Kanak

SQL

```
SELECT * FROM [REPLACEMENT]
WHERE [CONDITIONS]
ORDER BY [FIELD]
SELECT * FROM [REPLACEMENT]
GROUP BY [FIELD]
HAVING [CONDITIONS]
```

SQL KEYWORDS

KEYWORDS	DESCRIPTION
ADD	Adding a column in an existing table. ALTER TABLE table1 ADD column1 datatype;
ADD CONSTRAINT	Adding constraints in an existing table. ALTER TABLE table1 ADD CONSTRAINT constraint1 PRIMARY KEY (column1);
ALTER TABLE	Adding, deleting, or modifying columns in a table. ALTER TABLE table1 {ADD|DROP|MODIFY} column1 datatype;
ALTER COLUMN	Changing a column's data type. ALTER TABLE table1 ALTER COLUMN column1 datatype;
ALL	Returns true if all of the subquery values meet the condition. SELECT * FROM table1 WHERE column1 = ALL (SELECT column1 FROM table2);
AND	Used in conjunction with other operators. SELECT * FROM table1 WHERE condition1 AND condition2;
ANY	Returns true if any of the subquery values meet the condition. SELECT * FROM table1 WHERE column1 = ANY (SELECT column1 FROM table2);
AS	Renaming a table or column. SELECT column1 AS name1 FROM table1;
ASC	Sorting the result set in ascending order. SELECT * FROM table1 ORDER BY column1 ASC;
BETWEEN	Selecting values within a range. SELECT * FROM table1 WHERE column1 BETWEEN value1 AND value2;
CASE	Conditional statements. CASE WHEN condition1 THEN result1 WHEN condition2 THEN result2 ELSE result3 END;
CHECK	Ensuring that all values in a column satisfy a specific condition. CREATE TABLE table1 (column1 datatype, CHECK (column1 condition)); CREATE TABLE table1 (column1 datatype, CONSTRAINT constraint1 CHECK (column1 condition));
CREATE DATABASE	Creating a database. CREATE DATABASE database1;
CREATE INDEX	Creating an index. CREATE INDEX index1 ON table1 (column1);
DELETE	Deleting rows from a table. DELETE FROM table1 WHERE condition1;
DESC	Sorting the result set in descending order. SELECT * FROM table1 ORDER BY column1 DESC;
DISTINCT	Selecting unique values. SELECT DISTINCT column1 FROM table1;
DROP	Deleting a table, a database, or an index. DROP {TABLE|DATABASE|INDEX} name1;
DROP COLUMN	Deleting a column in an existing table. ALTER TABLE table1 DROP COLUMN column1;
EXISTS	Checking for the existence of a record in a subquery. SELECT column1 FROM table1 WHERE EXISTS (SELECT column1 FROM table2 WHERE condition1);
FROM	Specifying the table from which to retrieve or delete data. SELECT * FROM table1;
GROUP BY	Grouping rows with similar values in a column. SELECT column1, COUNT(column2) FROM table1 GROUP BY column1;
HAVING	Filtering groups based on a condition. SELECT column1, COUNT(column2) FROM table1 GROUP BY column1 HAVING COUNT(column2) > 1;
IN	Checking if a value is in a specified list of values. SELECT * FROM table1 WHERE column1 IN (value1, value2, value3);
INDEX WITH	Specifying a storage method for an index. CREATE INDEX index1 ON table1 (column1) WITH (FILLFACTOR = 70);
IN TO	Specifying a destination table for data being copied. SELECT * INTO table2 FROM table1;
INSERT INTO	Inserting new rows into a table. INSERT INTO table1 (column1, column2) VALUES (value1, value2);
INSERT INTO SELECT	Inserting data from one table into another. INSERT INTO table2 (column1, column2) SELECT column1, column2 FROM table1;
JOIN	Combining rows from two or more tables based on a related column. SELECT * FROM table1 JOIN table2 ON table1.column1 = table2.column1;
LEFT JOIN	Returning all rows from the left table, and the matching rows from the right table. SELECT * FROM table1 LEFT JOIN table2 ON table1.column1 = table2.column1;
LIKE	Searching for a specified pattern in a column. SELECT * FROM table1 WHERE column1 LIKE 'pattern%';
LIMIT	Limiting the number of rows returned. SELECT * FROM table1 LIMIT 10;
NOT	Negating a condition. SELECT * FROM table1 WHERE NOT condition1;
NOT NULL	Ensuring that a column cannot have NULL values. CREATE TABLE table1 (column1 datatype NOT NULL);
OR	Used in conjunction with other operators. SELECT * FROM table1 WHERE condition1 OR condition2;
ORDER BY	Sorting the result set in ascending or descending order. SELECT * FROM table1 ORDER BY column1 ASC|DESC;
OUTER JOIN	Returning all rows when there is a match in one of the tables. SELECT * FROM table1 OUTER JOIN table2 ON table1.column1 = table2.column1;
PRIMARY KEY	Uniquely identifying each record in a table. CREATE TABLE table1 (column1 datatype PRIMARY KEY);
RIGHT JOIN	Returning all rows from the right table, and the matching rows from the left table. SELECT * FROM table1 RIGHT JOIN table2 ON table1.column1 = table2.column1;
ROWNUM	Limiting the number of rows returned (Oracle). SELECT * FROM table1 WHERE ROWNUM < 10;
SELECT	Retrieving data from a database. SELECT column1, column2 FROM table1;
SELECT DISTINCT	Selecting unique values. SELECT DISTINCT column1, column2 FROM table1;
SELECT INTO	Copying data into a new table. SELECT column1, column2 INTO table2 FROM table1;
SET	Specifying new values for columns in an update statement. UPDATE table1 SET column1 = value1, column2 = value2 WHERE condition1;
SOME	Returns true if some of the subquery values meet the condition. SELECT * FROM table1 WHERE column1 = SOME (SELECT column1 FROM table2);
TOP	Limiting the number of rows returned (SQL Server). SELECT TOP 10 * FROM table1;
TRUNCATE TABLE	Deleting all rows in a table, but keeping the table structure. TRUNCATE TABLE table1;
UNION	Combining the results of two or more queries. SELECT column1 FROM table1 UNION SELECT column1 FROM table2;
UNION ALL	Combining the results of two or more queries, including duplicates. SELECT column1 FROM table1 UNION ALL SELECT column1 FROM table2;
UNIQUE	Ensuring that all values in a column are unique. CREATE TABLE table1 (column1 datatype UNIQUE);
UPDATE	Updating rows in a table. UPDATE table1 SET column1 = value1, column2 = value2 WHERE condition1;
VALUES	Specifying the values to be inserted in a table. INSERT INTO table1 (column1, column2) VALUES (value1, value2);
WHERE	Filtering rows based on a condition. SELECT * FROM table1 WHERE condition1;

