Kanak

SQL

```
SELECT * FROM [TABLE]
WHERE [CONDITION]

SELECT * FROM [TABLE]
ORDER BY [COLUMN]

SELECT * FROM [TABLE]
GROUP BY [COLUMN]
```

SQL KEYWORDS

KEYWORDS	DESCRIPTION
ADD	Adding a column in an existing table, or adding a primary or foreign key in an existing table
ADD CONSTRAINT	Adding a constraint after an existing table has been created
ALTER DATABASE	Modifying a database
ALTER TABLE	Adding, deleting, or modifying columns in an existing table, and changing the data type of columns in an existing table
ALTER COLUMN	Changing the data type of a column in a table
ALL	Returns true if all of the subqueries meet the condition
AND	Used in conditions to combine two or more conditions
ANY	Returns true if any of the subqueries meet the condition
AS	Renaming a table or column temporarily
ASC	Sorting the result set in ascending order
BACKUP DATABASE	Creating a backup of a database
BETWEEN	Selecting values within a given range. The values can be numbers, text, or dates
CASE	Creating different outputs based on conditions
CHECK	Adding a check constraint to a column when a table is created
CREATE DATABASE	Creating a database
CREATE INDEX	Creating an index (allows you to select data from the database quickly)
DEFAULT	Adding a default value to a column when a table is created
DELETE	Deleting rows from a table
DESC	Sorting the result set in descending order
DISTINCT	Selecting only distinct values

SQL KEYWORDS

KEYWORDS	DESCRIPTION
DROP	Deleting a column in an existing table, or deleting a primary or foreign key in an existing table
DROP CONSTRAINT	Deleting a constraint after an existing table has been created
DROP DATABASE	Deleting a database
DROP TABLE	Deleting a table
EXEC	Executing a stored procedure
EXISTS	Returns true if the subquery returns one or more records
FOR	Specifying which tables or columns to delete from
FOREIGN KEY	Creating a foreign key constraint when a table has been created
FROM	Specifying which tables or columns to select from
GROUP BY	Grouping the result set by one or more columns
HAVING	Used instead of WHERE in conditions with aggregate functions
IN	Specifying multiple values for a column
INDEX	Creating or deleting an index in a table
INNER JOIN	Returns rows that have matching values in both tables
INSERT INTO	Inserting new rows in a table
IS NULL	Testing for empty values
IS NOT NULL	Testing for non-empty values
LIKE	Searching for a specified pattern in a column
LIMIT	Specifying how many rows to return, starting from the first one
NOT	Used in conditions to negate a condition
NOT NULL	Specifying that a column can not have NULL values when a table is created
OR	Used in conditions to combine two or more conditions
ORDER BY	Sorting the result set by one or more columns
PRIMARY KEY	Creating a primary key when a table has been created
PROCEDURE	Creating a stored procedure
RIGHT JOIN	Returns rows that have matching values in both tables
ROWNUM	Specifying how many rows to return, starting from the first one
SELECT	Selecting data from a database
SELECT DISTINCT	Selecting distinct values from a column
SELECT TOP	Specifying how many rows to return, starting from the first one
SET	Specifying which values or columns to update
SOME	Returns true if any of the subqueries meet the condition
TOP	Specifying how many rows to return, starting from the first one
TRUNCATE TABLE	Deleting the data inside a table, but not the table itself
UNION	Combining the result sets of two or more SELECT statements (only distinct values)
UNION ALL	Combining the result sets of two or more SELECT statements (allows duplicate values)
UNIQUE	Specifying that the values in a column must be unique when a table has been created
UPDATE	Updating existing rows in a table
VALUES	Specifying the values of an INSERT INTO statement
WHERE	Specifying a condition when fetching data from a table

Kanak Infosystems LLP

COMMENTS

SINGLE LINE COMMENTS

```
-- This is a single line comment
SELECT * FROM table1;
```

MULTILINE COMMENTS

```
/* This is a multiline comment
It can be used anywhere
Including in the middle of a statement
SELECT
/* Nested comments are also allowed! */
* FROM table1
WHERE [condition];
```

MySQL Data Types

(version 8.0)

