MySQL

https://www.studocu.com/in/document/marathwada-mitra-mandals-polytechnic/computer-science/leetcodesql-questions-with-solutions/32128690

Master SQL in 16 Pages.pdf

Most Frequently Used Analytics Patterns

1. **Aggregation:** Count things by other things
2. **Experimentation:** Splitting people into groups and doing mind manipulation on them
3. **Prediction:** Whats happenening and whats gonna happen
4. **Clustering:** How do we put into groups
5. **Decision Trees:** what are the decisions and points we have made to get into this point
6. **Cumulative/Derivative:** Rolling sums, monthly totals, Year over Year growth
7. **Funnel Analysis:** From top to bottom Funnel how much users have gone through

C1_oct7_2023:

1.) What is Database?

2.) Difference between Transactional Databases and NoSQL databases

3.) What is RDBMS?

4.) Setup MySQL Workbench

5.) Setup MySQL Using Docker

6.) DDL, DML, DQL, DCL

7.) CREATE Command

8.) INSERT Command

9.) Integrity Constraints

SQL_Class_1_Rough_Notes.pdf

SQL_Class_1_PPT.pdf

mysql_docker_compose.yml

SQL

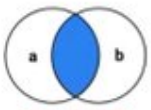
CHEATSHEET

By @AbzAaron
https://twitter.com/AbzAaron

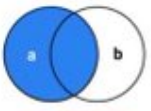
Common Commands

SELECT	Select data from database
AS	Rename column or table with alias
FROM	Specify table we're pulling from
WHERE	Filter query to match a condition
JOIN	Combine rows from 2 or more tables
AND	Combine conditions in a query. All must be met
OR	Combine conditions in a query. One must be met
LIKE	Search for patterns in a column
IN	Specify multiple values when using WHERE
IS NULL	Return only rows with a NULL value
LIMIT	Limit the number of rows returned
CASE	Return value on a specified condition
CREATE	Create TABLE, DATABASE, INDEX or VIEW
DROP	Delete TABLE, DATABASE, or INDEX
UPDATE	Update table data
DELETE	Delete rows from a table
ALTER TABLE	Add/Remove columns from table
GROUP BY	Group rows that have same values into summary rows
ORDER BY	Set order of result. Use DESC to reverse order
HAVING	Same as WHERE but used for aggregate functions
SUM	Return sum of column
AVG	Return average of column
MIN	Return min value of column
MAX	Return max value of column
COUNT	Count number of rows

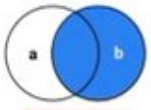
Joins



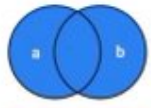
a INNER JOIN b



a LEFT JOIN b



a RIGHT JOIN b



a FULL OUTER JOIN b

Examples

Select all rows from table with filter applied

```
SELECT * FROM tbl WHERE col1 > 5;
```

Select first 10 rows for 2 columns

```
SELECT col1, col2 FROM tbl LIMIT 10;
```

Select all rows with multiple filters applied

```
SELECT * FROM tbl WHERE col1 > 5 AND col2 < 2;
```

Select all rows from col1 and col2 ordering by col1

```
SELECT col1, col2 FROM tbl ORDER BY 1;
```

Return count of rows in table

```
SELECT COUNT(*) FROM tbl;
```

Return sum of col1

```
SELECT SUM(col1) FROM tbl;
```

Return max value from col1

```
SELECT MAX(col1) FROM tbl;
```

Computer summary statistics by grouping col2

```
SELECT AVG(col1) FROM tbl GROUP BY col2;
```

Combine data from two tables using a left join

```
SELECT * FROM tbl1 AS t1
LEFT JOIN tbl2 AS t2 ON t2.col1 = t1.col1;
```

Aggregate and filter results

```
SELECT
    col1,
    AVG(col2) * AVG(col3) AS total
FROM tbl
GROUP BY col1
HAVING total > 2
```

Implementation of CASE statement

```
SELECT col1,
CASE
  WHEN col1 > 10 THEN "more than 10"
  WHEN col1 < 10 THEN "less than 10"
  ELSE "10"
END AS NewColumnName
FROM tbl;
```

Create

```
CREATE DATABASE MyDatabase;
```

```
CREATE INDEX IndexName
ON TableName(col1);
```

```
CREATE TABLE OurTable (
    id int,
    name varchar(12)
);
```

Update Table

```
UPDATE OurTable
SET col1 = 56
WHERE col2 = 'something';
```

Delete Records

```
DELETE FROM OurTable
WHERE col1 = 'something';
```

Delete

```
DROP DATABASE OurDatabase;
```

```
DROP TABLE OurTable;
```

Add/Remove Column

```
ALTER TABLE OurTable
ADD col5 int;
```

```
ALTER TABLE OurTable
DROP COLUMN col5;
```

