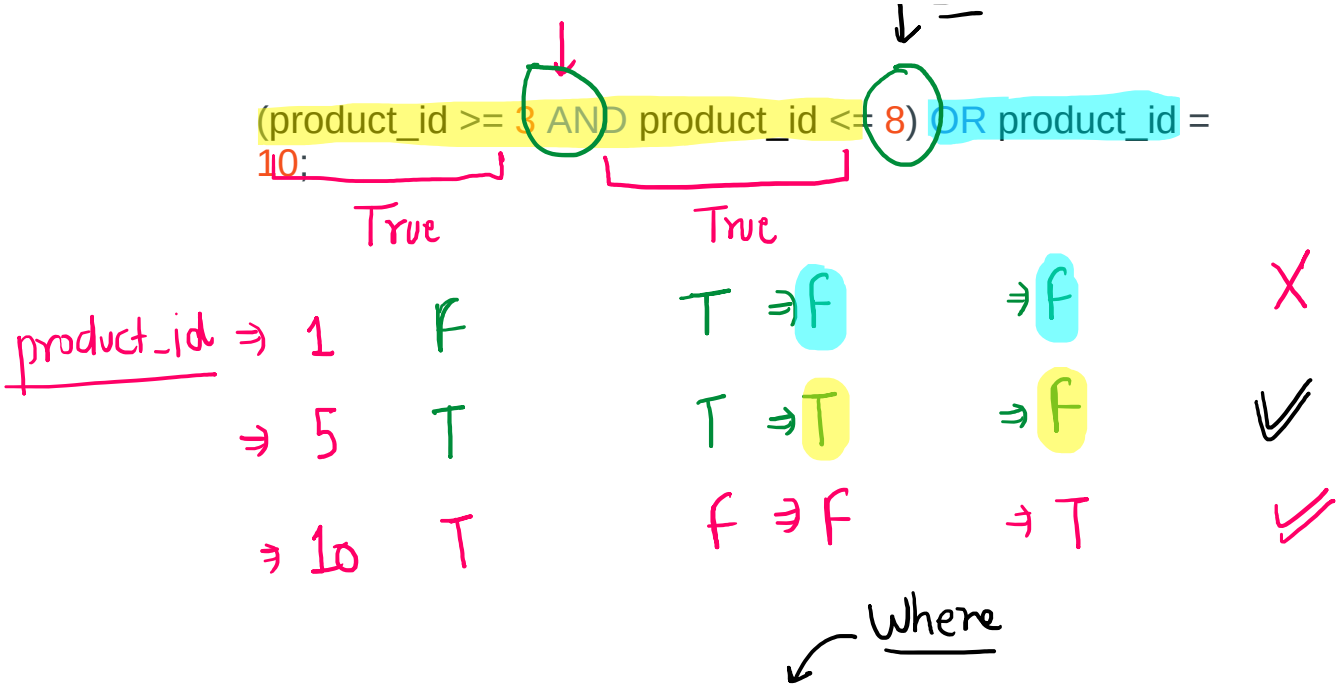
B
Select customer_name from DB.table where customer_name = "Rajesh" ✗

C
Select * from DB.table where customer_name = "rajesh" ✗

D
Select * from DB.table where customer_name = "Rajesh" ✓

Q (product_id >= 3, product_id <= 8, product_id = 10) } AND OR NOT

AND



Where

To get all transactions made by customer "A" but not on the date "2019-02-04"?

A
Select * from DB.table where customer_name = "A" and date = "2019-02-04"

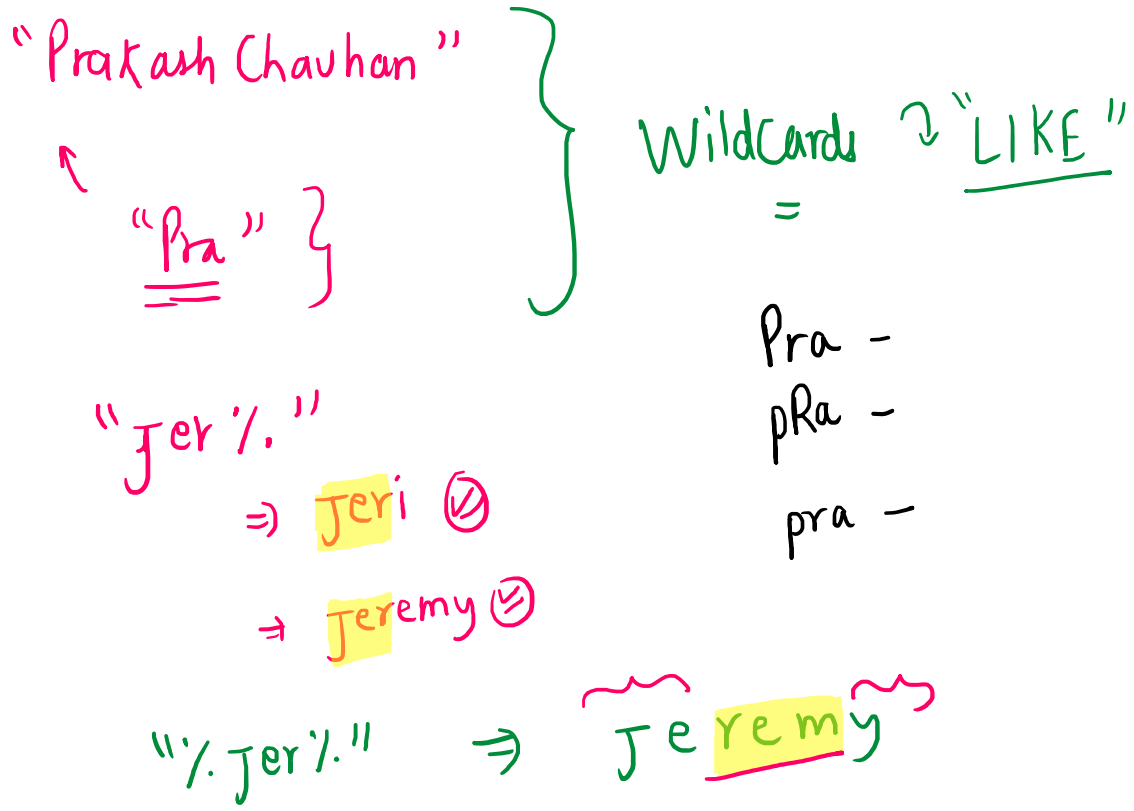
B
Select * from DB.table where customer_name = "A" and date <> "2019-02-04"

C
Select * from DB.table where customer_name = "A" or date = "2019-02-04"

D
Select * from DB.table where customer_name = "A" or date <> "2019-02-04"

Question: Find the booth assignments for vendor_id 7 for all dates between April 3, 2019, and May 16, 2019, including the 2 dates.

```
select *
from `farmers_market.vendor_booth_assignments`
where vendor_id = 7
AND ( market_date >= "2019-04-03" AND market_date <= "2019-05-16");

select *
from `farmers_market.vendor_booth_assignments`
where vendor_id = 7 AND (market_date BETWEEN "2019-04-03" AND "2019-05-16");
```



Here are some examples showing different `LIKE` operators with '%' and '_' wildcards:

LIKE Operator	Description
WHERE CustomerName LIKE 'a%'	Finds any values that starts with "a"
WHERE CustomerName LIKE '%a'	Finds any values that ends with "a"
WHERE CustomerName LIKE '%or%'	Finds any values that have "or" in any position
WHERE CustomerName LIKE '_r%'	Finds any values that have "r" in the second position
WHERE CustomerName LIKE 'a__%'	Finds any values that starts with "a" and are at least 3 characters in length
WHERE ContactName LIKE 'a%o'	Finds any values that starts with "a" and ends with "o"

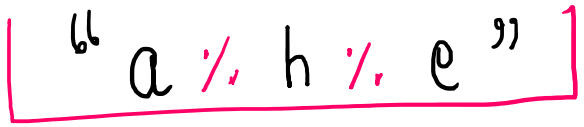
What is the wildcard string to get all names that start with ‘a’, end with ‘e’ and ‘h’ can be anywhere in between?

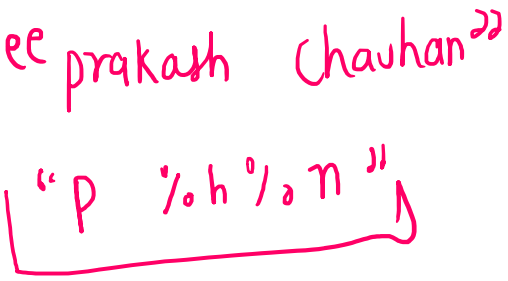
A
“%ah%e”

B
“a_h%e”

C
“a%h%e”

D
“%a_h_e”





Queries Covered in Session

```
select concat("Prakash"," ","Chauhan");

select concat("Prakash","--","Chauhan");

select upper("Prakash");

select lower("PRAkaSh");

select SUBSTR("Prakash Chauhan", 1, 7);

select SUBSTR("Prakash Chauhan", 1, 1);

select SUBSTR("Prakash Chauhan", -1, 7);

select SUBSTR("Prakash Chauhan", -7, 7);
```

```
SELECT
  customer_first_name,
  customer_last_name,
  CONCAT(UPPER(customer_first_name), " ", UPPER(customer_last_name)) full_name
FROM
  `farmers_market.customer`
LIMIT
  10;
```

```
SELECT
  customer_first_name,
  customer_last_name,
  UPPER(CONCAT(customer_first_name, " ", customer_last_name)) full_name
FROM
  `farmers_market.customer`
LIMIT
  10;
```

```
SELECT
  customer_first_name,
  customer_last_name,
  UPPER(CONCAT(customer_first_name, " ", SUBSTR(customer_last_name, 1, 1)))
full_name
FROM
  `farmers_market.customer`
LIMIT
  10;
```

```
SELECT
  product_id,
  product_name,
  product_category_id
FROM
  `farmers_market.product`
where product_category_id = 1
limit 3;
```

```
SELECT
  product_id,
  product_name,
  product_category_id
FROM
  `farmers_market.product`
where product_category_id <> 1;
```

```sql
select
  customer_id,
  market_date,
  quantity,
  cost_to_customer_per_qty,
  round(quantity * cost_to_customer_per_qty, 2) as total_amt
from `farmers_market.customer_purchases`
where customer_id = 4
order by market_date;



select *
from `farmers_market.product`
where (product_id >= 3 AND product_id <= 8) OR product_id = 10;



select *
from `farmers_market.product`
where product_id BETWEEN 3 and 8;



select *
from `farmers_market.vendor_booth_assignments`
where vendor_id = 7
AND ( market_date >= "2019-04-03" AND market_date <= "2019-05-16");



select *
from `farmers_market.vendor_booth_assignments`
where vendor_id = 7 AND (market_date BETWEEN "2019-04-03" AND "2019-05-16");



select *
from `farmers_market.customer`
where customer_last_name = "Diaz"
OR customer_last_name = "Wilson";



select *
from `farmers_market.customer`
where customer_last_name IN ("Diaz", "Wilson");



select *
from `farmers_market.customer`
where customer_last_name NOT IN ("Diaz", "Wilson");



select *
from `farmers_market.customer`
where UPPER(customer_first_name) LIKE "%REM%";
```

Session 4 - Filtering Continued & Subqueries

28 August 2024 18:27

Agenda

1. Problem Statement
2. DISTINCT
3. Null vs. Blank
4. IS NULL, IS NOT NULL
5. TRIM()
6. IFNULL()
7. Subqueries
8. CASE statement
9. IF() function

Amazon Farmers Market (AFM), a beloved community market, faces a growing threat from large grocery chains that are expanding their local produce offerings. To maintain its customer base and thrive in this competitive environment, AFM needs to leverage data analytics for a deeper understanding of:

- **Customer Behavior**: Analyze purchase history, spending habits, and visit frequency to identify customer segments, preferences, and potential churn risks.
- **Vendor Performance**: Analyze sales data and customer attraction metrics to identify top-performing vendors, evaluate product popularity, and optimize vendor selection.
- **Market Operations**: Analyze traffic trends and seasonal sales patterns to optimize inventory management, staffing levels, and marketing campaigns.
- **Inventory management**: Analyze future demand for specific products based on historical sales data to optimize the right amount of inventory to meet customer needs while minimizing waste and storage costs.

Questions: Your manager wants to see all the unique customer IDs present in the `customer_purchases` table. How would you get this data?

DISTINCT

```
Select distinct col1, col2, ...
from db.table
```

```
Select distinct customer_id
from farmers_market.customer_purchases;
```

From
↓
Where
↓
Select

from farmers_market.customer_purchases;

Select
↓
distinct
↓
order by
↓
offset → Limit.

```
select distinct customer_id
from `farmers_market.customer_purchases`;
```

```
select customer_id
from `farmers_market.customer_purchases`;
```

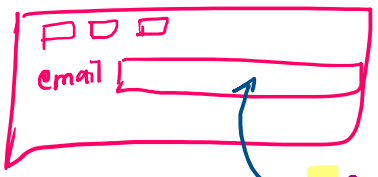
Question: Find all of the products from the `product` table which don't have their sizes mentioned.

NULL ?

"Prakash Chauhan"

```
select *
from `farmers_market.product`
where product_size IS NULL
OR trim(product_size) = ""
OR trim(product_qty_type) = "";
```


```
select product_id,
      ifnull(product_size, "Medium") product_size
from `farmers_market.product`;
```



pratas._1@scaler.com

→ [trim(email)]

Question: Fetch all the product IDs from the `products` table and set a default value "medium" in rows where the product size is NULL?

Subqueries

Question: Analyze purchases made at the market on days when it rained

```
SELECT market_date, market_rain_flag
FROM farmers_market.market_date_info
WHERE market_rain_flag = 1
```

```
SELECT *
   FROM farmers_market.customer_purchases
   WHERE market_date IN ("2020-07-11", "2020-07-08");
```

Question: List down all the product details where product_category_name contains “Fresh” in it.

Select columns from DB.table1 where col1 in (select col1, col2 from DB.table2)

```
Select *
from DB.table1
Where Col1 IN (Select Col1, Col2 from DB.table2)
```

- **Select columns from DB.table1 where col1 = (select col1 from DB.table2 limit 1).**

```
Select *
from db.table
Wher wer_email = "abc.abc.com"
```

- **Select * from DB.tables3 where col1 in (select col1 from DB.table1) and col2 in (select col2 from DB.table2)**

```
Select *
from db.table3
Where Col1 IN (Sub Query)
AND Col2 IN (Sub Query)
```

CASE Statement =>

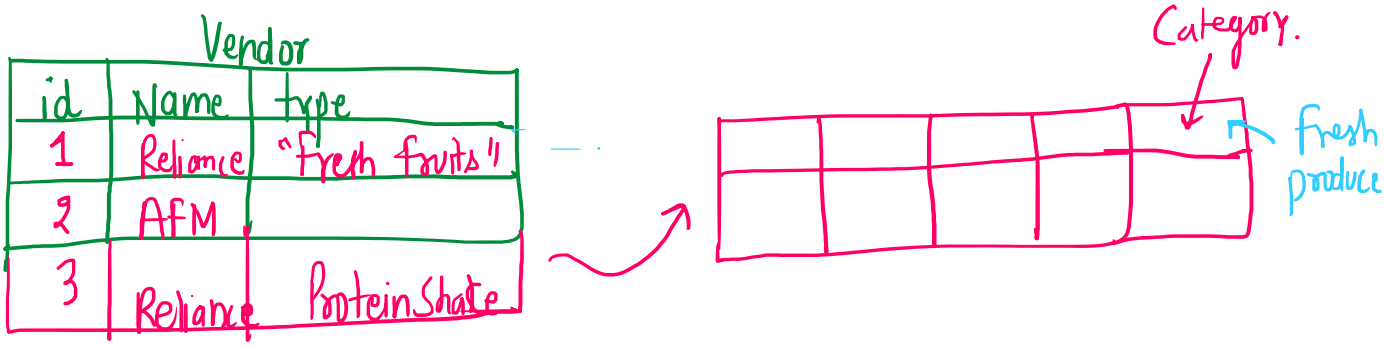
if <Condition> is True, the <do this>

Otherwise <do that>

```
SELECT cols,
CASE
  WHEN <condition> THEN
  WHEN <condition> THEN
END as <Alaias>
```

IF()

Question: Find out which vendors primarily sell fresh products and which don't.




```
select vendor_id,
vendor_name,
vendor_type,
CASE
  WHEN LOWER(vendor_type) like "%fresh%"
    THEN "Fresh Produce"
  ELSE "Other"
END AS Category
from `farmers_market.vendor`;
```

```
select *,
IF(LOWER(vendor_type) LIKE "%fresh%", "Fresh", "Not Fresh") AS Category
from `farmers_market.vendor`;
```

What if we want to add 1 for vendors who sell fresh products and 0 for those who don't?.

```
select vendor_id,
vendor_name,
vendor_type,
CASE
  WHEN LOWER(vendor_type) like "%fresh%"
    THEN 1
  ELSE 0
END AS Category
from `farmers_market.vendor`;
```

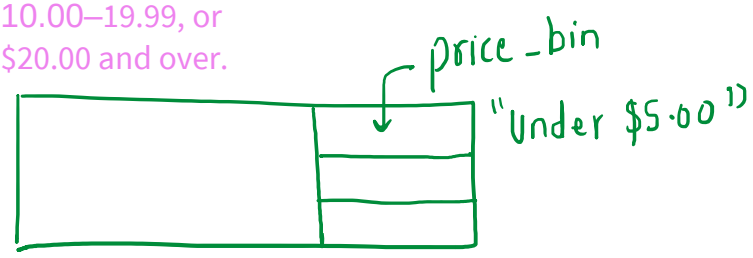
Question: Put the total cost to customer purchases into bins of -

- under $5.00,
- 5.00–9.99,
- 10.00–19.99, or

Under $5 - $9.99

- under $5.00,
- 5.00–9.99,
- 10.00–19.99, or
- $20.00 and over.

Under $5 - $9.99



Will both of these queries give the same result?

- **Select case when col1 = 'a' then 'call me a' else 'call me b' end as new_col from DB.table**
- **Select if(col1 = 'a', 'call me a', 'call me b') as new_col from DB.tbl.**

F
T

```
Case
  When Col1 = 'a' then 'Call me a'
  else 'Call me b'
end as new_col
from table
```

if (col1 = "a", "this", "that")

why can't I simply use select Name from Vendor where upper(type) like"%fresh%"

Upper(type) like "%FRESH%"

```
select vendor_id,
vendor_name,
vendor_type,
CASE
  WHEN LOWER(vendor_type) like "%fresh%"
    THEN "Fresh Produce"
  ELSE "Other"
END AS Category
from `farmers_market.vendor`;
```

Please explain why no blank i.e "" , is not null?

"" " "

[""] String ~ Blank String

[" "] ← with a space character.

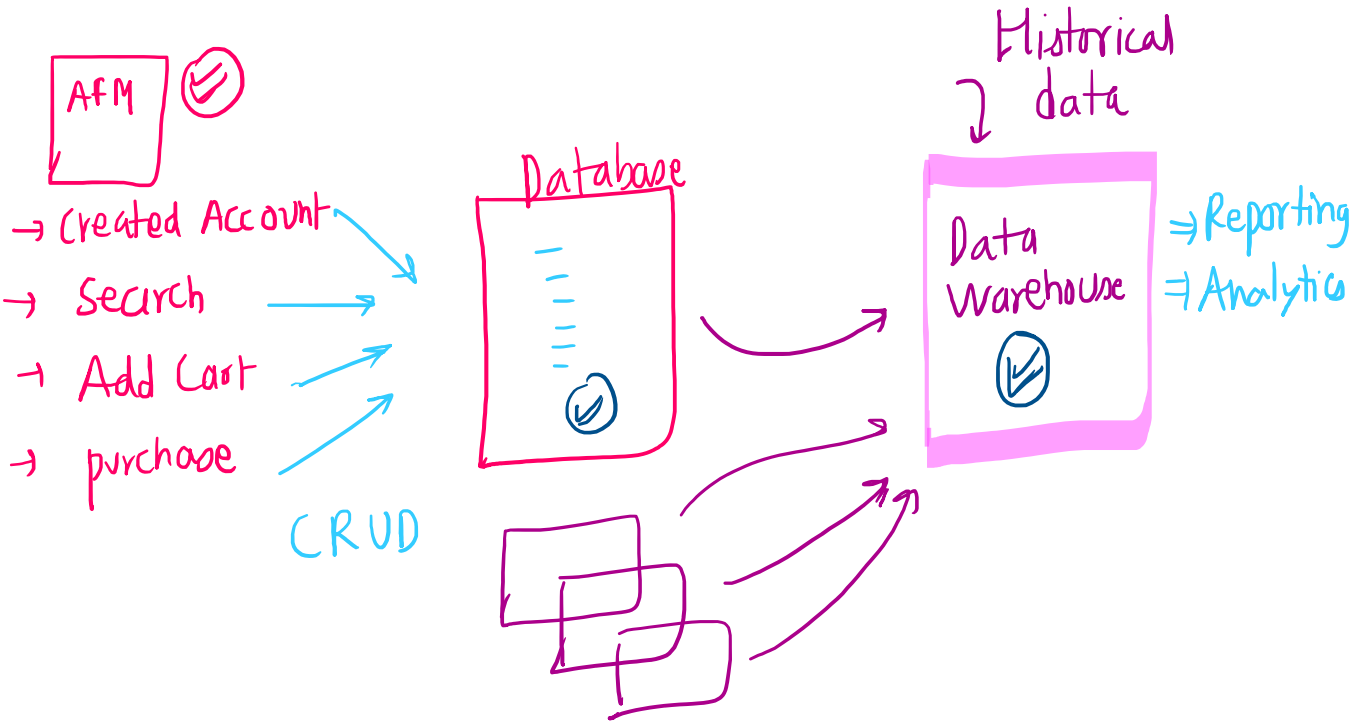
[" "] ~ String with space character.

can we use round function without giving integer value for decimal places ? example round(salary) instead of round(salary,2)

↳ price = 21.12345 ≈ 21.12

[Query Outer
↳ (Inner Query) ↖ Alias] CTE ↘

Data gets refresh from source system as and when needed--this is wrong statement from the first class. could you please explain



Shaik Vazeer
how to get the longest name in a column, i saw this question somewhere the other day. ?

QUERIES DISUCUSSED IN SESSION

```
SELECT
  *
FROM
  `farmers_market.product`
WHERE
  product_size IS NULL
  OR TRIM(product_size) = ""
  OR TRIM(product_qty_type) = "";
SELECT
  product_id,
  IFNULL(product_size, "Medium") product_size
FROM
  `farmers_market.product`;


SELECT market_date FROM `farmers_market.market_date_info` WHERE
market_rain_flag = 1;

SELECT *
FROM `farmers_market.customer_purchases`
WHERE market_date IN (
    SELECT
    market_date
  FROM
    `farmers_market.market_date_info`
  WHERE
    market_rain_flag = 1
);


select vendor_id,
vendor_name,
vendor_type,
CASE
  WHEN LOWER(vendor_type) like "%fresh%"
    THEN "Fresh Produce"
  ELSE "Other"
END AS Category
from `farmers_market.vendor`;


select *,
IF(LOWER(vendor_type) LIKE "%fresh%", "Fresh", "Not Fresh") AS Category
from `farmers_market.vendor`;



select vendor_id,
vendor_name,
vendor_type,
CASE
  WHEN LOWER(vendor_type) like "%fresh%"
    THEN 1
  ELSE 0
END AS Category
from `farmers_market.vendor`;


select round(1234.1234, 0);
```

Session 5 - Group By and Aggregation

30 August 2024 19:05

Agenda

Problem Statement
Aggregate functions
- MIN, MAX
- SUM
- AVG
- COUNT

COUNT(*), COUNT (1), COUNT DISTINCT
Break & Doubt Resolution
Group By
Impact of the analysis
Practice Question

Challenges:

- With a vast and diverse patient base, Apollo Hospitals is struggling to understand patient needs and tailor services effectively across different age groups, genders, and ethnicities.
- Ensuring that the hospitals are adequately staffed and resourced to handle varying levels of demand and emergencies is a continuous challenge.

Goal: The goal of this project is to conduct a comprehensive analysis of patient demographics, health metrics, hospital performance, and financial statistics using SQL.

- **Patient Profiles:**
- **Health Conditions:**
- **Hospital Performance:**
- **Financial Statistics:**

Aggregate Functions

```
Select max(age) maxage,
       min(age) min_age,
       avg(age) avg_age
from db.table;
```

≈ Query - min
Query - min

Q. Determine the age range of patients admitted to the hospital.

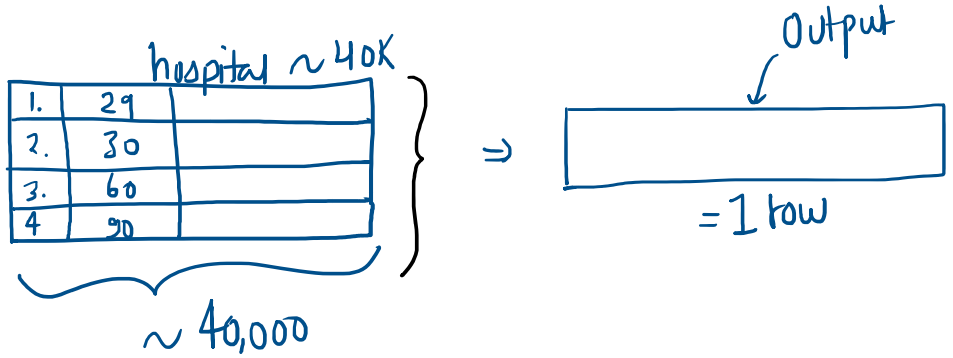
```
select Age
from `apollo_hosptal.hospital`
order by Age
limit 1;

select Age
from `apollo_hosptal.hospital`
order by Age desc
limit 1;

select min(Age) min_age,
      max(Age) max_age,
      round(avg(Age),2) avg_age
from `apollo_hosptal.hospital`;
```

BMI

Q. What is the average BMI (Body Mass Index) of the patients diagnosed with Obesity?

Medical_condition

```
select round(avg(BMI),2) avg_bmi
from `apollo_hosptal.hospital`
where lower(Medical_Condition) = "obesity";
```



Q. How many patients' records do we have in our

database?

Pid	Name	Age
1.	A	20
2.	B	30
3	C	40

⇒ Count(*) ⇒ 3 ?

Purchase

Pid	Amt.
1.	1000
1.	1500

⇒ Count (*) ⇒ ?

Count(*)	Count(1)	Count(Column)	Count(distinct Column name)
Count NULL Commonly Used	Exactly same Better to use Count(*)	Counts Duplicates Ignore NULL	No Duplicates ✓ No NULL ✓

Q. Which aggregate function will be used to calculate top-performing customers in terms of sales?

```
SELECT customer_id, (???) AS total_sales
FROM customer_sales
ORDER BY total_sales DESC
LIMIT 1;
```

Customer_sales

Customer_id	C_name	sales
1.	A	100
1.	A	500
1.	A	400

```
Select customer_id, SUM(sales) AS total_sales
from customer_sales
order by ...
limit 1;
```

Q. To identify the customer who made the highest number of purchases, which aggregate function should be used?

```
SELECT customer_id, (????) AS total_purchases
FROM customer_purchases
ORDER BY total_purchases DESC
LIMIT 1;
```

Customer_sales

Customer_id	C_name	sales
1.	A	100
1.	A	500
1.	A	400
2.	B	500

(3) (1)

```
LIMIT 1;
```

Select customer_id , Count(*) ...

1.	A	400
2.	B	500
3	C	100
3	C	50

① ②

Desc

1	3
2	1
3	2

Q. To find out how many different products were sold during the month July 2023, which aggregate function should you use?

product	Amount	P_date
iphone15pm	1.8L	July-23
iphone15pm	1.8L	July-23
samsung	1.2L	May-23

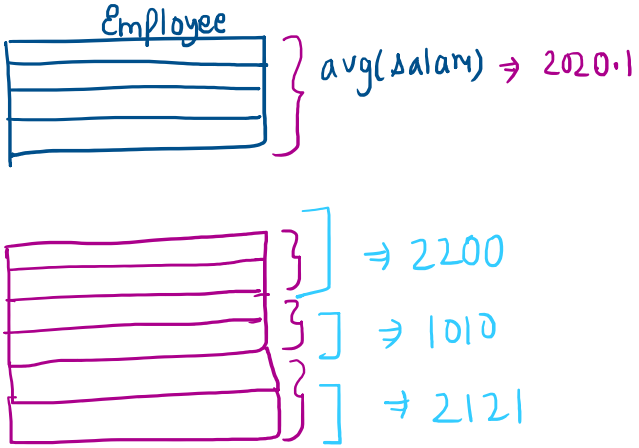
=

(A) Count(product) = 2

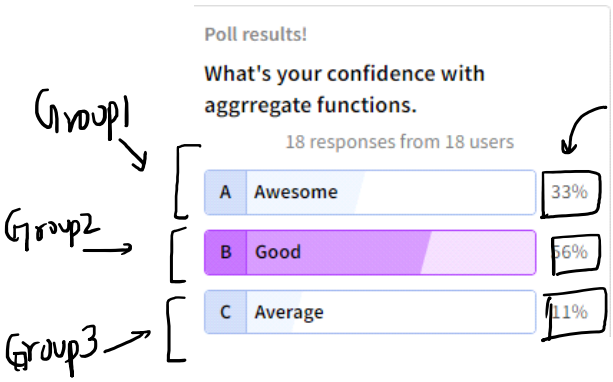
(B) Count(distinct product) = 1

= 22:23 ✓

Group By



GROUP BY ✓

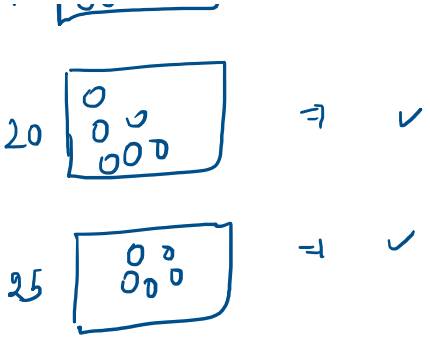


Awesome → 33%
Good → 60%
Avg. → 15%

Q. What is the distribution of patients' ages across the data?

Age,	Patient_Count

18 ⇒ ✓



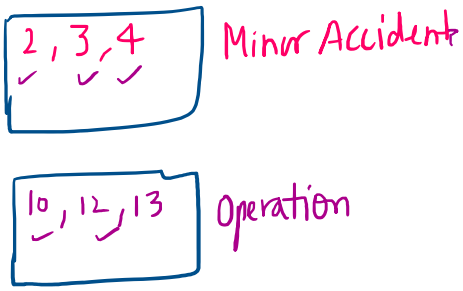

```
Select Age, count(*) patient_count
from `apollo_hosptal.hospital`
group by Age
order by Age;
```

Q. How do billing amounts vary based on the patient's insurance provider?

↳ insurance_provider = sum(billing)
↳ billing_details =

```
Select Insurance_Provider, sum(Billing_Amount) total_claim
from `apollo_hosptal.hospital`
group by Insurance_Provider
order by total_claim desc;
```

Q. Analyze and compare the average duration of hospitalization for various medical conditions.




```
Select Medical_Condition,
      round(avg(Days_Hospitalised),2)
avg_days_Hospitalised
from `apollo_hosptal.hospital`
group by Medical_Condition;
```

Q. Calculate the average billing amount for cancer patients in each hospital.

AVG
Where
Group

```
Select Hospital, avg(Billing_Amount) avg_billing_Amount
from `apollo_hosptal.hospital`
where lower(Medical_Condition) = "cancer"
group by Hospital
order by avg_billing_Amount desc;
```

Blood Group

Q. How are the different blood types distributed among

Q. How are the different blood types distributed among patients with diabetes?

Where

```
Select Blood_Type, count(*) num_patients
from `apollo_hosptal.hospital`
where lower(Medical_Condition) = "diabetes"
group by Blood_Type;
```

% of percentage.

Q. What percentage of patients are diagnosed with each medical condition?

Medical_Condition ✓ Obesity = 100, total = 2000 { 100/2000 x 100 }

⇒

0

Q. Which is the right query to get overall sales per customer in India?

A
SELECT cust_id, SUM(sales) FROM DB.tbl HAVING country = 'India' sales GROUP BY cust_id;

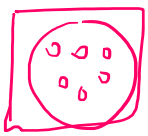
B
SELECT cust_id, SUM(sales) FROM DB.tbl WHERE country = 'India' GROUP BY cust_id;

C
SELECT country, cust_id, SUM(sales) FROM DB.tbl GROUP BY cust_id, market_date;

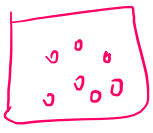
Sharique Rahi
Does GROUP BY always need to be used with aggregate functions?

No

Select AVG(Salary) from Employees;



Happy ⇒ 15



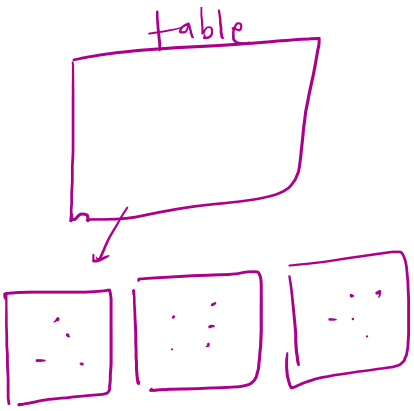
Happy ⇒ 7

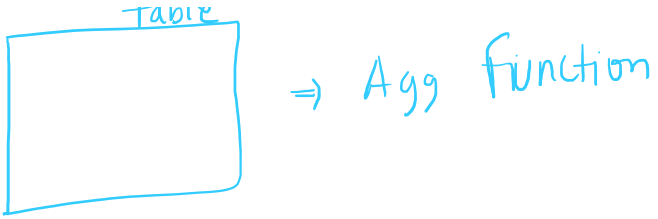
table

SELECT department
FROM employees
GROUP BY department;

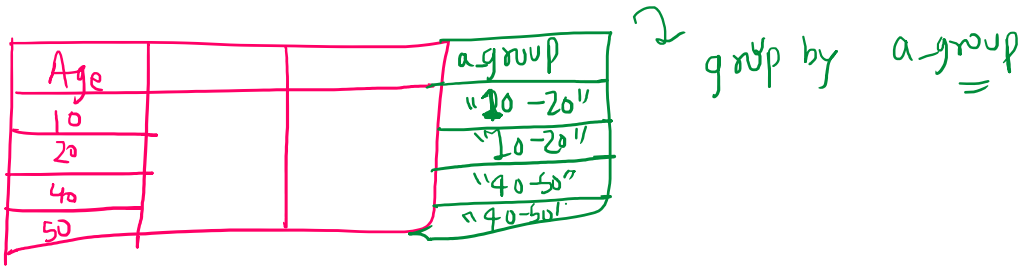
Select count(distinct department) from employees;

table ⇒ Agg function





Akifa Iram
what if we want to group age in a specific time interval ex:10-20, 21-30 and so on can we do this using group by



Karan
select COUNT(Medical_Condition = "diabetes") as diabetes_count
from dsml-scalar-sql.Hospital.hospital;

select count(*) num_patients from dsml-scalar-sql.Hospital.hospital;

both these queries are giving same results. Why is that?

Sharique Rahi
In this query , the one which Karan shared (

select Blood_Type, COUNT(Medical_Condition = "diabetes") as diabetes_count
from dsml-scalar-sql.Hospital.hospital
group by Blood_Type
order by diabetes_count desc;

do we allow to use where condition in count funct itself ? If yes , what will be the order of execution here ?
Same as before?

QUERIES EXECUTED IN CLASS

```
select Age
from `apollo_hosptal.hospital`
order by Age
limit 1;
select Age
from `apollo_hosptal.hospital`
order by Age desc
limit 1;

select min(Age) min_age,
      max(Age) max_age,
      round(avg(Age),2) avg_age
from `apollo_hosptal.hospital`;
```

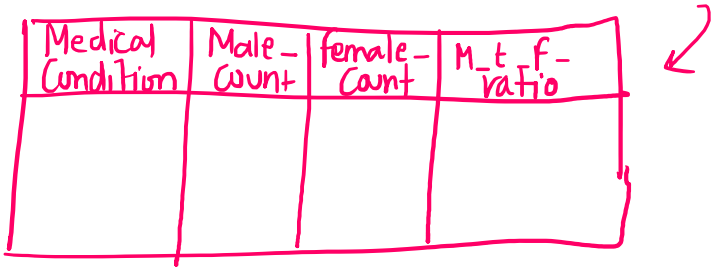
```sql
select round(avg(BMI),2) avg_bmi
from `apollo_hosptal.hospital`
where lower(Medical_Condition) = "obesity";

select count(*) total_patients
from `apollo_hosptal.hospital`;
select count(1) total_patients
from `apollo_hosptal.hospital`;

select count(DISTINCT Medical_Condition) medical_condition
from `apollo_hosptal.hospital`;
select count(Medical_Condition) medical_condition
from `apollo_hosptal.hospital`;


Select Age,count(*) patient_count
from `apollo_hosptal.hospital`
group by Age
order by Age;

Select Insurance_Provider, sum(Billing_Amount) total_claim
from `apollo_hosptal.hospital`
group by Insurance_Provider
order by total_claim desc;


Select Medical_Condition,
	round(avg(Days_Hospitalised),2) avg_days_Hospitalised
from `apollo_hosptal.hospital`
group by Medical_Condition;


Select Hospital, avg(Billing_Amount) avg_billing_Amount
from `apollo_hosptal.hospital`
where lower(Medical_Condition) = "cancer"
group by Hospital
order by avg_billing_Amount desc;

Select Blood_Type, count(*) num_patients
from `apollo_hosptal.hospital`
where lower(Medical_Condition) = "diabetes"
group by Blood_Type;

Select Medical_Condition,
	concat(round(count(*) * 100 / (select count(*) from
`apollo_hosptal.hospital`),2),'%') perct_patients
from `apollo_hosptal.hospital`
group by Medical_Condition;

select COUNT(*) as diabetes_count
from `apollo_hosptal.hospital`; --WRONG
select count(*) num_patients from `apollo_hosptal.hospital`
where Medical_Condition = "Diabetes";
```

Session - 6 Applied SQL 1

02 September 2024 20:37

Q. What is the gender ratio (male to female) for each medical condition?

Output -



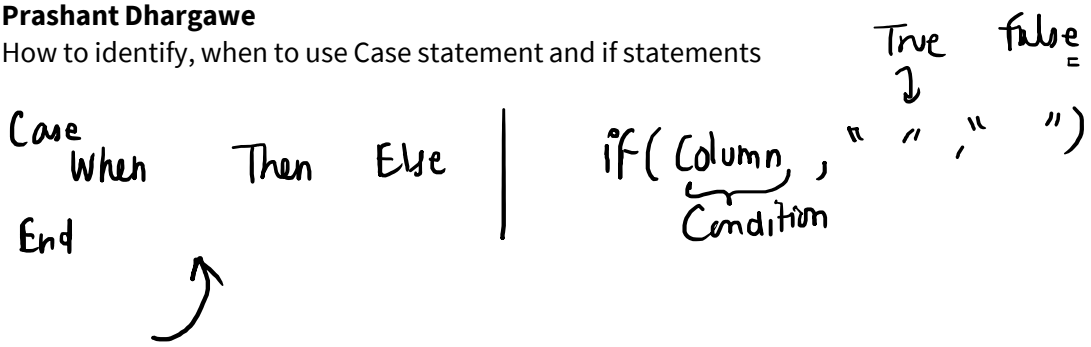
Group By

```
SELECT Medical_Condition,
      SUM(CASE WHEN Gender = "Male" THEN 1 ELSE 0 END) AS
male_count,
      SUM(CASE WHEN Gender = "Female" THEN 1 ELSE 0 END) AS
female_count,
      CASE
       WHEN SUM(CASE WHEN Gender = "Female" THEN 1 ELSE 0
END) > 0
        THEN ROUND(SUM(CASE WHEN Gender = "Male" THEN 1 ELSE
0 END) / SUM(CASE WHEN Gender = "Female" THEN 1 ELSE 0 END),2)
       ELSE NULL
      END as male_female_ratio
FROM `apollo_hosptal.hospital`
GROUP BY Medical_Condition;
```

```
select medical_condition, male_count/female_count from (
SELECT Medical_Condition,
      SUM(CASE WHEN Gender = "Male" THEN 1 ELSE 0 END) AS
male_count,
      SUM(CASE WHEN Gender = "Female" THEN 1 ELSE 0 END) AS
female_count,
FROM `apollo_hosptal.hospital`
GROUP BY Medical_Condition ) t1;
```

Prashant Dhargawe

How to identify, when to use Case statement and if statements



IF(condition, true_value, false_value)

(i) (ii) Subquery

(i) (ii) Subquery

```
Select * from Emp
Where dept = "Admin"
OR dept = "HR"
OR dept = "IT";
```

price
NULL
NULL

Select ifNull(price, "NA")

replace x

Spark

Data → Data → Data → Data pipeline → D.E, D.A, D.S

(✓) → "US", "USA", "U....."

