SELECT

SELECT { *, columnlist, expn, DISTINCT columnlist}

FROM tablename

WHERE condition {IN, LIKE, IS NULL, AND, OR, BETWEEN}

GROUP BY

HAVING {group functions (COUNT,MIN,MAX,AVG,SUM) condition}

ORDER BY columnlist asc/desc;

INSERT

INSERT INTO tablename [(columnlist)]

VALUES (valuelist);

DELETE

DELETE FROM tablename

[WHERE condition];

UPDATE

UPDATE tablename

SET column = { value, expn, sub query}

[,column = { value, expn, sub query}]...

[WHERE condition/subquery];

CREATE

CREATE TABLE tablename

(columnname [datatype][(size)] [column constraint] [AUTOINCREMENT]

[,columnname [datatype][(size)] [column constraint]]

[,table constraint]);

COLUMN CONSTRAINT

[CONSTRAINT constraintname]

NOT NULL

UNIQUE

PRIMARY KEY

CHECK (condition)

REFERENCES tablename[(columnlist)]

[ON DELETE CASCADE]

TABLE CONSTRAINT

[CONSTRAINT constraintname]

UNIQUE(columnlist) //composite unique key

PRIMARY KEY(columnlist) // composite primary key

CHECK(condition)

FOREIGN KEY columnlist REFERENCES tablename[(columnlist)]

[ON DELETE CASCADE]

ALTER TABLE

- The ALTER TABLE command in SQLite allows the user to rename a table or add a new column to an existing table.
- It is not possible to rename a column, remove a column, or add or remove constraints from a table.

ALTER TABLE tablename RENAME TO new_tablename;

ALTER TABLE tablename ADD COLUMN column_def;

DROP TABLE

DROP TABLE tablename;

CREATE DATABASE

sqlite3 database-name.db;

ATTACH DATABASE

ATTACH DATABASE ‘database-name’ AS ‘alias-name’;

DETACH DATABASE

DETACH [DATABASE] ‘alias-name’;

BEGIN TRANSACTION

BEGIN;

OR

BEGIN EXCLUSIVE TRANSACTION;

COMMIT

COMMIT [TRANSACTION];

OR

END [TRANSACTION];

ROLLBACK

ROLLBACK;

LIMIT

SELECT column1, column2, columnN

FROM tablename

LIMIT no_of_rows [OFFSET row_num];

Pragma foreign_key=ON;

CREATE TABLE dept

(deptno int CONSTRAINT deptno_pk PRIMARY KEY,

dname text,

loc text);

COLUMN CONSTRAINT

CREATE TABLE emp

(empno int CONSTRAINT empno_pk PRIMARY KEY,

ename text,

job text ,

mgr int CONSTRAINT mgr_fk REFERENCES emp(empno),

hiredate text,

sal real,

comm real,

deptno int CONSTRAINT dept_fk REFERENCES dept(deptno));

TABLE CONSTRAINT

create table emp

(empno integer,

ename text,

job text,

mgr int DEFAULT 7839,

hiredate text,

sal real,

comm real,

deptno int,

CONSTRAINT empno_pk PRIMARY KEY (empno),

CONSTRAINT mgr_fk FOREIGN KEY (mgr) REFERENCES emp(empno) ON DELETE SET DEFAULT,

CONSTRAINT dept_fk FOREIGN KEY (deptno) REFERENCES dept(deptno) ON DELETE CASCADE);

ON DELETE CASCADE, ON UPDATE CASCADE

ON DELETE SET NULL

ON DELETE SET DEFAULT

```
INSERT INTO dept VALUES (10,'accounting','surat');
INSERT INTO dept VALUES (20,'research','mumbai');
INSERT INTO dept VALUES (30,'sales','baroda'),(40,'operations','pune');

INSERT INTO emp VALUES (7839,'king','president',null,'1988-11-17',5000,null,10);
INSERT INTO emp VALUES (7698,'jones','manager',7839,'1983-05-07',3000,300,30);
INSERT INTO emp VALUES (7654,'allen','salesman',7839,'1980-09-01',1000,500,30);
INSERT INTO emp VALUES (7369,'smit','clerk',7698,'1982-06-17',3000,null,20);
INSERT INTO emp VALUES (7788,'scott','analyst',7698,'1988-11-07',5000,null,10);
INSERT INTO emp VALUES (7934,'millen','clerk',7788,'1989-01-12',1500,null,10);
INSERT INTO emp VALUES (7739,'smit jones','salesman',null,'198-11-17',1000,null,10);
INSERT INTO emp VALUES (7730,'ford','salesman',null,'1980-11-17',1200,null,30);
INSERT INTO emp VALUES (7731,'martin','clerk',null,'1981-11-17',1200,null,10);
INSERT INTO emp VALUES (7360,'james','salesman',7698,'1987-06-17',500,null,20);
```