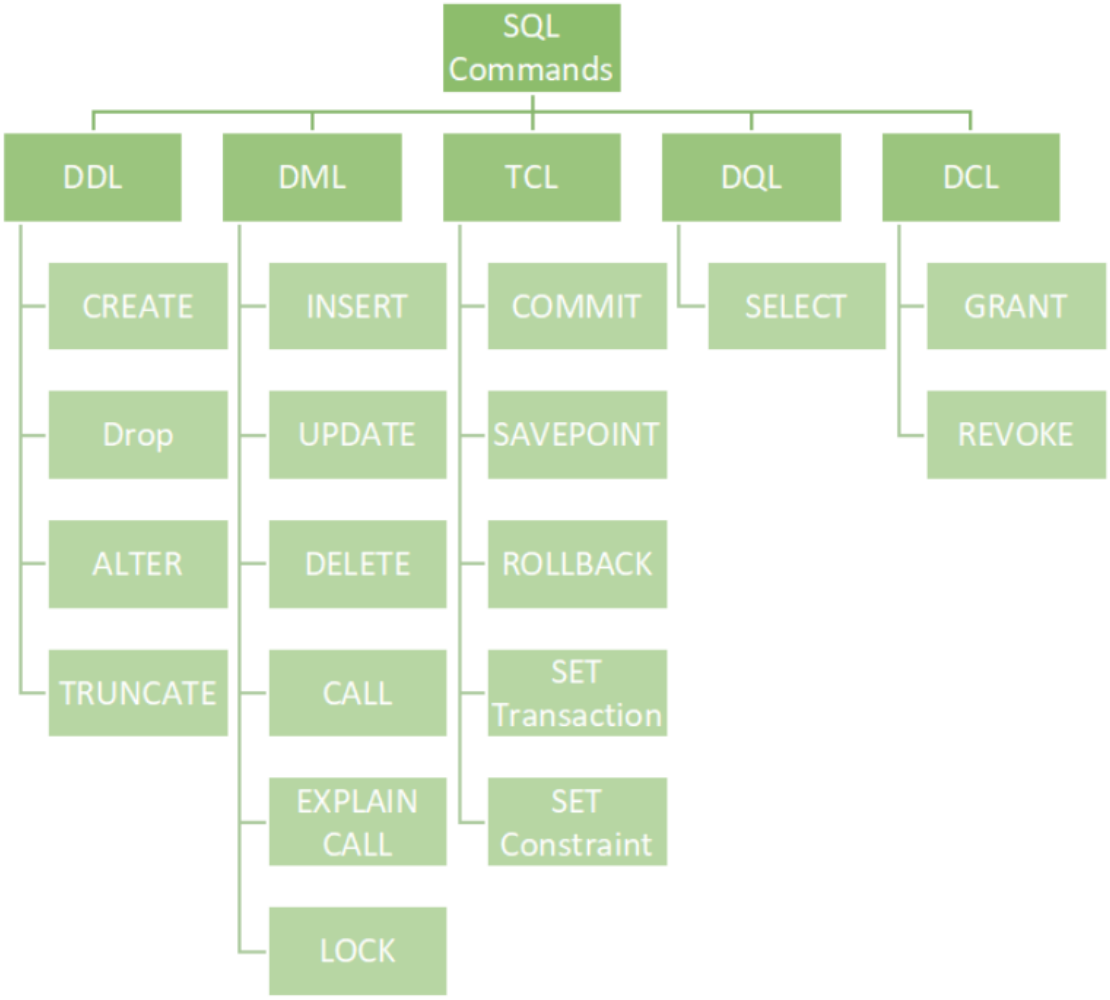
Order of Execution

FROM
WHERE
GROUP BY
HAVING
SELECT
ORDER BY
LIMIT

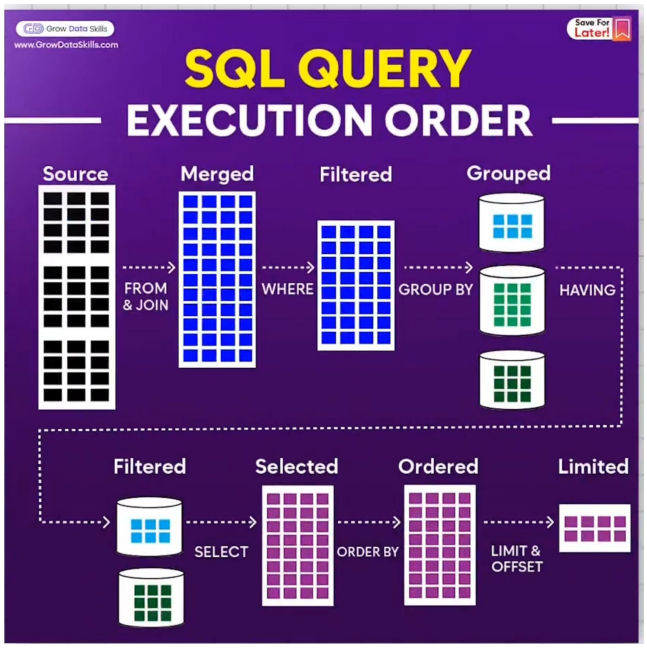
Sources:

https://www.dataquest.io/blog/sql-commands/#1-1613505022898

https://www.dataquest.io/blog/sql-joins-tutorial/



- DDL - **Data Definition Language**
- DML - **Data Modification Language**
- DQL - **Data Query Language**
- DCL - **Data Control Language**
- TCL - **Transaction Control Languages**


Grow Data Skills
www.GrowDataSkills.com
Save For Later!
SQL QUERY
EXECUTION ORDER
Source
Merged
Filtered
Grouped
FROM & JOIN
WHERE
GROUP BY
HAVING
Filtered
Selected
Ordered
Limited
SELECT
ORDER BY
LIMIT & OFFSET

```sql
SELECT
    d.department_name,
    SUM(e.salary) AS total_salary
FROM
    employees e
INNER JOIN
    departments d ON e.department_id = d.department_id
WHERE
    e.salary > 30000
GROUP BY
    d.department_name
HAVING
    SUM(e.salary) > 100000
ORDER BY
    total_salary DESC
LIMIT 5;
```

```sql
--- Command to see the list of databases
Show databases;

--- Command Create database
create database noob_db;

--- Command to delete database
drop database first_demo;

--- Command to use database for specific data queries inside it
```

```
use noob_db;

--- Command to create a table
CREATE TABLE if not EXISTS employee
(
    id INT,
  emp_name VARCHAR(20)
);

--- Command to see the list of tables
show tables;

--- Command to see the table definition
show create table employee;

--- Create employee table with few more columns
CREATE TABLE if not EXISTS employee_v1
(
    id INT,
  name VARCHAR(50),
  salary DOUBLE,
  hiring_date DATE
);

--- Syntax 1 to insert data into a table
insert into employee_v1 values(1,'Shashank',1000,'2021-09-15');

--- This statement will fail cuz salary element is missing
insert into employee_v1 values(1,'Shashank','2021-09-15');

--- Syntax 2 to insert data into a table
insert into employee_v1(salary,name,id)
values(2000,'Rahul',2);

--- Command to insert multiple records into a table
insert into employee_v1 values(3,'Amit',5000,'2021-10-28'),
```

```
(4,'Nitin',3500,'2021-09-16'),
(5,'Kajal',4000,'2021-09-20');

--- How to query or fetch the data from a table
select * from employee_v1;


--- Example table for integrity constraints
CREATE TABLE if not EXISTS employee_with_constraints
(
    id INT,
  name VARCHAR(50) NOT NULL,
  salary DOUBLE,
  hiring_date DATE DEFAULT '2021-01-01',
  UNIQUE (id),
  CHECK (salary > 1000)
);

--- Example 1 for IC failure
--- Exception - Column 'name' cannot be null
insert into employee_with_constraints values(1,null,3000,'2021-1

--- Correct record
insert into employee_with_constraints values(1,'Shashank',3000,

--- Example 2 for IC failure
--- Exception - Duplicate entry '1' for key 'employee_with_const
insert into employee_with_constraints values(1,'Rahul',5000,'202

--- Another correct record because Unique can accept NULL as wel
insert into employee_with_constraints
values(null,'Rahul',5000,'2021-10-23');

--- Example 3 for IC failure
--- Exception - Duplicate entry NULL for key 'employee_with_cons
insert into employee_with_constraints
```

```
values(null,'Rajat',2000,'2020-09-20');


--- Example 4 for IC failure
--- Exception - Check constraint 'employee_with_constraints_chk_
insert into employee_with_constraints
values(5,'Amit',500,'2023-10-24');

--- Test IC for default date
insert into employee_with_constraints(id,name,salary)
values(7,'Neeraj',3000);

select * from employee_with_constraints;


--- Example table for integrity constraints
CREATE TABLE if not EXISTS employee_with_constraints_tmp
(
    id INT,
  name VARCHAR(50) NOT NULL,
  salary DOUBLE,
  hiring_date DATE DEFAULT '2021-01-01',
  Constraint unique_emp_id UNIQUE (id),
  Constraint salary_check CHECK (salary > 1000)
);

--- Check the name of constraint when it fails
--- Exception - Check constraint 'salary_check' is violated
insert into employee_with_constraints_tmp
values(5,'Amit',500,'2023-10-24');
```

