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 was created, adds an index
ADD CONSTRAINT	Creates an constraint after a table was created, for example a primary key, a unique key, a check constraint
ALTER TABLE	Adds, deletes, or modifies columns in a table, or changes the data type of a column in a table. Also used to add various constraints to a table
ALTER COLUMN	Changes the data type of a column in a table. Also used to add a default value to a column in a table
ALL	Returns true if all of the subqueries meet the condition. Used with SELECT, WHERE and HAVING statements
AND	Used in a WHERE or HAVING clause to add a second condition. Also used in JOIN statements
ANY	Returns true if any of the subqueries meet the condition. Used with SELECT, WHERE and HAVING statements
AS	Used to create an alias for a table or column in a table. Also used to rename a table or column in a table when a query is run
ASC	Used in an ORDER BY clause to sort the results in ascending order. Also used in CREATE INDEX statements
BETWEEN	Used in a WHERE clause to select a range of values. Also used in a JOIN statement
CASE	Used in a SELECT statement to return a value when a specified condition is met. Also used in UPDATE statements
CHECK	Used in a CREATE TABLE statement to add a check constraint to a column in a table. Also used in an ALTER TABLE statement
CREATE DATABASE	Creates a database. Also used in a DROP DATABASE statement
CREATE INDEX	Creates an index in a table. Also used in a DROP INDEX statement
DELETE	Deletes rows from a table. Also used in a JOIN statement
DESC	Used in an ORDER BY clause to sort the results in descending order
DISTINCT	Used in a SELECT statement to return only distinct values. Also used in a COUNT statement
DROP	Deletes a database, table, index, or view
DROP COLUMN	Deletes a column in a table
DROP CONSTRAINT	Deletes a constraint after a table was created
DROP DATABASE	Deletes an existing database
DROP DEFAULT	Deletes a default value for a column in a table
DROP INDEX	Deletes an index in a table
EXEC	Executes a stored procedure
EXISTS	Used in a WHERE clause to test for the existence of any record in a subquery
FOR	Used in a stored procedure to specify what statement to execute when a specified event occurs
FROM	Used in a SELECT statement to select data from a table. Also used in a DELETE statement
GROUP BY	Used in a SELECT statement to group the results by one or more columns
HAVING	Used in a SELECT statement to specify a condition for a group or aggregate. Used after GROUP BY
IN	Used in a WHERE clause to specify a list of values. Also used in an INSERT INTO statement
INDEX	Used in a CREATE INDEX statement to create an index in a table. Also used in a DROP INDEX statement
INNER JOIN	Used in a SELECT statement to join two tables. Returns rows when there is a match in both tables
INSERT INTO	Used in a SELECT statement to copy data into a table. Also used in an INSERT INTO statement to add new records to a table
IS NULL	Used in a WHERE clause to test for empty values
IS NOT NULL	Used in a WHERE clause to test for non-empty values
JOIN	Used in a SELECT statement to join two tables. Returns rows when there is a match in one of the tables
LEFT JOIN	Used in a SELECT statement to join two tables. Returns all rows from the first table, and the rows from the second table that match the first table
LIKE	Used in a WHERE clause to search for a specified pattern in a column
NOT	Used in a WHERE clause to select records that do not match the specified condition. Also used in a CREATE TABLE statement to add a NOT NULL constraint to a column in a table
OR	Used in a WHERE clause to add a second condition. Also used in a JOIN statement
ORDER BY	Used in a SELECT statement to sort the results by one or more columns in ascending or descending order
OUTER JOIN	Used in a SELECT statement to join two tables. Returns all rows when there is a match in one of the tables
PRIMARY KEY	Used in a CREATE TABLE statement to add a primary key to a column in a table. Also used in an ALTER TABLE statement
PROCEDURE	Used in a CREATE PROCEDURE statement to create a stored procedure. Also used in a DROP PROCEDURE statement
RIGHT JOIN	Used in a SELECT statement to join two tables. Returns all rows from the second table, and the rows from the first table that match the second table
ROWNUM	Used in a SELECT statement to select a specified number of records from a table
SELECT	Used in a SELECT statement to select data from a table. Also used in a SELECT INTO statement to copy data into a new table
SELECT DISTINCT	Used in a SELECT statement to select only distinct values. Also used in a SELECT INTO statement
SELECT INTO	Used in a SELECT statement to copy data into a new table. Also used in an INSERT INTO statement to copy data into an existing table
SET	Used in an UPDATE statement to specify which columns and values that should be updated in a table. Also used in a SELECT INTO statement to specify what columns and values that should be inserted into a new table
SOME	Returns true if any of the subqueries meet the condition. Used with SELECT, WHERE and HAVING statements
TOP	Used in a SELECT statement to specify the number of records to return. Also used in a DELETE statement
TRUNCATE TABLE	Deletes the data inside a table, but not the table itself
UNION	Combines the results of two or more SELECT statements. Also used in a CREATE VIEW statement
UNION ALL	Combines the results of two or more SELECT statements. Also used in a CREATE VIEW statement
UNIQUE	Used in a CREATE TABLE statement to add a unique constraint to a column in a table. Also used in an ALTER TABLE statement
UPDATE	Used in a SELECT statement to update a table. Also used in a CREATE VIEW statement
VALUES	Used in an INSERT INTO statement to specify the values to be inserted into a table. Also used in a DELETE statement
WHERE	Used in a SELECT statement to select records that match a specified condition. Also used in an UPDATE statement

