Kanak

SQL

```
SELECT * FROM table;
SELECT * FROM table WHERE
condition;
SELECT * FROM table
ORDER BY column;
SELECT * FROM table
GROUP BY column;
```

SQL KEYWORDS

KEYWORDS	DESCRIPTION
ADD	Adds a column in an existing table, adds a constraint after a table has been created
ADD CONSTRAINT	Creates a constraint after a table has been created
ALL	Returns true if all of the subqueries values meet the condition
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	Logically AND operator. Displays a record if both the first condition and the second condition are true
ANY	Returns true if any of the subqueries values meet the condition
AS	Renames a column or table with an alias
ASC	Sorts the result set in ascending order
BACKUP DATABASE	Creates a back-up of a database
BETWEEN	Selects values within a given range. Inclusive of the begin and end values
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 or altered
CREATE DATABASE	Creates a database
CREATE INDEX	Creates an index (allow duplicate values)
DEFAULT	Sets a default value for a column when a table is created. Returns the default value if no other value is assigned
DELETE	Deletes rows from a table
DESC	Sorts the result set in descending order

SQL KEYWORDS

KEYWORDS	DESCRIPTION
DISTINCT	Selects only distinct values
DROP	Deletes a database, table, index, or view
DROP COLUMN	Deletes a column in a table
DROP CONSTRAINT	Deletes a constraint
EXEC	Executes a function or a stored procedure
EXISTS	Returns true if a subquery contains any records
FROM	Specifies which table to select or delete 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
HAVING	Specifies a condition for a group or aggregate. Used instead of WHERE with aggregate functions
IN	Checks if a value is in a set of values
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
IS NULL	Tests for empty values
IS NOT NULL	Tests for non-empty values
JOIN	Joins tables or views
LEFT	Returns all records from the left table, and the matched records from the right table
LIKE	Searches for a specified pattern in a column
LIMIT	Specifies the number of records to return in the result set
NOT	Displays a record if the condition(s) is NOT true
OR	Logical OR operator. Displays a record if any of the conditions are true
ORDER BY	Sorts 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	Creates a primary key when a table is created or altered
PROCEDURE	Creates a stored procedure
RIGHT	Returns all records from the right table, and the matched records from the left table
ROWNUM	Specifies the number of records in an operation
SELECT	Selects data from a database
SELECT DISTINCT	Selects only distinct values
SELECT TOP	Specifies the number of records to return in the result set
SET	Specifies which values or columns are to be updated in a table
SOME	Returns true if any of the subqueries 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 (removes duplicate values)
UNION ALL	Combines the result set of two or more SELECT statements (allows duplicate values)
UNIQUE	Creates a unique constraint for a column when a table is created or altered
UPDATE	Updates existing data in a table
VALUES	Specifies the values for an INSERT INTO statement
WHERE	Specifies a condition while fetching the data from a table or joining with other tables

Kanak Infosystems LLP.

