

1.1. Create a table soil_sensors if it does not exist with the following columns:

- sensor_id (INT, Primary Key)
- moisture_level (INT)
- recorded_time (DATETIME)

```
9 • CREATE TABLE IF NOT EXISTS soil_sensors (
10     sensor_id INT NOT NULL,
11     moisture_level INT,
12     recorded_time DATETIME,
13     PRIMARY KEY (sensor_id)
14 );
15
16 • INSERT INTO soil_sensors (sensor_id, moisture_level, recorded_time) VALUES
17     (1, 30, '2026-1-1'),
18     (2, 55, '2026-1-2'),
19     (3, 20, '2026-1-3'),
20     (4, 75, '2026-1-4'),
21     (5, 40, '2026-1-5');
```

	sensor_id	moisture_level	recorded_time
▶	1	30	2026-01-01 00:00:00
	2	55	2026-01-02 00:00:00
	3	20	2026-01-03 00:00:00
	4	75	2026-01-04 00:00:00
	5	40	2026-01-05 00:00:00

1.2. Then Create a stored procedure GetDryFields that:

- Accepts a moisture threshold as an input parameter
- Displays all sensor records where moisture_level is below the given threshold.

```

24    -- 1.2. Then Create a stored procedure GetDryFields that:
25    -- • Accepts a moisture threshold as an input parameter
26    -- • Displays all sensor records where moisture_level is less than the threshold
27    -- DROP PROCEDURE GetDryFields;
28    DELIMITER $$ 
29 •  CREATE PROCEDURE GetDryFields(IN moisture_threshold INT)
30  BEGIN
31      SELECT sensor_id, moisture_level, recorded_time
32      FROM soil_sensors
33      WHERE moisture_level < moisture_threshold;
34  END$$
35  DELIMITER ;
36
37 •  CALL GetDryFields(50);

```

37 • `CALL GetDryFields(50);`

	sensor_id	moisture_level	recorded_time
▶	1	30	2026-01-01 00:00:00
	3	20	2026-01-03 00:00:00
	5	40	2026-01-05 00:00:00

38 • `CALL GetDryFields(70);`

	sensor_id	moisture_level	recorded_time
▶	1	30	2026-01-01 00:00:00
	2	55	2026-01-02 00:00:00
	3	20	2026-01-03 00:00:00
	5	40	2026-01-05 00:00:00

Result 3 X

2.1. Create a table fields if it does not exist with the following columns:

- field_id (INT, Primary Key)
- area_sq_meters (INT)
- crop_type (VARCHAR(50))

```

45 • CREATE TABLE IF NOT EXISTS fields (
46     field_id INT NOT NULL,
47     area_sq_meters INT,
48     crop_type VARCHAR(50),
49     PRIMARY KEY (field_id)
50 );
51
52 • INSERT INTO fields (field_id, area_sq_meters, crop_type) VALUES
53     (1, 100, 'Rice'),
54     (2, 200, 'Corn'),
55     (3, 300, 'Lettuce'),
56     (4, 400, 'Wheat'),
57     (5, 500, 'Eggplant');
58 • select * from fields;

```

	field_id	area_sq_meters	crop_type
▶	1	100	Rice
	2	200	Corn
	3	300	Lettuce
	4	400	Wheat
	5	500	Eggplant

fields 4 ×

2.2. Create a stored function CalculateWaterRequirement that:

- Accepts area_sq_meters as input
- Returns the total water required per irrigation cycle
- Assume 5 liters of water per square meter

```
58 •    select * from fields;
59      -- 2.2. Create a stored function CalculateWaterRequirement that:
60      -- • Accepts area_sq_meters as input
61      -- • Returns the total water required per irrigation cycle
62      -- • Assume 5 liters of water per square meter
63      DELIMITER $$

64 •    CREATE FUNCTION CalculateWaterRequirement(area_sq_meters INT)
65      RETURNS INT DETERMINISTIC
66      BEGIN
67          DECLARE total_water INT;
68          SET total_water = area_sq_meters * 5;
69          return total_water;
70      END$$
71      DELIMITER ;

72

73 •    SELECT CalculateWaterRequirement(1000) AS water_required;
74
75
73 •    SELECT CalculateWaterRequirement(1000) AS water_required;
```

