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 a table is already created, such as to make a column unique |
| ALTER COLUMN | Changing the data type of a column in a table, or adding/removing a default constraint |
| ALTER TABLE | Adding, deleting, or modifying columns in a table, or adding and dropping various constraints on an existing table |
| ALTER VIEW | Updating a view that was previously created by the CREATE VIEW statement |
| AND | Used in a WHERE clause to filter data based on more than one condition |
| AS | Renaming a table or column temporarily for the duration of a query |
| ASC | Sorting the result set in ascending order (default) |
| AVG | Returns the average value of a numeric column |
| BETWEEN | Used in a WHERE clause to filter data based on a range of values |
| BIGINT | Used in a CREATE TABLE statement to create a column that can store 8-byte integers |
| BIT | Used in a CREATE TABLE statement to create a column that can store a bit-wise value |
| CASE | Used in a SELECT statement to return different values based on conditions, similar to an IF-THEN-ELSE statement |
| CHECK | Used in a CREATE TABLE statement to create a constraint that limits the values that can be stored in a column |
| CREATE DATABASE | Creating a database |
| CREATE INDEX | Creating an index on a table, allowing for faster retrieval of data |
| CREATE TABLE | Creating a table in a database |
| DELETE | Deleting rows from a table |
| DESC | Sorting the result set in descending order |
| DISTINCT | Returns only unique values in a column |

## SQL KEYWORDS

| KEYWORDS | DESCRIPTION |
|---|---|
| DROP | Deleting a table, view, or database |
| DROP CONSTRAINT | Removing a constraint from a table |
| DROP COLUMN | Deleting a column in a table |
| EXEC | Executes the SQL statements within a stored procedure |
| EXISTS | Used in a WHERE clause to check if a subquery returns any rows |
| FROM | Specifies the table from which data is to be retrieved or deleted |
| FULL | Returns all rows when there is a match in either table in a JOIN operation |
| GO | Signals the end of a batch of SQL statements to be executed |
| GROUP BY | Used in a SELECT statement to group rows with similar values in a column |
| HAVING | Used in a SELECT statement to filter groups created by the GROUP BY clause, similar to WHERE but for groups |
| IN | Used in a WHERE clause to filter data based on a set of values |
| INDEX | Used in a CREATE TABLE statement to create an index on a column |
| INNER JOIN | Returns rows when there is a match in both tables in a JOIN operation |
| INSERT INTO | Inserting new rows into a table |
| INT | Used in a CREATE TABLE statement to create a column that can store 4-byte integers |
| IS NULL | Used in a WHERE clause to filter data based on whether a column is null or not |
| IS NOT NULL | Used in a WHERE clause to filter data based on whether a column is not null |
| JOIN | Combines rows from two or more tables based on a related column |
| LEFT JOIN | Returns all rows from the left table in a JOIN operation, even if there is no match in the right table |
| LIKE | Used in a WHERE clause to filter data based on a pattern |
| LIMIT | Limits the number of rows returned by a query |
| MAX | Returns the maximum value in a numeric column |
| MIN | Returns the minimum value in a numeric column |
| NOT | Used in a WHERE clause to negate a condition |
| NULL | Represents a missing or unknown value in a column |
| OFFSET | Used in conjunction with LIMIT to skip a specified number of rows before returning the result set |
| OR | Used in a WHERE clause to filter data based on more than one condition |
| ORDER BY | Used in a SELECT statement to sort the result set based on one or more columns |
| OUTER JOIN | Returns all rows when there is a match in either table in a JOIN operation, even if there is no match in the other table |
| PRIMARY KEY | Used in a CREATE TABLE statement to create a column that uniquely identifies each row in the table |
| RIGHT JOIN | Returns all rows from the right table in a JOIN operation, even if there is no match in the left table |
| SELECT | Used to retrieve data from a database |
| SELECT DISTINCT | Returns only unique values in a column, eliminating duplicate rows |
| SELECT TOP | Limits the number of rows returned by a query, similar to LIMIT in other DBMS systems |
| SET | Used in an UPDATE statement to specify the new values for the columns being updated |
| SUM | Returns the sum of values in a numeric column |
| TOP | Limits the number of rows returned by a query, similar to LIMIT in other DBMS systems |
| TRUNCATE TABLE | Deleting all rows in a table, but not the table itself |
| UNION | Combines the results of two or more SELECT statements into a single result set |
| UNIQUE | Used in a CREATE TABLE statement to create a column that ensures all values in the column are unique |
| UPDATE | Updating existing rows in a table |
| VALUES | Used in an INSERT INTO statement to specify the values to be inserted into a table |
| WHERE | Used to filter data based on a condition |
| WHERE EXISTS | Used in a WHERE clause to filter data based on whether a subquery returns any rows |

