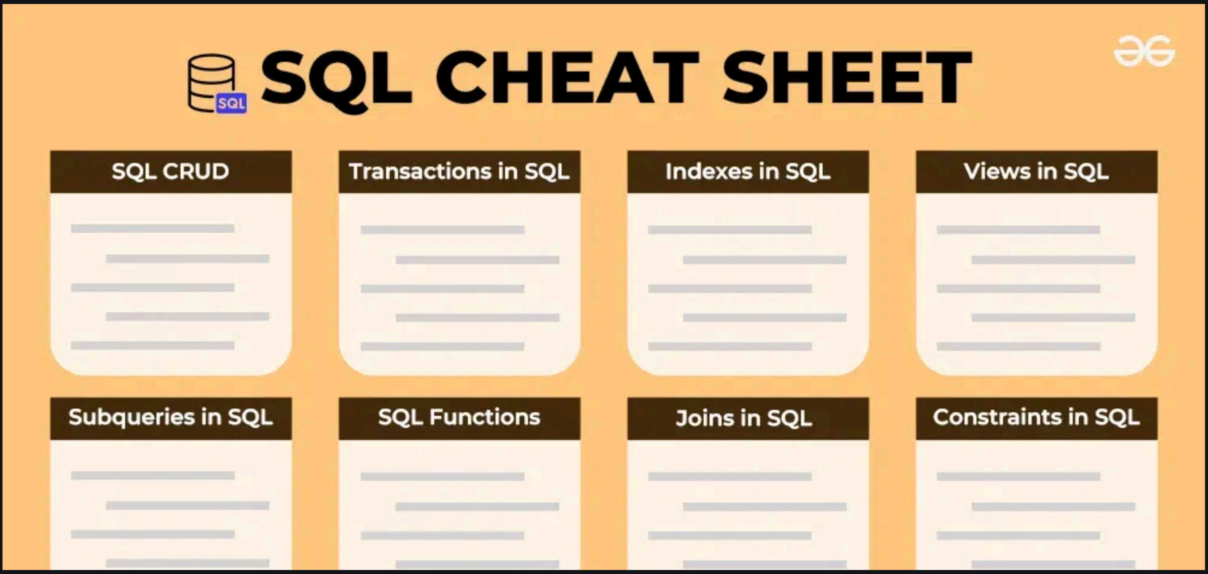
SQL Cheat Sheet (Basic to Advanced)

Last Updated : 02 Sep, 2024

Creating and **managing** databases in **SQL** involves various commands and concepts that handle the **structuring**, **querying**, and **manipulation** of data. In this guide, we will see a comprehensive **cheat sheet** for essential SQL operations, offering a practical reference for tasks ranging from database creation to advanced data handling techniques.

It includes fundamental SQL commands like **`CREATE DATABASE`** and **`DROP DATABASE`**, data manipulation commands such as `INSERT INTO` and `UPDATE`, as well as querying techniques using **`SELECT`**, **`WHERE`** and **aggregate functions**.



SQL Cheat Sheet

Table of Content

Create a Database in SQL

Explore this section to get hands on all the cheat sheet that help you in order to create a database in SQL.

1. CREATE DATABASE: Create a New Database

```
CREATE DATABASE company;
```

This command creates a new database named "**company**."

2. USE: Select a Specific Database to Work With

```
USE company;
```

This command selects the database named "**company**" for further operations.

3. ALTER DATABASE: Modify a Database's Attributes

```
ALTER DATABASE database_name
```

4. DROP DATABASE: Delete an Existing Database

```
DROP DATABASE company;
```

This command deletes the database named "company" and all its associated data.

Download SQL Cheat Sheet PDF

Creating Data in SQL

Here in this SQL cheat sheet we have listed down all the cheat sheet that help to create, insert, alter data in table.

5. CREATE: Create a New Table, Database or Index

```
CREATE TABLE employees (
  employee_id INT PRIMARY KEY,
  first_name VARCHAR(50),
  last_name VARCHAR(50),
  department VARCHAR(50),
  salary DECIMAL(10, 2)
);
```

This command creates a table named "employees" with columns for employee ID, first name, last name, department, and salary. The employee_id column is set as the primary key.

