Git commands:

C:\Users\SDO>cd C:\Users\SDO\Desktop\Project\Pd-Np\git – to enter in a specific file

Git – to check wheather git is installed or not

Git –version – 2.44.0…..

Git config -l 🡪 user name/ pass dekha jbe if it was set previously

Git config –global user.name ‘soumik’ 🡪 to set username

Git config –global user.email ‘….@gmail.com’

Git config -l 🡪 to check wheather the name is set or not

Git init🡪 to initalise a file as git repo

Git status🡪 wheather the file is tracked or not, …etc

Wildcard = if we want to stage some particular files but not all the files , and without the naming of all files we can stage them . let suppose there are 10 files with the starting name is ‘prod’ then we just need to run

Git add prod\*

Git add . 🡪 all the files will staged for commit

Git reset <file name> or git rm –cached <file name> 🡪 for unstage the file which is already staged

Git diff 🡪 [what is changed but not staged] modified file a ki ki change hoyeche (add, delete)

Git diff –staged 🡪 changes staged but not commited

Git commit -m “ file added to master branch”

Git commit -a -m “add and commit both at a single command”

Git rm <file name > = to untrack the file

Git revert 🡪 delete

[primary key](https://www.geeksforgeeks.org/primary-key-constraint-in-sql/): A [primary key](https://www.geeksforgeeks.org/primary-key-constraint-in-sql/) is used to ensure that data in the specific column is unique. A column cannot have NULL values.

| **PRIMARY KEY** | **FOREIGN KEY** |
| --- | --- |
| A primary key is used to ensure data in the specific column is unique. | A foreign key is a column or group of columns in a relational database table that provides a link between data in two tables. |
| It uniquely identifies a record in the relational database table. | It refers to the field in a table which is the primary key of another table. |
| Only one primary key is allowed in a table. | Whereas more than one foreign key is allowed in a table. |
| It is a combination of UNIQUE and Not Null constraints. | It can contain duplicate values and a table in a relational database. |
| It does not allow NULL values. | It can also contain NULL values. |
| Its value cannot be deleted from the parent table. | Its value can be deleted from the child table. |
| It constraint can be implicitly defined on the temporary tables. | It constraint cannot be defined on the local or global temporary tables. |

Aggregate () is used for max, min, avg, sum, count cases.

Q-1)If i want to update all the elements of a column let suppose column name 'dept\_id' values (101,102,103,104) I want to update it with (1,2,3,4)

Existing tables

select \* from employees

inner join department

on employees.dept\_id = department.dept\_id;

UPDATE employees

SET dept\_id = CASE

    WHEN dept\_id = 101 THEN 1

    WHEN dept\_id = 102 THEN 2

    WHEN dept\_id = 103 THEN 3

    WHEN dept\_id = 104 THEN 4

    ELSE dept\_id

END;

* The **ELSE dept\_id** clause ensures that if **dept\_id** does not match any of the specified values (101, 102, 103, 104), it remains unchanged.
* The **END** keyword marks the end of the **CASE** expression.

CREATE TABLE employee (

    ID SERIAL PRIMARY KEY,

    EMPNO VARCHAR(10) UNIQUE NOT NULL,

    NAME VARCHAR(100) NOT NULL,

    DEPT\_ID INT,

);

INSERT INTO employee (EMPNO, NAME, DEPT\_ID, MANAGER\_ID, PHONE\_NO, HIRE\_DATE, SALARY, BONUS, COMM)

VALUES

('E001', 'John Doe', 1, NULL, '123-456-7890', '2020-01-15', 60000, 5000, 2000),

('E002', 'Jane Smith', 1, 1, '234-567-8901', '2019-05-20', 65000, 5500, 2500),

('E003', 'Emily Johnson', 2, 1, '345-678-9012', '2021-03-10', 70000, 6000, 3000),

('E004', 'Michael Brown', 2, 2, '456-789-0123', '2020-07-05', 75000, 6500, 3500),

**Some important Quaries:**

-- SELECT Statement

SELECT \* FROM employees;

-- INSERT Statement

INSERT INTO employees (first\_name, last\_name, email) VALUES ('John', 'Doe', 'john.doe@example.com');

-- UPDATE Statement

UPDATE employees SET email = 'johndoe@example.com' WHERE employee\_id = 1;

-- DELETE Statement

DELETE FROM employees WHERE employee\_id = 1;

-- JOIN Operations

SELECT employees.first\_name, departments.department\_name

FROM employees

INNER JOIN departments ON employees.department\_id = departments.department\_id;

-- GROUP BY and Aggregation Functions

SELECT department\_id, COUNT(\*) as employee\_count

FROM employees

GROUP BY department\_id;