Kanak Infosystems LLP

COMMENTS

SINGLE LINE COMMENTS

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

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

MULTILINE COMMENTS

Multiline comments begin with /* and end with */ and can span multiple lines.

```
/* This is a multiline comment
   that spans two lines */
SELECT * FROM table1;
/* Multiline comment
   with more details
   about the query */
```

MySQL Data Types

Click to Tab open

Click to close

NUMERIC DATA TYPES

DATA TYPE	DESCRIPTION
TINYINT(M)	Signed: -128 to 127. Unsigned: 0 to 255. Optional display width M.
SMALLINT(M)	Signed: -32,768 to 32,767. Unsigned: 0 to 65,535. Optional display width M.
MEDIUMINT(M)	Signed: -8,388,608 to 8,388,607. Unsigned: 0 to 16,777,215. Optional display width M.
INT(M), INTEGER(M)	Signed: -2,147,483,648 to 2,147,483,647. Unsigned: 0 to 4,294,967,295. Optional display width M.
BIGINT(M)	Signed: -9,223,372,036,854,775,808 to 9,223,372,036,854,775,807. Unsigned: 0 to 18,446,744,073,709,551,615. Optional display width M.
FLOAT(M,D)	Floating point number. Optional display width M and decimal places D.
DOUBLE(M,D), REAL(M,D)	Double precision floating point number. Optional display width M and decimal places D.
DECIMAL(M,D), DEC(M,D), NUMERIC(M,D), FIXED(M,D)	Fixed point number. Optional display width M and decimal places D.
BIT(M)	Bit value. Optional display width M (1 to 64).
BOOL, BOOLEAN	Synonyms for TINYINT(1). Zero is false, non-zero is true.