1. List details of departments that are located in either surat or pune.

   ```
   SELECT * FROM dept
   WHERE loc='surat' OR loc = 'pune';
   ```

2. List details of departments that are located in neither surat nor pune.

   ```
   SELECT * FROM dept
   WHERE loc <> 'surat' AND loc <> 'pune';
                        OR
   SELECT * FROM dept
   WHERE NOT(loc='surat' OR loc='pune');
   ```

3. List details of clerks of deptno 10 and 30.

   ```
   SELECT * FROM emp
   WHERE job='clerk' AND (deptno=10 OR deptno=30);
   ```

4. List details of all clerks,analyst and manager.

   ```
   SELECT * FROM emp
   WHERE job IN ('clerk','analyst','manager');
   ```

5. List details of employees that contain the string 'le' in their names.

   ```
   SELECT * FROM emp
   WHERE ename LIKE '%le%';
   ```

6. List details of employees who are not assigned any commission.

   ```
   SELECT * FROM emp
   WHERE comm IS NULL;
   ```

7. List details of employees who are assigned any commission.

   ```
   SELECT * FROM emp
   WHERE comm IS NOT NULL;
   ```

8. List employees whose salary between 1500 and 4000.

   ```
   SELECT * FROM emp
   WHERE sal BETWEEN 1500 AND 4000;
                        OR
   SELECT * FROM emp
   WHERE sal >= 1500 AND sal <= 4000;
   ```

9. List empno, ename, job and total earnings of all the employees (earnings = sal+comm).

SELECT empno,ename,job,sal+coalesce(comm,0) FROM emp;

OR

SELECT empno,ename,job,sal+ifnull(comm,0) FROM emp;

10. List details of employees whose name contain two or more words.

    SELECT * FROM emp

    WHERE ename LIKE '% %';

11. List all jobs from emp table.

    SELECT distinct job FROM emp;

12. List department-wise, salary-wise details of all employees (where deptno-asc order and sal-desc order).

    SELECT * FROM emp

    ORDER BY deptno asc,sal desc;

13. Find total number of employees who do not get any commission.

    SELECT count(*) FROM emp

    WHERE comm IS NULL;

14. Find total number of jobs in the company.

    SELECT count(distinct job) FROM emp;

15. Find total number of clerks in the company.

    SELECT count(*) FROM emp

    WHERE job = 'clerk';

16. Find total salary of all the clerks.

    SELECT sum(sal),max(sal),min(sal),avg(sal) FROM emp

    WHERE job = 'clerk';

17. Find sum of salary of employees of each department.

    SELECT deptno,sum(sal) FROM emp

    GROUP BY deptno;

18. Find total number of jobs in each department.

    SELECT deptno,count(distinct job) FROM emp

    GROUP BY deptno;

19. Find total number of clerks in each department.

    SELECT deptno,count(*) FROM emp

```
WHERE job='clerk'
GROUP BY deptno;
```

20. List DEPTNO in which there are more than 3 employees.

```
SELECT deptno,count(*) FROM emp
GROUP BY deptno
HAVING count(*)>3;
```

21. List department-wise sum of salaries in descending order of the total salaries.

```
SELECT deptno ,sum(sal) FROM emp
GROUP BY deptno
ORDER BY sum(sal) desc;
```

QUERIES USING SUB-QUERY

22. List details of all the employees of sales department.

```
SELECT * FROM emp
WHERE deptno = (SELECT deptno FROM dept
                WHERE dname ='sales');
```

23. List details of the departments that have at least 1 employee.

```
SELECT * FROM dept
WHERE deptno IN (SELECT deptno FROM emp);
```

