

BASICS OF SQL

1) Create a table students(id, name, marks) and insert 2 rows.

```
CREATE TABLE students (  
  id INT,  
  name VARCHAR(50),  
  marks INT  
);
```

```
INSERT INTO students VALUES (1, 'Harini', 72);  
INSERT INTO students VALUES (2, 'Janani', 85);  
SELECT * FROM students
```

id	name	marks
1	Harini	72
2	Janani	85

2) Create a table employees(emp_id, emp_name, salary) and insert 2 sample rows.

```
CREATE TABLE employees (  
  emp_id INT,  
  emp_name VARCHAR(50),  
  salary INT  
);
```

```
INSERT INTO employees VALUES (111, 'KAVIN', 40000);  
INSERT INTO employees VALUES (112, 'KANI', 60000);  
SELECT * FROM employees
```

emp_id	emp_name	salary
111	KAVIN	40000
112	KANI	60000

3) Create a table products(product_id, product_name, price) and insert 3 products.

```
CREATE TABLE products (  
    productid INT,  
    productname VARCHAR(50),  
    price INT  
);
```

```
INSERT INTO products VALUES (1, 'PENCIL', 10);  
INSERT INTO products VALUES (2, 'PEN', 50);  
INSERT INTO products VALUES (3, 'SCALE', 115);  
SELECT * FROM products
```

productid	productname	price
1	PENCIL	10
2	PEN	50
3	SCALE	115

4) Update an employee's salary by 10% in the employees table.

```
UPDATE employees  
SET salary = salary + (salary * 0.1)  
WHERE emp_id IS NOT NULL;  
SELECT * FROM employees
```

emp_id	emp_name	salary
111	KAVIN	44000
112	KANI	66000

5) Delete a product where the price is > 500.

```
1 INSERT INTO products VALUES (1, 'BAG', 1000);
2 INSERT INTO products VALUES (2, 'SHOES', 700);
3 INSERT INTO products VALUES (3, 'PENCILBOX', 115);
4 SELECT * FROM products
5
```

productid	productname	price
1	PENCIL	10
2	PEN	50
3	SCALE	115
1	BAG	1000
2	SHOES	700
3	PENCILBOX	115

```
DELETE FROM products
WHERE price > 500;
SELECT * FROM products
```

productid	productname	price
1	PENCIL	10
2	PEN	50
3	SCALE	115
3	PENCILBOX	115

- 6) Select all students with marks > 80.
SELECT id,name, marks FROM students
WHERE marks > 80;

id	name	marks
2	Janani	85

- 7) Use a recursive CTE to simulate a WHILE loop to increase salary by 1000 until it reaches 10,000.

```
1 INSERT INTO employees VALUES (111, 'KAVIN', 4000);
2 INSERT INTO employees VALUES (112, 'KANI', 6000);
3 SELECT * FROM employees
4
```

emp_id	emp_name	salary
111	KAVIN	4000
112	KANI	6000

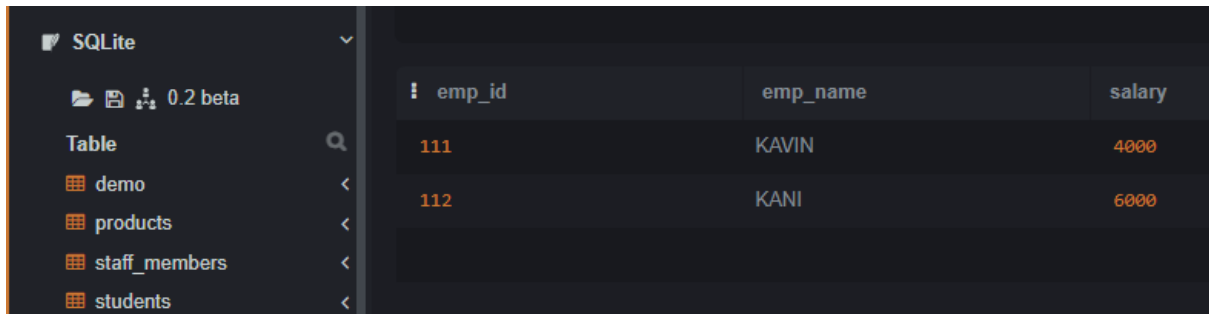
```
WITH RECURSIVE inc(emp_id, emp_name, salary) AS (
  SELECT emp_id, emp_name, salary FROM employees
  UNION ALL
  SELECT emp_id, emp_name, salary + 1000 FROM inc
  WHERE salary < 10000
)
SELECT * FROM inc
WHERE salary >= 10000;
```

emp_id	emp_name	salary
112	KANI	10000
111	KAVIN	10000

8) Rename the table employees to staff_members

```
ALTER TABLE employees RENAME TO staff_members;
```

```
SELECT * FROM staff_members
```



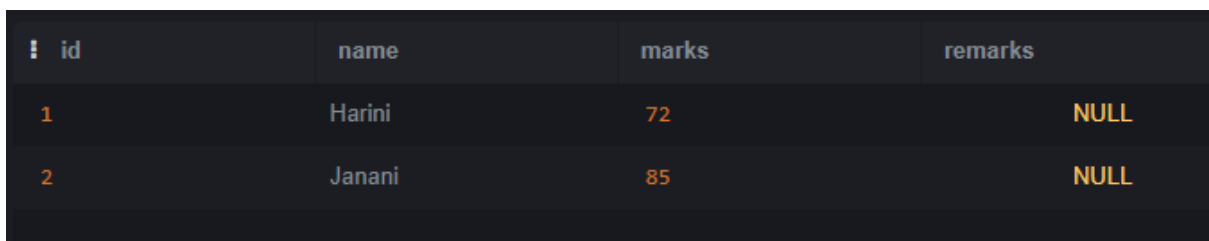
The screenshot shows the SQLite application interface. On the left, a sidebar lists the database '0.2 beta' and its tables: 'demo', 'products', 'staff_members', and 'students'. The 'staff_members' table is selected. The main area displays the table's data:

emp_id	emp_name	salary
111	KAVIN	4000
112	KANI	6000

9) Add a new column remarks to the students table.

```
ALTER TABLE students ADD remarks VARCHAR(100);
```

```
SELECT * FROM students
```



The screenshot shows the SQLite application interface with the 'students' table selected. The table now has four columns: 'id', 'name', 'marks', and 'remarks'. The data is as follows:

id	name	marks	remarks
1	Harini	72	NULL
2	Janani	85	NULL

10) Create a backup table students_backup with all rows from students.

```
CREATE TABLE students_backup AS
```

```
SELECT * FROM students;
```

```
SELECT * FROM students_backup;
```



The screenshot shows the SQLite application interface with the 'students_backup' table selected. The table structure and data are identical to the 'students' table:

id	name	marks	remarks
1	Harini	72	NULL
2	Janani	85	NULL