Nancy Aspin
normally subquery used in where condition . when we use subquery in from statement and select statement .

Why ?

Emp: id | Name | dep_id

dept: id | Name

Admin

Output: id | Emp | Dept Name

(i)

IN

```
Select Medical_Condition,
      concat(round(count(*) * 100 / (select count(*) from
`apollo_hosptal.hospital`),2),'%') perct_patients
from `apollo_hosptal.hospital`
group by Medical_Condition;
```

Correlate sub-query ?

22:18

Q2

We want to find all employees who share the **same job** as the employee with employee_id 107. Additionally, you need to create a new column "**full_name**" by concatenating **first_name** and **last_name**, separated by **space**.

- The result should include the columns **full_name**, **salary**, **department_id**, and **job_id**.
- The result should be sorted by **full_name** in **ascending order** for easier reference.

Aggregation X

Query + Subquery

Sanket
it is always the case that we need to use "in" can't we equals(=) when working with where clause and subquery as it is expected that subquery will always return multiple values ?

i) name = "Prakash"

ii) name IN ("Prakash", "Sanket")

≈ name = "prakash" OR name = "Sanket"

iii) Where name = (select name .... limit 1) ;

Where name = (select name ....

Mayur Dilip Rajgurav
How does UPDATE work with multiple conditions in the WHERE clause?

Emp table

```
Select emp_id,
      Case
        When commission_pct is NULL Then 0 Else round(-,2)
        end as commision_pct
 from <table>
Order by (      >
```

Sharique Rahi

1. What is the difference between IN and EXIST operator?**
2. Why order by not working on distinct query like “select distinct columnA from table order by columnA”

1. IN

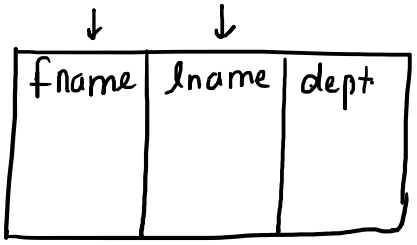
EXIST ← to check if a subquery returns any rows

```
Select <column>
from <table>
Where EXIST (select 1 from <table> where <condition>);
```

2.

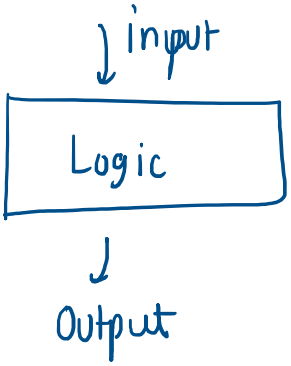
```
select Distinct Medical_Condition
from `apollo_hosptal.hospital`
order by Medical_Condition desc;
```

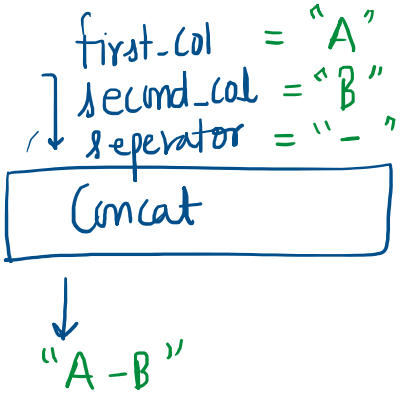
Nancy Aspin
sql inline functions. say about it sir.



⇒

```
Select concat(fname, "-", lname) as full_name
, dept
from <table>
```





- Concat
- round
- substr
- upper
- lower

- sum
- avg
- max
- min
- length.

Aggregation

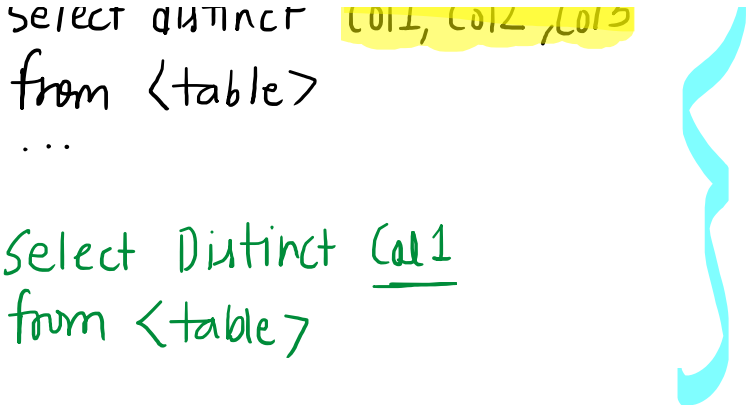
Shaik Vazeer
how to get the longest name in a column

↳ Subquery
↳ Windows

```
Select name
from table
order by length(name) desc
limit 1;
```

Sanket
can we use Select name, distinct age ?

```
Select distinct col1, col2, col3
from <table>
```



In SQL, you cannot refer to an alias directly in the HAVING clause if it was defined in the SELECT clause. This is because SQL processes the HAVING clause before it processes the **SELECT clause where aliases are defined.**

Shaik Vazeer
how to get the longest name in a column - Sub Query

```
SELECT name
FROM <table> WHERE LENGTH(name) = (
        SELECT MAX(LENGTH(name)) FROM <table>
);
```

Session 7 - Group By and Aggregation continued

02 September 2024 23:05

Agenda

1. Problem Statement
2. Group By contd.
3. HAVING clause
4. WHERE vs. HAVING

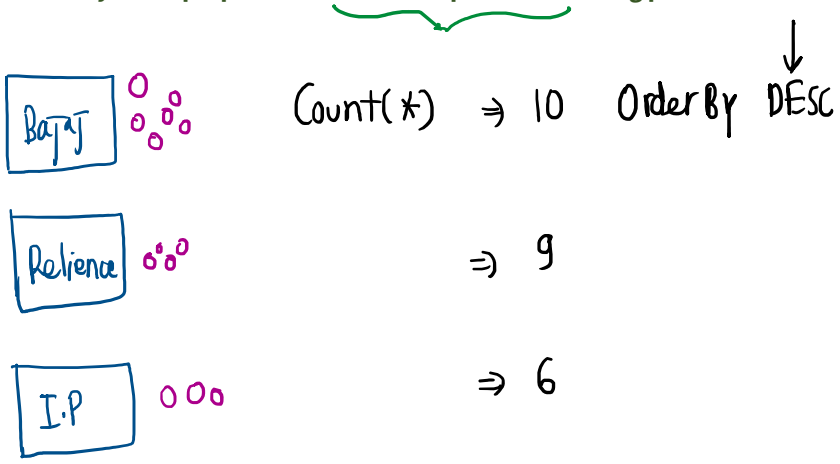
Apollo Hospital Dataset

https://drive.google.com/file/d/18alZG0kYF3fmoFfk7vstNDnHbfNr4H8P/view?usp=drive_link

GROUP BY

"GROUP BY"

Q1. Identify the top 3 preferred insurance providers among patients.




```
select Insurance_Provider,
count(*) patient_count
from `apollo_hosptal.hospital`
group by Insurance_Provider
order by patient_count desc
limit 3;
```

Q2. What are the most common medical conditions among patients aged 60 and above?

Group By Where

```
select Medical_Condition,
      count(*) as patient_count
from `apollo_hosptal.hospital`
where Age >= 60
group by Medical_Condition
order by patient_count desc;
```

Q3. Determine the count of universal blood donors and recipients within the patient population.

Blood type

O -(ve) , AB +

Where blood_type IN("O-", "AB+")


```
select Blood_Type,
      count(*) as num_patients
from `apollo_hosptal.hospital`
where Blood_Type IN ("O-", "AB+")
group by Blood_Type;
```

```
SELECT
  SUM(CASE WHEN Blood_Type = 'O-' THEN 1 ELSE 0 END) AS universal_donor_count,
  SUM(CASE WHEN Blood_Type = 'AB+' THEN 1 ELSE 0 END) AS universal_recipient_count
FROM `apollo_hosptal.hospital`;
```

Q4. How many patients fall into different BMI categories (underweight, normal weight, overweight, obese)?

Count?

```
BMI < 18.5  => 'Underweight'
BMI BETWEEN 18.5 AND 24.9 => 'Normal weight'
BMI BETWEEN 25 AND 29.9 THEN => 'Overweight'
```

Group

```
select
      CASE
        WHEN BMI < 18.5 THEN 'Underweight'
        WHEN BMI BETWEEN 18.5 AND 24.9 THEN 'Normal weight'
        WHEN BMI BETWEEN 25 AND 29.9 THEN 'Overweight'
        ELSE 'Obese'
      END AS weight_status,
      count(*) patient_count
from `apollo_hosptal.hospital`
group by weight_status
order by patient_count desc;
```

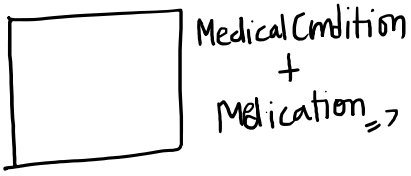
Q5. Which ethnic group exhibits the highest susceptibility to cancer?

Group By

Where

```
SELECT
  Ethnicity,
  count(*) as cancer_patient_cnt
FROM `apollo_hosptal.hospital`
where Medical_Condition = "Cancer"
Group by Ethnicity
order by cancer_patient_cnt desc
limit 1;
```

Q6. Find the most frequently prescribed medication for specific medical conditions.,

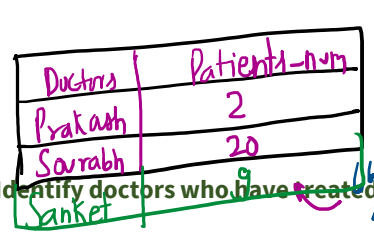


Group By

```
SELECT
    Medical_Condition,
    Medication,
    count(*) prescription_count
FROM `apollo_hosptal.hospital`
group by Medical_Condition,Medication
order by prescription_count desc;
```

Break = 22:23 ✓

Filtering with Having



Group By =

"Having clause is used to filter aggregated data"

Q. Identify doctors who have treated more than 10 patients.

"Aggregated Data"

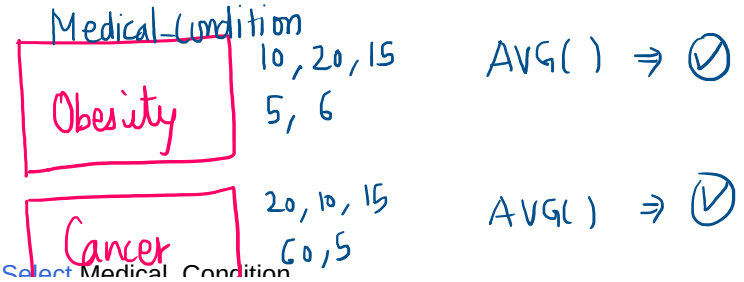
```
Select Doctor,
      count(distinct Name) as num_patients
from `apollo_hosptal.hospital`
group by Doctor
having count(distinct Name) > 10
order by num_patients desc;
```

From
↓
Where
↓
Group By
↓
Aggregate functions
↓
HAVING
↓
SELECT → Order By -

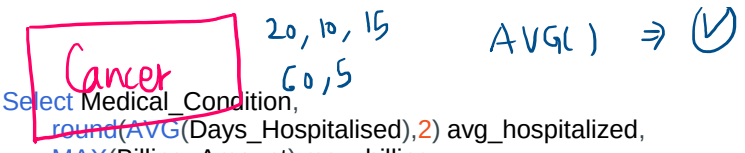
Q. List medical conditions that have an average hospitalization period greater than 15 days and where the maximum billing amount exceeds $25,000.

Group By

AVG ✓

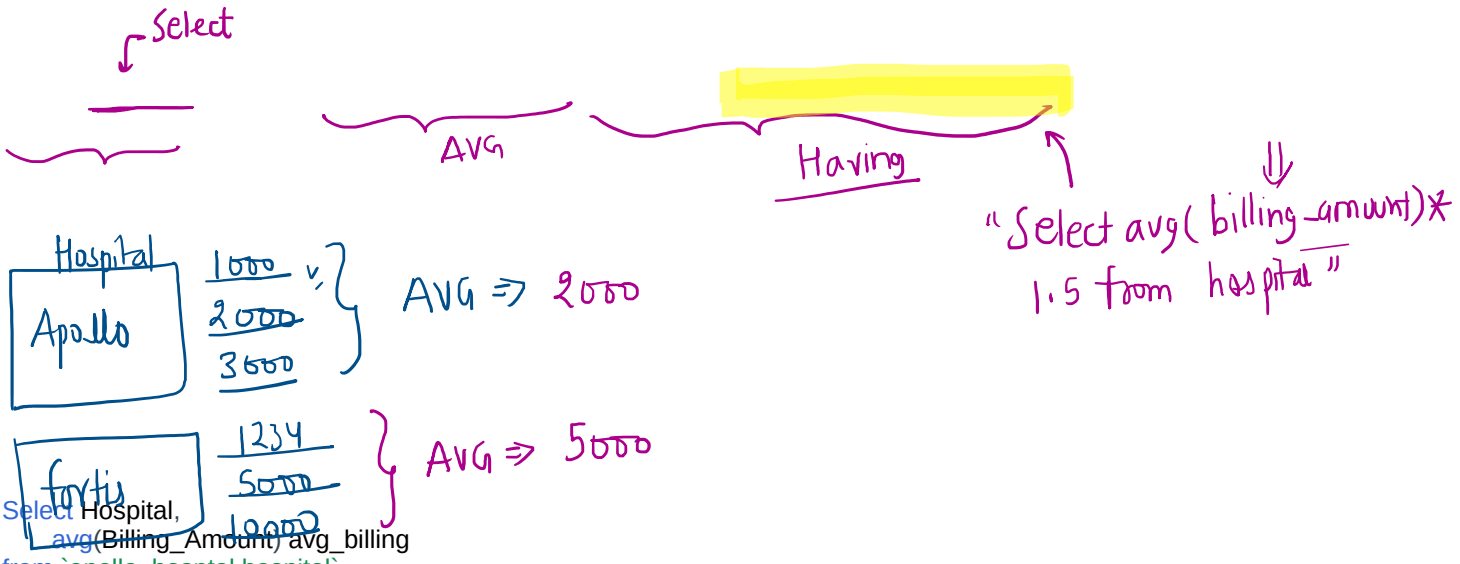


Select Medical_Condition




```
Select Medical_Condition,
    round(AVG(Days_Hospitalised),2) avg_hospitalized,
    MAX(Billing_Amount) max_billing
from `apollo_hosptal.hospital`
group by Medical_Condition
having avg(Days_Hospitalised) > 15 and MAX(Billing_Amount) > 25000;
```

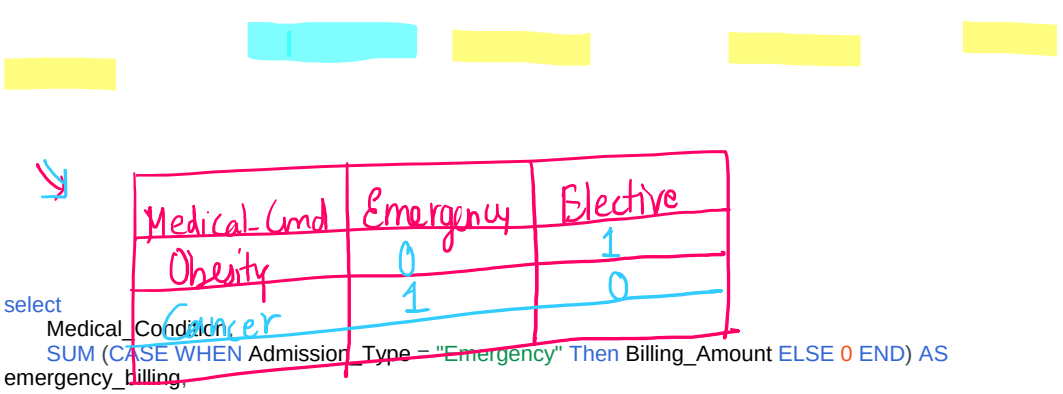
Q. Find hospitals where the average billing amount exceeds the overall average billing amount by at least 50%.




```
Select Hospital,
    avg(Billing_Amount) avg_billing
from `apollo_hosptal.hospital`
group by Hospital
having avg(Billing_Amount) > (
        select avg(Billing_Amount) * 1.5 from `apollo_hosptal.hospital`
);
```

```
select avg(Billing_Amount) * 1.5 from `apollo_hosptal.hospital`;
```

Q. Calculate the total billing amount for emergency and elective admissions for each medical condition. Show only those medical conditions where emergency billing is less than elective billing.




```
select
    Medical_Condition,
    SUM (CASE WHEN Admission_Type = "Emergency" Then Billing_Amount ELSE 0 END) AS
emergency_billing,
    SUM( CASE WHEN Admission_Type = "Elective" Then Billing_Amount ELSE 0 END) AS elective_billing
from `apollo_hosptal.hospital`
group by Medical_Condition
having SUM (CASE WHEN Admission_Type = "Emergency" Then Billing_Amount ELSE 0 END) <
```

```
    SUM( CASE WHEN Admission_Type = "Elective" Then Billing_Amount ELSE 0 END)
order by Medical_Condition;
```

When grouping, we can have a column B in the select statement which is not present in the GROUP BY clause.

Medical_Condition =

FALSE

Select name, medical_condition, count(*)
from table
Group by medical_condition.

Where Having ← filtration after grouping.

↑ filtration before grouping.

Q. Find the doctors who have treated more than 3 cancer patients, and among those doctors, identify the ones whose average billing amount is greater than $25,000.

Where =

Having =

Doctor, Count(*) num_patients, avg(billing_amount)

```
SELECT
   Doctor,
   COUNT(*) AS Number_of_Patients,
   ROUND(AVG(Billing_Amount), 2) AS Average_Billing_Amount
FROM `apollo_hosptal.hospital`
WHERE Medical_Condition = 'Cancer'
GROUP BY Doctor
HAVING COUNT(*) >= 3 AND AVG(Billing_Amount) > 25000
ORDER BY Doctor;
```

Consider a **transactions** table. To find customers who have made an average purchase of more than $200, which clause should be used?

Aggregation (HAVING)

transactions

c_id	Name	sale
1	A	200
1	A	500
1	A	1000
2	B	500
2	B	5000

```
Select c_id, avg(sale)
from transactions
group by c_id;
```

In the **transactions** table, consider every row as a sale value per transaction. We only want to analyze the sale value per transaction above $200. Which clause should we use?

In the **transactions** table, consider every row as a sale value per transaction.
We only want to analyze the sale value per transaction above $200. Which clause should we use?

transactions

id	Name	sale
1	iphone	100
2	Samsung	200

Select * from transactions
Where sale > 200;

"WHERE"

group by c_id;

c_id	avg_sale
1	300
2	2500

having avg_sale > 300;

QUERIES EXECUTED IN CLASS

```
select Insurance_Provider,
count(*) patient_count
from `apollo_hosptal.hospital`
group by Insurance_Provider
order by patient_count desc
limit 3;

select Medical_Condition,
      count(*) as patient_count
from `apollo_hosptal.hospital`
where Age >= 60
group by Medical_Condition
order by patient_count desc;

select Blood_Type,
      count(*) as num_patients
from `apollo_hosptal.hospital`
where Blood_Type IN ("O-", "AB+")
group by Blood_Type;

SELECT
  SUM(CASE WHEN Blood_Type = 'O-' THEN 1 ELSE 0 END) AS universal_donor_count,
  SUM(CASE WHEN Blood_Type = 'AB+' THEN 1 ELSE 0 END) AS universal_recipient_count
FROM `apollo_hosptal.hospital`;

select
      CASE
        WHEN BMI < 18.5 THEN 'Underweight'
        WHEN BMI BETWEEN 18.5 AND 24.9 THEN 'Normal weight'
        WHEN BMI BETWEEN 25 AND 29.9 THEN 'Overweight'
        ELSE 'Obese'
      END AS weight_status,
      count(*) patient_count
from `apollo_hosptal.hospital`
group by weight_status
order by patient_count desc;

SELECT
  Ethnicity,
  count(*) as cancer_patient_cnt
FROM `apollo_hosptal.hospital`
where Medical_Condition = "Cancer"
Group by Ethnicity
order by cancer_patient_cnt desc
limit 1;

SELECT
    Medical_Condition,
    Medication,
    count(*) prescription_count
FROM `apollo_hosptal.hospital`
group by Medical_Condition,Medication
order by prescription_count desc;


Select Doctor,
      count(distinct Name) as num_patients
from `apollo_hosptal.hospital`
group by Doctor
having count(distinct Name) > 10
order by num_patients desc;

Select Medical_Condition,
```

```
    round(AVG(Days_Hospitalised),2) avg_hospitalized,
    MAX(Billing_Amount) max_billing
from `apollo_hosptal.hospital`
group by Medical_Condition
having avg(Days_Hospitalised) > 15 and MAX(Billing_Amount) > 25000;



Select Hospital,
    avg(Billing_Amount) avg_billing
from `apollo_hosptal.hospital`
group by Hospital
having avg(Billing_Amount) > (
        select avg(Billing_Amount) * 1.5 from `apollo_hosptal.hospital`
);

select avg(Billing_Amount) * 1.5 from `apollo_hosptal.hospital`;


select
    Medical_Condition,
    SUM (CASE WHEN Admission_Type = "Emergency" Then Billing_Amount ELSE 0 END) AS
emergency_billing,
    SUM( CASE WHEN Admission_Type = "Elective" Then Billing_Amount ELSE 0 END) AS
elective_billing
from `apollo_hosptal.hospital`
group by Medical_Condition
having SUM (CASE WHEN Admission_Type = "Emergency" Then Billing_Amount ELSE 0 END)
<
    SUM( CASE WHEN Admission_Type = "Elective" Then Billing_Amount ELSE 0 END)
order by Medical_Condition;
```

Session 8 - JOINS

06 September 2024 17:35

"Class will start at 09:05"

1. Joins - 6th sep ✓
2. Joins Cntd... - 9th sep ✓
3. Doubt clearing session - 11th sep

Agenda

1. Problem Statement
2. Intro to Joins
3. Types of Joins
   a. INNER JOIN
   b. LEFT JOIN
   c. RIGHT JOIN
   d. FULL OUTER JOIN
      - Union
      - UNION DISTINCT
      - UNION ALL

DATASET : CineFlix Entertainment Store

LINK: https://drive.google.com/drive/folders/1df47rqe6XrhLX9lwIOPpLQwcSHmwR7xb

CineFlix Entertainment Store is a premier rental service provider specialising in CDs, DVDs, and Blu-rays. With a wide range of movies.

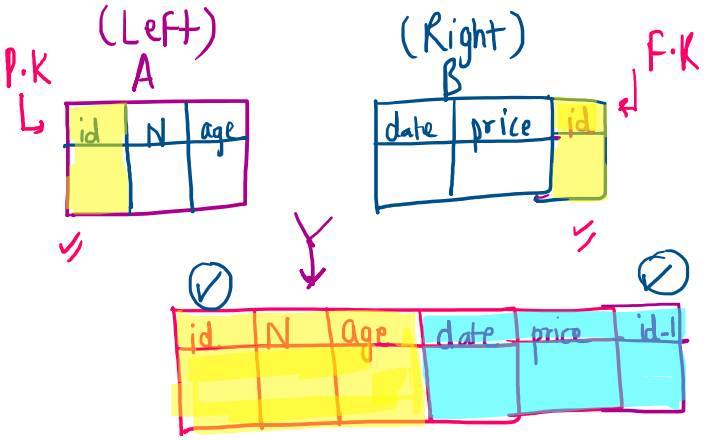
Challenges: CineFlix is facing increased competition from digital streaming platforms, which offer convenience and instant access to content.

CineFlix needs to:

1. Understand Rental Patterns.
2. Customer Behavior Analysis.
3. Store Performance.
4. Inventory Management.

Intro to Joins

"JOIN"



1. INNER JOIN /JOIN ✓
2. LEFT JOIN (Left outer)
3. RIGHT JOIN (Right Outer Join)
3. FULL JOIN (Full outer)
4. Self join
5. Cross join

3. FULL JOIN (Full outer)

Table: product

p_id	p_name	p_size	p_cat_id	p_qty_type
1	Carrots	sold by weight	1	Lbs
2	Maple Syrup - Jar	8 oz	2	Unit
3	Eggs	1 dozen	6	Unit
4	Red Bull	small		

Table: product_category

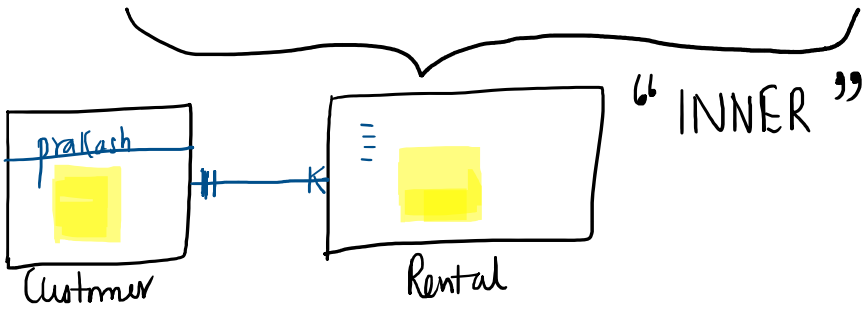
p_cat_id	p_cat_name
1	Fresh Fruits & Vegetables
2	Packaged Pantry Goods
6	Eggs & Meat (Fresh or Frozen)
8	Beverages

p_id	p_name	p_size	p_cat_id	p_qty_type	p_cat_id_1	p_cat_name
1	Carrot	-	1	Lbs	1	fresh fruits...
2	Maple	-	2	Unit	2	Packag...
3	Eggs	-	6	Unit	6	Eggs..

p_id	p_name	p_size	p_cat_id	p_qty_type	p_cat_id_1	p_cat_name
1	Carrot	-	1	Lbs	1	fresh fruits...
2	Maple	-	2	Unit	2	Packag...
3	Eggs	-	6	Unit	6	Eggs..
4	RedBull	-	NULL	NULL	NULL	NULL

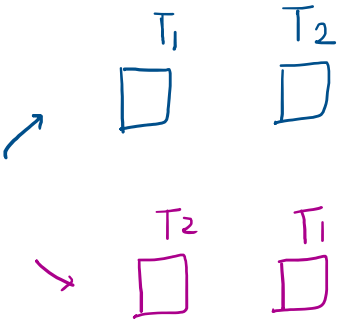
INNER JOIN:

Question: Get a list of all customers who have rented more films than the average number of rentals.




```
select c.customer_id, c.first_name, c.last_name, count(*)
num_rentals
from `cineflix.customer` c JOIN  `cineflix.rental` r
ON c.customer_id = r.customer_id
group by c.customer_id, c.first_name, c.last_name
having count(*) > 27
order by num_rentals desc;
```

```
select c.customer_id, c.first_name, c.last_name, count(*)
num_rentals
from `cineflix.rental` r JOIN `cineflix.customer` c
ON r.customer_id = c.customer_id
group by c.customer_id, c.first_name, c.last_name
having count(*) > 27
order by num_rentals desc;
```



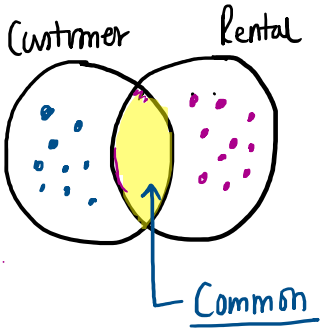

```
select c.customer_id, c.first_name, c.last_name, count(*)
num_rentals
from `cineflix.customer` c JOIN `cineflix.rental` r
USING (customer_id)
group by c.customer_id, c.first_name, c.last_name
having count(*) > 27
order by num_rentals desc;
```

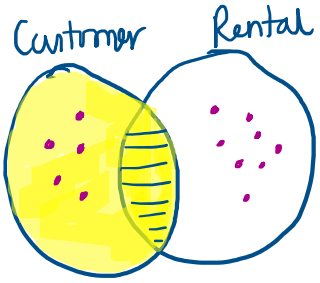
```
select c.customer_id, c.first_name, c.last_name, count(*)
num_rentals
from `cineflix.customer` c JOIN `cineflix.rental` r
ON c.customer_id = r.customer_id
group by c.customer_id, c.first_name, c.last_name
having count(*) > (select round(avg(num_rentals)) avg_rental
from (select count(*) as num_rentals from `cineflix.rental` r group
by customer_id))
order by num_rentals desc;
```

QUIZ: Both of the queries will give the same result.

Select t1.*, t2.* from DB.tbl1 t1 join DB.tbl2 t2 on t1.col1 = t2.col1

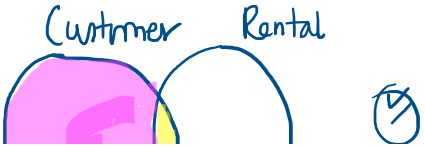
Select t1.*, t2.* from DB.tbl2 t2 join DB.tbl1 t1 on t2.col1 = t1.col1

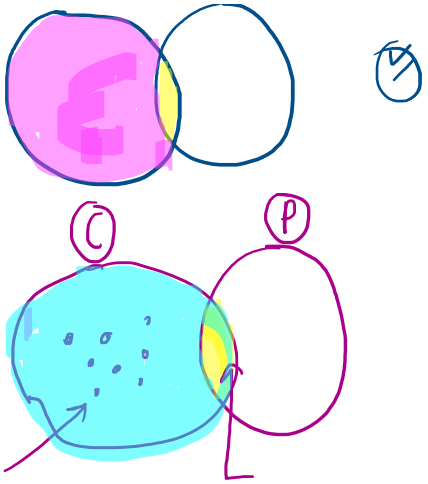




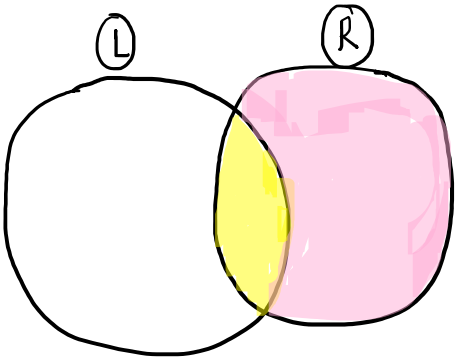
Left Join

Question: List all customers and the total amount they have spent on rentals, including customers who have never rented a film.

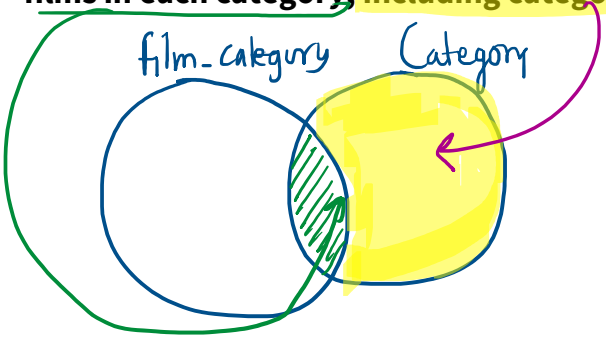




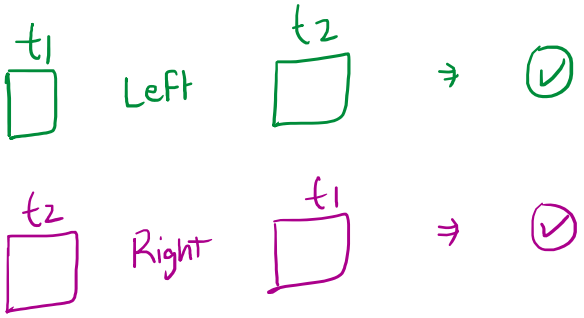
Rigth JOIN

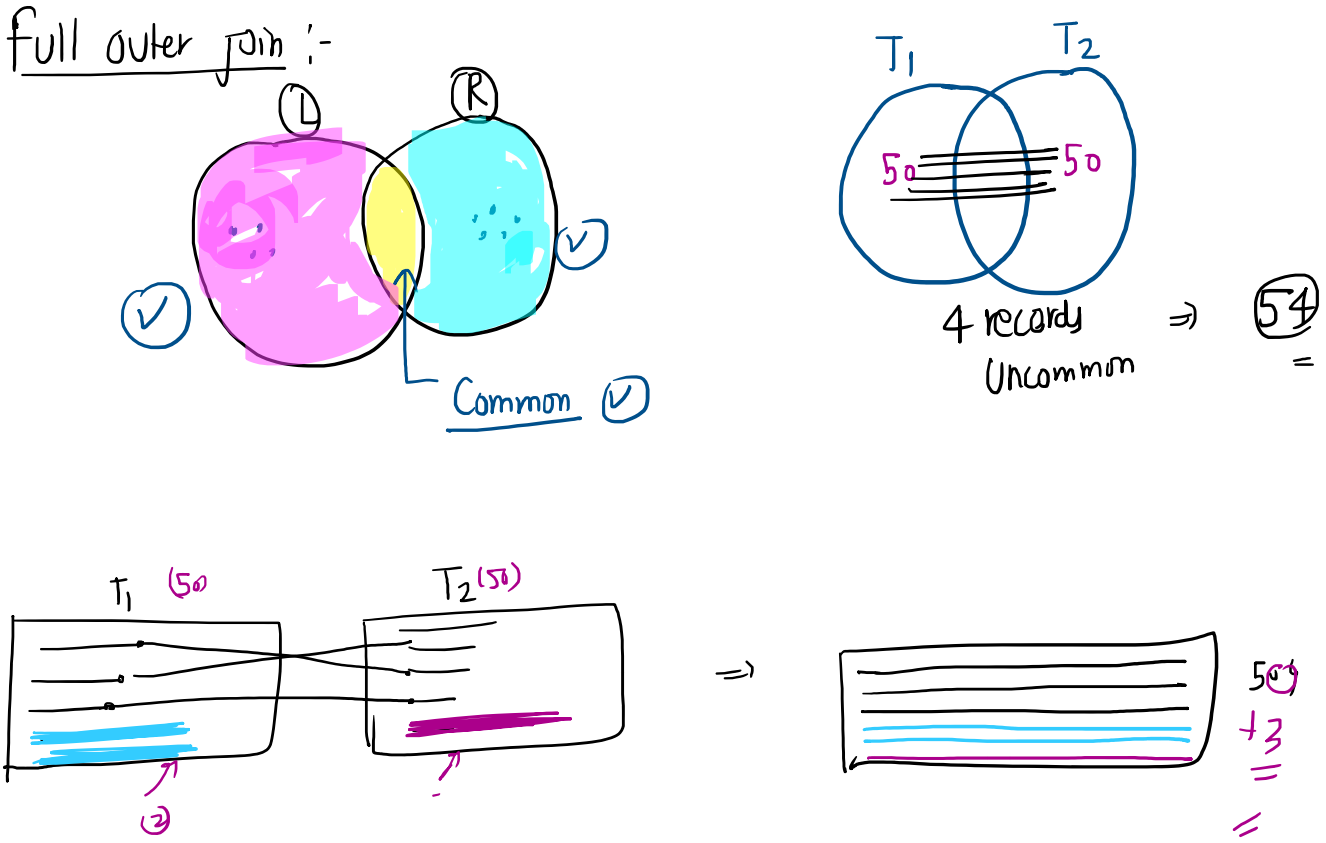


Question: List all film categories along with the number of films in each category, including categories with no films.

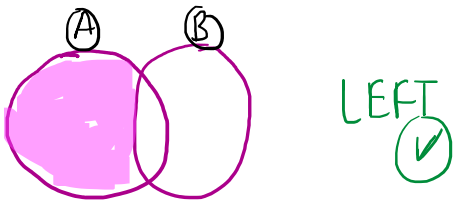


Q. Any result that is achieved with right join can also be achieved with left join by just swapping the tables





Q. Which join is used to return the book list which only Jim has read (Jim: Table A and Vinny: Table B)



Q. A join is performed between two tables and the resultant table contains every row on the left table, every row on the right table, and the common rows as well. What join, is it?

https://www.scaler.com/hire/test/problem/41071/

```
select distinct buyer_id
from Sales s join Product p
ON s.product_id = p.product_id
where product_name = "S8"
```

```
    and buyer_id not in (
        select distinct buyer_id
        from Sales s join Product p
        ON s.product_id = p.product_id
        where product_name = "iPhone"
    )
    order by buyer_id;
```

QUERIES EXECUTED IN CLASS

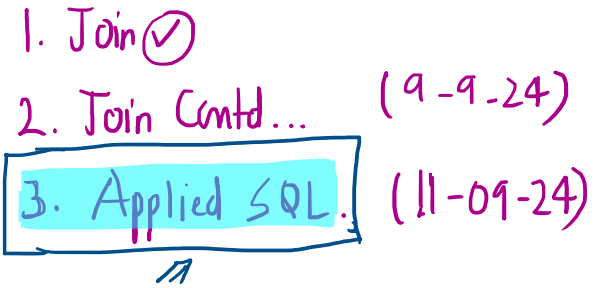
```
select c.customer_id, c.first_name, c.last_name, count(*) num_rentals
from `cineflix.customer` c JOIN  `cineflix.rental` r
ON c.customer_id = r.customer_id
group by c.customer_id, c.first_name, c.last_name
having count(*) > 27
order by num_rentals desc;

select c.customer_id, c.first_name, c.last_name, count(*) num_rentals
from `cineflix.rental` r JOIN `cineflix.customer` c
ON r.customer_id = c.customer_id
group by c.customer_id, c.first_name, c.last_name
having count(*) > 27
order by num_rentals desc;


select c.customer_id, c.first_name, c.last_name, count(*) num_rentals
from `cineflix.customer` c JOIN  `cineflix.rental` r
USING (customer_id)
group by c.customer_id, c.first_name, c.last_name
having count(*) > 27
order by num_rentals desc;

select c.customer_id, c.first_name, c.last_name, count(*) num_rentals
from `cineflix.customer` c JOIN  `cineflix.rental` r
ON c.customer_id = r.customer_id
group by c.customer_id, c.first_name, c.last_name
having count(*) > (select round(avg(num_rentals)) avg_rental from (select count(*) as
num_rentals from `cineflix.rental` r group by customer_id))
order by num_rentals desc;


-- LEFT JOIN EXAMPLE
select c.customer_id, c.first_name, c.last_name, count(*) num_rentals
from `cineflix.customer` c LEFT JOIN  `cineflix.rental` r
ON c.customer_id = r.customer_id
group by c.customer_id, c.first_name, c.last_name
having count(*) > 27
order by num_rentals desc;

select c.customer_id,
      c.first_name,
      c.last_name,
      round(ifnull(sum(p.amount),0),2) as total_amount
from `cineflix.customer` c LEFT JOIN  `cineflix.payment` p
ON c.customer_id = p.customer_id
group by c.customer_id, c.first_name, c.last_name;

-- RIGHT JOIN
select c.customer_id,
      c.first_name,
      c.last_name,
      round(ifnull(sum(p.amount),0),2) as total_amount
from `cineflix.payment` p RIGHT JOIN `cineflix.customer` c
ON c.customer_id = p.customer_id
group by c.customer_id, c.first_name, c.last_name;

-- FULL JOIN
select c.customer_id,
      c.first_name,
      c.last_name,
      round(ifnull(sum(p.amount),0),2) as total_amount
from `cineflix.payment` p FULL JOIN `cineflix.customer` c
ON c.customer_id = p.customer_id
group by c.customer_id, c.first_name, c.last_name;
```

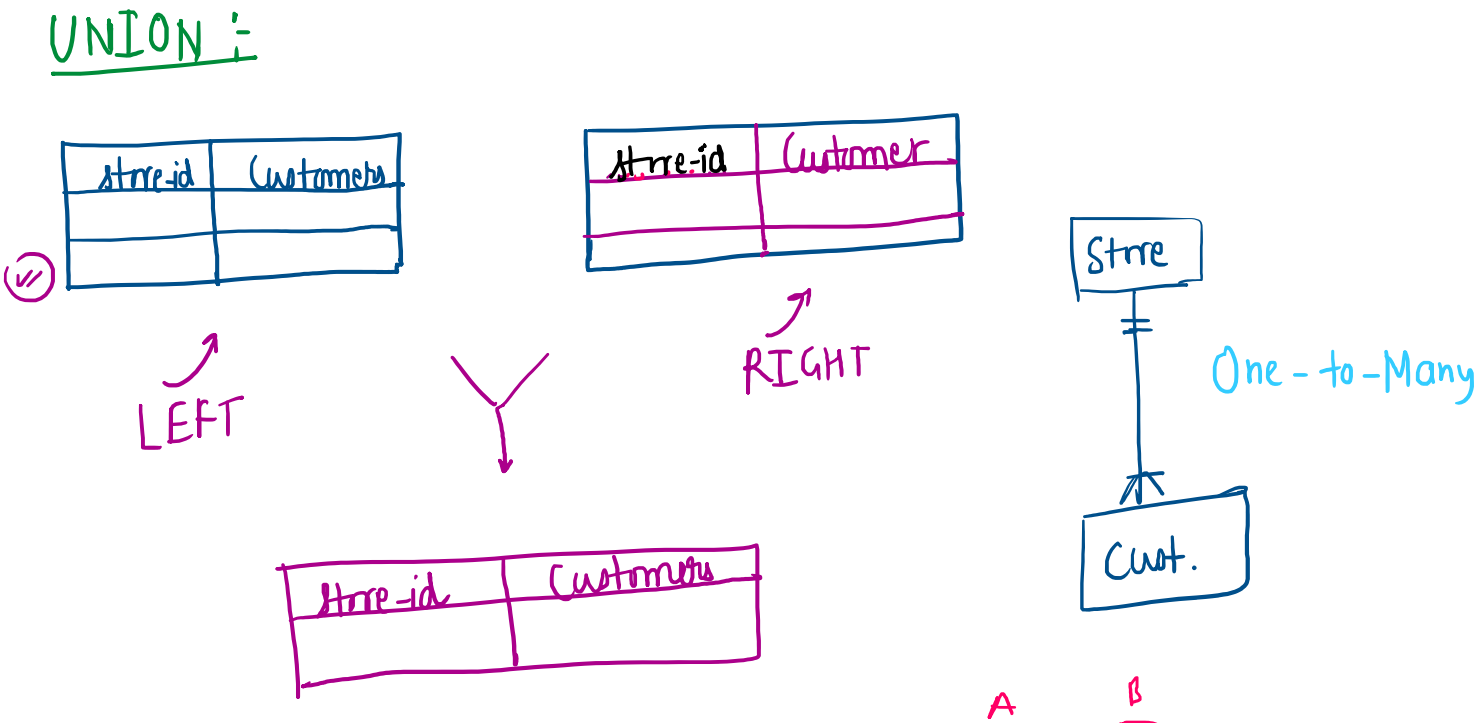
Session 9 - JOINS Contd...

