**About SQL :**

Used to operate and manage relational database

FOUR FUNDAMENTAL OPERATIONS -

CRUD -

1. Create
2. Read
3. Update
4. Delete

The purpose of a Database is to organize the data

Relational Database is like a table

* Each ROW is a Record
* Each COLUMN is a Field

**CREATE statement:**

CREATE TABLE table\_name(a number, b string);

NOTE - TO CHECK TYPE OF VALUE :

SELECT TYPEOF(1+1); // whatever you enter

**SELECT statement :**

This is how you retrieve data from the database

Ex: SELECT ‘Hello, World’;

(NOTE: single quotes is for string literals

Double quotes are for an identifier (COLUMN NAMES))

**Selecting ROWS -**

SELECT \* / (column names separated by ‘,’) FROM (Tablename) WHERE (columnnames) = ‘.....’ ORDER BY (columnname) LIMIT (number) OFFSET (number) ;

**Selecting COLUMNS-**

SELECT \* / (column names separated by ‘,’) AS (optional name) FROM (Tablename);

**Count ROWS -**

SELECT COUNT(\* / Column name) FROM (table\_name) WHERE (Condition statement);

**INSERTING data -**

INSERT INTO (Table\_name) (column names in-between ( ) separated by ‘,’) VALUES ( respective column values separated by ‘,’ and individual value wrapped in single quotes (‘’));

NOTE - leaves NULL values if not interesting data into a column

**UPDATING data -**

UPDATE (table\_name) SET (column\_1) = ‘insert value here’, (column\_2) = ‘insert value here’ WHERE (Which row to update);

NOTE - Can insert NULL values

**DELETING data -**

DELETE FROM (table\_name) WHERE (which row);

NOTE - if there is no WHERE clause all the rows are DELETED.

**CREATE table -**

CREATE TABLE table\_name(

Col\_name Datatype (Can be followed by -

* NOT NULL to make it a mandatory column,
* DEFAULT ‘value’ to make default value,
* UNIQUE to have no duplicates),

.

.

)

**DELETING table -**

DROP TABLE table\_name;

DROP TABLE IF EXISTS table\_name;

**INSERTING rows -**

INSERT INTO table\_name (leave this option for all columns / column\_names) VALUES ( respective column values separated by ‘,’ and individual value wrapped in single quotes (‘’));

INSERT INTO table\_name DEFAULT VALUES; ### gives a NULL row

INSERT INTO table\_name (leave this option for all columns / column\_names) SELECT (\* / column\_names) FROM table\_name; ### inserting from another table

**DELETING rows -**

DELETE FROM table\_name WHERE (conditional statement - row number);

NOTE - A better practice is to use a SELECT \* FROM table\_name WHERE (conditional statement - row number);

* Check the data
* Replace SELECT \* -> DELETE

**NULL values -**

SELECT ( \* / column\_name) FROM table\_name WHERE column\_name IS NULL / NOT NULL;

**CHANGE SCHEMA -**

ALTER TABLE table\_name ADD column\_name Datatype ;

**ID column -**

CREATE TABLE table\_name(

column \_name INTEGER PRIMARY KEY,

.

.

);

PRIMARY KEY - the system will automatically generate a unique id and will force UNIQUE constraint

NOTE -

* This column will populate itself
* A table can only have one PRIMARY KEY

**FILTERING data -**

SELECT ( column\_names ) FROM table\_name WHERE conditional\_statement ORDER BY column\_name ( default or ASC is ascending / add DESC for descending order);

* The conditional statement is the filtering criteria

This conditional statement may contain -

* OR
* AND
* IN
* LIKE - see Documentation for more details
* And more see Documentation link - <https://www.sqlite.org/lang_select.html>

**REMOVE DUPLICATES -**

SELECT DISTINCT ( \* / column\_names) FROM table\_name;

**ORDER BY -**

……………….. ORDER BY col\_name; - default ascending order

……………….. ORDER BY col\_name DESC; - descending order

……………….. ORDER BY col\_name ASC; - ascending order

……………….. ORDER BY col\_name, col\_name; - sort a column and within another column (ex: cities in countries) and can also apply individual ASC DESC on each column if needed

**CONDITIONAL EXP -**

SELECT

CASE a WHEN condition THEN result ELSE else\_result END AS col\_name,

CASE b WHEN condition THEN result ELSE else\_result END AS col\_name,

FROM table\_name;

**JOIN -**

|  |  |
| --- | --- |
|  | SELECT a.column\_name AS alt\_colname ,  b.column\_name AS alt\_colname  FROM table\_name as a  JOIN table\_name as b  // followed by condition where these column values match  ON a.col\_name = b.col\_name;  NOTE -   1. AS alt\_colname is optional to change the col\_name 2. Multiple JOIN ON statements can be used for multiple JOIN |
|  | SELECT a.column\_name AS alt\_colname ,  b.column\_name AS alt\_colname  FROM table\_name as a  LEFT JOIN table\_name as b  // followed by condition where these column values match  ON a.col\_name = b.col\_name;  NOTE -   1. AS alt\_colname is optional to change the col\_name 2. Multiple JOIN ON statements can be used for multiple JOIN |
|  | NO Implementation available  BUT can be implemented using LEFT JOIN |
|  | NO Implementation available  BUT can be implemented using LEFT JOIN |