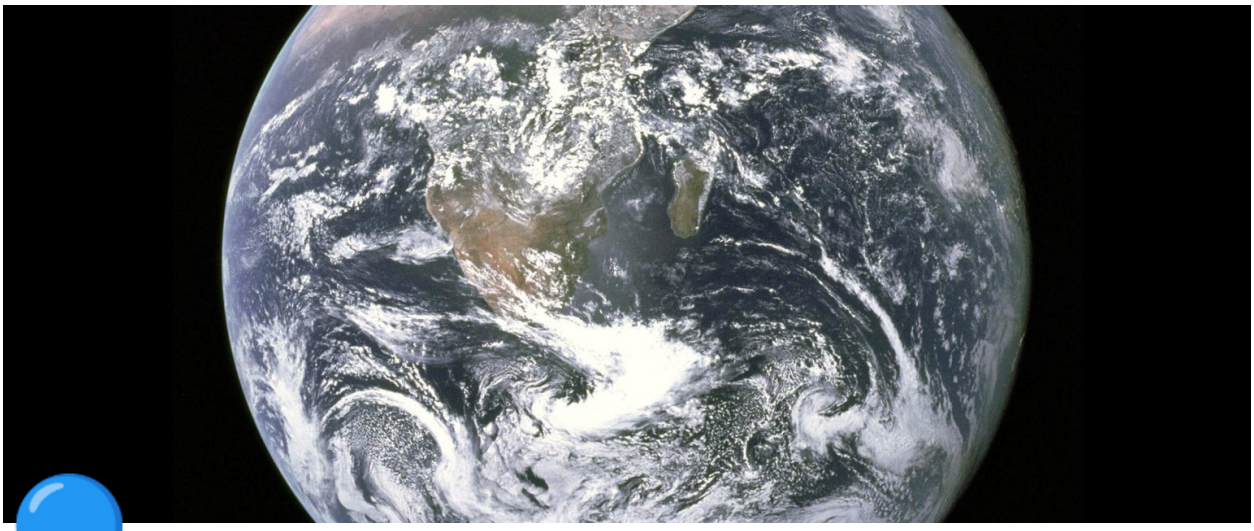

MySQL

List all databases → show databases;

Create new database → create database db_name;

Change to a custom db → use db_name;

To delete a database → drop database db_name;

To know the current using database → select database();

Creating a table → create table table_name(column_name datatype, column_name datatype);

To list columns of a table → show columns from table_name;

or desc table_name;

To drop a table → drop table table_name;

Inserting data into table → insert into table_name values (column_value, column_value);

or insert into table_name(column_name, column_name) values (column_value, column_value);

Inserting multiple data into table at a time → INSERT INTO table_name(column_name, column_name) VALUES (value, value), (value, value), (value, value);

Setting not null while creating table → create table table_name(column_name datatype not null, column_name datatype not null);

Creating table with a primary key → create table table_name(column_name datatype, column_name not null datatype, primary key(column_name))

Setting the primary key field as auto increment → create table table_name(column_name datatype, column_name not null auto_increment datatype, primary key(column_name))

Select particular columns from a table → select column_name ,column_name from table_name;

Using where clause while selecting → select column_name from table_name where column_name='value';

Using alias → select column_name as name from table_name;

Update fields → UPDATE table_name SET column_name='new_value'
WHERE column_name='old_value';

Delete values → delete from tables where name='value';

Concat function → select concat(field1, separator, field2) from table_name;

Concat_Ws function → select concat_ws('separator', field1, field2) from table_name;

substring function → select substring('field_value', arg1, arg2);

Ex: select substring('Hello World', 1, 4) → Hell

select substring('Hello World', 7) → World

select substring('Hello World', -7) → o World

replace function → select replace(string, old_value, new_value);

reverse function → select reverse('string');

char_length function → select char_length('string');

upper function → select upper('string');

lower function → select lower('string');

distinct → select distinct column_name from table_name;

order by → select field_name from table_name order by field_name;

order by descending order → select field_name from table_name order by field_name desc;

limit → select field_name from table_name limit value;

like → select field_name from table_name where field_name like '%value%';

Ex: select title from books where author_fname like '%da%' → select all the titles where the author name includes 'da';

Ex: select title from books where author_fname like 'da%' → all the names that start with 'da'

Ex: select title from books where author_fname like '%da' → all the names that end with 'da'

Ex: select title, stock_quantity from books where stock_quantity like '____'; → select all the books where the stock_quantity is of 4 characters

Ex: select title from books where title like '%\%%' → all the books that have % in it.