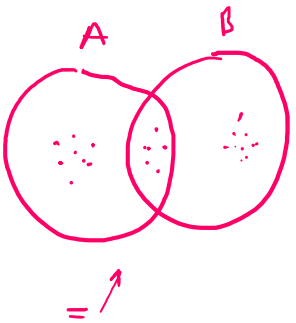
08 September 2024 18:57

Agenda

1. Joining multiple tables
2. Self JOIN
3. Cross JOIN
4. Non-Equi JOINS







UNION

```
select s.store_id,
      count(c.customer_id) AS customer_cnt
from `cineflix.store` as s LEFT JOIN `cineflix.customer` c
ON s.store_id = c.store_id
group by s.store_id
```

UNION DISTINCT

```
select s.store_id,
      count(c.customer_id) AS customer_cnt
from `cineflix.store` as s RIGHT JOIN `cineflix.customer` c
ON s.store_id = c.store_id
group by s.store_id;
```

FULL OUTER JOIN

```
select s.store_id,
      count(c.customer_id) AS customer_cnt
```

```
from `cineflix.store` as s FULL OUTER JOIN `cineflix.customer` c
ON s.store_id = c.store_id
group by s.store_id;
```

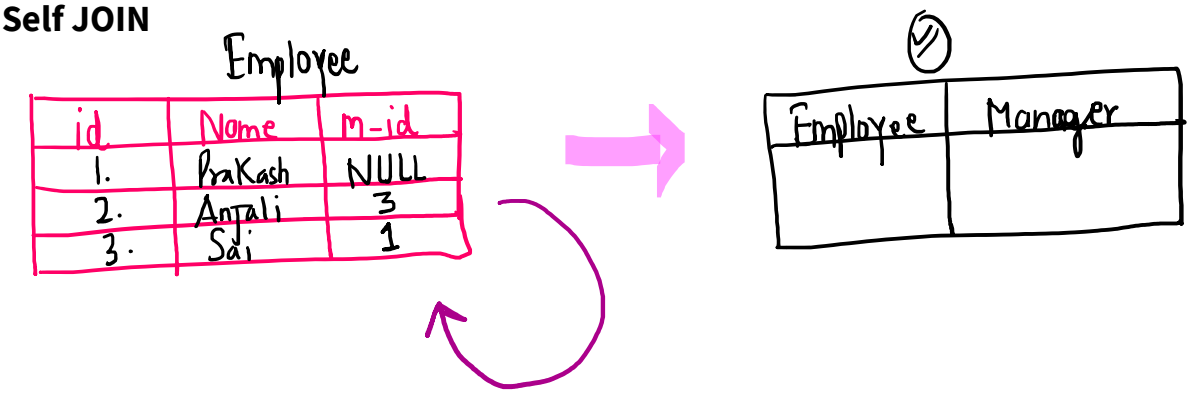
JOINs with more than two tables

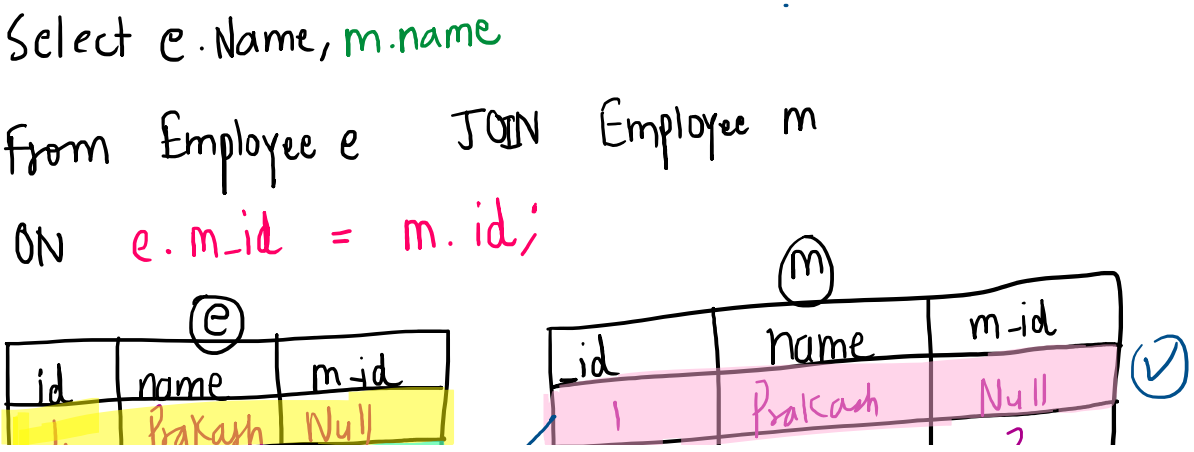
Question: For each store, list the total number of films and the number of currently rented films.

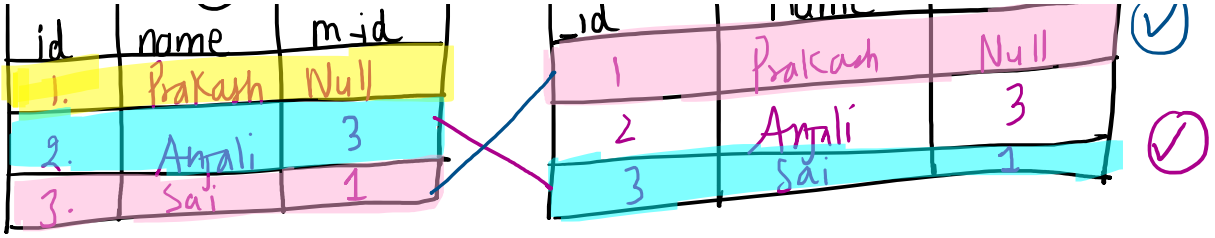
Tables Required :
Store -> Customer -> Rental -> Inventory -> Film

```
SELECT
    s.store_id,
    count(f.film_id) as total_films,
    count(case when r.return_date IS NULL AND inv.inventory_id IS NOT NULL then 1 else NULL end) as
rented_films
from `cineflix.store` s
LEFT JOIN `cineflix.customer` c
    ON s.store_id = c.store_id
LEFT JOIN `cineflix.rental` r
    ON c.customer_id = r.customer_id
LEFT JOIN `cineflix.inventory` inv
    ON r.inventory_id = inv.inventory_id
LEFT JOIN `cineflix.film` f
    ON f.film_id = inv.film_id
GROUP BY s.store_id;
```

Self JOIN





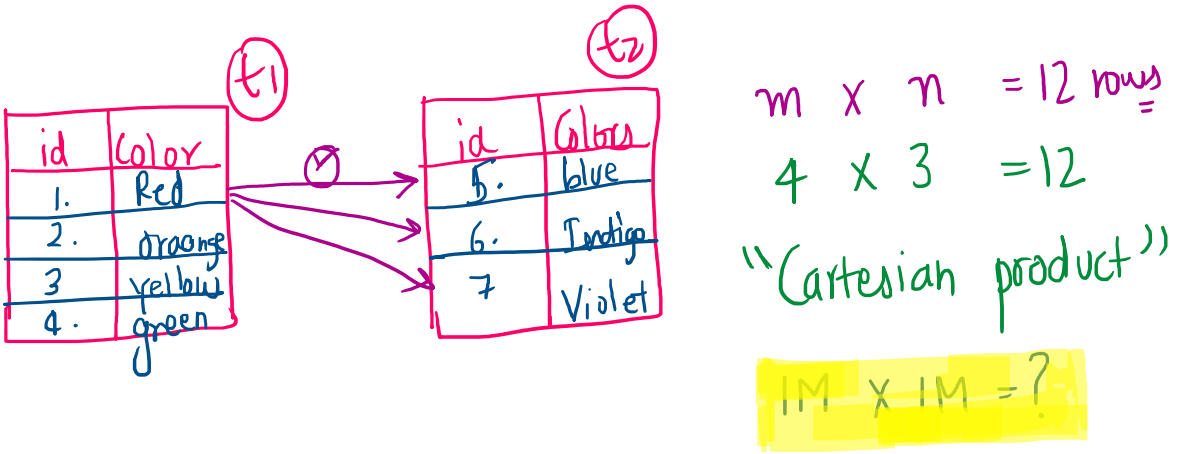


Question: Find the customers and how many pairs of rentals they have, where the same customer rented both rentals and later returned.

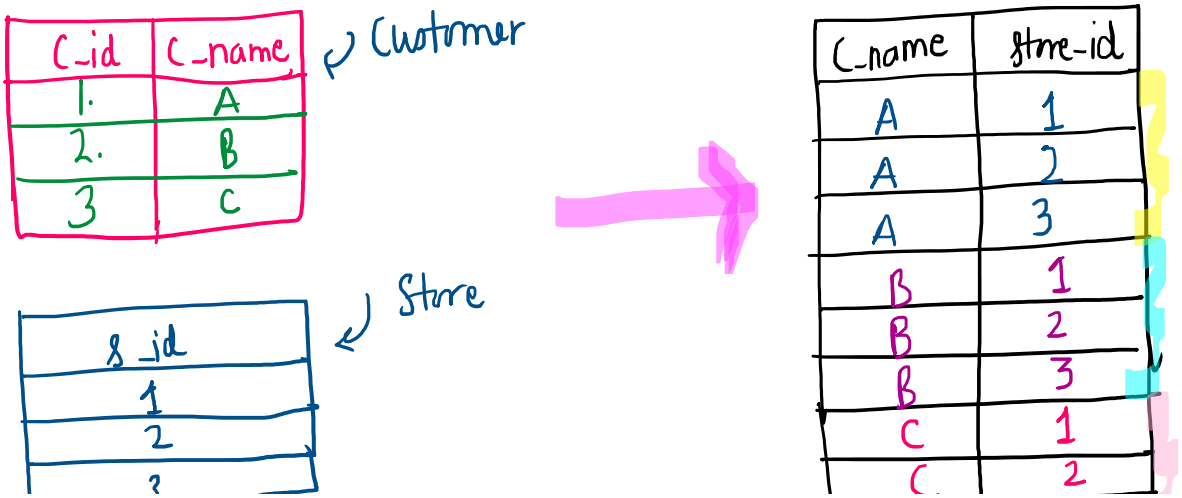
```
select
      r1.customer_id,
      count(*) pair_count
from `cineflix.rental` r1 JOIN `cineflix.rental` r2
ON r1.customer_id = r2.customer_id
AND r1.rental_date < r2.rental_date
AND r1.return_date IS NOT NULL
AND r2.return_date IS NOT NULL
GROUP by r1.customer_id
order by pair_count desc;
```

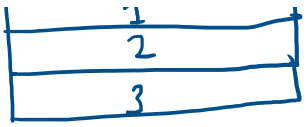
BREAK TIME TILL : 22:26

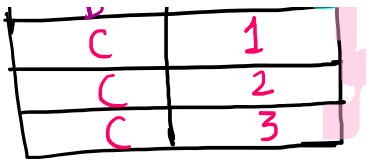
Cross JOIN



Question: List all possible pairs of customers and stores.

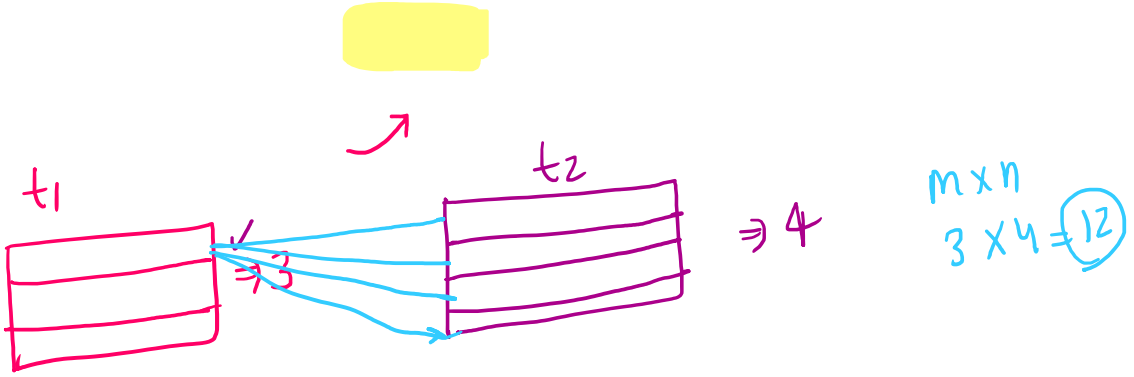




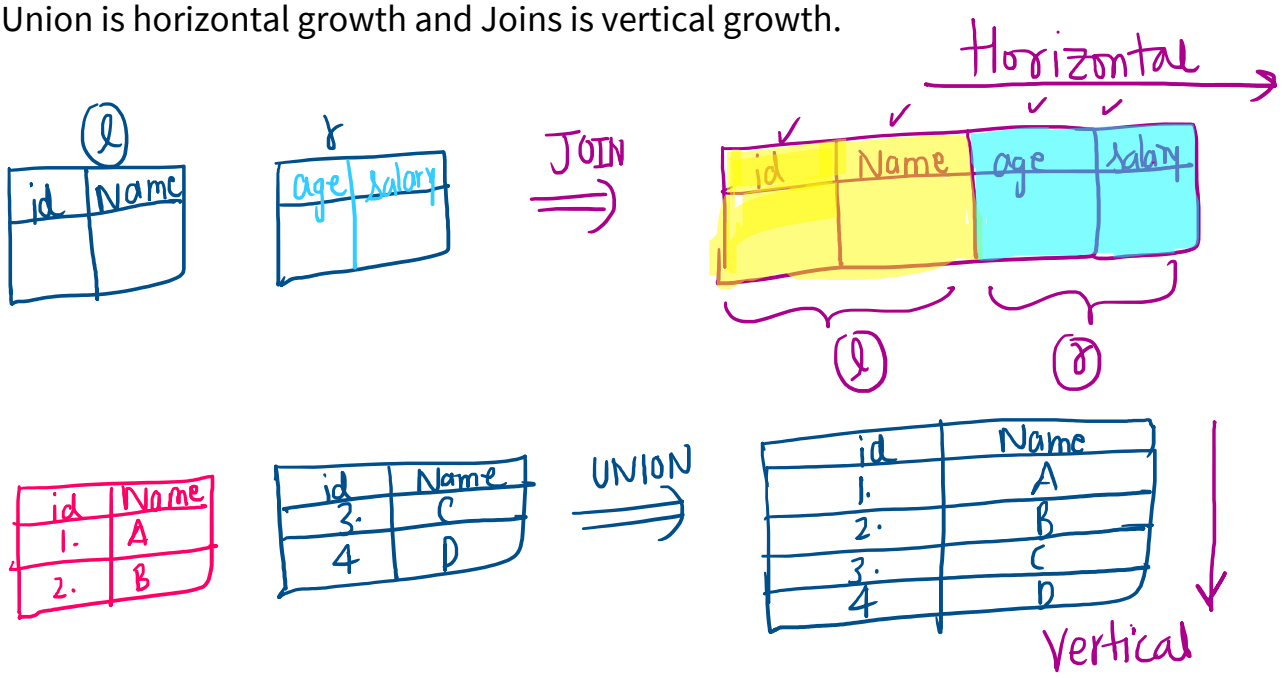



```
SELECT
    c.first_name,
    s.store_id
FROM `cineflix.customer` c CROSS JOIN `cineflix.store` s
order by c.first_name;
```

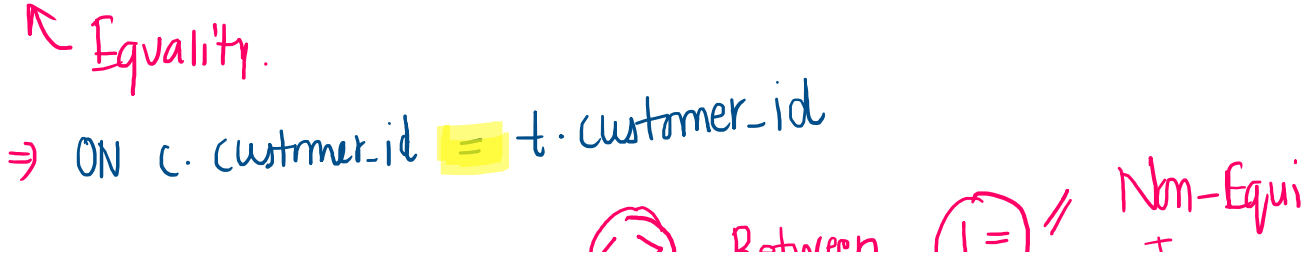
Q. Table1 has 3 rows and Table2 has 4 rows.
What is the output of the cross-join of these 2 tables?



Q. Union is horizontal growth and Joins is vertical growth.



Equi & Non-equi Joins



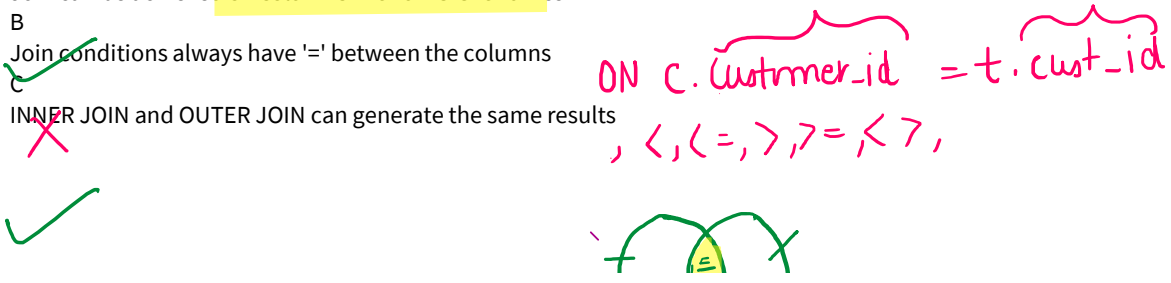
Question 1: List all films with rental rates higher than the average rental rate.

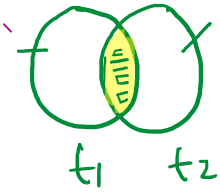

```
select f.film_id,
       f.title,
       f.rental_rate
from `cineflix.film` f
JOIN (
     select avg(rental_rate) rental_rate
     from `cineflix.film`
) avg_table
ON f.rental_rate > avg_table.rental_rate
order by f.rental_rate desc;
```

Question 2: For each customer, identify the names of other customers who have rented the same inventory item(s) at least once **(H.W)**

Q. Select the incorrect statement about joins.

A
Join can be achieved on columns with different names
B
Join conditions always have '=' between the columns
C
INNER JOIN and OUTER JOIN can generate the same results



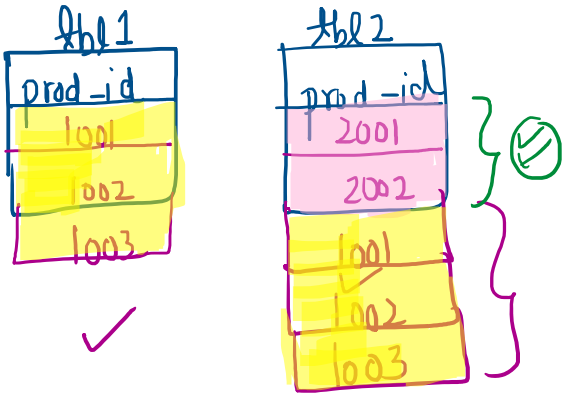


= Equi Join

<, <=, >, >=, <>, between Non-Equi Join

Q. Table 1 contains products the customer has already purchased. Table 2 is the product table. Which type of join to use to recommend products the customer does not already have

A
Select * from DB.tbl1 = DB.tbl2 on product_id = product_id
B
Select * from DB.tbl1 = DB.tbl2 on product_id > product_id
C
Select * from DB.tbl1 = DB.tbl2 on product_id <> product_id
D
Select * from DB.tbl1 = DB.tbl2 on product_id <= product_id




```
Select prod_id
from tbl1 JOIN tbl2
ON tbl.prod_id <> tbl2.prod_id;
```

2001 and 2002

Akifa Iram :
Can I use = and use condition in where such as <, > ,<= so on as a replacement for non equi join

Session 9 - UNION

09 September 2024 20:36

The UNION operator in SQL is used to combine the results of two or more SELECT queries into a single result set. It removes duplicate rows from the result set. If you want to include duplicates, you can use UNION ALL.
Here's a basic example to illustrate how UNION works.

Example

Suppose you have two tables:

Table 1: employees_2023

employee_id	name	department
1	Alice	HR
2	Bob	IT
3	Charlie	IT

Table 2: employees_2024

employee_id	name	department
4	Dave	Marketing
5	Eve	HR
6	Frank	IT

If you want to get a combined list of all employees from both years, you can use UNION.

SQL Query

```
SELECT employee_id, name, department
FROM employees_2023

UNION

SELECT employee_id, name, department
FROM employees_2024;
```

Result

The result set will include all employees from both tables, with duplicates removed:

employee_id	name	department
1	Alice	HR
2	Bob	IT
3	Charlie	IT
4	Dave	Marketing
5	Eve	HR

6 Frank IT

In this result, you see a combined list of employees, but any duplicate rows (if there were any) are removed.

NOTE: Using UNION ALL

If you want to include all rows, including duplicates, you can use UNION ALL

Session 9 - SELF JOIN

09 September 2024 18:23

A self-join is a type of join in SQL where a table is joined with itself. This is useful for querying hierarchical data or finding relationships within the same table. Here's a simple example to illustrate how it works.
Suppose you have an employees table structured like this:

employee_id	name	manager_id
1	Alice	NULL
2	Bob	1
3	Charlie	1
4	Dave	2
5	Eve	2

In this table:

- employee_id is the unique identifier for each employee.
- name is the employee's name.
- manager_id is the employee_id of the employee's manager.

If you want to get a list of employees along with the name of their manager, you can use a self-join. Here's how you can do it:

```
SELECT  e1.name AS employee_name,
        e2.name AS manager_name
FROM employees e1
LEFT JOIN employees e2
ON e1.manager_id = e2.employee_id;
```

In this query:

- e1 is an alias for the first instance of the employees table.
- e2 is an alias for the second instance of the employees table (representing the managers).
- The LEFT JOIN ensures that you get all employees even if they don't have a manager (e.g., Alice, who has a NULL manager_id).

RESULT

employee_name	manager_name
Alice	NULL
Bob	Alice
Charlie	Alice
Dave	Bob
Eve	Bob

NOTE: This shows each employee along with their manager's name, or NULL if they don't have a manager.

Session 9 - CROSS JOIN

09 September 2024 18:31

A CROSS JOIN produces the Cartesian product of two tables, meaning it returns every possible combination of rows from the two tables. This can be useful for generating combinations or for specific calculations where every pair of rows needs to be considered.

Here's a basic example to illustrate how a CROSS JOIN works.

Example

Suppose you have two tables:

Table 1: colors

color_id	color_name
1	Red
2	Green
3	Blue

Table 2: sizes

size_id	size_name
1	Small
2	Medium
3	Large

If you want to get a list of all possible combinations of colors and sizes, you can use a CROSS JOIN.

SQL Query

```
SELECT   c.color_name,
         s.size_name
FROM colors c
CROSS JOIN sizes s;
```

Result

This query will produce a result set where every color is paired with every size:

color_name	size_name
Red	Small
Red	Medium
Red	Large
Green	Small
Green	Medium
Green	Large

Blue	Small
Blue	Medium
Blue	Large

Here, each row from the colors table is combined with each row from the sizes table, resulting in a total of 3 colors × 3 sizes = 9 rows in the output.

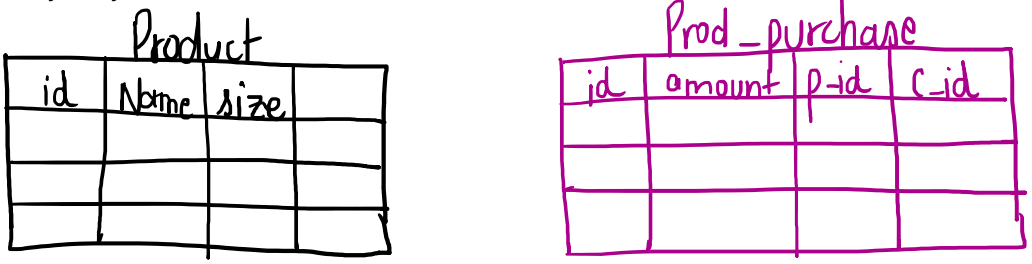
Note

Be cautious when using CROSS JOIN, especially with large tables, as it can produce a very large number of rows, which might impact performance and readability.

Session 10 - Applied SQL 2

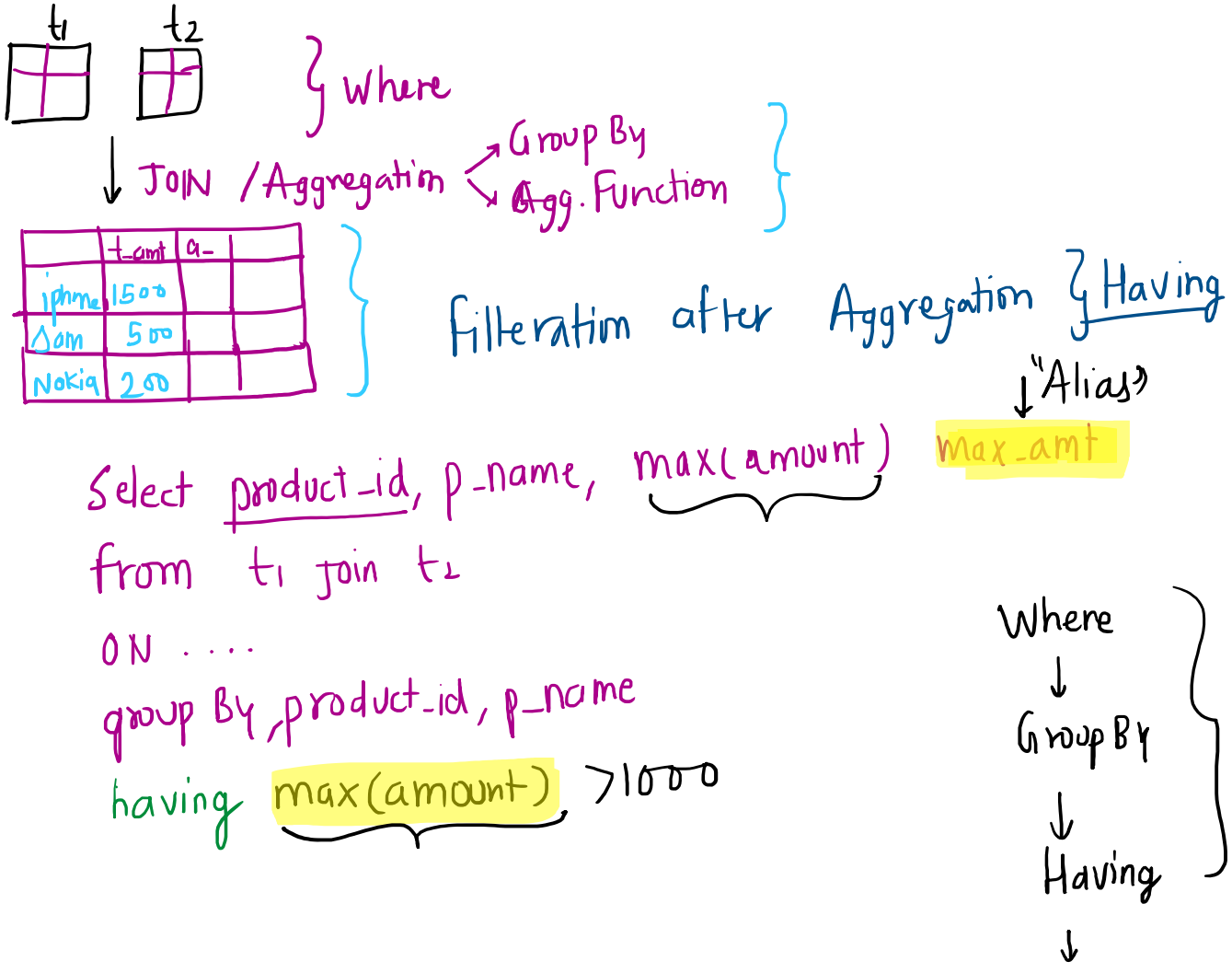
11 September 2024 17:07

Sanket
i am having problem of understanding in **group by and filtering data** for the group using having .....from session "Joins" question 8
why i cant use this query : SELECT s.buyer_id from Sales s
left join Product p on s.product_id = p.product_id
group by s.buyer_id
having p.product_name <> "iPhone"
order by s.buyer_id



Return all medium size products ? Where size = "Medium"

Return products where total purchase amount in greater than 1000 ?



Select

Prob2 : Self-Join :-

Employee

e_id	e_name	m_id
1.	Prakash	NULL
2.	Sanket	3
3	Kiran	1

⇒

Employee	Manager
Sanket	Kiran
Kiran	Prakash

Self-Join

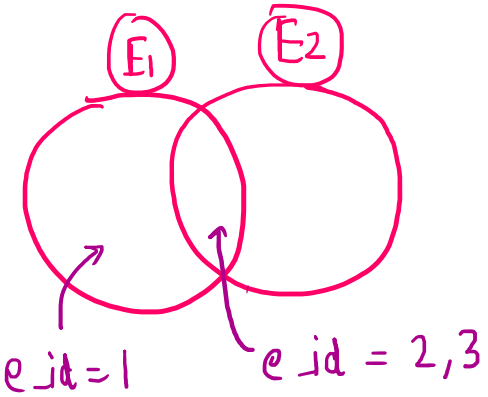
E1

e_id	e_name	m_id
1.	Prakash	NULL
2.	Sanket	3
3	Kiran	1

x

E2

e_id	e_name	m_id
1.	Prakash	NULL
2.	Sanket	3
3	Kiran	1



Sourabh Yadav
session 9 question 6

x	y
-1	-1
0	0
-1	-2

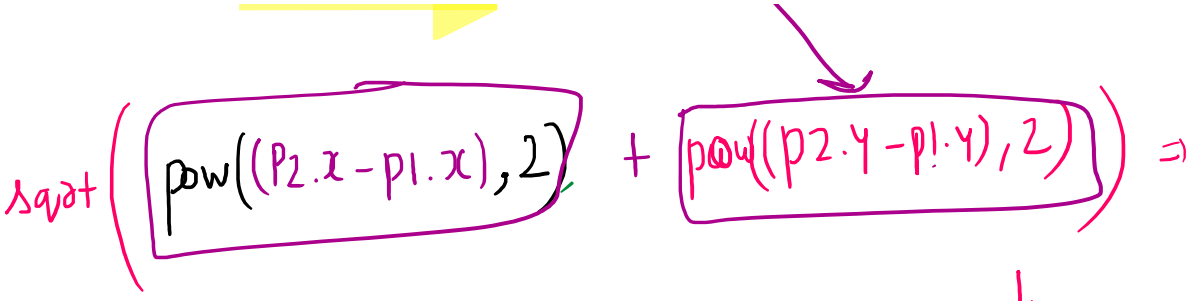
(x, y)

⇒ (-1,-1) (0,0)
(-1,-1) (-1,-2)
(0,0) (-1,-2)

$$\sqrt{(x_2-x_1)^2+(y_2-y_1)^2} =$$

=

```
round(sqrt(pow((p2.x - p1.x),2) + pow((p2.y - p1.y),2)),2) as shortest
```



pow(2,2) ⇒ 2^2

pow(10,2) ⇒ 10^2

round(100.243, 2) ⇒ 100.24

sqrt(100) = 10

sqrt(4) = 2

x	y
-1	-1
0	0
-1	-2

3 x 3 = 9

(-1,-1) (0,0)

(-1,-1) (-1,-2)

(0,0) (-1,-1)

(0,0) (-1,-2)

(-1,-2) (. . .

(≡)

x	y	x	y
-1	-1	-1	-1
0	0	0	0
-1	2	-1	-2
P1		P2	

Where NOT

(P2.x = P1.x AND

P2.y = P1.y)

P1.x < P2.x

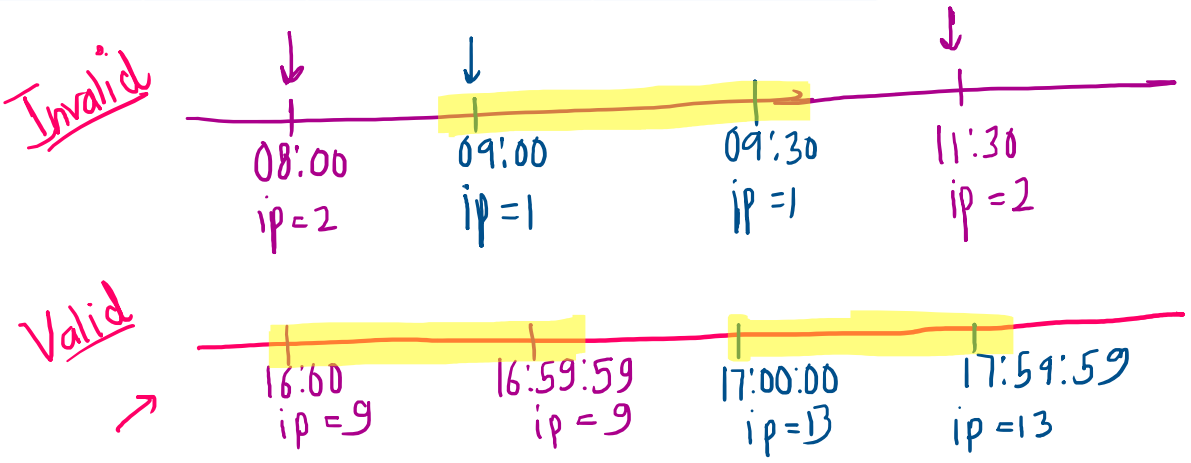
Shaik Vazeer
are non equi joins optimized? i can see a lot of rows getting returned based upon the condition

Session - 9, Q8

"Prakash"

account_id	ip_address	login	logout
1	1	2021-02-01 09:00:00	2021-02-01 09:30:00
1	2	2021-02-01 08:00:00	2021-02-01 11:30:00
2	6	2021-02-01 20:30:00	2021-02-01 22:00:00
2	7	2021-02-02 20:30:00	2021-02-02 22:00:00
3	9	2021-02-01 16:00:00	2021-02-01 16:59:59
3	13	2021-02-01 17:00:00	2021-02-01 17:59:59
4	10	2021-02-01 16:00:00	2021-02-01 17:00:00
4	11	2021-02-01 17:00:00	2021-02-01 17:59:59

← Nancy



Break till 22:23

Sanket
session "Joins" question 8

```
SELECT s.buyer_id from Sales s
left join Product p on s.product_id = p.product_id
group by s.buyer_id
having (sum(p.product_name = "S8")>0 and sum(p.product_name = "iPhone")=0)
order by s.buyer_id
```

```
SELECT s.buyer_id from Sales s
left join Product p on s.product_id = p.product_id
group by s.buyer_id
having p.product_name <> "iPhone"
order by s.buyer_id
```

buyer1 = (1, 2, 3, 4, 5) ↙ S8 =

buyer2 = (1, 2) ↖ iphone =

buyer1 not in (1, 2) =

SANKET:

```
SELECT candidate_id,
from candidates
where skill in ("Python","Tableau","MySQL")
group by candidate_id
having count(distinct(skill))=3
order by candidate_id
```

↑ Aggregation

⇓ Where

C_id	skill
1	MySQL
1	MySQL
2	Python
3	

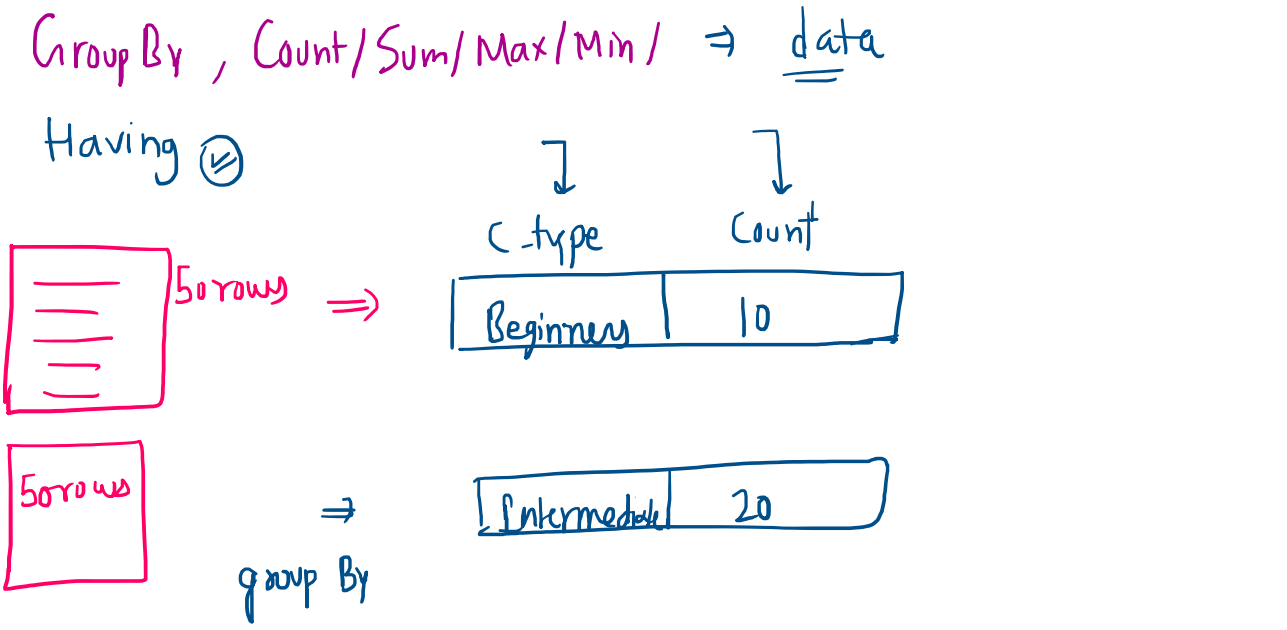
Select Candidate_id from

```
order by candidate_id
```

Aggregation

Select Candidate_id from Having Count(distinct(skill)) = 3

```
SELECT candidate_id
 from candidates
where skill in ("Python","Tableau","MySQL")
group by candidate_id
having (skill="Python" and skill = "Tableau" and skill = "MySQL")
order by candidate_id
```



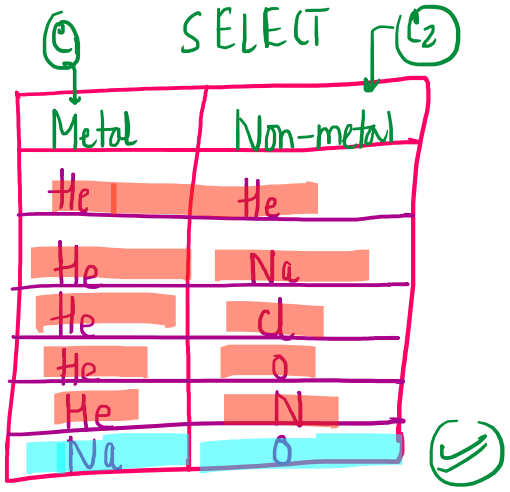
Nancy Aspin
Joins continued 7th question explain sir.

symbol	type	electrons
He	Noble	0
Na	Metal	1
Ca	Metal	2
La	Metal	3
Cl	Nonmetal	1
O	Nonmetal	2
N	Nonmetal	3

e1

symbol	type	electrons
He	Noble	0
Na	Metal	1
Ca	Metal	2
La	Metal	3
Cl	Nonmetal	1
O	Nonmetal	2
N	Nonmetal	3

e2

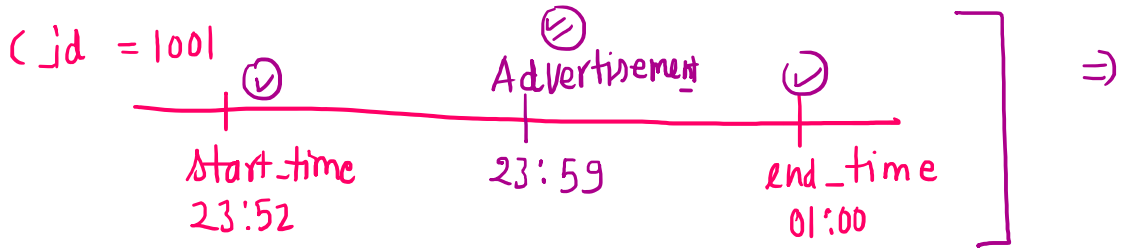


Where e1.type = "Metal"
AND e2.type = "Nonmetal"

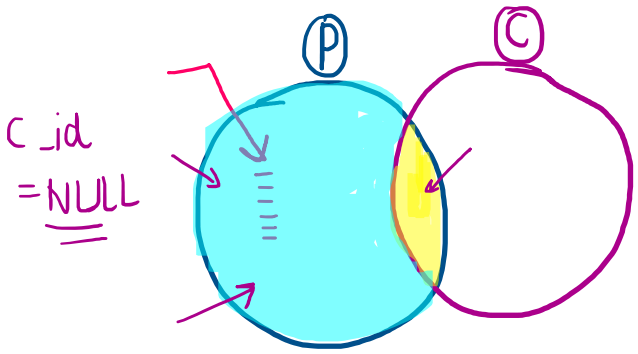
Sharique Rahi
Session 8 - addition problem Q2. Ad-Free Sessions , From Join first class. Can you please explain this .

