

# Placement Readiness Assignment

## SQL

### Problem Statement

You are working with a pharmacy that wants to track its medicine inventory, suppliers, purchases, and sales. Your task is to design, create, and populate a relational database that allows users to perform meaningful queries for operations and analysis.

### Schema Description

#### 1. Medicines

- medicine\_id (INT, Primary Key)
- name (VARCHAR)
- type (VARCHAR: Tablet, Syrup, Injection)
- unit\_price (DECIMAL)
- stock\_quantity (INT)
- expiry\_date (DATE)

#### 2. Suppliers

- supplier\_id (INT, Primary Key)
- name (VARCHAR)
- contact\_number (VARCHAR)
- city (VARCHAR)

#### 3. Purchases

- purchase\_id (INT, Primary Key)
- supplier\_id (INT, Foreign Key -> Suppliers)
- medicine\_id (INT, Foreign Key -> Medicines)
- quantity (INT)
- purchase\_date (DATE)
- batch\_number (VARCHAR)

#### 4. Sales

- sale\_id (INT, Primary Key)
- medicine\_id (INT, Foreign Key -> Medicines)
- quantity (INT)
- sale\_date (DATE)
- total\_price (DECIMAL)

# Placement Readiness Assignment

## SQL

**DDL:**

- 1. Add a column 'manufacturer' to the Medicines table.**
- 2. Rename the column 'contact\_number' to 'phone' in Suppliers.**

-- 1. Add a column 'manufacturer' to the Medicines table

```
create database pharmacy;
```

```
use pharmacy;
```

```
ALTER TABLE Medicines
```

```
ADD COLUMN manufacturer VARCHAR(100);
```

-- 2. Rename the column 'contact\_number' to 'phone' in Suppliers

```
ALTER TABLE Suppliers
```

```
CHANGE COLUMN contact_number phone VARCHAR(15);
```

**3. Insert a new medicine 'Amoxicillin'.**

**4. Update all medicine prices by 10%.**

**5. Delete medicines that are expired.**

-- 3. Insert a new medicine 'Amoxicillin'

-- (Assuming it's a different variant or batch; using a new medicine\_id)

```
INSERT INTO Medicines (medicine_id, name, type, unit_price, stock_quantity, expiry_date, manufacturer)
```

## **Placement Readiness Assignment**

### **SQL**

```
VALUES (21, 'Amoxicillin 500mg', 'Capsule', 9.50, 200, '2027-06-30', 'MediPharm Ltd.');
```

```
-- 4. Update all medicine prices by 10%
```

```
SET SQL_SAFE_UPDATES = 0;
```

```
UPDATE Medicines
```

```
SET unit_price = unit_price * 1.10;
```

```
SET SQL_SAFE_UPDATES = 1;
```

```
-- 5. Delete medicines that are expired (expiry_date is before today)
```

```
DELETE FROM Medicines
```

```
WHERE expiry_date < CURDATE();
```

### **Aggregations:**

**6. Find total sales revenue per medicine.**

**7. Find the supplier who provided the maximum quantity.**

**8. Total units sold per medicine type**

```
-- 6. Find total sales revenue per medicine
```

```
SELECT
```

```
m.name AS medicine_name,
```

```
SUM(s.total_price) AS total_revenue
```

```
FROM Sales s
```

```
JOIN Medicines m ON s.medicine_id = m.medicine_id
```

```
GROUP BY m.medicine_id, m.name
```

# Placement Readiness Assignment

## SQL

ORDER BY total\_revenue DESC;

```

32 • SELECT
33     m.name AS medicine_name,
34     SUM(s.total_price) AS total_revenue
35     FROM Sales s
36     JOIN Medicines m ON s.medicine_id = m.medicine_id
37     GROUP BY m.medicine_id, m.name
38     ORDER BY total_revenue DESC;
  
```

The screenshot shows the SQL code in the editor. Below it is the result grid, which displays the following data:

medicine_name	total_revenue
Omeprazole 20mg	156.00
Pantoprazole 40mg	156.00
Atorvastatin 10mg	150.00
Vitamin C 500mg	140.00
Montelukast 10mg	140.00

Below the result grid is the action output log:

#	Time	Action	Message	Duration / Fetch
4	20:09:14	INSERT INTO Medicines (medicine_id, name, type, unit_price, stock_qty)	INSERT INTO Medicines (medicine_id, name, type, unit_price, stock_qty, expiry_date, manufacturer) VALUES (21, 'Amoxicillin 500mg', 'Capsule', 9.50, 200, '2027-06-30', 'MediPharm Ltd.')	0.000 sec
5	20:09:19	SET SQL_SAFE_UPDATES = 0		
6	20:09:31	SET SQL_SAFE_UPDATES = 1	0 row(s) affected	0.000 sec
7	20:09:55	DELETE FROM Medicines WHERE expiry_date < CURDATE()	Error Code: 1175. You are using safe update mode and you tried to update a table ...	0.000 sec
8	20:10:34	SELECT m.name AS medicine_name, SUM(s.total_price) AS total_revenue F...	20 row(s) returned	0.000 sec / 0.000 sec

-- 7. Find the supplier who provided the maximum quantity

SELECT

```

sup.name AS supplier_name,
SUM(p.quantity) AS total_quantity_supplied
  
```

FROM Purchases p

JOIN Suppliers sup ON p.supplier\_id = sup.supplier\_id

GROUP BY sup.supplier\_id, sup.name

ORDER BY total\_quantity\_supplied DESC

LIMIT 1;

```

40 -- 7. Find the supplier who provided the maximum quantity
41 • SELECT
42     sup.name AS supplier_name,
43     SUM(p.quantity) AS total_quantity_supplied
44     FROM Purchases p
45     JOIN Suppliers sup ON p.supplier_id = sup.supplier_id
46     GROUP BY sup.supplier_id, sup.name
  
```

The screenshot shows the SQL code in the editor. Below it is the result grid, which displays the following data:

supplier_name	total_quantity_supplied
Global Pharma Co.	730

Below the result grid is the action output log:

#	Time	Action	Message	Duration / Fetch
5	20:09:19	SET SQL_SAFE_UPDATES = 0	0 row(s) affected	0.000 sec
6	20:09:31	SET SQL_SAFE_UPDATES = 1	0 row(s) affected	0.000 sec
7	20:09:55	DELETE FROM Medicines WHERE expiry_date < CURDATE()	Error Code: 1175. You are using safe update mode and you tried to update a table ...	0.000 sec
8	20:10:34	SELECT m.name AS medicine_name, SUM(s.total_price) AS total_revenue F...	20 row(s) returned	0.000 sec / 0.000 sec
9	20:12:28	SELECT sup.name AS supplier_name, SUM(p.quantity) AS total_quantity_sup...	1 row(s) returned	0.000 sec / 0.000 sec

# Placement Readiness Assignment

## SQL

-- 8. Total units sold per medicine type

SELECT

```
m.type AS medicine_type,  
SUM(s.quantity) AS total_units_sold  
FROM Sales s  
JOIN Medicines m ON s.medicine_id = m.medicine_id  
GROUP BY m.type  
ORDER BY total_units_sold DESC;
```

The screenshot shows a SQL query being run in a database environment. The query is:

```
-- 8. Total units sold per medicine type  
SELECT  
    m.type AS medicine_type,  
    SUM(s.quantity) AS total_units_sold  
FROM Sales s  
JOIN Medicines m ON s.medicine_id = m.medicine_id  
GROUP BY m.type
```

The results are displayed in a grid:

medicine_type	total_units_sold
Tablet	213
Capsule	19
Syrup	10
Inhaler	2

Below the results, the 'Output' section shows the execution log:

#	Time	Action	Message	Duration / Fetch
6	20:09:31	SET SQL_SAFE_UPDATES = 1	0 row(s) affected	0.000 sec
7	20:09:55	DELETE FROM Medicines WHERE expiry_date < CURDATE()	Error Code: 1175. You are using safe update mode and you tried to update a table ...	0.000 sec
8	20:10:34	SELECT m.name AS medicine_name, SUM(s.total_price) AS total_revenue F...	20 row(s) returned	0.000 sec / 0.000 sec
9	20:12:28	SELECT sup.name AS supplier_name, SUM(p.quantity) AS total_quantity_sup...	1 row(s) returned	0.000 sec / 0.000 sec
10	20:14:16	SELECT m.type AS medicine_type, SUM(s.quantity) AS total_units_sold F...	4 row(s) returned	0.000 sec / 0.000 sec