- String types
- Numeric types
- Date and time types

STRING DATA TYPES

DATA TYPE	DESCRIPTION
CHAR(SIZE)	A fixed length string (can contain letters, numbers, and special characters). The size parameter specifies the column length in characters - can be from 0 to 255. Default is 1
VARCHAR(SIZE)	A variable length string (can contain letters, numbers, and special characters). The size parameter specifies the maximum column length in characters - can be from 0 to 65535
BINARY(SIZE)	Equal to CHAR(), but stores binary byte strings. The size parameter specifies the column length in bytes. Default is 1
VARBINARY(SIZE)	Equal to VARCHAR(), but stores binary byte strings. The size parameter specifies the maximum column length in bytes.
TINYBLOB	For BLOBs (Binary Large Objects). Holds up to 255 bytes
TINYTEXT	Holds a string with a maximum length of 255 characters
TEXT(SIZE)	Holds a string with a maximum length of 65,535 bytes
BLOB(SIZE)	For BLOBs (Binary Large Objects). Holds up to 65,535 bytes of data
MEDIUMTEXT	Holds a string with a maximum length of 16,777,215 characters
MEDIUMBLOB	For BLOBs (Binary Large Objects). Holds up to 16,777,215 bytes of data
LONGTEXT	Holds a string with a maximum length of 4,294,967,295 characters
LONGBLOB	For BLOBs (Binary Large Objects). Holds up to 4,294,967,295 bytes of data
ENUM(val1, val2, val3, ...)	A string object that can have only one value, chosen from a list of possible values. You can enter up to 65535 values in an ENUM list. If a value is inserted that is not in the list, a blank value will be inserted. The values are sorted in the order you enter them
SET(val1, val2, val3, ...)	A string object that can have 0 or more values, chosen from a list of possible values. You can enter up to 64 values in a SET list

NUMERIC DATA TYPES

DATA TYPE	DESCRIPTION
BIT(SIZE)	A bit-value type. The number of bits per value is specified in SIZE. The possible values for SIZE are from 1 to 64. Default is 1
TINYINT(SIZE)	A very small integer. Signed range is from -128 to 127. Unsigned range is from 0 to 255. The SIZE parameter specifies the maximum display width (which is 255)
BOOL or BOOLEAN	Zero is considered as false, non-zero values are considered as true.
SMALLINT(SIZE)	A small integer. Signed range is from -32768 to 32767. Unsigned range is from 0 to 65535. The SIZE parameter specifies the maximum display width (which is 255)
MEDIUMINT(SIZE)	A medium integer. Signed range is from -8388608 to 8388607. Unsigned range is from 0 to 16777215. The SIZE parameter specifies the maximum display width (which is 255)
INT(SIZE)	A medium integer. Signed range is from -2147483648 to 2147483647. Unsigned range is from 0 to 4294967295. The SIZE parameter specifies the maximum display width (which is 255)
BIGINT(SIZE)	A large integer. Signed range is from -9223372036854775808 to 9223372036854775807. Unsigned range is from 0 to 18446744073709551615. The SIZE parameter specifies the maximum display width (which is 255)
FLOAT(p)	Used to specify a floating point number. MySQL uses the p value to determine whether to use FLOAT or DOUBLE. If p is between 0 and 24, the data type becomes FLOAT(). If p is from 25 to 53, the data type becomes DOUBLE()
DOUBLE(size, d)	A floating point number. The total number of digits is specified in size. The number of digits after the decimal point is specified in the d parameter
DECIMAL(size, d), NUMERIC(size, d)	Stores a fixed point number. The total number of digits is specified in size. The number of digits after the decimal point is specified in the d parameter
DOUBLE(size, d)	Deprecated in MySQL 8.0.17

DATE AND TIME DATA TYPES

DATA TYPE	DESCRIPTION
DATE	Date in format YYYY-MM-DD
DATETIME(fsp)	Date and time in format YYYY-MM-DD HH:MM:SS. The supported range is from '1000-01-01 00:00:00' to '9999-12-31 23:59:59'. Adding DEFAULT and ON UPDATE in the column definition to get automatic initialization and updating to the current date and time
TIMESTAMP(fsp)	Stores the number of seconds since the Unix epoch ('1970-01-01 00:00:00' UTC). Its format is YYYY-MM-DD HH:MM:SS. Adding DEFAULT and ON UPDATE in the column definition to get automatic initialization and updating to the current date and time
TIME(fsp)	Time in format HH:MM:SS
YEAR	A year in four-digit format (4 digits allowed by default). Values allowed in four digits format: 1901 to 2155, and 0000