24. List details of the employees that are drawing same salary as that of martin.

```
SELECT * FROM emp
WHERE sal = (SELECT sal from emp where ename ='martin')
and ename <> 'martin';
```

25. List details of the employees that have same salary and job as that of martin.

```
SELECT * FROM emp
WHERE sal = (SELECT sal FROM emp WHERE ename ='martin')
and job= (SELECT job FROM emp WHERE ename ='martin')
and ename <> 'martin';
                    OR
SELECT * FROM emp
```

```
WHERE (sal,job) = (SELECT sal,job FROM emp
                        WHERE ename ='martin')
and   ename <> 'martin';
```

26. List details of employees of DEPTNO 20 that have salary greater than that of an employee of DEPTNO 10.

```
SELECT * FROM emp
WHERE deptno = 20
and sal >(SELECT min(sal) FROM emp
        WHERE deptno = 10);
```

27. List details of the employees of DEPTNO 10 that have salary greater than that of all employee of DEPTNO 20.

```
SELECT * FROM emp
WHERE deptno = 10
and sal >(SELECT max(sal) FROM emp
        WHERE deptno = 20);
```

28. List DEPTNO in which there are more than 2 employees.

```
SELECT deptno,count(*) FROM emp
GROUP BY deptno
HAVING count(*)>2;
```

29. List details of the department in which there are more than 2 employees.

```
SELECT * FROM dept
WHERE deptno IN (SELECT deptno FROM emp
                GROUP BY deptno
                HAVING count(*) >2);
```

30. List details of employees of sales department that have salary greater than that of

an employee of accounting department.

```
SELECT * FROM emp
WHERE deptno = (SELECT deptno FROM dept where dname = 'sales')
and sal>(SELECT min(sal) FROM emp
        WHERE deptno = (select deptno from dept where dname = 'accounting'));


SELECT * FROM emp
```

WHERE deptno = ___ and sal > ____

LIMIT

31. List details of first 5 employees.
    SELECT * FROM emp LIMIT 5;
32. List details of 4 employees starting from 3rd position.
    SELECT * FROM emp LIMIT 4 OFFSET 2;

SELF JOIN

SELECT e1.empno,e1.ename,e1.mgr,e2.ename as manager

FROM emp e1

INNER JOIN emp e2

ON e1.mgr=e2.empno;

SELECT e1.empno,e1.ename,e1.mgr,e2.ename as manager

FROM emp e1,emp e2

WHERE e1.mgr=e2.empno;

TRIGGER

INSERT

```
CREATE TRIGGER  IF NOT EXISTS trg_validate_emp_before_insert
BEFORE INSERT
ON emp
BEGIN
SELECT CASE
WHEN NEW.sal <= 0 OR NEW.comm < 0 THEN RAISE(ABORT, 'Invalid Salary or Commission')
END;
END;

CREATE TABLE emp_log
(empno integer,
operation text,
old_sal real,
new_sal real,
old_comm real,
new_comm real);

CREATE TRIGGER trg_emp_after_insert
AFTER INSERT
ON emp
BEGIN
INSERT into emp_log(empno,operation) VALUES(NEW.empno,'INSERT' || date('now'));
END;
```

UPDATE

```
CREATE TRIGGER trg_emp_after_update
AFTER UPDATE
ON emp
WHEN OLD.sal <> NEW.sal OR OLD.comm <> NEW.comm
```

```
BEGIN
INSERT INTO emp_log(empno,operation,old_sal,new_sal,old_comm,new_comm)
VALUES(OLD.empno,'UPDATE'|| date('now'), OLD.sal,NEW.sal,OLD.comm,NEW.comm);
END;

CREATE TRIGGER trg_validate_emp_before_update
BEFORE UPDATE
ON emp
BEGIN
SELECT CASE
WHEN NEW.sal<=0 THEN RAISE(ABORT, 'Invalid Salary')
WHEN NEW.comm<0 THEN RAISE(ABORT,'Invalid Commission’)
END;
END;
```

DELETE

```
CREATE TRIGGER trg_dept_before_delete
 BEFORE DELETE
 ON dept
 BEGIN
 SELECT CASE
WHEN (SELECT count(deptno) FROM emp WHERE deptno=OLD.deptno)>0 THEN
RAISE(ABORT,"Employee table having employees..Can’t delete this department")
END;
END;
```