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 an existing table. Syntax: ALTER TABLE table1 ADD column1 datatype;
ADD CONSTRAINT	Creates an constraint after a table is created. Syntax: ALTER TABLE table1 ADD CONSTRAINT constraint1 PRIMARY KEY (column1);
ALTER TABLE	Adds, deletes, or modifies columns in a table, or changes the data type of columns in a table. Syntax: ALTER TABLE table1 ADD column1 datatype; ALTER TABLE table1 DROP column1; ALTER TABLE table1 ALTER COLUMN column1 datatype;
ALTER COLUMN	Changes the data type of a column in a table. Syntax: ALTER TABLE table1 ALTER COLUMN column1 datatype;
ALL	Returns true if all of the subqueries meet the condition. Syntax: SELECT column1 FROM table1 WHERE column1 operator ALL (SELECT column1 FROM table2 WHERE condition);
AND	Adds a condition in a WHERE clause or JOIN clause. Syntax: SELECT column1 FROM table1 WHERE condition1 AND condition2;
ANY	Returns true if any of the subqueries meet the condition. Syntax: SELECT column1 FROM table1 WHERE column1 operator ANY (SELECT column1 FROM table2 WHERE condition);
AS	Renames a column or table with an alias. Syntax: SELECT column1 AS name1 FROM table1;
ASC	Sorts the result set in ascending order. Syntax: SELECT column1 FROM table1 ORDER BY column1 ASC;
BETWEEN	Selects values within a given range. Syntax: SELECT column1 FROM table1 WHERE column1 BETWEEN value1 AND value2;
CASE	Creates a table with a column of type varchar(n) and no other constraints. Syntax: CREATE TABLE table1 (column1 varchar(n));
CHECK	Creates a table with a check constraint on one or more columns. Syntax: CREATE TABLE table1 (column1 datatype, CHECK (column1 condition));
CREATE DATABASE	Creates a database. Syntax: CREATE DATABASE database1;
CREATE INDEX	Creates an index on a table. Syntax: CREATE INDEX index1 ON table1 (column1);
DEFAULT	Sets a default value for a column when no value is specified. Syntax: CREATE TABLE table1 (column1 datatype, column2 datatype DEFAULT 'abc');
DELETE	Deletes rows from a table. Syntax: DELETE FROM table1 WHERE condition;
DESC	Sorts the result set in descending order. Syntax: SELECT column1 FROM table1 ORDER BY column1 DESC;

SQL KEYWORDS

KEYWORDS	DESCRIPTION
DROP CONSTRAINT	Deletes a constraint. Syntax: ALTER TABLE table1 DROP CONSTRAINT constraint1;
DROP DATABASE	Deletes a database. Syntax: DROP DATABASE database1;
DROP INDEX	Deletes an index. Syntax: DROP INDEX table1.index1;
DROP TABLE	Deletes a table. Syntax: DROP TABLE table1;
EXISTS	Tests for the existence of any record in a subquery. Syntax: SELECT column1 FROM table1 WHERE EXISTS (SELECT column1 FROM table2 WHERE condition);
FROM	Specifies which tables to select or delete from. Syntax: SELECT column1 FROM table1;
GROUP BY	Groups the result set by one or more columns. Syntax: SELECT column1, function(column2) FROM table1 GROUP BY column1;
HAVING	Specifies a condition for a group or aggregate function. Syntax: SELECT column1, function(column2) FROM table1 GROUP BY column1 HAVING condition;
IN	Checks if a value is in a set of values. Syntax: SELECT column1 FROM table1 WHERE column1 IN (value1, value2, ...);
INDEX (MY SQL)	Creates an index on a table. Syntax: CREATE INDEX index1 ON table1 (column1);
IN NULL	Selects all records that have a null value in a column. Syntax: SELECT column1 FROM table1 WHERE column1 IS NULL;
IN NOT NULL	Selects all records that have a not null value in a column. Syntax: SELECT column1 FROM table1 WHERE column1 IS NOT NULL;
LIKE	Compares a value to similar values using wildcard operators. Syntax: SELECT column1 FROM table1 WHERE column1 LIKE pattern;
LIMIT	Limits the number of rows returned by a query. Syntax: SELECT column1 FROM table1 LIMIT 10;
NOT	Negates a condition. Syntax: SELECT column1 FROM table1 WHERE NOT condition;
OR	Adds a condition in a WHERE clause or JOIN clause. Syntax: SELECT column1 FROM table1 WHERE condition1 OR condition2;
ORDER BY	Sorts the result set in ascending or descending order. Syntax: SELECT column1 FROM table1 ORDER BY column1 ASC;
SELECT	Selects data from a database. Syntax: SELECT column1 FROM table1;
SELECT INTO	Copies data from one table into a new table. Syntax: SELECT column1 INTO table2 FROM table1;
SELECT TOP	Specifies the number of records to return in a query. Syntax: SELECT TOP 10 column1 FROM table1;
SET	Specifies which values or columns are to be updated. Syntax: UPDATE table1 SET column1 = value1;
SOME	Returns true if any of the subqueries meet the condition.
TOP	Specifies the number of records to return. Syntax: SELECT TOP 10 column1 FROM table1;
TRUNCATE TABLE	Deletes all records in a table, but not the table itself. Syntax: TRUNCATE TABLE table1;
UNION	Combines the result sets of two or more SELECT statements (only distinct values). Syntax: SELECT column1 FROM table1 UNION SELECT column1 FROM table2;
UNION ALL	Combines the result sets of two or more SELECT statements (allows duplicate values). Syntax: SELECT column1 FROM table1 UNION ALL SELECT column1 FROM table2;
UNIQUE	Creates a table with a unique constraint on one or more columns. Syntax: CREATE TABLE table1 (column1 datatype, UNIQUE (column1));
UPDATE	Updates existing data in a table. Syntax: UPDATE table1 SET column1 = value1 WHERE condition;
VALUES	Specifies the values for an INSERT INTO statement. Syntax: INSERT INTO table1 (column1, column2, ...) VALUES (value1, value2, ...);
WHERE	Specifies a condition for selecting records. Syntax: SELECT column1 FROM table1 WHERE condition;