Kanak Infosystems LLP

OPERATORS

ARITHMETIC OPERATORS

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

COMPARISON OPERATORS

OPERATOR	DESCRIPTION
=	Equal to
>	Greater than
<	Less than
>=	Greater than or equal
<=	Less than or equal
<>	Not equal to
!=	Not equal to

BITWISE OPERATORS

OPERATOR	DESCRIPTION
&	Bitwise AND
|	Bitwise OR
^	Bitwise XOR
<<	Left shift
>>	Right shift
~	Bitwise NOT

COMPOUND OPERATORS

OPERATOR	DESCRIPTION
+=	Add equal
-=	Subtract equal
*=	Multiply equal
/=	Divide equal
%=	Modulo equal
&=	Bitwise AND equal
^-=	Bitwise XOR equal
|*=	Bitwise OR equal

LOGICAL OPERATORS

OPERATOR	DESCRIPTION
ALL	TRUE if all of the subqueries values meet the condition
AND	TRUE if all the conditions separated by AND is TRUE
ANY	TRUE if any of the subqueries values meet the condition
BETWEEN	TRUE if the operand is within the range of comparisons
EXISTS	TRUE if the subquery returns one or more records
IN	TRUE if the operand is equal to one of a list of expressions
LIKE	TRUE if the operand matches a pattern
NOT	Displays a record if the condition(s) is NOT TRUE
OR	TRUE if any of the conditions separated by OR is TRUE
SOME	TRUE if any of the subqueries values meet the condition

Kanak Infosystems LLP

FUNCTIONS

STRING FUNCTIONS

DATA TYPE	DESCRIPTION
ASCII(x)	Returns the ASCII value for the specified character
CHAR_LENGTH(x)	Returns the length of the string (in characters)
CHARACTER_LENGTH(x)	Returns the length of the string (in characters)
CONCAT(x, y, ...)	Adds two or more expressions together
CONCAT_WS(x, y, ...)	Adds two or more expressions together with a separator
FIELD(x, y, ...)	Returns the index position of a value in a list of values
FIND_IN_SET(x, y)	Returns the position of a value in a set of values
FORMAT(x, y)	Formats a number to a format like "#,###,###.##", rounded to a specified number of decimal places
INSERT(x, y, z, w)	Inserts a string within a string at the specified position and for a specified number of characters
INSTR(x, y)	Returns the first position of the first occurrence of a specified value in a string
LCASE(x)	Converts a string to lower-case
LEFT(x, y)	Returns the left part of a string with the specified number of characters
LENGTH(x)	Returns the length of a string (in bytes)
LOCATE(x, y, z)	Returns the position of the first occurrence of a specified value in a string
LOWER(x)	Converts a string to lower-case
LPAD(x, y, z)	Adds a specified expression x to the left side of another expression, a specified number of times
LTRIM(x)	Removes leading spaces from a string
MID(x, y, z)	Extracts a substring from a string (starting at any position)
POSITION(x IN y)	Returns the position of the first occurrence of a specified value in a string
REPEAT(x, y)	Repeats a specified expression x a specified number of times
REPLACE(x, y, z)	Replaces all occurrences of a specified expression with another expression
REVERSE(x)	Reverses a string and returns the result
RIGHT(x, y)	Returns the right part of a string with the specified number of characters
RPAD(x, y, z)	Adds a specified expression x to the right side of another expression, a specified number of times
RTRIM(x)	Removes trailing spaces from a string
SPACE(x)	Returns a specified number of space characters
STRCMP(x, y)	Compares two strings
SUBSTR(x, y, z)	Extracts a substring from a string (starting at any position)
SUBSTRING(x, y, z)	Extracts a substring from a string (starting at any position)
SUBSTRING_INDEX(x, y, z)	Returns a substring of a string before a specified number of delimiter occurrences
TRIM(x)	Removes leading and trailing spaces from a string
UCASE(x)	Converts a string to upper-case
UPPER(x)	Converts a string to upper-case