COMMENTS

SINGLE LINE COMMENTS

```
-- Select all:
SELECT * FROM city
```

MULTILINE COMMENTS

```
/* Select all
the records
in the table
-- this is just a regular comment inside a multiline comment
#define DEBUG_MODE 1
*/
SELECT * FROM table;
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). Specify the string length in parenthesis. Can store up to 255 characters. Default size is 1
VARCHAR(SIZE)	A VARIABLE length string (can contain letters, numbers, and special characters). Specify the maximum string length in parenthesis. Can store up to 255 characters. Note: If you put more than 255 characters in a VARCHAR column, it will be converted to a TEXT type
TINYTEXT	A string with a maximum length of 255 characters
TEXT(SIZE)	A string with a maximum length of 65,535 characters
MEDIUMTEXT	A string with a maximum length of 16,777,215 characters
LONGTEXT	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
BINARY(SIZE)	Equal to CHAR(), but stores binary byte strings. Specify the string length in parenthesis. Default size is 1
VARBINARY(SIZE)	Equal to VARCHAR(), but stores binary byte strings. Specify the maximum string length in parenthesis
TINYBLOB	A BLOB (Binary Large Object) with a maximum length of 255 bytes
BLOB(SIZE)	A BLOB (Binary Large Object) with a maximum length of 65,535 bytes
MEDIUMBLOB	A BLOB (Binary Large Object) with a maximum length of 16,777,215 bytes
LONGBLOB	A BLOB (Binary Large Object) with a maximum length of 4,294,967,295 bytes

NUMERIC DATA TYPES

DATA TYPE	DESCRIPTION
BIT(SIZE)	A bit-value type. Number of bits per value specified in parenthesis, from 1 to 64. The default value for SIZE 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)
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)	A floating point number. MySQL uses the p value to determine whether to use FLOAT or DOUBLE for the resulting data type. If p is from 0 to 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 number of digits may be specified in size. The number of digits after the decimal point is specified in the d parameter
DECIMAL(size, d), NUMERIC(size, d)	An exact fixed-point number. The number of digits may be 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. Format: YYYY-MM-DD
DATETIME(fsp)	Date and time combination. Format: YYYY-MM-DD hh:mm:ss. Adding fsp (fractional seconds precision) specifies the precision of fractional seconds, from 0 to 6
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. Adding fsp (fractional seconds precision) specifies the precision of fractional seconds, from 0 to 6
TIME(fsp)	Time. Format: hh:mm:ss. Adding fsp (fractional seconds precision) specifies the precision of fractional seconds, from 0 to 6
YEAR	A year in four-digit format (4 digits). Allowed values are from 1901 to 2155, and 0000

Kanak Infosystems LLP.

COMPARISON OPERATORS

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

COMPOUND OPERATORS

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

Kanak Infosystems LLP.

LOGICAL OPERATORS

OPERATOR	DESCRIPTION
ALL	True if all of the subquery values meet the condition
AND	True if all the conditions separated by AND is true
ANY	True if any of the subquery 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 subquery values meet the condition

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, ...)	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 value x in the list of values y, ...
FIND_IN_SET(x, y)	Returns the position of a value x in a list of values y, separated by a symbol
FORMAT(x, y)	Formats the specified value x to specified format y
INSERT(x, y, z, w)	Inserts a string w into string x, at position y, for z characters
INSTR(x, y)	Returns the first position of the string y in string x
LCASE(x)	Converts a string x into lower-case
LEFT(x, y)	Returns the left part of a string x with the specified number of characters y
LENGTH(x)	Returns the length of the string x in bytes
LOCATE(x, y, z)	Returns the position of a value x in a string y, starting at position z
LPAD(x, y, z)	Pads a string x from the left, with string z, to a length of y characters
LTRIM(x)	Removes leading spaces from a string x
MID(x, y, z)	Extracts a substring from a string x (starting at position y, extending z characters)
POSITION(x IN y)	Returns the position of the first occurrence of value x in string y
REPEAT(x, y)	Repeats a string x a specified number of times y
REPLACE(x, y, z)	Replaces all occurrences of a substring y within a string x, with a new substring z
REVERSE(x)	Reverses a string x and returns the result
RIGHT(x, y)	Returns the right part of a string x with the specified number of characters y
RPAD(x, y, z)	Pads a string x from the right, with string z, to a length of y characters
RTRIM(x)	Removes trailing spaces from a string x
SPACE(x)	Returns a string consisting of x spaces
STRCMP(x, y)	Compares two strings x and y
SUBSTR(x, y, z), SUBSTRING(x, y, z)	Returns a substring of a string x, starting at position y, extending z characters
SUBSTRING_INDEX(x, y, z)	Returns a substring of a string x before z occurrences of the delimiter y
TRIM([char FROM] x)	Removes leading and trailing spaces, or specified characters, from a string x
UCASE(x)	Converts a string x into upper-case
UPPER(x)	Converts a string x into upper-case

NUMERIC FUNCTIONS

DATA TYPE	DESCRIPTION
ABS(x)	Returns the absolute value of a number x
ACOS(x)	Returns the arccosine of x
ASIN(x)	Returns the arcsine of x
ATAN(x)	Returns the arctangent of x
ATAN2(x, y)	Returns the arctangent of the two variables x and y
AVG(x)	Returns the average value of the expression x
CEIL(x)	Returns the smallest integer value that is bigger than or equal to the specified numeric value x
CEILING(x)	Returns the smallest integer value that is bigger than or equal to the specified numeric value x
COS(x)	Returns the cosine of a given number x
COT(x)	Returns the cotangent of a given number x
COUNT(x)	Returns the number of records in a table
DIV(x, y)	Returns the integer quotient of the division of x by y
EXP(x)	Returns e raised to the power of x
FLOOR(x)	Returns the largest integer value that is smaller than or equal to the specified numeric value x
GREATEST(x, y, ...)	Returns the greatest value of the list of arguments
LN(x)	Returns the natural logarithm of a number x
LOG(x)	Returns the natural logarithm of a number x
LOG10(x)	Returns the base-10 logarithm of a number x
LOG2(x)	Returns the base-2 logarithm of a number x
MAX(x)	Returns the maximum value in a set of values x
MIN(x)	Returns the minimum value in a set of values x
MOD(x, y)	Returns the remainder of x divided by y
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 a degree value x into radians
RAND()	Returns a random number
ROUND(x, y)	Rounds a number x to y decimal places
SIGN(x)	Returns the sign of a given number x, indicating whether x is positive, negative or zero
SIN(x)	Returns the sine of a given number x
SQRT(x)	Returns the square root of a given number x
SUM(x)	Returns the total sum of a set of values x
TAN(x)	Returns the tangent of a given number x
TRUNCATE(x, y)	Truncates a number x to y decimal places

DATE FUNCTIONS

DATA TYPE	DESCRIPTION
ADDDATE(x, y)	Adds a time/date interval y to a date x
ADDTIME(x, y)	Adds a time interval y to a time/date x
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)	Returns the date part from a date/time expression x
DATEDIFF(x, y)	Returns the difference between two date values x and y
DATE_FORMAT(x, y)	Formats a date x according to the specified format y
DATE_SUB(x, y)	Subtracts a time/date interval y from a date x
DAY(x)	Returns the day of the month for a given date x
DAYNAME(x)	Returns the name of the weekday for a given date x
DAYOFMONTH(x)	Returns the day of the month for a given date x
DAYOFWEEK(x)	Returns the weekday index for a given date x
DAYOFYEAR(x)	Returns the day of the year for a given date x
FROM_DAYS(x)	Returns a date from a given number of days x
HOUR(x)	Returns the hour for a given date x
LAST_DAY(x)	Returns the last day of the month for a given date x
LOCALTIME()	Returns the current date and time
LOCALTIMESTAMP()	Returns the current date and time
MICROSECOND(x)	Returns the microsecond for a given date x
MINUTE(x)	Returns the minute for a given date x
MONTH(x)	Returns the month for a given date x
MONTHNAME(x)	Returns the name of the month for a given date x
NOW()	Returns the current date and time
PERIOD_ADD(x, y)	Adds y months to period x
PERIOD_DIFF(x, y)	Returns the difference between two periods x and y
QUARTER(x)	Returns the quarter for a given date x
SECOND(x)	Returns the second for a given date x
SEC_TO_TIME(x)	Returns a time value in the format hh:mm:ss
STR_TO_DATE(x, y)	Returns a date based on a string x and a format y
SUBDATE(x, y)	Subtracts a time/date interval y from a date x
SUBTIME(x, y)	Subtracts a time interval y from a time/date x
SYSDATE()	Returns the current date and time
TIME(x)	Returns the time part from a given time/date expression x
TIMEDIFF(x, y)	Returns the difference between two time/date expressions x and y
TIMESTAMP(x)	Returns a date/time value based on a single value x
TIMESTAMPADD(x, y, z)	Returns a date/time value based on adding an interval y to a date/time x
TIMESTAMPDIFF(x, y, z)	Returns the difference between two date/time expressions x and y
TIME_FORMAT(x, y)	Formats a time x according to the specified format y
TIME_TO_SEC(x)	Returns the time x to seconds
TO_DAYS(x)	Returns the number of days between a date x and year 0
WEEK(x)	Returns the week number for a given date x
WEEKDAY(x)	Returns the weekday number for a given date x
WEEKOFYEAR(x)	Returns the week number for a given date x
YEAR(x)	Returns the year for a given date x
YEARWEEK(x)	Returns the year and week for a given date x

Kanak Infosystems LLP.

KEYS

FOREIGN KEY

