Kanak

SQL

```
SELECT * FROM table;
SELECT * FROM table
WHERE condition;
SELECT * FROM table
ORDER BY column;
```

SQL KEYWORDS

KEYWORDS	DESCRIPTION
ADD	Adds a column in an existing table, or adds a constraint on a table (some systems only support this after adding a column)
ADD CONSTRAINT	Adds a constraint after a table is already created, such as a primary key, foreign key, check, or unique constraint
ALTER TABLE	Changes the structure of an existing table, such as adding, dropping, or modifying columns ALTER TABLE table_name ALTER TABLE table_name ADD column_name datatype ALTER TABLE table_name DROP COLUMN column_name ALTER TABLE table_name ALTER COLUMN column_name datatype
ALTER COLUMN	Changes the data type or other properties of a column in a table ALTER TABLE table_name ALTER COLUMN column_name datatype
ALL	Returns true if all subqueries meet the condition SELECT * FROM table WHERE column > ALL (SELECT column FROM table WHERE condition);
AND	Combines multiple conditions in a WHERE or HAVING clause, all must be true SELECT * FROM table WHERE condition1 AND condition2;
ANY	Returns true if any subquery meet the condition SELECT * FROM table WHERE column > ANY (SELECT column FROM table WHERE condition);
AS	Assigns a temporary name (alias) to a table or column SELECT column AS alias_name FROM table;
ASC	Sorts results in ascending order (default if not specified) SELECT * FROM table ORDER BY column ASC;
BETWEEN	Filters results within a range (inclusive) SELECT * FROM table WHERE column BETWEEN value1 AND value2;
CASE	Adds conditional logic (like if-else) to queries SELECT column, CASE WHEN condition1 THEN result1 WHERE condition2 THEN result2 ELSE result3 END FROM table;
CHECK	Ensures values in a column meet a specified condition CREATE TABLE table_name (column datatype CHECK (condition)); ALTER TABLE table_name ADD CHECK (condition);
CREATE DATABASE	Creates a new database CREATE DATABASE database_name;
CREATE INDEX	Creates an index on a table to speed up queries CREATE INDEX index_name ON table (column);
CREATE TABLE	Creates a new table in the database CREATE TABLE table_name (column1 datatype, column2 datatype, ...);
DEFAULT	Sets a default value for a column when no value is specified CREATE TABLE table_name (column datatype DEFAULT value); ALTER TABLE table_name ADD column datatype DEFAULT value;
DELETE	Removes rows from a table DELETE FROM table WHERE condition;
DESC	Sorts results in descending order SELECT * FROM table ORDER BY column DESC;
DISTINCT	Removes duplicate rows from results SELECT DISTINCT column FROM table;
DROP	Deletes a table, database, or index from the database DROP TABLE table_name; DROP DATABASE database_name; DROP INDEX index_name;
DROP COLUMN	Deletes a column from an existing table ALTER TABLE table_name DROP COLUMN column_name;
DROP CONSTRAINT	Removes a constraint from a table ALTER TABLE table_name DROP CONSTRAINT constraint_name;
EXISTS	Checks if a subquery returns any rows SELECT * FROM table WHERE EXISTS (SELECT 1 FROM table WHERE condition);
FROM	Specifies the table to select data from SELECT column FROM table;
GROUP BY	Groups rows with the same values for aggregation SELECT column, COUNT(*) FROM table GROUP BY column;
HAVING	Filters groups after GROUP BY (like WHERE but for groups) SELECT column, COUNT(*) FROM table GROUP BY column HAVING COUNT(*) > 1;
IN	Checks if a value matches any in a list or subquery SELECT * FROM table WHERE column IN (value1, value2, value3);
INDEX	Creates an index on a table to speed up queries CREATE INDEX index_name ON table (column);