## Subqueries:

-- 9. Medicines with stock below average

SELECT

```
medicine_id,  
name,  
stock_quantity  
FROM Medicines  
WHERE stock_quantity < (  
    SELECT AVG(stock_quantity)  
    FROM Medicines
```

# Placement Readiness Assignment

## SQL

');

```

58
59      -- 9. Medicines with stock below average
60 •   SELECT
61       medicine_id,
62       name,
63       stock_quantity
64  FROM Medicines

```

**Result Grid**

medicine_id	name	stock_quantity
3	Omeprazole 20mg	200
5	Atorvastatin 10mg	250
8	Cetirizine 10mg	180
9	Azithromycin 500mg	150
10	Diazepam 5mg	100

**Medicines 4** ×

**Action Output**

#	Time	Action	Message	Duration / Fetch
7	20:09:55	DELETE FROM Medicines WHERE expiry_date < CURDATE()	Error Code: 1175. You are using safe update mode and you tried to update a table ...	0.000 sec
8	20:10:34	SELECT m.name AS medicine_name, SUM(s.total_price) AS total_revenue F...	20 row(s) returned	0.000 sec / 0.000 sec
9	20:12:28	SELECT sup.name AS supplier_name, SUM(p.quantity) AS total_quantity_sup...	1 row(s) returned	0.000 sec / 0.000 sec
10	20:14:16	SELECT m.type AS medicine_type, SUM(s.quantity) AS total_units_sold FRO...	4 row(s) returned	0.000 sec / 0.000 sec
11	20:14:58	SELECT medicine_id, name, stock_quantity FROM Medicines WHERE sto...	13 row(s) returned	0.000 sec / 0.000 sec

### -- 10. Medicines never sold

```

SELECT
    medicine_id,
    name
FROM Medicines
WHERE medicine_id NOT IN (
    SELECT DISTINCT medicine_id
    FROM Sales
    WHERE medicine_id IS NOT NULL
);

```

```

70      -- 10. Medicines never sold
71 •   SELECT
72       medicine_id,
73       name
74  FROM Medicines
75  WHERE medicine_id NOT IN (
76      SELECT DISTINCT medicine_id

```

**Result Grid**

medicine_id	name
21	Amoxicillin 500mg
HOLD	HOLD

**Medicines 5** ×

**Action Output**

#	Time	Action	Message	Duration / Fetch
8	20:10:34	SELECT m.name AS medicine_name, SUM(s.total_price) AS total_revenue F...	20 row(s) returned	0.000 sec / 0.000 sec
9	20:12:28	SELECT sup.name AS supplier_name, SUM(p.quantity) AS total_quantity_sup...	1 row(s) returned	0.000 sec / 0.000 sec
10	20:14:16	SELECT m.type AS medicine_type, SUM(s.quantity) AS total_units_sold FRO...	4 row(s) returned	0.000 sec / 0.000 sec
11	20:14:58	SELECT medicine_id, name, stock_quantity FROM Medicines WHERE sto...	13 row(s) returned	0.000 sec / 0.000 sec
12	20:18:43	SELECT medicine_id, name FROM Medicines WHERE medicine_id NOT IN (...)	1 row(s) returned	0.000 sec / 0.000 sec

