DATA SCIENCE

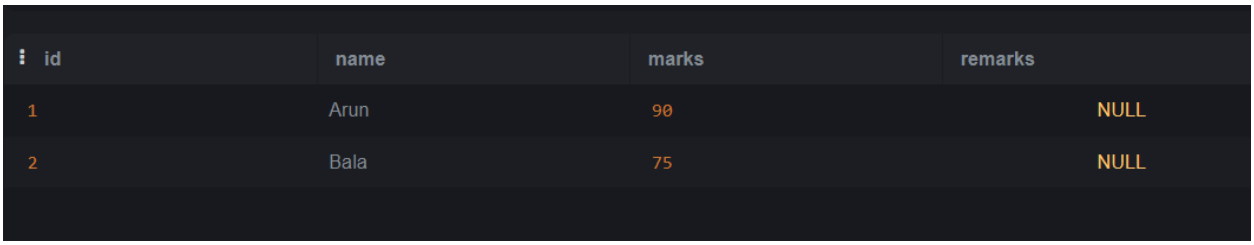
Task 1: Create a table `students(id, name, marks)` and insert 2 rows. (DDL/DML)

SQL

```
-- Create Table
CREATE TABLE students(
  id INT PRIMARY KEY,
  name VARCHAR(50),
  marks INT
);

-- Insert 2 Rows
INSERT INTO students(id, name, marks) VALUES (1, 'Arun', 90);
INSERT INTO students(id, name, marks) VALUES (2, 'Bala', 75);

-- Check the result
SELECT * FROM students;
```



id	name	marks	remarks
1	Arun	90	NULL
2	Bala	75	NULL

Task 2: Create a table `employees(emp_id, emp_name, salary)` and insert 2 sample rows. (DDL/DML)

SQL

```
-- Create Table
CREATE TABLE employees(
  emp_id INT PRIMARY KEY,
  emp_name VARCHAR(50),
  salary INT
);

-- Insert 2 Rows
INSERT INTO employees(emp_id, emp_name, salary) VALUES (101, 'Charlie',
5000);
INSERT INTO employees(emp_id, emp_name, salary) VALUES (102, 'Devi', 8000);

-- Check the result
```

```
SELECT * FROM employees;
```

emp_id	emp_name	salary
101	Charlie	5000
102	Devi	8000

Task 3: Create a table `products(product_id, product_name, price)` and insert 3 products. (DDL/DML)

SQL

```
-- Create Table
CREATE TABLE products(
  product_id INT PRIMARY KEY,
  product_name VARCHAR(100),
  price INT
);

-- Insert 3 Rows
INSERT INTO products(product_id, product_name, price) VALUES (10, 'Laptop',
1200);
INSERT INTO products(product_id, product_name, price) VALUES (11, 'Mouse',
450);
INSERT INTO products(product_id, product_name, price) VALUES (12, 'Monitor',
600);

-- Check the result
SELECT * FROM products;
```

product_id	product_name	price
10	Laptop	1200
11	Mouse	450
12	Monitor	600

Task 4: Update an employee's salary by 10% in the `employees` table. (DML)

We will increase the salary of the employee with `emp_id = 101` (Charlie) from 5000 to $5000 \times 1.10 = 5500$.

SQL

```
-- Update the salary for a specific employee
```

```
UPDATE employees
SET salary = salary * 1.10
WHERE emp_id = 101;

-- Check the result (Charlie's salary should now be 5500)
SELECT * FROM employees;
```

emp_id	emp_name	salary
101	Charlie	5500
102	Devi	8000

Task 5: Delete a product where the price is > 500. (DML)

We will use the `DELETE` command with a `WHERE` clause.