INNER JOIN	Returns only matching rows from both tables SELECT * FROM table1 INNER JOIN table2 ON table1.column = table2.column;
INSERT INTO	Adds new rows to a table INSERT INTO table (column1, column2) VALUES (value1, value2);
IS NULL	Checks for null values SELECT * FROM table WHERE column IS NULL;
IS NOT NULL	Checks for non-null values SELECT * FROM table WHERE column IS NOT NULL;
JOIN	Combines rows from two or more tables based on a related column SELECT * FROM table1 JOIN table2 ON table1.column = table2.column;
LEFT JOIN	Returns all rows from the left table, plus matching rows from the right table SELECT * FROM table1 LEFT JOIN table2 ON table1.column = table2.column;
LIKE	Filters results using pattern matching with wildcards (% = any characters, _ = one character) SELECT * FROM table WHERE column LIKE 'pattern%';
LIMIT	Limits the number of rows returned SELECT * FROM table LIMIT 10;
NOT	Reverses a condition (returns rows where the condition is false) SELECT * FROM table WHERE NOT condition;
NOT IN	Checks if a value does not match any in a list or subquery SELECT * FROM table WHERE column NOT IN (value1, value2, value3);
OFFSET	Skips a specified number of rows before returning results SELECT * FROM table LIMIT 10 OFFSET 20;
ON	Specifies the join condition for combining tables SELECT * FROM table1 JOIN table2 ON table1.column = table2.column;
OR	Combines multiple conditions in a WHERE or HAVING clause, at least one must be true SELECT * FROM table WHERE condition1 OR condition2;
ORDER BY	Sorts results by one or more columns SELECT * FROM table ORDER BY column1, column2 DESC;
OUTER JOIN	Returns all rows from both tables, even if no match exists SELECT * FROM table1 OUTER JOIN table2 ON table1.column = table2.column;
PRIMARY KEY	Uniquely identifies each row in a table (combines NOT NULL and UNIQUE) CREATE TABLE table_name (column datatype PRIMARY KEY); ALTER TABLE table_name ADD PRIMARY KEY (column);
RIGHT JOIN	Returns all rows from the right table, plus matching rows from the left table SELECT * FROM table1 RIGHT JOIN table2 ON table1.column = table2.column;
SELECT	Retrieves data from a database SELECT column1, column2 FROM table;
SELECT DISTINCT	Returns unique values, removing duplicate rows SELECT DISTINCT column FROM table;
SELECT TOP	Limits the number of rows returned (SQL Server/MS Access syntax) SELECT TOP 10 * FROM table;
SET	Specifies the columns and values to be updated in an UPDATE statement UPDATE table SET column1 = value1, column2 = value2 WHERE condition;
SUM	Calculates the total of a numeric column SELECT SUM(column) FROM table;
TOP	Limits the number of rows returned (SQL Server/MS Access syntax) SELECT TOP 10 * FROM table;
TRUNCATE TABLE	Deletes all rows in a table but keeps the table structure (faster than DELETE) TRUNCATE TABLE table_name;
UNION	Combines results from two queries, removing duplicate rows SELECT column FROM table1 UNION SELECT column FROM table2;
UNION ALL	Combines results from two queries, keeping all rows including duplicates SELECT column FROM table1 UNION ALL SELECT column FROM table2;
UNIQUE	Ensures all values in a column are different CREATE TABLE table_name (column datatype UNIQUE); ALTER TABLE table_name ADD UNIQUE (column);
UPDATE	Modifies existing rows in a table UPDATE table SET column1 = value1, column2 = value2 WHERE condition;
VALUES	Specifies the values to be inserted in an INSERT INTO statement INSERT INTO table (column1, column2) VALUES (value1, value2);
WHERE	Filters rows based on a condition SELECT * FROM table WHERE column = value;