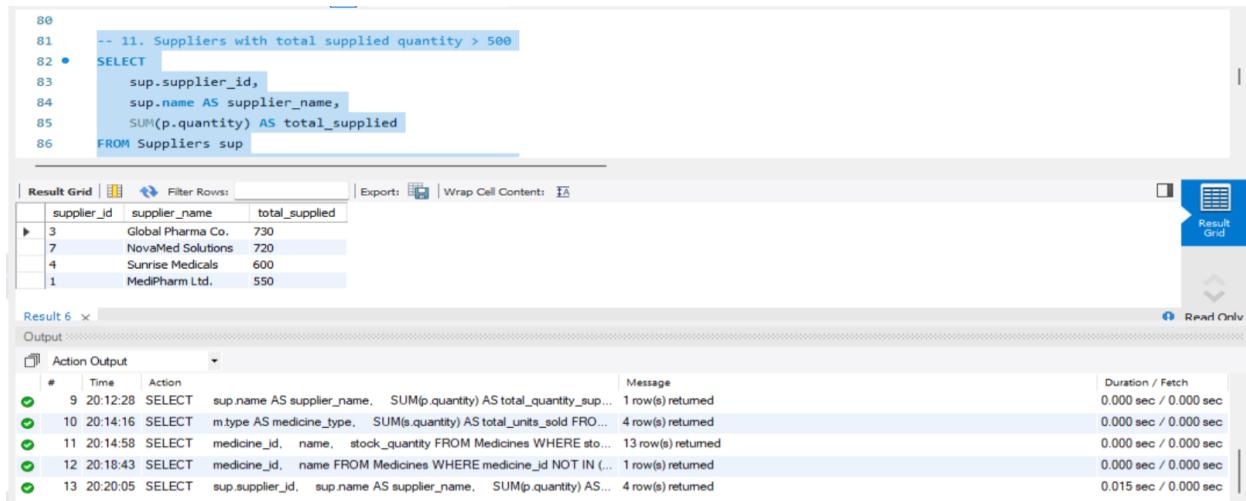
# Placement Readiness Assignment

## SQL

### -- 11. Suppliers with total supplied quantity > 500

SELECT

```
sup.supplier_id,  
sup.name AS supplier_name,  
SUM(p.quantity) AS total_supplied  
  
FROM Suppliers sup  
  
JOIN Purchases p ON sup.supplier_id = p.supplier_id  
  
GROUP BY sup.supplier_id, sup.name  
  
HAVING total_supplied > 500  
  
ORDER BY total_supplied DESC;
```



The screenshot shows a SQL editor with the following code:

```
80  
81 -- 11. Suppliers with total supplied quantity > 500  
82 • SELECT sup.supplier_id,  
83     sup.name AS supplier_name,  
84     SUM(p.quantity) AS total_supplied  
85 FROM Suppliers sup
```

Below the code is a Result Grid table:

supplier_id	supplier_name	total_supplied
3	Global Pharma Co.	730
7	NovaMed Solutions	720
4	Sunrise Medicals	600
1	MediPharm Ltd.	550

At the bottom, there is an Action Output log:

#	Time	Action	Message	Duration / Fetch
9	20:12:28	SELECT sup.name AS supplier_name, SUM(p.quantity) AS total_quantity_sup...	1 row(s) returned	0.000 sec / 0.000 sec
10	20:14:16	SELECT m.type AS medicine_type, SUM(s.quantity) AS total_units_sold FRO...	4 row(s) returned	0.000 sec / 0.000 sec
11	20:14:58	SELECT medicine_id, name, stock_quantity FROM Medicines WHERE sto...	13 row(s) returned	0.000 sec / 0.000 sec
12	20:18:43	SELECT medicine_id, name FROM Medicines WHERE medicine_id NOT IN (...)	1 row(s) returned	0.000 sec / 0.000 sec
13	20:20:05	SELECT sup.supplier_id, sup.name AS supplier_name, SUM(p.quantity) AS...	4 row(s) returned	0.015 sec / 0.000 sec

### -- 12. Medicine types where average price > ₹100

SELECT

```
type AS medicine_type,  
AVG(unit_price) AS average_price  
  
FROM Medicines  
  
GROUP BY type  
  
HAVING average_price > 100  
  
ORDER BY average_price DESC;
```

# Placement Readiness Assignment

## SQL

```

92 -- 12. Medicine types where average price > ₹100
93 • SELECT
94     type AS medicine_type,
95     AVG(unit_price) AS average_price
96 FROM Medicines
97 GROUP BY type
98 HAVING average_price > 100

```

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

medicine_type	average_price

Action Output

#	Time	Action	Message	Duration / Fetch
10	20:14:16	SELECT m.type AS medicine_type, SUM(s.quantity) AS total_units_sold FRO...	4 row(s) returned	0.000 sec / 0.000 sec
11	20:14:58	SELECT medicine_id, name, stock_quantity FROM Medicines WHERE sto...	13 row(s) returned	0.000 sec / 0.000 sec
12	20:18:43	SELECT medicine_id, name FROM Medicines WHERE medicine_id NOT IN (...)	1 row(s) returned	0.000 sec / 0.000 sec
13	20:20:05	SELECT sup.supplier_id, sup.name AS supplier_name, SUM(p.quantity) AS...	4 row(s) returned	0.015 sec / 0.000 sec
14	20:21:08	SELECT type AS medicine_type, AVG(unit_price) AS average_price FROM M...	0 row(s) returned	0.000 sec / 0.000 sec

