Importance ■■■■■ most asked ■■■□□ often ■□□□□ rarely

SELECT queries

Write the clauses in this order; leave out the ones you don't need:

```
SELECT DISTINCT col, expr AS alias
FROM table
WHERE condition
GROUP BY col
HAVING group_condition
ORDER BY col DESC;
```

IN WHERE	EXAMPLE
Compare	`Marks >= 80` · `Class <> '12A'`
AND, OR, NOT	`Class='12A' AND Marks>80`
BETWEEN	`Marks BETWEEN 70 AND 90` (both ends included)
IN	`City IN ('Delhi','Pune')`
LIKE	`%` any number of characters, `_` exactly one
NULL	`Marks IS NULL` · `IS NOT NULL`

LIKE PATTERN	MATCHES
`'A%'`	starts with A
`'%a'`	ends with a
`'%an%'`	has "an" anywhere
`'_a%'`	a is the second letter
`'____'`	exactly four characters

ORDER BY is ascending by default; `DESC` applies only to the column it follows. **DISTINCT** removes duplicate rows. An **alias** (`AS`) renames only the output heading.

! Never `= NULL`: write **IS NULL**. A NULL row fails every comparison, so it is in neither `Marks > 80` nor `Marks <= 80`. Text and dates need quotes; numbers don't.

Aggregates, GROUP BY, HAVING

Table S, used for every output below:

ROLL	NAME	CLASS	MARKS
1	Aman	12A	91
2	Riya	12B	78
3	Kabir	12A	NULL
4	Sara	12B	85
5	Dev	12A	62

FUNCTION	RESULT	WHY
`COUNT(*)`	5	counts rows, NULLs too
`COUNT(Marks)`	4	skips the NULL
`SUM(Marks)`	316	
`AVG(Marks)`	79.0000	316 ÷ 4, not ÷ 5
`MAX` · `MIN`	91 · 62	
`COUNT(DISTINCT Class)`	2	12A, 12B

```
SELECT Class, COUNT(*), AVG(Marks)
FROM S GROUP BY Class;          -- 12A 3 76.5000
                                -- 12B 2 81.5000
SELECT Class, COUNT(*) FROM S
GROUP BY Class HAVING COUNT(*) > 2;   -- 12A 3
```

WHERE filters rows **before** grouping; **HAVING** filters groups **after**. "Class-wise", "gender-wise", "category-wise" → GROUP BY that column, and put it in SELECT too.

! Aggregates can't go in WHERE: `WHERE COUNT(*) > 5` must become **HAVING COUNT(*) > 5**, written after GROUP BY. All aggregates except COUNT(*) ignore NULLs.

Single-row functions

MATH	RESULT
`POWER(2,3)`	8
`ROUND(15.193,1)` · `ROUND(15.5)`	15.2 · 16
`ROUND(1567.8,-2)` · `ROUND(1567.8,-1)`	1600 · 1570
`MOD(29,6)`	5 (remainder)

TEXT	RESULT
`UCASE('sql')` / `UPPER` · `LCASE('MySQL')` / `LOWER`	SQL · mysql
`MID('Informatics',3,4)` = `SUBSTR`	form
`LENGTH('Hello World')`	11 (space counts)
`LEFT('Database',4)` · `RIGHT('Database',4)`	Data · base
`INSTR('Database','base')` · `INSTR('Database','z')`	5 · 0
`LTRIM` · `RTRIM` · `TRIM(' Hi ')`	'Hi ' · ' Hi' · 'Hi'

DATE, FOR '2026-03-14'	RESULT
`NOW()`	current date and time
`DATE('2026-03-14 10:30:00')`	2026-03-14
`MONTH` · `MONTHNAME`	3 · March
`DAY` · `DAYNAME`	14 · Saturday
`YEAR`	2026

! Positions start at **1**. INSTR gives **0** when not found. Nested calls work inside out: `ROUND(MOD(29,6)*2.456,1)` → 5 × 2.456 = 12.28 → **12.3**.