Sharique Kazi

Session 8 - addition problem Q2. Ad-Free Sessions , From Join first class. Can you please explain this .




```
select session_id
from Playback p
left join Ads a
on a.customer_id = p.customer_id and
a.timestamp between p.start_time and p.end_time
where a.customer_id is null
order by session_id;
```



Session 10 - Applied SQL 2

08 September 2024 19:06

SUB-QUERIES

Quiz 1: Average Salary Comparison

Tables:
employees

employee_id	employee_name	salary	department_id
1	Alice	60000	101
2	Bob	50000	101
3	Charlie	70000	102
4	David	80000	102
5	Eve	90000	103

Departments

department_id	department_name
101	HR
102	Finance
103	IT

Question:
Find employees who earn more than the average salary of their department.

```
SELECT employee_name
FROM employees e
WHERE salary > (SELECT AVG(salary)
        FROM employees
        WHERE department_id = e.department_id);
```

Subquery referring Column of Outer query. ↖ (Correlated Subquery)

Quiz 2: Top Order Count

Tables:
customers

customer_id	customer_name
1	Alice
2	Bob
3	Charlie

orders

order_id	customer_id	order_total
1	1	200
2	2	300
3	1	400
4	3	150
5	2	500

(for a customer) . (for a table)

Sum (order_total) > Avg (order_total)

Question:
Find customers whose total order amount is greater than the average order total for all customers.

```
SELECT customer_name          ] = Q2
FROM customers c
WHERE (SELECT SUM(order_total)
    FROM orders o
    WHERE o.customer_id = c.customer_id
) > (SELECT AVG(order_total) FROM orders);   = Q1
```

Correlate Subquery = Q3

Q1 = Execute Once ⇒ 100

Q2 = Q3 (Execute) ⇒ 500

Q2 (Execute)

Reference from farmers_maret database:

```
select c.customer_first_name
```

```
from `farmers_market.customer` c
where (
    select sum(p.quantity * p.cost_to_customer_per_qty)
    from `farmers_market.customer_purchases` p
    where p.customer_id = c.customer_id
) > (select avg(quantity * cost_to_customer_per_qty) from `farmers_market.customer_purchases`);
```

Quiz 3: Total Orders per Product

Tables:

products

product_id	product_name
1	Laptop
2	Phone
3	Tablet

order_details

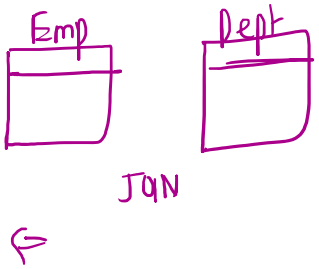
order_id	product_id	quantity
1	1	2
2	2	1
3	1	3

Question:
For each product, find the total number of orders placed.

```
SELECT p.product_name,
    (SELECT COUNT(*)
     FROM order_details od
     WHERE od.product_id = p.product_id) AS total_orders
FROM products p;
```

JOINS

Quiz 4: Employee and Department Names



Tables:

employees

employee_id	employee_name	department_id
1	Alice	101
2	Bob	102
3	Charlie	NULL

departments

department_id	department_name
101	HR
102	Finance
103	IT

Question:
Retrieve all employee names along with their department names. Show employees even if they don't belong to a department.

```
SELECT employee_name, department_name
FROM employees e
LEFT JOIN departments d
ON e.department_id = d.department_id;
```

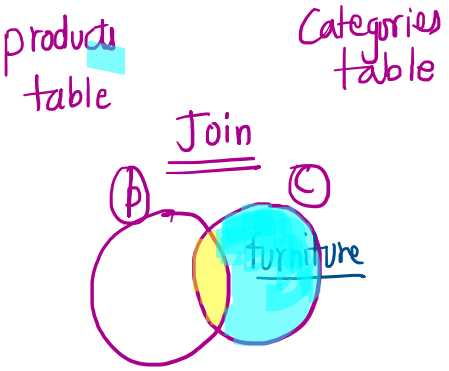
Quiz 5: Products and Categories

Tables:
products

product_id	product_name	category_id
1	Laptop	101
2	Phone	102

categories

category_id	category_name
101	Electronics
102	Gadgets
103	Furniture



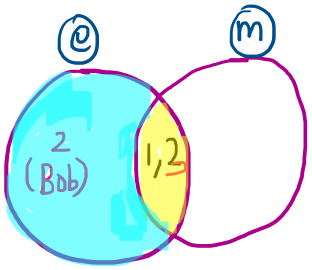
Question:
Retrieve all product names and their categories. Include categories that don't have any products.

```
SELECT product_name, category_name
FROM products p
RIGHT JOIN categories c
ON p.category_id = c.category_id;
```

Quiz 6: Employees and Managers

Tables:
employees

employee_id	employee_name	manager_id
1	Alice	2
2	Bob	NULL
3	Charlie	2



Question:
Retrieve all employees and their managers. Show employees even if they don't have a manager.

```
SELECT e.employee_name, m.employee_name AS manager_name
FROM employees e
LEFT JOIN employees m
ON e.manager_id = m.employee_id;
```

Quiz 7: Students and Courses

Tables:
students

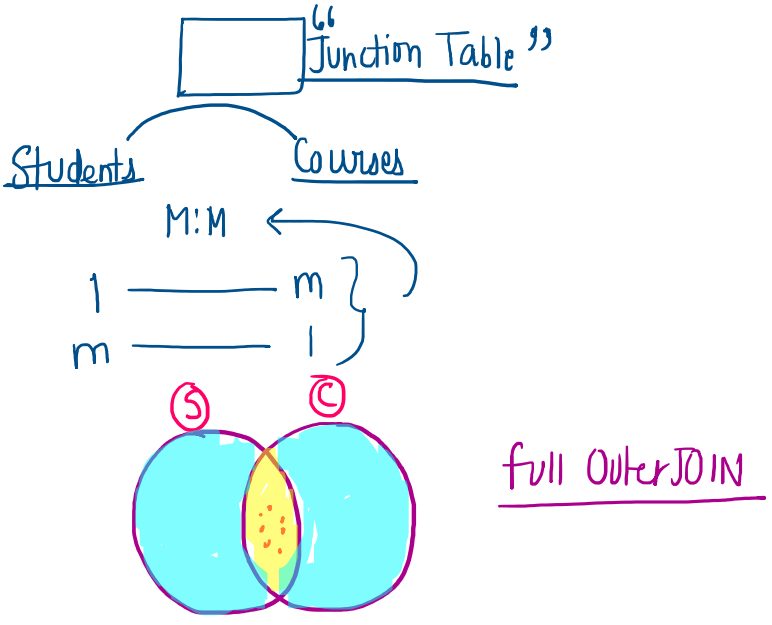
student_id	student_name
1	Alice
2	Bob
3	Charlie

enrollments

student_id	course_id
1	101
2	102

courses

course_id	course_name
101	Math
102	Physics
103	Chemistry



Question:

List all students and the courses they are enrolled in. Show all courses, even if no students are enrolled in them.

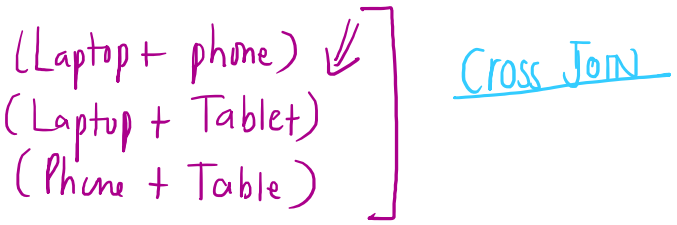
```
SELECT s.student_name, c.course_name
FROM students s
FULL OUTER JOIN enrollments e ON s.student_id = e.student_id
FULL OUTER JOIN courses c ON e.course_id = c.course_id;
```

Quiz 8: Cross-selling Products

Tables:

products

product_id	product_name
1	Laptop
2	Phone
3	Tablet



Question:
You want to create a list of all possible pairs of products to suggest cross-selling combinations.

Answer:
You would use a CROSS JOIN because you need to create all possible pairs between rows of the same table.

```
SELECT p1.product_name AS product_1,
       p2.product_name AS product_2
FROM products p1
CROSS JOIN products p2;
```

Quiz 9: Total Sales by Category with Filter

Tables:

sales

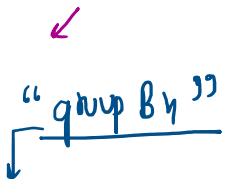
sale_id	product_id	quantity	sale_amount
1	1	10	500
2	2	5	300
3	3	8	400
4	1	2	100

products

product_id	product_name	category_id
1	Laptop	101
2	Phone	102
3	Tablet	101

categories

category_id	category_name
101	Electronics
102	Gadgets



Question:
Find the total sales amount by category, but only include categories where the total sales amount exceeds 400.

```
SELECT c.category_name,
       SUM(s.sale_amount) AS total_sales
FROM sales s
JOIN products p ON s.product_id = p.product_id
JOIN categories c ON p.category_id = c.category_id
GROUP BY c.category_name
HAVING SUM(s.sale_amount) > 400;
```

Quiz 10: Count of Employees in Each Department

Tables:
employees

employee_id	employee_name	department_id	salary
1	Alice	101	50000
2	Bob	101	60000
3	Charlie	102	55000
4	David	103	70000

departments

department_id	department_name
101	HR
102	Finance
103	IT

Question:
Count the number of employees in each department, but only include departments with more than one employee.

```
SELECT d.department_name,
       COUNT(e.employee_id) AS employee_count
FROM employees e
JOIN departments d ON e.department_id = d.department_id
GROUP BY d.department_name
HAVING COUNT(e.employee_id) > 1;
```

Quiz 11: Product Sales Above Average

Tables:
sales

sale_id	product_id	sale_amount
1	1	500
2	2	300
3	3	400
4	1	600

products

product_id	product_name
1	Laptop
2	Phone
3	Tablet

total sale ? > avg-sale accross all product =

JOIN = ?

Sub-query = ?

Question:
Find the products whose total sales amount exceeds the average sales amount across all products.

```
SELECT p.product_name,
       SUM(s.sale_amount) AS total_sales
FROM sales s
JOIN products p ON s.product_id = p.product_id
GROUP BY p.product_name
HAVING SUM(s.sale_amount) > (SELECT AVG(sale_amount) FROM sales);
```

"Subquery with Having"

Quiz 12: Top-N Query - Highest Total Salary Departments

Tables:

employees

employee_id	employee_name	department_id	salary
1	Alice	101	50000

2	Bob	102	60000
3	Charlie	103	55000
4	David	101	70000

departments

department_id	department_name
101	HR
102	Finance
103	IT

Question:
Find the top 2 departments with the highest total salary.

```
SELECT d.department_name,
       SUM(e.salary) AS total_salary
FROM employees e
JOIN departments d ON e.department_id = d.department_id
GROUP BY d.department_name
ORDER BY total_salary DESC
LIMIT 2;
```

Quiz 13: Percentage of Total Sales by Product

Tables:
sales

sale_id	product_id	sale_amount
1	1	500
2	2	300
3	3	400
4	1	600

Group By =

Laptop = $\frac{1000}{x} \times 100$ %

$60/100 = \frac{60}{100} \times 100$

products

product_id	product_name
1	Laptop
2	Phone
3	Tablet

Question:
For each product, calculate its percentage of the total sales amount across all products.

```
SELECT p.product_name,
       (SUM(s.sale_amount) * 100.0 / (SELECT SUM(sale_amount) FROM sales))
AS sales_percentage
FROM sales s
JOIN products p ON s.product_id = p.product_id
GROUP BY p.product_name;
```

Quiz 14: Highest Sale per Product

Tables:
sales

sale_id	product_id	sale_amount
1	1	500
2	1	600
3	2	300
4	2	400

Group By , max(—)

products

4	2	400

products

product_id	product_name
1	Laptop
2	Phone

Question:
For each product, find the highest sale amount recorded.

```
SELECT p.product_name,
       MAX(s.sale_amount) AS highest_sale
FROM sales s
JOIN products p ON s.product_id = p.product_id
GROUP BY p.product_name;
```

Session 11 - Windowing Functions

12 September 2024 17:25

GlobalTech Solutions

GlobalTech Solutions, a leading technology firm renowned for its innovative solutions and cutting-edge technology, boasts a diverse and talented workforce.

Challenges:

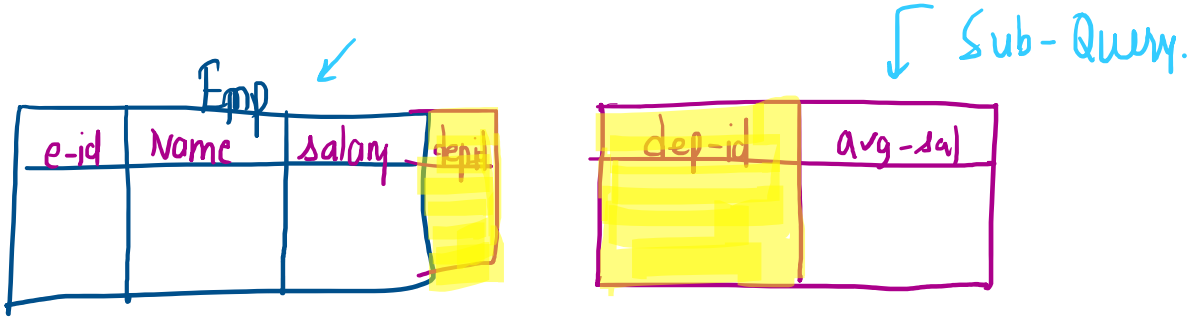
- As GlobalTech Solutions continues to expand and diversify its workforce, the variation in compensation packages presents a challenge in maintaining consistency and fairness across the organization.
- Addressing salary disparities and managerial inconsistencies is crucial to avoid potential legal issues and reputational damage, underscoring the importance of comprehensive data analysis.

Q. Display detailed information on employees earning above the company-wide average salary.

```
select  *
from `hr.employees`
where salary > (
        select avg(salary) as salary
        from `hr.employees`
);
```

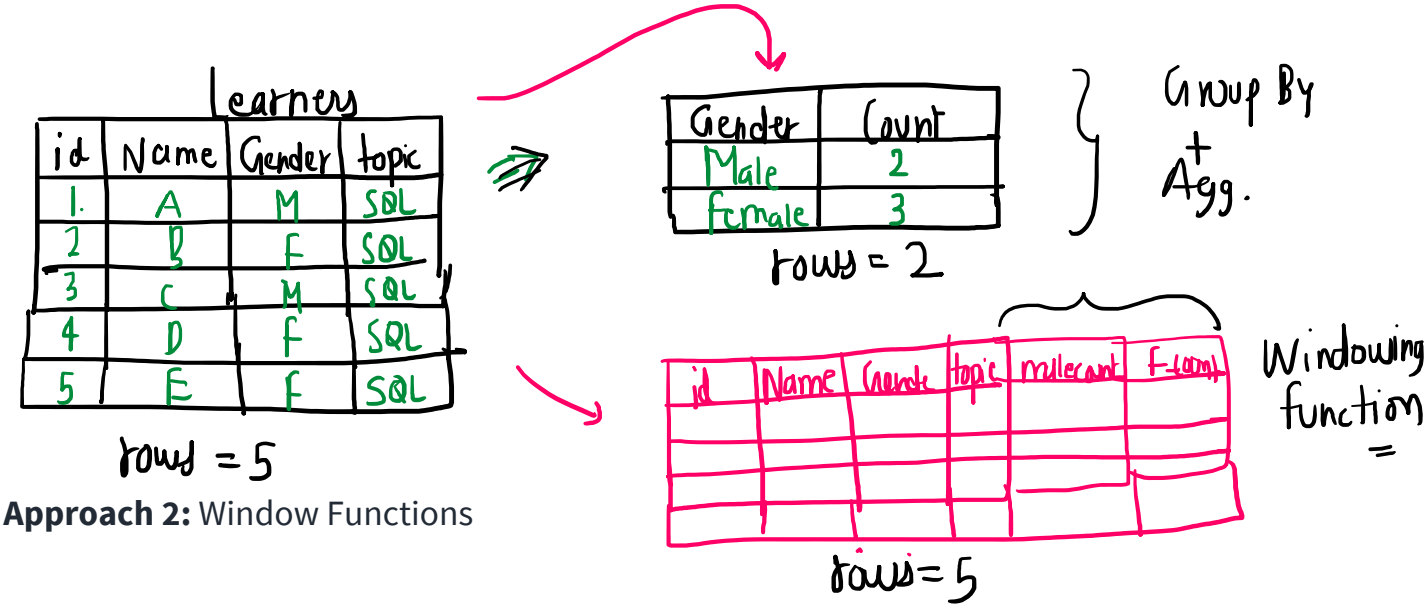
Q: Show details of employees whose salaries exceed the average salary of their respective departments.

Approach 1: Using group By clause.

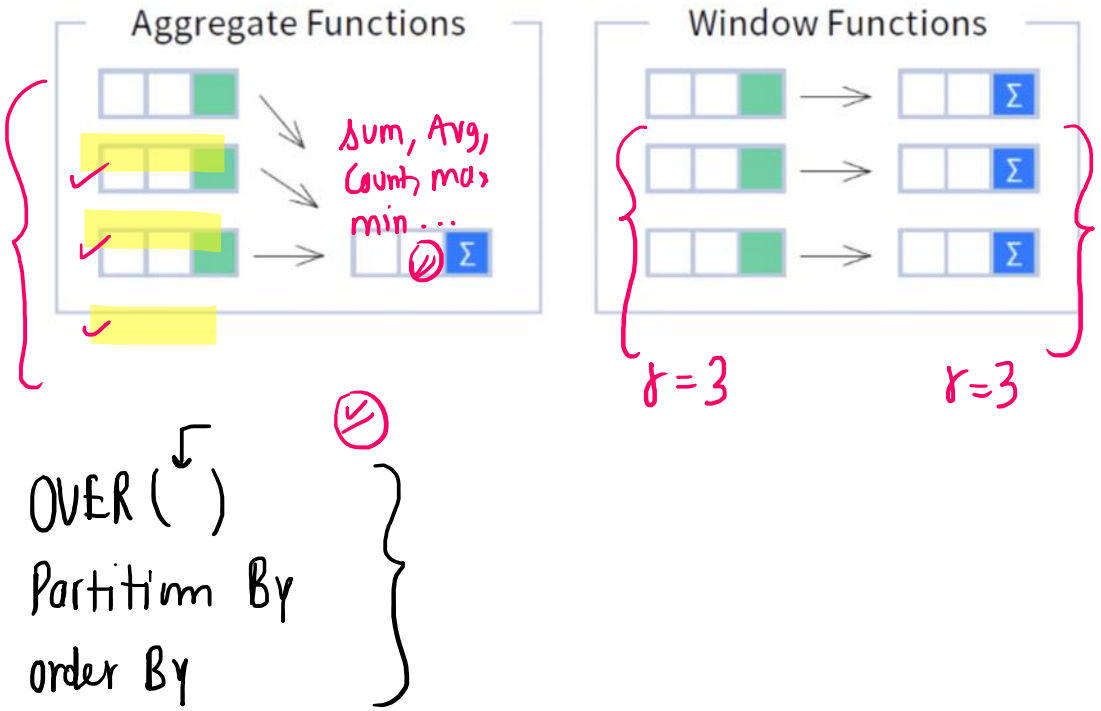



```
select  e.employee_id,
        e.first_name,
        e.last_name,
        e.salary,
```

```
        dept_avg.avg_salary
from `hr.employees` e
JOIN (
        select e1.department_id, round(avg(e1.salary),2) avg_salary
        from `hr.employees` e1
        group by e1.department_id
) dept_avg

ON e.department_id = dept_avg.department_id
where e.salary > dept_avg.avg_salary;
```



Approach 2: Window Functions



Q: Display the average salary of all employees in a new column in the employees table.

```
SELECT
 employee_id,
 department_id,
 salary,
 AVG(salary) OVER() AS avg_salary
FROM hr.employee;
```

Q: How can we display the average salary of employees in each department by

adding a new column next to the employees in their respective departments?

```
SELECT
 employee_id,
 department_id,
 salary,
 AVG(salary) OVER(PARTITION BY department_id) AS avg_salary
FROM hr.employee;
```

Comment

Q: Display detailed information on employees earning above the company-wide average salary.

```
SELECT *
 FROM
 (SELECT
   employee_id,
   department_id,
   salary,
   AVG(salary) OVER() AS avg_salary
 FROM hr.employee) as tbl
WHERE salary > avg_salary;
```

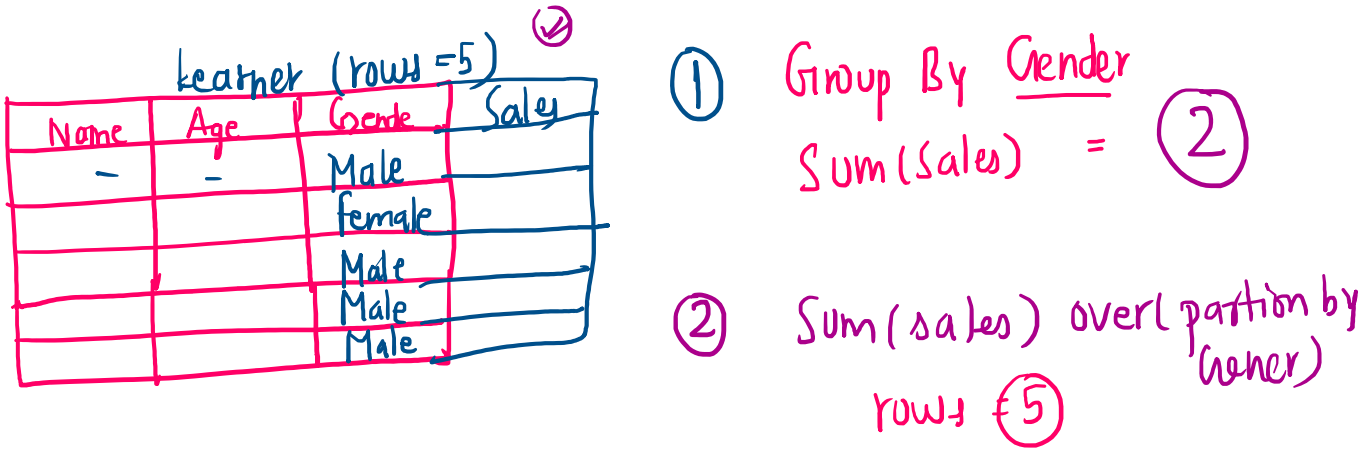
Q: Show details of employees whose salaries exceed the average salary of their respective departments.

```
SELECT *
 FROM
 (SELECT
   employee_id,
   department_id,
   salary,
   ROUND(AVG(salary) OVER(PARTITION BY department_id)) AS avg_salary
  FROM hr.employee) tbl
WHERE salary > avg_salary;
```

QUIZ: Assume a table of 5 rows with 2 unique categories in the category column and sales as another column.

How many rows do we get if -

1. **Group BY on the category and SUM(sales)**
2. **SUM(sales) using Window functions**

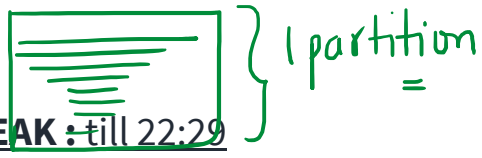


ROW_NUMBER:

Q: Identify the top 5 highest-earning employees in the company.

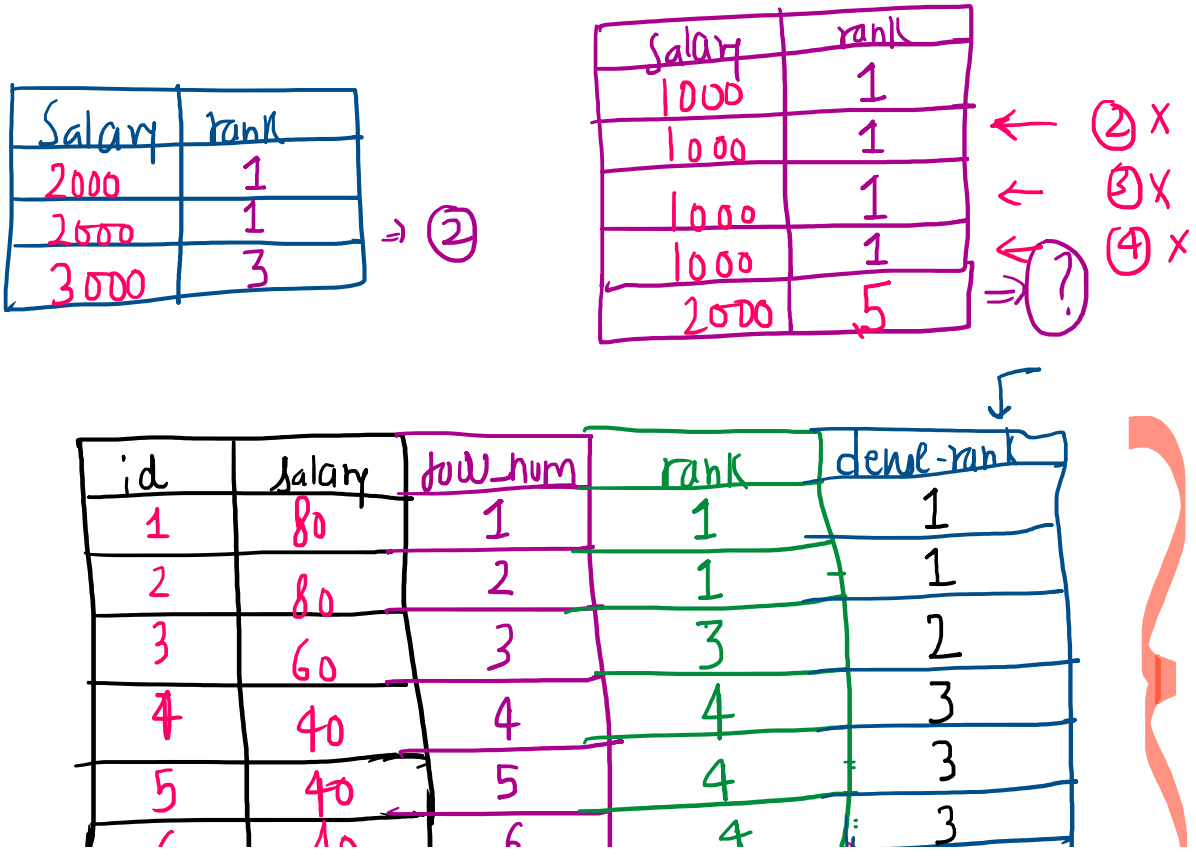
```sql
SELECT *
from (
    select e.employee_id,
        e.first_name,
        e.last_name,
        e.department_id,
        salary,
        row_number() OVER (order by e.salary desc) AS row_num
    from `hr.employees` e
) t1
where t1.row_num <=5;
```

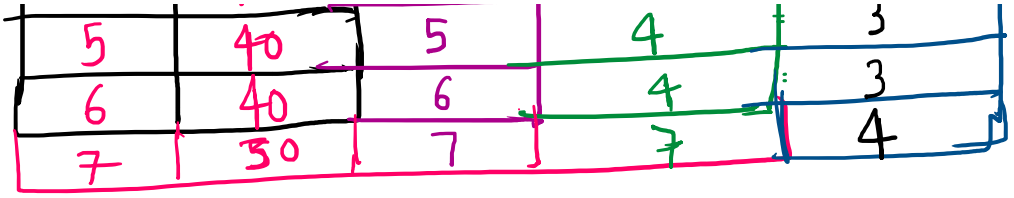
QUIZ: When we havn't mentioned any **PARTITION BY** clause, then the entire table is considered as one partition?



BREAK : ~~till 22:29~~

RANK and DENSE_RANK:





DISTINCT (Not Efficient)

Select * from
(Select *, row_number() over (partition by emp_id) as rnk) t1
where t1.rnk = 1;

Q: Fetch details of employees with the highest salaries in each department.

```
SELECT *
from (
   select e.employee_id,
        e.first_name,
        e.last_name,
        e.department_id,
        salary,
        row_number() OVER (order by e.salary desc) AS row_num,
        rank() OVER (order by e.salary desc) As rnk,
        dense_rank() OVER (order by e.salary desc) As dense_rnk,
   from `hr.employees` e
) t1
where t1.row_num <=5
order by rnk, dense_rnk, row_num;
```

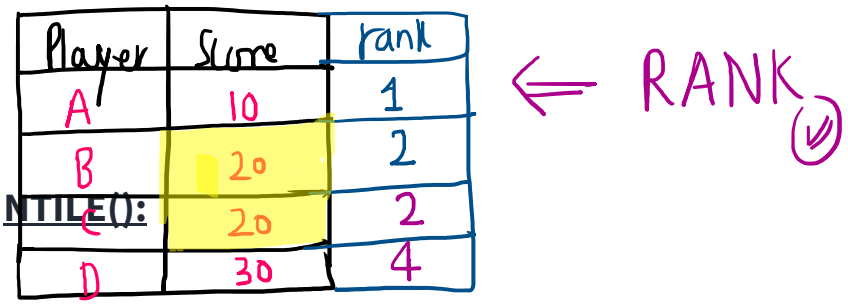
Homework for the learners -

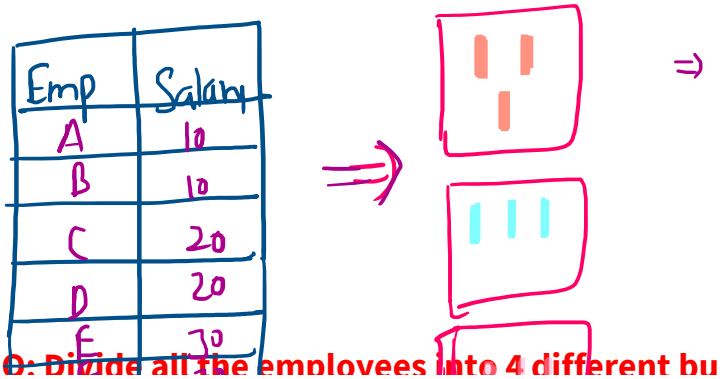
Q: From each department, display the details of the employee having the 3rd highest salary. Display multiple people if they have the same salaries.

```
SELECT *
from (
    select e.employee_id,
           e.first_name,
           e.last_name,
           e.department_id,
           salary,
           dense_rank() OVER (partition by e.department_id order by e.salary desc)
As dense_rnk
    from `hr.employees` e
) t1
where t1.dense_rnk =3;
```

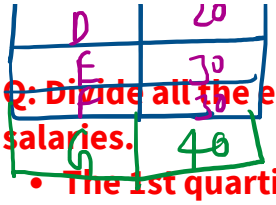
QUIZ: Joe wants to rank players by their scores. If there is a tie, he wants the next rank to jump by the number of ties.

Which window function should he use here?



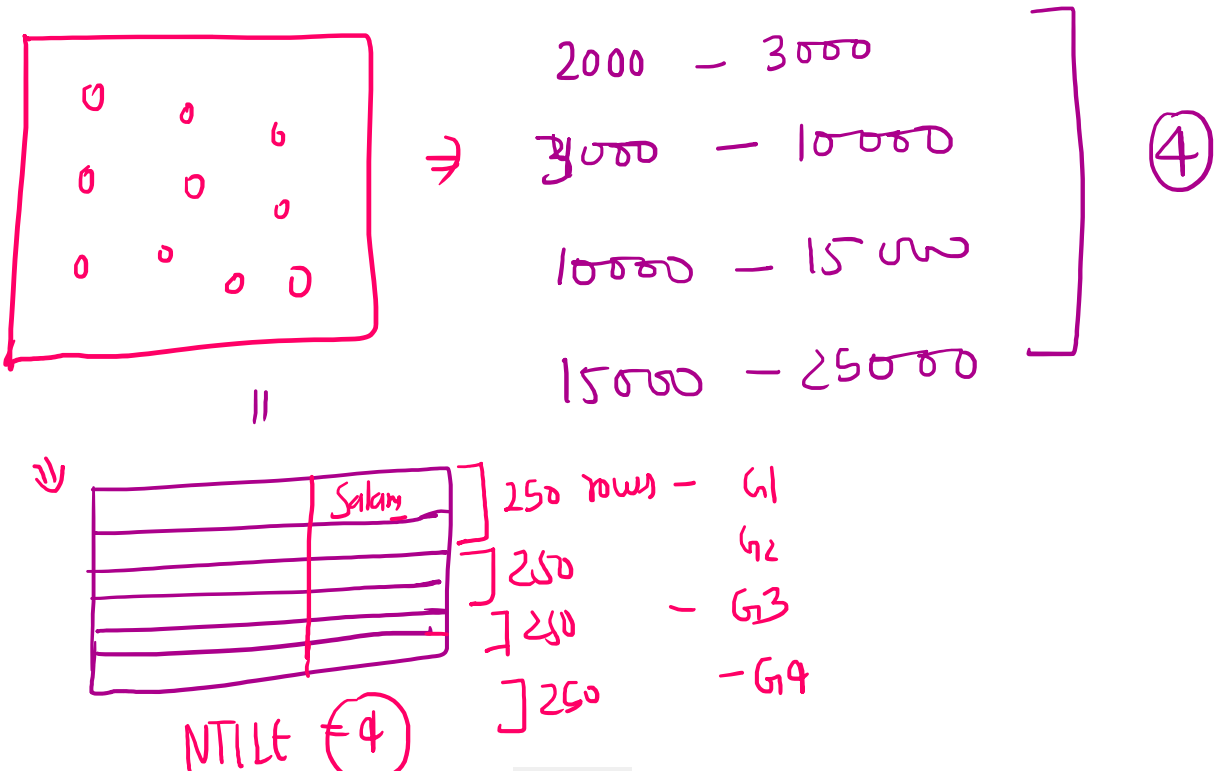


Q: Divide all the employees into 4 different buckets (i.e. quartiles) based on their



Q: Divide all the employees into 4 different buckets (i.e. quartiles) based on their salaries.

- **The 1st quartile should contain the employees with the lowest salaries.**
- **The 4th quartile should contain the employees with the highest salaries.**



QUIZ: Is it true or false that the **NTILE()** function creates groups solely based on the number of rows and the specified NTILE value?

rows = 1000
Ntile = 4
} 250 / Group

rows = 1001
Ntile = 4
} 250, 250, 250, 251

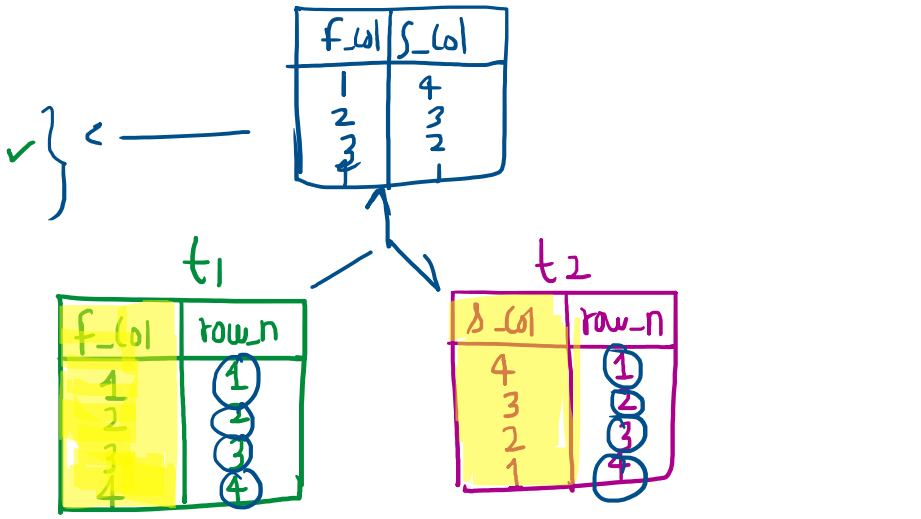
Problem Statement:
Write a query to independently:

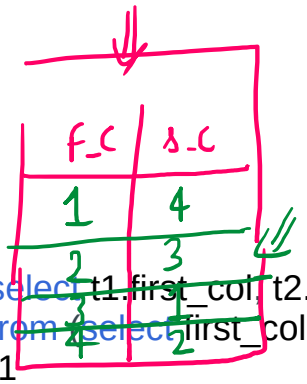
- order first_col in ascending order.

- order second_col in descending order.

Sample Input:
Table: data

first_col	second_col
4	2
2	3
3	1
1	4



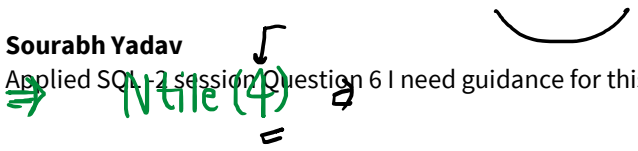



```
select t1.first_col, t2.second_col
from (select first_col, row_number() over (order by first_col) row_num from data)
t1
JOIN (select second_col, row_number() over (order by second_col desc)
row_num  from data) t2
ON t1.row_num = t2.row_num;
```

Anjali
1)If there is huge variation in salary (Values are not close) how can we utilize ntile then from analysis perspective?

2)How can we decide the number of groups for ntile without hit and try?