-- 13. List all purchase records along with medicine and supplier names

SELECT

```

p.purchase_id,
p.purchase_date,
p.batch_number,
p.quantity,
m.name AS medicine_name,
s.name AS supplier_name

```

FROM Purchases p

JOIN Medicines m ON p.medicine\_id = m.medicine\_id

JOIN Suppliers s ON p.supplier\_id = s.supplier\_id

ORDER BY p.purchase\_date DESC;

```

101 -- 13. List all purchase records along with medicine and supplier names
102 • SELECT
103     p.purchase_id,
104     p.purchase_date,
105     p.batch_number,
106     p.quantity,
107     m.name AS medicine_name,

```

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

purchase_id	purchase_date	batch_number	quantity	medicine_name	supplier_name
39	2024-04-20	B039AM	60	Doxycycline 100mg	PharmaLink India
38	2024-04-18	B038AL	100	Pantoprazole 40mg	CureWell Distributors
37	2024-04-15	B037AK	90	Montelukast 10mg	NovaMed Solutions
36	2024-04-12	B036AJ	50	Clindamycin 300mg	Vitalis Pharma

Action Output

#	Time	Action	Message	Duration / Fetch
11	20:14:58	SELECT medicine_id, name, stock_quantity FROM Medicines WHERE sto...	13 row(s) returned	0.000 sec / 0.000 sec
12	20:18:43	SELECT medicine_id, name FROM Medicines WHERE medicine_id NOT IN (...)	1 row(s) returned	0.000 sec / 0.000 sec
13	20:20:05	SELECT sup.supplier_id, sup.name AS supplier_name, SUM(p.quantity) AS...	4 row(s) returned	0.015 sec / 0.000 sec
14	20:21:08	SELECT type AS medicine_type, AVG(unit_price) AS average_price FROM M...	0 row(s) returned	0.000 sec / 0.000 sec
15	20:22:34	SELECT p.purchase_id, p.purchase_date, p.batch_number, p.quantity, ...	40 row(s) returned	0.000 sec / 0.000 sec

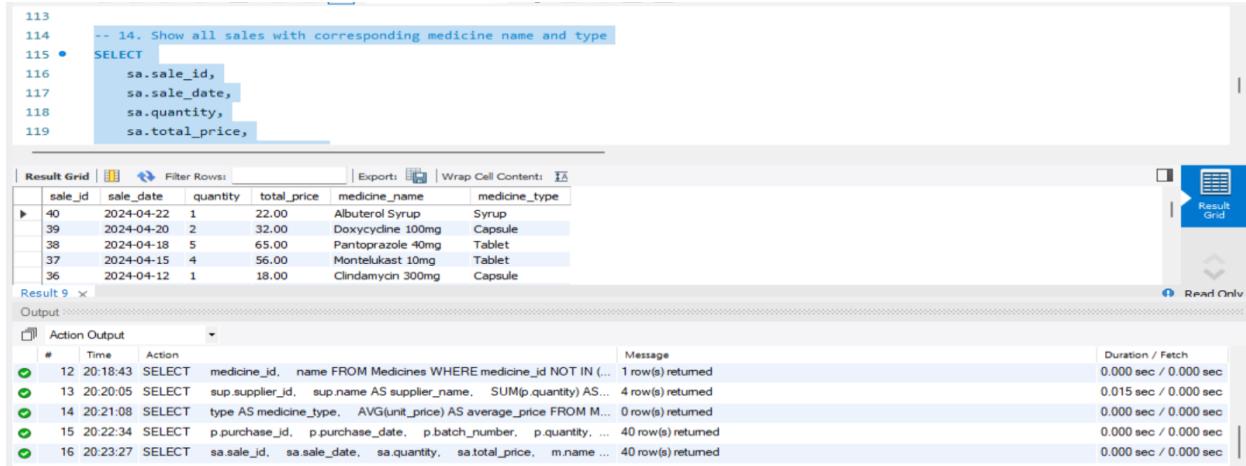
# Placement Readiness Assignment

## SQL

-- 14. Show all sales with corresponding medicine name and type

SELECT

```
sa.sale_id,  
sa.sale_date,  
sa.quantity,  
sa.total_price,  
m.name AS medicine_name,  
m.type AS medicine_type  
  
FROM Sales sa  
  
JOIN Medicines m ON sa.medicine_id = m.medicine_id  
  
ORDER BY sa.sale_date DESC;
```