SQL KEYWORDS

KEYWORDS	DESCRIPTION
ADD	Adds a column in an existing table, adds a constraint after a table was created, adds an index
ADD CONSTRAINT	Creates an constraint after a table was created, for example a primary key, a unique key, a check constraint
ALTER TABLE	Adds, deletes, or modifies columns in a table, or changes the data type of a column in a table. Also used to add various constraints to a table
ALTER COLUMN	Changes the data type of a column in a table. Also used to add a default value to a column in a table
ALL	Returns true if all of the subqueries meet the condition. Used with SELECT, WHERE and HAVING statements
AND	Used in a WHERE or HAVING clause to add a second condition. Also used in JOIN statements
ANY	Returns true if any of the subqueries meet the condition. Used with SELECT, WHERE and HAVING statements
AS	Used to create an alias for a table or column in a table. Also used to rename a table or column in a table when a query is run
ASC	Used in an ORDER BY clause to sort the results in ascending order. Also used in CREATE INDEX statements
BETWEEN	Used in a WHERE clause to select a range of values. Also used in a JOIN statement
CASE	Used in a SELECT statement to return a value when a specified condition is met. Also used in UPDATE statements
CHECK	Used in a CREATE TABLE statement to add a check constraint to a column in a table. Also used in an ALTER TABLE statement
CREATE DATABASE	Creates a database. Also used in a DROP DATABASE statement
CREATE INDEX	Creates an index in a table. Also used in a DROP INDEX statement
DELETE	Deletes rows from a table. Also used in a JOIN statement
DESC	Used in an ORDER BY clause to sort the results in descending order
DISTINCT	Used in a SELECT statement to return only distinct values. Also used in a COUNT statement
DROP	Deletes a database, table, index, or view
DROP COLUMN	Deletes a column in a table
DROP CONSTRAINT	Deletes a constraint after a table was created
DROP DATABASE	Deletes an existing database
DROP DEFAULT	Deletes a default value for a column in a table
DROP INDEX	Deletes an index in a table
EXEC	Executes a stored procedure
EXISTS	Used in a WHERE clause to test for the existence of any record in a subquery
FOR	Used in a stored procedure to specify what statement to execute when a specified event occurs
FROM	Used in a SELECT statement to select data from a table. Also used in a DELETE statement
GROUP BY	Used in a SELECT statement to group the results by one or more columns
HAVING	Used in a SELECT statement to specify a condition for a group or aggregate. Used after GROUP BY
IN	Used in a WHERE clause to specify a list of values. Also used in an INSERT INTO statement
INDEX	Used in a CREATE INDEX statement to create an index in a table. Also used in a DROP INDEX statement
INNER JOIN	Used in a SELECT statement to join two tables. Returns rows when there is a match in both tables
INSERT INTO	Used in a SELECT statement to copy data into a table. Also used in an INSERT INTO statement to add new records to a table
IS NULL	Used in a WHERE clause to test for empty values
IS NOT NULL	Used in a WHERE clause to test for non-empty values
JOIN	Used in a SELECT statement to join two tables. Returns rows when there is a match in one of the tables
LEFT JOIN	Used in a SELECT statement to join two tables. Returns all rows from the first table, and the rows from the second table that match the first table
LIKE	Used in a WHERE clause to search for a specified pattern in a column
NOT	Used in a WHERE clause to select records that do not match the specified condition. Also used in a CREATE TABLE statement to add a NOT NULL constraint to a column in a table
OR	Used in a WHERE clause to add a second condition. Also used in a JOIN statement
ORDER BY	Used in a SELECT statement to sort the results by one or more columns in ascending or descending order
OUTER JOIN	Used in a SELECT statement to join two tables. Returns all rows when there is a match in one of the tables
PRIMARY KEY	Used in a CREATE TABLE statement to add a primary key to a column in a table. Also used in an ALTER TABLE statement
PROCEDURE	Used in a CREATE PROCEDURE statement to create a stored procedure. Also used in a DROP PROCEDURE statement
RIGHT JOIN	Used in a SELECT statement to join two tables. Returns all rows from the second table, and the rows from the first table that match the second table
ROWNUM	Used in a SELECT statement to select a specified number of records from a table
SELECT	Used in a SELECT statement to select data from a table. Also used in a SELECT INTO statement to copy data into a new table
SELECT DISTINCT	Used in a SELECT statement to select only distinct values. Also used in a SELECT INTO statement
SELECT INTO	Used in a SELECT statement to copy data into a new table. Also used in an INSERT INTO statement to copy data into an existing table
SET	Used in an UPDATE statement to specify which columns and values that should be updated in a table. Also used in a SELECT INTO statement to specify what columns and values that should be inserted into a new table
SOME	Returns true if any of the subqueries meet the condition. Used with SELECT, WHERE and HAVING statements
TOP	Used in a SELECT statement to specify the number of records to return. Also used in a DELETE statement
TRUNCATE TABLE	Deletes the data inside a table, but not the table itself
UNION	Combines the results of two or more SELECT statements. Also used in a CREATE VIEW statement
UNION ALL	Combines the results of two or more SELECT statements. Also used in a CREATE VIEW statement
UNIQUE	Used in a CREATE TABLE statement to add a unique constraint to a column in a table. Also used in an ALTER TABLE statement
UPDATE	Used in a SELECT statement to update a table. Also used in a CREATE VIEW statement
VALUES	Used in an INSERT INTO statement to specify the values to be inserted into a table. Also used in a DELETE statement
WHERE	Used in a SELECT statement to select records that match a specified condition. Also used in an UPDATE statement

