Kanak

SQL

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

SQL KEYWORDS

KEYWORDS	DESCRIPTION
ADD	Adds a column in an existing table, or adds a constraint on a column in an existing table. Syntax: ALTER TABLE table ADD column
ADD CONSTRAINT	Adds a constraint after a table is already created. Syntax: ALTER TABLE table ADD CONSTRAINT constraint
ALTER TABLE	Adds, deletes, or modifies columns in a table, or changes the data type of columns in a table. Syntax: ALTER TABLE table ADD column data type; ALTER TABLE table DROP COLUMN column; ALTER TABLE table ALTER COLUMN column data type
ALTER COLUMN	Changes the data type of a column in a table. Syntax: ALTER TABLE table ALTER COLUMN column data type
ALL	Returns true if all of the subqueries meet the condition. Syntax: SELECT column FROM table WHERE condition ALL (subquery)
AND	Used in place of a comma (,) to add another condition in a WHERE clause. Syntax: SELECT column FROM table WHERE condition AND condition
ANY	Returns true if any of the subqueries meet the condition. Syntax: SELECT column FROM table WHERE condition ANY (subquery)
AS	Renames a column or table with an alias. Syntax: SELECT column AS alias FROM table AS alias
ASC	Used in place of ASCENDING. Syntax: SELECT column FROM table ORDER BY column ASC
AUTO INCREMENT	Automatically increases the value of a field. Syntax: CREATE TABLE table (column AUTO INCREMENT); INSERT INTO table (column) VALUES (NULL);
BACKUP DATABASE	Creates a back-up of a database. Syntax: BACKUP DATABASE database TO DISK filename
BETWEEN	Selects values within a given range. Syntax: SELECT column FROM table WHERE column BETWEEN value1 AND value2; SELECT column FROM table WHERE column NOT BETWEEN value1 AND value2
CASE	Creates a field based on a condition. Syntax: SELECT CASE column WHEN condition1 THEN result1 WHEN condition2 THEN result2 ELSE result3 END FROM table;
CHECK	Adds a constraint to a column when a table is created. Syntax: CREATE TABLE table (column CHECK condition); ALTER TABLE table ADD CHECK condition
CREATE DATABASE	Creates a database. Syntax: CREATE DATABASE database; SHOW DATABASES;
CREATE INDEX	Creates an index. Syntax: CREATE INDEX ON table (column); CREATE UNIQUE INDEX ON table (column);
DEFAULT	Adds a default value to a column when a table is created. Syntax: CREATE TABLE table (column DEFAULT value); ALTER TABLE table ALTER column SET DEFAULT value; ALTER TABLE table ALTER column DROP DEFAULT;
DELETE	Deletes rows from a table. Syntax: DELETE FROM table WHERE condition;
DESC	Used in place of DESCENDING. Syntax: SELECT column FROM table ORDER BY column DESC
DISTINCT	Selects only distinct values. Syntax: SELECT DISTINCT column FROM table;
DROP	Deletes a database, table, index, or column. Syntax: DROP DATABASE database; DROP TABLE table; DROP INDEX table.index; ALTER TABLE table DROP column;
DROP COLUMN	Deletes a column in a table. Syntax: ALTER TABLE table DROP COLUMN column
DROP CONSTRAINT	Deletes a constraint after a table is created. Syntax: ALTER TABLE table DROP CONSTRAINT constraint
DROP DATABASE	Deletes a database. Syntax: DROP DATABASE database
DROP INDEX	Deletes an index. Syntax: DROP INDEX table.index;
DROP TABLE	Deletes a table. Syntax: DROP TABLE table;
EXEC	Executes a function or a stored procedure. Syntax: EXEC function(parameters); EXEC procedure
EXECUTE	Executes a function or a stored procedure. Syntax: EXECUTE function(parameters); EXECUTE procedure
EXISTS	Tests for the existence of any record in a subquery. Syntax: SELECT column FROM table WHERE EXISTS (subquery)
FOREIGN KEY	Adds a foreign key to a table. Syntax: CREATE TABLE table (FOREIGN KEY column); ALTER TABLE table ADD FOREIGN KEY column
FROM	Used in place of a comma (,) before a table name. Syntax: SELECT column FROM table
GROUP BY	Groups rows with similar values. Syntax: SELECT column, COUNT(column) FROM table GROUP BY column;
HAVING	Used in place of WHERE with aggregate functions. Syntax: SELECT column, COUNT(column) FROM table GROUP BY column HAVING COUNT(column) > value;
IN	Selects values within a given range. Syntax: SELECT column FROM table WHERE column IN (value1, value2); SELECT column FROM table WHERE column NOT IN (value1, value2);
INDEX	Creates an index. Syntax: CREATE INDEX ON table (column); CREATE UNIQUE INDEX ON table (column);