```
CREATE TABLE orders
(
    order_id int(11) NOT NULL,
    product_id int(11) NOT NULL,
    order_date date DEFAULT NULL,
    PRIMARY KEY (order_id,product_id),
    KEY product_id (product_id),
    CONSTRAINT FK_ProductOrder FOREIGN KEY (product_id)
    REFERENCES products (product_id)
)
```

ORDERS

order_id	product_id	order_date
1	1	2008-11-11
2	1	2008-11-11
3	2	2008-11-11
4	3	2008-11-11

PRIMARY KEY

FOREIGN KEY

PRODUCTS

product_id	product_name
1	Gummy Bears
2	Candy Canes
3	Chocolate Bar

PRIMARY KEY

```
CREATE TABLE products
(
    product_id int(11) NOT NULL auto_increment,
    product_name varchar(60) NOT NULL default '',
    PRIMARY KEY (product_id)
)
```

EXAMPLE (MYSQL):

```
CREATE TABLE IF NOT EXISTS `tbl_name`
(`col1` INTEGER NOT NULL,
`col2` VARCHAR(30),
`col3` INTEGER NOT NULL,
`col4` DATE NOT NULL,
PRIMARY KEY (`col1`, `col3`),
KEY `col3` (`col3`),
KEY `col4` (`col4`)
);
```

PRIMARY KEY

EXAMPLE (MYSQL):

```
CREATE TABLE IF NOT EXISTS `tbl_name`
(`col1` INTEGER NOT NULL,
`col2` VARCHAR(30),
`col3` INTEGER NOT NULL,
`col4` DATE NOT NULL,
PRIMARY KEY (`col1`, `col3`),
KEY `col3` (`col3`),
KEY `col4` (`col4`)
);
```

EXAMPLE (MSSQL):

