Kanak

SQL

```
SELECT * FROM [REPLACE]
WHERE [REPLACE]
ORDER BY [REPLACE]

SELECT * FROM [REPLACE]
GROUP BY [REPLACE]
HAVING [REPLACE]
```

SQL KEYWORDS

KEYWORDS	DESCRIPTION
ADD	Adds a column in an existing table, or adds a constraint on an existing table
ADD CONSTRAINT	Adds a constraint after a table is already created
ALL	Returns true if all of the subquery values meet the condition
ALTER	Adds, deletes, or modifies columns in a table, or changes the data type of a column in a table
ALTER TABLE	Adds, deletes, or modifies columns in a table, or changes the data type of a column in a table
ALTER COLUMN	Changes the data type of a column in a table
AND	Logical operator AND compares two conditions
ANY	Returns true if any of the subquery values meet the condition
AS	Renames a column or table with an alias
ASC	Sort the result set in ascending order
BETWEEN	Selects values within a given range. The values can be numbers, text, or dates
CASE	Creates different outputs based on conditions, an IF-THEN-ELSE statement
CHECK	Adds a check constraint to a column when a table is created
CREATE DATABASE	Creates a database
CREATE INDEX	Creates an index (search key)
CREATE OR REPLACE	Adds a column if it does not exist, or modifies the table if it exists
DEFAULT	Sets a default value for a column when a table is created. The default value will be added to all new records, if no other value is specified
DELETE	Deletes records in a table
DESC	Sort the result set in descending order
DISTINCT	Returns only distinct (different) values

SQL KEYWORDS

KEYWORDS	DESCRIPTION
DROP	Deletes a column in an existing table, deletes an index, or deletes a database
DROP CONSTRAINT	Deletes a constraint after a table is already created
DROP DATABASE	Deletes an existing database
DROP INDEX	Deletes an index (search key)
EXCEPT	Returns distinct values in the given table, minus those in the second table
EXISTS	Returns true if the subquery returns one or more records
FROM	Specifies the table to select or delete data from
FULL	Returns all records when there is a match in either left or right table
GROUP BY	Groups the result set by one or more columns, often used with aggregate functions (COUNT, SUM, AVG, MIN, MAX)
HAVING	Specifies a condition for a group or aggregate. Used instead of WHERE with aggregate functions
IN	Checks if a value is within a set of values or a result of a subquery
INDEX	Creates or deletes an index in a table
INNER JOIN	Returns records that have matching values in both tables
INSERT INTO	Inserts new records in a table
INTERSECT	Returns distinct values in the given table, also found in the second table
IN NULL	Returns true if a specified value is null
IS NOT NULL	Returns true if a specified value is not null
JOIN	Joins two or more tables, based on a related column between them
LEFT JOIN	Returns all records from the left table, and the matched records from the right table
LIKE	Compares a value to similar values using wildcard operator
LIMIT	Specifies the maximum number of records to return in the result set
NOT	Displays a record if the condition(s) is NOT true
NOT NULL	Specifies that a column can not contain a null value
OR	Logical operator OR compares two conditions
ORDER BY	Sort the result set in ascending or descending order
OUTER JOIN	Returns all records when there is a match in either left or right table
PRIMARY KEY	Specifies that a column can not contain values that are not unique and not null
RIGHT JOIN	Returns all records from the right table, and the matched records from the left table
ROLLUP	An extension to GROUP BY. Also calculates sub-totals and grand totals
SCHEMA_DATA	Creates a database schema
SYSTEM_DATA	Deletes an existing database schema
SELECT	Selects data from a database
SELECT *	Selects all records from a table
SELECT DISTINCT	Selects only distinct values from a table
SELECT TOP	Specifies the number of records to return in the result set
SET	Specifies which columns and values that should be updated in a table
SOME	Returns true if any of the subquery values meet the condition
TOP	Specifies the number of records to return in the result set
TRUNCATE TABLE	Deletes the data inside a table, but not the table itself
UNION	Combines the result-set of two or more SELECT statements (only distinct values)
UNION ALL	Combines the result-set of two or more SELECT statements (allows duplicate values)
UNIQUE	Specifies that all values in a column must be different
UPDATE	Updates existing records in a table
VALUES	Specifies the values to be inserted into a table
WHERE	Specifies a condition for a record or group to be selected