INNER JOIN	Returns rows that have matching values in both tables. Syntax: SELECT column FROM table1 INNER JOIN table2 ON table1.column=table2.column;
INSERT INTO	Inserts new records in a table. Syntax: INSERT INTO table (column) VALUES (value); INSERT INTO table (column) SELECT column FROM table;
INSERT INTO SELECT	Copies data from one table into another table. Syntax: INSERT INTO table2 (column) SELECT column FROM table1;
IS NULL	Tests for empty values. Syntax: SELECT column FROM table WHERE column IS NULL
IS NOT NULL	Tests for non-empty values. Syntax: SELECT column FROM table WHERE column IS NOT NULL
JOIN	Used in place of INNER JOIN. Syntax: SELECT column FROM table1 JOIN table2 ON table1.column=table2.column;
LEFT JOIN	Returns all rows from the left table, even if there are no matches in the right table. Syntax: SELECT column FROM table1 LEFT JOIN table2 ON table1.column=table2.column;
LIKE	Searches for a specified pattern in a column. Syntax: SELECT column FROM table WHERE column LIKE pattern;
LIMIT	Specifies the number of records to return in the result set. Syntax: SELECT column FROM table LIMIT number;
MODIFY	Allows you to modify the data type of a column in a table. Syntax: ALTER TABLE table MODIFY column data type;
NOT	Used in place of !=. Syntax: SELECT column FROM table WHERE column NOT value;
NOT NULL	Used in place of IS NOT NULL. Syntax: SELECT column FROM table WHERE column NOT NULL
NULL	Used in place of IS NULL. Syntax: SELECT column FROM table WHERE column NULL
OFFSET	Used in place of LIMIT. Syntax: SELECT column FROM table LIMIT number OFFSET number;
ON	Used in place of WHERE in a JOIN statement. Syntax: SELECT column FROM table1 JOIN table2 ON table1.column=table2.column;
ORDER BY	Sorts the result set in ascending or descending order. Syntax: SELECT column FROM table ORDER BY column; SELECT column FROM table ORDER BY column DESC;
OR	Used in place of	in a WHERE statement. Syntax: SELECT column FROM table WHERE condition OR condition
OUTER JOIN	Returns all rows when there is a match in either table. Syntax: SELECT column FROM table1 OUTER JOIN table2 ON table1.column=table2.column;
PRIMARY KEY	Adds a primary key to a table. Syntax: CREATE TABLE table (PRIMARY KEY column); ALTER TABLE table ADD PRIMARY KEY column
RIGHT JOIN	Returns all rows from the right table, even if there are no matches in the left table. Syntax: SELECT column FROM table1 RIGHT JOIN table2 ON table1.column=table2.column;
ROLLBACK	Rollbacks a transaction in progress. Syntax: ROLLBACK
ROWNUM	Returns a specified number of records. Syntax: SELECT column FROM table WHERE ROWNUM < number;
SELECT	Selects data from a database. Syntax: SELECT column FROM table; SELECT * FROM table;
SELECT DISTINCT	Selects only distinct values. Syntax: SELECT DISTINCT column FROM table;
SELECT INTO	Copies data from one table into a new table. Syntax: SELECT column INTO newtable FROM table;
SELECT TOP	Used in place of LIMIT. Syntax: SELECT TOP number column FROM table; SELECT TOP percent column FROM table;
SET	Used in place of WHERE with UPDATE. Syntax: UPDATE table SET column=value;
SOME	Used in place of ANY. Syntax: SELECT column FROM table WHERE condition SOME (subquery)
TOP	Used in place of LIMIT. Syntax: SELECT TOP number column FROM table;
TRUNCATE TABLE	Deletes all records in a table, but not the table itself. Syntax: TRUNCATE TABLE table;
UNION	Combines the result sets of two or more SELECT statements (only distinct values). Syntax: SELECT column FROM table1 UNION SELECT column FROM table2;
UNION ALL	Combines the result sets of two or more SELECT statements (allows duplicate values). Syntax: SELECT column FROM table1 UNION ALL SELECT column FROM table2;
UNIQUE	Adds a constraint to a column when a table is created. Syntax: CREATE TABLE table (column UNIQUE); ALTER TABLE table ADD UNIQUE column
UPDATE	Updates existing records in a table. Syntax: UPDATE table SET column=value WHERE condition; UPDATE table SET column=value;
VALUES	Specifies the values for an INSERT INTO statement. Syntax: INSERT INTO table (column) VALUES (value); INSERT INTO table (column) VALUES (value), (value), (value);
WHERE	Specifies a condition. Syntax: SELECT column FROM table WHERE condition;

