Sreenidhi Rajakrishnan
Automation Architect
@sreenidhirajakrishnan

SQL Commands Cheatsheet

Command	Sample Query	Explanation
SELECT	SELECT * FROM employees;	Fetches data from a table
INSERT	INSERT INTO employees VALUES (1,'John');	Inserts a new record
UPDATE	UPDATE employees SET name='Sam' WHERE id=1;	Updates existing records
DELETE	DELETE FROM employees WHERE id=1;	Deletes specific records
TRUNCATE	TRUNCATE TABLE employees;	Deletes all records permanently
CREATE	CREATE TABLE emp(id INT);	Creates database objects
ALTER	ALTER TABLE emp ADD salary INT;	Modifies table structure
DROP	DROP TABLE emp;	Deletes database objects
RENAME	RENAME TABLE emp TO employee;	Renames an object
WHERE	SELECT * FROM emp WHERE salary>50000;	Filters rows
DISTINCT	SELECT DISTINCT dept FROM emp;	Removes duplicates
ORDER BY	SELECT * FROM emp ORDER BY salary DESC;	Sorts result
GROUP BY	SELECT dept,COUNT(*) FROM emp GROUP BY dept;	Groups rows
HAVING	SELECT dept FROM emp GROUP BY dept HAVING COUNT(*)>5;	Filters groups
LIMIT	SELECT * FROM emp LIMIT 5;	Limits result rows
OFFSET	SELECT * FROM emp LIMIT 5 OFFSET 10;	Skips rows
INNER JOIN	SELECT * FROM A INNER JOIN B ON A.id=B.id;	Returns matching rows
LEFT JOIN	SELECT * FROM A LEFT JOIN B ON A.id=B.id;	All left + matches
RIGHT JOIN	SELECT * FROM A RIGHT JOIN B ON A.id=B.id;	All right + matches
FULL JOIN	SELECT * FROM A FULL JOIN B ON A.id=B.id;	All matching & non-matching
CROSS JOIN	SELECT * FROM A CROSS JOIN B;	Cartesian product
SELF JOIN	SELECT A.name,B.name FROM emp A, emp B;	Table joins itself
UNION	SELECT name FROM A UNION SELECT name FROM B;	Combines without duplicates
UNION ALL	SELECT name FROM A UNION ALL SELECT name FROM B;	Combines with duplicates
IN	SELECT * FROM emp WHERE dept IN ('HR','IT');	Matches list values
BETWEEN	SELECT * FROM emp WHERE salary BETWEEN 40k AND 80k;	Range filter
LIKE	SELECT * FROM emp WHERE name LIKE 'A%';	Pattern matching
IS NULL	SELECT * FROM emp WHERE bonus IS NULL;	Checks NULL
EXISTS	SELECT * FROM emp WHERE EXISTS (SELECT 1 FROM dept);	Checks subquery
COUNT	SELECT COUNT(*) FROM emp;	Counts rows
SUM	SELECT SUM(salary) FROM emp;	Adds values
AVG	SELECT AVG(salary) FROM emp;	Calculates average
MIN	SELECT MIN(salary) FROM emp;	Finds minimum
MAX	SELECT MAX(salary) FROM emp;	Finds maximum

Command	Sample Query	Explanation
ROUND	SELECT ROUND(AVG(salary),2) FROM emp;	Rounds values
UPPER	SELECT UPPER(name) FROM emp;	Converts to uppercase
LOWER	SELECT LOWER(name) FROM emp;	Converts to lowercase
SUBSTRING	SELECT SUBSTRING(name,1,3) FROM emp;	Extracts string
CONCAT	SELECT CONCAT(fname,lname) FROM emp;	Combines strings
NOW	SELECT NOW();	Current timestamp
CAST	SELECT CAST(salary AS CHAR) FROM emp;	Converts datatype
COALESCE	SELECT COALESCE(bonus,0) FROM emp;	Handles NULL
CASE	SELECT CASE WHEN salary>50k THEN 'High' END FROM emp;	Conditional logic
PRIMARY KEY	id INT PRIMARY KEY	Unique row identifier
FOREIGN KEY	dept_id INT REFERENCES dept(id)	Enforces relation
UNIQUE	email VARCHAR UNIQUE	Prevents duplicates
NOT NULL	name VARCHAR NOT NULL	Disallows NULL
CHECK	salary INT CHECK (salary>0)	Enforces condition
DEFAULT	status VARCHAR DEFAULT 'ACTIVE'	Sets default value
CREATE INDEX	CREATE INDEX idx_sal ON emp(salary);	Improves performance
DROP INDEX	DROP INDEX idx_sal;	Removes index
CREATE VIEW	CREATE VIEW emp_v AS SELECT * FROM emp;	Creates virtual table
DROP VIEW	DROP VIEW emp_v;	Deletes view
CREATE PROCEDURE	CREATE PROCEDURE getEmp()	Stored SQL logic
CALL	CALL getEmp();	Executes procedure
CREATE FUNCTION	CREATE FUNCTION bonus()	Returns a value
CREATE TRIGGER	CREATE TRIGGER trg BEFORE INSERT	Auto execution
TRANSACTION	START TRANSACTION;	Begins transaction
COMMIT	COMMIT;	Saves changes
ROLLBACK	ROLLBACK;	Reverts changes
SAVEPOINT	SAVEPOINT sp1;	Sets rollback point
GRANT	GRANT SELECT ON emp TO user;	Gives permission
REVOKE	REVOKE SELECT ON emp FROM user;	Removes permission
EXPLAIN	EXPLAIN SELECT * FROM emp;	Shows execution plan
MERGE	MERGE INTO emp USING emp2	Insert or update
WITH (CTE)	WITH temp AS (SELECT * FROM emp)	Temporary result
ROW_NUMBER	ROW_NUMBER() OVER(PARTITION BY dept)	Sequential numbering
RANK	RANK() OVER(ORDER BY salary)	Ranking with gaps
DENSE_RANK	DENSE_RANK() OVER(ORDER BY salary)	Ranking without gaps