Kanak Infosystems LLP

COMMENTS

SINGLE LINE COMMENTS

Single line comments start with two dashes (--) and end at the end of the line.

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

MULTILINE COMMENTS

Multiline comments begin with a slash followed by an asterisk (/*) and end with an asterisk followed by a slash (*/). Syntax:

```
/* This is a multiline comment
SELECT * FROM table1;
SELECT * FROM table2;
*/
```

MySQL Data Types

(click to view again)

- MySQL Data Types
- PHP Data Types

NUMERIC DATA TYPES

DATA TYPE	DESCRIPTION
TINYINT(size)	-128 to 127 normal. 0 to 255 UNSIGNED. Optional: specify the number of digits in parenthesis.
SMALLINT(size)	-32768 to 32767 normal. 0 to 65535 UNSIGNED. Optional: specify the number of digits in parenthesis.
MEDIUMINT(size)	-8388608 to 8388607 normal. 0 to 16777215 UNSIGNED. Optional: specify the number of digits in parenthesis.
INT(size)	-2147483648 to 2147483647 normal. 0 to 4294967295 UNSIGNED. Optional: specify the number of digits in parenthesis.
BIGINT(size)	-9223372036854775808 to 9223372036854775807 normal. 0 to 18446744073709551615 UNSIGNED. Optional: specify the number of digits in parenthesis.
FLOAT(p)	Floating point number. MySQL uses p to determine whether to use FLOAT or DOUBLE.
DOUBLE(size, d)	Floating point number. Optional: specify the number of digits in parenthesis.
DECIMAL(size, d), NUMERIC(size, d)	Stored as a string, allows for a fixed decimal point.

DATE AND TIME DATA TYPES

DATA TYPE	DESCRIPTION
DATE	Date. Format: YYYY-MM-DD. Supported range is from '1000-01-01' to '9999-12-31'.
DATETIME(fsp)	Date and time combination. Format: YYYY-MM-DD hh:mm:ss. 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). Format: YYYY-MM-DD hh:mm:ss. Supported range is from '1970-01-01 00:00:01' UTC to '2038-01-19 03:14:07' UTC. Adding DEFAULT and ON UPDATE in the column definition to get automatic initialization and updating to the current date and time.
TIME(fsp)	Time. Format: hh:mm:ss. Supported range is from '-838:59:59' to '838:59:59'.
YEAR	Year in 4-digit format. Allowed: 1901 to 2155, and 0000.