6. INSERT INTO: Add New Records To A Table

```
INSERT INTO employees (employee_id, first_name, last_name,
department, salary)
VALUES
  (1, 'John', 'Doe', 'HR', 50000.00),
  (2, 'Jane', 'Smith', 'IT', 60000.00),
  (3, 'Alice', 'Johnson', 'Finance', 55000.00),
  (4, 'Bob', 'Williams', 'IT', 62000.00),
  (5, 'Emily', 'Brown', 'HR', 48000.00);
```

This command inserts sample data into the "employees" table with values for **employee ID, first name, last name, department, and salary.**

7. ALTER TABLE: Modify An Existing Table's Structure

```
ALTER TABLE employees
ADD COLUMN new_column INT;
```

This command adds a new column named "new_column" of integer type to the existing "employees" table.

8. DROP TABLE: Delete A Table And Its Data

```
DROP TABLE employees;
```

This command deletes the entire "**employees**" table along with all its data.

Reading/Querying Data in SQL

Explore this section to get the cheat sheet on how to use select, distinct and other querying data in SQL.

9. SELECT: Retrieve Data From One Or More Tables

```
SELECT * FROM employees;
```

This query will retrieve all columns from the employees table.

10. DISTINCT: Select Unique Values From A Column

```
SELECT DISTINCT department FROM employees;
```

This query will return unique department names from the employees table.

11. WHERE: Filter Rows Based On Specified Conditions

```
SELECT * FROM employees WHERE salary > 55000.00;
```

This query will return employees whose salary is greater than 55000.00.

12. LIMIT: Limit The Number Of Rows Returned In The Result Set

```
SELECT * FROM employees LIMIT 3;
```

This query will limit the result set to the first 3 rows.

13. OFFSET: Skip A Specified Number Of Rows Before Returning The Result Set

```
SELECT * FROM employees OFFSET 2;
```

This query will skip the first 2 rows and return the rest.

14. FETCH: Retrieve A Specified Number Of Rows From The Result Set

```
SELECT * FROM employees FETCH FIRST 3 ROWS ONLY;
```

This query will fetch the first 3 rows from the result set.

15. CASE: Perform Conditional Logic In A Query

```
SELECT
  first_name,
  last_name,
  CASE
    WHEN salary > 55000 THEN 'High'
    WHEN salary > 50000 THEN 'Medium'
    ELSE 'Low'
  END AS salary_category
FROM employees;
```

This query will categorize employees based on their salary into 'High', 'Medium', or 'Low'.

Updating/Manipulating Data in SQL

Get a cheat sheet on how to update or manipulate data in SQL by exploring this section.

16. UPDATE: Modify Existing Records In A Table

```
UPDATE employees
SET salary = 55000.00
WHERE employee_id = 1;
```

This query will update the salary of the employee with employee_id 1 to 55000.00.

Deleting Data in SQL

17. DELETE: Remove Records From A Table

```
DELETE FROM employees
WHERE employee_id = 5;
```

This query will delete the record of the employee with employee_id 5 from the employees table.

Filtering Data in SQL

18. WHERE: Filter Rows Based On Specified Conditions

```
SELECT * FROM employees
WHERE department = 'IT';
```

This query will retrieve all employees who work in the IT department.

19. LIKE: Match A Pattern In A Column

```
SELECT * FROM employees
WHERE first_name LIKE 'J%';
```

This query will retrieve all employees whose first name starts with 'J'.

20. IN: Match Any Value In A List

```
SELECT * FROM employees
WHERE department IN ('HR', 'Finance');
```

This query will retrieve all employees who work in the HR or Finance departments.

21. BETWEEN: Match Values Within A Specified Range

```
SELECT * FROM employees
WHERE salary BETWEEN 50000 AND 60000;
```

This query will retrieve all employees whose salary is between 50000 and 60000.

22. IS NULL: Match NULL Values

```
SELECT * FROM employees
WHERE department IS NULL;
```

This query will retrieve all employees where the department is not assigned (NULL).

23. ORDER BY: Sort The Result Set

```
SELECT * FROM employees
ORDER BY salary DESC;
```

This query will retrieve all employees sorted by salary in descending order.

SQL Operator

Here in this section we have added a cheat sheet for SQL Operators. So, explore and learn how to use AND, OR, NOT and others oprtators.