```
CREATE TABLE [dbo].[tbl_name]
([col1] [smallint] NOT NULL,
[col2] [nvarchar] (50) NULL,
[col3] [datetime] NULL,
[col4] [bit] NULL,
CONSTRAINT [PK_tbl_name] PRIMARY KEY [col1], [col2]
) ON [PRIMARY]
GO
CREATE INDEX [IX_tbl_name] ON [dbo].[tbl_name] ([col3], [col4])
GO
```

Kanak Infosystems LLP.

DATA TYPES

DATA TYPES

DATA TYPE	DESCRIPTION
CREATE TABLE myTABLE (myCOLUMN DATA TYPE);	Creates a table with a column of the specified data type
CREATE TABLE myTABLE AS SELECT myCOLUMN FROM otherTABLE;	Creates a table with a column of the same data type as the column in the table selected from
DROP TABLE myTABLE;	Deletes a table
DROP DATABASE myDB;	Deletes a database

CREATE VIEWS

```
CREATE VIEW view_name AS
SELECT column_name(s)
FROM table_name
WHERE condition;
```

REPLACE VIEWS

```
CREATE OR REPLACE VIEW view_name AS
SELECT column_name(s)
FROM table_name
WHERE condition;
```

DELETING VIEWS

```
DROP VIEW view_name;
```

JOINS

ORDERS

id	cust_id	amount	order_date
1	3	540	2005-10-03
2	1	1354	2005-10-08
3	3	877	2005-10-08
4	2	354	2005-10-10

PRODUCTS

id	name	cost	qty
1	Lemon	0.25	100
2	Orange	0.35	200
3	Apple	0.15	150

INNER JOIN

```
SELECT o.id, o.amount, o.order_date, p.name, p.qty
FROM orders AS o
INNER JOIN products AS p
ON o.id = p.id;
```

id	amount	order_date	name	qty
1	540	2005-10-03	Lemon	100
2	1354	2005-10-08	Orange	200
3	877	2005-10-08	Apple	150

LEFT JOIN

```
SELECT o.id, o.amount, o.order_date, p.name, p.qty
FROM orders AS o
LEFT JOIN products AS p
ON o.id = p.id;
```

id	amount	order_date	name	qty
1	540	2005-10-03	Lemon	100
2	1354	2005-10-08	Orange	200
3	877	2005-10-08	Apple	150
4	354	2005-10-10	NULL	NULL

RIGHT JOIN

```
SELECT o.id, o.amount, o.order_date, p.name, p.qty
FROM orders AS o
RIGHT JOIN products AS p
ON o.id = p.id;
```

id	amount	order_date	name	qty
1	540	2005-10-03	Lemon	100
2	1354	2005-10-08	Orange	200
3	877	2005-10-08	Apple	150
NULL	NULL	NULL	Candy	300

OUTER JOIN

```
SELECT o.id, o.amount, o.order_date, p.name, p.qty
FROM orders AS o
OUTER JOIN products AS p
ON o.id = p.id;
```

id	amount	order_date	name	qty
1	540	2005-10-03	Lemon	100
2	1354	2005-10-08	Orange	200
3	877	2005-10-08	Apple	150
4	354	2005-10-10	NULL	NULL
NULL	NULL	NULL	Candy	300

Kanak Infosystems LLP.

CRUDS

id	fn	ln	hire_date	salary	dept
1	Jim	Smith	2000-02-14	45000.00	Accounting
2	Jane	Doe	2000-03-15	55000.00	Accounting
3	Jack	Jones	2000-04-16	65000.00	Sales
4	Nancy	Davis	2000-05-17	75000.00	Sales
5	Linda	Jones	2000-06-18	85000.00	Management
6	Sarah	Marshall	2000-07-19	95000.00	Management

```
INSERT INTO table (column1, column2, column3, ...)
VALUES (value1, value2, value3, ...);
```

```
SELECT column1, column2, column3, ...
FROM table;
```

```
UPDATE table
SET column1 = value1, column2 = value2, column3 = value3, ...
WHERE condition;
```

```
DELETE FROM table
WHERE condition;
```

id	fn	ln	hire_date	salary	dept
1	Jim	Smith	2000-02-14	45000.00	Accounting
2	Jane	Doe	2000-03-15	55000.00	Accounting
3	Jack	Jones	2000-04-16	65000.00	Sales
4	Nancy	Davis	2000-05-17	75000.00	Sales
5	Linda	Jones	2000-06-18	85000.00	Management
6	Sarah	Marshall	2000-07-19	95000.00	Management

```
SELECT * FROM table
ORDER BY column1, column2, column3, ...
LIMIT offset, count;
```

```
SELECT * FROM table
WHERE column1 = value1
AND column2 = value2
AND column3 = value3
ORDER BY column1, column2, column3, ...
LIMIT offset, count;
```

www.kanakinfosystems.com

Kanak Infosystems LLP.