Kanak Infosystems LLP

## COMMENTS

### SINGLE LINE COMMENTS

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

### MULTILINE COMMENTS

```
/* This is a multiline comment
that can span
several lines and is useful for
longer explanations */
SELECT * FROM table1;
```

## 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 bytes of data |
| 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 default value for SIZE is 1. Ranges from 1 to 64 |
| 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 between 25 and 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 |

### 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. Adding optional fsp parameter allows for up to six digits for fractional seconds precision |
| TIMESTAMP(fsp) | Stores the number of seconds since the Unix epoch (1970-01-01 00:00:00 UTC). Adding optional fsp parameter allows for up to six digits for fractional seconds precision |
| TIME(fsp) | Time in format HH:MM:SS. Adding optional fsp parameter allows for up to six digits for fractional seconds precision |
| YEAR | A year in format yyyy or yy. Can be 2 or 4 digits long. |

Kanak Infosystems LLP

## OPERATORS

### ARITHMETIC OPERATORS

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

### 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 |

### BITWISE OPERATORS

| OPERATOR | DESCRIPTION |
|---|---|
| & | Bitwise AND |
| \| | Bitwise OR |
| ^ | Bitwise XOR |
| ~ | Bitwise not (unary) |
| << | Bitwise shift left |
| >> | Bitwise shift right |

### COMPOUND OPERATOR

| 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 x |
| CHAR_LENGTH(x) | Returns the length of the string x in characters |
| CHR(x) | Returns the character from the specified ASCII value x |
| CONCAT(x, y) | Concatenates two strings x and y |
| CONCAT_WS(x, y, z) | Concatenates strings x, y, and z with a separator x |
| FIELD(x, y, z) | Returns the index position of x in the list y, z |
| FORMAT(x, y) | Formats the number x to a format like #,###,###.## rounded to y decimal places |
| INSERT(x, y, z, w) | Inserts string w into string x, starting at position y, replacing z characters |
| LCASE(x) | Converts string x to lowercase |
| LEFT(x, y) | Returns the leftmost y characters from string x |
| LENGTH(x) | Returns the length of the string x in bytes |
| LOCATE(x, y, z) | Returns the position of x in y, starting search at position z |
| LOWER(x) | Converts string x to lowercase |
| LPAD(x, y, z) | Pads string x on the left with string z to a total length of y characters |
| LTRIM(x) | Removes leading spaces from string x |
| MID(x, y, z) | Extracts z characters from string x, starting at position y |
| POSITION(x IN y) | Returns the position of x in y |
| REPEAT(x, y) | Repeats string x y times |
| REPLACE(x, y, z) | Replaces all occurrences of y in x with z |
| REVERSE(x) | Reverses string x |
| RIGHT(x, y) | Returns the rightmost y characters from string x |
| RPAD(x, y, z) | Pads string x on the right with string z to a total length of y characters |
| RTRIM(x) | Removes trailing spaces from string x |
| SPACE(x) | Returns a string with x spaces |
| STRCMP(x, y) | Compares string x and y |
| SUBSTR(x, y, z) | Extracts z characters from string x, starting at position y |
| SUBSTRING(x, y, z) | Extracts z characters from string x, starting at position y |
| SUBSTRING_INDEX(x, y, z) | Returns a substring of x before y occurrences of delimiter z |
| TRIM(x) | Removes leading and trailing spaces from string x |
| UCASE(x) | Converts string x to uppercase |
| UPPER(x) | Converts string x 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 arctan of x |
| ATAN2(x, y) | Returns the arctan2 of x and y |
| AVG(x) | Returns the average value of expression x |
| CEIL(x) | Returns the ceiling value of x |
| CEIL(x, y) | Returns the ceiling value of x, rounded to y decimal places |
| COS(x) | Returns the cosine of x |
| COT(x) | Returns the cotangent of x |
| COUNT(x) | Returns the number of records in expression x |
| DIV(x, y) | Returns the integer quotient of x divided by y |
| EXP(x) | Returns e to the x power |
| FLOOR(x) | Returns the largest integer value not greater than x |
| GREATEST(x, y, z) | Returns the greatest value of the list x, y, z |
| 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 |
| MAX(x) | Returns the maximum value in expression x |
| MIN(x) | Returns the minimum value in expression x |
| MOD(x, y) | Returns x % y |
| PI() | Returns PI |
| POW(x, y) | Returns x to the y power |
| POWER(x, y) | Returns x to the y power |
| RADIANS(x) | Converts x from degrees to radians |
| RAND() | Returns a random number |
| ROUND(x) | Rounds x to the nearest integer |
| ROUND(x, y) | Rounds x to y decimal places |
| SIGN(x) | Returns the sign of x (-, 0, +) |
| SIN(x) | Returns the sine of x |
| SQRT(x) | Returns the square root of x |
| SUM(x) | Returns the sum of expression x |
| TAN(x) | Returns the tangent of x |
| TRUNCATE(x, y) | Returns x truncated to y decimal places |