-- ORDER BY

SELECT \* FROM employees

ORDER BY last\_name ASC, first\_name ASC;

-- LIMIT and OFFSET

SELECT \* FROM employees

LIMIT 10 OFFSET 5;

-- Subqueries

SELECT first\_name, last\_name

FROM employees

WHERE department\_id IN (SELECT department\_id FROM departments WHERE department\_name = 'IT');

-- CTE (Common Table Expressions)

WITH EmployeeCTE AS (

    SELECT first\_name, last\_name

    FROM employees

    WHERE department\_id = 1

)

SELECT \* FROM EmployeeCTE;

Some more…

-- DISTINCT

SELECT DISTINCT department\_id FROM employees;

-- Retrieves unique department IDs from the employees table, eliminating duplicates.

-- LIKE Operator

SELECT \* FROM employees WHERE first\_name LIKE 'J%';

-- Retrieves employees whose first names start with the letter 'J'.

-- BETWEEN Operator

SELECT \* FROM employees WHERE salary BETWEEN 50000 AND 80000;

-- Retrieves employees with salaries between $50,000 and $80,000.

-- IN Operator

SELECT \* FROM employees WHERE department\_id IN (1, 2, 3);

-- Retrieves employees from departments with IDs 1, 2, or 3.

-- NOT Operator

SELECT \* FROM employees WHERE department\_id NOT IN (4, 5);

-- Retrieves employees not belonging to departments with IDs 4 or 5.

-- COUNT Function

SELECT COUNT(\*) FROM employees;

-- Counts the total number of records in the employees table.

-- MAX and MIN Functions

SELECT MAX(salary) AS max\_salary, MIN(salary) AS min\_salary FROM employees;

-- Finds the maximum and minimum salaries from the employees table.

-- AVG Function

SELECT AVG(salary) AS avg\_salary FROM employees;

-- Calculates the average salary of employees.

-- SUM Function

SELECT SUM(salary) AS total\_salary FROM employees;

-- Calculates the total salary of all employees.

-- GROUP BY with HAVING

SELECT department\_id, COUNT(\*) as employee\_count

FROM employees

GROUP BY department\_id

HAVING COUNT(\*) > 10;

-- Groups employees by department and filters departments with more than 10 employees.

-- JOIN with Multiple Tables

SELECT employees.first\_name, departments.department\_name, locations.city

FROM employees

INNER JOIN departments ON employees.department\_id = departments.department\_id

INNER JOIN locations ON departments.location\_id = locations.location\_id;

-- Joins employees, departments, and locations tables to retrieve employee names, department names, and city locations.

-- UNION Operator

SELECT first\_name, last\_name FROM employees

UNION

SELECT first\_name, last\_name FROM managers;

-- Combines unique records from the employees and managers tables.

-- EXISTS Operator

SELECT first\_name, last\_name

FROM employees e

WHERE EXISTS (

    SELECT 1 FROM managers m WHERE e.employee\_id = m.employee\_id

);

-- Retrieves employees who are also managers based on the existence of matching IDs in the managers table.

-- CASE Statement

SELECT first\_name,

       CASE

           WHEN salary > 70000 THEN 'High'

           WHEN salary BETWEEN 50000 AND 70000 THEN 'Medium'

           ELSE 'Low'

       END AS salary\_category

FROM employees;

-- Categorizes employees based on salary ranges as High, Medium, or Low.

4. What Is The SQL Query Used To Find The 2nd /

3rd / nth Highest Salary

select salary from employee

order by salary limit 2 offset 1;

5. What Is The SQL Query Used To Find All Employees

Who Also Hold The Managerial Position?

select e1.name as employeename, e2.name as managername

from employee e1

join employee e2 on e2.id = e1.manager\_id;

**Inner join**

## SQL INNER JOIN Syntax

SELECT *column\_name(s)*  
FROM *table1*  
INNER JOIN *table2*ON *table1.column\_name*=*table2.column\_name*;

Ex-1)

SELECT ProductID, ProductName, CategoryName  
FROM Products  
INNER JOIN Categories ON Products.CategoryID = Categories.CategoryID;

|  |  |  |
| --- | --- | --- |
| **ProductID** | **ProductName** | **CategoryName** |
| 39 | Chartreuse verte | Beverages |
| 2 | Chang | Beverages |
| 24 | Guaraná Fantástica | Beverages |

SELECT \* FROM Employee

INNER JOIN Employee\_new ON Employee.ID = Employee\_new.ID;

6. What Is The SQL Query Used To Find The Names Of

The Employees That Begin With ‘E’ & ‘J’ ?

select name from employee

where name like 'E%' or name like 'J%';

*To Fetch The Even Number Records:-*

**SELECT \* FROM Employee WHERE id%2=0;**