SQL

```
-- Delete rows where the price value is greater than 500
DELETE FROM products
WHERE price > 500;

-- Check the result
-- The Laptop (1200) and Monitor (600) should now be gone.
SELECT * FROM products;
```

product_id	product_name	price
11	Mouse	450

Task 6: Select all students with marks > 80. (DML)

SQL

```
-- Retrieve the 'id', 'name', and 'marks' columns for students
-- where the 'marks' value is greater than 80.
SELECT id, name, marks
FROM students
WHERE marks > 80;
```

id	name	marks
1	Arun	90

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

This code does **not** modify the existing `employees` table. It *simulates* the loop and displays the progression of a salary starting at 5000 until it hits 10,000.

SQL

```
WITH RECURSIVE SalaryIncrease (current_salary, increment_count) AS (
  -- →□ ANCHOR MEMBER: Defines the starting point (salary 5000, 0 increments)
  SELECT 5000 AS current_salary, 0 AS increment_count

  UNION ALL

  -- →□ RECURSIVE MEMBER: Defines the iteration (add 1000, increase count)
  SELECT current_salary + 1000, increment_count + 1
  FROM SalaryIncrease
  -- →□ TERMINATION CONDITION: Stops the loop when the salary exceeds 10000
  WHERE current_salary + 1000 <= 10000
)
-- →□ FINAL SELECT: Shows the result of each step of the simulation
SELECT current_salary, increment_count
FROM SalaryIncrease;
```

current_salary	increment_count
5000	0
6000	1
7000	2
8000	3
9000	4
10000	5

Task 8: Rename the table `employees` to `staff_members`. (DDL)

The document lists `RENAME TABLE`, but the standard SQLite syntax is `ALTER TABLE RENAME TO`.

SQL
```
-- Rename the table 'employees' to 'staff_members'
ALTER TABLE employees
RENAME TO staff_members;

-- Verify the change by trying to select from the old name (should fail)
-- SELECT * FROM employees;

-- Verify the change by selecting from the new name
SELECT * FROM staff_members;
```

emp_id	emp_name	salary
101	Charlie	6050.000000000001
102	Devi	8000

Task 9: Add a new column `remarks` to the `students` table. (DDL)

We will use the `ALTER TABLE ADD COLUMN` syntax.

SQL
```
-- Add a new column named 'remarks' with a text data type (VARCHAR)
ALTER TABLE students
ADD COLUMN remarks VARCHAR(255);

-- Check the new structure and see the empty 'remarks' column
SELECT * FROM students;
```

id	name	marks	remarks
1	Arun	90	
2	Bala	75	

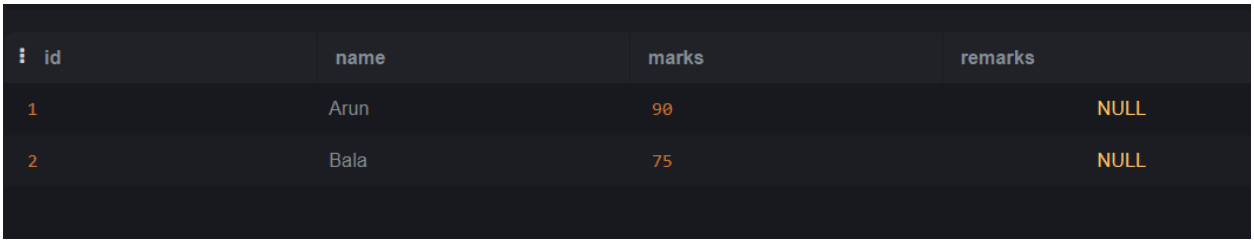
Task 10: Create a backup table `students_backup` with all rows from `students`. (DDL/DML)

We use a combination command: `CREATE TABLE AS SELECT`. This command creates a new table (`students_backup`) and populates it with the results of the `SELECT` query in one step.

SQL

```
-- Create a new table 'students_backup'
-- and copy all columns and rows ('SELECT *') from the 'students' table.
CREATE TABLE students_backup AS
SELECT *
FROM students;

-- Check the contents of the new backup table
SELECT * FROM students_backup;
```



id	name	marks	remarks
1	Arun	90	NULL
2	Bala	75	NULL