Kanak Infosystems LLP

COMMENTS

SINGLE LINE COMMENTS

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

MULTILINE COMMENTS

```
/* This is a multiline comment
It can be used anywhere
and can span across
many lines
SELECT * FROM table;
*/
```

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). Fixed size is specified in parenthesis. Can store up to 255 characters
VARCHAR(SIZE)	A variable length string (can contain letters, numbers, and special characters). Maximum size is specified in parenthesis. Can store up to 255 characters. Values are stored with a leading length field (1 byte or 2 bytes)
BINARY(SIZE)	Similar to CHAR(SIZE), but stores binary byte strings. Size is specified in parenthesis. Can contain binary byte strings
VARBINARY(SIZE)	Similar to VARCHAR(SIZE), but stores binary byte strings. Size is specified in parenthesis. Can contain binary byte strings
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. Optional parameter for specifying the maximum length in characters
BLOB(SIZE)	For BLOBs (Binary Large Objects). Holds up to 65,535 bytes of data. Optional parameter for specifying the maximum length in bytes
MEDIUMTEXT	Holds a string with a maximum length of 16,777,215 characters
MEDIUMBLOB	For BLOBs (Binary Large Objects). Holds up to 16,777,215 bytes of data
LONGTEXT	Holds a string with a maximum length of 4,294,967,295 characters
LONGBLOB	For BLOBs (Binary Large Objects). Holds up to 4,294,967,295 bytes of data
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 65,535 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. Number of bits per value is specified in parenthesis, from 1 to 64. The default is 1 if not specified
TINYINT(SIZE)	A very small integer. Signed range is from -128 to 127. Unsigned range is from 0 to 255. Optional parameter for specifying the maximum display width in parenthesis
BOOL(SIZE)	Zero is considered as false, non-zero values are considered as true. Optional parameter for specifying the maximum display width in parenthesis
SMALLINT(SIZE)	A small integer. Signed range is from -32768 to 32767. Unsigned range is from 0 to 65535. Optional parameter for specifying the maximum display width in parenthesis
MEDIUMINT(SIZE)	A medium integer. Signed range is from -8388608 to 8388607. Unsigned range is from 0 to 16777215. Optional parameter for specifying the maximum display width in parenthesis
INT(SIZE)	A medium integer. Signed range is from -2147483648 to 2147483647. Unsigned range is from 0 to 4294967295. Optional parameter for specifying the maximum display width in parenthesis
BIGINT(SIZE)	A large integer. Signed range is from -9223372036854775808 to 9223372036854775807. Unsigned range is from 0 to 18446744073709551615. Optional parameter for specifying the maximum display width in parenthesis
FLOAT(p)	A floating point number. MySQL uses a parameter value to decide 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 d
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 d
DEC(size, d)	Synonym of DECIMAL

DATE AND TIME DATA TYPES

DATA TYPE	DESCRIPTION
DATE	Date. Format: YYYY-MM-DD. The supported range is from '1000-01-01' to '9999-12-31'
DATETIME(fsp)	Date and time combination. Format: YYYY-MM-DD hh:mm:ss[.fractional seconds]. The supported range is from '1000-01-01 00:00:00.000000' to '9999-12-31 23:59:59.999999'. 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). Format: YYYY-MM-DD hh:mm:ss[.fractional seconds]. The supported range is from '1970-01-01 00:00:01.000000' to '2038-01-19 03:14:07.999999'. Adding DEFAULT and ON UPDATE in the column definition to get automatic initialization and updating to the current date and time
TIME(fsp)	Time. Format: hh:mm:ss[.fractional seconds]. The supported range is from '-838:59:59.000000' to '838:59:59.999999'
YEAR	A year in four-digit format (4 digits). Allowed values are from 1901 to 2155, and 0000

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
<=	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