Kanak Infosystems LLP

COMMENTS

SINGLE LINE COMMENTS

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

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

MULTILINE COMMENTS

Multiline comments start with /* and end with */. Everything between these two characters is treated as a comment.

```
/* This is a multiline comment
SELECT * FROM table;
SELECT * FROM table;
SELECT * FROM table;
*/
```

MySQL Data Types

Click to view types.

```
mysql>
mysql>
```

NUMERIC DATA TYPES

DATA TYPE	DESCRIPTION
TINYINT(size)	Signed range is from -128 to 127. Unsigned range is from 0 to 255. The optional parameter (size) specifies the display width (not affecting the storage size)
SMALLINT(size)	Signed range is from -32768 to 32767. Unsigned range is from 0 to 65535. The optional parameter (size) specifies the display width (not affecting the storage size)
MEDIUMINT(size)	Signed range is from -8388608 to 8388607. Unsigned range is from 0 to 16777215. The optional parameter (size) specifies the display width (not affecting the storage size)
INT(size)	Signed range is from -2147483648 to 2147483647. Unsigned range is from 0 to 4294967295. The optional parameter (size) specifies the display width (not affecting the storage size)
BIGINT(size)	Signed range is from -9223372036854775808 to 9223372036854775807. Unsigned range is from 0 to 18446744073709551615. The optional parameter (size) specifies the display width (not affecting the storage size)
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 used. If p is between 25 and 53, DOUBLE is used
DOUBLE(size, d)	Floating point number. The number of digits after the decimal point is specified by the d parameter. The size parameter specifies the total number of digits
DECIMAL(size, d)	Stored as a string, allowing for a fixed decimal point. The number of digits after the decimal point is specified by the d parameter. The size parameter specifies the total number of digits
DEC(size, d)	Synonym for DECIMAL

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:MI: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 allows to specify the default value and automatic update of the current date and time
TIMESTAMP(fsp)	Stored as 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 allows to specify the default value and automatic update of the current date and time
TIME(fsp)	Time in format HH:MI:SS. Supported range is from '-838:59:59' to '838:59:59'
YEAR	2-digit or 4-digit integer type of year. Default is 4-digits