Sourabh Yadav
Applied SQL - 2 session Question 6 I need guidance for this question



Session 12 - Windowing Functions 2

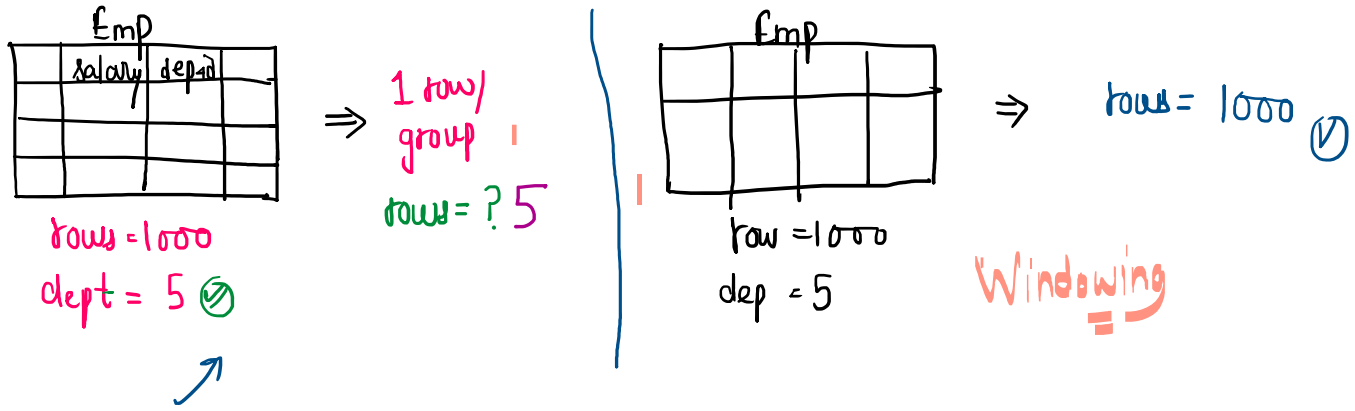
14 September 2024 18:22

Agenda

1. Lead(), Lag()
2. Cumulative Sum
3. Window Frames
4. Break & Doubt Resolution
5. Moving Average
6. ROWS vs. RANGE
7. First_value(), Last_value()
8. Impact of the analysis
9. Practice Question

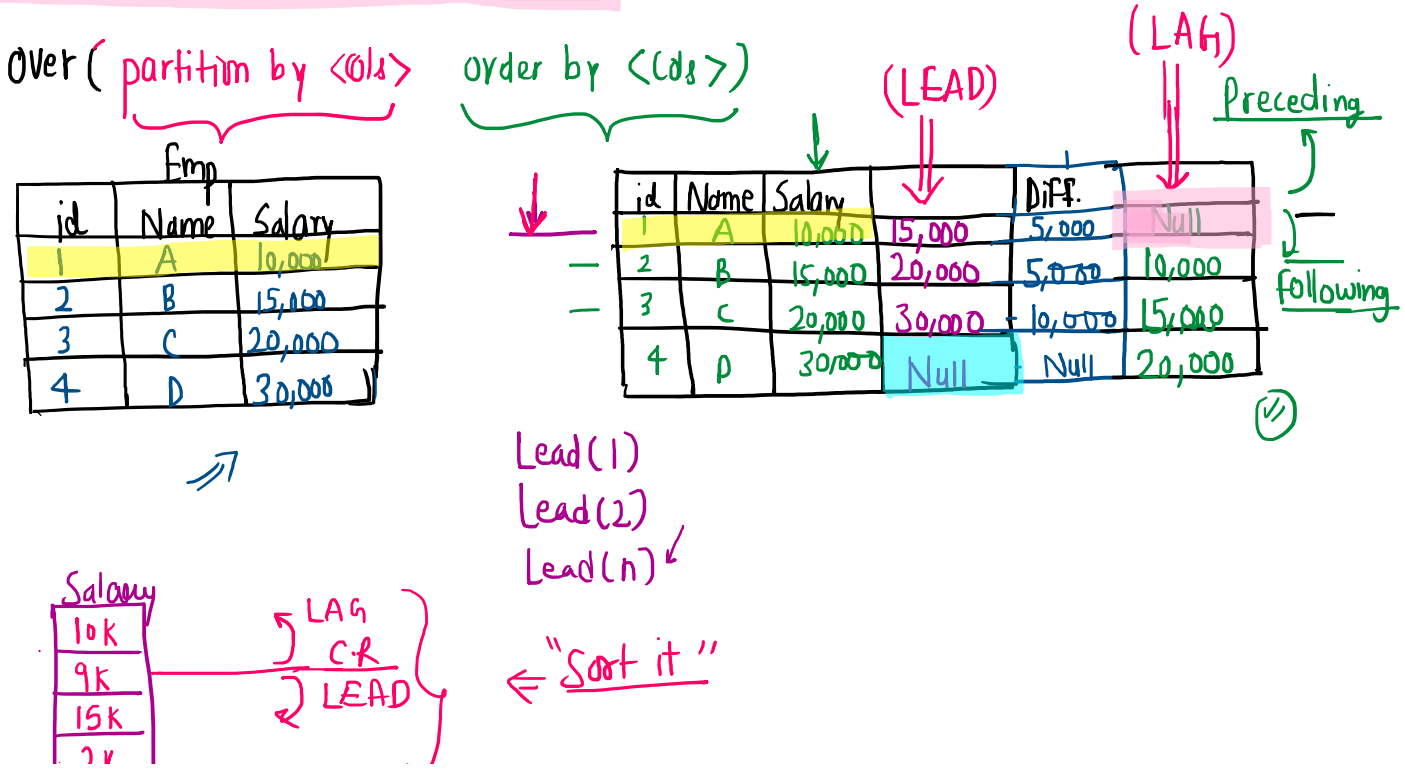
Dataset: LINK

We need to find out for each employee, who earns more than them in the company, and by how much?



LEAD() and LAG()

Q: For each employee, provide their details and current salary. Additionally, identify the salary of the next highest-earning employee within the company. Present the difference in their salaries in a new column labelled sal_diff.

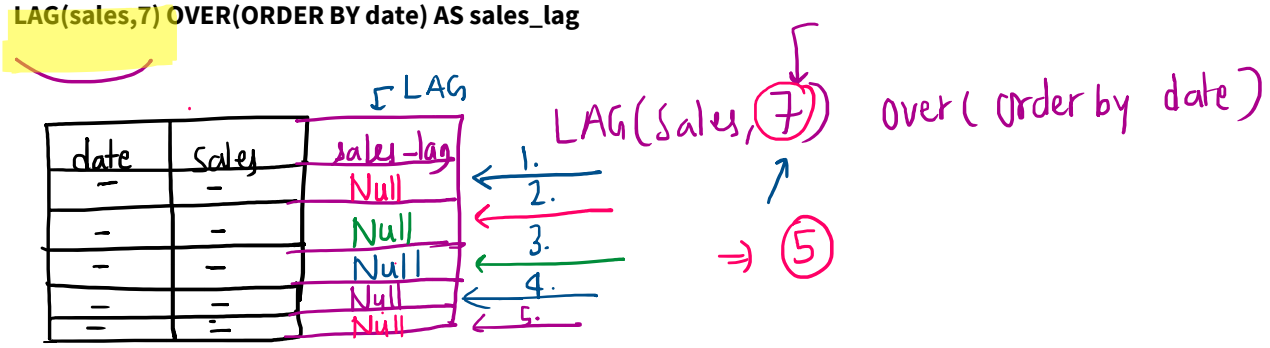



```
select  e.employee_id,
        e.first_name,
        e.last_name,
        e.department_id,
        e.salary,
        LEAD(salary) OVER (order by salary) AS next_salary,
        LEAD(salary) OVER (order by salary) - salary AS salary_diff
from `hr.employees` e
order by e.salary, next_salary;
```

QUIZ: LAG() and LEAD() can work even without the ORDER BY clause within the window function statement.

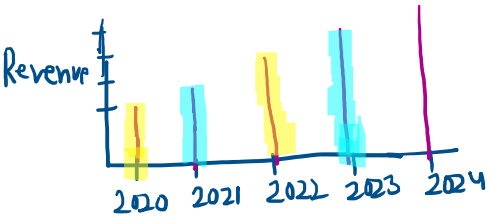
QUIZ: Assume we have 5 rows of date and sales. How many NULL values will be in the sales_lag column based on this query?

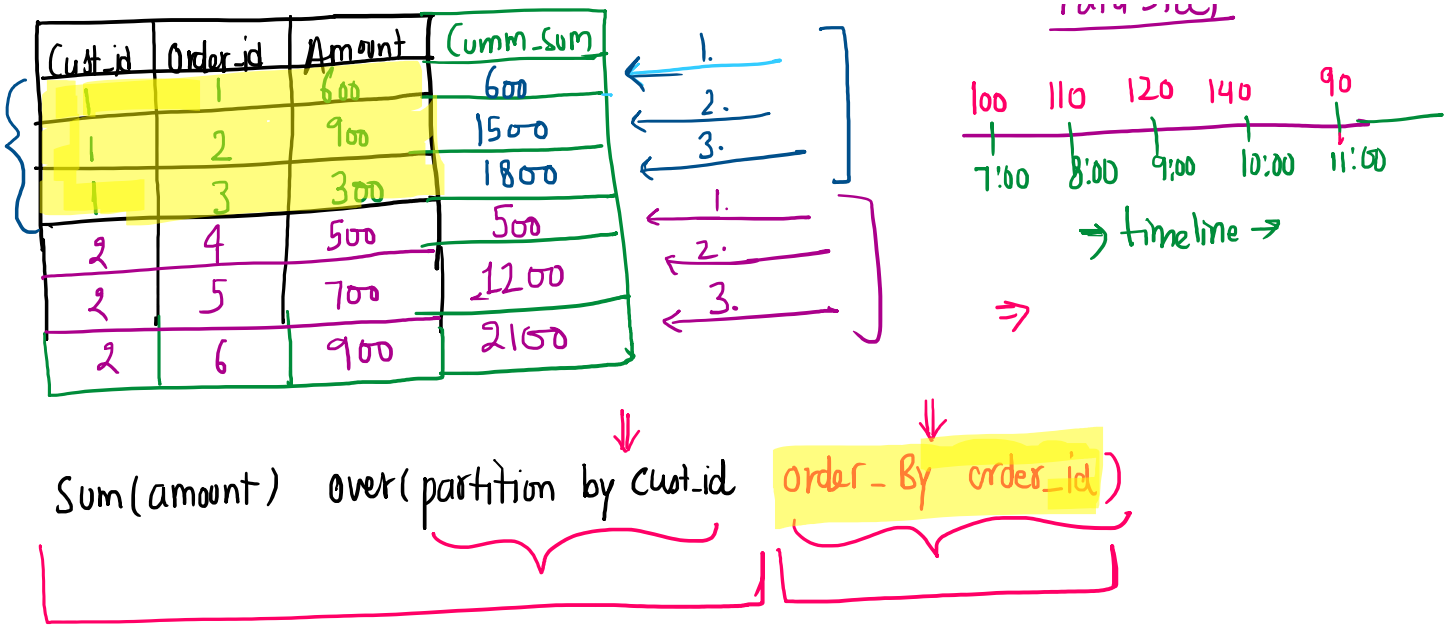
LAG(sales,7) OVER(ORDER BY date) AS sales_lag



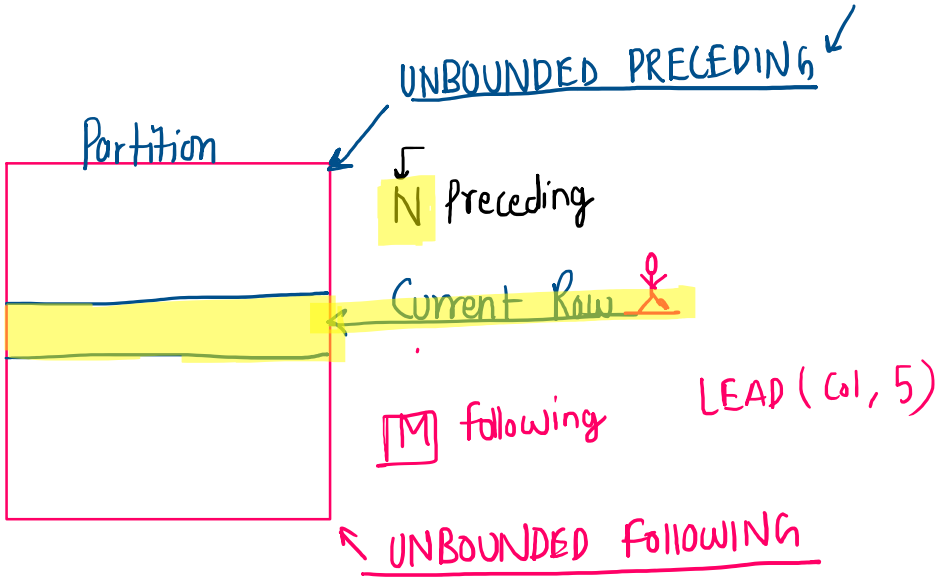

```
select  e.employee_id,
        e.first_name,
        e.last_name,
        e.department_id,
        e.salary,
        LAG(salary) OVER (order by salary) AS previous_salary,
        LEAD(salary) OVER (order by salary) AS next_salary
from `hr.employees` e
order by e.salary, previous_salary, next_salary;
```

Cumulative / Running Sum



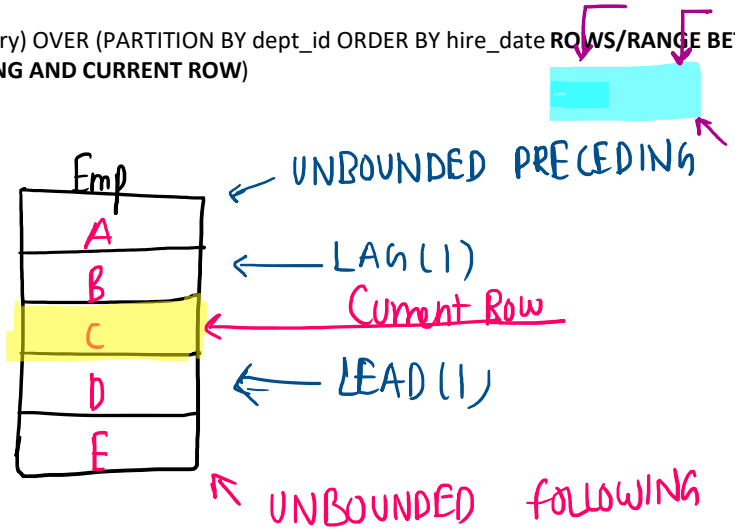


SUM(salary) OVER (PARTITION BY dept_id ORDER BY hire_date)



SUM(salary) OVER (PARTITION BY dept_id ORDER BY hire_date)

SUM(salary) OVER (PARTITION BY dept_id ORDER BY hire_date **ROWS/RANGE BETWEEN UNBOUNDED PRECEDING AND CURRENT ROW**)

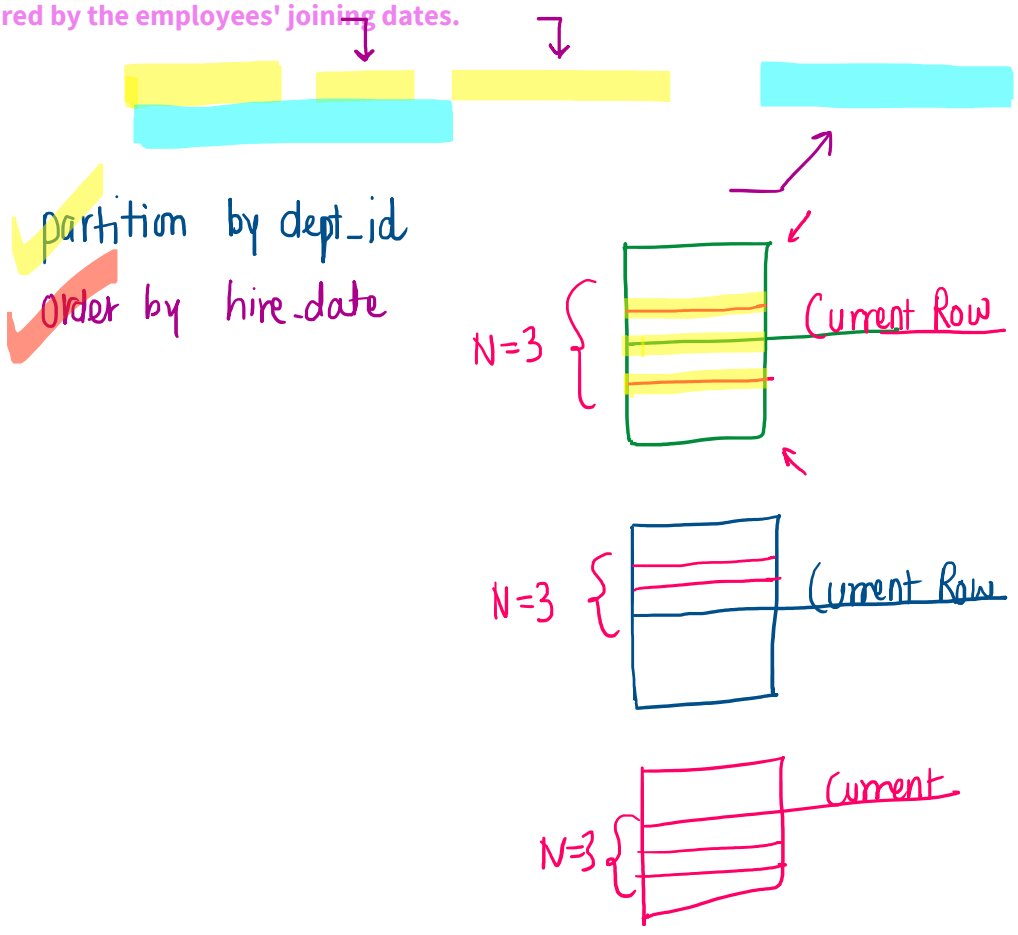



```
select  e.employee_id,
```

```
        e.first_name,
        e.last_name,
        e.department_id,
        e.hire_date,
        e.salary,
        sum(salary) over(partition by e.department_id order by e.hire_date rows between
unbounded preceding and current row) as sum_salary
from `hr.employees` e;

select  e.employee_id,
        e.first_name,
        e.last_name,
        e.department_id,
        e.hire_date,
        e.salary,
        sum(salary) over(partition by e.department_id order by e.hire_date) as sum_salary
from `hr.employees` e;
```

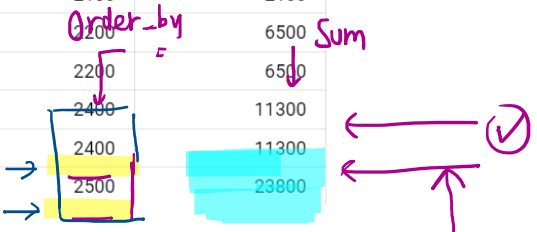
Break till 22:38

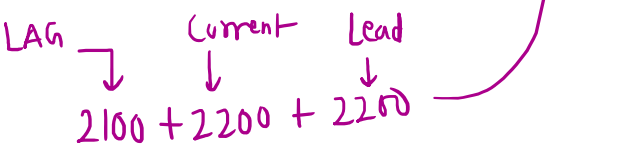
Q: Display the moving sum of salaries in each department, with a window size of N=3, ordered by the employees' joining dates.




```
select  e.employee_id,
        e.department_id,
        e.salary,
        e.hire_date,
        sum(e.salary) over(partition by e.department_id order by e.hire_date rows between
1 preceding and 1 following) as moving_sum,
        sum(e.salary) over(partition by e.department_id order by e.hire_date rows between
2 preceding and current row) as moving_sum2
from `hr.employees` e
order by e.department_id, e.hire_date, moving_sum;
```

132	50	1999-04-10	2100	2100
128	50	2000-03-08	2200	6500
136	50	2000-02-06	2200	6500
127	50	1999-01-14	2400	11300
135	50	1999-12-12	2400	11300
131	50	1997-02-16	2500	23800

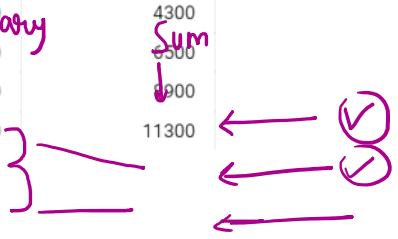




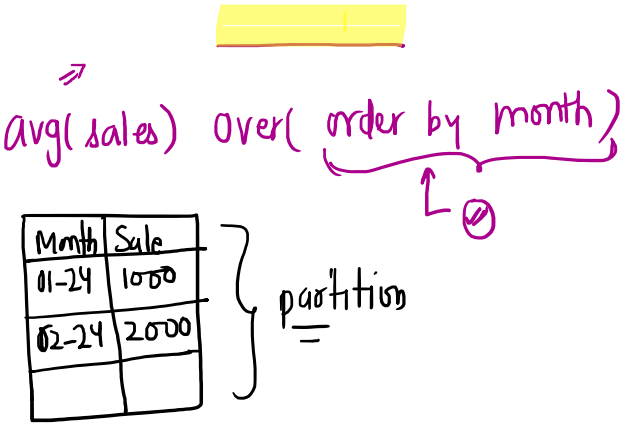
OPTION 2:

```
select  e.employee_id,
        e.department_id,
        e.hire_date,
        e.salary,
        sum(salary) over(partition by e.department_id order by e.salary rows between
unbounded preceding and current row) as sum_salary
from `hr.employees` e
order by e.department_id,e.salary,sum_salary;
```

132	50	1999-04-10	2100	2100
128	50	2000-03-08	2200	4300
136	50	2000-02-06	2200	6500
127	50	1999-01-14	2400	8900
135	50	1999-12-12	2400	11300



QUIZ: Assume we have monthly sales data. Which is the right query to get the running average of sales by month?



Which one is the right function to get the moving average of past 3 days, including today?

A
AVG(sales) OVER (ORDER BY date ROWS BETWEEN CURRENT ROW AND 3 FOLLOWING)

B

A
AVG(sales) OVER (ORDER BY date ROWS BETWEEN CURRENT ROW AND 3 FOLLOWING)

B
AVG(sales) OVER (ORDER BY date)

C
AVG(sales) OVER (ORDER BY date ROWS BETWEEN 3 PRECEDING AND CURRENT ROW)

D
AVG(sales) OVER (ORDER BY date ROWS BETWEEN 2 PRECEDING AND 1 FOLLOWING)

Q: Compare each employee's salary with the salaries of the first and last hired employees in their department.

```
select  e.employee_id,
        e.department_id,
        e.hire_date,
        e.salary,
        first_value(salary) over(partition by e.department_id order by e.salary rows between unbounded preceding and unbounded following) as first_val,
        last_value(salary) over(partition by e.department_id order by e.salary rows between unbounded preceding and unbounded following) as last_val
from `hr.employees` e
order by e.department_id, e.salary;
```

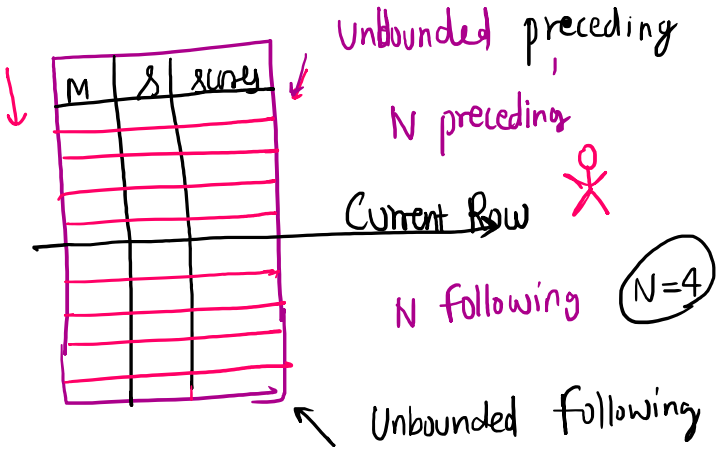
Assume a table contains marks, students and scores. When we want to see the students that have scored the 4th highest marks in each subject and the name of the student should be displayed in each row within each subject, then which one to use.

A
RANK()

B
DENSE_RANK()

C
NTH_VALUE()

D
FIRST_VALUE()



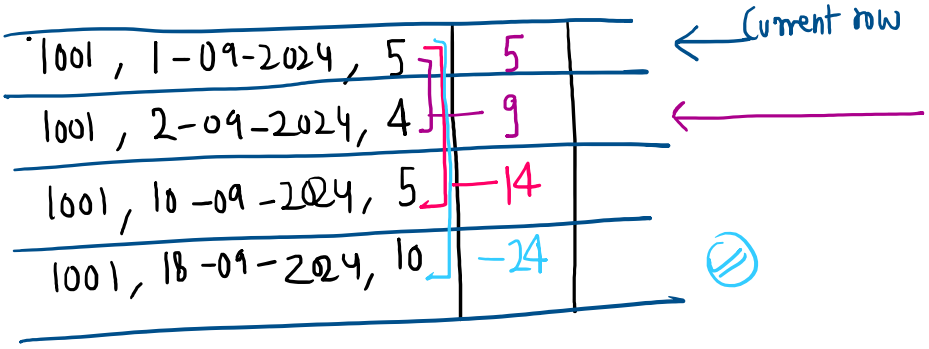
Question: Write a query to report for each player and date, how many games have been played so far by the player. That is, the total number of games played by the player until that date.

player_id	device_id	event_date	games_played
1	2	2016-03-01	5
1	2	2016-05-02	6
1	3	2017-06-25	1
3	1	2016-03-02	0
3	4	2018-07-03	5

3	1	2016-03-02	0
3	4	2018-07-03	5

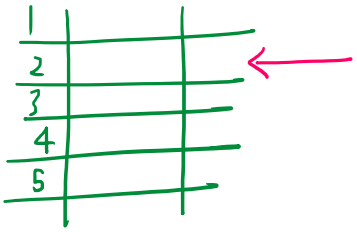
player_id	event_date	games_played_so_far
1	2016-03-01	5
1	2016-05-02	11
1	2017-06-25	12
3	2016-03-02	0
3	2018-07-03	5

1, 2016-03-01, 5
1, 2016-05-02, 11 ←
1, 2017-06-25, 12




```
Select  id,
        date,
        game_played
        sum(game_played) over( partition by id  order by date) as
        cum_sum
from    table;
```

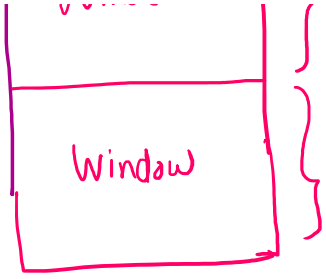
Order by <Column>



OVER(partition by dept_id order by salary)

↑ Commulative Sum

Window } Admin

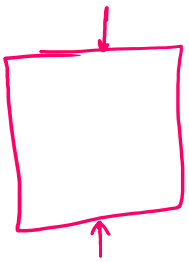


D.E

Unbounded Preceding =

Sum(Amount)

1001	Akifa Iram	1-Sep	100	⇒ 100	← Current row (100)
1001	"	2-Sep	200	⇒ 300	← Current row (100+300)
1001	"	3-Sep	300	⇒ 600	← Current row (100+300+600)
1001	"	18-09	1000	⇒ 1600	← Current row (100+300+600+1000)



Unbounded following

Session 13 - Date & Time Functions

19 September 2024 20:12

→ Windowing ⑧ +④
→ Windowing - 2 ⑧ +④
→ Date/Time functions ⑧ +④
Questions = 36
✓ Applied SQL - 3 ✓
⇒ 10 questions

Agenda

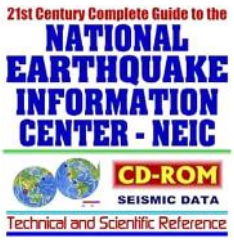
1. Problem Statement
2. PARSE_DATETIME()
3. EXTRACT()
4. FORMAT_DATE()
5. CURRENT_DATE()
6. DATE_ADD(), DATE_SUB()
7. DATE_DIFF()
8. Impact of the analysis
9. Practice Question

NOTE:
Date & Time functions may have a different syntax over platforms other than BigQuery (like MySQL, SQL Server, Oracle DB, etc.)

All solutions with MySQL 8.0 syntax - **Document**

MySQL Reference Link: https://dev.mysql.com/doc/refman/8.0/en/date-and-time-functions.html

Problem Statement:



The **National Earthquake Information Center (NEIC)** maintains a comprehensive dataset recording significant earthquakes worldwide, dating back to 1965. This dataset includes essential details such as the date, time, location, depth, magnitude, and source o f each earthquake with a reported magnitude of 5.5 or higher. The NEIC utilizes this dataset to support scientific research, inform national and international agencies, and provide critical information to the public.

Challenges: As more people move to urban areas, often located in seismically active regions, the potential impact of earthquakes on human life and infrastructure increases. This necessitates detailed analysis to mitigate risks.

Objective: Perform a thorough SQL analysis of the earthquake data to derive meaningful insights.

PARSE_DATETIME:

The PARSE_DATETIME() function combines the values from the Date and Time columns into a single string formatted as 'YYYY -MM-DD HH:MM:SS'

Here are some commonly used datetime format tokens:

- %Y: Year with century (e.g., 2024)
- %m: Month (01-12)
- %d: Day of the month (01-31)
- %I: Hour (00-12)
- %H: Hour (00-23)
- %M: Minute (00-59)
- %S: Second (00-59)
- %P: AM/PM indicator

Q: How can you combine the separate Date and Time columns into a single "Datetime" column?

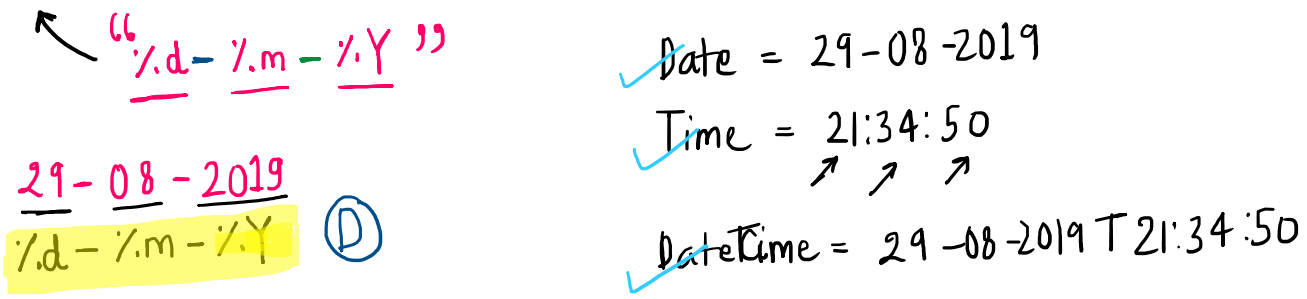
```
select  Date ,
        Time,
        concat(Date, " ", Time) as date_time_string,
        parse_datetime("%Y-%m-%d %H:%M:%S", concat(Date, " ", Time)) as date_time
from `neic.earthquakes`;
```

```
    parse_datetime("%Y-%m-%d %H:%M:%S", concat(Date, " ", Time)) as date_time
from `neic.earthquakes`;
```

Q: What is alternate of PARSE_DATETIME in MySQL ?

MySQL : str_to_date()
https://dev.mysql.com/doc/refman/8.0/en/date-and-time-functions.html#function_str-to-date

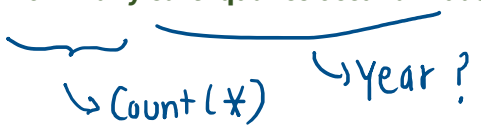
Q: "30-01-2020", what is the right parsing string to convert this into a date format?



EXTRACT()

Note: The syntax of the EXTRACT() function remains the same in MySQL.

Q: How many earthquakes occur annually, and what trends can be observed over the years?




```
-- select extract(datetime_part FROM timestamp_expr [AT TIME ZONE tz_spec])

select Datetime,
       extract(YEAR from Datetime) year,
       extract(MONTH from Datetime) month,
       extract(DAY from Datetime) day
from `neic.earthquakes`;


select extract(YEAR from Datetime) year,
       count(ID) earthquake_cnt
from `neic.earthquakes`
group by extract(YEAR from Datetime)
order by year desc;
```

Q: Which year experienced the highest number of significant earthquakes (magnitude greater than 7)?

↑ where or Having (filter grouped data)

↑ Where or Having (filter grouped data)

```
select extract(YEAR from Datetime) year,
      count(ID) earthquake_cnt
from `neic.earthquakes`
where Magnitude > 7
group by extract(YEAR from Datetime)
order by earthquake_cnt desc
limit 1;
```

Desc limit 1.

Year	Cnt
1968	100
1950	50

Q: What are the maximum and average magnitudes of earthquakes recorded annually?

max avg(Magnitude) group by Year

```
select extract(YEAR from Datetime) year,
      avg(Magnitude) avg_mag,
      max(Magnitude) max_mag
from `neic.earthquakes`
group by extract(YEAR from Datetime)
order by year;
```

```
select  Datetime,
      extract(YEAR from Datetime) year,
      extract(MONTH from Datetime) month,
      extract(DAY from Datetime) day,
      extract(QUARTER from Datetime) quarter,
      extract(HOUR from Datetime) hour,
      extract(MINUTE from Datetime) minute,
      extract(SECOND from Datetime) second,
      extract(DAYOFWEEK from Datetime) day_of_week,
from `neic.earthquakes`;
```

Q: How are earthquakes distributed across different months for the entire data, and which month experiences the most activity?

Sort desc. ↑ Count ↑ group by

```
select extract(MONTH from Datetime) month,
      count(ID) as earthquake_cnt
from `neic.earthquakes`
group by extract(MONTH from Datetime)
order by earthquake_cnt desc;
```

Q: Which day of the week has the highest frequency of earthquakes across all years, and what patterns emerged over time?

"dayofweek" ?

```
select extract(DAYOFWEEK from Datetime) day_of_week,
      count(ID) as earthquake_cnt
from `neic.earthquakes`
group by extract(DAYOFWEEK from Datetime)
order by earthquake_cnt desc;
```

Q: During which hours of the day do earthquakes mostly occur, indicating peak periods of seismic activity?

```
select extract(HOUR from Datetime) hour,
      count(ID) as earthquake_cnt
from `neic.earthquakes`
group by extract(HOUR from Datetime)
order by earthquake_cnt desc;
```

Hour, Time

Break till : 22:20

FORMAT_DATE()

https://cloud.google.com/bigquery/docs/reference/standard-sql/format-elements#format_elements_date_time
https://cloud.google.com/bigquery/docs/reference/standard-sql/format-elements#format_elements_date_time

string → Datetime "2024-09-20" / "2024-09"
extract →

Q: How does the frequency of earthquakes vary across different months of each year?

Count(ID)

Year-Month
2024-01
2024-02 ....

```
select format_date("%Y-%m", Datetime) year_month,
      count(ID) as earthquake_cnt
from `neic.earthquakes`
group by format_date("%Y-%m", Datetime)
order by year_month;
```

extract (year)

extract (year)
extract (month)] Concat

```
select format_date("%Y-Q%Q", Datetime) year_quarter,
    count(ID) as earthquake_cnt
from `neic.earthquakes`
group by format_date("%Y-Q%Q", Datetime)
order by year_quarter;
```

2024-Q1
2024-Q2

CURRENT_DATE()

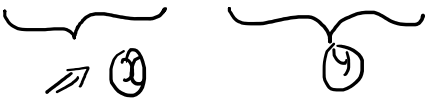
Note:

- In MySQL, the syntax for CURRENT_DATE() and CURRENT_TIME() functions remains the same.
- However, the **NOW()** function is used as an alternative to BigQuery's **CURRENT_DATETIME()** function.

Q: How many earthquakes have occurred so far this month?

```
select count(ID) num
from `neic.earthquakes`
where extract(year from Datetime) = extract(year from current_date())
    AND extract(month from Datetime) = extract(month from current_date());
```

Q: How many earthquakes have occurred from January 1965 to December 2016?



year >= 1965 month =
year <= 2016

```
select count(ID) num
from `neic.earthquakes`
where extract(year from Datetime)  >  extract(year from parse_date("%d-%m-%Y", "01-01-1965"))
    AND extract(month from Datetime)  >  extract(month from parse_date("%d-%m-%Y",
"01-01-1965"))
    AND extract(year from Datetime) < extract(year from parse_date("%d-%m-%Y", "31-12-2016"))
    AND extract(month from Datetime) < extract(month from parse_date("%d-%m-%Y",
"31-12-2016"));
```

DATE_ADD() & DATE_SUB()

Note: The syntax of DATE_ADD() & DATE_SUB() functions remains the same in MySQL.

SYNTAX:

DATE_SUB(date_expression, INTERVAL integer_expression date_part)

Q: How many earthquakes have been recorded in the past 90 days? What was the average depth and magnitude of those earthquakes?

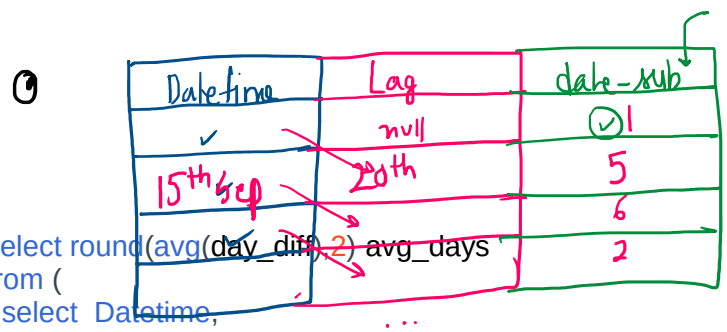
```
select count(ID) num,
       avg(Depth) avg_depth,
       avg(Magnitude) avg_mag
from `neic.earthquakes`
where Date >= date_sub(current_date(), INTERVAL 90 DAY);
```

Current_date() - 90
2024-09-20

```
-- 31-12-2016
select count(ID) num,
       avg(Depth) avg_depth,
       avg(Magnitude) avg_mag
from `neic.earthquakes`
where Date >= date_sub(parse_date("%d-%m-%Y","31-12-2016"), INTERVAL 90
DAY);
```

```
select count(ID) num,
       avg(Depth) avg_depth,
       avg(Magnitude) avg_mag
from `neic.earthquakes`
where Date >= date_sub(current_date(), INTERVAL 2000 DAY);
```

Q: What is the average interval of days between significant earthquakes with a magnitude of 7 or higher?

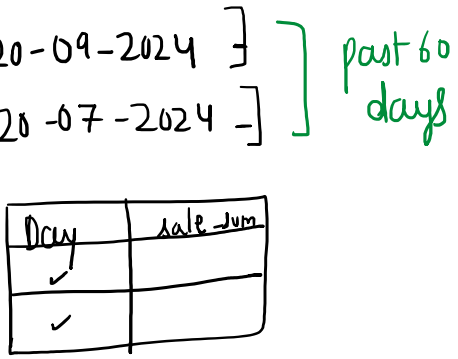


avg(date-sub)
=

```
select round(avg(day_diff),2) avg_days
from (
  select  Datetime,
        LAG(Datetime) OVER(Order by Datetime) AS prev_day,
        DATE_DIFF(Datetime, LAG(Datetime) OVER(Order by Datetime), DAY) as day_diff
  from `neic.earthquakes`
  where Magnitude > 7
  order by Datetime ) t1;
```

QUIZ:

```
SELECT
 EXTRACT(DAY FROM Date),
 SUM(Sales)
FROM tbl
WHERE
 Date BETWEEN CURRENT_DATE()
 AND DATE_SUB(CURRENT_DATE(), INTERVAL 60 Days)
GROUP BY
 EXTRACT(DAY FROM Date);
```



The sum of daily sales from the start of one month to the end of the next month.

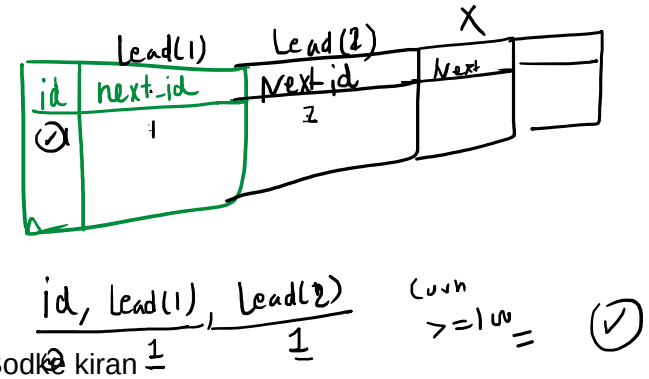
The sum of sales for every 60-day period.

The sum of daily sales for the past 60 days, starting from today.

Q: How many days have elapsed since the most recent significant earthquake with a magnitude of 7 or higher?

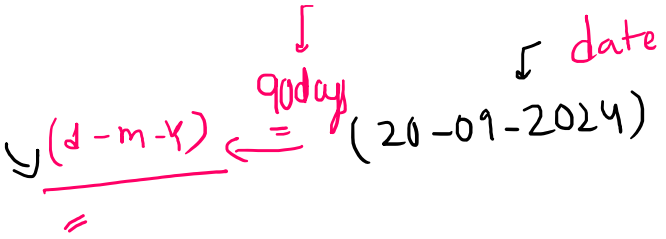
```
select
   date_diff(parse_date("%d-%m-%Y","31-12-2016"), max(Date), DAY) d_sinc_last_sig_eq
from `neic.earthquakes`
where Magnitude >=7;
```

Nancy Aspin
Greatest funtion explain sir



Bodke kiran
sir how to extract year using substring i don't know how it works just one example
`select substr("2024-09-20", 1, 4);`

Nancy Aspin
y dont mention month in the end



NEXT SESSION:
Q: What is the seasonal distribution of earthquake occurrences each year? (Winter, Spring, Summer, Fall)

Q: Based on historical data, how many earthquakes are expected in the upcoming quarter?