C2_oct8_2023:

1.) INTEGRITY Constraints with custom name

2.) Primary Key vs Foreign Key

3.) ALTER Command

4.) UPDATE Command

5.) WHERE Clause

6.) Conditional Updates

7.) Conditional and Logical Operators

8.) SELECT Command Operations

9.) LIKE Operation

SQL_Class_2_Rough_Notes.pdf

```
Create database class2_db;
use class2_db;

create table if not exists employee(
    id int,
    name VARCHAR(50),
    address VARCHAR(50),
    city VARCHAR(50)
);

insert into employee values(1, 'Shashank', 'RJPM', 'Lucknow');

select * from employee;

--- add new column named DOB in the TABLE
alter table employee add DOB date;

select * from employee;
```

```
----- DML,  DELETE == remove specific row based on condition
----- DDL,  TRUNCATE == remove data without effecting the struct
----- DDL,  DROP  == remove entire data & structure of the table


--- modify existing column in a TABLE or change datatype of name
alter table employee modify column name varchar(100);

--- delete existing column from given TABLE or remove city colum
alter table employee drop column city;

select * from employee;


--- rename the column name to full_name
alter table employee rename column name to full_name;

drop table employee;

create table if not exists employee(
    id int,
    name VARCHAR(50),
    age int,
    hiring_date date,
    salary int,
    city varchar(50)
);
insert into employee values(1,'Shashank', 24, '2021-08-10', 1000
(2,'Rahul', 25, '2021-08-10', 20000, 'Khajuraho'),
(3,'Sunny', 22, '2021-08-11', 11000, 'Banaglore'),
(5,'Amit', 25, '2021-08-11', 12000, 'Noida'),
(6,'Puneet', 26, '2021-08-12', 50000, 'Gurgaon');


--- add unique integrity constraint on id COLUMN
alter table employee add constraint id_unique UNIQUE(id);
```

```
--- the below query doen't add data cuz of adding unique constra
insert into employee values(1,'XYZ', 25, '2021-08-10', 50000, 'C

--- drop constraint from existing TABLE
alter table employee drop constraint id_unique;

insert into employee values(1,'XYZ', 25, '2021-08-10', 50000, 'C

--- create table with Primary_Key
Create table persons
(
    id int,
    name varchar(50),
    age int,
    ---Primary Key (id)
    constraint pk Primary Key (id)
);

insert into persons values(1,'Shashank',29);

-- Try to insert duplicate value for primary key COLUMN
-- it says unique constraint fails cuz primary key is always uni
insert into persons values(1,'Rahul',28);

-- Try to insert null value for primary key COLUMN
-- primary key doesn't allow null values but unique key allows n
insert into persons values(null,'Rahul',28);

--- To check difference between Primary Key and Unique
alter table persons add constraint age_unq UNIQUE(age);

-- shows structure of table
show create table persons
```

```
select * from persons;
insert into persons values(2,'Rahul',28),(3,'Amit',28),(3,'Amit

select * from persons;


-- unique key treats multiple null key as unique
insert into persons values(4,'Charan',null);
insert into persons values(5,'Deepak',null);


--- create tables for Foreign Key demo
create table customer
(
    cust_id int,
    name VARCHAR(50),
    age int,
    constraint pk Primary Key (cust_id)
);

create table orders
(
    order_id int,
    order_num int,
    customer_id int,
    constraint pk Primary Key (order_id),
    constraint fk Foreign Key (customer_id) REFERENCES customer(
);

insert into customer values(1,"Shashank",29),(2,"Rahul",30);

select * from customer;

insert into orders values(1001, 20, 1),(1002, 30, 2);

select * from orders;
```

```
--- It will not allow to insert because referncial integrity wi
insert into orders values(1004, 35, 5);

select * from persons;
truncate table persons;

select * from persons;
drop table persons; /* deletes entire persons table */


--- Operations with Select Command

select * from employee;
drop table employee; /* deletes entire employee table */

create table if not exists employee(
    id int,
    name VARCHAR(50),
    age int,
    hiring_date date,
    salary int,
    city varchar(50)
);

insert into employee values(1,'Shashank', 24, '2021-08-10', 100
insert into employee values(2,'Rahul', 25, '2021-08-10', 20000,
insert into employee values(3,'Sunny', 22, '2021-08-11', 11000,
insert into employee values(5,'Amit', 25, '2021-08-11', 12000,
insert into employee values(1,'Puneet', 26, '2021-08-12', 50000,

select * from employee;

--- how to count total records
select count(*) from employee;
```

```
--- alias declaration
select count(*) as total_row_count from employee;


--- display all columns in the final result
select * from employee;

--- display specific columns in the final result
select name, salary from employee;

--- aliases for mutiple columns
select name as employee_name, salary as employee_salary from emp

select * from employee;

--- print unique hiring_dates from the employee table when emplo
select Distinct(hiring_date) as distinct_hiring_dates from emplo


select * from employee;

--- How many unique age values in the table??

select count(distinct(age)) as total_unique_ages from employee;

--- Increment salary of each employee by 20% and display final r
SELECT  id,
        name,
        salary as old_salary,
        (salary + salary * 0.2) as new_salary
FROM employee;


-- Syntax for update command
UPDATE table_name SET column_name = value  -- (for entire columm
```

```
UPDATE table_name SET column_name = value where conditon

--- Upadtes will be made for all rows
UPDATE employee SET age = 20;

select * from employee;

--- update the salary of employee after giving 20% increment
UPDATE employee SET salary = salary + salary * 0.2;

select * from employee;

--- How to filter data using WHERE Clauses
select * from employee where hiring_date = '2021-08-10';

select * from employee;

--- Update the salary of employees who joined the company on 202
update employee SET salary = 80000 where hiring_date = '2021-08-

select * from employee;


--- how to delete specific records from table using delete comma
--- delete records of those employess who joined company on 2021

delete from employee where hiring_date = '2021-08-10';

select * from employee;

--- How to apply auto increment
create table auto_inc_exmp
(
  id int auto_increment,
  name varchar(20),
```

```
  primary key (id)
);

insert into auto_inc_exmp(name) values('Shashank');
insert into auto_inc_exmp(name) values('Rahul');

select * from auto_inc_exmp;

--- Use of limit
--- limits no.of rows to display

select * from employee;
select * from employee limit 2;

-- sorting data in mysql by using 'Order By'
select * from employee order by name;

# arrage data in ascending order
select * from employee order by name asc;


# arrage data in descending order
select * from employee order by name desc;

# display employee data in desc order of salary and if salaries
# one employees arrange their data in ascedinding order of name

select * from employee order by salary desc, name asc;

# when we ignore multilevel ordering
select * from employee order by salary desc;

# Write a query to find the employee who is getting maximum sala
select * from employee order by salary desc limit 1;
```

```
# Write a query to find the employee who is getting minium salar
select * from employee order by salary limit 1;


# Write a query to find the employee who is getting minium salar
select * from employee order by salary limit 1;



/* Conditional Operators ->    < , > , <= , >=
Logical Operator -> AND, OR, NOT */

select * from employee;

# list all employees who are getting salary more than 20000
select * from employee where salary>20000;

# list all employees who are getting salary more than or equal t
select * from employee where salary>=20000;

# list all employees who are getting less than 20000
select * from employee where salary<20000;

# list all employees who are getting salary less than or equal t
select * from employee where salary<=20000;


# filter the record where age of employees is equal to 20
select * from employee where age=20;

# filter the record where age of employees is not equal to 20
# we can use != or we can use <>
select * from employee where age != 20;
select * from employee where age <> 20;

# find those employees who joined the company on 2021-08-11 and
```

```
select * from employee where hiring_date = '2021-08-11' and sala

# find those employees who joined the company after 2021-08-11 c
select * from employee where hiring_date > '2021-08-11' or salar

# how to use Between operation in where clause
# get all employees data who joined the company between hiring_c
select * from employee where hiring_date between '2021-08-05' ar


# get all employees data who are getting salary in the range of
select * from employee where salary between 10000 and 28000;


# how to use LIKE operation in where clause
# % -> Zero, one or more than one characters
# _ -> only one character


# get all those employees whose name starts with 'S'
select * from employee where name like 'S%';


# get all those employees whose name starts with 'Sh'
select * from employee where name like 'Sh%';


# get all those employees whose name ends with 'l'
select * from employee where name like '%l';


# get all those employees whose name starts with 'S' and ends wi
select * from employee where name like 'S%k';

# Get all those employees whose name will have exact 5 character
select * from employee where name like '_____';
```

```
# Return all those employees whose name contains atleast 5 chara
select * from employee where name like '%_____';
select * from employee where name like '_____%';
select * from employee where name like '%_____%';
```