Kanak Infosystems LLP

COMMENTS

SINGLE LINE COMMENTS

```
-- This is a single line comment
SELECT * FROM table;
```

MULTILINE COMMENTS

```
/* This is a multiline comment
that can span
multiple lines */
SELECT * FROM table;
```

MySQL Data Types

Click to Zoom

MySQL Data Types

NUMERIC DATA TYPES

DATA TYPE	DESCRIPTION
TINYINT(M)	Stores very small integers (-128 to 127 signed, 0 to 255 unsigned)
SMALLINT(M)	Stores small integers (-32,768 to 32,767 signed, 0 to 65,535 unsigned)
MEDIUMINT(M)	Stores medium integers (-8,388,608 to 8,388,607 signed, 0 to 16,777,215 unsigned)
INT(M)	Stores standard integers (-2,147,483,648 to 2,147,483,647 signed, 0 to 4,294,967,295 unsigned)
BIGINT(M)	Stores large integers (-9,223,372,036,854,775,808 to 9,223,372,036,854,775,807 signed, 0 to 18,446,744,073,709,551,615 unsigned)
DECIMAL(M,D)	Stores exact decimal numbers with M total digits and D decimal places (e.g., DECIMAL(10,2) for 10 digits with 2 decimal places)
FLOAT(M,D)	Stores approximate floating-point numbers with M total digits and D decimal places (e.g., FLOAT(8,2))
DOUBLE(M,D)	Stores approximate floating-point numbers with M total digits and D decimal places (e.g., DOUBLE(16,4))
BIT(M)	Stores bit values (1 to 64 bits, default is 1 if not specified)
BOOL(EAN)	Alias for TINYINT(1), commonly used for true/false values (0 = false, 1 = true)

STRING DATA TYPES

DATA TYPE	DESCRIPTION
CHAR(M)	Stores fixed-length strings with M characters (padded with spaces if shorter)
VARCHAR(M)	Stores variable-length strings up to M characters (more efficient than CHAR for varying lengths)
TINYTEXT	Stores very small text (up to 255 characters)
TEXT	Stores standard text (up to 65,535 characters)
MEDIUMTEXT	Stores medium text (up to 16,777,215 characters)
LONGTEXT	Stores large text (up to 4,294,967,295 characters)
ENUM('val1', 'val2', ...)	Stores one value from a predefined list of values (e.g., ENUM('small', 'medium', 'large'))
SET('val1', 'val2', ...)	Stores zero or more values from a predefined list (e.g., SET('red', 'blue', 'green'))
BINARY(M)	Stores fixed-length binary data with M bytes
VARBINARY(M)	Stores variable-length binary data up to M bytes
TINYBLOB	Stores very small binary large object (up to 255 bytes)
BLOB	Stores standard binary large object (up to 65,535 bytes)
MEDIUMBLOB	Stores medium binary large object (up to 16,777,215 bytes)
LONGBLOB	Stores large binary large object (up to 4,294,967,295 bytes)

DATE AND TIME DATA TYPES

DATA TYPE	DESCRIPTION
DATE	Stores date in YYYY-MM-DD format (e.g., 2025-12-31)
DATETIME(fsp)	Stores date and time in YYYY-MM-DD HH:MM:SS format (e.g., 2025-12-31 23:59:59). Optional fsp (fractional seconds precision) for microseconds (0-6)
TIMESTAMP(fsp)	Stores date and time as seconds since 1970-01-01 00:00:00 UTC (Unix timestamp). Optional fsp for microseconds (0-6). Automatically updates when row is modified
TIME(fsp)	Stores time in HH:MM:SS format (e.g., 14:30:00). Optional fsp for microseconds (0-6)
YEAR	Stores year in 2-digit or 4-digit format (e.g., 2025)

Kanak Infosystems LLP

OPERATORS

ARITHMETIC OPERATORS

OPERATOR	DESCRIPTION
+	Addition
-	Subtract
*	Multiply
/	Divide
%	Modulo

COMPARISON OPERATORS

OPERATOR	DESCRIPTION
=	Equal to
!= or <>	Not equal to
>	Greater than
<	Less than
>=	Greater than or equal to
<=	Less than or equal to
IN	Checks if value is in a list
BETWEEN	Checks if value is between two values
LIKE	Pattern matching with wildcards (% = any characters, _ = one character)
IS NULL	Checks if value is null
IS NOT NULL	Checks if value is not null

LOGICAL OPERATORS

OPERATOR	DESCRIPTION
AND	Logical AND
OR	Logical OR
NOT	Logical NOT

BITWISE OPERATORS

OPERATOR	DESCRIPTION
&	Bitwise AND
|	Bitwise OR
^	Bitwise XOR
~	Bitwise NOT
<<	Bitwise shift left
>>	Bitwise shift right