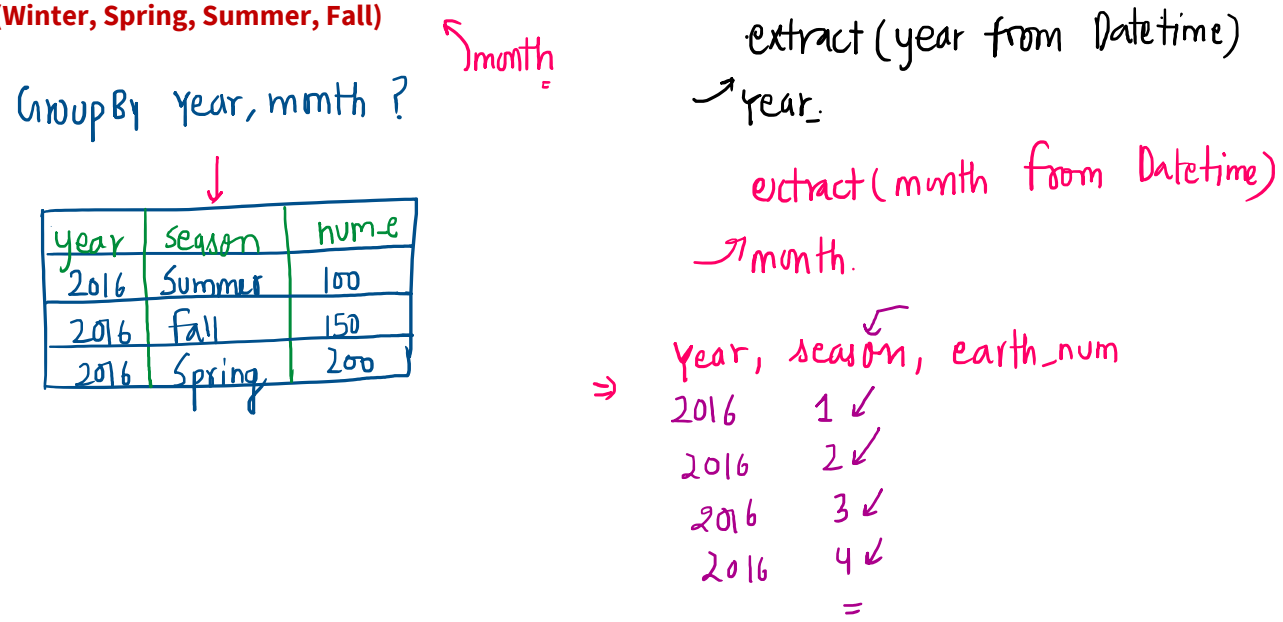
Q: What is the frequency of earthquakes and their average magnitude for each quarter over the past five years?

Session 14 -Applied SQL 3

22 September 2024 12:24

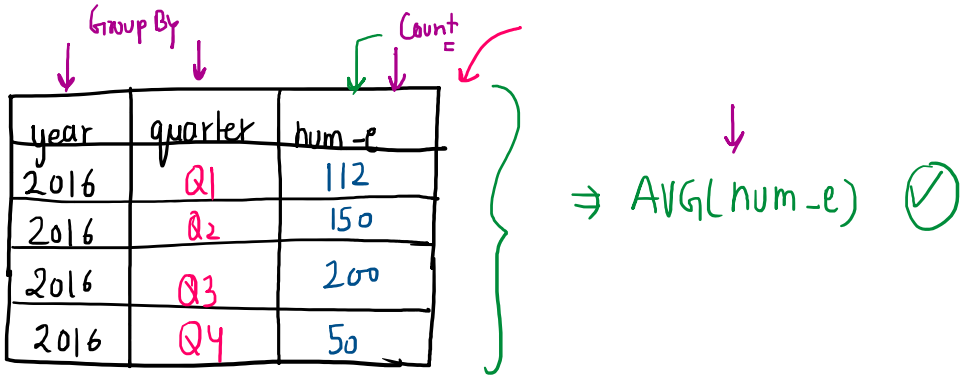
NEXT SESSION:

Q: What is the seasonal distribution of earthquake occurrences each year? (Winter, Spring, Summer, Fall)




```
select extract(YEAR from Datetime) AS year,
      CASE
       WHEN extract(MONTH from Datetime) IN (3,4,5) THEN "Spring"
       WHEN extract(MONTH from Datetime) IN (12,1,2) THEN "Winter"
       WHEN extract(MONTH from Datetime) IN (6,7,8) THEN "Summer"
       ELSE "Fall"
      END as season,
      count(*) AS num_earthquakes
from `neic.earthquakes` e
group by year,season
order by year desc,season;
```

Q: Based on historical data, how many earthquakes are expected in the upcoming quarter?




```
select round(AVG(t1.num_earthquake)) AS avg_earthquakes
from (
   select extract(YEAR from Datetime) as year,
```

```
        extract(QUARTER from Datetime) as quarter,
        count(ID) num_earthquake
    from `neic.earthquakes` e
    group by year, quarter
) t1;
```

Q: What is the frequency of earthquakes and their average magnitude for each quarter over the past five years?

yearquarter	num_earthqke	avg_mag
2016-Q1	112	6.52
2016-Q2	200	5.2
2016-Q3	100	6.1
2016-Q4	50	7.2

format_date ??

format_date(, Date)

"2016-12-31"

Where Date > date_sub(parse_date("%Y-%m-%d", "2016-12-31"), INTERVAL 5 YEAR)

"2016-12-31"

date_sub(Current_date() , INTERVAL 5 YEAR)

parse_date ↓ i/p → String ("2016-12-31")

↘ o/p → Date (2016-12-31)

↘ 2024-09-23 ✓

↘ 09-23-2024 ✓

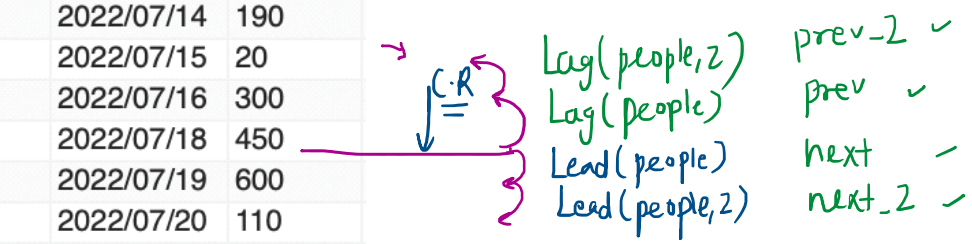
↘ 23-09-2024 ✓

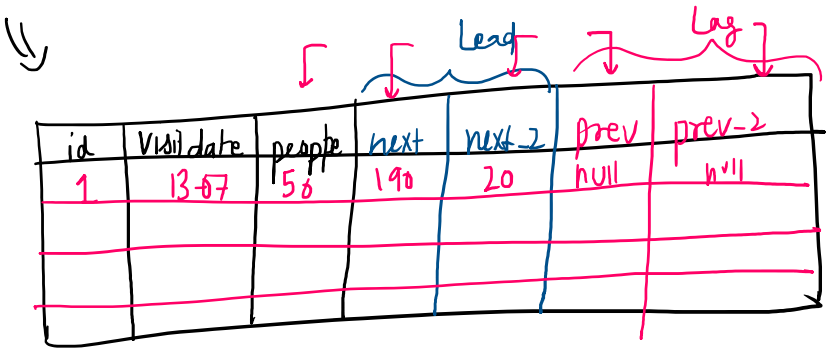
↘ 23/09/2024 ✓ ← %d / %m / %Y

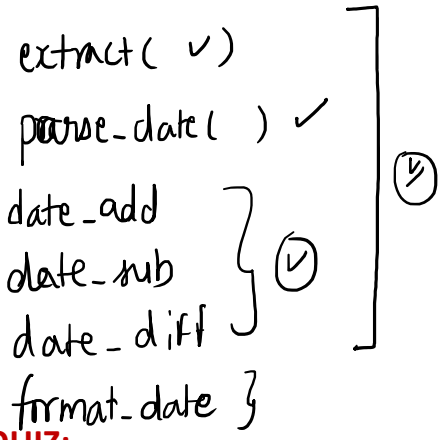
```
select format_date("%Y-Q%Q", Date) year_quarter,
      count(ID) frequency,
      round(avg(Magnitude),2) avg_mag
from  `neic.earthquakes` e
where Date > date_sub(parse_date("%Y-%m-%d", "2016-12-31"), INTERVAL 5 YEAR)
group by format_date("%Y-Q%Q", Date)
order by year_quarter desc;

-- 2016-12-31
```

Break Till : 22:16

id	visit_date	people
1	2022/07/13	50
2	2022/07/14	190
3	2022/07/15	20
4	2022/07/16	300
5	2022/07/18	450
6	2022/07/19	600
7	2022/07/20	110
8	2022/07/21	220





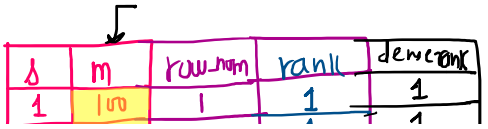


QUIZ:

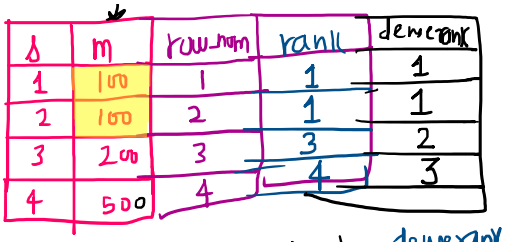
1. What is a Window Function in SQL?
   a) A function that returns a single row
   b) A function that operates over a range of rows
   c) A function that removes duplicates
   d) A function that changes data types

2. Which clause is mandatory when using a window function?
   a) GROUP BY
   b) ORDER BY
   c) PARTITION BY
   d) OVER

3. What is the difference between RANK() and ROW_NUMBER()?
   a) RANK() skips ranks when there is a tie, ROW_NUMBER() doesn't
   b) ROW_NUMBER() skips rows when there is a tie, RANK() doesn't
   c) RANK() is faster
   d) There is no difference

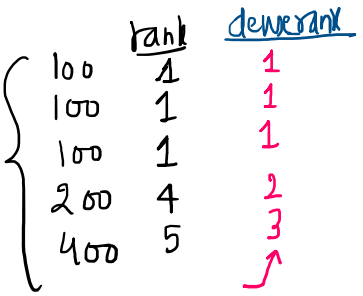


b) ROW_NUMBER() skips rows when there is a tie, RANK() doesn't
c) RANK() is faster
d) There is no difference



4. Consider the following query:

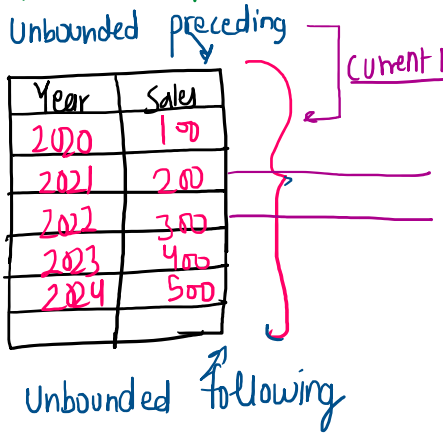
SELECT department, employee, salary,
RANK() OVER (PARTITION BY department ORDER BY salary DESC) as rank
FROM employees;
What does the query do?



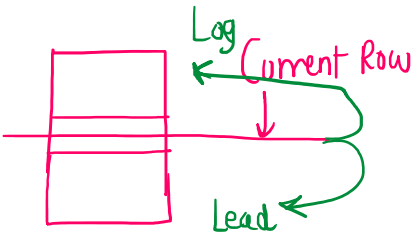
a) It ranks employees across all departments
b) It ranks employees within each department based on salary
c) It assigns a unique rank to every employee
d) It partitions salaries by department and does nothing else

5. Which window function would you use to calculate a cumulative sum?
a) RANK()
b) ROW_NUMBER()
c) SUM()
d) CUME_DIST()

① sum(revenue) over (partition by year) as revenue.
② sum(revenue) over (partition by year Order by Sales) as revenue



6. What is the difference between LAG() and LEAD() in window functions?
a) LAG() gets the previous row, LEAD() gets the next row
b) LEAD() gets the previous row, LAG() gets the next row
c) LAG() returns the highest row value
d) LEAD() returns the highest row value



7. What does NTILE(4) do in a window function?
a) Divides rows into 4 equal parts and assigns a group number
b) Returns the top 4 rows
c) Divides rows based on salary
d) Assigns the number 4 to each row

rows = 100
rows = 101 ? =>

8. Which window function would you use to find the average value within a partition?
   a) RANK()
   b) AVG()
   c) ROW_NUMBER()
   d) LEAD()

→ Select avg(salary) over (partition by department) as avg_sal

9. What will the following query return?

```
SELECT employee, salary,
SUM(salary) OVER (ORDER BY salary ROWS BETWEEN UNBOUNDED PRECEDING AND CURRENT ROW) as
running_total
FROM employees;
```

Default

   a) The total salary for all employees
   b) A running total of salaries ordered by employee names
   c) A cumulative sum of salaries in ascending order
   d) The average salary

Syntax Error
→ Consumple sum

sum(_) over(order by <col> rows b/w unbounded preceding & Unbounded following) ⇒

10. How would you reset a rank within a window function?
   a) Use ORDER BY
   b) Use GROUP BY
   c) Use PARTITION BY
   d) Use WHERE

11. Which of the following is true about DENSE_RANK()?
   a) It behaves the same as RANK()
   b) It skips rank numbers when there is a tie
   c) It assigns consecutive rank numbers, even in the case of ties
   d) It gives the highest rank to the first row

12. What will the following query return?

```
SELECT employee, salary,
LEAD(salary, 1) OVER (ORDER BY salary) AS next_salary
FROM employees;
```

Lead(salary) ≈ Lead(salary, 1)

Salary

   a) The salary of the next employee in the salary order
   b) The salary of the previous employee
   c) The sum of all salaries
   d) The cumulative salary of all employees

d) The cumulative salary of all employees

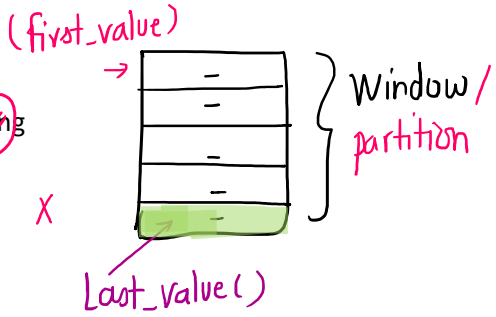
13. What does the following query achieve?

```
SELECT department, employee, salary,
PERCENT_RANK() OVER (PARTITION BY department ORDER BY salary DESC) AS pct_rank
FROM employees;
```

a) It ranks employees within each department based on their salary, as a percentage between 0 and 1
b) It gives employees a random percentage based on salary
c) It assigns an exact rank to each employee
d) It gives the percentage of total salary for each department

14. What does the FIRST_VALUE() function do in SQL?

a) Returns the first row from a result set
b) Returns the first value in a partition based on the ORDER BY clause
c) Returns the first alphabetically ordered value
d) Returns the first value that appears in a column, regardless of partitioning

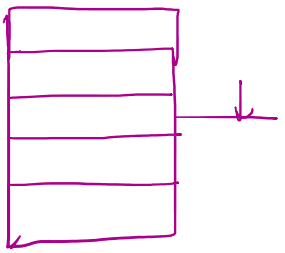


15. What does the LAST_VALUE() function do in SQL?

a) Returns the last row from a result set
b) Returns the last value in a partition based on the ORDER BY clause
c) Returns the highest value in a set of rows
d) Returns the last value that appears in a column, regardless of partitioning

16. How can you calculate a rolling average of the last 3 rows in a result set?

a) AVG() OVER (ROWS BETWEEN 2 PRECEDING AND CURRENT ROW)
b) AVG() OVER (ROWS BETWEEN 3 PRECEDING AND CURRENT ROW)
c) AVG() OVER (ROWS BETWEEN UNBOUNDED PRECEDING AND CURRENT ROW)
d) AVG() OVER (PARTITION BY employee ORDER BY salary)



17. Which of the following window functions provides a relative rank between 0 and 1?

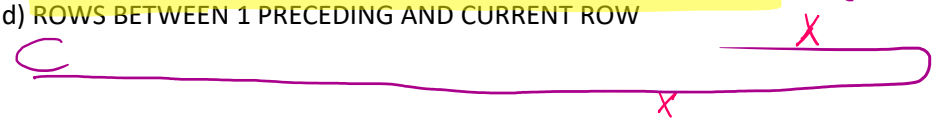
a) NTILE()
b) PERCENT_RANK()
c) CUME_DIST()
d) ROW_NUMBER()

18. What does CUME_DIST() return?

a) The relative rank of a row within a partition as a percentage
b) The cumulative distribution of a value within a set
c) The number of rows before the current row
d) The sum of the distribution values in a partition

19. What is the default window frame in a window function when ROWS is not specified?

a) ROWS BETWEEN UNBOUNDED PRECEDING AND CURRENT ROW
b) ROWS BETWEEN CURRENT ROW AND UNBOUNDED FOLLOWING
c) ROWS BETWEEN UNBOUNDED PRECEDING AND UNBOUNDED FOLLOWING
d) ROWS BETWEEN 1 PRECEDING AND CURRENT ROW



20. What will the following query return?

```
SELECT employee, salary,
SUM(salary) OVER (PARTITION BY department ORDER BY salary) as cumulative_salary
FROM employees;
```

a) The total salary of all employees
b) The cumulative salary per department, ordered by salary
c) The sum of all salaries ordered by department
d) The average salary per department

Session 15 - Advanced Constructs - CTEs & Views

24 September 2024 18:10

Agenda DSML Aug24 Beginner 2

1. Problem Statement
2. Ad-hoc Reporting
3. CTEs & their Advantages
4. Break
5. Views
6. When to use CTE vs View
7. Practice Question

Problem Statement

RelianceGuard Insurance is a leading provider of insurance policies, committed to offering exceptional coverage and services to its diverse customer base. However, the company has recently observed a decline in customer retention rates. To address this issue, RelianceGuard Insurance aims to analyze its customer and policy data to gain insights into customer behaviour and identify factors influencing policy renewals. By leveraging this analysis, the company seeks to ultimately improve its overall customer retention rates and maintain a competitive edge in the insurance market.

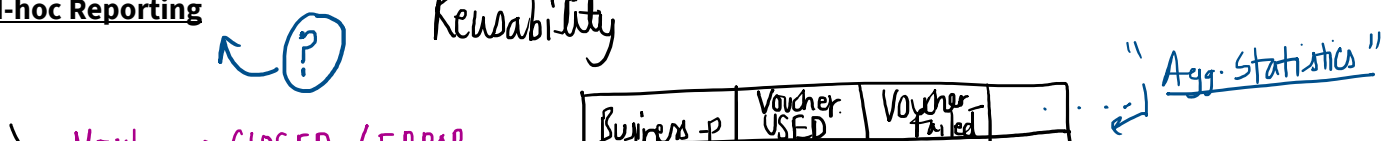
Data Dictionary:

- **Customer:** Customer ID number
- **Customer_Lifetime_Value:** Customer's total worth to the business over the life of the relationship
- **Response:** True or False response to a renewal offer
- **Coverage:** Type of policy (Basic, Extended, Premium)
- **Monthly_Premium_Auto:** Amount of customer's monthly insurance payments
- **Months_Since_Last_Claim:** Number of months between the customer's last reported insurance claim
- **Months_Since_Policy_Inception:** Number of months since the customer began an insurance policy
- **Number_of_Open_Complaints:** Number of unresolved customer complaints
- **Number_of_Policies:** Number of policies the customer currently owns
- **Policy_Type:** (Corporate Auto, Personal Auto, Special Auto)
- **Policy:** 3 levels (L1, L2, L3) per Policy Type (Corporate, Personal, Special)
- **Renew_Offer_Type:** 4 types of renewal offers (Offer 1, Offer 2, Offer 3, Offer 4)
- **Sales_Channel:** Channels to purchase a policy (Agent, Branch, Call Center, Web)
- **Total_Claim_Amount:** Cumulative amount of claims since policy inception

Dataset link: LINK

Ad-hoc Reporting

Reusability



↘ Voucher ⇒ CLOSED / ERROR

↘ Airtel, Idea, MTN

Business	Voucher USED	Voucher failed	
Airtel	100	50	
Idea	-	-	
T-mobile	-	-	
MTN-			

"Agg. Statistics"

← CTE

Q. Analyse customer retention by sales channel to identify which channels are most effective at retaining customers.

↑ group by

Step 1: Filter out the data that we need.

```
SELECT Sales_Channel,
    SUM(CASE WHEN Response = true THEN 1 ELSE 0 END) AS
retained_customers
from `rg.relianceguard_insurance`
group by Sales_Channel;
```

Channel	retained cust.
Web	100

retained cust.

100 / (500) = 20%

↖ total Cust.

Step 2: Calculate retention rates.

retained_cust. ↓ total Count ↓

```
SELECT Sales_Channel,
    (SUM(CASE WHEN Response = true THEN 1 ELSE 0 END) * 1.0 / count(*)) AS
retained_ratio
from `rg.relianceguard_insurance`
group by Sales_Channel;
```

	2023	2024
Bajaj =	900	300
Reliance =	500 ↑	800 ↑

Step 3: Extract the top Sales channel.

```
SELECT Sales_Channel,
    (SUM(CASE WHEN Response = true THEN 1 ELSE 0 END) * 1.0 / count(*)) AS
retained_ratio
from `rg.relianceguard_insurance`
group by Sales_Channel
order by retained_ratio desc
limit 1;
```

Common Table Expressions (CTEs)

Goal : store queries (and the results of queries) for reuse in reports and other analyses.

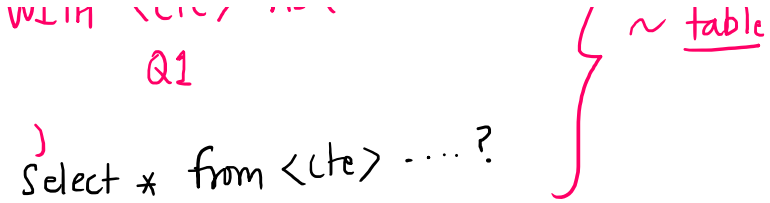
↘ 1. CTE (Commn table Expression) ✓

2. Views ✓

WITH <cte> AS (Alias

Q1

} ~ table



QUIZ:

CTEs can only be referenced once in a single SQL query. "False"

CTE Usecase & Advantages

Question 1: Among RelianceGuard Insurance's various sales channels, which channel boasts the highest customer retention rate?

```
WITH sales_by_channel AS (
    SELECT  Sales_Channel,
            count(*) AS total_customers,
            SUM(CASE WHEN Response = true THEN 1 ELSE 0 END) AS
retained_customers
    from `rg.relianceguard_insurance`
    group by Sales_Channel
),
retention_rates AS (
    SELECT Sales_Channel,
           (retained_customers * 1.0 / total_customers) * 100 AS retention_rates
    from sales_by_channel
)
select * from retention_rates
order by retention_rates desc
limit 1;
```

Recursive CTE ?

Break till : 22:28

Homework: - Provided at the end.

Question 2: Write a query to analyze customer profiles and renewal offer types to identify the most receptive customer segments for each offer.

Step 1: Calculate response rates for each combination of renewal offer type and customer profile attributes

Renew_Offer_type
State,
Gender,
Education,
Employment_Status,
Marital_Status,
Vehicle_Class,
Vehicle_Size,

Step 2: Rank these combinations by response rate.

Step 3: Select the top combinations with the highest response rates.

SOLUTION:

Views = Another approach to CTE. ☺

Query → View ≈ "table"

Views ~ A bit slower in execution time.

QUIZ:

Queries that reference views typically execute faster than queries that directly access tables

View ≈ table
(re-usable Query)
)

false

Question: RelianceGuard Insurance is concerned about potential customer churn and wants to identify customers who might be at risk of not renewing their policies. To achieve this, write a query to analyze customer data from the past month (as of March 1st, 2024).

CUSTOMER
Low Engagement - Months_Since_Last_Claim
Complaints - Number_of_Open_Complaints

Prakash	-	-	0
Saurabh			1
Sanket			0
Kiran			1

→ Sum(Response) =

Table → Data

View → Query =

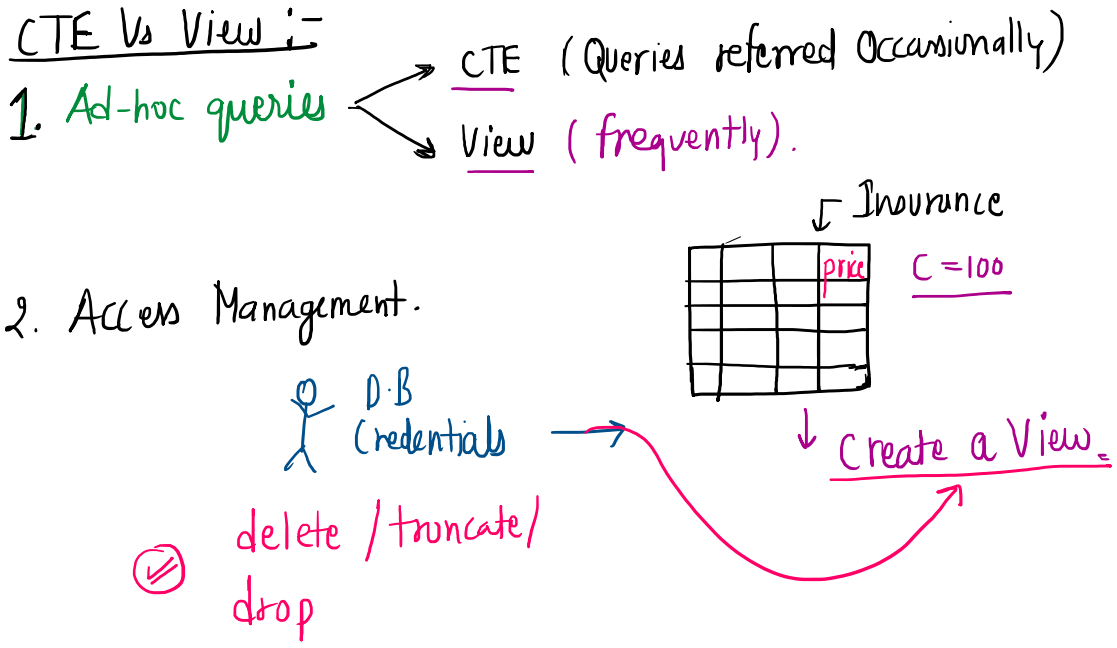
QUIZ:

What is the primary difference between a database view and a table in SQL?

Query ↑ view

↑ table → data

CTE Vs View :-



Homework:

Question 2: Write a query to analyze customer profiles and renewal offer types to identify the most receptive customer segments for each offer.

Step 1: Calculate response rates for each combination of renewal offer type and customer profile attributes

Renew_Offer_type
State,
Gender,
Education,
Employment_Status,
Marital_Status,
Vehicle_Class,
Vehicle_Size,

Step 2: Rank these combinations by response rate.

Step 3: Select the top combinations with the highest response rates.

SOLUTION:

```
WITH Response_Rates_CTE AS (
    SELECT
        Renew_Offer_Type,
        State,
        Gender,
        Education,
        Employment_Status,
        Marital_Status,
        Vehicle_Class,
        Vehicle_Size,
        COUNT(*) AS Total_Customers,
        SUM(CASE WHEN Response = true THEN 1 ELSE 0 END) AS Total_Accepted,
        (SUM(CASE WHEN Response = true THEN 1 ELSE 0 END) * 1.0 / COUNT(*)) AS Response_Rate
    FROM `rg.relianceguard_insurance`
    GROUP BY
        Renew_Offer_Type,
        State,
        Gender,
        Education,
        Employment_Status,
```

```
        Marital_Status,
        Vehicle_Class,
        Vehicle_Size
),
Ranked_Response_Rates_CTE AS (
    SELECT
        Renew_Offer_Type,
        State,
        Gender,
        Education,
        Employment_Status,
        Marital_Status,
        Vehicle_Class,
        Vehicle_Size,
        Response_Rate,
        DENSE_RANK() OVER (PARTITION BY Renew_Offer_Type ORDER BY Response_Rate DESC) AS Response_Rank
    FROM Response_Rates_CTE
)
SELECT  Renew_Offer_Type,
        State,
        Gender,
        Education,
        Employment_Status,
        Marital_Status,
        Vehicle_Class,
        Vehicle_Size,
        Response_Rate
  FROM Ranked_Response_Rates_CTE
  WHERE Response_Rank = 1;
```

Session 16 - SQL Query Optimization

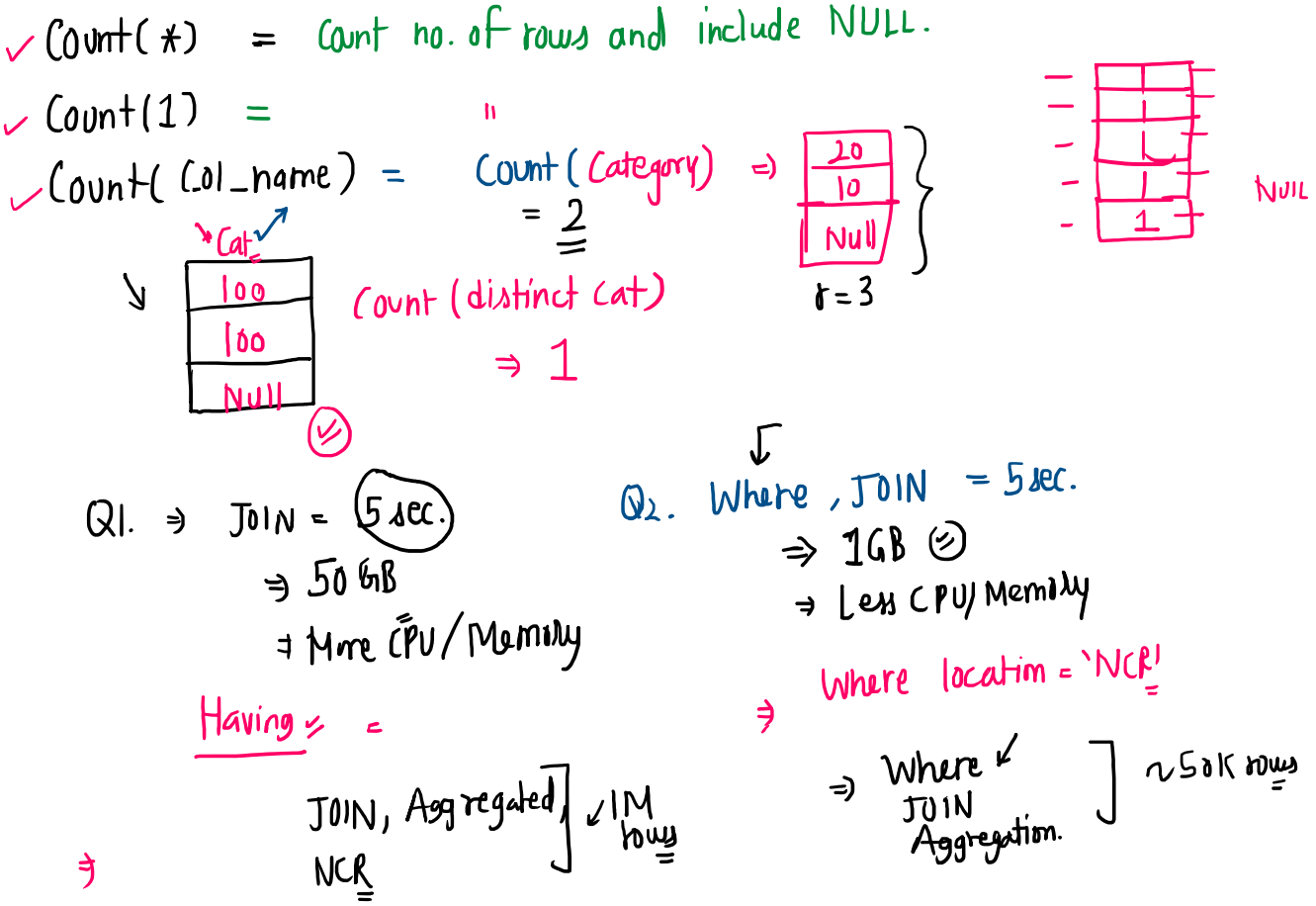
26 September 2024 21:49

DSML Aug24 Beginner 2

Agenda

1. Problem Statement
2. Busting a common myth about the COUNT()
3. Only select the columns that you really need
4. LIMIT is a trap
5. Use EXISTS() instead of COUNT()
6. Use APPROX_COUNT_DISTINCT instead of COUNT(DISTINCT)
7. Replace Self-Join with Windows Function
8. Trim your data early and often
9. Use MAX() instead of RANK()
10. Order your JOINs from larger tables to smaller tables
11. Does WHERE sequence matter?
12. Should we push ORDER BY to the end of the query?

1. Busting a common myth about the COUNT() function



2. Only select the columns that you really need

```
SELECT
    timestamp,
    number,
    transactions_root, state_root, receipts_root,
    miner,
    difficulty, total_difficulty,
    size,
    extra_data,
    gas_limit, gas_used,
    transaction_count,
```

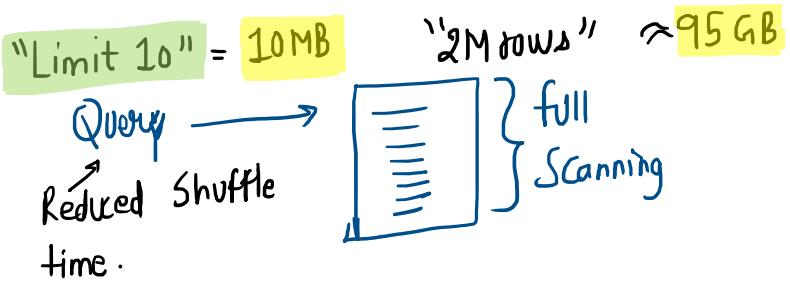
bytes shuffled reduced by a factor of 1/4.

```
    base_fee_per_gas
FROM bigquery-public-data.crypto_ethereum.blocks;
```

```
SELECT * FROM
  bigquery-public-data.crypto_ethereum.blocks;
```

3. LIMIT is a trap

↳ Speed up performance.
↳ Doesn't reduce cost.



4. Use EXISTS() instead of COUNT()

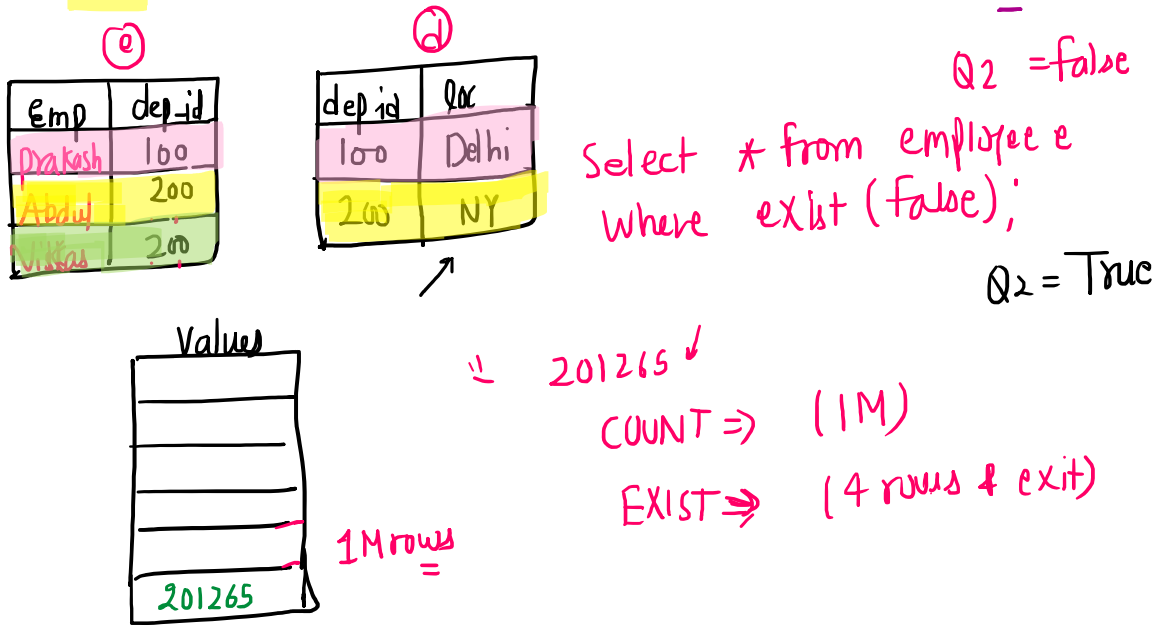
. Show all emp where dep_loc = "NY" ✓

```
Select * from employee
Where dep_id IN ( select id from department where loc = "NY");
```

More time

```
Select * from employee e
Where EXIST (select 1 from department d where d.dep_id = e.dep_id and d.loc = "NY")
```

Less time Q2



Q2 = false

Select * from employee e
Where exist (false);

Q2 = True

201265
COUNT ⇒ (1M)
EXIST ⇒ (4 rows & exit)

5. Use APPROX_COUNT_DISTINCT instead of COUNT(DISTINCT) for large datasets

↘ Count(DISTINCT u_name)
↘ ⇒ APPROX_COUNT_DISTINCT ( )

```
SELECT
  APPROX_COUNT_DISTINCT(miner)
  FROM bigquery-public-data.crypto_ethereum.blocks
  WHERE
  timestamp BETWEEN '2015-01-01' AND '2023-12-31';
```

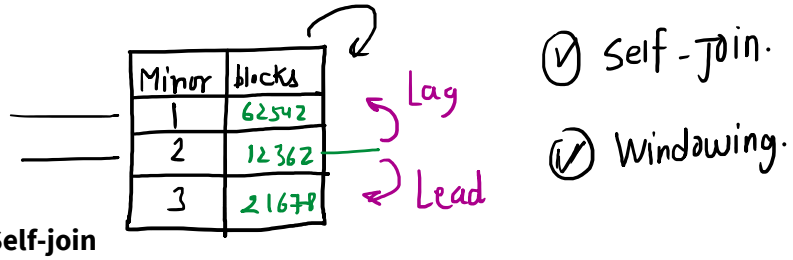
```
SELECT
  COUNT(DISTINCT miner)
  FROM bigquery-public-data.crypto_ethereum.blocks
  WHERE
  timestamp BETWEEN '2015-01-01' AND '2023-12-31'
```

Break Till : 22:32

6. Replace Self-Join with Window Functions

Self-join usually requires more reads than window functions, therefore slower.

Question: We want to know the difference between the number of Ethereum blocks mined today and yesterday by each miner.



Self-join

```
WITH cte_table AS (
        SELECT
          DATE(timestamp) AS date,
          miner,
          COUNT(DISTINCT number) AS block_count
        FROM bigquery-public-data.crypto_ethereum.blocks
        WHERE
          DATE(timestamp) BETWEEN "2015-01-01" AND "2023-12-31"
        GROUP BY 1,2 )
      SELECT
        a.miner,
        a.date AS today,
        a.block_count AS today_count,
        b.date AS tmr,
        b.block_count AS tmr_count,
        b.block_count - a.block_count AS diff
      FROM cte_table a
      LEFT JOIN cte_table b
      ON
        DATE_ADD(a.date, INTERVAL 1 DAY) = b.date
        AND a.miner = b.miner
      ORDER BY
        a.miner, a.date
```

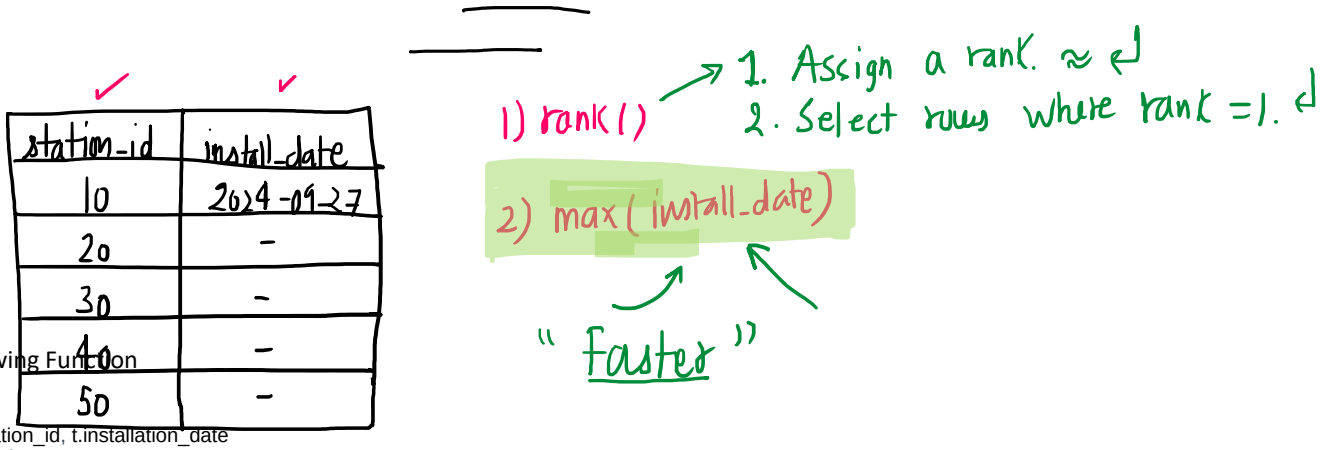
Windowing

```
WITH cte_table AS (
        SELECT
        DATE(timestamp) AS date,
        miner,
        COUNT(DISTINCT number) AS block_count
      FROM bigquery-public-data.crypto_ethereum.blocks
      WHERE
        DATE(timestamp) BETWEEN "2015-01-01" AND "2023-12-31"
      GROUP BY 1,2 )
      SELECT
        miner,
        date AS today,
        block_count AS today_count,
        LEAD(date, 1) OVER (PARTITION BY miner ORDER BY date) AS tmr,
        LEAD(block_count, 1) OVER (PARTITION BY miner ORDER BY date) AS tmr_count,
        LEAD(block_count, 1) OVER (PARTITION BY miner ORDER BY date) -
      block_count AS diff
      FROM cte_table a
```

7. **Trim your data early and often**
   Filter as early as possible

8. **Use MAX() instead of RANK()**

Question: The team has a general assumption that the older the establishment the more popular it'll be. To verify the same assumption, fetch the station ids and their respective date of installation in order starting from the one installed most recently.