C3_oct14_2023:

1.) IS NULL, IS NOT NULL

2.) Group By, Having Clause

3.) Group Concat, Group RollUP

4.) Sub Queries, IN and NOT IN

5.) CASE-When

6.) SQL Joins

7.) Views

SQL_Class_3_PDF_Notes.pdf

SQL_Class_3_Rough_Notes.pdf

```
# How to use IS NULL or IS NOT NULL in the where clause
insert into employee values(10,'Kapil', null, '2021-08-10', 100
insert into employee values(11,'Nikhil', 30, '2021-08-10', null,

select * from employee;

# get all those employees whos age value is null
```

```
select * from employee where age is null;


# get all those employees whos salary value is not null
select * from employee where salary is not null;


# Table and Data for Group By
create table orders_data
(
 cust_id int,
 order_id int,
 country varchar(50),
 state varchar(50)
);


insert into orders_data values(1,100,'USA','Seattle');
insert into orders_data values(2,101,'INDIA','UP');
insert into orders_data values(2,103,'INDIA','Bihar');
insert into orders_data values(4,108,'USA','WDC');
insert into orders_data values(5,109,'UK','London');
insert into orders_data values(4,110,'USA','WDC');
insert into orders_data values(3,120,'INDIA','AP');
insert into orders_data values(2,121,'INDIA','Goa');
insert into orders_data values(1,131,'USA','Seattle');
insert into orders_data values(6,142,'USA','Seattle');
insert into orders_data values(7,150,'USA','Seattle');

select * from orders_data;

# calculate total order placed country wise
select country, count(*) as order_count_by_each_country from or

# Write a query to find the total salary by each age group
select * from employee;
```

```
select age, sum(salary) as total_salary_by_each_age_group from e


# calculate different aggregated metrices for salary
select age,
       sum(salary) as total_salary_by_each_age_group,
     max(salary) as max_salary_by_each_age_group,
     min(salary) as min_salary_by_each_age_group,
     avg(salary) as avg_salary_by_each_age_group,
     count(*) as total_employees_by_each_age_group
from employee group by age;

# Group by on multiple columns
select
   country,
   state,
   count(*) as state_wise_order
from orders_data
group by country, state;


# Use of Having Clause
# Write a query to find the country where only 1 order was place
select country from orders_data group by country having count(*)



# Where Clause and Group By Clause --> What should be the proper
# Answer -> Where Clause and then Group By


/* How to use GROUP_CONCAT
 Query - Write a query to print distinct states present in the c
 GROUP_CONCAT concats the column values into one specific coulur
select country, GROUP_CONCAT(state) as states_in_country from or
```

```
select country, GROUP_CONCAT(distinct state) as states_in_countr

select country, GROUP_CONCAT(distinct state order by state desc)

select country, GROUP_CONCAT(distinct state order by state desc



## Subqueries in SQL
create table employees
(
    id int,
    name varchar(50),
    salary int
);
insert into employees values(1,'Shashank',5000),(2,'Amit',5500),
(4,'Rohit',6000),(5,'Nitin',4000),(6,'Sunny',7500);

select * from employees;

#Write a query to print all those employee records who are getti
# Wrong solution -> select * from employees where salary > 6000;
select * from employees where salary > (select salary from emplo



# Use of IN and NOT IN
# Write a query to print all orders which were placed in 'Seattl
select * from orders_data;

SELECT * FROM orders_data WHERE state in ('Seattle', 'Goa');

create table customer_order_data
(
    order_id int,
    cust_id int,
    supplier_id int,
```

```
    cust_country varchar(50)
);


insert into customer_order_data values(101,200,300,'USA'),(102,2

create table supplier_data
(
    supplier_id int,
    sup_country varchar(50)
);
insert into supplier_data values(300,'USA'),(303,'UK');


# write a query to find all customer order data where all cousto
# as the suppliers
select * from customer_order_data where cust_country in
(select distinct sup_country from supplier_data);

# Another example of Sub-Query
select *
from (select
       country,
       count(*) as country_wise_order
    from orders_data
    group by country) result
where country_wise_order=1;

# Uber SQL Interview questions
create table tree
(
    node int,
    parent int
);

insert into tree values (5,8),(9,8),(4,5),(2,9),(1,5),(3,9),(8,n
```

```
select * from tree;
| NODE  | PARENT |
| --- | --- |
| 5 | 8 |
| 9 | 8 |
| 4 | 5 |
| 2 | 9 |
| 1 | 5 |
| 3 | 9 |
| 8 | NULL |
/* parent lo nodes levu ante avi last lo vache leaves ani ardam
node ki parent a ledu ante adi root node ani ardam
migilnavi inner nodes ani
*/
select node,
       CASE
            when node not in (select distinct parent from tree w
            when parent is null then 'ROOT'
            else 'INNER'
       END as node_type
from tree;


----------------------- JOINS --------------------------------

# Examples for join
create table orders
(
    order_id int,
    cust_id int,
    order_dat date,
    shipper_id int
);
create table customers
(
```

```
    cust_id int,
    cust_name varchar(50),
    country varchar(50)
);
create table shippers
(
    ship_id int,
    shipper_name varchar(50)
);

insert into orders values(10308, 2, '2022-09-15', 3);
insert into orders values(10309, 30, '2022-09-16', 1);
insert into orders values(10310, 41, '2022-09-19', 2);

insert into customers values(1, 'Neel', 'India');
insert into customers values(2, 'Nitin', 'USA');
insert into customers values(3, 'Mukesh', 'UK');

insert into shippers values(3,'abc');
insert into shippers values(1,'xyz');

select * from orders;
select * from customers;
select * from shippers;


# perform inner JOIN (2 tables nunchi common values testundi)
# get the customer informations for each order order, if value c
select
o.*, c.*
from orders o
inner join customers c on o.cust_id = c.cust_id;


/* Left Join (left table complete data + left table tho overlap
kuda testundi) */
```

```
select
o.*, c.*
from orders o
left join customers c on o.cust_id = c.cust_id;


/* Right Join (right table complete data + right table tho overl
kuda testundi) */
select
o.*, c.*
from orders o
right join customers c on o.cust_id = c.cust_id;



# How to join more than 2 datasets?
# perform inner JOIN
# get the customer informations for each order order, if value c
# also get the information of shipper name

select
o.*, c.*, s.*
from orders o
inner join customers c on o.cust_id = c.cust_id
inner join shippers s on o.shipper_id = s.ship_id;

create table employees_full_data
(
    emp_id int,
    name varchar(50),
    mgr_id int
);
insert into employees_full_data values(1, 'Shashank', 3);
insert into employees_full_data values(2, 'Amit', 3);
insert into employees_full_data values(3, 'Rajesh', 4);
insert into employees_full_data values(4, 'Ankit', 6);
```

```
insert into employees_full_data values(6, 'Nikhil', null);

select * from employees_full_data;

# Write a query to print the distinct names of managers??
select
emp.name as manager
from employees_full_data emp
inner join (select distinct mgr_id as mgr_id from employees_ful
```