The screenshot shows a SQL query editor with the following details:

- Code Area:** Displays the SQL query for question 14.
- Result Grid:** Shows the query results in a tabular format:

sale_id	sale_date	quantity	total_price	medicine_name	medicine_type
40	2024-04-22	1	22.00	Albuterol Syrup	Syrup
39	2024-04-20	2	32.00	Doxycycline 100mg	Capsule
38	2024-04-18	5	65.00	Pantoprazole 40mg	Tablet
37	2024-04-15	4	56.00	Montelukast 10mg	Tablet
36	2024-04-12	1	18.00	Clindamycin 300mg	Capsule

- Action Output:** Displays the execution log with the following entries:

#	Time	Action	Message	Duration / Fetch
12	20:18:43	SELECT medicine_id, name FROM Medicines WHERE medicine_id NOT IN (...)	1 row(s) returned	0.000 sec / 0.000 sec
13	20:20:05	SELECT sup.supplier_id, sup.name AS supplier_name, SUM(p.quantity) AS...	4 row(s) returned	0.015 sec / 0.000 sec
14	20:21:08	SELECT type AS medicine_type, AVG(unit_price) AS average_price FROM M...	0 row(s) returned	0.000 sec / 0.000 sec
15	20:22:34	SELECT p.purchase_id, p.purchase_date, p.batch_number, p.quantity, ...	40 row(s) returned	0.000 sec / 0.000 sec
16	20:23:27	SELECT sa.sale_id, sa.sale_date, sa.quantity, sa.total_price, m.name ...	40 row(s) returned	0.000 sec / 0.000 sec

-- 15. List medicines that will expire within the next 90 days

SELECT

```
medicine_id,  
name,  
type,
```

# Placement Readiness Assignment

## SQL

```
stock_quantity,  
expiry_date  
FROM Medicines  
WHERE expiry_date BETWEEN CURDATE() AND DATE_ADD(CURDATE(), INTERVAL 90 DAY)  
ORDER BY expiry_date;
```

```
125  
126 -- 15. List medicines that will expire within the next 90 days  
127 • SELECT  
128     medicine_id,  
129     name,  
130     type,  
131     stock_quantity,
```

medicine_id	name	type	stock_quantity	expiry_date
HOL1	HOL2	HOL3	HOL4	HOL5
•				
•				
•				
•				
•				
•				
•				
•				
•				

Medicines 10 ×  
Output  
Action Output

#	Time	Action	Message	Duration / Fetch
13	20:20:05	SELECT sup.supplier_id, sup.name AS supplier_name, SUM(p.quantity) AS...	4 row(s) returned	0.015 sec / 0.000 sec
14	20:21:08	SELECT type AS medicine_type, AVG(unit_price) AS average_price FROM M...	0 row(s) returned	0.000 sec / 0.000 sec
15	20:22:34	SELECT p.purchase_id, p.purchase_date, p.batch_number, p.quantity, ...	40 row(s) returned	0.000 sec / 0.000 sec
16	20:23:27	SELECT sa.sale_id, sa.sale_date, sa.quantity, sa.total_price, m.name ...	40 row(s) returned	0.000 sec / 0.000 sec
17	20:24:46	SELECT medicine_id, name, type, stock_quantity, expiry_date FROM ...	0 row(s) returned	0.000 sec / 0.000 sec

### -- 16. Count number of sales made in the last 30 days

```
SELECT  
    COUNT(*) AS sales_count,  
    SUM(total_price) AS total_revenue_last_30_days  
FROM Sales  
WHERE sale_date >= DATE_SUB(CURDATE(), INTERVAL 30 DAY);
```

```
138 • SELECT  
139     COUNT(*) AS sales_count,  
140     SUM(total_price) AS total_revenue_last_30_days  
141 FROM Sales  
142 WHERE sale_date >= DATE_SUB(CURDATE(), INTERVAL 30 DAY);  
143  
-- 17. List supplier names that start with the letter 'S'
```