Windowing Function

```
SELECT
    t.station_id, t.installation_date
  FROM (
    SELECT station_id,
        installation_date,
    RANK() OVER(PARTITION BY station_id ORDER BY installation_date DESC) AS rnk
    FROM bigquery-public-data.san_francisco.bikeshare_stations) t
  WHERE rnk = 1
  ORDER BY t.installation_date DESC;
```

Max Function

```
SELECT
    station_id,
    MAX(installation_date) AS doi
  FROM bigquery-public-data.san_francisco.bikeshare_stations
  GROUP BY 1
  ORDER BY doi DESC;
```

9. **Order your JOINs from larger tables to smaller tables**

Question: San Francisco is a big city and usually has a good number of bike trips. Find the number of bikes and docks currently available at all stations in San Francisco so that proper restocking can be done.

t1 JOIN t2 ↓

↑ Larger

Customer = 1M

Orders = ✓ Large ☑]

Bike_station =

bikeshare_status =

Case 1:

```
SELECT
   t1.station_id,
   t1.name,
   t2.bikes_available,
   t2.docks_available
FROM
    `bigquery-public-data.san_francisco.bikeshare_stations` t1
JOIN
   `bigquery-public-data.san_francisco.bikeshare_status` t2
ON
   t1.station_id = t2.station_id
WHERE
   t1.landmark = 'San Francisco';
```

Case 2: Larger table on Left side of join.

```
 SELECT
     t2.station_id,
     t2.name,
     t1.bikes_available,
     t1.docks_available
  FROM bigquery-public-data.san_francisco.bikeshare_status t1
  JOIN bigquery-public-data.san_francisco.bikeshare_stations t2
  ON t1.station_id = t2.station_id
  WHERE t2.landmark = 'San Francisco';
```

10. Does WHERE sequence matters?

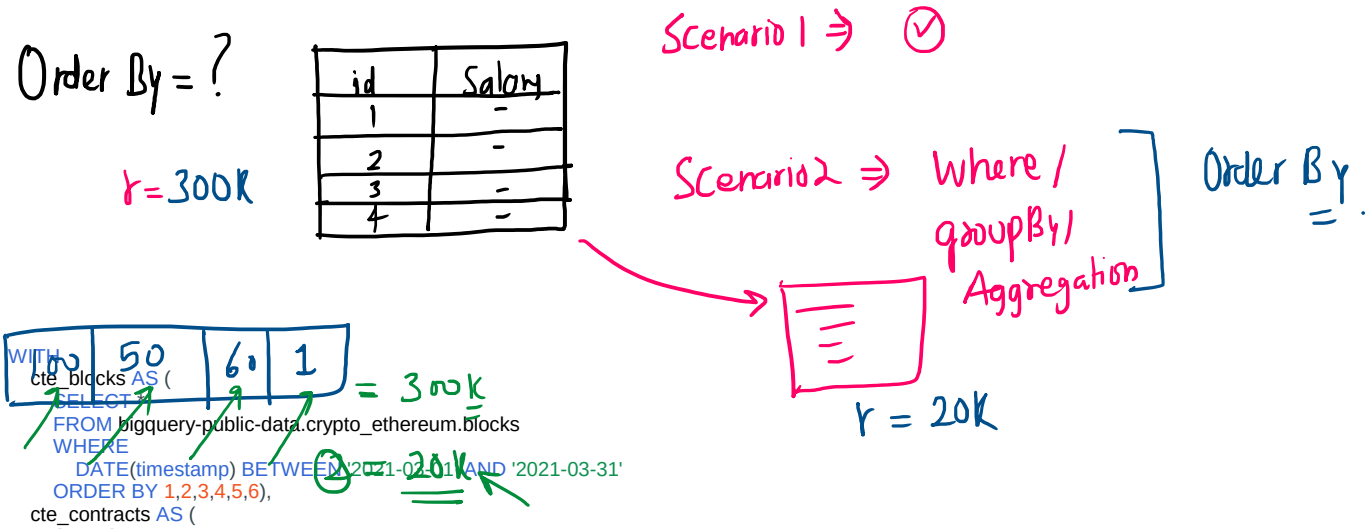
Select * from table
where miner like "%a"
and miner like "%b"
and miner = "1001" Q1

Select * from table
where miner = "1001"
and miner like "%a"
and miner like "%b" Q2

```
SELECT
  miner
  FROM bigquery-public-data.crypto_ethereum.blocks
  WHERE
  miner LIKE '%a%'
  AND miner LIKE '%b%'
  AND miner = '0xc3348b43d3881151224b490e4aa39e03d2b1cdea';
```

```
SELECT
  miner
  FROM bigquery-public-data.crypto_ethereum.blocks
  WHERE
  miner = '0xc3348b43d3881151224b490e4aa39e03d2b1cdea'
  AND miner LIKE '%a%'
  AND miner LIKE '%b%';
```

11. Should we push ORDER BY to the end of the query?




```
WITH
  cte_blocks AS (
    SELECT *
    FROM bigquery-public-data.crypto_ethereum.blocks
    WHERE
      DATE(timestamp) BETWEEN '2021-03-01' AND '2021-03-31'
    ORDER BY 1,2,3,4,5,6),
  cte_contracts AS (
    SELECT *
    FROM bigquery-public-data.crypto_ethereum.contracts
    WHERE
      DATE(block_timestamp) BETWEEN '2021-03-01' AND '2021-03-31'
    ORDER BY 1,2,4,5,6,7)
  SELECT *
  FROM cte_blocks b
  LEFT JOIN cte_contracts c
  ON c.block_number = b.number
  ORDER BY size, block_hash;
```

```
WITH
  cte_blocks AS (
    SELECT *
    FROM `bigquery-public-data.crypto_ethereum.blocks`
    WHERE
      DATE(timestamp) BETWEEN '2021-03-01' AND '2021-03-31'),
  cte_contracts AS (
    SELECT *
    FROM `bigquery-public-data.crypto_ethereum.contracts`
    WHERE
      DATE(block_timestamp) BETWEEN '2021-03-01' AND '2021-03-31')
  SELECT *
  FROM cte_blocks b
  LEFT JOIN cte_contracts c
  ON c.block_number = b.number
  ORDER BY size, block_hash;
```

https://www.scaler.com/hire/test/problem/23553/

```
select      jh.employee_id,
            concat(first_name,' ',last_name) 'full_name',
            job_title
from employees emp join job_history  jh
on  jh.employee_id = emp.employee_id
join jobs job
on  jh.job_id = job.job_id
where (datediff(end_date,start_date) /365) < 1
order by employee_id, job_title;
```

Sanket
exists if returns once it find the condition then what will happen to the rows below which needs to be counted

Nancy Aspin
count(*)=1 explain sir

Session 17 - MySQL setup & Intro to DDL, DML commands

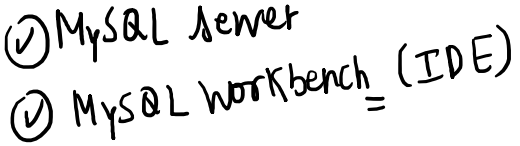
28 September 2024 11:18

LinkedIn : https://www.linkedin.com/in/chauhan-prakash/

Class will start at 09:05

Agenda:

1. Problem Statement
2. MySQL Workbench Setup
3. Types of SQL commands ✓
4. Constraints ✓
5. Data Types ✓
6. DDL commands
   a. CREATE
   b. ALTER
   c. TRUNCATE
   d. DROP
7. DML commands
   a. INSERT
   b. UPDATE
   c. DELETE
8. TRUNCATE vs. DROP vs. DELETE

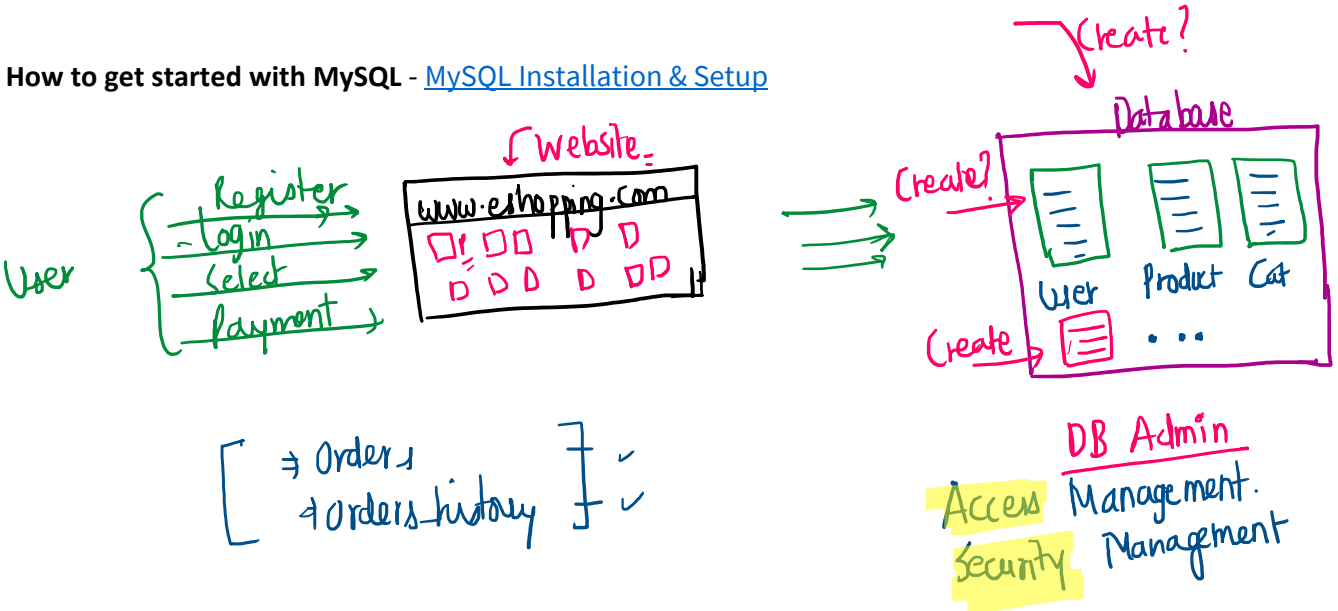


Bookworm Paradise

Bookworm Paradise, established in 2022 by Aleksander Vlad, is a leading online bookstore offering over 1 million titles, from classics to bestsellers, across various genres, including fiction, non-fiction, children's books, etc.
As Bookworm Paradise expands globally, it continues to innovate and enhance its digital platform, aiming to be the go-to destination for book lovers worldwide.

Challenges: As a data administrator for Bookworm Paradise, you need to design and implement a customer database to effectively manage customer information and track their activity.
This database will be crucial for personalizing customer experiences and analysing purchasing trends to optimize marketing strategies.

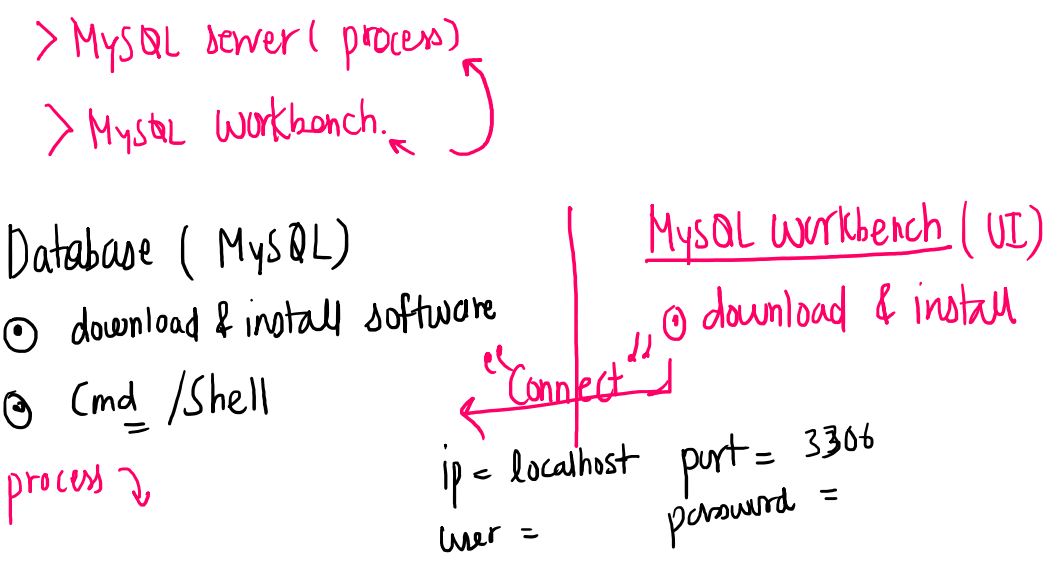
How to get started with MySQL - MySQL Installation & Setup



SQL Commands

- DDL - Data Definition Language — CREATE, ALTER, DROP, TRUNCATE ✓

- **DDL** - Data Definition Language
- **DML** - Data Manipulation Language
- **TCL** - Transaction Control Language
- **DQL** - Data Query Language - most important
- **DCL** - Data Control Language



CREATE command

CREATE [Database | Table | View | Procedure] <Name>;

```
create database bookworm_paradise;

USE bookworm_paradise;

CREATE TABLE customers (
    ID INT AUTO_INCREMENT PRIMARY KEY,
    first_name VARCHAR(20) NOT NULL,
    last_name VARCHAR(20),
    age INT CHECK(age>=18),
    gender ENUM("M", "F"),
    phone_no CHAR(10) NOT NULL UNIQUE,
    email_id VARCHAR(30),
    dob DATE,
    address VARCHAR(100)
);

select * from customers;
```

Constraints

Constraints

1. Ensure that a value for a column is unique = UNIQUE
2. Ensure that a column can't be Null = NOT NULL
3. Ensure => (Not Null + Unique) = Primary Key.
4. foreign key =>
5. CHECK => Value in a Column satisfies a condition.
6. Default => a defaut value when there is no value provided. "NA"

QUIZ:

You are creating a table to store user login information. Each user should have a different username. ← "Unique"
Additionally, you want to ensure that the password column cannot contain NULL values.
Which combination of SQL constraints should you use for these columns?

A
UNIQUE for username, NOT NULL for password

B
FOREIGN KEY for username, CHECK for password

C
UNIQUE for username, PRIMARY KEY for password

D
CHECK for username, NOT NULL for password

QUIZ:

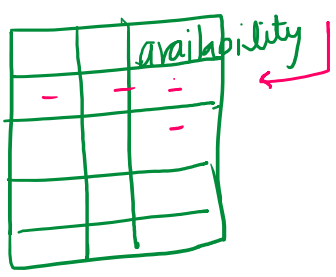
You are designing a table to store product data for an e-commerce website. For the "availability" column, you want to set a value of "In Stock" if no value is specified during insertion.
Which SQL constraint should you use?

A
UNIQUE

B
NOT NULL

C
PRIMARY KEY

D
DEFAULT



Data Types

https://www.w3schools.com/sql/sql_datatypes.asp

1. String = Char(3), [IND, INDIA, IN] = fixed Length.

= Varchar(5), [INDIA, IND, IN, AMERICA] = Variable Length

= Enum ("Male", "Female")

("Yes", "No")

("COMPLETED", "OPEN", "PENDING", "FAILED")

2. Numeric = INT (2, 10, -2, -10, 100)

= float (size, d) (100.23)

= Bool (True/false) 1 0

3. Date/Time = Date ("2024-09-30")
%Y %m %d

= Time (22:15:00)
HH MM SS

= Datetime (2024-09-30 22:15:00)

= Timestamp ('2024-09-30 22:15:00' UTC)

QUIZ:

You are designing a database for a product catalogue.
You want to store the product's weight, which can include decimal values.
Which SQL data type should you use for the "weight" column?

float?

A
Int ✗

+ weight

B
Char(5) ✗

C
Float(8, 2) ✓

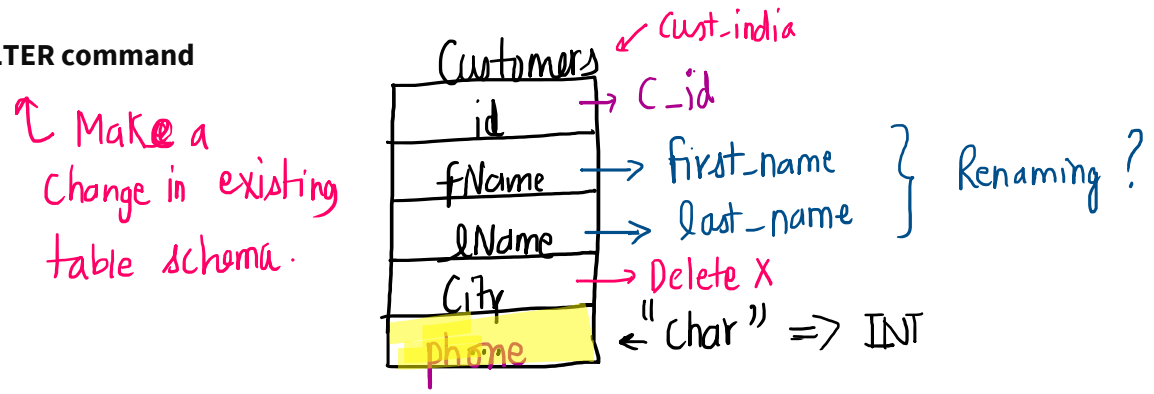
D
Varchar(10) ✗

Break Till : 22:27

ALTER command cust_india

Break Till : 22:27

ALTER command



- Add a new column named 'is_active' to the "customers" table.
  ALTER TABLE customers
  ADD is_active varchar(10);

- Change the data type of the 'is_active' column to INT.
  ALTER TABLE customers
  MODIFY is_active INT;

- Add constraint to the 'is_active' column.
  -- 1/0
  ALTER TABLE customers
  ADD constraint con CHECK(is_active IN (1,0));

- Rename the 'ID' column to 'cust_id'.
  ALTER TABLE customers
  rename column ID to cust_id;

- Delete the 'address' column from the "customers" table.
  ALTER TABLE customers
  drop column address;

- Rename the "customers" table to "cust_info".
  ALTER TABLE customers
  rename to cust_info;

QUIZ:

You have a table named "employees" with a column called "salary." Due to a change in business rules, you need to change the data type of the "salary" column from INT to DECIMAL(10, 2).
Which SQL command should you use?

A
ALTER TABLE employees MODIFY COLUMN salary DECIMAL(10, 2);

B
ALTER TABLE employees CHANGE COLUMN salary DECIMAL(10, 2);

C
ALTER TABLE employees ADD salary DECIMAL(10, 2);

D
ALTER TABLE employees ALTER COLUMN salary DECIMAL(10, 2);

ALTER TABLE employees ADD salary DECIMAL(10, 2);

D
ALTER TABLE employees ALTER COLUMN salary DECIMAL(10, 2);

QUIZ:
You have a table named "students" with a column called "student_id" and you want to change it to "id". Which SQL command should you use to perform this operation?

A
ALTER TABLE students MODIFY COLUMN student_id TO id;

B
ALTER TABLE students ALTER COLUMN student_id TO id;

C
ALTER TABLE students CHANGE COLUMN student_id TO id;

D
ALTER TABLE students RENAME COLUMN student_id TO id;

INSERT command

```
-- INSERT INTO customers(col1,col2,...) VALUES(val1,val2,...);

INSERT INTO cust_info
VALUES(1001,"John","Doe",30,"M","1929929290","admin@admin.com","2000-01-01",1);

INSERT INTO cust_info
VALUES(1002,"John","Doe",20,"M","1929929291","admin@admin.com","2000-01-01",1);

INSERT INTO cust_info(cust_id,first_name,last_name,age,gender,phone_no,email_id,dob,is_active)
VALUES(1003,"John","Doe",20,"M","1929929299","admin@admin.com","2000-01-01",1);

INSERT INTO cust_info(cust_id,first_name,last_name,age,gender,phone_no,email_id,dob)
VALUES(1004,"John","Doe",20,"M","9718707585","admin@admin.com","2000-01-01");

INSERT INTO cust_info(first_name,last_name,age,gender,phone_no,email_id,dob)
VALUES("Prakash","Chauhan",20,"M","9718707581","admin@admin.com","2000-01-01");

select * from cust_info;
```

UPDATE command

```
UPDATE cust_info
 set age = 60,
        email_id = "john_doe@gmail.com"
 where cust_id = 1001;
```

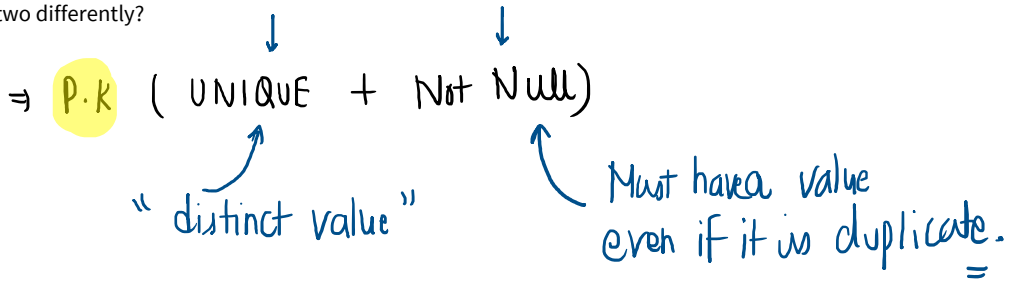
TRUNCATE vs DROP vs DELETE

```
DELETE from cust_info
        where cust_id = 1002;

select * from cust_info;

TRUNCATE table cust_info;

DROP table cust_info;
```

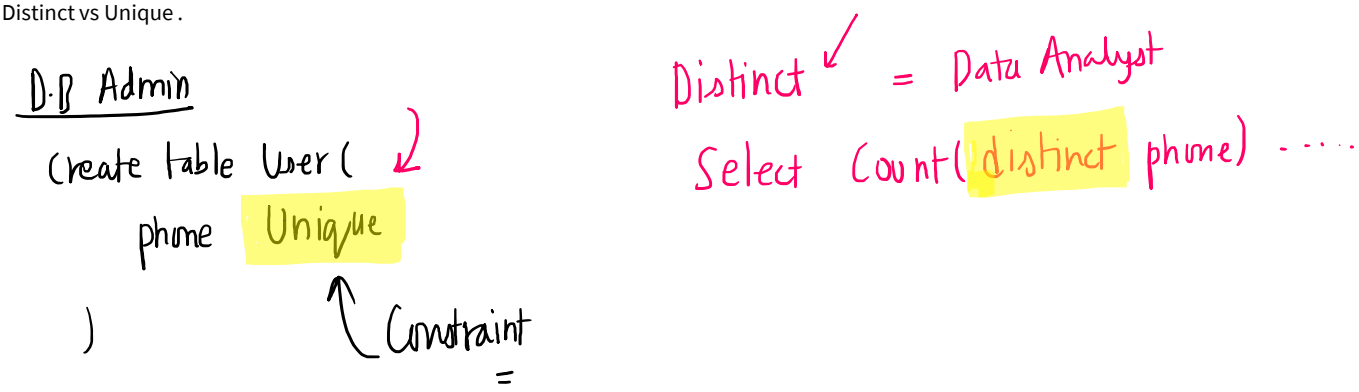
DROP table cust_info;

TRUNCATE	DROP	DELETE
It is a **DDL** command	It is a **DDL** command	It is a **DML** command
Used to delete all the records from a table leaving only the columns.	Used to drop a table or even a database.	Used to delete one or more specific records from a table.
TRUNCATE TABLE table_name;	**DROP** TABLE table_name;	**DELETE** FROM table_name WHERE condition;

Vikas Saini
what is the underlying difference between NOT NULL UNIQUE versus PRIMARY KEY in terms of how the DBMS treat those two differently?



Anjali
Distinct vs Unique .



Session 18 - Miscellaneous Topics

01 October 2024 22:22

LinkedIn : https://www.linkedin.com/in/chauhan-prakash/

DSML Aug24 Beginner 2

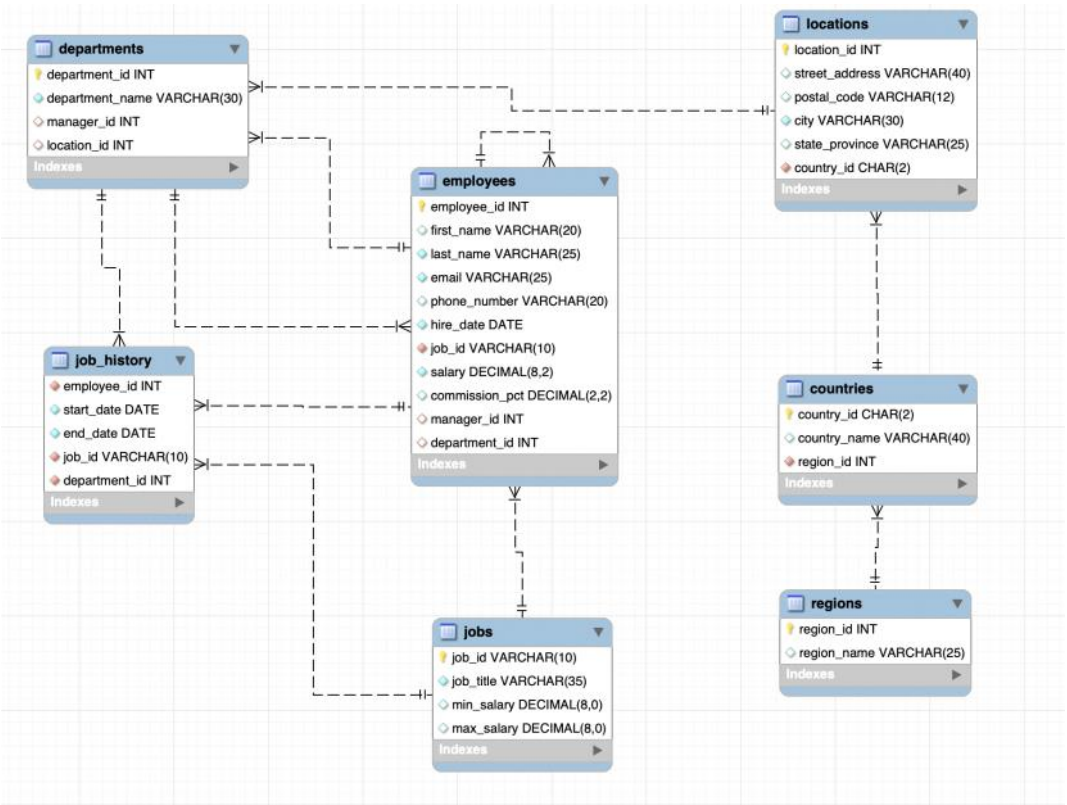
Agenda

1. Problem Statement
2. Correlated Subqueries
3. EXISTS keyword
4. Break & Doubt Resolution
5. ANY and ALL operators
6. Fetch & delete duplicate records
7. GROUP_CONCAT()
8. COALESCE()
9. Impact of the analysis
10. Practice Question

GlobalTech Solutions

GlobalTech Solutions, a leading technology firm with a diverse workforce, has noticed inconsistencies and anomalies in its hu man resources data.
To improve employee satisfaction and streamline HR processes, the HR department aims to conduct a thorough analysis of employ ee data.
This includes identifying salary disparities, commission inconsistencies, managerial structures, and potential data integrity issues.
The primary objective is to gain actionable insights into employee compensation structures, ensure data integrity, and improv e reporting accuracy.

Dataset: LINK



Correlated Subqueries:

Subquery ?

Outer Query (

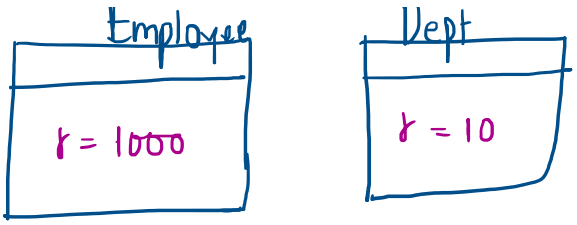
Subquery ?

Outer Query (

Inner Query

)

↑ independent of outer query.



> Outer Query and inner query are correlated ...
> for every iteration of outer query, inner query will also run.

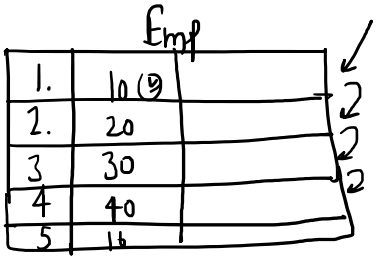
Correlated Subquery =

r = 1000 (Emp)

r = 10 (Dept)

(?)

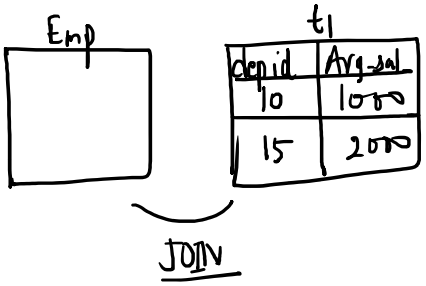
dept	avg_sal
10	1000
20	1500



Q: Retrieve the details of employees whose salary is higher than the average salary of employees in their respective departments.

```
select e.first_name,
       e.last_name,
       e.department_id,
     e.salary
from employees e
where salary > (
    select avg(e1.salary) avg_sal from employees e1
   where e1.department_id = e.department_id
);
```

Using JOIN:



Homework

Solution:

Q: Retrieve the details of employees whose commission is lower than the average commission of employees in the same job category (job_id).

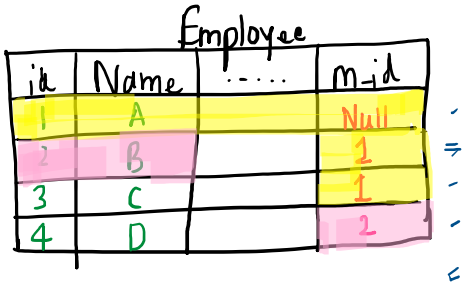
```
select e.first_name,
        e.last_name,
        e.department_id,
      e.commission_pct,
      e.salary
from employees e
where e.commission_pct < (
      select avg(e1.commission_pct) comm_pct from employees e1
   where e1.job_id = e.job_id
);
```

EXISTS keyword:

Q: Retrieve the details of employees who have at least one employee reporting to them.

Columns:

employee_id	int
first_name	text
last_name	text
email	text
phone_number	text
hire_date	text
job_id	text
salary	int
commission_pct	text
manager_id	bigint
department_id	int




```
Select *
from employees e
Where EXISTS (
    Select 1
    from Employees
    where m_id = e.id
)
```

1, A where EXISTS (true)
2, B " (true)
3, C " (false) X

```
select *
from employees e
where EXISTS (
      Select 1 from employees e1
   where e1.manager_id = e.employee_id
);
```

QUIZ:

What does the following SQL query achieve?

Query:

```
SELECT department_name
FROM demo.departments d
WHERE EXISTS (
  SELECT 1
  FROM demo.employees e
```

Employees who are Manager and having salary <= avg(sal) of all emp in their department.

```
WHERE EXISTS (
  SELECT 1
  FROM demo.employees e
  WHERE e.employee_id = d.manager_id
  AND e.salary <= (
    SELECT AVG(salary)
    FROM demo.employees e2
    WHERE e2.department_id = d.department_id
  )
);
```

having salary <= avg(salary) ... in their department.

A
Identifies the departments where the manager is earning less compared to the average earning of an employee from that departm ent.

B
Identifies the departments where the manager's salary is less than or equal to the average salary of employees in that depart ment.

C
Identifies the departments where the average salary of an employee is less than or equal to the manager's salary within that department.

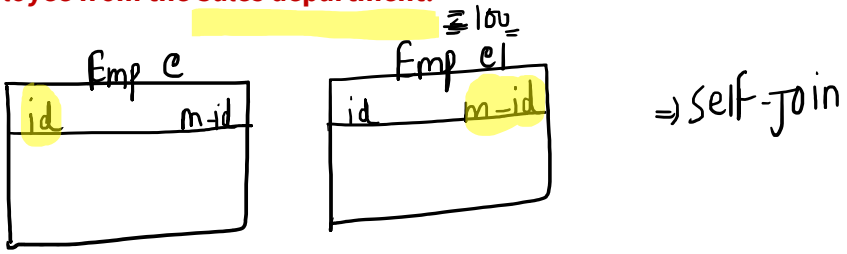
D
Identifies the departments where the manager's salary is greater than or equal to the average salary of employees in that dep artment.

ANY and ALL operators:

These operators are used in combination with comparison operators to compare a value with a set of values returned by a subqu ery.

- The ANY operator returns true if the comparison holds true for **at least one value** in the set.
- The ALL operator returns true if the comparison holds true for **all values** in the set.

Q: Retrieve the details of employees who get a commission equal to the commission percentage of any manager level employee from the Sales department.



Approach 2: ANY

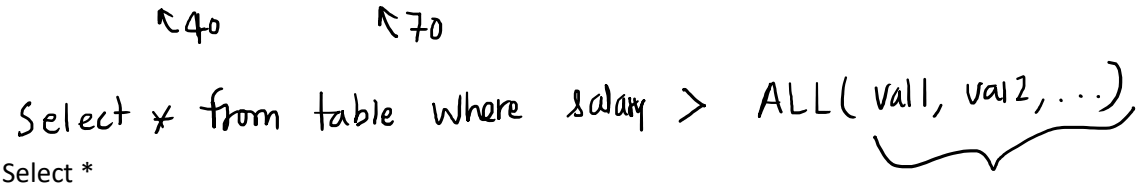
Select * from table1 where col = ANY(val1, val2, ...)

```
Select e.employee_id,
          e.first_name,
      e.last_name
from employees e
where e.commission_pct = ANY (
     select e1.commission_pct
   from employees e1
     where e1.department_id = 80
   and lower(e1.job_id) like "%man%"
```

);

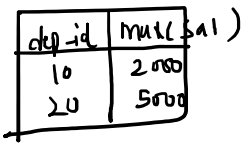
Break Till : 22:27

Q: Retrieve the details of employees who have a higher salary than all employees in both of these departments - Human Resource and Public Relations.




```
Select *
from employees e
where e.salary > ALL (
      select salary
   from employees e1 JOIN departments d
   ON e1.department_id = d.department_id
   where department_name IN ("Human Resource", "Public Relations")
);
```

Approach 2:



Fetching duplicate records

Q: How would you identify if there are any duplicate records present in a table?

```
CREATE TABLE job_hist (
   employee_id INT,
   start_date DATE,
   end_date DATE,
   job_id VARCHAR(20),
   department_id INT
);
```

```
INSERT INTO job_hist (employee_id, start_date, end_date, job_id, department_id) VALUES
(101, '1989-09-21', '1993-10-27', 'AC_ACCOUNT', 110),
(101, '1993-10-28', '1997-03-15', 'AC_ACCOUNT', 110),
(101, '1993-10-28', '1997-03-15', 'AC_ACCOUNT', 110),
(102, '1993-01-13', '1998-07-24', 'IT_PROG', 60),
(114, '1998-03-24', '1999-12-31', 'ST_CLERK', 50),
(122, '1999-01-01', '1999-12-31', 'ST_CLERK', 50),
(122, '1999-01-01', '1999-12-31', 'ST_CLERK', 50),
(122, '1999-01-01', '1999-12-31', 'ST_CLERK', 50),
(122, '1999-01-01', '1999-12-31', 'ST_CLERK', 50),
(122, '1999-01-01', '1999-12-31', 'ST_CLERK', 50),
(176, '1998-03-24', '1998-12-31', 'SA_REP', 80),
(176, '1999-01-01', '1999-12-31', 'SA_MAN', 80),
(200, '1987-09-17', '1993-06-17', 'AD_ASST', 90),
(200, '1987-09-17', '1993-06-17', 'AD_ASST', 90),
(200, '1987-09-17', '1993-06-17', 'AD_ASST', 90),
(200, '1994-07-01', '1998-12-31', 'AC_ACCOUNT', 90),
(201, '1996-02-27', '1999-12-19', 'MK_REP', 20);
```

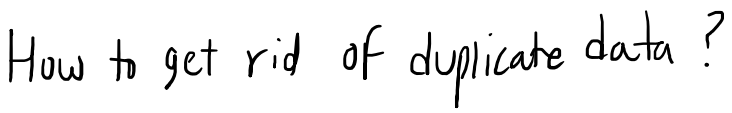
GROUP BY:

```
select employee_id,
		start_date,
	end_date,
		job_id,
	department_id,
	count(*) dup_cnt
from job_hist
group by employee_id,
		start_date,
	end_date,
		job_id,
	department_id
having count(*) >1 ;
```

WINDOWING:

```
select * from (
select employee_id,
		start_date,
	end_date,
		job_id,
	department_id,
	row_number() over (
			partition by employee_id,start_date,end_date,job_id,department_id
	) as row_num
from job_hist) t1
where t1.row_num = 1;
```

distinct



→ DISTINCT ⇒

```
CREATE TABLE new_job_hist AS
SELECT DISTINCT employee_id,
		start_date,
	end_date,
		job_id,
	department_id
FROM job_hist;
```

GROUP_CONCAT()

Input:

order_id	customer_id	product
1	101	Product A
2	102	Product B
3	101	Product C
4	103	Product A
5	101	Product B
6	102	Product D
7	104	Product A
8	101	Product D
9	105	Product C
10	103	Product B

Output:

customer_id	ordered_products
101	Product A, Product C, Product B, Product D
102	Product B, Product D
103	Product A, Product B
104	Product A
105	Product C

Q: Let's say we want to concatenate the first_name of employees within each department_id into a single string separated by commas.

```
select e.department_id,
        group_concat(e.first_name order by e.first_name separator ",") emp_list
from employees e
group by e.department_id;
```

```
select e.department_id,
        group_concat(CONCAT(e.first_name," ",e.last_name ) order by e.first_name separator ",") emp_list
from employees e
group by e.department_id;
```

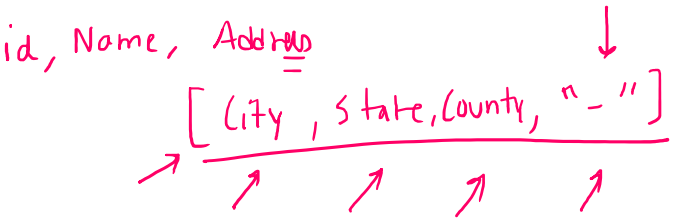
COALESCE()

```
select first_name,
        last_name,
    coalesce(salary,0) as salary
from employees e;
```

Q: Retrieve the names of all employees from our database according to the following guidelines -

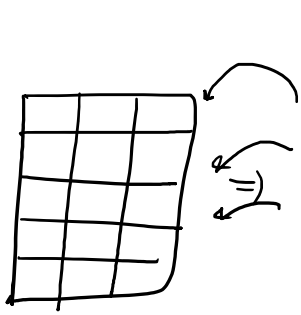
- If the employee's first name is provided, retrieve it.
- If the employee's first name is not available, retrieve their last name.
- If neither the employee's first name nor their last name is specified, set the default value to "UNKNOWN".

```
select coalesce(first_name, last_name, "UNKNOWN") as emp_name
from employees;
```




```
WITH user_platform AS (
SELECT user_id, spend_date,
SUM(CASE WHEN platform = 'mobile' THEN 1 ELSE 0 END) as mobile_use,
SUM(CASE WHEN platform = 'desktop' THEN 1 ELSE 0 END) as desktop_use,
SUM(CASE WHEN platform = 'mobile' THEN amount ELSE 0 END) as mobile_amount,
SUM(CASE WHEN platform = 'desktop' THEN amount ELSE 0 END) as desktop_amount
FROM Spending
GROUP BY user_id, spend_date
),
```

```
SELECT spend_date,
CASE
WHEN mobile_use > 0 AND desktop_use > 0 THEN 'both'
WHEN mobile_use > 0 THEN 'mobile'
ELSE 'desktop'
END as platform,
mobile_amount + desktop_amount as total_amount
FROM user_platform
```



(valesce (first_name, last_name, "NA")

= ≈ ifnull (fist_name, "NA")

QUERIES:

```
use hr;

select e.first_name,
          e.last_name,
          e.department_id,
     e.salary
from employees e
where salary > (
        select avg(e1.salary) avg_sal from employees e1
   where e1.department_id = e.department_id
);

select * from employees limit 1000;

select e.first_name,
          e.last_name,
          e.department_id,
     e.commission_pct,
     e.salary
from employees e
where e.commission_pct < (
        select avg(e1.commission_pct) comm_pct from employees e1
   where e1.job_id = e.job_id
);


select *
from employees e
where EXISTS (
        Select 1 from employees e1
   where e1.manager_id = e.employee_id
);

Select e.employee_id,
               e.first_name,
      e.last_name
from employees e
where e.commission_pct = ANY (
        select e1.commission_pct
   from employees e1
        where e1.department_id = 80
   and lower(e1.job_id) like "%man%"
);

select distinct job_id from employees;



Select *
from employees e
where e.salary > ALL (
        select salary
   from employees e1 JOIN departments d
```

```
   ON e1.department_id = d.department_id
   where department_name IN ("Human Resource", "Public Relations")
);

CREATE TABLE job_hist (
   employee_id INT,
   start_date DATE,
   end_date DATE,
   job_id VARCHAR(20),
   department_id INT
);


INSERT INTO job_hist (employee_id, start_date, end_date, job_id, department_id) VALUES
(101, '1989-09-21', '1993-10-27', 'AC_ACCOUNT', 110),
(101, '1993-10-28', '1997-03-15', 'AC_ACCOUNT', 110),
(101, '1993-10-28', '1997-03-15', 'AC_ACCOUNT', 110),
(102, '1993-01-13', '1998-07-24', 'IT_PROG', 60),
(114, '1998-03-24', '1999-12-31', 'ST_CLERK', 50),
(122, '1999-01-01', '1999-12-31', 'ST_CLERK', 50),
(122, '1999-01-01', '1999-12-31', 'ST_CLERK', 50),
(122, '1999-01-01', '1999-12-31', 'ST_CLERK', 50),
(122, '1999-01-01', '1999-12-31', 'ST_CLERK', 50),
(122, '1999-01-01', '1999-12-31', 'ST_CLERK', 50),
(176, '1998-03-24', '1998-12-31', 'SA_REP', 80),
(176, '1999-01-01', '1999-12-31', 'SA_MAN', 80),
(200, '1987-09-17', '1993-06-17', 'AD_ASST', 90),
(200, '1987-09-17', '1993-06-17', 'AD_ASST', 90),
(200, '1987-09-17', '1993-06-17', 'AD_ASST', 90),
(200, '1994-07-01', '1998-12-31', 'AC_ACCOUNT', 90),
(201, '1996-02-27', '1999-12-19', 'MK_REP', 20);


select * from job_history;

select employee_id,
		start_date,
	end_date,
		job_id,
	department_id,
	count(*) dup_cnt
from job_hist
group by employee_id,
		start_date,
	end_date,
		job_id,
	department_id
having count(*) >1 ;

select * from (
select employee_id,
		start_date,
	end_date,
		job_id,
	department_id,
	row_number() over (
			partition by employee_id,start_date,end_date,job_id,department_id
	) as row_num
from job_hist) t1
where t1.row_num > 1;

CREATE TABLE new_job_hist AS
SELECT DISTINCT employee_id,
		start_date,
	end_date,
		job_id,
	department_id
FROM job_hist;

select * from new_job_hist;

delete from job_hist;

select * from job_hist;

insert into job_hist (select * from new_job_hist);

drop table new_job_hist;

CREATE TABLE new_job_hist AS
select * from (
select employee_id,
		start_date,
```

```
	end_date,
		job_id,
	department_id,
	row_number() over (
			partition by employee_id,start_date,end_date,job_id,department_id
	) as row_num
from job_hist) t1
where t1.row_num = 1;


select e.department_id,
		group_concat(CONCAT(e.first_name," ",e.last_name ) order by e.first_name separator ",") emp_list
from employees e
group by e.department_id;

select first_name,
		last_name,
	coalesce(salary,0) as salary
from employees e;


select coalesce(first_name, last_name, "UNKNOWN") as emp_name
from employees;
```

Session 19 - SQL Advanced Concepts Contd.