Result Grid		Filter Rows:	Export:	Wrap Cell Content:
	water_required			
▶	5000			

```
74 •    SELECT CalculateWaterRequirement(1500) AS water_required;
75
76      -- calculate water for each field
```

Result Grid		Filter Rows:	Export:	Wrap Cell Content:
	water_required			
▶	7500			

```

76      -- calculate water for each field
77 •   SELECT field_id, area_sq_meters,
78     CalculateWaterRequirement(area_sq_meters) AS water_for_fields
79   FROM fields;

```

Result Grid | Filter Rows: Export: Wrap Cell Content:

	field_id	area_sq_meters	water_for_fields
▶	1	100	500
	2	200	1000
	3	300	1500
	4	400	2000
	5	500	2500

3.1. Create a table irrigation_logs if it does not exist with the following columns:

- log_id (INT, Primary Key)
- field_id (INT)
- water_used_liters (DECIMAL(10,2))
- irrigation_date (DATE)

```

87 • CREATE TABLE IF NOT EXISTS irrigation_logs (
88     log_id INT NOT NULL,
89     field_id INT,
90     water_used_liters DECIMAL(10,2),
91     irrigation_date DATE,
92     PRIMARY KEY (log_id)
93 );
94
95 • INSERT INTO irrigation_logs (log_id, field_id, water_used_liters, irrigation_date) VALUES
96     (1, 1, 100, '2026-01-01'),
97     (2, 2, 200, '2026-01-01'),
98     (3, 3, 300, '2026-01-02'),
99     (4, 4, 400, '2026-01-02'),
100    (5, 1, 500, '2026-01-03'),
101    (6, 5, 600, '2026-01-04');
102 • select * from irrigation_logs;

```

```

101      (o, o, ooo, 2020-01-04 )j
102 •   select * from irrigation_logs;

```

Result Grid | Filter Rows: | E

	log_id	field_id	water_used_liters	irrigation_date
▶	1	1	100.00	2026-01-01
	2	2	200.00	2026-01-01
	3	3	300.00	2026-01-02
	4	4	400.00	2026-01-02
	5	1	500.00	2026-01-03

3.2. Create a stored procedure GetTotalWaterUsage that:

- Accepts a date as an input parameter
- Returns the total water used on that date using an OUT parameter.

```

108      DELIMITER $$
109 •   CREATE PROCEDURE GetTotalWaterUsage(IN p_irrigation_date DATE, OUT total_water_used DECIMAL(10,2)
110      )
111      BEGIN
112          SELECT IFNULL(SUM(water_used_liters), 0)
113          INTO total_water_used
114          FROM irrigation_logs
115          WHERE irrigation_date = p_irrigation_date;
116      END$$
117      DELIMITER ;
118
119 •   CALL GetTotalWaterUsage('2026-01-01', @total);
120 •   SELECT @total AS total_water_used_liters;
121
119 •   CALL GetTotalWaterUsage('2026-01-01', @total);
120 •   SELECT @total AS total_water_used_liters;

```

Result Grid | Filter Rows: | Export: | Wrap

	total_water_used_liters
▶	300.00

```

122 •   CALL GetTotalWaterUsage('2026-01-02', @total);
123 •   SELECT @total AS total_water_used_liters;

```

Result Grid | Filter Rows: | Export: | Wrap

	total_water_used_liters
▶	700.00

4.1. Create a table field_status if it does not exist with the following columns:

- field_id (INT, Primary Key)
- moisture_level (INT)

```
128      -- drop table field_status;
129 • CREATE TABLE IF NOT EXISTS field_status (
130      field_id INT NOT NULL,
131      moisture_level INT,
132      PRIMARY KEY (field_id)
133 );
134
135 • INSERT INTO field_status (field_id, moisture_level) VALUES
136      (1, 45),
137      (2, 25),
138      (3, 60),
139      (4, 15),
140      (5, 30),
141      (6, 10);
142 • select * from field_status;
```