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 over many lines
Anything between the slash and asterisk
is a comment
*/
SELECT * FROM table1;
SELECT * FROM table2;
```

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 of data
TINYTEXT	Holds a string with a maximum length of 255 characters
TEXT(SIZE)	Holds a string with a maximum length of 65,535 characters
MEDIUMBLOB	For BLOBs (Binary Large Objects). Holds up to 16,777,215 bytes of data
MEDIUMTEXT	Holds a string with a maximum length of 16,777,215 characters
LONGBLOB	For BLOBs (Binary Large Objects). Holds up to 4,294,967,295 bytes of data
LONGTEXT	Holds a string with a maximum length of 4,294,967,295 characters
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

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 bit to 64 bits. Default is 1 bit
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)	Floating point number. MySQL uses p to determine whether to use FLOAT or DOUBLE. If p is between 0 and 24, FLOAT() is a single precision floating point number. If p is between 25 and 53, FLOAT() is a double precision floating point number
DOUBLE(m, d)	Double precision floating point number. The number of digits after the decimal point is specified in the d parameter. The maximum number of digits (m) is 65. Default is 6 digits for decimals
DECIMAL(m, d) or DEC(m, d)	Exact fixed-point number. The total number of digits is specified in m. The number of digits after the decimal point is specified in the d parameter. The maximum number of digits (m) is 65. The default value for m, when not specified, is 10. The default value for d, when not specified, is 0

DATE AND TIME DATA TYPES

DATA TYPE	DESCRIPTION
DATE	Date in format YYYY-MM-DD. Supported range is from '1000-01-01' to '9999-12-31'
DATETIME(fsp)	Date and time in 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). 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 in format HH:MM:SS. Supported range is from '-838:59:59' to '838:59:59'
YEAR	A year in four-digit format. Allowed values are from 1901 to 2155, and 0000

Kanak Infosystems LLP

OPERATORS

ARITHMETIC OPERATORS

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

BITWISE OPERATORS

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

COMPARISON OPERATORS

OPERATOR	DESCRIPTION
=	Equal to
>	Greater than
<	Less than
>=	Greater than or equal
<=	Less than or equal
<>	Not equal to. Note that in some versions of SQL this operator may be written as !=
!=	Not equal to. Note that this is not in some versions of SQL

COMPOUND OPERATORS

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

Kanak Infosystems LLP

FUNCTIONS

STRING FUNCTIONS

DATA TYPE	DESCRIPTION
ASCII(s)	Returns the ASCII value for the specified character
CHAR_LENGTH(s)	Returns the length of the string in characters
CHARACTER_LENGTH(s)	Returns the length of the string in characters
CONCAT(s1, s2, ...)	Adds two or more expressions together
CONCAT_WS(x, s1, s2, ...)	Adds two or more expressions together with a separator
FIELD(s, s1, s2, ...)	Returns the index position of a value in a list of values
FIND_IN_SET(s, s1)	Returns the position of a string within a list of strings
FORMAT(x, n)	Formats the specified number to a format like "#,###,###.##", rounded to n decimal places
INSERT(s1, x, len, s2)	Inserts a string within a string at the specified position and for a certain number of characters
INSTR(s, x)	Returns the position of the first occurrence of a string in another string
LCASE(s)	Converts a string to lower-case
LEFT(s, n)	Returns the left part of a string with the specified number of characters
LENGTH(s)	Returns the length of a string, in bytes
LOCATE(x, s, start)	Returns the position of the first occurrence of a string in another string, starting at a specified position
LOWER(s)	Converts a string to lower-case
LPAD(s, len, c)	Left-pads a string with another string, a certain number of times
LTRIM(s)	Removes leading spaces from a string
MID(s, start, len)	Extracts a substring from a string (starting at any position)
POSITION(x IN s)	Returns the position of the first occurrence of a substring in a string
REPEAT(s, n)	Repeats a string as many times as specified
REPLACE(s, from_s, to_s)	Replaces all occurrences of a substring within a string, with a new substring
REVERSE(s)	Reverses a string and returns the result
RIGHT(s, n)	Returns the right part of a string with the specified number of characters
RPAD(s, len, c)	Right-pads a string with another string, a certain number of times
RTRIM(s)	Removes trailing spaces from a string
SPACE(n)	Returns a string of the specified number of space characters
STRCMP(s1, s2)	Compares two strings
SUBSTR(s, start, len)	Extracts a substring from a string (starting at any position)
SUBSTRING(s, start, len)	Extracts a substring from a string (starting at any position)
SUBSTRING_INDEX(s, delim, n)	Returns a substring of a string before a specified number of delimiter occurrences
TRIM(s)	Removes leading and trailing spaces from a string
UCASE(s)	Converts a string to upper-case
UPPER(s)	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(y, x)	Returns the arctan2 of a pair of coordinates
CEIL(x)	Returns the ceiling value of a number
CEILING(x)	Returns the ceiling value of 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
EXP(x)	Returns e to the power of a number
FLOOR(x)	Returns the largest integer value that is smaller than, or equal to, a number
GREATEST(x1, x2, x3, ...)	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 value in degrees to radians