06 October 2024 07:56 PM

LinkedIn : https://www.linkedin.com/in/chauhan-prakash/

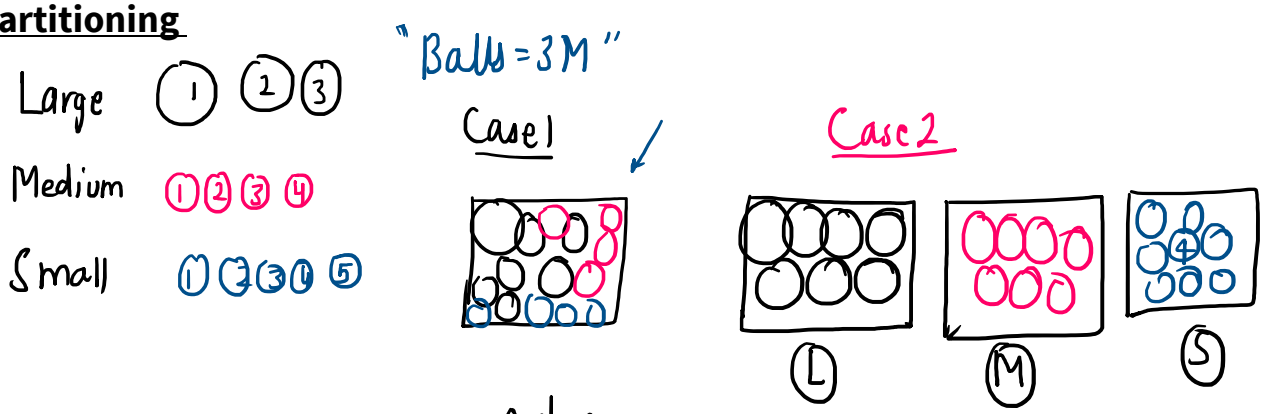
DSML Aug24 Beginner 2

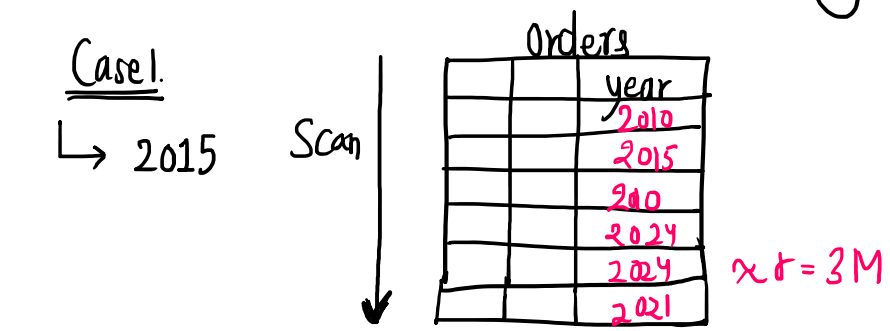
> functions / procedures
> Applied SQL session.

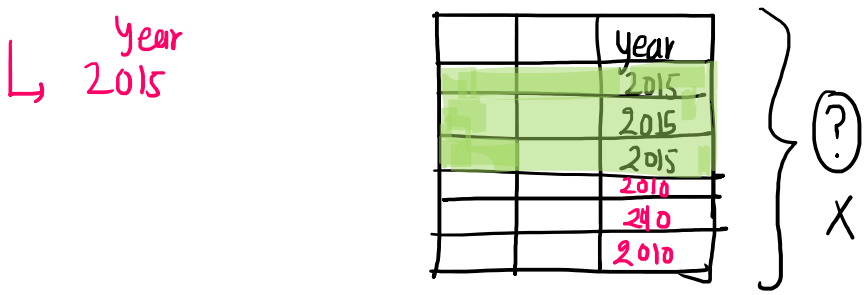
Agenda

1. What is Partitioning?
2. Types of Partitioning
   - RANGE partition
   - LIST partition
   - HASH partition
3. What is an Index?
4. EXPLAIN
5. CREATE & DROP INDEX
6. Break & Doubt Resolution
7. User-defined Functions
8. Stored Procedures
9. Quiz-5
10. Functions vs. Stored Procedures
11. Practice Question

Partitioning

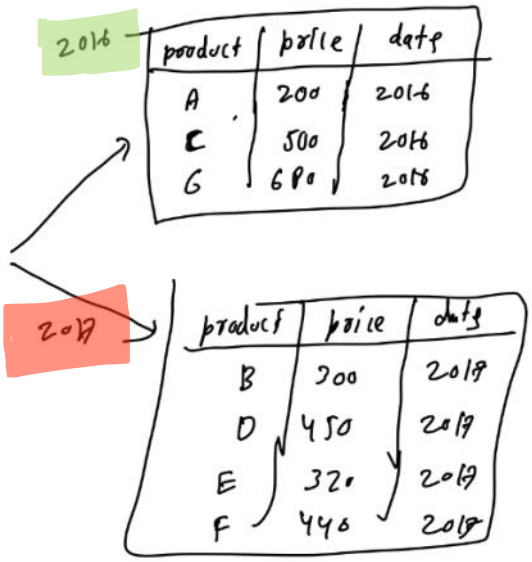






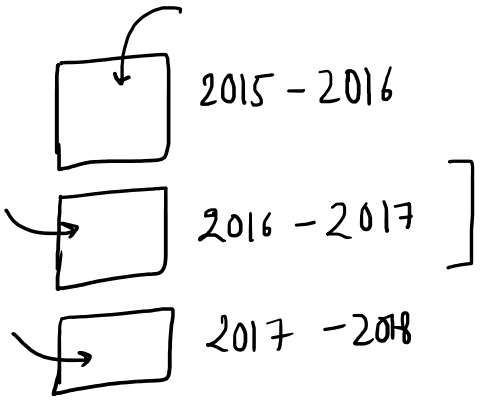
Scenario:

Imagine you're managing a database for Amazon's sales data, and you need to filter out transactions from the year 2016 to calculate the total sales amount.



1. RANGE Partitions

p0	2016
P1	2017
P2	2018
03	2019




```
create database demo_db;

use demo_db;

CREATE TABLE Sales (
    cust_id INT NOT NULL,
    name VARCHAR(40),
    store_id VARCHAR (20) NOT NULL,
    bill_no INT NOT NULL,
```

```
    bill_date DATE PRIMARY KEY NOT NULL,
    amount DECIMAL (8,2) NOT NULL)
partition by range(year(bill_date))(
        partition p1 values less than (2016),
        partition p2 values less than (2017),
    partition p3 values less than (2018),
    partition p4 values less than (2020)
);


 INSERT INTO Sales VALUES
   (1, 'Mike', 'S001', 101, '2015-01-02', 125.56),
   (2, 'Robert', 'S003', 103, '2015-01-25', 476.50),
   (3, 'Peter', 'S012', 122, '2016-02-15', 335.00),
   (4, 'Joseph', 'S345', 121, '2016-03-26', 787.00),
   (5, 'Harry', 'S234', 132, '2017-04-19', 678.00),
   (6, 'Stephen', 'S743', 111, '2017-05-31', 864.00),
   (7, 'Jacson', 'S234', 115, '2018-06-11', 762.00),
   (8, 'Smith', 'S012', 125, '2019-07-24', 300.00),
   (9, 'Adam', 'S456', 119, '2019-08-02', 492.20);

select * from sales;

select table_schema,
    partition_name,
    table_rows,
    avg_row_length,
    data_length
from information_schema.partitions
where table_schema = "demo_db"
and table_name = "sales";

alter table sales truncate partition p1;
```

2. LIST Partitions

4 partitions [0 - 3]

23 % 4 = 3 ✓

24 % 4 = 0 ✓

3 % 4 = 3

4 % 4 = 0

1 % 4 = 1 ✓

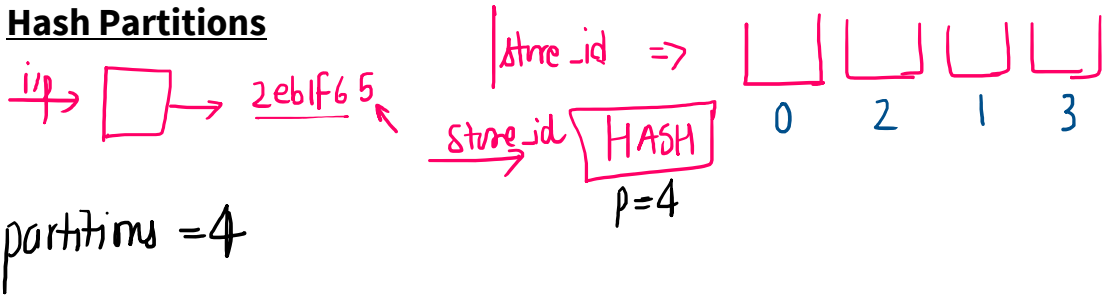
2 % 4 = 2

total_count % 4 =

```
use demo_db;

CREATE TABLE Stores_list (
    cust_name VARCHAR(40),
    bill_no VARCHAR (20) NOT NULL,
    store_id INT PRIMARY KEY NOT NULL,
    bill_date DATE NOT NULL,
    amount DECIMAL(8,2) NOT NULL
)
partition by LIST(store_id)(
     partition p1east values in (101,103,105),
     partition p1west values in (102,104,106) ,
     partition p1north values in (107,109,111) ,
     partition p1south values in (108,110,112)
);


select table_schema,
    partition_name,
    table_rows,
    avg_row_length,
```

```
    data_length
from information_schema.partitions
where table_schema = "demo_db"
and table_name = "Stores_list";
```

3. Hash Partitions




```
use demo_db;

CREATE TABLE Stores_hash (
   cust_name VARCHAR(40),
   bill_no VARCHAR (20) NOT NULL,
   store_id INT PRIMARY KEY NOT NULL,
   bill_date DATE NOT NULL,
   amount DECIMAL(8,2) NOT NULL
)
partition by HASH(store_id)
partitions 4;
```

LIST ✓

QUIZ:

In a list partitioned table of customer data based on their location, you want to add a new partition for customers in a "Central" location and its values.
Which SQL command should you use to add this new partition?

A
```
ALTER TABLE Customers ADD PARTITION P_Central VALUES IN ('Central');
```
Name, List ✓

B
```
ALTER TABLE Customers ADD PARTITION P_Central;
```
X

C
```
CREATE PARTITION P_Central VALUES IN ('Central');
```
X

D
```
CREATE PARTITION P_Central;
```
X

QUIZ

You are managing a large database with millions of records. You frequently need to query sales data by year, and you want to optimize the performance.
Which partitioning method is most suitable for this scenario?

A
List partitioning

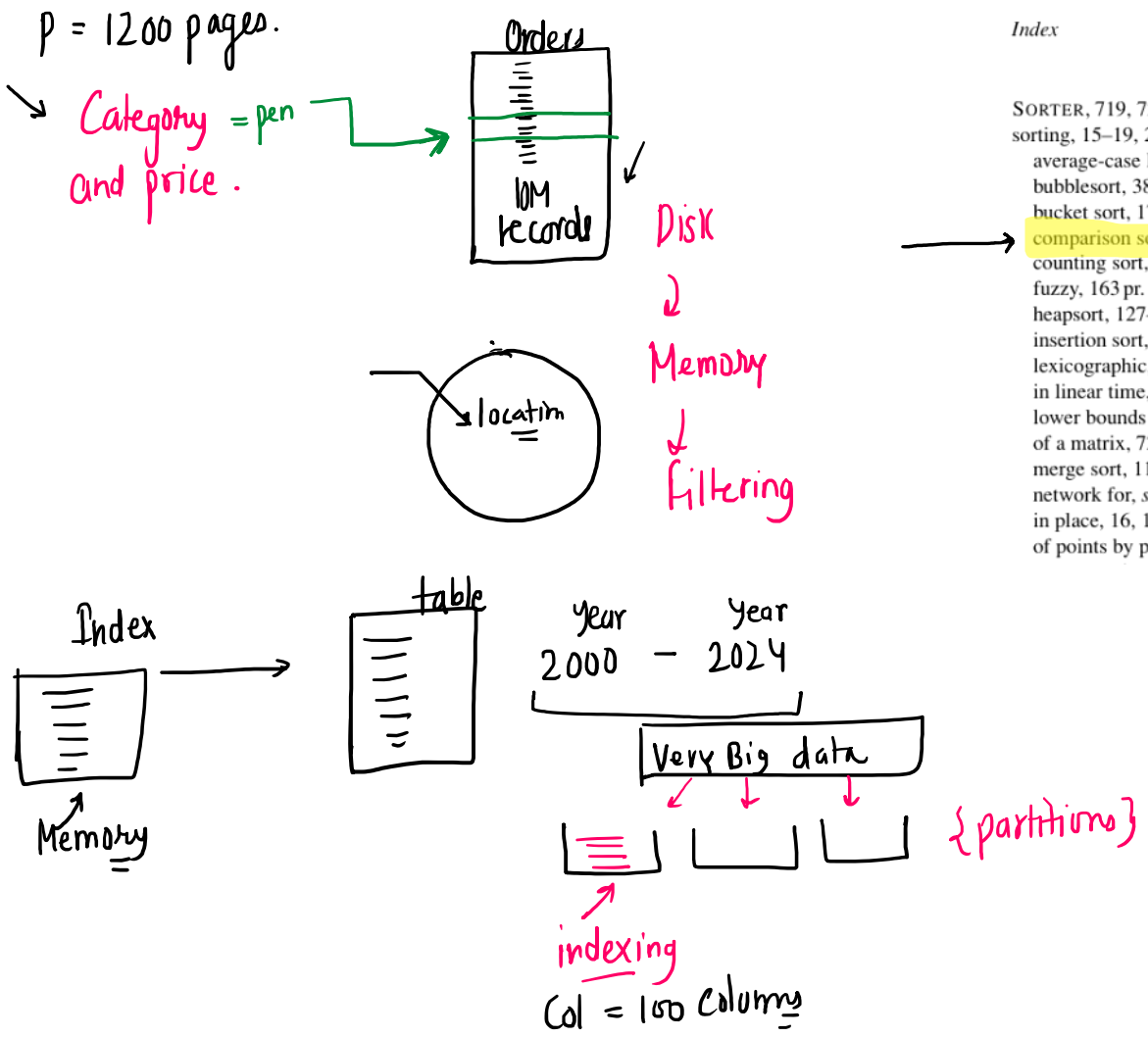
B
Hash partitioning

C
Range partitioning ✓

> → Range (Range)
- between 31-12-2016 and today
- Last 6 months
- Last 2 years.

Break till: 10:30

Indexes




```
use demo_db;

CREATE TABLE names (
    id int(11) unsigned NOT NULL AUTO_INCREMENT,
    first_name varchar(50) CHARACTER SET latin1 DEFAULT NULL,
    last_name varchar(50) CHARACTER SET latin1 DEFAULT NULL,
  PRIMARY KEY (id)
);

INSERT INTO names VALUES (1, 'shivank', 'agrawal');
INSERT INTO names VALUES (2, 'shiva', 'agrawal');
INSERT INTO names VALUES (3, 'shi', 'agrawal');
INSERT INTO names VALUES (4, 'sh', 'agrawal');
INSERT INTO names VALUES (5, 'shivank1', 'agrawal');
INSERT INTO names VALUES (6, 'shivank', 'agrawal');

select * from names;

EXPLAIN select * from names where first_name = 'shivank';

create index first_name_index ON names(first_name);

select * from names where first_name = 'shivank';

drop index first_name_index on names;
```

QUIZ:

You've used the EXPLAIN keyword in MySQL to analyze a query's execution plan. The output indicates that the query is using an index efficiently. What does this mean for query performance?

A
The query will be slower because it's using an index. X

B
The query will be faster because it's using an index. ✓

C
The query's performance is not affected by the index. X

D
The query will be deleted due to the index. X

User-defined Functions

✓ upper ✓ date_add
✓ lower ✓ date_diff
✓ substr ✓ date_sub
✓ parse_date ✓ format_date


```
use demo_db;

-- add_num(num1,num2)
DELIMITER $$
create function add_num1(a INT, b INT) RETURNS INT
deterministic
BEGIN
	RETURN a + b;
END$$
DELIMITER ; -- resetting delimiter to default.


select add_num(1,2) as new_num;



DELIMITER $$
create function double_value(a INT) RETURNS INT
deterministic
BEGIN
	RETURN a * 2;
END$$
DELIMITER ;

select double_value(2345) as double_val;
```

QUIZ:

What is the purpose of creating a user-defined function in SQL, such as the "get_fiscal_year" function?

A
To insert data into a table X

B
To perform complex mathematical calculations ✓

max,]

To insert data into a table X

B
To perform complex mathematical calculations X

C
To encapsulate frequently used SQL logic into a reusable function ✓

D
To delete records from a table X

max,]

Q: Fetch the data for the customer with customer_id=7 and FY 2019? (starting from Oct 2018 to Sept 2019)

year(date_add(market_date, Inkval 3 mnths)) ↵

Stored Procedures

select upper(f_name) from User;

↑ function

QUIZ:

You want to calculate the total sales amount for a specific customer in a given fiscal year and assign them a badge based on their purchase amount.

Which SQL construct allows you to achieve this?

A
View X

B
Stored Procedure ✓

C
Partitions X

D
Indexes X

[i/p = year, c_id = 7] input

[Badge = "Gold", "platinum"]

Queries = ?

sale > 5$ = Gold
else = Silver]

Upper( ) "prakash" i/p Upper → PRAKASH

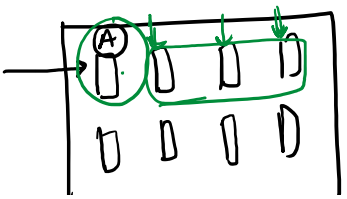
· lower, max, min, avg

Select

Procedure(insert / update / delete)

↳

User (5$) ✓



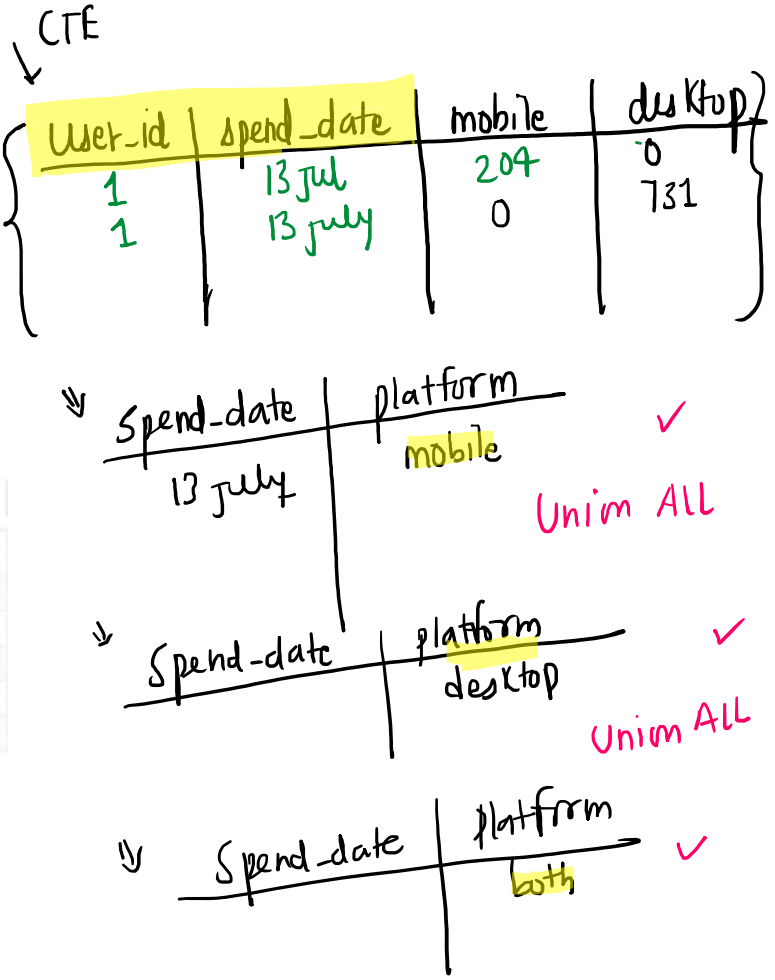
Advanced Constructs - CTEs & Views

Q5:

Input

user_id	spend_date	platform	amount
1	2022/07/13	mobile	204
1	2022/07/13	desktop	731
2	2022/07/13	mobile	431
2	2022/07/14	mobile	346
3	2022/07/13	desktop	686
3	2022/07/14	desktop	74

spend_date	platform	total_amount	total_users
2022/07/13	both	935	1
2022/07/13	desktop	686	1
2022/07/13	mobile	431	1
2022/07/14	both	0	0
2022/07/14	desktop	74	1
2022/07/14	mobile	346	1




```
WITH CTE AS (SELECT user_id, spend_date,
   SUM(CASE platform WHEN 'mobile' THEN amount ELSE 0 END) ma,
   SUM(CASE platform WHEN 'desktop' THEN amount ELSE 0 END) da
FROM Spending
GROUP BY user_id, spend_date)
SELECT spend_date, 'desktop' platform,
   SUM(CASE WHEN da > 0 AND ma = 0 THEN da ELSE 0 END) total_amount,
   SUM(CASE WHEN da > 0 AND ma = 0 THEN 1 ELSE 0 END) total_users
FROM CTE
GROUP BY spend_date
UNION ALL
SELECT spend_date, 'mobile' platform,
        SUM(CASE WHEN ma > 0 AND da = 0 THEN ma ELSE 0 END) total_amount,
        SUM(CASE WHEN ma > 0 AND da = 0 THEN 1 ELSE 0 END) total_users
FROM CTE
GROUP BY spend_date
UNION ALL
SELECT spend_date, 'both' platform,
   SUM(CASE WHEN da > 0 AND ma > 0 THEN ma + da ELSE 0 END) total_amount,
   SUM(CASE WHEN da > 0 AND ma > 0 THEN 1 ELSE 0 END) total_users
FROM CTE
GROUP BY spend_date
```

```
order by spend_date, platform ;
```

Session 20 - Applied SQL 4

09 October 2024 03:23 PM

Last Session today :(

LinkedIn : https://www.linkedin.com/in/chauhan-prakash/

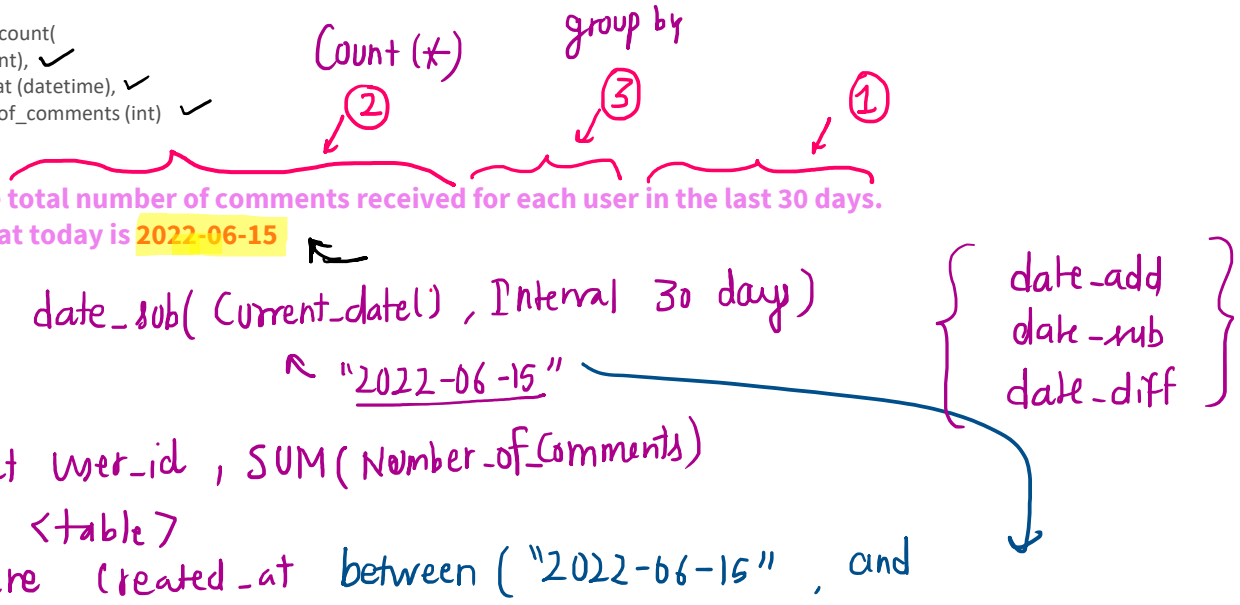
FAANG Questions

Doubts

1. **kiran bodke**
   SQL: Miscellaneous Topics
   Fri, 4 Oct 2024 assignment Q7. Finding the Topic of Each Post

Q1. Given a table:

```
fb_comments_count(
      User_id (int),
      Created_at (datetime),
      Number_of_comments (int)
);
```



Q2. Given two relations/tables:

```
fb_comments_count(
       user_id VARCHAR,
       date DATE, post_id VARCHAR,
       num_comments INT);

 fb_active_users(
       row_no INT,
       user_id VARCHAR,
       Country VARCHAR);
```

Which countries have risen in the rankings based on the number of comments between Dec 2021 vs Jan 2022?

Hint: Avoid gaps between ranks when ranking countries.

fb_comments_count table -

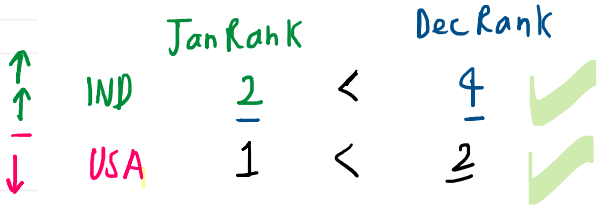
user_id	date	post_id	num_comments
usr1	11/20/2021	p01	10
usr4	11/25/2021	p02	83
usr4	12/1/2021	p03	96
usr5	12/3/2021	p04	40
usr8	12/10/2021	p05	23
usr7	12/19/2021	p06	50
usr6	12/23/2021	p07	35
usr2	12/29/2021	p08	12
usr3	1/5/2022	p09	23
usr8	1/8/2022	p10	18
usr1	1/12/2022	p11	50
usr2	1/16/2022	p12	46
usr3	1/21/2022	p13	6
usr6	1/22/2022	p14	80
usr5	1/25/2022	p15	20
usr6	1/26/2022	p16	23
usr8	1/8/2022	p17	55
usr1	2/5/2022	p18	10
usr5	2/8/2022	p19	12

fb_active_users table -

row_no	user_id	Country
1	usr1	India
2	usr2	India
3	usr3	Australia
4	usr4	Australia
5	usr5	USA
6	usr6	USA
7	usr7	UK
8	usr8	UK

Country	num_comments_dec	country_rank
Australia	96	1
USA	75	2
UK	73	3
India	12	4

	Country	num_comments_jan	country_rank
1	USA	123	1
2	India	96	2
3	UK	73	3
4	Australia	29	4

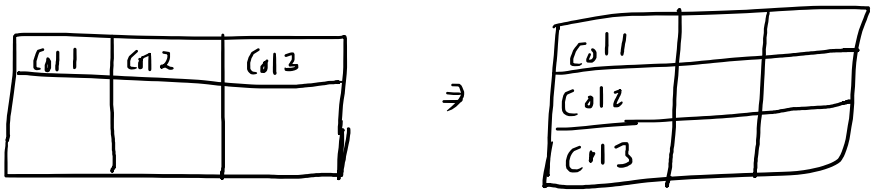


j.country_rank < d.country_rank;

```
WITH dec_summary AS (
  select Country,
        SUM(num_comments) as num_comments_dec,
        dense_rank() OVER(order by SUM(num_comments) DESC) as
country_rank
  from `fb.fb_active_users` a LEFT JOIN `fb.fb_comments_count` b
  ON a.user_id = b.user_id
  where date >= "2021-12-01" and date <= "2021-12-31"
  AND Country IS NOT NULL
  GROUP BY Country
```

```
),
jan_summary AS (
  select Country,
       SUM(num_comments) as num_comments_jan,
       dense_rank() OVER(order by SUM(num_comments) DESC) as
country_rank
  from `fb.fb_active_users` a LEFT JOIN `fb.fb_comments_count` b
  ON a.user_id = b.user_id
  where date >= "2022-01-01" and date <= "2022-01-31"
  AND Country IS NOT NULL
  GROUP BY Country
)
SELECT j.Country from
dec_summary d JOIN jan_summary j
ON j.Country = d.Country
WHERE j.country_rank < d.country_rank;
```

Break Till : 10:14



Problem Description :

Write a query to rearrange the Products table so each row has (product_id, store, price). If a product is unavailable in a store, do not include a row with that product_id and store combination in the result table.

- Order the output by **product_id**, and **store** in ascending order.
- Select MySQL 8.0 in the drop-down.

Sample Input:

Table: Products

product_id	store1	store2	store3
0	95	105	110
1	NULL	87	85
2	NULL	30	40

Sample Output:

product_id	store	price
0	store1	90
0	store2	105
0	store3	110
1	store2	87
1	store3	85
2	store2	30
2	store3	40

Sample Explanation:

- Product 0 is available in all three stores with prices 95, 105, and 110 respectively.
- Product 1 is available in store2 with price 87 and store3 with price 85. The product 1 is not available in store1.
- Product 2 is available in store2 with price 30 and store3 with price 40. The product 2 is not available in store1.

```
Select * from
(Select product_id, "store1" AS store, Store1 AS price
From Products
where Store1 IS NOT NULL

UNION

Select product_id, "store2" AS store, Store2 AS price
From Products
where Store2 IS NOT NULL

UNION

Select product_id, "store3" AS store, Store3 AS price
From Products
```

"Store1"

product_id	store	price
0	store1	90
0	store2	105
0	store3	110

Store → store2 ⇒ store → 105

Union store2

Select product_id, **"store3" AS store**, Store3 AS price
From Products
where Store3 IS NOT NULL) t1

order by product_id, store;

Write a query to find the number of grand slam tournaments won by each player and save the column as 'grand_slams_count'. Do not include the players who did not win any tournament.

- Order the output by player_id in ascending order.
- Select MySQL 8.0 in the drop-down.
- Use the CTE method.

Sample Input:

Table: players

player_id	player_name
1	Andrew
2	Erin
3	Stanley
4	Jim
5	Oscar

Select wimbledon as p_id from championships;

- Each row in this table contains the name and the ID of a tennis player.

Table: championships

year	Wimbledon	Fr_open	US_open	Au_open
2013	1	1	1	2
2014	1	2	4	1
2015	4	2	1	5

- Each row of this table contains the IDs of the players who won one each tennis tournament of the grand slam.

Sample Output:

player_id	player_name	grand_slams_count
1	Andrew	6
2	Erin	3
4	Jim	2
5	Oscar	1

Sample Explanation:

- Player 1 (Andrew) won 6 titles: Wimbledon (2013, 2014), Fr_open (2013), US_open (2013,2015), and Au_open (2014).
- Player 2 (Erin) won 3 titles: Fr_open (2014, 2015), and Au_open (2013).
- Player 3 (Stanley) did not win anything, hence did not include him in the result table.
- Similarly, grand_slams_count is calculated for Players 4 and 5.

Union

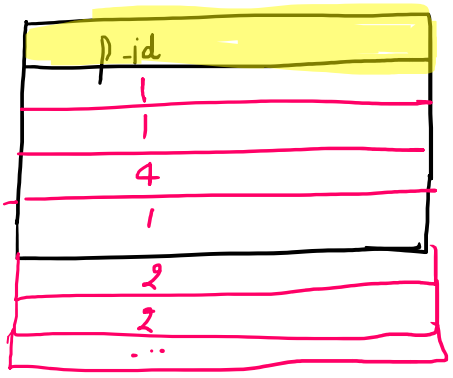
CTE

select wimbledon as p_id from championships;

select Fr_open as p_id from championships;

select US_Open as p_id from championships;

select Au_open as p_id from championships;



count(*)

p_id, p_name, count(*)

```
WITH CTE AS (
  select wimbledon as p_id from championships;
  UNION ALL
  select Fr_open as p_id from championships;
  UNION ALL
  select US_Open as p_id from championships;
  UNION ALL
  select Au_open as p_id from championships;
)
Select player_id, player_name, count(*) as championship_won
From players p JOIN CTE c
ON p.player_id = c.p_id
Group by player_id, player_name
Order by player_id, player_name;
```

kiran bodke

SQL: Miscellaneous Topics

Fri, 4 Oct 2024 assignment Q7. Finding the Topic of Each Post

topic_id	word
1	handball
1	football
3	WAR
2	Vaccine

t_1

$\Rightarrow$ LIKE

post_id	content
1	We call it soccer They call it football hahaha
2	Americans prefer basketball while Europeans lo…
3	stop the war and play handball
4	warning I planted some flowers this morning an…

t_2

```
SELECT P.post_id,
  CASE WHEN COUNT(DISTINCT K.topic_id) = 0
    THEN  'Ambiguous!'
    ELSE GROUP_CONCAT(DISTINCT K.topic_id ORDER BY K.topic_id  SEPARATOR',')
    END AS topic
FROM
  Posts P LEFT JOIN Keywords K
ON
  LOWER(P.content) LIKE CONCAT('% ', LOWER(K.word), ' %') OR
  LOWER(P.content) LIKE CONCAT('% ', LOWER(K.word)) OR
  LOWER(P.content) LIKE CONCAT(LOWER(K.word), ' %')
GROUP BY P.post_id
ORDER BY P.post_id;
```

count(distinct topic_id)

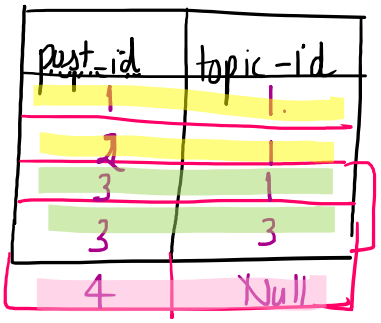
$\Rightarrow 0$, = "Ambiguous!"

group_concat(_ _)

topic_id	word
1	handball
1	football
3	WAR
2	Vaccine

post_id	content
1	We call it soccer They call it football hahaha
2	Americans prefer basketball while Europeans lo…
3	stop the war and play handball
4	warning I planted some flowers this morning an…

$\Rightarrow$



$\Rightarrow$

post_id	topic_id
4	Ambiguous
3	1,3

```
SELECT P.post_id, topic_id
FROM
  Posts P LEFT JOIN Keywords K
```

```
ON
  LOWER(P.content) LIKE CONCAT('% ', LOWER(K.word), ' %') OR
  LOWER(P.content) LIKE CONCAT('% ', LOWER(K.word)) OR
  LOWER(P.content) LIKE CONCAT(LOWER(K.word), ' %');
```

Sanket

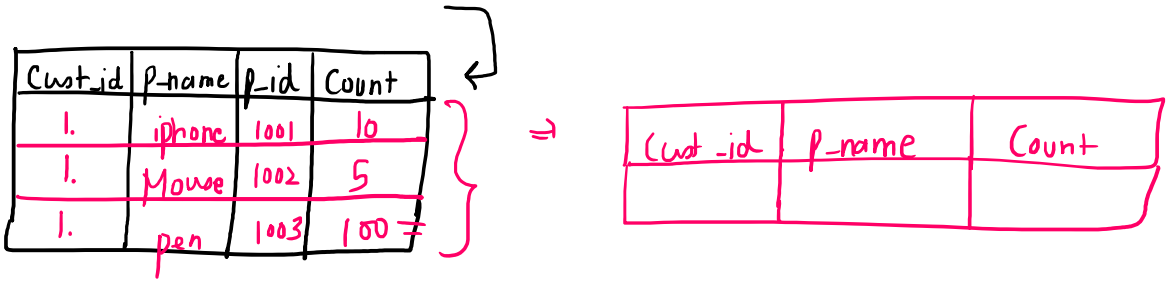
```
select customer_number,
        count()
from orders
group by customer_number
order by count(customer_number) desc
limit 1;
```

1. in this query we haven't use count in coloumn along with customer_number coloumn....
2. if we need to use count we need to collect this query output as a table then in outer query we need to sort customer based on count

```
# Write your MySQL query statement below
select customer_number from (
    select customer_number,
        count(order_number) cnt
    from Orders
    group by customer_number
    order by count(order_number) desc
limit 1 ) t1;
```

Akifa Iram

advanced constructs cte and views Wed, 25 Sep 2024 addtional problem Q1. most frequently bought




```
WITH orders_count AS(
   select o.customer_id,
        p.product_name,
        p.product_id,
        count(o.order_id) as orders_count
   from orders o join products p
   on o.product_id = p.product_id
   group by o.customer_id,
        p.product_name,
        p.product_id
)
select    oc.customer_id,
        oc.product_id,
        oc.product_name
from orders_count oc
where oc.orders_count = (
```

```
    select max(orders_count)
    from orders_count o1
    where o1.customer_id = oc.customer_id
)
order by customer_id, product_id;
```