Kanak Infosystems LLP

FUNCTIONS

STRING FUNCTIONS

DATA TYPE	DESCRIPTION
ASCII(char)	Returns the ASCII value for the specified character
CHAR_LENGTH(str)	Returns the length of the string in characters
CONCAT(str1, str2, ...)	Concatenates (joins) together two or more strings
CONCAT_WS(separator, str1, str2, ...)	Concatenates with separator (WS = With Separator)
FIELD(str, str1, str2, str3, ...)	Returns the index position of the first argument within the remaining arguments (returns 0 if not found)
FIND_IN_SET(str, strlist)	Returns the position of a string within a comma-separated list of strings
FORMAT(num, decimal places)	Formats a number with specified decimal places (e.g., FORMAT(12345.678, 2) → '12,345.68')
INSERT(str, pos, len, newstr)	Inserts a string into another string at position pos, replacing len characters
INSTR(str, substr)	Returns the position of the first occurrence of substr in str (case-insensitive)
LCASE(str)	Converts string to lowercase (alias for LOWER)
LEFT(str, len)	Returns the leftmost len characters from the string
LENGTH(str)	Returns the length of the string in bytes
LOCATE(substr, str, pos)	Returns the position of substr in str, starting search at position pos
LOWER(str)	Converts string to lowercase
LPAD(str, len, padstr)	Pads the left side of str with padstr to a total length of len characters
LTRIM(str)	Removes leading spaces from the string
MID(str, pos, len)	Returns len characters from str, starting at position pos (alias for SUBSTR)
POSITION(substr IN str)	Returns the position of substr in str (alternative to INSTR)
REPEAT(str, count)	Repeats the string str count times
REPLACE(str, from_str, to_str)	Replaces all occurrences of from_str with to_str in str
REVERSE(str)	Reverses the characters in the string
RIGHT(str, len)	Returns the rightmost len characters from the string
RPAD(str, len, padstr)	Pads the right side of str with padstr to a total length of len characters
RTRIM(str)	Removes trailing spaces from the string
SUBSTR(str, pos, len)	Returns len characters from str, starting at position pos (alias for MID)
SUBSTRING(str, pos, len)	Returns len characters from str, starting at position pos (alternative to SUBSTR)
TRIM([remstr FROM] str)	Removes leading and trailing spaces (or remstr if specified) from the string
UCASE(str)	Converts string to uppercase (alias for UPPER)
UPPER(str)	Converts string to uppercase

NUMERIC FUNCTIONS

DATA TYPE	DESCRIPTION
ABS(num)	Returns the absolute value of the number
ACOS(num)	Returns the arccosine (inverse cosine) of the number
ASIN(num)	Returns the arcsine (inverse sine) of the number
ATAN(num)	Returns the arctangent of the number
ATAN2(y, x)	Returns the arctangent of the two variables y and x
CEIL(num)	Rounds up to the nearest integer (ceiling)
CEILING(num)	Rounds up to the nearest integer (alias for CEIL)
COS(num)	Returns the cosine of the number (in radians)
COT(num)	Returns the cotangent of the number
CRC32(str)	Returns the cyclic redundancy check 32-bit checksum of the string
DEGREES(num)	Converts radians to degrees
EXP(num)	Returns e (2.718...) raised to the power of the number
FLOOR(num)	Rounds down to the nearest integer
GREATEST(num1, num2, ...)	Returns the largest value among the numbers
LN(num)	Returns the natural logarithm (base e) of the number
LOG(num)	Returns the natural logarithm of the number
LOG10(num)	Returns the base-10 logarithm of the number
LOG2(num)	Returns the base-2 logarithm of the number
MOD(num, divisor)	Returns the remainder of num divided by divisor (modulo operation)
PI()	Returns π (3.141592653589793)
POW(num, power)	Returns num raised to the power power
POWER(num, power)	Returns num raised to the power power (alias for POW)
RADIANS(num)	Converts degrees to radians
RAND()	Returns a random floating point value between 0 and 1
ROUND(num, decimal places)	Rounds the number to decimal places decimal places (e.g., ROUND(3.14159, 2) → 3.14)
SIGN(num)	Returns the sign of the number (-1 for negative, 0 for zero, 1 for positive)
SIN(num)	Returns the sine of the number (in radians)
SQRT(num)	Returns the square root of the number
TAN(num)	Returns the tangent of the number (in radians)
TRUNCATE(num, decimal places)	Truncates (cuts off) the number to decimal places decimal places (e.g., TRUNCATE(3.14159, 2) → 3.14)