Relational terms and keys

TERM	MEANS
Relation	a table
Tuple	a row (record)
Attribute	a column (field)
Domain	the set of values a column may take
Degree	number of **columns**
Cardinality	number of **rows** (cardinality counts records)

KEY	MEANS
Candidate key	any column that can identify each row uniquely
Primary key	the candidate key chosen; unique, never NULL, one per table
Alternate key	candidate keys not chosen as primary
Foreign key	a column that refers to the primary key of another table; links tables

! 4 rows × 3 columns; add 2 rows, remove 1 column → degree **2**, cardinality **6**. "Keeps data integrity between tables" → **foreign key**.

DDL and DML commands

TYPE	COMMANDS	CHANGES
DDL (definition)	CREATE, ALTER, DROP	the structure
DML (manipulation)	INSERT, UPDATE, DELETE	the data
DQL (query)	SELECT	nothing; only reads

```
CREATE DATABASE School;  USE School;
CREATE TABLE S (Roll INT PRIMARY KEY,
  Name VARCHAR(20), Class CHAR(3), Marks INT);
ALTER TABLE S ADD City VARCHAR(15);
ALTER TABLE S DROP City;
ALTER TABLE S DROP PRIMARY KEY;
ALTER TABLE S ADD PRIMARY KEY (Roll);
INSERT INTO S VALUES (6,'Neha','12B',88);
UPDATE S SET Marks = Marks + 5 WHERE Roll = 2;
DELETE FROM S WHERE Marks IS NULL;
DROP TABLE S;      -- removes the whole table
```

Also: `SHOW DATABASES; SHOW TABLES; DESCRIBE S;` (or `DESC S;`) shows the structure.

	CHAR(N)	VARCHAR(N)
Length	fixed: always n characters	variable: up to n
'Hi' in (10)	uses 10, padded with spaces	uses 2

! ALTER changes the **degree**; INSERT and DELETE change the **cardinality**; UPDATE changes neither. Without WHERE, UPDATE and DELETE hit **every** row. DROP removes the table; DELETE only its rows.

Joins

- **Equi-join**: join on equal values of a common column; name the table for that column.

```
SELECT Name, GName FROM PLAYER, GAMES
WHERE PLAYER.GCode = GAMES.GCode;
```

Database concepts

A **database** is organised data; a **DBMS** is the software that manages it. Over a **file system** it removes **redundancy** (the same data stored many times) and **inconsistency** (copies that disagree). **MySQL** is an open-source **RDBMS**; **SQL** is the language it understands, and it is not case-sensitive for keywords.

What's the output?

On table S above. Cover the right column and answer; every result was run on MySQL.

QUERY	OUTPUT
`SELECT Name FROM S WHERE Name LIKE '%a';`	Riya, Sara
`SELECT Name FROM S WHERE Name LIKE '_a%';`	Kabir, Sara
`SELECT COUNT(*) FROM S WHERE Marks > 80;`	2
`SELECT MAX(Marks) - MIN(Marks) FROM S;`	29
`SELECT Class, MAX(Marks) FROM S GROUP BY Class;`	12A 91 · 12B 85
`SELECT Name FROM S WHERE Marks BETWEEN 78 AND 85 ORDER BY Marks DESC;`	Sara, Riya
`SELECT COUNT(*) FROM S WHERE Marks = NULL;`	0
`SELECT DISTINCT Class FROM S;`	12A, 12B
`SELECT UCASE(LEFT(Name,3)) FROM S WHERE Roll = 4;`	SAR
`SELECT LENGTH(Name), INSTR(Name,'a') FROM S WHERE Roll = 3;`	5 · 2
`SELECT ROUND(AVG(Marks)) FROM S;`	79

Find the error

WRONG	RIGHT
`WHERE COUNT(*) > 2`	`HAVING COUNT(*) > 2`
`WHERE Marks = NULL`	`WHERE Marks IS NULL`
`DELETE * FROM S;`	`DELETE FROM S;`
`WHERE Name = Riya`	`WHERE Name = 'Riya'`
`... ORDER BY Marks WHERE Class='12A'`	WHERE before ORDER BY
`... HAVING ... GROUP BY Class`	GROUP BY before HAVING
`WHERE Marks BETWEEN 70, 90`	`BETWEEN 70 AND 90`
`UPDATE S Marks = 90 ...`	`UPDATE S SET Marks = 90 ...`

Expand it

SQL	Structured Query Language
DBMS	Database Management System
RDBMS	Relational DBMS
DDL · DML	Data Definition · Data Manipulation Language
DQL	Data Query Language