### NUMERIC FUNCTIONS

| DATA TYPE | DESCRIPTION |
|---|---|
| ADDDATE(x, y) | Returns the date x and time y added together |
| ADDTIME(x, y) | Returns the time x and time y added together |
| CURDATE() | Returns the current date |
| CURRENT_DATE() | Returns the current date |
| CURRENT_TIME() | Returns the current time |
| CURTIME() | Returns the current time |
| DATE(x) | Returns the date part of date x |
| DATE_FORMAT(x, y) | Formats date x according to the specified format y |
| DATE_SUB(x, y) | Subtracts time y from date x |
| DAY(x) | Returns the day of the month from date x |
| DAYNAME(x) | Returns the name of the weekday for date x |
| DAYOFMONTH(x) | Returns the day of the month for date x |
| DAYOFWEEK(x) | Returns the weekday index for date x |
| DAYOFYEAR(x) | Returns the day of the year for date x |
| EXTRACT(x FROM y) | Extracts x from date y |
| FROM_DAYS(x) | Returns the date value given the specified number of days x |
| HOUR(x) | Returns the hour for time x |
| LAST_DAY(x) | Returns the last day of the month for the specified date x |
| LOCALTIME() | Returns the current date and time |
| LOCALTIMESTAMP() | Returns the current date and time |
| MAKEDATE(x, y) | Returns a date given the year x and a number of days y |
| MAKETIME(x, y, z) | Returns a time value based on the hour x, minute y, and second z |
| MICROSECOND(x) | Returns the microsecond for time x |
| MINUTE(x) | Returns the minute for time x |
| MONTH(x) | Returns the month for date x |
| MONTHNAME(x) | Returns the name of the month for date x |
| NOW() | Returns the current date and time |
| PERIOD_ADD(x, y) | Returns a period x added by y months |
| PERIOD_DIFF(x, y) | Returns the difference between periods x and y in months |
| QUARTER(x) | Returns the quarter for date x |
| SECOND(x) | Returns the second for time x |
| SEC_TO_TIME(x) | Returns time in hh:mm:ss format for x seconds |
| STR_TO_DATE(x, y) | Returns a date based on string x and format y |
| SUBDATE(x, y) | Returns date x minus y days |
| SUBTIME(x, y) | Returns time x minus y |
| SYSDATE() | Returns the current date and time |
| TIME(x) | Returns the time part of time x |
| TIME_FORMAT(x, y) | Formats time x according to the specified format y |
| TIME_TO_SEC(x) | Returns time x in seconds |
| TIMEDIFF(x, y) | Returns the difference between time x and time y |
| TIMESTAMP(x, y) | Returns date x and time y added together |
| TO_DAYS(x) | Returns the number of days between date x and year 0 |
| WEEK(x) | Returns the week number for date x |
| WEEKDAY(x) | Returns the weekday index for date x |
| WEEKOFYEAR(x) | Returns the week number for date x |
| YEAR(x) | Returns the year for date x |
| YEARWEEK(x) | Returns the year and week number for date x |