DATE AND TIME FUNCTIONS

DATA TYPE	DESCRIPTION
ADDDATE(date, INTERVAL value unit)	Adds a specified time interval to a date (alias for DATE_ADD)
ADDTIME(time, time2)	Adds time2 to time
CURDATE()	Returns the current date in YYYY-MM-DD format
CURRENT_DATE()	Returns the current date (alias for CURDATE)
CURRENT_TIME()	Returns the current time in HH:MM:SS format
CURRENT_TIMESTAMP()	Returns the current date and time in YYYY-MM-DD HH:MM:SS format
CURTIME()	Returns the current time
DATE(date)	Extracts the date part from a date or datetime expression
DATEDIFF(date1, date2)	Returns the number of days between date1 and date2
DATE_ADD(date, INTERVAL value unit)	Adds a specified time interval to a date
DATE_FORMAT(date, format)	Formats a date according to the specified format string
DAY(date)	Returns the day of the month (1-31)
DAYNAME(date)	Returns the name of the weekday (Monday, Tuesday, ...)
DAYOFMONTH(date)	Returns the day of the month (1-31)
DAYOFWEEK(date)	Returns the weekday index (1 = Sunday, 2 = Monday, ..., 7 = Saturday)
DAYOFYEAR(date)	Returns the day of the year (1-366)
EXTRACT(unit FROM date)	Extracts a specified unit from a date (e.g., EXTRACT(YEAR FROM date))
FROM_DAYS(N)	Returns a date from the number of days since 0000-00-00
HOUR(time)	Returns the hour (0-23)
LAST_DAY(date)	Returns the last day of the month for the given date
MAKEDATE(year, day of year)	Creates a date from the year and day of year
MAKETIME(hour, minute, second)	Creates a time from hour, minute, second
MICROSECOND(time)	Returns the microseconds (0-999999)
MINUTE(time)	Returns the minute (0-59)
MONTH(date)	Returns the month (1-12)
MONTHNAME(date)	Returns the name of the month (January, February, ...)
NOW()	Returns the current date and time in YYYY-MM-DD HH:MM:SS format
PERIOD_ADD(period, months)	Adds months to a period in YYMM or YYYYMM format
PERIOD_DIFF(period1, period2)	Returns the difference between two periods in months
QUARTER(date)	Returns the quarter (1-4)
SECOND(time)	Returns the second (0-59)
SEC_TO_TIME(seconds)	Converts seconds to time format
STR_TO_DATE(str, format)	Converts a string to a date using the specified format
SUBDATE(date, INTERVAL value unit)	Subtracts a specified time interval from a date
SUBTIME(time, time2)	Subtracts time2 from time
SYSDATE()	Returns the current date and time (similar to NOW() but evaluated at execution time)
TIME(expr)	Extracts the time part from a date or datetime expression
TIMEDIFF(time1, time2)	Returns the difference between two times
TIMESTAMP(date, time)	Combines a date and time into a datetime value
TIMESTAMPADD(unit, interval, datetime)	Adds a specified interval to a datetime
TIMESTAMPDIFF(unit, datetime1, datetime2)	Returns the difference between two datetimes in specified units
TIME_FORMAT(time, format)	Formats a time according to the specified format string
TIME_TO_SEC(time)	Converts time to seconds
TIMEDIFF(time1, time2)	Returns the difference between two times
TO_DAYS(date)	Returns the number of days since 0000-00-00
WEEK(date)	Returns the week number (0-53)
WEEKDAY(date)	Returns the weekday index (0 = Monday, 1 = Tuesday, ..., 6 = Sunday)
WEEKOFYEAR(date)	Returns the week of the year (1-53)
YEAR(date)	Returns the year
YEARWEEK(date)	Returns the year and week number

AGGREGATE FUNCTIONS

DATA TYPE	DESCRIPTION
AVG(column)	Returns the average value of the numeric column
COUNT(column)	Returns the number of non-null values in the column
COUNT(*)	Returns the number of rows in the table (including nulls)
COUNT(DISTINCT column)	Returns the number of unique values in the column
GROUP_CONCAT(column)	Concatenates values from a group into a single string (default separator is comma)
GROUP_CONCAT(DISTINCT column SEPARATOR ';')	Concatenates unique values from a group with a specified separator
MAX(column)	Returns the maximum value in the column
MIN(column)	Returns the minimum value in the column
STD(column)	Returns the standard deviation of the column
STDDEV(column)	Returns the standard deviation of the column (alias for STD)
STDDEV_POP(column)	Returns the population standard deviation of the column
STDDEV_SAMP(column)	Returns the sample standard deviation of the column
SUM(column)	Returns the sum of the numeric column
VAR(column)	Returns the variance of the column
VARIANCE(column)	Returns the variance of the column (alias for VAR)
VAR_POP(column)	Returns the population variance of the column
VAR_SAMP(column)	Returns the sample variance of the column