C4_oct15_2023:

1.) Exists and Not Exists

2.) Window Functions

3.) Frame Clause

4.) Coalesce Function

5.) Common Table Expressions - Iterative and Recursive

SQL_Class_4_PDF_Notes_Part_1.pdf

SQL_Class_4_PDF_Notes_Part_2.pdf

SQL_Class_4_Rough_Notes.pdf

```
--- Group  RollUp
--- It will combine those particular column data into single chu
```

```
CREATE TABLE payment (payment_amount decimal(8,2),
payment_date date,
store_id int);

INSERT INTO payment VALUES
(1200.99, '2018-01-18', 1),(189.23, '2018-02-15', 1),(33.43, '20
(7382.10, '2019-01-11', 2),(382.92, '2019-02-18', 1),(322.34, '2
(2929.14, '2020-01-03', 2),(499.02, '2020-02-19', 3),(994.11, '2
(394.93, '2021-01-22', 2),(3332.23, '2021-02-23', 3),(9499.49,
(3002.43, '2018-02-25', 2),(100.99, '2019-03-07', 1),(211.65, '2
(500.73, '2021-01-06', 3);

--  Write a query to calculate total reveue of each shop
--  per year, also display year wise revenue

SELECT
  SUM(payment_amount),
  YEAR(payment_date) AS 'Payment Year',
  store_id AS 'Store'
FROM payment
GROUP BY YEAR(payment_date), store_id WITH ROLLUP
ORDER BY YEAR(payment_date), store_id;


-- Write a query to calculate total revenue per year
SELECT
  SUM(payment_amount),
  YEAR(payment_date) AS 'Payment Year'
FROM payment
GROUP BY YEAR(payment_date)
ORDER BY YEAR(payment_date);


# Any Operation
CREATE TABLE Students (
```

```
    StudentID INT,
    StudentName VARCHAR(50)
);

Describe table Students;

INSERT INTO Students VALUES
(1, 'John'),(2, 'Alice'),(3, 'Bob');

CREATE TABLE Courses (
    CourseID INT,
    CourseName VARCHAR(50)
);
INSERT INTO Courses VALUES
(100, 'Math'),(101, 'English'),(102, 'Science');

CREATE TABLE Enrollments (
    StudentID INT,
    CourseID INT
);
INSERT INTO Enrollments VALUES
(1, 100),(1, 101),(2, 101),(2, 102),(3, 100),(3, 102);


-- Example: Lets find the students who are enrolled in any cours
-- class way
Select Distinct s.StudentName
From Students s
INNER JOIN Enrollments e ON s.StudentID = e.StudentID
WHERE s.StudentName <> 'John' and e.CourseID = ANY (SELECT e2.C
    FROM Enrollments e2
    INNER JOIN Students s2 ON e2.StudentID = s2.StudentID
    WHERE s2.StudentName = 'John');

-- my way
WITH cte AS (
```

```
SELECT e.courseid
FROM enrollments e
LEFT JOIN students s ON e.studentid = s.studentid
WHERE s.studentname = 'John'
)
SELECT s.studentname
FROM students s
LEFT JOIN enrollments e ON e.studentid = s.studentid
WHERE s.studentname != 'John' AND e.courseid IN (SELECT * FROM c



-----------------------------------------------------------------
# ALL
CREATE TABLE Products (
    ProductID INT,
    ProductName VARCHAR(50),
    Price DECIMAL(5,2)
);
INSERT INTO Products VALUES
(1, 'Apple', 1.20),(2, 'Banana', 0.50),(3, 'Cherry', 2.00),(4,


CREATE TABLE Orders (
    OrderID INT,
    ProductID INT,
    Quantity INT
);
INSERT INTO Orders VALUES
(1001, 1, 10),(1002, 2, 20),(1003, 3, 30),(1004, 1, 5),(1005, 4,


/* Now, suppose we want to find the products that have a price l
price of all products ordered in order 1001 */

SELECT p.ProductName
FROM Products p
```

```
WHERE p.Price < ALL (
    SELECT pr.Price
    FROM Products pr
    INNER JOIN Orders o ON pr.ProductID = o.ProductID
    WHERE o.OrderID = 1001
);


------------ EXISTS Operation -------------------------

CREATE TABLE Customers (
    CustomerID INT,
    CustomerName VARCHAR(50)
);
INSERT INTO Customers VALUES
(1, 'John Doe'),(2, 'Alice Smith'),(3, 'Bob Johnson'),(4, 'Char

CREATE TABLE Orders (
    OrderID INT,
    CustomerID INT,
    OrderDate DATE
);
INSERT INTO Orders VALUES
(1001, 1, '2023-01-01'),(1002, 2, '2023-02-01'),(1003, 1, '2023
(1004, 3, '2023-04-01'),(1005, 5, '2023-05-01');


--- Example: Lets find the customers who have placed at least or
SELECT c.CustomerName
FROM Customers c
WHERE EXISTS (
    SELECT 1
    FROM Orders o
    WHERE o.CustomerID = c.CustomerID
);
```

```
------------------------ NOT EXISTS Operation ------------------
-- Example: Lets find the customers who have not placed any orde

SELECT c.CustomerName
FROM Customers c
WHERE NOT EXISTS (
    SELECT 1
    FROM Orders o
    WHERE o.CustomerID = c.CustomerID
);


----------------------  WINDOW FUNCTIONS  ----------------------

create table shop_sales_data
(
sales_date date,
shop_id varchar(5),
sales_amount int
);
insert into shop_sales_data values('2022-02-14','S1',200),('2022
('2022-02-14','S2',600),('2022-02-15','S3',500),('2022-02-18','S
('2022-02-17','S2',250),('2022-02-20','S3',300);


/* Total count of sales for each shop using window function
   Working functions - SUM(), MIN(), MAX(), COUNT(), AVG() */


# If we only use Order by In Over Clause

select *,
       sum(sales_amount) over(order by sales_amount desc) as run
from shop_sales_data;

| sales_date | shop_id | sales_amount | running_sum_of_sales |
```

```
| 2022-02-14 | S2 | 600 | 1800 |
| 2022-02-15 | S2 | 600 | 1800 |
| 2022-02-16 | S2 | 600 | 1800 |--
| 2022-02-15 | S3 | 500 | 3300 |
| 2022-02-16 | S3 | 500 | 3300 |
| 2022-02-18 | S3 | 500 | 3300 |--
| 2022-02-18 | S1 | 400 | 4500 |
| 2022-02-19 | S1 | 400 | 4500 |
| 2022-02-20 | S1 | 400 | 4500 |--
| 2022-02-15 | S1 | 300 | 5700 |
| 2022-02-20 | S3 | 300 | 5700 |
| 2022-02-22 | S1 | 300 | 5700 |
| 2022-02-19 | S3 | 300 | 5700 |--
| 2022-02-17 | S2 | 250 | 5950 |--
| 2022-02-14 | S1 | 200 | 6350 |
| 2022-02-25 | S1 | 200 | 6350 |--

/* here over adds all the sales amount consequetly(previous row
 and shows the sum in current row as running_sum_of_sales
 over clause makes partition on sales_amount level but aggregat



# If we only use Partition By
select *,
       sum(sales_amount) over(partition by shop_id) as total_sum
from shop_sales_data;
| sales_date | shop_id | sales_amount | total_sum_of_sales |

| 2022-02-14 | S1 | 200 | 2200 |
| 2022-02-15 | S1 | 300 | 2200 |
| 2022-02-18 | S1 | 400 | 2200 |
| 2022-02-19 | S1 | 400 | 2200 |
| 2022-02-20 | S1 | 400 | 2200 |
| 2022-02-22 | S1 | 300 | 2200 |
```

```
| 2022-02-25 | S1 | 200 | 2200 | ---
| 2022-02-14 | S2 | 600 | 2050 |
| 2022-02-17 | S2 | 250 | 2050 |
| 2022-02-15 | S2 | 600 | 2050 |
| 2022-02-16 | S2 | 600 | 2050 | ---
| 2022-02-15 | S3 | 500 | 2100 |
| 2022-02-20 | S3 | 300 | 2100 |
| 2022-02-16 | S3 | 500 | 2100 |
| 2022-02-18 | S3 | 500 | 2100 |
| 2022-02-19 | S3 | 300 | 2100 |---
