

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;

The screenshot shows a SQL query in a text editor and its execution results in a table.

```
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;
```

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 table, the 'Action Output' section shows the execution log:

#	Time	Action	Message	Duration / Fetch
4	20:09:14	INSERT INTO Medicines (medicine_id, name, type, unit_price, stock_q...	INSERT INTO Medicines (medicine_id, name, type, unit_price, stock_quantity, expiry_date, manufacturer) VALUES (21, 'Amoxicillin 500mg', 'Capsule', 9.50, 200, '2027-06-30', 'MediPharm Ltd.')	
5	20:09:19	SET SQL_SAFE_UPDATES = 0		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

-- 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;

The screenshot shows a SQL query in a text editor and its execution results in a table.

```
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
```

supplier_name	total_quantity_supplied
Global Pharma Co.	730

Below the table, the 'Action Output' section shows the execution 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 IDE interface. The top pane contains the following SQL query:

```
-- 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 bottom pane displays the 'Result Grid' with the following data:

medicine_type	total_units_sold
Tablet	213
Capsule	19
Syrup	10
Inhaler	2

Below the result grid, the 'Action Output' pane 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 FRO...	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

);

The screenshot shows a SQL IDE with a query editor and a results grid. The query is as follows:

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

The results grid displays the following data:

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

The output pane shows the following actions:

#	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

);

The screenshot shows a SQL IDE with a query editor and a results grid. The query is as follows:

```
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
```

The results grid displays the following data:

medicine_id	name
21	Amoxicillin 500mg

The output pane shows the following actions:

#	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;

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

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

Result 6 ×

Output

#	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

medicine_type	average_price

Result 7

Output

#	Time	Action	Message	Duration / Fetch
10	20:14:16	SELECT m.type AS medicine_type, SUM(s.quantity) AS total_units_sold FROM Medicines m JOIN Suppliers s ON m.supplier_id = s.supplier_id	4 row(s) returned	0.000 sec / 0.000 sec
11	20:14:58	SELECT medicine_id, name, stock_quantity FROM Medicines WHERE stock_quantity > 100	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 (SELECT medicine_id FROM Purchases)	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 total_quantity FROM Suppliers s JOIN Purchases p ON s.supplier_id = p.supplier_id	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 Medicines GROUP BY type HAVING average_price > 100	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

purchase_id	purchase_date	batch_number	quantity	medicine_name	supplier_name
40	2024-04-22	B040AN	40	Albuterol Syrup	MediSource Co.
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

Result 8

Output

#	Time	Action	Message	Duration / Fetch
11	20:14:58	SELECT medicine_id, name, stock_quantity FROM Medicines WHERE stock_quantity > 100	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 (SELECT medicine_id FROM Purchases)	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 total_quantity FROM Suppliers s JOIN Purchases p ON s.supplier_id = p.supplier_id	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 Medicines GROUP BY type HAVING average_price > 100	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, 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	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 IDE interface. The top pane displays the following SQL query:

```
-- 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 bottom pane shows the 'Result Grid' with the following data:

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

Below the result grid, the 'Output' pane shows the execution log:

#	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;

The screenshot shows a SQL IDE with a query editor and a results pane. The query editor contains the following SQL code:

```
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,
```

The results pane shows the execution of the query. The top section displays the columns: medicine_id, name, type, stock_quantity, and expiry_date. Below this, the results are shown in a table format. The bottom section shows the execution log with the following entries:

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

The screenshot shows a SQL IDE with a query editor and a results pane. The query editor contains the following SQL code:

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

The results pane shows the execution of the query. The top section displays the columns: sales_count and total_revenue_last_30_days. Below this, the results are shown in a table format. The bottom section shows the execution log with the following entries:

#	Time	Action	Message	Duration / Fetch
15	20:22:34	SELECT	p.purchase_id, p.purchase_date, p.batch_number, p.quantity, ...	0.000 sec / 0.000 sec
16	20:23:27	SELECT	sa.sale_id, sa.sale_date, sa.quantity, sa.total_price, m.name ...	0.000 sec / 0.000 sec
17	20:24:46	SELECT	medicine_id, name, type, stock_quantity, expiry_date FROM ...	0.000 sec / 0.000 sec
18	20:26:22	SELECT	COUNT(*) AS sales_count, SUM(total_price) AS total_revenue_last...	0.000 sec / 0.000 sec
19	20:30:25	SELECT	COUNT(*) AS sales_count, SUM(total_price) AS total_revenue_last...	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 IDE interface. The top pane displays the following SQL query:

```
-- 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 bottom pane shows the 'Result Grid' with the following data:

supplier_id	supplier_name	phone	city
4	Sunrise Medicals	6543210987	Hyderabad

Below the result grid, the 'Suppliers 13' tab is active, showing the 'Output' pane with the following action output:

#	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;

```
153 -- 18. Convert all medicine names to uppercase
154 SELECT
155     medicine_id,
156     UPPER(name) AS medicine_name_upper,
157     type,
158     unit_price,
159     stock_quantity,
```

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

Result 14

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

-- 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
```

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

Result 15

#	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
19	20:30:25	SELECT	COUNT(*) AS sales_count, SUM(total_price) AS total_revenue_last...	1 row(s) returned
20	20:31:45	SELECT	supplier_id, name AS supplier_name, phone, city FROM Suppli...	1 row(s) returned
21	20:32:46	SELECT	medicine_id, UPPER(name) AS medicine_name_upper, type, u...	21 row(s) returned
22	20:33:47	SELECT	m.name AS medicine_name, m.type, SUM(s.quantity) AS total_qu...	3 row(s) returned

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 a SQL IDE interface. The top panel displays the following SQL query:

```
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
```

Below the editor, the 'Result Grid' panel shows a table with the following columns: medicine_id, name, type, stock_quantity. The table is currently empty, with all cells showing 'NULL'.

The 'Action Output' panel shows the execution log with the following entries:

#	Time	Action	Message	Duration / Fetch
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
23	20:34:48	SELECT	m.medicine_id, m.name, m.type, m.stock_quantity FROM Medic...	0 row(s) returned 0.000 sec / 0.000 sec