24. AND: Combines Multiple Conditions In A WHERE Clause

```
SELECT * FROM employees
WHERE department = 'IT' AND salary > 60000;
```

This query will retrieve employees who work in the IT department and have a salary greater than 60000.

25. OR: Specifies Multiple Conditions Where Any One Of Them Should Be True

```
SELECT * FROM employees
WHERE department = 'HR' OR department = 'Finance';
```

This query will retrieve employees who work in either the HR or Finance department.

26. NOT: Negates A Condition

```
SELECT * FROM employees
WHERE NOT department = 'IT';
```

This query will retrieve employees who do not work in the IT department.

27. LIKE: Searches For A Specified Pattern In A Column

```
SELECT * FROM employees
WHERE first_name LIKE 'J%';
```

This query will retrieve employees whose first name starts with '**J**'.

28. IN: Checks If A Value Matches Any Value In

```
SELECT * FROM employees
WHERE department IN ('HR', 'Finance');
```

This query will retrieve employees who work in the HR or Finance departments.

29. BETWEEN: Selects Values Within a Specified Range

```
SELECT * FROM employees
WHERE salary BETWEEN 50000 AND 60000;
```

This query will retrieve employees whose salary is between 50000 and 60000.

30. IS NULL: Checks if a Value is NULL

```
SELECT * FROM employees
WHERE department IS NULL;
```

This query will retrieve employees where the department is not assigned (NULL).

31. ORDER BY: Sorts the Result Set in Ascending or Descending Order

```
SELECT * FROM employees
ORDER BY salary DESC;
```

This query will retrieve all employees sorted by salary in descending order.

32. GROUP BY: Groups Rows that have the Same Values into Summary Rows

```
SELECT department, COUNT(*) AS employee_count
FROM employees
GROUP BY department;
```

This query will group employees by department and count the number of employees in each department.

Aggregation Data in SQL

Get an hands in aggregation data in SQL. Here you will find cheat sheet for how to count numbers, sum of numbers and more.

33. COUNT: Count The Number Of Rows In A Result Set

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

This query will count the total number of employees.

34. SUM: Calculate The Sum Of Values In A Column

```
SELECT SUM(salary) FROM employees;
```

This query will calculate the total salary of all employees.

35. AVG: Calculate The Average Value Of A Column

```
SELECT AVG(salary) FROM employees;
```

This query will calculate the average salary of all employees.

36. MIN: Find the Minimum Value in a Column

```
SELECT MIN(salary) FROM employees;
```

This query will find the minimum salary among all employees.

37. MAX: Find the Maximum Value in a Column

```
SELECT MAX(salary) FROM employees;
```

This query will find the maximum salary among all employees.

38. GROUP BY: Group Rows Based on a Specified Column

```
SELECT department, COUNT(*) AS employee_count
FROM employees
GROUP BY department;
```

This query will group employees by department and count the number of employees in each department.

39. HAVING: Filter Groups Based on Specified Conditions

```
SELECT department, AVG(salary) AS avg_salary
FROM employees
GROUP BY department
HAVING AVG(salary) > 55000;
```

This query will calculate the average salary for each department and return only those departments where the average salary is greater than 55000.

Constraints in SQL

Constraints in SQL act as data quality guardrails, enforcing rules to ensure accuracy, consistency, and integrity within your database tables.

40. PRIMARY KEY: Uniquely Identifies Each Record in a Table

```
CREATE TABLE employees (
  employee_id INT PRIMARY KEY,
  first_name VARCHAR(50),
  last_name VARCHAR(50)
);
```

employee_id is designated as the primary key, ensuring that each employee record has a unique identifier.

41. FOREIGN KEY: Establishes a Relationship Between Two Tables

```
CREATE TABLE departments (
  department_id INT PRIMARY KEY,
  department_name VARCHAR(50)
);

CREATE TABLE employees (
  employee_id INT PRIMARY KEY,
  first_name VARCHAR(50),
  last_name VARCHAR(50),
  department_id INT,
  FOREIGN KEY (department_id) REFERENCES
departments(department_id)
);
```

department_id column in the employees table is a foreign key that references the department_id column in the departments table, establishing a relationship between the two tables.