/* here partition happens based on shop_id and sums up all the v
 value in every column of that particular shop_id
 aggregation calculations happens on partition chunk only */


# If we only use Partition By & Order By together
select *,
       sum(sales_amount) over(partition by shop_id order by sale
from shop_sales_data;

| sales-date | shop_id | sales_amount | running_sales_amount |

| 2022-02-18 | S1 | 400 | 1200 |
| 2022-02-19 | S1 | 400 | 1200 |
| 2022-02-20 | S1 | 400 | 1200 |---
| 2022-02-15 | S1 | 300 | 1800 |
| 2022-02-22 | S1 | 300 | 1800 |---
| 2022-02-14 | S1 | 200 | 2200 |
| 2022-02-25 | S1 | 200 | 2200 |---  ====
| 2022-02-14 | S2 | 600 | 1800 |
| 2022-02-15 | S2 | 600 | 1800 |
| 2022-02-16 | S2 | 600 | 1800 |---
| 2022-02-17 | S2 | 250 | 2050 |---  ====
| 2022-02-15 | S3 | 500 | 1500 |
| 2022-02-16 | S3 | 500 | 1500 |
| 2022-02-18 | S3 | 500 | 1500 |---
```

```
| 2022-02-20 | S3 | 300 | 2100 |
| 2022-02-19 | S3 | 300 | 2100 |--- =====

/* partition happens based on shop_id and then every column adds
 specific shop_id (similar to over above but divides specific li
 partition created chunk on top of that orderby creates mini chu
 calculation
 */


select *,
       sum(sales_amount) over(partition by shop_id order by sale
       as running_sum_of_sales,
       avg(sales_amount) over(partition by shop_id order by sale
       as running_avg_of_sales,
       max(sales_amount) over(partition by shop_id order by sale
       as running_max_of_sales,
       min(sales_amount) over(partition by shop_id order by sale
       as running_min_of_sales
from shop_sales_data;


-------------------------------------------------------------

create table amazon_sales_data
(
    sales_date date,
    sales_amount int
);
insert into amazon_sales_data values('2022-08-21',500),('2022-08
('2022-08-18',200),('2022-08-25',800);

# Query - Calculate the date wise rolling average of amazon sale

select * from amazon_sales_data;
```

```
select *,
       avg(sales_amount) over(order by sales_date) as rolling_av
from amazon_sales_data;

select *,
       avg(sales_amount) over(order by sales_date) as rolling_av
       sum(sales_amount) over(order by sales_date) as rolling_su
from amazon_sales_data;
```

```
---- Rank(), Row_Number(), Dense_Rank() window functions -------

Row_number vs Rank vs Dense_Rank

Row_number  == 1,2,3,4,5,6,7,8,9
Rank        == 1,1,1,4,4,6,6,8,9
Dense_Rank  == 1,1,1,2,2,3,3,3,4


insert into shop_sales_data values('2022-02-19','S1',400),('2022
('2022-02-22','S1',300),('2022-02-25','S1',200),('2022-02-15','S
('2022-02-16','S2',600),('2022-02-16','S3',500),('2022-02-18','S
('2022-02-19','S3',300);


select *,
       row_number() over(partition by shop_id order by sales_amo
       rank() over(partition by shop_id order by sales_amount de
       dense_rank() over(partition by shop_id order by sales_amo
from shop_sales_data;



create table employees
(
```

```
    emp_id int,
    salary int,
    dept_name VARCHAR(30)

);

insert into employees values(1,10000,'Software'),(2,11000,'Softw
(4,11000,'Software'),(5,15000,'Finance'),(6,15000,'Finance'),(7,
(8,12000,'HR'),(9,12000,'HR'),(10,11000,'HR')


/*  Query - get one employee from each department who is getting
(employee can be random if salary is same) */

select
    tmp.*
from (select *,
        row_number() over(partition by dept_name order by salary
    from employees) tmp
where tmp.row_num = 1;

# Query - get all employees from each department who are getting
select
    tmp.*
from (select *,
        rank() over(partition by dept_name order by salary desc)
    from employees) tmp
where tmp.rank_num = 1;


# Query - get all top 2 ranked employees from each department wh
select
    tmp.*
from (select *,
        dense_rank() over(partition by dept_name order by salary
    from employees) tmp
```

```
where tmp.dense_rank_num <= 2;



----------------------  EXAMPLE for LAG  &  LEAD ---------------

create table daily_sales
(
sales_date date,
sales_amount int
);

insert into daily_sales values('2022-03-11',400),('2022-03-12',5
('2022-03-14',600),('2022-03-15',500),('2022-03-16',200);

select * from daily_sales;

select *,
      lag(sales_amount, 1) over(order by sales_date) as pre_day_
from daily_sales;


# we can use this to replace null with defualt value like 0
select *,
  coalesce(lag(sales_amount,1) over(order by sales_date), 0) as
from daily_sales;

# Query - Calculate the differnce of sales with previous day sal
# Here null will be derived
select sales_date,
       sales_amount as curr_day_sales,
       lag(sales_amount, 1) over(order by sales_date) as prev_da
       sales_amount - lag(sales_amount, 1) over(order by sales_d
from daily_sales;
```

```
# Here we can replace null with 0
select sales_date,
       sales_amount as curr_day_sales,
       lag(sales_amount, 1, 0) over(order by sales_date) as prev
       sales_amount - lag(sales_amount, 1, 0) over(order by sale
from daily_sales;



# Diff between lead and lag
select *,
      lag(sales_amount, 1) over(order by sales_date) as pre_day_
from daily_sales;

select *,
      lead(sales_amount, 1) over(order by sales_date) as next_da
from daily_sales;


-----------------------------------------------

create table employees(
  emp_id int,
  emp_name varchar(50),
  mobile BIGINT,
  dept_name varchar(50),
  salary int
);
insert into employees values(1,'Shashank',778768768,'Software',1
(3,'Amit',098798998,'HR',5000),(4,'Nikhil',67766767,'IT',3000);

select * from employees;


-------------------------  Create VIEWS in SQL  ----------------
```

```
/* Views are temporary tables to hide complex queries and show p
 from entire data of table/tables */

create view employee_data_for_finance as select emp_id, emp_name

select * from employee_data_for_finance;


--- Create logic for department wise salary sum
create view department_wise_salary as select dept_name, sum(sala

drop view department_wise_salary;

create view department_wise_salary as select dept_name, sum(sala




------------------  How to use FRAME CLAUSE - Rows BETWEEN  ----

select * from daily_sales;

select *,
      sum(sales_amount) over(order by sales_date rows between 1
from daily_sales;


select *,
      sum(sales_amount) over(order by sales_date rows between 1
from daily_sales;


select *,
      sum(sales_amount) over(order by sales_date rows between cu