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 1 to 64. The default value for size is 1
TINYINT(size)	Signed range is from -128 to 127. Unsigned range is from 0 to 255. The optional parameter (size) specifies the display width (not affecting the storage size)
BOOL	Synonym for TINYINT(1). A value of zero is considered false. Non-zero values are considered true
SMALLINT(size)	Signed range is from -32768 to 32767. Unsigned range is from 0 to 65535. The optional parameter (size) specifies the display width (not affecting the storage size)
MEDIUMINT(size)	Signed range is from -8388608 to 8388607. Unsigned range is from 0 to 16777215. The optional parameter (size) specifies the display width (not affecting the storage size)
INT(size)	Signed range is from -2147483648 to 2147483647. Unsigned range is from 0 to 4294967295. The optional parameter (size) specifies the display width (not affecting the storage size)
BIGINT(size)	Signed range is from -9223372036854775808 to 9223372036854775807. Unsigned range is from 0 to 18446744073709551615. The optional parameter (size) specifies the display width (not affecting the storage size)
DECIMAL(size, d)	Stored as a string, allowing for a fixed decimal point. The number of digits after the decimal point is specified by the d parameter. The size parameter specifies the total number of digits
DEC(size, d)	Synonym for DECIMAL

STRING DATA TYPES

DATA TYPE	DESCRIPTION
CHAR(size)	Fixed length character string. Size is specified in parenthesis. Max 255 bytes
VARCHAR(size)	Variable length character string. Max size is specified in parenthesis. Max 65535 bytes
TINYTEXT(size)	Max length is 255 characters
TEXT(size)	Max length is 65535 characters
MEDIUMTEXT(size)	Max length is 16777215 characters
LONGTEXT(size)	Max length is 4294967295 characters
ENUM(val1, val2, val3, ...)	A string object that can have only one value, chosen from the list of values 'val1', 'val2', 'val3', ... In addition, the value can be NULL. The maximum number of values is 65535
SET(val1, val2, val3, ...)	A single value chosen from a set of up to 64 values. Syntax: SET('val1', 'val2', 'val3', ...)
BINARY(size)	Similar to CHAR(size), but stores binary byte strings rather than non-binary character strings. Size is specified in parenthesis. Max 255 bytes
VARBINARY(size)	Similar to VARCHAR(size), but stores binary byte strings rather than non-binary character strings. Size is specified in parenthesis. Max 65535 bytes
TINYBLOB(size)	Max length is 255 bytes
BLOB(size)	Max length is 65535 bytes
MEDIUMBLOB(size)	Max length is 16777215 bytes
LONGBLOB(size)	Max length is 4294967295 bytes

Kanak Infosystems LLP

OPERATORS

ARITHMETIC OPERATORS

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

COMPARISON OPERATORS

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

BITWISE OPERATORS

OPERATOR	DESCRIPTION
&	Bitwise AND
|	Bitwise OR
^	Bitwise XOR
~	Unary bit inversion
>>	Bit shift right
<<	Bit shift left

COMPOUND OPERATORS

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

Kanak Infosystems LLP

FUNCTIONS

STRING FUNCTIONS

DATA TYPE	DESCRIPTION
ASCII(s)	Returns the integer ASCII value of a character
CHAR_LENGTH(s)	Returns the length of a string (in characters)
CHARACTER_LENGTH(s)	Returns the length of a string (in characters)
CONCAT(s1, s2, ...)	Returns a string that is the result of concatenating one or more strings
CONCAT_WS(sep, s1, s2, ...)	Returns a string that is the result of concatenating one or more strings, separated by the separator specified in the first argument
FIELD(s, s1, s2, ...)	Returns the index position of a value in a list of values
FIND_IN_SET(s, s1)	Returns the index position of a value in a list of values
FORMAT(x, d)	Formats a number to a format like "#,###,###.##", rounded to a specified number of decimal places
INSERT(s1, x, len, s2)	Inserts a string into a specified position in another string. Starts at position x, and replaces len characters of s1 with s2
INSTR(s, s1)	Returns the position of the first occurrence of a string in another string
LCASE(s)	Converts a string to lowercase
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(s1, s)	Returns the position of the first occurrence of a string in another string
LOWER(s)	Converts a string to lowercase
LPAD(s, len, s1)	Left pads a string with another string, to a certain length
LTRIM(s)	Removes leading spaces from a string
MID(s, n, len)	Extracts a substring from a string (starting at any position)
POSITION(s1 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, s1, s2)	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, s1)	Right pads a string with another string, to a certain length
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)
TRIM(s)	Removes leading and trailing spaces from a string
UCASE(s)	Converts a string to uppercase
UPPER(s)	Converts a string to uppercase