42. UNIQUE: Ensures That All Values in a Column Are Unique

```
CREATE TABLE employees (
  employee_id INT PRIMARY KEY,
  email VARCHAR(100) UNIQUE
);
```

email column must contain unique values for each employee.

43. NOT NULL: Ensures That a Column Does Not Contain NULL Values

```
CREATE TABLE employees (
  employee_id INT PRIMARY KEY,
  first_name VARCHAR(50) NOT NULL,
  last_name VARCHAR(50) NOT NULL
);
```

first_name and **last_name** columns must have values and cannot be NULL.

44. CHECK: Specifies a Condition That Must Be Met for a Column's Value

```
CREATE TABLE employees (
  employee_id INT PRIMARY KEY,
  age INT CHECK (age >= 18)
);
```

age column must have a value of 18 or greater due to the CHECK constraint.

Joins in SQL

Explore different join types to seamlessly merge data from multiple tables in your SQL queries.

45. INNER JOIN: Retrieves Records That Have Matching Values in Both Tables

```
SELECT * FROM employees
INNER JOIN departments ON employees.department_id =
departments.department_id;
```

This query will retrieve records from both the employees and departments tables where there is a match on the department_id column.

46. LEFT JOIN: Retrieves All Records from the Left Table and the Matched Records from the Right Table

```
SELECT * FROM employees
LEFT JOIN departments ON employees.department_id =
departments.department_id;
```

This query will retrieve all records from the employees table and only the matching records from the departments table.

47. RIGHT JOIN: Retrieves All Records from the Right Table and the Matched Records from the Left Table

```
SELECT * FROM employees
RIGHT JOIN departments ON employees.department_id =
departments.department_id;
```

This query will retrieve all records from the departments table and only the matching records from the employees table.

48. FULL OUTER JOIN: Retrieves All Records When There Is a Match in Either the Left or Right Table

```
SELECT * FROM employees
FULL OUTER JOIN departments ON employees.department_id =
departments.department_id;
```

This query will retrieve all records from both the employees and departments tables, including unmatched records.

49. CROSS JOIN: Retrieves the Cartesian Product of the Two Tables

```
SELECT * FROM employees
CROSS JOIN departments;
```

This query will retrieve all possible combinations of records from the employees and departments tables.

50. SELF JOIN: Joins a Table to Itself

```
SELECT e1.first_name, e2.first_name
FROM employees e1, employees e2
WHERE e1.employee_id = e2.manager_id;
```

In this example, the employees table is joined to itself to find employees and their respective managers based on the manager_id column.

SQL Functions

In this section we have compiled SQL cheat sheet for SQL functions. It is used for common tasks like aggregation, filtering, date/time manipulation, and more!

51. Scalar Functions: Functions That Return a Single Value

```
SELECT UPPER(first_name) AS upper_case_name FROM employees;
```

This query uses the UPPER() scalar function to convert the first_name column values to uppercase.

52. Aggregate Functions: Functions That Operate on a Set of Values and Return a Single Value

```
SELECT AVG(salary) AS average_salary FROM employees;
```

This query uses the AVG() aggregate function to calculate the average salary of all employees.

53. String Functions: Functions That Manipulate String Values

```
SELECT CONCAT(first_name, ' ', last_name) AS full_name FROM
employees;
```

This query uses the CONCAT() string function to concatenate the first_name and last_name columns into a single column called full_name.

```
SELECT SUBSTR(first_name, 1, 3) AS short_name FROM employees;
```

This query uses the SUBSTR() function to extract the first three characters of the first_name column for each employee. The result is displayed in a new column called short_name.

```
SELECT INSERT(full_name, 6, 0, 'Amazing ') AS modified_name
FROM (SELECT CONCAT(first_name, ' ', last_name) AS full_name FROM
employees) AS employee_names;
```

This query first concatenates the first_name and last_name columns into a single column called full_name. Then, it uses the INSERT() function to insert the string 'Amazing ' at the 6th position of the full_name column for each employee. The modified names are displayed in a new column called modified_name.

54. Date and Time Functions: Functions That Operate on Date and Time Values

```
SELECT CURRENT_DATE AS current_date FROM dual;
```

This query uses the CURRENT_DATE date function to retrieve the current date.