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 the records if the statement is false
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 a character x
CHAR_LENGTH(x)	Returns the length of a string x (in characters)
CHARACTER_LENGTH(x)	Returns the length of a string x (in characters)
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 a value x in a list of values y, ...
FIND_IN_SET(x,y)	Returns the first occurrence of a string x in a string y, where y is a list of comma-separated values
FORMAT(x,y)	Formats a number x to a format like #,###,###.## rounded to y decimal places, and returns the result as a string
INSERT(x,y,z,w)	Returns a string x, beginning at position y, z characters long, and replaced with w
INSTR(x,y)	Returns the first position of a string y in a string x
LCASE(x)	Converts a string x to lower-case
LEFT(x,y)	Returns the left y characters of a string x
LENGTH(x)	Returns the length of a string x (in bytes)
LOCATE(x,y)	Returns the first position of a string x in a string y
LOWER(x)	Converts a string x to lower-case
LPAD(x,y,z)	Adds a string z to the left of a string x, such that the result is y characters long
LTRIM(x)	Removes leading spaces from a string x
MID(x,y,z)	Returns a substring starting from position y, z characters long, from a string x
POSITION(x IN y)	Returns the first position of a string x in a string y
REPEAT(x,y)	Repeats a string x y times
REPLACE(x,y,z)	Replaces all occurrences of a string y in a string x, with a string z
REVERSE(x)	Reverses a string x and returns the result
RIGHT(x,y)	Returns the right y characters of a string x
RPAD(x,y,z)	Adds a string z to the right of a string x, such that the result is y characters long
RTRIM(x)	Removes trailing spaces from a string x
SPACE(x)	Returns a string x spaces long
STRCMP(x,y)	Compares two strings x and y
SUBSTR(x,y,z)	Extracts a substring from a string x, starting from position y, z characters long
SUBSTRING(x,y,z)	Extracts a substring from a string x, starting from position y, z characters long
SUBSTRING_INDEX(x,y,z)	Returns a substring of a string x before z occurrences of a delimiter y
TRIM(x)	Removes leading and trailing spaces from a string x
UCASE(x)	Converts a string x to upper-case
UPPER(x)	Converts a string x to upper-case

NUMERIC FUNCTIONS

DATA TYPE	DESCRIPTION
ABS(x)	Returns the absolute value of a given number x
ACOS(x)	Returns the arccosine of a given number x
ASIN(x)	Returns the arcsine of a given number x
ATAN(x)	Returns the arctan of a given number x
ATAN2(x,y)	Returns the arctan2 of a given number x and y
AVG(x)	Returns the average value of an expression x
CEIL(x)	Rounds a given number x up to the next nearest integer
CEILING(x)	Rounds a given number x up to the next nearest integer
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 to the x power
FLOOR(x)	Rounds a given number x down to the next nearest integer
GREATEST(x,y,...)	Returns the greatest value of the list of arguments
GCD(x,y)	Returns the greatest common divisor of the given non-negative integers x and y
LN(x)	Returns the natural logarithm of a given number x
LOG(x)	Returns the natural logarithm of a given number x
LOG10(x)	Returns the base-10 logarithm of a given number x
LOG2(x)	Returns the base-2 logarithm of a given number x
MAX(x)	Returns the largest value of an expression x
MIN(x)	Returns the smallest value of an 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 a degree value x into radians
RAND()	Returns a random number
ROUND(x,y)	Rounds a given 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 an expression x
TAN(x)	Returns the tangent of a given number x
TRUNCATE(x,y)	Returns x truncated to y decimal places

DATE FUNCTIONS

DATA TYPE	DESCRIPTION
ADDDATE(x,y)	Returns date x with the specified time/date y added to it
ADDTIME(x,y)	Returns time x with the specified time y added to it
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()	Extracts the date part from a given date
DATE_ADD(x,y)	Returns date x with the specified time/date y added to it
DATE_FORMAT(x,y)	Formats a date x according to the specified format y
DATE_SUB(x,y)	Returns date x with the specified time/date y subtracted from it
DATEDIFF(x,y)	Returns the difference between two dates x and y
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
EXTRACT(x FROM y)	Returns a specified part x from a given date y
FROM_DAYS(x)	Returns a date value from a given numeric value x
HOUR(x)	Returns the hour part of 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
LOCALTIME()	Returns the current date and time
MAKEDATE(x,y)	Returns a date value from a given year x and a given number of days y
MAKETIME(x,y,z)	Returns a time value from a given hour x, a given minute y, and a given second z
MICROSECOND(x)	Returns the microsecond part of a given time x
MINUTE(x)	Returns the minute part of a given time x
MONTH(x)	Returns the month part of 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)	Returns a period x plus y number of months
PERIOD_DIFF(x,y)	Returns the difference between two periods x and y
QUARTER(x)	Returns the quarter for a given date x
SEC_TO_TIME(x)	Converts a given number of seconds x into hours, minutes and seconds
SECOND(x)	Returns the second part of a given time x
STR_TO_DATE(x,y)	Returns a date based on a string x and a format y
SUBDATE(x,y)	Returns date x with the specified time/date y subtracted from it
SUBTIME(x,y)	Returns time x with the specified time y subtracted from it
SYSDATE()	Returns the current date and time
TIME(x)	Extracts the time part from a given date x
TIME_FORMAT(x,y)	Formats a given time x according to the specified format y
TIME_TO_SEC(x)	Converts a given time x into seconds
TIMEDIFF(x,y)	Returns the difference between two time/date values x and y
TIMESTAMP(x,y)	Returns a date value from a given date x and a given time y
TO_DAYS(x)	Returns the number of days from year 0 to a given date x
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 part of a given date x
YEARWEEK(x)	Returns the year and week number for a given date x