WINDOW FUNCTIONS

DATA TYPE	DESCRIPTION
ROW_NUMBER()	Assigns a unique sequential integer to each row within the partition
RANK()	Assigns a rank to each row within the partition (gaps in ranking after ties)
DENSE_RANK()	Assigns a rank to each row within the partition (no gaps in ranking after ties)
NTILE(n)	Divides the rows into n buckets and assigns each row a bucket number
LEAD(column, offset, default)	Returns the value from the row offset rows after the current row (default if no row exists)
LAG(column, offset, default)	Returns the value from the row offset rows before the current row (default if no row exists)
FIRST_VALUE(column)	Returns the first value in the window frame
LAST_VALUE(column)	Returns the last value in the window frame
NTH_VALUE(column, n)	Returns the nth value in the window frame
PERCENT_RANK()	Returns the relative rank of the row (percentage from 0 to 1)
CUME_DIST()	Returns the cumulative distribution of the value (percentage of rows ≤ current row)

Kanak Infosystems LLP

KEYS

EXAMPLE (MYSQL)

```
CREATE TABLE users (
    id INT PRIMARY KEY,
    name VARCHAR(50),
    email VARCHAR(100)
);
CREATE TABLE orders (
    id INT PRIMARY KEY,
    user_id INT,
    order_date DATE,
    total DECIMAL(10,2),
    FOREIGN KEY (user_id) REFERENCES users(id)
);
```

EXAMPLE (MYSQL)

```
CREATE TABLE products (
    id INT PRIMARY KEY,
    name VARCHAR(50),
    price DECIMAL(10,2)
);
```

PRIMARY KEY

EXAMPLE (MYSQL)

```
CREATE TABLE users (
    id INT PRIMARY KEY,
    name VARCHAR(50),
    email VARCHAR(100)
);
```

EXAMPLE (MYSQL)

```
ALTER TABLE users
ADD PRIMARY KEY (id);
```

Kanak Infosystems LLP

DATA TYPES

DATA TYPE	DESCRIPTION
CREATE TABLE	Creates a new table in the database CREATE TABLE table_name (column1 datatype, column2 datatype, ...);
ALTER TABLE	Modifies an existing table, adding, dropping, or modifying columns ALTER TABLE table_name ADD column datatype; ALTER TABLE table_name DROP COLUMN column_name; ALTER TABLE table_name MODIFY column datatype;
DROP TABLE	Deletes a table and all its data from the database DROP TABLE table_name;
TRUNCATE TABLE	Deletes all rows in a table but keeps the table structure (faster than DELETE) TRUNCATE TABLE table_name;

VIEWS

CREATING VIEWS

```
CREATE VIEW view_name AS
SELECT column1, column2
FROM table
WHERE condition;
```

REPLACING VIEWS

```
CREATE OR REPLACE VIEW view_name AS
SELECT column1, column2
FROM table
WHERE condition;
```

DELETING VIEWS

```
DROP VIEW view_name;
```

JOINS

ORDERS

id	user_id	order_date	total
1	1	2025-01-15	150.00
2	2	2025-01-16	200.00
3	1	2025-01-17	75.50

PRODUCTS

id	name	price	stock_quantity
1	Laptop	1200.00	50
2	Mouse	25.00	100
3	Keyboard	75.00	75

CROSS

id	first_name	last_name	email	phone
1	John	Doe	johndoe@mail.com	1234567890
2	Jane	Smith	janesmith@mail.com	2345678901
3	Bob	Johnson	bobjohnson@mail.com	3456789012

id	product_name	price	stock_quantity
1	Laptop	1200.00	50
2	Mouse	25.00	100
3	Keyboard	75.00	75

CROSS JOIN combines every row from one table with every row from another table, producing all possible combinations.

Example:
If table A has 3 rows and table B has 4 rows, CROSS JOIN returns 12 rows (3 × 4).

Kanak Infosystems LLP