55. Mathematical Functions: Functions That Perform Mathematical Operations

```
SELECT SQRT(25) AS square_root FROM dual;
```

This query uses the SQRT() mathematical function to calculate the square root of 25.

Subqueries in SQL

This SQL cheat sheet explains how to nest queries for powerful data filtering and manipulation within a single statement.

56. Single-row Subquery: Returns One Row of Result

```
SELECT first_name, last_name
FROM employees
WHERE salary = (SELECT MAX(salary) FROM employees);
```

In this example, the subquery (SELECT MAX(salary) **FROM** employees) returns a single row containing the maximum salary, and it's used to filter employees who have the maximum salary.

57. Multiple-row Subquery: Returns Multiple Rows of Result

```
SELECT department_name
FROM departments
WHERE department_id IN (SELECT department_id FROM employees);
```

In this example, the subquery (SELECT department_id FROM employees) returns multiple rows containing department IDs, and it's used to filter department names based on those IDs.

58. Correlated Subquery: References a Column from the Outer Query

```
SELECT first_name, last_name
FROM employees e
WHERE salary > (SELECT AVG(salary) FROM employees WHERE
department = e.department);
```

In this example, the subquery (SELECT **AVG**(salary) FROM employees WHERE department = e.department) is correlated with the outer query by referencing the department column from the outer query. It calculates the average salary for each department and is used to filter employees whose salary is greater than the average salary of their respective department.

59. Nested Subquery: A Subquery Inside Another Subquery

```
SELECT first_name, last_name
FROM employees
WHERE department_id IN (
    SELECT department_id
    FROM departments
    WHERE department_name = 'IT'
);
```

In this example, the subquery (SELECT department_id FROM departments WHERE department_name = 'IT') is nested within the outer query. It retrieves the department ID for the IT department, which is then used in the outer query to filter employees belonging to the IT department.

Views in SQL

Here in this SQL cheat sheet unveils how to create virtual tables based on existing data for streamlined access.

60. CREATE VIEW: Create a Virtual Table Based on the Result of a SELECT Query

```
CREATE VIEW high_paid_employees AS
SELECT *
FROM employees
WHERE salary > 60000;
```

This query creates a views named high_paid_employees that contains all employees with a salary greater than 60000.

61. DROP VIEW: Delete a View

```
DROP VIEW IF EXISTS high_paid_employees;
```

This query drops the high_paid_employees view if it exists.

Indexes in SQL

Speed up your SQL queries with our Indexes Cheat Sheet! Learn how to create and optimize indexes to dramatically improve database performance.

62. CREATE INDEX: Create an Index on a Table

```
CREATE INDEX idx_department ON employees (department);
```

This query creates an index named idx_department on the department column of the employees table.

63. DROP INDEX: Remove an Index

```
DROP INDEX IF EXISTS idx_department;
```

This query drops the idx_department index if it exists.

Transactions in SQL

Learn how to manage groups of database operations as a single unit for reliable data updates.

64. BEGIN TRANSACTION: Start a New Transaction

```
BEGIN TRANSACTION;
```

This statement starts a new transaction.

65. COMMIT: Save Changes Made During the Current Transaction

```
COMMIT;
```

This statement saves all changes made during the current transaction.

66. ROLLBACK: Undo Changes Made During the Current Transaction

```
ROLLBACK;
```

This statement undoes all changes made during the current transaction.

Advanced Mixed Data in SQL

In the last we have complied all the imprtant queries under the one advanced SQL cheat sheet.

67. Stored Procedures: Precompiled SQL Statements That Can Be Executed with a Single Command

```
CREATE PROCEDURE get_employee_count()
BEGIN
  SELECT COUNT(*) FROM employees;
END;
```

This query creates a stored procedure named get_employee_count that returns the count of employees.

68. Triggers: Automatically Execute a Set of SQL Statements When a Specified Event Occurs

```
CREATE TRIGGER before_employee_insert
BEFORE INSERT ON employees
FOR EACH ROW
BEGIN
  SET NEW.creation_date = NOW();
END;
```

This query creates a trigger named before_employee_insert that sets the creation_date column to the current date and time before inserting a new employee record.