Kanak Infosystems LLP

## KEYS

### FOREIGN KEY

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

CREATE TABLE orders
(order_id INT PRIMARY KEY,
user_id INT,
order_date DATE,
FOREIGN KEY (user_id) REFERENCES users(user_id));

CREATE TABLE products
(product_id INT PRIMARY KEY,
name VARCHAR(255),
price DECIMAL(10,2));
```

### PRIMARY KEY

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

Kanak Infosystems LLP

## DATA TYPES

| DATA TYPE | DESCRIPTION |
|---|---|
| CREATE TABLE | Creates a table in a database. Specifies the table name and defines its columns and their data types. |
| CHAR(x) | A fixed length string (can contain letters, numbers, and special characters). The parameter x specifies the column length in characters - can be from 0 to 255. Default is 1 |
| VARCHAR(x) | A variable length string (can contain letters, numbers, and special characters). The parameter x specifies the maximum column length in characters - can be from 0 to 65535 |
| BINARY(x) | Equal to CHAR(), but stores binary byte strings. The parameter x specifies the column length in bytes. Default is 1 |
| VARBINARY(x) | Equal to VARCHAR(), but stores binary byte strings. The parameter x 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(x) | Holds a string with a maximum length of 65,535 bytes of data |
| 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. |
| BIT(x) | A bit-value type. The number of bits per value is specified in x. Ranges from 1 to 64 |
| TINYINT(x) | A very small integer. Signed range is from -128 to 127. Unsigned range is from 0 to 255. The parameter x specifies the maximum display width (which is 255) |
| BOOL or BOOLEAN | Zero is considered as false, non-zero values are considered as true. |
| SMALLINT(x) | A small integer. Signed range is from -32768 to 32767. Unsigned range is from 0 to 65535. The parameter x specifies the maximum display width (which is 255) |
| MEDIUMINT(x) | A medium integer. Signed range is from -8388608 to 8388607. Unsigned range is from 0 to 16777215. The parameter x specifies the maximum display width (which is 255) |
| INT(x) | A medium integer. Signed range is from -2147483648 to 2147483647. Unsigned range is from 0 to 4294967295. The parameter x specifies the maximum display width (which is 255) |
| BIGINT(x) | A large integer. Signed range is from -9223372036854775808 to 9223372036854775807. Unsigned range is from 0 to 18446744073709551615. The parameter x 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 between 25 and 53, the data type becomes DOUBLE() |
| DOUBLE(x, d) | A floating point number. The total number of digits is specified in x. The number of digits after the decimal point is specified in the d parameter |
| DECIMAL(x, d), NUMERIC(x, d) | Stores a fixed point number. The total number of digits is specified in x. The number of digits after the decimal point is specified in the d parameter |

Kanak Infosystems LLP

## INSERT

```
INSERT INTO table1
(name, age, city)
VALUES
('John', 25, 'New York');
```

## UPDATE

```
UPDATE table1
SET age = 26
WHERE name = 'John';
```

## DELETE

```
DELETE FROM table1
WHERE name = 'John';
```

## JOINS

## ORDERS

| id | user_id | product_id | quantity | order_date |
|---|---|---|---|---|
| 1 | 1 | 101 | 2 | 2023-01-15 |
| 2 | 2 | 102 | 1 | 2023-01-16 |
| 3 | 1 | 103 | 3 | 2023-01-17 |
| 4 | 3 | 101 | 1 | 2023-01-18 |

## PRODUCTS

| id | name | category | price |
|---|---|---|---|
| 101 | Laptop | Electronics | 1200.00 |
| 102 | Mouse | Electronics | 25.00 |
| 103 | Keyboard | Electronics | 75.00 |
| 104 | Monitor | Electronics | 300.00 |

## INDEX

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

CREATE UNIQUE INDEX idx_email
ON table1 (email);

DROP INDEX idx_name;
```

## VIEW

```
CREATE VIEW view_name
AS
SELECT * FROM table1
WHERE condition;

SELECT * FROM view_name;

DROP VIEW view_name;
```

## TRANSACTION

```
BEGIN TRANSACTION;
UPDATE table1
SET column1 = value1
WHERE condition;
COMMIT;

BEGIN TRANSACTION;
DELETE FROM table1
WHERE condition;
ROLLBACK;
```