from daily_sales;
```

```
select *,
      sum(sales_amount) over(order by sales_date rows between 2
from daily_sales;


select *,
      sum(sales_amount) over(order by sales_date rows between un
from daily_sales;


select *,
      sum(sales_amount) over(order by sales_date rows between cu
from daily_sales;


select *,
      sum(sales_amount) over(order by sales_date rows between un
from daily_sales;


# Alternate way to esclude computation of current row
select *,
      sum(sales_amount) over(order by sales_date rows between un
from daily_sales;


# How to work with Range Between
select *,
      sum(sales_amount) over(order by sales_amount range betwee
from daily_sales;


# Calculate the running sum for a week
# Calculate the running sum for a month
```

```
insert into daily_sales values('2022-03-20',900),('2022-03-23',2
('2022-03-29',250);

select * from daily_sales;

select *,
       sum(sales_amount) over(order by sales_date range between
from daily_sales;


--------------- EXTRACT, DATE_ADD, DATE_DIFF  -------------

-- https://onecompiler.com/mysql/3yzzmunte

** SELECT DATE(activity_date) FROM your_table;
select EXTRACT(week from '2023-1-23') == it extracts which week,


** TIMESTAMPDIFF(unit, start_datetime, end_datetime),
SELECT TIMESTAMPDIFF(DAY, '2023-11-22', '2023-12-05') AS days_di


DATEDIFF(first_date,second_date)= -1  == we see 2nd date higher


SELECT DATE_ADD("2017-06-15", INTERVAL 2 DAY);= adds 2 days to t

SELECT DATE_SUB("2017-06-15", INTERVAL 2 DAY); = sees differenc

DATE_FORMAT(trans_date,'%Y-%m') AS month -- trans_date = 2023-02
SUBSTR(trans_date,1,7) AS month -- we get the same above output


CREATE TABLE date_functions_demo (
    id INT ,
    start_date DATE,
```

```
    end_date DATE,
    created_at TIMESTAMP,
    updated_at TIMESTAMP,
 system_date varchar(10)
);

INSERT INTO date_functions_demo (id,start_date, end_date, create
(1,'2024-01-01', '2024-12-31', '2024-01-01 10:00:00', '2024-12-3
(2,'2023-06-15', '2024-06-15', '2023-06-15 08:30:00', '2024-06-1
(3,'2022-05-20', '2023-05-20', '2022-05-20 12:15:00', '2023-05-2

# 1.EXTRACT()
SELECT id,start_date,EXTRACT(YEAR FROM start_date) AS sd FROM da
SELECT id,updated_at,EXTRACT(HOUR FROM updated_at) AS sd FROM da

# 2. NOW() / CURRENT_TIMESTAMP()
SELECT NOW();

# 3. CURRENT_DATE()
SELECT CURRENT_DATE();

# 4. CURRENT_TIME()
SELECT CURRENT_TIME() AS current_tiem_col FROM date_functions_de

# 5. DATE()
SELECT id,created_at,DATE(created_at), CAST(created_at AS date)

# 6. DATE_FORMAT()
SELECCT id,start_date, DATE_FORMAT(start_date,'%m/%d/%Y') FROM d

# 7. DATEDIFF()
SELECT id,start_date,end_date, DATEDIFF(end_date,start_date) AS
SELECT id,start_date,end_date, end_date-start_date AS no_of_date

# 8. DATE_ADD()
SELECT id,start_date,DATE_ADD(start_date, interval 5 hour) AS st
```

```
# 9. DATE_SUB()
SELECT id,start_date,DATE_SUB(start_date, interval 1 day) AS sta

# 10. DAY()
SELECT id, start_date,DAY(start_date) AS day_sd FROM date_functi

# 11. MONTH()
SELECT id, start_date, MONTH(start_date) AS month_sd FROM date_f

# 12. YEAR()
SELECT id, start_date, DAYOFWEEK(start_date) AS sd from date_fun

# 13. DAYOFWEEK()
SELECT id, start_date, DAYOFWEEK(start_date) AS sd FROM date_fun

# 14. DAYOFYEAR()
SELECT id, start_date, DAYOFYEAR(start_date) AS sd FROM date_fun

# 15.DAYOFMONTH()
SELECT id, start_date, DAYOFMONTH(start_date) AS sd FROM date_fu

# 16. MONTHNAME()
SELECT id, start_date, MONTHNAME(start_date) AS sd FROM date_fun

# 17. DAYNAME()
SELECT id, start_date, DAYNAME(start_date) AS sd FROM date_funct

# 18. STR_TO_DATE() - date format should be given in string_date
SELECT id,system_date,STR_TO_DATE(system_date,%m/%d%Y) AS sd FRO

# 19. TIMESTAMPDIFF()
SELECT *, TIMESTAMPDIFF(MINUTE,created_at,updated_at) AS sd FROM
SELECT *, TIMESTAMPDIFF(DAY,created_at,updated_at) AS sd FROM da
```

```
# 20. LAST_DAY()
SELECT id,start_date,LAST_DAY(start_date) AS sd FROM date_functi
```

```
------------------------ Common Table Expression ---------------

create table amazon_employees(
    emp_id int,
    emp_name varchar(20),
    dept_id int,
    salary int

 );
insert into amazon_employees values(1,'Shashank', 100, 10000),(2
(3,'Amit', 101, 15000),(4,'Mohit', 101, 17000),(5,'Nikhil', 102,

 create table department
 (
    dept_id int,
    dept_name varchar(20)
  );
 insert into department values(100, 'Software'),(101, 'HR'),(102


/* Write a query to print the name of department along with the
department
Normal approach */

select d.dept_name, tmp.total_salary
from (select dept_id , sum(salary) as total_salary from amazon_e
inner join department d on tmp.dept_id = d.dept_id;
```

```
--- how to do it using with clause??
with dept_wise_salary as(
select dept_id , sum(salary) as total_salary
from amazon_employees
group by dept_id)

select d.dept_name, tmp.total_salary
from dept_wise_salary tmp
inner join department d on tmp.dept_id = d.dept_id;


with dept_wise_salary as(
select dept_id , sum(salary) as total_salary
from amazon_employees
group by dept_id),
dept_wise_max_salary as(
select dept_id , max(salary) as max_salary
from amazon_employees
group by dept_id)

select * from dept_wise_max_salary;


---------------- RECURSIVE CTE ----------------
--- Write a Query to generate numbers from 1 to 10 in SQL

with recursive generate_numbers as
(
  select 1 as n                    -- anchor member
  union
  select n+1 from generate_numbers where n<10
)
select * from generate_numbers;
```

```
-----------------------------
create table emp_mgr
(
id int,
name varchar(50),
manager_id int,
designation varchar(50),
primary key (id)
);
insert into emp_mgr values(1,'Shripath',null,'CEO'),(2,'Satya',5
(4,'David',5,'DS'),(5,'Michael',7,'Manager'),(6,'Arvind',7,'Arch
(7,'Asha',1,'CTO'),(8,'Maryam',1,'Manager');

select * from emp_mgr;


--- for our CTO 'Asha', present her org chart

with recursive emp_hir as
(
   select id, name, manager_id, designation from emp_mgr where n
   UNION
   select em.id, em.name, em.manager_id, em.designation from emp
)
select * from emp_hir;


--- Print level of employees as well
with recursive emp_hir as
(
   select id, name, manager_id, designation, 1 as lvl from emp_m
   UNION
   select em.id, em.name, em.manager_id, em.designation, eh.lvl
```

```
)
select * from emp_hir;
```