69. User-defined Functions (UDFs): Custom SQL Functions Created by Users to Perform Specific Tasks

```
CREATE FUNCTION calculate_bonus(salary DECIMAL) RETURNS DECIMAL
BEGIN
  RETURN salary * 0.1; -- 10% bonus
END;
```

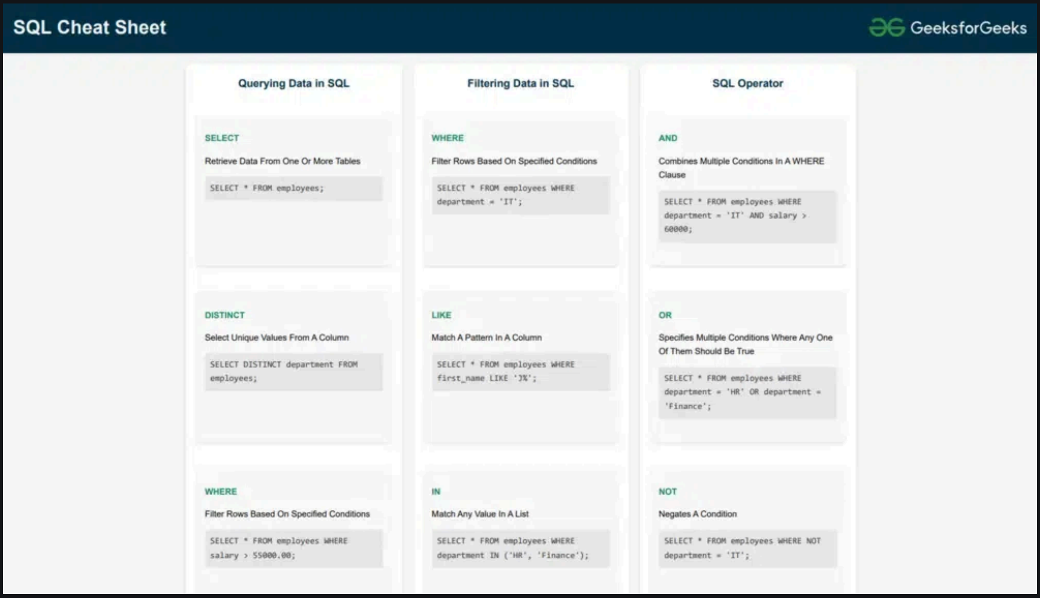
This query creates a user-defined function named calculate_bonus that calculates the bonus based on the salary.

70. Common Table Expressions (CTEs): Temporary Result Sets That Can Be Referenced Within a SELECT, INSERT, UPDATE, or DELETE Statement

```
WITH high_paid_employees AS (
  SELECT * FROM employees WHERE salary > 60000
)
SELECT * FROM high_paid_employees;
```

This query uses a common table expression named high_paid_employees to retrieve all employees with a salary greater than 60000.

SQL Cheat Sheet PDF



SQL Cheat Sheet

GeeksforGeeks

Querying Data in SQL

SELECT

Retrieve Data From One Or More Tables

```
SELECT * FROM employees;
```

DISTINCT

Select Unique Values From A Column

```
SELECT DISTINCT department FROM
employees;
```

WHERE

Filter Rows Based On Specified Conditions

```
SELECT * FROM employees WHERE
salary > 55000.00;
```

Filtering Data in SQL

WHERE

Filter Rows Based On Specified Conditions

```
SELECT * FROM employees WHERE
department = 'IT';
```

LIKE

Match A Pattern In A Column

```
SELECT * FROM employees WHERE
first_name LIKE 'J%';
```

IN

Match Any Value In A List

```
SELECT * FROM employees WHERE
department IN ('HR', 'Finance');
```

SQL Operator

AND

Combines Multiple Conditions In A WHERE Clause

```
SELECT * FROM employees WHERE
department = 'IT' AND salary >
60000;
```

OR

Specifies Multiple Conditions Where Any One Of Them Should Be True

```
SELECT * FROM employees WHERE
department = 'HR' OR department =
'Finance';
```

NOT

Negates A Condition

```
SELECT * FROM employees WHERE NOT
department = 'IT';
```