## CREATE DATABASE

```
CREATE DATABASE database_name;

USE database_name;

DROP DATABASE database_name;
```

## CREATE TABLE

```
CREATE TABLE table1
(id INT PRIMARY KEY,
name VARCHAR(255),
email VARCHAR(255),
age INT,
city VARCHAR(255));

CREATE TABLE table2
(id INT PRIMARY KEY,
user_id INT,
product_id INT,
quantity INT,
order_date DATE);
```

## ALTER TABLE

```
ALTER TABLE table1
ADD column1 INT;

ALTER TABLE table1
DROP COLUMN column1;

ALTER TABLE table1
ALTER COLUMN column1 VARCHAR(255);

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

ALTER TABLE table1
DROP CONSTRAINT constraint_name;
```

## ORDERS

| id | user_id | product_id | quantity | order_date |
|---|---|---|---|---|
| 1 | 1 | 101 | 2 | 2023-01-15 |
| 2 | 2 | 102 | 1 | 2023-01-16 |
| 3 | 1 | 103 | 3 | 2023-01-17 |
| 4 | 3 | 101 | 1 | 2023-01-18 |

## PRODUCTS

| id | name | category | price |
|---|---|---|---|
| 101 | Laptop | Electronics | 1200.00 |
| 102 | Mouse | Electronics | 25.00 |
| 103 | Keyboard | Electronics | 75.00 |
| 104 | Monitor | Electronics | 300.00 |

## SELECT

```
SELECT * FROM table1;

SELECT column1, column2 FROM table1;

SELECT DISTINCT column1 FROM table1;

SELECT TOP 10 * FROM table1;

SELECT * FROM table1
WHERE condition;

SELECT * FROM table1
ORDER BY column1 ASC;

SELECT * FROM table1
ORDER BY column1 DESC;

SELECT * FROM table1
GROUP BY column1;

SELECT * FROM table1
HAVING condition;

SELECT * FROM table1
LIMIT 10;

SELECT * FROM table1
OFFSET 10 ROWS;

SELECT * FROM table1
WHERE column1 IN (value1, value2, value3);

SELECT * FROM table1
WHERE column1 BETWEEN value1 AND value2;

SELECT * FROM table1
WHERE column1 LIKE 'pattern';

SELECT * FROM table1
WHERE column1 IS NULL;

SELECT * FROM table1
WHERE column1 IS NOT NULL;

SELECT * FROM table1
WHERE EXISTS (SELECT * FROM table2 WHERE condition);

SELECT * FROM table1
WHERE column1 = (SELECT column1 FROM table2 WHERE condition);

SELECT * FROM table1
WHERE column1 > (SELECT AVG(column1) FROM table2);

SELECT * FROM table1
WHERE column1 IN (SELECT column1 FROM table2);

SELECT * FROM table1
WHERE column1 NOT IN (SELECT column1 FROM table2);

SELECT * FROM table1
WHERE column1 = ANY (SELECT column1 FROM table2);

SELECT * FROM table1
WHERE column1 = ALL (SELECT column1 FROM table2);

SELECT * FROM table1
WHERE column1 > ANY (SELECT column1 FROM table2);

SELECT * FROM table1
WHERE column1 > ALL (SELECT column1 FROM table2);

SELECT * FROM table1
WHERE column1 = (SELECT column1 FROM table2 WHERE condition);

SELECT * FROM table1
WHERE column1 IN (SELECT column1 FROM table2);

SELECT * FROM table1
WHERE column1 NOT IN (SELECT column1 FROM table2);

SELECT * FROM table1
WHERE column1 = ANY (SELECT column1 FROM table2);

SELECT * FROM table1
WHERE column1 = ALL (SELECT column1 FROM table2);

SELECT * FROM table1
WHERE column1 > ANY (SELECT column1 FROM table2);

SELECT * FROM table1
WHERE column1 > ALL (SELECT column1 FROM table2);
```

## UNION

```
SELECT column1 FROM table1
UNION
SELECT column1 FROM table2;

SELECT column1 FROM table1
UNION ALL
SELECT column1 FROM table2;

SELECT column1 FROM table1
INTERSECT
SELECT column1 FROM table2;

SELECT column1 FROM table1
EXCEPT
SELECT column1 FROM table2;
```

Kanak Infosystems LLP

www.kanakinfosystems.com