NUMERIC FUNCTIONS

DATA TYPE	DESCRIPTION
ABS(x)	Returns the absolute value of a number
ACOS(x)	Returns the arccosine of a number
ASIN(x)	Returns the arcsine of a number
ATAN(x)	Returns the arctan of a number
ATAN2(x, y)	Returns the arctan2 of a pair of coordinates
CEIL(x)	Returns the smallest integer value that is bigger than or equal to a number
CEILING(x)	Returns the smallest integer value that is bigger than or equal to 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
DIV	Used for integer division
EXP(x)	Returns e to the power of x
FLOOR(x)	Returns the largest integer value that is smaller than or equal to a number
GREATEST(x1, x2, ...)	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

NUMERIC 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
DATE_FORMAT(d, f)	Displays date/time data in different formats as specified by the format specifier
DAY(d)	Returns the day part of a date
DAYNAME(d)	Returns the name of the weekday for a date
DAYOFMONTH(d)	Returns the day part of a date
DAYOFWEEK(d)	Returns the weekday index for a date
DAYOFYEAR(d)	Returns the day of the year for a date
FROM_DAYS(n)	Returns a date from a numeric representation of the day
HOUR(t)	Returns the hour part of a time/datetime
LAST_DAY(d)	Returns the last day of the month for a given date
LOCALTIME()	Returns the current date and time
LOCALTIMESTAMP()	Returns the current date and time
MICROSECOND(d)	Returns the microsecond part of a time/datetime
MINUTE(t)	Returns the minute part of a time/datetime
MONTH(d)	Returns the month part of a date
MONTHNAME(d)	Returns the name of the month for a date
NOW()	Returns the current date and time
PERIOD_ADD(p, n)	Adds a specified number of months to a period
PERIOD_DIFF(p1, p2)	Returns the difference between two periods
QUARTER(d)	Returns the quarter of a date
SECOND(t)	Returns the second part of a time/datetime
SEC_TO_TIME(s)	Returns the time in hours, minutes, and seconds
STR_TO_DATE(s, f)	Returns a date based on a string and a format specifier
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
SYSDATE()	Returns the current date and time
TIME(t)	Returns the time part of a time/datetime
TIME_FORMAT(t, f)	Displays time data in different formats as specified by the format specifier
TIME_TO_SEC(t)	Returns the number of seconds from a time/datetime value
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
TO_DAYS(d)	Returns the number of days between a date and year zero
WEEK(d)	Returns the week number for a given date
WEEKDAY(d)	Returns the weekday number for a given date
WEEKOFYEAR(d)	Returns the week number for a given date
YEAR(d)	Returns the year part of a date
YEARWEEK(d)	Returns the year and week number for a given date

Kanak Infosystems LLP

KEYS

FOREIGN KEY

```
EXAMPLE (MYSQL):
CREATE TABLE users (
id INT PRIMARY KEY,
name VARCHAR(100),
email VARCHAR(100)
);
CREATE TABLE products (
id INT PRIMARY KEY,
name VARCHAR(100),
price DECIMAL(10,2)
);
CREATE TABLE orders (
id INT PRIMARY KEY,
user_id INT,
product_id INT,
order_date DATE,
FOREIGN KEY (user_id) REFERENCES users(id),
FOREIGN KEY (product_id) REFERENCES products(id)
);
```

PRIMARY KEY

```
EXAMPLE (MYSQL):
CREATE TABLE table (
id INT PRIMARY KEY,
name VARCHAR(100)
);
```

Kanak Infosystems LLP

DATA TYPES

DATA TYPE	DESCRIPTION
CREATE TABLE table (column1 datatype, column2 datatype, ...);	Creates a table in the database. Specify the table name and the column names and their data types
CREATE TABLE table1 AS SELECT * FROM table2 WHERE condition;	Creates a table based on the result of a query. Copies the column structure and data from the source table
DROP TABLE table;	Deletes a table and all its data, indexes, triggers, constraints, and permissions
DROP TABLE IF EXISTS table;	Deletes a table if it exists, avoiding an error if the table does not exists

VIEWS

CREATING VIEWS

```
CREATE VIEW view_name AS
SELECT column1, column2, ...
FROM table
WHERE condition;
```