REGEX expressions:

Pattern	What the Pattern matches
*	Zero or more instances of string preceding it
+	One or more instances of strings preceding it
.	Any single character
?	Match zero or one instances of the strings preceding it.
^	caret(^) matches Beginning of string
$	End of string
[abc]	Any character listed between the square brackets
[^abc]	Any character not listed between the square brackets
[A-Z]	match any upper case letter.
[a-z]	match any lower case letter
[0-9]	match any digit from 0 through to 9.
[[:<:]]	matches the beginning of words.
[[:>:]]	matches the end of words.
[:class:]	matches a character class i.e. [:alpha:] to match letters, [:space:] to match white space, [:punct:] is match punctuations and [:upper:] for upper class letters.
p1|p2|p3	Alternation; matches any of the patterns p1, p2, or p3
{n}	n instances of preceding element
{m,n}	m through n instances of preceding element

Examples with explanation :

- **Match beginning of string(^):** Gives all the names starting with ‘sa’.Example-sam,samarth.

```
SELECT name FROM student_tbl WHERE name REGEXP '^sa';
```

- **Match the end of a string($):** Gives all the names ending with ‘on’.Example – norton,merton.

```
SELECT name FROM student_tbl WHERE name REGEXP 'on$';
```

- **Match zero or one instance of the strings preceding it(?):** Gives all the titles containing ‘com’.Example – comedy , romantic comedy.

```
SELECT title FROM movies_tbl WHERE title REGEXP 'com?';
```

- **matches any of the patterns p1, p2, or p3(p1|p2|p3):** Gives all the names containing ‘be’ or ‘ae’.Example – Abel, Baer.

```
SELECT name FROM student_tbl WHERE name REGEXP 'be|ae' ;
```

- **Matches any character listed between the square brackets([abc]):** Gives all the names containing ‘j’ or ‘z’.Example – Lorentz, Rajs.

```
SELECT name FROM student_tbl WHERE name REGEXP '[jz]' ;
```

- **Matches any lower case letter between ‘a’ to ‘z’- ([a-z]) ([a-z] and (.)):** Retrieve all names that contain a letter in the range of ‘b’ and ‘g’, followed by any character, followed by the letter ‘a’.Example – Tobias, sewall. Matches any single character(.)

```
SELECT name FROM student_tbl WHERE name REGEXP '[b-g].[a]' ;
```

- **Matches any character not listed between the square brackets. ([^abc]):** Gives all the names not containing ‘j’ or ‘z’. Example – nerton, sewall.

```
SELECT name FROM student_tbl WHERE name REGEXP '[^jz]' ;
```

- **Matches the end of words[[:>:]]:** Gives all the titles ending with character “ack”. Example – Black.

```
SELECT title FROM movies_tbl WHERE REGEXP 'ack[[:>:]]';
```

- **Matches the beginning of words[[:<:]]:** Gives all the titles starting with character “for”. Example – Forgetting Sarah Marshal.

```
SELECT title FROM movies_tbl WHERE title REGEXP '[[:<:]]for';
```

- **Matches a character class[:class:]:** i.e [:lower:]- lowercase character ,[:digit:] – digit characters etc. Gives all the titles containing alphabetic character only. Example – stranger things, Avengers.

```
SELECT title FROM movies_tbl WHERE REGEXP '[:alpha:]' ;
```

- **Matches the beginning of all words by any character listed between the square brackets.(^[abc]):** Gives all the names starting with ‘n’ or ‘s’. Example – nerton, sewall.

```
SELECT name FROM student_tbl WHERE name REGEXP '^[ns]' ;
```

- **Matches the ending of all words by any character listed between the square brackets.([abc]$):** Gives all the names ending with ‘n’ or ‘s’. Example – pen, pass.

```
SELECT name FROM student_tbl WHERE name REGEXP '[ns]$' ;
```

Practise queries

LeetCode_SQL