NON-NUMERIC DATA TYPES

DATA TYPE	DESCRIPTION
BINARY(size)	Stores fixed-length binary strings. Optional: specify the number of bytes in parenthesis.
VARBINARY(size)	Stores variable-length binary strings. Optional: specify the maximum number of bytes in parenthesis.
TINYBLOB(size)	Stores binary strings with a maximum length of 255 bytes. Optional: specify the number of bytes in parenthesis.
BLOB(size)	Stores binary strings with a maximum length of 65535 bytes. Optional: specify the number of bytes in parenthesis.
MEDIUMBLOB(size)	Stores binary strings with a maximum length of 16777215 bytes. Optional: specify the number of bytes in parenthesis.
LONGBLOB(size)	Stores binary strings with a maximum length of 4294967295 bytes. Optional: specify the number of bytes in parenthesis.
TINYTEXT(size)	Stores text with a maximum length of 255 characters. Optional: specify the number of characters in parenthesis.
TEXT(size)	Stores text with a maximum length of 65535 characters. Optional: specify the number of characters in parenthesis.
MEDIUMTEXT(size)	Stores text with a maximum length of 16777215 characters. Optional: specify the number of characters in parenthesis.
LONGTEXT(size)	Stores text with a maximum length of 4294967295 characters. Optional: specify the number of characters in parenthesis.
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.
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.

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 OR
&	Bitwise AND
^	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

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.
CHARACTER_LENGTH(x)	Returns the length of the string.
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 string within a list of strings.
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 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 the string, measured in bytes.
LOCATE(x, y, z)	Returns the position of a value in a string.
LOWER(x)	Converts a string to lower-case.
LPAD(x, y, z)	Adds a string to the left side of another string, a specified number of times.
LTRIM(x)	Removes leading spaces.
MID(x, y, z)	Returns a substring starting from the specified position.
POSITION(x IN y)	Returns the position of a value in a string.
REPEAT(x, y)	Repeats a string as many times as specified.
REPLACE(x, y, z)	Replaces all occurrences of a specified string value with another string value.
REVERSE(x)	Reverses a string.
RIGHT(x, y)	Returns the right part of a string with the specified number of characters.
RPAD(x, y, z)	Adds a string to the right side of another string, a specified number of times.
RTRIM(x)	Removes trailing spaces.
SPACE(x)	Returns a string of the specified number of space characters.
STRCMP(x, y)	Compares two strings.
SUBSTR(x, y, z), SUBSTRING(x, y, z)	Returns a substring of a string, starting at a specified 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.
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.
AVG(x)	Returns the average value of a group of numbers.
CEIL(x)	Rounds a number up to the next whole number.
CEILING(x)	Rounds a number up to the next whole number.
COS(x)	Returns the cosine of a number.
COT(x)	Returns the cotangent of a number.
COUNT(x)	Returns the number of records in a group.
DIV(x, y)	Returns the integer quotient of a division.
EXP(x)	Returns e raised to the power of a number.
FLOOR(x)	Rounds a number down to the next whole number.
GREATEST(x, y, ...)	Returns the greatest value in a group of values.
LN(x)	Returns the natural logarithm of a number.
LOG(x)	Returns the base-10 logarithm of a number.
LOG10(x)	Returns the base-10 logarithm of a number.
LOG2(x)	Returns the base-2 logarithm of a number.
MAX(x)	Returns the maximum value in a group of values.
MIN(x)	Returns the minimum value in a group of values.
MOD(x, y)	Returns the remainder of a division.
PI()	Returns PI.
POW(x, y)	Returns x raised to the power of y.
POWER(x, y)	Returns x raised to the power of y.
RADIANS(x)	Converts degrees 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.
STD(x), STDDEV(x)	Returns the standard deviation of a group of values.
SUM(x)	Returns the total of a group of numbers.
TAN(x)	Returns the tangent of a number.
TRUNCATE(x, y)	Truncates a number to a specified number of decimal places.