DATE AND TIME DATA TYPES

DATA TYPE	DESCRIPTION
DATE	Date in 'YYYY-MM-DD' format. Supported range: '1000-01-01' to '9999-12-31'.
DATETIME(fsp)	Date and time in 'YYYY-MM-DD HH:MM:SS' format. Optional fractional seconds precision fsp (0-6). Supported range: '1000-01-01 00:00:00' to '9999-12-31 23:59:59'.
TIMESTAMP(fsp)	Stores the number of seconds since the Unix epoch ('1970-01-01 00:00:00' UTC). Optional fractional seconds precision fsp (0-6). Supported range: '1970-01-01 00:00:01' to '2038-01-19 03:14:07'. Automatically updated to the current date and time upon insertion or update.
TIME(fsp)	Time in 'HH:MM:SS' format. Optional fractional seconds precision fsp (0-6). Supported range: '-838:59:59' to '838:59:59'.
YEAR	4-digit year in supported range of '1901' to '2155'.

STRING DATA TYPES

DATA TYPE	DESCRIPTION
CHAR(M)	Fixed-length character string. Optional length M (0 to 255). Trailing spaces are removed when retrieved.
VARCHAR(M)	Variable-length character string. Optional maximum length M (0 to 65,535). Trailing spaces are preserved.
TINYTEXT	Small text data. Maximum length 255 characters.
TEXT(M)	Text data. Optional maximum length M characters.
MEDIUMTEXT	Medium text data. Maximum length 16,777,215 characters.
LONGTEXT	Large text data. Maximum length 4,294,967,295 characters.
ENUM('val1','val2',...)	Enumeration. Stores one value from a predefined list of up to 65,535 values.
SET('val1','val2',...)	Set. Stores zero or more values from a predefined list of up to 64 values.
BINARY(M)	Fixed-length binary data. Optional length M (0 to 255). Trailing spaces are preserved.
VARBINARY(M)	Variable-length binary data. Optional maximum length M (0 to 65,535). Trailing spaces are preserved.
TINYBLOB	Small binary data. Maximum length 255 bytes.
BLOB(M)	Binary data. Optional maximum length M bytes.
MEDIUMBLOB	Medium binary data. Maximum length 16,777,215 bytes.
LONGBLOB	Large binary data. Maximum length 4,294,967,295 bytes.

BINARY DATA TYPES

DATA TYPE	DESCRIPTION
BINARY(M)	Fixed-length binary data. Optional length M (0 to 255). Trailing spaces are removed when retrieved.
VARBINARY(M)	Variable-length binary data. Optional maximum length M (0 to 65,535). Trailing spaces are preserved.
TINYBLOB	Small binary data. Maximum length 255 bytes.
BLOB(M)	Binary data. Optional maximum length M bytes.
MEDIUMBLOB	Medium binary data. Maximum length 16,777,215 bytes.
LONGBLOB	Large binary data. Maximum length 4,294,967,295 bytes.

Kanak Infosystems LLP

OPERATORS

ARITHMETIC OPERATORS

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

BITWISE OPERATORS

OPERATOR	DESCRIPTION
&	Bitwise AND
|	Bitwise OR
^	Bitwise XOR
<<	Left shift
>>	Right shift

COMPARISON OPERATORS

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

COMPOUND OPERATORS

OPERATOR	DESCRIPTION
+=	Add and assign
-=	Subtract and assign
*=	Multiply and assign
/=	Divide and assign
%=	Modulo and assign
&=	Bitwise AND and assign
|=	Bitwise OR and assign
^=	Bitwise XOR and assign
*=	Bitwise NOT and assign
||=	Concatenate and assign