RAND()	Returns a random number
ROUND(x, d)	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, d)	Truncates a number to specified number of decimal places

DATE FUNCTIONS

DATA TYPE	DESCRIPTION
ADDDATE(d, INTERVAL expr type)	Adds a time/date interval to a date, then returns the date
ADDTIME(t, n)	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(d)	Extracts the date part from a datetime expression
DATEDIFF(d1, d2)	Returns the difference between two dates, in days
DATE_FORMAT(d, f)	Displays date/time data in different formats as specified by the format expression
DATE_SUB(d, INTERVAL expr type)	Subtracts a time/date interval from a date, then returns the date
SUBDATE(d, INTERVAL expr type)	Subtracts a time/date interval from a date, then returns the date
SUBTIME(t, n)	Subtracts a time interval from a time/datetime, then returns the time
SYSDATE()	Returns the current date and time
TIMEDIFF(t1, t2)	Returns the difference between two time/datetime expressions
TIMESTAMP(d, t)	Returns a datetime value based on a date or datetime value
TIMESTAMPADD(unit, interval, d)	Adds an interval to a date or datetime expression
TIMESTAMPDIFF(unit, d1, d2)	Returns the difference between two date or datetime expressions
TO_DAYS(d)	Returns the number of days between a date and year zero (0000)
WEEK(d)	Returns the week number for the given date
WEEKDAY(d)	Returns the weekday number for the given date
WEEKOFYEAR(d)	Returns the week number for the given date
YEAR(d)	Returns the year part for the given date
YEARWEEK(d)	Returns the year and week number for the given date

NUMERIC FUNCTIONS

DATA TYPE	DESCRIPTION
BINARY	Converts a value to a binary string
CAST(x AS type)	Converts a value (of any type) into specified datatype
CONVERT(x, type)	Converts a value (of any type) into specified datatype
COALESCE(x1, x2, x3, ...)	Returns the first non-null value in a list
CONNECTION_ID()	Returns the unique connection ID for the current connection
CURRENT_USER()	Returns the user name and host name for the MySQL account that the server used to authenticate the current client
DATABASE()	Returns the name of the current database
GROUP_UNIQUE()	Returns a unique string for a group within an aggregate function
HEXDIGS(x)	Returns a hexadecimal representation of a hash value
IF(expr, x, y)	Returns a value if a specified expression is true, and another value if it is false
IFNULL(x, y)	Returns a specified value if the expression is NULL, otherwise returns the expression
ISNULL(x)	Returns 1 or 0 depending on if an expression is NULL
LAST_INSERT_ID()	Returns the last auto-increment value inserted into a table in the last INSERT statement
NULLIF(x, y)	Returns NULL if two expressions are equal, otherwise returns the first expression
SESSION_USER()	Returns the current user and hostname
SYSTEM_USER()	Returns the SESSION_USER()
USER()	Returns the current MySQL user name and host name
VERSION()	Returns the version of the MySQL database

MISCELLANEOUS

NAME	DESCRIPTION
*	Selects all columns from a table. Example: SELECT * FROM table1;
.	Selects a specified table from a database. Example: SELECT * FROM database1.table1;
[ ]	Specifies an alias. Example: SELECT * FROM table1 [alias1];
?	Specifies a parameter. Example: SELECT * FROM table1 WHERE id = ?;
@	Specifies a variable. Example: SELECT @variable1 := 10;
@@	Specifies a global variable. Example: SELECT @@version;
;	Ends a statement. Example: SELECT * FROM table1;
( )	Specifies a priority for a calculation, or encloses a subquery. Example: SELECT * FROM table1 WHERE (column1 > 5 OR column1 < 2) AND column2 = 5;
,	Separates values in a list. Example: SELECT column1, column2, column3 FROM table1;
:	Specifies a variable. Example: SELECT @variable1 := 10;
|	Concatenates two strings. Example: SELECT 'string1' | 'string2';
||	Concatenates two strings. Example: SELECT 'string1' || 'string2';
~	Bitwise NOT. Example: SELECT ~ 10;
|	Bitwise OR. Example: SELECT 10 | 10;
&	Bitwise AND. Example: SELECT 10 & 10;
^	Bitwise XOR. Example: SELECT 10 ^ 10;
<<	Left shift. Example: SELECT 10 << 1;
>>	Right shift. Example: SELECT 10 >> 1;
--	Comment. Example: -- This is a comment
/*	Comment. Example: /* This is a comment */
*/	Comment. Example: /* This is a comment */
%	Modulus. Example: SELECT 10 % 3;
=	Equal to. Example: SELECT * FROM table1 WHERE column1 = 5;
:=	Assigns a value. Example: SELECT @variable1 := 10;
!	Not. Example: SELECT * FROM table1 WHERE column1 != 5;
!	Factorial. Example: SELECT 5!;
!!	Double factorial. Example: SELECT 5!!;
!~	Not match regex. Example: SELECT 'string' !~ 'regex';
!~*	Not match regex. Example: SELECT 'string' !~* 'regex';
%=	Modulus equal. Example: SELECT @variable1 %= 3;
&=	Bitwise AND equal. Example: SELECT @variable1 &= 3;
^=	Bitwise XOR equal. Example: SELECT @variable1 ^= 3;
|=	Bitwise OR equal. Example: SELECT @variable1 |= 3;
*=	Multiply equal. Example: SELECT @variable1 *= 3;
/=	Divide equal. Example: SELECT @variable1 /= 3;
+=	Add equal. Example: SELECT @variable1 += 3;
-=	Subtract equal. Example: SELECT @variable1 -= 3;
<>	Not equal. Example: SELECT * FROM table1 WHERE column1 <> 5;
<=>	Null safe equal. Example: SELECT * FROM table1 WHERE column1 <=> 5;
||/	Square root. Example: SELECT ||/ 25;
|||/	Cube root. Example: SELECT |||/ 27;
~*	Match regex. Example: SELECT * FROM table1 WHERE column1 ~* 'regex';
<<	Left shift. Example: SELECT 10 << 1;
>>	Right shift. Example: SELECT 10 >> 1;

Kanak Infosystems LLP

KEYS

FOREIGN KEY