sales_count	total_revenue_last_30_days
0	HOL1

Result 12 ×  
Output  
Action Output

#	Time	Action	Message	Duration / Fetch
15	20:22:34	SELECT p.purchase_id, p.purchase_date, p.batch_number, p.quantity, ...	40 row(s) returned	0.000 sec / 0.000 sec
16	20:23:27	SELECT sa.sale_id, sa.sale_date, sa.quantity, sa.total_price, m.name ...	40 row(s) returned	0.000 sec / 0.000 sec
17	20:24:46	SELECT medicine_id, name, type, stock_quantity, expiry_date FROM ...	0 row(s) returned	0.000 sec / 0.000 sec
18	20:26:22	SELECT COUNT(*) AS sales_count, SUM(total_price) AS total_revenue_last_... 1 row(s) returned	0.000 sec / 0.000 sec	0.000 sec / 0.000 sec
19	20:30:25	SELECT COUNT(*) AS sales_count, SUM(total_price) AS total_revenue_last_... 1 row(s) returned	0.000 sec / 0.000 sec	0.000 sec / 0.000 sec

# Placement Readiness Assignment

## SQL

-- 17. List supplier names that start with the letter 'S'

SELECT

```
supplier_id,  
name AS supplier_name,  
phone,  
city  
FROM Suppliers  
WHERE name LIKE 'S%';
```

The screenshot shows a SQL editor window with the following content:

```
144 -- 17. List supplier names that start with the letter 'S'  
145 • SELECT  
146     supplier_id,  
147     name AS supplier_name,  
148     phone,  
149     city  
150    FROM Suppliers
```

Below the code, the results are displayed in a grid:

supplier_id	supplier_name	phone	city
4	Sunrise Medicals	6543210987	Hyderabad
•	NULL	NULL	NULL

At the bottom, the "Output" section shows the execution log:

#	Time	Action	Message	Duration / Fetch
16	20:23:27	SELECT	sa.sale_id, sa.sale_date, sa.quantity, sa.total_price, m.name ... 40 row(s) returned	0.000 sec / 0.000 sec
17	20:24:46	SELECT	medicine_id, name, type, stock_quantity, expiry_date FROM ... 0 row(s) returned	0.000 sec / 0.000 sec
18	20:26:22	SELECT	COUNT() AS sales_count, SUM(total_price) AS total_revenue_last_... 1 row(s) returned	0.000 sec / 0.000 sec
19	20:30:25	SELECT	COUNT() AS sales_count, SUM(total_price) AS total_revenue_last_... 1 row(s) returned	0.000 sec / 0.000 sec
20	20:31:45	SELECT	supplier_id, name AS supplier_name, phone, city FROM Suppli... 1 row(s) returned	0.000 sec / 0.000 sec

-- 18. Convert all medicine names to uppercase

SELECT

```
medicine_id,  
UPPER(name) AS medicine_name_upper,  
type,  
unit_price,  
stock_quantity,  
expiry_date,
```

# Placement Readiness Assignment

SQL

manufacturer

FROM Medicines;

```
-- 18. Convert all medicine names to uppercase
SELECT
    medicine_id,
    UPPER(name) AS medicine_name_upper,
    type,
    unit_price,
    stock_quantity,
```

Result Grid | Filter Rows: | Export: | Wrap Cell Contents: | Result Grid | Read Only

	medicine_id	medicine_name_upper	type	unit_price	stock_quantity	expiry_date	manufacturer
▶	1	PARACETAMOL 500MG	Tablet	2.50	500	2027-12-31	HULL
	2	AMOXICILLIN 250MG	Capsule	8.00	300	2026-11-15	HULL
	3	OMEPRAZOLE 20MG	Tablet	12.00	200	2026-08-20	HULL
	4	METFORMIN 500MG	Tablet	5.50	400	2027-03-10	HULL
	5	ATORVASTATIN 10MG	Tablet	15.00	250	2026-10-05	HULL

Result 14 ×  
Output

#	Time	Action	Message	Duration / Fetch
17	20:24:46	SELECT medicine_id, name, type, stock_quantity, expiry_date FROM ...	0 row(s) returned	0.000 sec / 0.000 sec
18	20:26:22	SELECT COUNT(*) AS sales_count, SUM(total_price) AS total_revenue FROM ...	1 row(s) returned	0.000 sec / 0.000 sec
19	20:30:25	SELECT COUNT(*) AS sales_count, SUM(total_price) AS total_revenue FROM ...	1 row(s) returned	0.000 sec / 0.000 sec
20	20:31:45	SELECT supplier_id, name AS supplier_name, phone, city FROM Supplier	1 row(s) returned	0.000 sec / 0.000 sec
21	20:32:46	SELECT medicine_id, UPPER(name) AS medicine_name_upper, type, u...	21 row(s) returned	0.000 sec / 0.000 sec