NUMERIC FUNCTIONS

DATA TYPE	DESCRIPTION
ABS(x)	Returns the absolute value of a number
ACOS(x)	Returns the arccosine of a number
ASIN(x)	Returns the arcsine of a number
ATAN(x)	Returns the arctan of a number
ATAN2(x, y)	Returns the arctan2 of a pair of coordinates
CEIL(x)	Returns the smallest integer value that is bigger than or equal to a number
CEILING(x)	Returns the smallest integer value that is bigger than or equal to a number
COS(x)	Returns the cosine of a number
COT(x)	Returns the cotangent of a number
DEGREES(x)	Converts a value in radians to degrees
DIV	Integer division. Similar to FLOOR() but with no decimal places
EXP(x)	Returns e to the power of x
FLOOR(x)	Returns the largest integer value that is smaller than or equal to a number
GREATEST(x, y, ...)	Returns the greatest value of the list of arguments
LN(x)	Returns the natural logarithm of a number
LOG(x)	Returns the natural logarithm of a number, or the logarithm of a number to specified base
LOG10(x)	Returns the base-10 logarithm of a number
LOG2(x)	Returns the base-2 logarithm of a number
MOD(x, y)	Returns the remainder of a number divided by another number
PI()	Returns PI
POW(x, y)	Returns x to the power of y
POWER(x, y)	Returns x to the power of y
RADIANS(x)	Converts a degree value to radians
RAND()	Returns a random number
ROUND(x, y)	Rounds a number to a specified number of decimal places
SIGN(x)	Returns the sign of a number
SIN(x)	Returns the sine of a number
SQRT(x)	Returns the square root of a number
TAN(x)	Returns the tangent of a number
TRUNCATE(x, y)	Returns x truncated to y decimal places

NUMERIC FUNCTIONS

DATA TYPE	DESCRIPTION
ADDDATE(d, n)	Adds a specified time/date value to a date
ADDTIME(d, n)	Adds a specified time value to a date
CURDATE()	Returns the current date
CURRENT_DATE()	Returns the current date
CURRENT_TIME()	Returns the current time
CURRENT_TIMESTAMP()	Returns the current date and time
CURTIME()	Returns the current time
DATE()	Extracts the date part from a date or datetime expression
DATEDIFF(d, n)	Returns the number of days between two dates
DATE_FORMAT(d, f)	Displays date/time data in different formats as specified by the format parameter
DAY(d)	Returns the day part of a date
DAYNAME(d)	Returns the weekday name for a date
DAYOFMONTH(d)	Returns the day of the month for a given date
DAYOFWEEK(d)	Returns the weekday number for a given date
DAYOFYEAR(d)	Returns the day of the year for a given date
FROM_DAYS(n)	Returns a date from a given numeric value
FROM_UNIXTIME(n)	Returns a date and time value based on a UNIX timestamp
HOUR(t)	Returns the hour part of a given time/datetime
LAST_DAY(d)	Returns the last day of the month for a given date
LOCALTIME()	Returns the current date and time
LOCALTIMESTAMP()	Returns the current date and time
MAKEDATE(y, d)	Creates a date from a given year and a given number of days
MAKETIME(h, m, s)	Creates a time from a given hour, minute, and second
MICROSECOND(t)	Returns the microsecond part of a given time/datetime
MINUTE(t)	Returns the minute part of a given time/datetime
MONTH(d)	Returns the month part of a given date
MONTHNAME(d)	Returns the name of the month for a given date
NOW()	Returns the current date and time
PERIOD_ADD(p, n)	Adds a specified number of months to a period
PERIOD_DIFF(p1, p2)	Returns the difference between two periods
QUARTER(d)	Returns the quarter for a given date value
SEC_TO_TIME(s)	Converts a specified number of seconds into hours, minutes, and seconds
STR_TO_DATE(s, f)	Returns a date based on a string and a format
SUBDATE(d, n)	Subtracts a specified time/date value from a date
SUBTIME(d, n)	Subtracts a specified time value from a date
SYSDATE()	Returns the current date and time
TIME()	Extracts the time part from a given time/datetime expression
TIME_FORMAT(t, f)	Displays time/datetime data in different formats as specified by the format parameter
TIME_TO_SEC(t)	Converts a time value into seconds
TIMEDIFF(t1, t2)	Returns the difference between two time/datetime expressions
TIMESTAMP(x, y)	Returns a datetime value based on a date or datetime value
TO_DAYS(d)	Returns the number of days between a date and year 0
WEEK(d)	Returns the week number for a given date
WEEKDAY(d)	Returns the weekday number for a given date
WEEKOFYEAR(d)	Returns the week number for a given date
YEAR(d)	Returns the year part for a given date
YEARWEEK(d)	Returns the year and week number for a given date