```
EXAMPLE 1 (MYSQL):
CREATE TABLE orders (
order_id int,
customer_id int,
order_date date,
PRIMARY KEY (order_id),
FOREIGN KEY (customer_id)
REFERENCES customers(customer_id)
);

EXAMPLE 2 (MYSQL):
ALTER TABLE orders
ADD FOREIGN KEY (customer_id)
REFERENCES customers(customer_id);

EXAMPLE 3 (MYSQL):
ALTER TABLE orders
DROP FOREIGN KEY customer_id;
```

PRIMARY KEY

```
EXAMPLE 1 (MYSQL):
CREATE TABLE customers (
customer_id int,
first_name varchar(255),
last_name varchar(255),
email varchar(255),
PRIMARY KEY (customer_id)
);

EXAMPLE 2 (MYSQL):
ALTER TABLE customers
ADD PRIMARY KEY (customer_id);

EXAMPLE 3 (MYSQL):
ALTER TABLE customers
DROP PRIMARY KEY;
```

Kanak Infosystems LLP

DATA TYPES

DATA TYPE	DESCRIPTION
CREATE TABLE	Creates a table in a database. Example: CREATE TABLE table1 (column1 datatype, column2 datatype, column3 datatype);
DROP TABLE	Deletes a table, including all records and data. Example: DROP TABLE table1;
DROP DATABASE	Deletes an existing database. Example: DROP DATABASE database1;
ALTER TABLE	Adds, deletes, or modifies columns in a table, or changes the data type of a column in a table. Example: ALTER TABLE table1 ADD column1 datatype;

VIEWS