Kanak Infosystems LLP

KEYS

FOREIGN KEY

```
CREATE TABLE orders
(
    order_id int,
    order_date date,
    customer_id int,
    FOREIGN KEY (customer_id) REFERENCES customers(customer_id)
);
```

```
CREATE TABLE products
(
    product_id int,
    product_name varchar(255),
    supplier_id int,
    FOREIGN KEY (supplier_id) REFERENCES suppliers(supplier_id)
);
```

EXAMPLE (MYSQL):

```
CREATE TABLE orders (order_id int, order_date date, customer_id int);
ALTER TABLE orders ADD FOREIGN KEY (customer_id) REFERENCES customers(customer_id);
```

EXAMPLE (MYSQL):

```
CREATE TABLE products (product_id int, product_name varchar(255), supplier_id int);
ALTER TABLE products ADD FOREIGN KEY (supplier_id) REFERENCES suppliers(supplier_id);
```

PRIMARY KEY

EXAMPLE (MYSQL):

```
CREATE TABLE persons
(
    ID int NOT NULL,
    last_name varchar(255),
    first_name varchar(255),
    address varchar(255),
    city varchar(255),
    PRIMARY KEY (ID)
);
```

EXAMPLE (MYSQL):

```
CREATE TABLE persons (ID int NOT NULL);
ALTER TABLE persons ADD PRIMARY KEY (ID);
```

Kanak Infosystems LLP

DATA TYPE

DATA TYPE	DESCRIPTION
CREATE TABLE table (column datatype);	Creates a table with the specified name and adds columns of specified type
DROP TABLE table;	Deletes a table, including all records, indexes, and privileges assigned to the table
DROP DATABASE database;	Deletes a database, including all tables in it, views, and privileges assigned to the database
CREATE DATABASE database;	Creates a database with the specified name
ALTER TABLE table ADD column datatype;	Adds a column of specified type in a table
ALTER TABLE table DROP COLUMN column;	Deletes a column in a table
ALTER TABLE table ALTER COLUMN column datatype;	Changes the data type of a column in a table
ALTER TABLE table RENAME TO newtable;	Changes the name of a table

VIEWS

CREATING VIEWS

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

SELECT * FROM view_name;

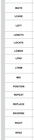
REPLACING VIEWS

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

DELETING VIEWS

DROP VIEW view_name;

JOINS



ORDERS

id	cust_id	order_id	order_date
1	3	308	2009-10-08
2	3	309	2009-10-08
3	9	310	2009-10-10

PRODUCTS

id	name	cost	date_added
1	Chai	18.00	2006-10-25
2	Chang	19.00	2006-10-25
3	Aniseed Syrup	10.00	2006-10-25

CRUDES

id	first_name	last_name	address	email
1	Cecil	Vinson	7894 Dolor Avenue	Cecil@Vinson.edu
2	Kaseem	Frost	494-8450 Vitae Avenue	Kaseem@Frost.edu
3	Hedda	Blackburn	53703 Ipsum Avenue	Hedda@Blackburn.ca
4	Hop	Slayton	P.O. Box 451, 5951 Sit Road	Hop@Slayton.edu
5	Kamal	Nash	587-8484 Ac Road	Kamal@Nash.ca

SELECT column1, column2, ...
FROM table_name;

SELECT column_name(s)
FROM table1
LEFT JOIN table2
ON table1.column_name = table2.column_name
ORDER BY table1.column_name;

Kanak Infosystems LLP

www.kanakinfosystems.com