REPLACING VIEWS

```
CREATE OR REPLACE VIEW view_name AS
SELECT column1, column2, ...
FROM table
WHERE condition;
```

DELETING VIEWS

```
DROP VIEW view_name;
```

JOINS

ORDERS

id	user_id	product_id	quantity	order_date
1	1	101	2	2024-01-15
2	2	102	1	2024-01-16
3	1	103	3	2024-01-17

PRODUCTS

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

CROSS

id	name	department	salary
1	John Smith	Engineering	75000.00
2	Jane Doe	Marketing	65000.00
3	Bob Wilson	Sales	55000.00

SELF JOIN

id	name	manager_id
1	John Smith	NULL
2	Jane Doe	1
3	Bob Wilson	1

NATURAL JOIN

id	name	department	salary
1	John Smith	Engineering	75000.00
2	Jane Doe	Marketing	65000.00
3	Bob Wilson	Sales	55000.00

FULL JOIN

id	name	department	salary
1	John Smith	Engineering	75000.00
2	Jane Doe	Marketing	65000.00
3	Bob Wilson	Sales	55000.00

CROSS JOIN

id	name	department	salary
1	John Smith	Engineering	75000.00
2	Jane Doe	Marketing	65000.00
3	Bob Wilson	Sales	55000.00

Kanak Infosystems LLP

INDEXES

```
CREATE INDEX idx_name ON table(column);
CREATE UNIQUE INDEX idx_name ON table(column);
CREATE INDEX idx_name ON table(column1, column2);
DROP INDEX idx_name;
```

Kanak Infosystems LLP

TRANSACTIONS

```
BEGIN;
SELECT * FROM table;
UPDATE table SET column = value WHERE condition;
DELETE FROM table WHERE condition;
INSERT INTO table (column1, column2) VALUES (value1, value2);
COMMIT;
ROLLBACK;
SAVEPOINT savepoint_name;
ROLLBACK TO savepoint_name;
RELEASE savepoint_name;
SET TRANSACTION ISOLATION LEVEL READ UNCOMMITTED;
SET TRANSACTION ISOLATION LEVEL READ COMMITTED;
SET TRANSACTION ISOLATION LEVEL REPEATABLE READ;
SET TRANSACTION ISOLATION LEVEL SERIALIZABLE;
```

Kanak Infosystems LLP

VIEWS

```
CREATE VIEW view_name AS
SELECT column1, column2, ...
FROM table
WHERE condition;
CREATE OR REPLACE VIEW view_name AS
SELECT column1, column2, ...
FROM table
WHERE condition;
DROP VIEW view_name;
```

TRIGGERS

```
CREATE TRIGGER trigger_name
BEFORE INSERT ON table
FOR EACH ROW
BEGIN
-- trigger logic
END;
CREATE TRIGGER trigger_name
AFTER UPDATE ON table
FOR EACH ROW
BEGIN
-- trigger logic
END;
DROP TRIGGER trigger_name;
```

PROCEDURES

```
CREATE PROCEDURE procedure_name
BEGIN
SELECT * FROM table;
END;
CALL procedure_name();
DROP PROCEDURE procedure_name;
```

FUNCTIONS

```
CREATE FUNCTION function_name(parameters)
RETURN datatype
BEGIN
-- function logic
END;
SELECT function_name(parameters);
DROP FUNCTION function_name;
```

INDEXES

```
CREATE INDEX idx_name ON table(column);
CREATE UNIQUE INDEX idx_name ON table(column);
CREATE INDEX idx_name ON table(column1, column2);
DROP INDEX idx_name;
```

TRANSACTIONS

```
BEGIN;
SELECT * FROM table;
UPDATE table SET column = value WHERE condition;
DELETE FROM table WHERE condition;
INSERT INTO table (column1, column2) VALUES (value1, value2);
COMMIT;
ROLLBACK;
SAVEPOINT savepoint_name;
ROLLBACK TO savepoint_name;
RELEASE savepoint_name;
SET TRANSACTION ISOLATION LEVEL READ UNCOMMITTED;
SET TRANSACTION ISOLATION LEVEL READ COMMITTED;
SET TRANSACTION ISOLATION LEVEL REPEATABLE READ;
SET TRANSACTION ISOLATION LEVEL SERIALIZABLE;
```

Kanak Infosystems LLP

www.kanakinfosystems.com