```
CREATE VIEW view1 AS
SELECT column1, column2, column3
FROM table1
WHERE condition;

SELECT * FROM view1;

DROP VIEW view1;
```

JOINS

ORDERS

id	user_id	product_id	quantity	order_date
1	1	1	1	2020-01-01
2	1	2	2	2020-01-01
3	2	1	1	2020-01-02
4	3	3	3	2020-01-03

PRODUCTS

id	name	category	price
1	product1	category1	20
2	product2	category1	30
3	product3	category2	40

USERS

id	name	email	phone
1	user1	email1@email.com	1111111111
2	user2	email2@email.com	2222222222
3	user3	email3@email.com	3333333333

CROSS

id	name	email	phone
1	user1	email1@email.com	1111111111
1	user1	email1@email.com	2222222222
1	user1	email1@email.com	3333333333
2	user2	email2@email.com	1111111111
2	user2	email2@email.com	2222222222
2	user2	email2@email.com	3333333333
3	user3	email3@email.com	1111111111
3	user3	email3@email.com	2222222222
3	user3	email3@email.com	3333333333

SELF JOIN

id	name	manager_id
1	user1	3
2	user2	3
3	user3	NULL

```
SELECT a.name, b.name
FROM table1 a, table1 b
WHERE a.manager_id = b.id;
```

NATURAL JOIN

```
SELECT * FROM table1
NATURAL JOIN table2;
```

CROSS JOIN

```
SELECT * FROM table1
CROSS JOIN table2;
```

SELF JOIN

```
SELECT * FROM table1 a, table1 b
WHERE a.column1 = b.column1
AND a.column2 = b.column2
AND a.column3 = b.column3;
```

Kanak Infosystems LLP

INDEXES

```
CREATE INDEX idx_name ON table1 (column1);

CREATE UNIQUE INDEX idx_name ON table1 (column1, column2);

DROP INDEX idx_name ON table1;

SHOW INDEX FROM table1;
```

ALTER TABLE

```
ALTER TABLE table1 ADD column1 datatype;

ALTER TABLE table1 DROP COLUMN column1;

ALTER TABLE table1 RENAME column1 TO column2;

ALTER TABLE table1 MODIFY column1 datatype;

ALTER TABLE table1 ADD CONSTRAINT constraint1 PRIMARY KEY (column1);

ALTER TABLE table1 DROP PRIMARY KEY;

ALTER TABLE table1 ADD CONSTRAINT constraint1 FOREIGN KEY (column1) REFERENCES table2(column2);

ALTER TABLE table1 DROP FOREIGN KEY constraint1;

ALTER TABLE table1 ADD CONSTRAINT constraint1 CHECK (column1 > 5);

ALTER TABLE table1 DROP CHECK constraint1;

ALTER TABLE table1 ALTER column1 SET DEFAULT 5;

ALTER TABLE table1 ALTER column1 DROP DEFAULT;
```

TRUNCATE TABLE

```
TRUNCATE TABLE table1;
```

SELECT

```
SELECT column1, column2, column3
FROM table1;

SELECT * FROM table1;

SELECT DISTINCT column1, column2, column3
FROM table1;

SELECT TOP 10 column1, column2, column3
FROM table1;

SELECT column1, column2, column3
FROM table1
WHERE condition;

SELECT column1, column2, column3
FROM table1
ORDER BY column1 ASC, column2 DESC;

SELECT column1, column2, column3
FROM table1
GROUP BY column1, column2
HAVING condition;

SELECT column1, column2, column3
FROM table1
WHERE condition1
UNION [ALL]
SELECT column1, column2, column3
FROM table2
WHERE condition2;

SELECT column1, column2, column3
FROM table1
INNER JOIN table2
ON table1.column1 = table2.column1;

SELECT column1, column2, column3
FROM table1
LEFT JOIN table2
ON table1.column1 = table2.column1;