Kanak Infosystems LLP

KEYS

FOREIGN KEY

```
EXAMPLE (MYSQL):
CREATE TABLE orders (
order_id INT,
customer_id INT,
order_date DATE,
PRIMARY KEY (order_id),
FOREIGN KEY (customer_id)
REFERENCES users(user_id)
);
```

```
EXAMPLE (MYSQL):
CREATE TABLE users (
user_id INT,
name VARCHAR(255),
email VARCHAR(255),
PRIMARY KEY (user_id)
);
```

PRIMARY KEY

```
EXAMPLE (MYSQL):
CREATE TABLE users (
user_id INT,
name VARCHAR(255),
email VARCHAR(255),
PRIMARY KEY (user_id)
);
```

```
EXAMPLE (MYSQL):
ALTER TABLE users
ADD PRIMARY KEY (user_id);
```

Kanak Infosystems LLP

DATA TYPES

DATA TYPE	DESCRIPTION
CREATE TABLE	Creates a table in the database. Syntax: CREATE TABLE table_name (column1 datatype, column2 datatype, column3 datatype, ...);
DROP TABLE	Deletes a table in the database. Syntax: DROP TABLE table_name;
DROP DATABASE	Deletes a database. Syntax: DROP DATABASE database_name;
ALTER TABLE - ADD	Adds a column in an existing table. Syntax: ALTER TABLE table_name ADD column_name datatype;
ALTER TABLE - DROP COLUMN	Deletes a column in an existing table. Syntax: ALTER TABLE table_name DROP COLUMN column_name;
ALTER TABLE - RENAME	Renames a table. Syntax: ALTER TABLE table_name RENAME TO new_table_name;
ALTER TABLE - ALTER/MODIFY COLUMN	Changes the data type of a column in a table. Syntax: ALTER TABLE table_name ALTER COLUMN column_name datatype; or ALTER TABLE table_name MODIFY column_name datatype;

INDEX

CREATING INDEX

```
CREATE INDEX idx_name ON table (column);
```

IMPLICIT INDEX

Implicit index is created by PRIMARY KEY constraint and UNIQUE constraint.

```
SHOW INDEX FROM table;
```

DELETING INDEX

```
DROP INDEX idx_name;
```

VIEWS

CREATE VIEW

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

REPLACE VIEW

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

DELETE VIEW

```
DROP VIEW view_name;
```

JOINS

ORDERS

id	user_id	product_id	quantity
1	1	1	2
2	1	2	1
3	2	1	1
4	3	3	1

PRODUCTS

id	name	manufacturer	price
1	Product 1	Company A	100
2	Product 2	Company A	200
3	Product 3	Company B	300
4	Product 4	Company C	400

CROSS

id	first_name	last_name	product_id	product_name
1	John	Doe	1	Product 1
1	John	Doe	2	Product 2
1	John	Doe	3	Product 3
1	John	Doe	4	Product 4
2	Jane	Smith	1	Product 1
2	Jane	Smith	2	Product 2
2	Jane	Smith	3	Product 3
2	Jane	Smith	4	Product 4
3	Bob	Brown	1	Product 1
3	Bob	Brown	2	Product 2
3	Bob	Brown	3	Product 3
3	Bob	Brown	4	Product 4

SELECT column1, column2, ...
FROM table1
CROSS JOIN table2;

SELECT column1, column2, ...
FROM table1, table2;

SELF JOIN

id	first_name	last_name	product_id
1	John	Doe	1
2	Jane	Smith	2
3	Bob	Brown	3

```
SELECT table1.column1, table1.column2, table2.column1, table2.column2
FROM table1 AS table1, table1 AS table2
WHERE table1.id = table2.id;
```

Kanak Infosystems LLP