	field_id	moisture_level
▶	1	45
	2	25
	3	60
	4	15
	5	30

4.2. Create a stored procedure CountCriticalFields that:

- Uses a cursor
- Iterates through all records
- Counts how many fields have a moisture level below 30
- Displays the total count

```
148      -- • Displays the total count
149      DELIMITER $$*
150  •  CREATE PROCEDURE CountCriticalFields()
151  BEGIN
152      DECLARE done INT DEFAULT 0;
153      DECLARE v_moisture INT;
154      DECLARE critical_count INT DEFAULT 0;
155      -- Cursor declaration
156      DECLARE field_cursor CURSOR FOR
157          SELECT moisture_level FROM field_status;
158      -- Handle cursor end
159      DECLARE CONTINUE HANDLER FOR NOT FOUND SET done = 1;
160      OPEN field_cursor;
161      count_loop: LOOP
162          FETCH field_cursor INTO v_moisture;
163
164          IF done = 1 THEN
165              LEAVE count_loop;
166          END IF;
167
168          IF v_moisture < 30 THEN
169              SET critical_count = critical_count + 1;
170          END IF;
171
172      END LOOP;
173      CLOSE field_cursor;
174
175      -- Display total count
176      SELECT critical_count AS total_critical_fields;
177  END$$
178  DELIMITER ;
179
180  •  CALL CountCriticalFields();
```

180 •	<code>CALL CountCriticalFields();</code>
Result Grid Filter Rows: <input type="text"/>	
	total_critical_fields
▶	3

5.1. Create a table irrigation_schedule if it does not exist with the following columns:

- schedule_id (INT, Primary Key)
- field_id (INT)
- irrigation_duration (INT) (in minutes)

```
187 • CREATE TABLE IF NOT EXISTS irrigation_schedule (
188     schedule_id INT NOT NULL,
189     field_id INT,
190     irrigation_duration INT,
191     PRIMARY KEY (schedule_id)
192 );
193
194 • INSERT INTO irrigation_schedule (schedule_id, field_id, irrigation_duration) VALUES
195     (1, 1, 30),
196     (2, 2, 40),
197     (3, 3, 50),
198     (4, 4, 60),
199     (5, 5, 70);
200 • select * from irrigation_schedule;
```

	schedule_id	field_id	irrigation_duration
▶	1	1	30
	2	2	40
	3	3	50
	4	4	60
	5	5	70

5.2. Create a stored procedure AdjustIrrigationDuration that:

- Uses a cursor
- Increases irrigation duration by 20%
- Updates each row individually

```
205      -- • Updates each row individually
206      DELIMITER $$
```

```
207  •  CREATE PROCEDURE AdjustIrrigationDuration()
```

```
208  BEGIN
```

```
209      DECLARE done INT DEFAULT 0;
```

```
210      DECLARE v_schedule_id INT;
```

```
211      DECLARE v_duration INT;
```

```
212
```

```
213      -- Declare cursor
214      DECLARE schedule_cursor CURSOR FOR
215          SELECT schedule_id, irrigation_duration
216          FROM irrigation_schedule;
```

```
217
```

```
218      -- handle cursor end
219      DECLARE CONTINUE HANDLER FOR NOT FOUND SET done = 1;
220      OPEN schedule_cursor;
221      duration_loop: LOOP
```

```
222          FETCH schedule_cursor INTO v_schedule_id, v_duration;
223
```

```
224          IF done = 1 THEN
225              LEAVE duration_loop;
226          END IF;
```

```
227
```

```
228          -- Duration increase 20%
229          UPDATE irrigation_schedule
230              SET irrigation_duration = irrigation_duration * 1.20
231              WHERE schedule_id = v_schedule_id;
232      END LOOP;
233      CLOSE schedule_cursor;
234  END$$
```

```
235  DELIMITER ;
```

```
236
```

```
237  •  CALL AdjustIrrigationDuration();
```

```
238  •  SELECT * FROM irrigation_schedule;
```

```
239
```

```
237 • CALL AdjustIrrigationDuration();
238 • SELECT * FROM irrigation_schedule;
239
```

	schedule_id	field_id	irrigation_duration
▶	1	1	36
	2	2	48
	3	3	60
	4	4	72
	5	5	84

```
237 • CALL AdjustIrrigationDuration();
238 • SELECT * FROM irrigation_schedule;
```

	schedule_id	field_id	irrigation_duration
▶	1	1	43
	2	2	58
	3	3	72
	4	4	86
	5	5	101

irrigation_schedule 17 × 236

```
237 • CALL AdjustIrrigationDuration();
238 • SELECT * FROM irrigation_schedule;
```

	schedule_id	field_id	irrigation_duration
▶	1	1	52
	2	2	70
	3	3	86
	4	4	103
	5	5	121

irrigation_schedule 18 ×