SELECT column1, column2, column3
FROM table1
RIGHT JOIN table2
ON table1.column1 = table2.column1;

SELECT column1, column2, column3
FROM table1
FULL JOIN table2
ON table1.column1 = table2.column1;

SELECT column1, column2, column3
FROM table1
CROSS JOIN table2;

SELECT column1, column2, column3
FROM table1
NATURAL JOIN table2;

SELECT column1, column2, column3
FROM table1 a, table1 b
WHERE a.column1 = b.column1
AND a.column2 = b.column2
AND a.column3 = b.column3;

SELECT column1, column2, column3
FROM table1
WHERE column1 IN (SELECT column1 FROM table2);

SELECT column1, column2, column3
FROM table1
WHERE column1 BETWEEN 5 AND 10;

SELECT column1, column2, column3
FROM table1
WHERE column1 LIKE 'pattern';

SELECT column1, column2, column3
FROM table1
WHERE column1 IS NULL;

SELECT column1, column2, column3
FROM table1
WHERE column1 IS NOT NULL;

SELECT column1, column2, column3
FROM table1
WHERE column1 = 5
AND column2 = 'string'
OR column3 = 'string';

SELECT column1, column2, column3
FROM table1
WHERE NOT column1 = 5;

SELECT column1, column2, column3
FROM table1
ORDER BY column1 ASC, column2 DESC;

SELECT column1, column2, column3
FROM table1
GROUP BY column1, column2
HAVING condition;

SELECT column1, column2, column3
FROM table1
GROUP BY ROLLUP(column1, column2);

SELECT column1, column2, column3
FROM table1
GROUP BY GROUPING SETS(column1, column2);

SELECT column1, column2, column3
FROM table1
GROUP BY CUBE(column1, column2);

SELECT column1, column2, column3
FROM table1
GROUP BY column1, column2
HAVING GROUPING(column1) = 1;

SELECT column1, column2, column3
FROM table1
WHERE column1 = (SELECT column1 FROM table2);

SELECT column1, column2, column3
FROM table1
WHERE column1 > ANY (SELECT column1 FROM table2);

SELECT column1, column2, column3
FROM table1
WHERE column1 > ALL (SELECT column1 FROM table2);

SELECT column1, column2, column3
FROM table1
WHERE column1 = CASE
WHEN condition1 THEN 1
WHEN condition2 THEN 2
WHEN condition3 THEN 3
ELSE 4
END;

SELECT column1, column2, column3
FROM table1
WHERE column1 = IFNULL(column2, 5);

SELECT column1, column2, column3
FROM table1
WHERE column1 = COALESCE(column2, column3, 5);

SELECT column1, column2, column3
FROM table1
WHERE column1 = CAST(column2 AS datatype);

SELECT column1, column2, column3
FROM table1
WHERE column1 = CONVERT(column2, datatype);

SELECT column1, column2, column3
FROM table1
WHERE column1 = IF(condition, column2, column3);

SELECT column1, column2, column3
FROM table1
WHERE column1 = NULLIF(column2, column3);

SELECT column1, column2, column3
FROM table1
WHERE column1 = (SELECT column1 FROM table2 LIMIT 1);

SELECT column1, column2, column3
FROM table1
WHERE column1 = (SELECT column1 FROM table2 LIMIT 1 OFFSET 2);

SELECT column1, column2, column3
FROM table1
WHERE column1 = (SELECT column1 FROM table2 ORDER BY column1 DESC LIMIT 1);

SELECT column1, column2, column3
FROM table1
WHERE column1 = (SELECT column1 FROM table2 ORDER BY column1 DESC LIMIT 1 OFFSET 2);

SELECT column1, column2, column3
FROM table1
WHERE column1 = (SELECT column1 FROM table2 WHERE condition);

SELECT column1, column2, column3
FROM table1
WHERE column1 = (SELECT column1 FROM table2 WHERE condition ORDER BY column1 DESC LIMIT 1);

SELECT column1, column2, column3
FROM table1
WHERE column1 = (SELECT column1 FROM table2 WHERE condition ORDER BY column1 DESC LIMIT 1 OFFSET 2);

SELECT column1, column2, column3
FROM table1
WHERE column1 = (SELECT column1 FROM table2 WHERE condition
```