Download SQL CheatSheet

Conclusion

This SQL cheat sheet provide a wide range of commands and techniques essential for effective database management and data manipulation. By familiarizing yourself with these SQL operations, you can streamline your workflow, optimize performance and ensure data integrity across your database. Whether you are creating databases, modifying data, querying information, or implementing advanced features like triggers and stored procedures, this guide provides the necessary tools to handle various SQL tasks with confidence

"This course is very well structured and easy to learn. Anyone with zero experience of data science, python or ML can learn from this. This course makes things so easy that anybody can learn on their own. It's helping me a lot. Thanks for creating such a great course."- **Ayushi Jain**

| Placed at Microsoft

Now's your chance to unlock high-earning job opportunities as a Data Scientist! Join our Complete Machine Learning & Data Science Program and get a 360-degree learning experience mentored by industry experts.

Get hands on practice with **40+ Industry Projects, regular doubt solving sessions**, and much more. Register for the Program today!

Comment More info

Next Article

SQL Interview Questions

Similar Reads

PostgreSQL - Cheat Sheet : Basic to Advanced

PostgreSQL is a powerful, open-source object-relational database management system (ORDBMS). It is designed to help developers build...

5 min read

MongoDB Cheat Sheet (Basic to Advanced)

MongoDB is a powerful NoSQL database known for its flexible, document-oriented storage that is ideal for handling large-scale, comple...

12 min read

NumPy Cheat Sheet: Beginner to Advanced (PDF)

NumPy stands for Numerical Python. It is one of the most important foundational packages for numerical computing & data analysis in Pytho...

15+ min read

Tkinter Cheat Sheet

Tkinter, the standard GUI library for Python, empowers developers to effortlessly create visually appealing and interactive desktop application...

8 min read

ggplot2 Cheat Sheet

Welcome to the ultimate ggplot2 cheat sheet! This is your go-to resource for mastering R's powerful visualization package. With ggplot2, you can...

13 min read

Python OpenCV Cheat Sheet

The Python OpenCV Cheat Sheet is your complete guide to mastering computer vision and image processing using Python. It's designed to be...

15+ min read

GeeksforGeeks Master Sheet - List of all Cheat Sheets

In this Master Sheet, we’ll cover all the important cheat sheets like SDE Sheets, DSA Topics Sheets, HTML, CSS, JavaScript, React, Angular,...

10 min read

Ansible Cheat Sheet

Ansible is a powerful open-source automation tool that is meant for configuration management and application deployment. It works with...

10 min read

Pandas Cheat Sheet for Data Science in Python

Pandas is a powerful and versatile library that allows you to work with data in Python. It offers a range of features and functions that make data...

15+ min read

Git Cheat Sheet

Git Cheat Sheet is a comprehensive quick guide for learning Git concepts, from very basic to advanced levels. By this Git Cheat Sheet, our aim is to...

10 min read

Advertise with us
GFG Corporate Solution
Placement Training Program

Master System Design
Master CP
GeeksforGeeks Videos
Geeks Community

Languages

Python
Java
C++
PHP
GoLang
SQL
R Language
Android Tutorial

DSA

Data Structures
Algorithms
DSA for Beginners
Basic DSA Problems
DSA Roadmap
DSA Interview Questions
Competitive Programming

Data Science & ML

Data Science With Python
Data Science For Beginner
Machine Learning
ML Maths
Data Visualisation
Pandas
NumPy
NLP
Deep Learning

Web Technologies

HTML
CSS
JavaScript
TypeScript
ReactJS
NextJS
NodeJs
Bootstrap
Tailwind CSS

Python Tutorial

Python Programming Examples
Django Tutorial
Python Projects
Python Tkinter
Web Scraping
OpenCV Tutorial
Python Interview Question

Computer Science

GATE CS Notes
Operating Systems
Computer Network
Database Management System
Software Engineering
Digital Logic Design
Engineering Maths

DevOps

Git
AWS
Docker
Kubernetes
Azure
GCP
DevOps Roadmap

System Design

High Level Design
Low Level Design
UML Diagrams
Interview Guide
Design Patterns
OOAD
System Design Bootcamp
Interview Questions

School Subjects

Mathematics
Physics
Chemistry

Commerce

Accountancy
Business Studies
Economics