-- 19. Show the top 3 best-selling medicines by quantity

## SELECT

```
m.name AS medicine_name,  
m.type,  
SUM(s.quantity) AS total_quantity_sold
```

FROM Sales s

JOIN Medicines m ON s.medicine\_id = m.medicine\_id

GROUP BY m.medicine\_id, m.name, m.type

ORDER BY total\_quantity\_sold DESC

LIMIT 3;

```
163
164 -- 19. Show the top 3 best-selling medicines by quantity
165 • SELECT
166     m.name AS medicine_name,
167     m.type,
168     SUM(s.quantity) AS total_quantity_sold
169 FROM Sales s
```

---

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

medicine_name	type	total_quantity_sold
Vitamin C 500mg	Tablet	35
Ibuprofen 400mg	Tablet	35
Paracetamol 500mg	Tablet	22

Result 15 ×

Output

Action Output

#	Time	Action	Message	Duration / Fetch
18	20:26:22	SELECT COUNT(*) AS sales_count, SUM(total_price) AS total_revenue_last_...	1 row(s) returned	0.000 sec / 0.000 sec
19	20:30:25	SELECT COUNT(*) AS sales_count, SUM(total_price) AS total_revenue_last_...	1 row(s) returned	0.000 sec / 0.000 sec
20	20:31:45	SELECT supplier_id, name AS supplier_name, phone, city FROM Suppli...	1 row(s) returned	0.000 sec / 0.000 sec
21	20:32:46	SELECT medicine_id, UPPER(name) AS medicine_name_upper, type, u...	21 row(s) returned	0.000 sec / 0.000 sec
22	20:33:47	SELECT m.name AS medicine_name, m.type, SUM(s.quantity) AS total_qu...	3 row(s) returned	0.000 sec / 0.000 sec

# Placement Readiness Assignment

## SQL

-- 20. List medicines that were purchased but never sold

SELECT

```
m.medicine_id,  
m.name,  
m.type,  
m.stock_quantity  
FROM Medicines m  
WHERE m.medicine_id IN (  
    SELECT DISTINCT p.medicine_id  
    FROM Purchases p  
)  
AND m.medicine_id NOT IN (  
    SELECT DISTINCT s.medicine_id  
    FROM Sales s  
    WHERE s.medicine_id IS NOT NULL  
);
```

The screenshot shows the execution of the provided SQL query in a database environment. The code is displayed in the top pane, and the results are shown in the bottom pane.

**Code (Top Pane):**

```
174  
175 -- 20. List medicines that were purchased but never sold  
176 • SELECT  
177     m.medicine_id,  
178     m.name,  
179     m.type,  
180     m.stock_quantity
```

**Results (Bottom Pane):**

medicine_id	name	type	stock_quantity
HULL	HULL	HULL	HULL

**Output (Bottom Pane):**

#	Time	Action	Message	Duration / Fetch
19	20:30:25	SELECT COUNT(*) AS sales_count, SUM(total_price) AS total_revenue FROM Sales	1 row(s) returned	0.000 sec / 0.000 sec
20	20:31:45	SELECT supplier_id, name AS supplier_name, phone, city FROM Suppliers	1 row(s) returned	0.000 sec / 0.000 sec
21	20:32:46	SELECT medicine_id, UPPER(name) AS medicine_name_upper, type, unit FROM Medicines	21 row(s) returned	0.000 sec / 0.000 sec
22	20:33:47	SELECT m.name AS medicine_name, m.type, SUM(s.quantity) AS total_quantity FROM Medicines m JOIN Sales s ON m.medicine_id = s.medicine_id GROUP BY m.medicine_id, m.type	3 row(s) returned	0.000 sec / 0.000 sec
23	20:34:48	SELECT m.medicine_id, m.name, m.type, m.stock_quantity FROM Medicines m WHERE m.medicine_id NOT IN (SELECT p.medicine_id FROM Purchases p)	0 row(s) returned	0.000 sec / 0.000 sec