Kanak Infosystems LLP

FUNCTIONS

STRING FUNCTIONS

DATA TYPE	DESCRIPTION
ASCII(s)	Returns the ASCII code value of a character.
CHAR_LENGTH(s), CHARACTER_LENGTH(s)	Returns the length of a string in characters.
CONCAT(s1,s2,...)	Concatenates strings.
CONCAT_WS(separator,s1,s2,...)	Concatenates strings with a separator.
FIELD(s,s1,s2,...)	Returns the index of a string in a list of strings.
FIND_IN_SET(s,slist)	Returns the index of a string in a comma-separated list of strings.
FORMAT(x,d)	Formats a number with d decimal places.
INSERT(s,x,y,s1)	Inserts a substring s1 into a string s at position x, replacing y characters.
INSTR(s,s1)	Returns the first occurrence of substring s1 in string s.
LCASE(s), LOWER(s)	Converts a string to lowercase.
LEFT(s,n)	Returns the leftmost n characters of a string.
LENGTH(s)	Returns the length of a string, in bytes.
LOCATE(s1,s)	Returns the first occurrence of substring s1 in string s.
LPAD(s,n,s1)	Pads a string s1 to the left.
LTRIM(s)	Removes leading spaces.
MID(s,n,m)	Returns a substring starting from position n, m characters long.
POSITION(s1 IN s)	Returns the first occurrence of substring s1 in string s.
REPEAT(s,n)	Repeats a string n times.
REPLACE(s,s1,s2)	Replaces all occurrences of substring s1 with s2.
REVERSE(s)	Reverses a string.
RIGHT(s,n)	Returns the rightmost n characters of a string.
RPAD(s,n,s1)	Pads a string s1 to the right.
RTRIM(s)	Removes trailing spaces.
SPACE(n)	Returns a string of n spaces.
STRCMP(s1,s2)	Compares two strings.
SUBSTR(s,n,m), SUBSTRING(s,n,m)	Returns a substring starting from position n, m characters long.
SUBSTRING_INDEX(s,s1,n)	Returns a substring before n occurrences of substring s1.
TRIM(s)	Removes leading and trailing spaces.
UCASE(s), UPPER(s)	Converts a string to uppercase.

NUMERIC FUNCTIONS

DATA TYPE	DESCRIPTION
ABS(x)	Returns the absolute value of x.
ACOS(x)	Returns the arccosine of x.
ASIN(x)	Returns the arcsine of x.
ATAN(x)	Returns the arctangent of x.
ATAN2(y,x)	Returns the arctangent of the two variables y and x.
CEIL(x)	Rounds x up to the nearest integer.
CEILING(x)	Rounds x up to the nearest integer.
COS(x)	Returns the cosine of x.
COT(x)	Returns the cotangent of x.
DEGREES(x)	Converts x from radians to degrees.
EXP(x)	Returns e raised to the power of x.
FLOOR(x)	Rounds x down to the nearest integer.
GREATEST(x1,x2,...)	Returns the greatest value among x1, x2, ...
LN(x)	Returns the natural logarithm of x.
LOG(x)	Returns the base-10 logarithm of x.
LOG10(x)	Returns the base-10 logarithm of x.
LOG2(x)	Returns the base-2 logarithm of x.
MOD(x,y)	Returns the remainder of x divided by y.
PI()	Returns π.
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,d)	Rounds x to d decimal places.
SIGN(x)	Returns the sign of x.
SIN(x)	Returns the sine of x.
SQRT(x)	Returns the square root of x.
TAN(x)	Returns the tangent of x.
TRUNCATE(x,d)	Truncates x to d decimal places.