Ex: select title from books where title like '%_%' → all the books that have _ in it.

count function → select count(*) from table_name;

distinct count → select count(distinct field_name) from books;

group by function → select field_name from table_name group by field_name;

Ex: select author_lname, count(*) from books group by author_lname;

max function → select max(field_name) from table_name

min function → select min(field_name) from table_name

sum function → select sum(field_name) from table_name;

average function → select avg(field_name) from books;

decimal data type → decimal(5, 2)

EX: CREATE TABLE items(price DECIMAL(5,2));

float data type →

Ex: CREATE TABLE thingies (price FLOAT);

date data type → 'YYYY-MM-DD'

Ex: CREATE TABLE people (name VARCHAR(100), birthdate DATE, birthtime TIME, birthdt DATETIME);

not equal operator → select field_name from table_name where field_name ≠ value

Ex: select title, released_year from books where released_year ≠ 2017;

not like operator → select field_name from table_name where field_name not like value;

Ex: select title from books where title not like 'w%';

greater than operator → select field_name from table_name where field_name > value;

Ex: select title from books where released_year > 2000;

greater than or equal to operator → select field_name from table_name where field_name ≥ value;

Ex: select title from books where released_year >= 2000;

less than operator → select field_name from table_name where field_name < value;

Ex: select title from books where released_year < 2000;

less than or equal to operator → select field_name from table_name where field_name ≤ value;

Ex: select title from books where released_year ≤ 2000;

and operator → select field_name from table_name where field_name > value and field_name > value;

Ex: select title from books where author_lname='Eggers' and released_year>2010;

or operator → select field_name from table_name where field_name > value or field_name > value;

Ex: select title from books where author_lname='Eggers' || released_year>2010;

between operator → select field_name from table_name where field_name between value1 and value2;

Ex: select title, released_year from books where released_year between 2001 and 2021;

not between operator → select field_name from table_name where field_name not between value1 and value2;

Ex: select title, released_year from books where released_year not between 2001 and 2021;

in operator → select field_name from table_name where field_name in (value1, value2, value3);

Ex: select title, author_lname from books where author_lname in ('Carver', 'Lahiri', 'Smith');

not in operator → select field_name from table_name where field_name not in (value1, value2, value3);

Ex: select title, released_year from books where released_year not in(2000, 2002, 2004, 2006, 2008, 2010, 2012, 2014, 2016);

modulo operator → select field_name from table_name where field_name %value > value;

Ex: SELECT title, released_year FROM books WHERE released_year >= 2000 AND released_year % 2 != 0;

case statement: select field_name CASE

WHEN *condition1* THEN *result1*

WHEN *condition2* THEN *result2*

WHEN *condition N* THEN *result N*

ELSE *result*

END from table_name;

Ex: select title, released_year,
-> case
-> when released_year >= 2000 then 'Modern Lit'
-> else '20th Century Lit'
-> end as genre
-> from books;

Foreign Key: foreign key(field_name_current_table) references table_name(field_name_parent_table);

Ex: FOREIGN KEY(customer_id) REFERENCES customers(id)

Inner Join: Implicit join

Ex: select * from customers, orders where customers.id = orders.customer_id;

Explicit inner join:

Ex: select * from customers join orders on customers.id = orders.customer_id;

Left Join:

Ex: select * from customers left join orders on customers.id = orders.customer_id;

- CREATE OR REPLACE VIEW full_reviews AS
-- The ROLLUP option allows you to include extra rows that represent the subtotals,
-- essentially creating a subtotal for each row

Trigger:

Syntax → CREATE TRIGGER trigger_name
trigger_time trigger_event ON table_name FOR EACH ROW
BEGIN
...
END;

Ex:

```
DELIMITER $$

CREATE TRIGGER must_be_adult
BEFORE INSERT ON users FOR EACH ROW
BEGIN
IF NEW.age < 18
THEN
SIGNAL SQLSTATE '45000'
SET MESSAGE_TEXT = 'Must be an adult!';
END IF;
END;
$$

DELIMITER ;
```

SQL state 45000 → a generic state representing unhandled user-defined exception

Trigger time → before, after

Trigger events → insert, update, delete

Delimiter → By default semi-colon is the delimiter in MySQL. While writing triggers, as it involves multiple lines, we change the delimiter to something else rather than a semi-colon.

In a relational database, a database is just a bunch of tables.

Primary key → an unique identifier

SQL is case-insensitive

GROUP BY summarizes or aggregates similar data into single rows

Inner join: Select all records from A and B where the join condition met.

Left Join: Select everything from A along with any matching records in B.

Right Join: Select everything from B and any matching record in A.

Triggers: Triggers are SQL statements that run automatically when a table is changed.

time data type → '10:07:35'

datetime data type → '1983-11-11 10:07:35'

curdate function → select curdate() → returns date

now function → select now() → returns date time

curtime function → select curtime() → returns curtime

What is meant by ERD?

ERD stands for Entity-relationship-diagram.

It is a graphical representation of tables with the relation between them.

What are indexes in the database?

Indexes are quick references for the fast retrieval of data from databases. There are two kinds of indexes:

- Clustered index
- Non-clustered index

Some commonly used commands in SQL

Top 25 SQL Questions for Interview

This article will focus on a few key concepts and practicals that one may find useful for preparations for an SQL interview. This article will primarily focus on constructing SQL queries and...

https://rajansahu713.medium.com/top-25-sql-questions-for-interview-9a28202dcd14

SQL Server COALESCE() Function

Return the first non-null value in a list:

SELECT COALESCE(NULL, NULL, NULL, 'W3Schools.com', NULL, 'Example.com');

w3schools