NUMERIC FUNCTIONS

DATA TYPE	DESCRIPTION
ADDDATE(x, y)	Adds a date and a time interval.
ADDTIME(x, y)	Adds a time interval to a time/datetime.
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(x)	Extracts the date part from a date/datetime.
DATEDIFF(x, y)	Returns the difference between two dates.
DATE_FORMAT(x, y)	Formats a date/datetime according to a specified format.
DAY(x)	Returns the day of the month for a date.
DAYNAME(x)	Returns the name of the weekday for a date.
DAYOFMONTH(x)	Returns the day of the month for a date.
DAYOFWEEK(x)	Returns the weekday index for a date.
DAYOFYEAR(x)	Returns the day of the year for a date.
EXTRACT(x FROM y)	Extracts a part from a given date.
FROM_DAYS(x)	Returns a date from a numeric representation of the day.
HOUR(x)	Returns the hour for a time.
LAST_DAY(x)	Extracts 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(x, y)	Creates a date from the year and a day number.
MAKETIME(x, y, z)	Creates a time from the hour, minute, and second.
MICROSECOND(x)	Returns the microsecond for a time/datetime.
MINUTE(x)	Returns the minute for a time/datetime.
MONTH(x)	Returns the month for a date.
MONTHNAME(x)	Returns the name of the month for a date.
NOW()	Returns the current date and time.
PERIOD_ADD(x, y)	Adds a period to a year-month.
PERIOD_DIFF(x, y)	Returns the difference between two periods.
QUARTER(x)	Returns the quarter for a date/datetime.
SECOND(x)	Returns the second for a time/datetime.
SEC_TO_TIME(x)	Returns a time in hh:mm:ss format.
STR_TO_DATE(x, y)	Returns a date/datetime from a string.
SUBDATE(x, y)	Subtracts a date and a time interval.
SUBTIME(x, y)	Subtracts a time interval from a time/datetime.
SYSDATE()	Returns the current date and time.
TIME(x)	Extracts the time part from a date/datetime.
TIME_FORMAT(x, y)	Formats a time according to a specified format.
TIME_TO_SEC(x)	Returns the number of seconds from a time.
TIMEDIFF(x, y)	Returns the difference between two time/datetime.
TIMESTAMP(x, y)	Returns a datetime value.
TIMESTAMPADD(x, y, z)	Adds a specified interval to a date/datetime.
TIMESTAMPDIFF(x, y, z)	Returns the difference between two date/datetime.
TO_DAYS(x)	Returns the number of days between a date and year zero.
WEEK(x)	Returns the week number for the given date.
WEEKDAY(x)	Returns the weekday number for the given date.
WEEKOFYEAR(x)	Returns the week number for the given date.
YEAR(x)	Returns the year for the given date.
YEARWEEK(x)	Returns the year and week number for the given date.

Kanak Infosystems LLP

KEYS

FOREIGN KEY

EXAMPLE (MYSQL):

```
CREATE TABLE users (
  user_id INT,
  name VARCHAR(50),
  email VARCHAR(100),
  PRIMARY KEY (user_id)
);
CREATE TABLE orders (
  order_id INT,
  customer_id INT,
  order_date DATE,
  PRIMARY KEY (order_id),
  FOREIGN KEY (customer_id) REFERENCES users(user_id)
);
```

PRIMARY KEY

EXAMPLE (MYSQL):

```
CREATE TABLE table1 (
  id_number int NOT NULL,
  last_name varchar(255),
  first_name varchar(255),
  address varchar(255),
  city varchar(255),
  PRIMARY KEY (id_number)
);
```

EXAMPLE (MYSQL):

```
ALTER TABLE table1
ADD CONSTRAINT pk_PersonID PRIMARY KEY (id_number);
```

Kanak Infosystems LLP

DATA TYPES

DATA TYPE	DESCRIPTION
CREATE TABLE table1 (column1 datatype);	Creates a table with a column of type varchar(n) and no other constraints.
CREATE TABLE table1 (column1 datatype, CHECK (column1 condition));	Creates a table with a check constraint on one or more columns.
CREATE TABLE table1 (column1 datatype, column2 datatype DEFAULT 'abc');	Sets a default value for a column when no value is specified.
CREATE INDEX index1 ON table1 (column1);	Creates an index on a table.
DROP INDEX table1.index1;	Deletes an index.
DROP TABLE table1;	Deletes a table.
DROP DATABASE database1;	Deletes a database.

VIEWS

CREATING VIEWS

```
CREATE VIEW view1 AS
SELECT column1, column2, ...
FROM table1
WHERE condition;
```

REPLACING VIEWS

```
CREATE OR REPLACE VIEW view1 AS
SELECT column1, column2, ...
FROM table1
WHERE condition;
```

DELETING VIEWS

```
DROP VIEW view1;
```

JOINS

ORDERS

id	user_id	product_id	quantity	created_at
1	1	101	2	2025-01-10
2	2	102	1	2025-01-11
3	1	103	5	2025-01-12

PRODUCTS

id	name	category	price
101	Laptop	Electronics	1200.00
102	Mouse	Electronics	25.00
103	Desk	Furniture	300.00

CRUDS

id	first_name	last_name	product_name
1	John	Doe	Laptop
2	Jane	Smith	Mouse
3	Alice	Brown	Desk
4	Bob	Lee	Chair
5	Charlie	Kim	Monitor

SELECT column1, column2, FROM table1 WHERE condition;

INSERT INTO table1 (column1, column2, ...) VALUES (value1, value2, ...);

UPDATE table1 SET column1 = value1 WHERE condition;

DELETE FROM table1 WHERE condition;