NUMERIC FUNCTIONS

DATA TYPE	DESCRIPTION
ADDDATE(d,n)	Returns the date d plus n days.
ADDTIME(t,n)	Returns the time t plus n seconds.
CURDATE()	Returns the current date.
CURTIME()	Returns the current time.
DATE(d)	Extracts the date part of a date or datetime expression d.
DATEDIFF(d1,d2)	Returns the difference between two dates d1 and d2.
DATE_FORMAT(d,f)	Formats date d according to format f.
DAY(d)	Returns the day of the month for date d.
DAYNAME(d)	Returns the name of the weekday for date d.
DAYOFMONTH(d)	Returns the day of the month for date d.
DAYOFWEEK(d)	Returns the weekday index for date d (1 = Sunday, 2 = Monday, ...).
DAYOFYEAR(d)	Returns the day of the year for date d.
FROM_DAYS(n)	Returns a date from the number of days n since year 0.
HOUR(t)	Returns the hour for time t.
LAST_DAY(d)	Returns the last day of the month for date d.
LOCALTIME()	Returns the current date and time.
LOCALTIMESTAMP()	Returns the current date and time.
MAKEDATE(y,d)	Creates a date from the year y and day of year d.
MAKETIME(h,m,s)	Creates a time from hour h, minute m, and second s.
MICROSECOND(t)	Returns the microseconds from time t.
MINUTE(t)	Returns the minute for time t.
MONTH(d)	Returns the month for date d.
MONTHNAME(d)	Returns the name of the month for date d.
NOW()	Returns the current date and time.
PERIOD_ADD(p,n)	Returns period p plus n months.
PERIOD_DIFF(p1,p2)	Returns the difference between periods p1 and p2 in months.
QUARTER(d)	Returns the quarter of the year for date d (1-4).
SECOND(t)	Returns the second for time t.
SEC_TO_TIME(s)	Returns time from the seconds s.
STR_TO_DATE(s,f)	Converts string s to a date according to format f.
SUBDATE(d,n)	Returns the date d minus n days.
SUBTIME(t,n)	Returns the time t minus n seconds.
SYSDATE()	Returns the current date and time.
TIME(t)	Returns the time part of time t.
TIMEDIFF(t1,t2)	Returns the difference between two times t1 and t2.
TIMESTAMP(d)	Returns the date and time for date d.
TIMESTAMPADD(u,n,d)	Returns date d plus n units of time u.
TIMESTAMPDIFF(u,d1,d2)	Returns the difference between dates d1 and d2 in units of time u.
TIME_FORMAT(t,f)	Formats time t according to format f.
TIME_TO_SEC(t)	Returns the seconds from time t.
TO_DAYS(d)	Returns the number of days since year 0 for date d.
WEEK(d)	Returns the week number for date d.
WEEKDAY(d)	Returns the weekday index for date d (0 = Monday, 1 = Tuesday, ...).
WEEKOFYEAR(d)	Returns the week of the year for date d.
YEAR(d)	Returns the year for date d.
YEARWEEK(d)	Returns the year and week for date d.

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 FOREIGN KEY REFERENCES users(id),
    order_date DATE
);
CREATE TABLE products (
    id INT PRIMARY KEY,
    name VARCHAR(50),
    price DECIMAL(10,2)
);
```

EXAMPLE (MYSQL)

```
CREATE TABLE table1 (
    column1 datatype PRIMARY KEY,
    column2 datatype,
    column3 datatype
);
```

Kanak Infosystems LLP

DELETE

DATA TYPE

DATA TYPE	DESCRIPTION
DROP TABLE	Deleting a table. DROP TABLE table1;
DROP DATABASE	Deleting a database. DROP DATABASE database1;
DROP INDEX	Deleting an index. DROP INDEX index1 ON table1;
DROP COLUMN	Deleting a column in an existing table. ALTER TABLE table1 DROP COLUMN column1;
DELETE	Deleting rows from a table. DELETE FROM table1 WHERE condition1;
TRUNCATE TABLE	Deleting all rows in a table, but keeping the table structure. TRUNCATE TABLE table1;

CREATING VIEWS

```
CREATE VIEW view1 AS
SELECT column1, column2
FROM table1
WHERE condition1;
```

SELECT * FROM view1;

REPLACING VIEWS

```
CREATE OR REPLACE VIEW view1 AS
SELECT column1, column2, column3
FROM table1
WHERE condition1;
```

SELECT * FROM view1;

DELETING VIEWS

DROP VIEW view1;

JOINS

ORDERS

id	user_id	order_date	total_amount
101	1	2023-01-15	150.00
102	2	2023-01-16	200.00
103	1	2023-01-17	75.00

PRODUCTS

id	name	category	price
1	Laptop	Electronics	1200.00
2	Mouse	Electronics	25.00
3	Keyboard	Electronics	75.00

CROSS

id	name	color
1	Shirt	Red
2	Pants	Blue
3	Shoes	Black

id	name	color
1	Red Shirt	Red
1	Blue Shirt	Blue
1	Black Shirt	Black
2	Red Pants	Red
2	Blue Pants	Blue
2	Black Pants	Black
3	Red Shoes	Red
3	Blue Shoes	Blue
3	Black Shoes	Black

SELF JOIN

id	name	department	manager_id
1	John Smith	Sales	NULL
2	Jane Doe	Marketing	1
3	Bob Johnson	Sales	1
4	Alice Brown	Marketing	2

SELECT e1.name AS employee, e2.name AS manager
FROM employees e1
LEFT JOIN employees e2 ON e1.manager_id = e2.id;

Kanak Infosystems LLP