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;

```
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
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

Result 1

Output

Action Output

#	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		
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;

```
-- 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
```

supplier_name	total_quantity_supplied
Global Pharma Co.	730

Result 2

Output

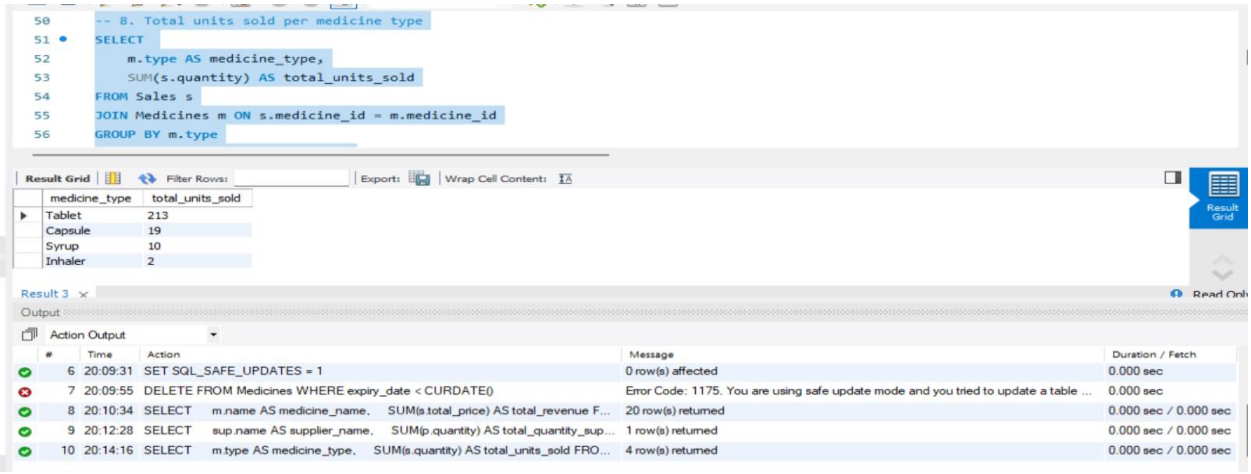
Action Output

#	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;
```



Subqueries:

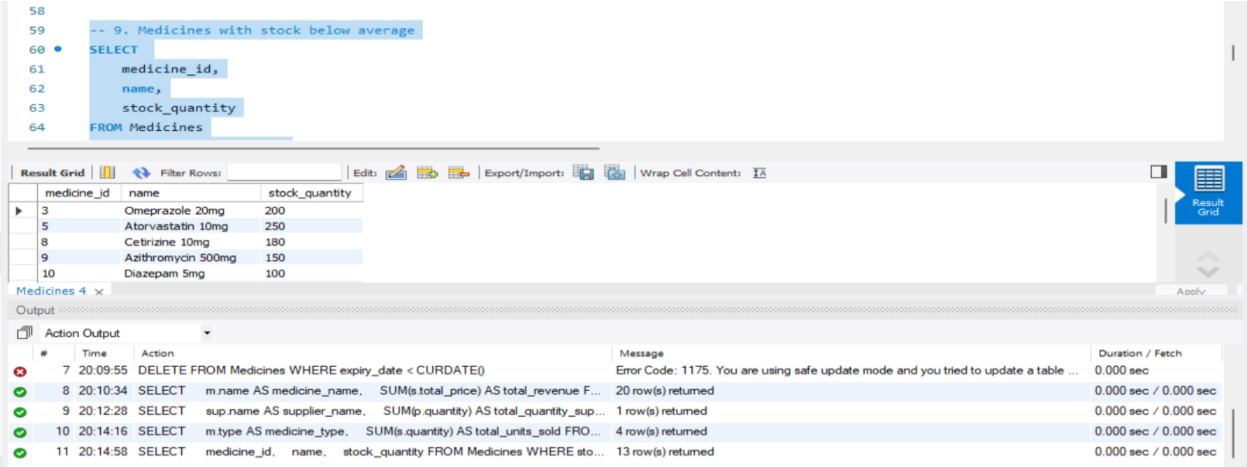
-- 9. Medicines with stock below average

```
SELECT

   medicine_id,

   name,

   stock_quantity

FROM Medicines

WHERE stock_quantity < (

   SELECT AVG(stock_quantity)

   FROM Medicines
```

Placement Readiness Assignment

SQL

);



-- 10. Medicines never sold

SELECT

medicine_id,

name

FROM Medicines

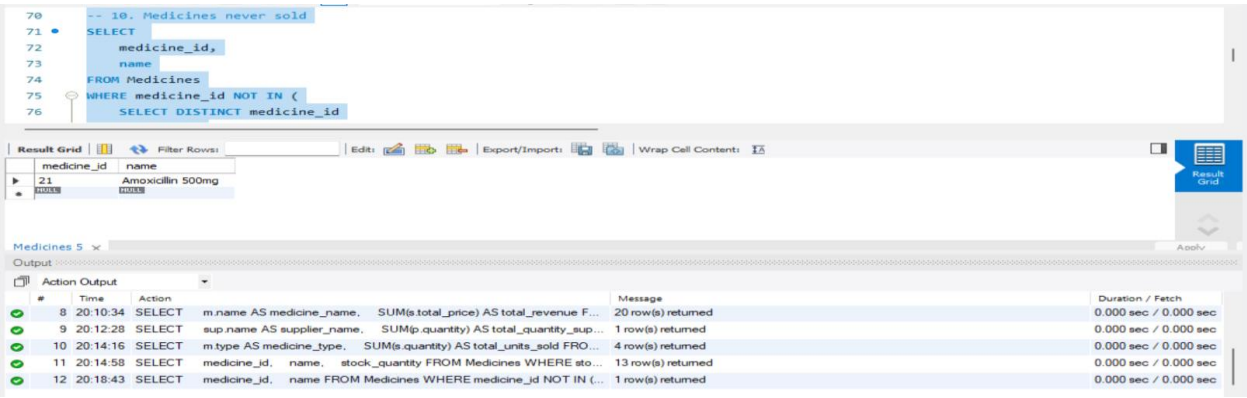
WHERE medicine_id NOT IN (

SELECT DISTINCT medicine_id

FROM Sales

WHERE medicine_id IS NOT NULL

);

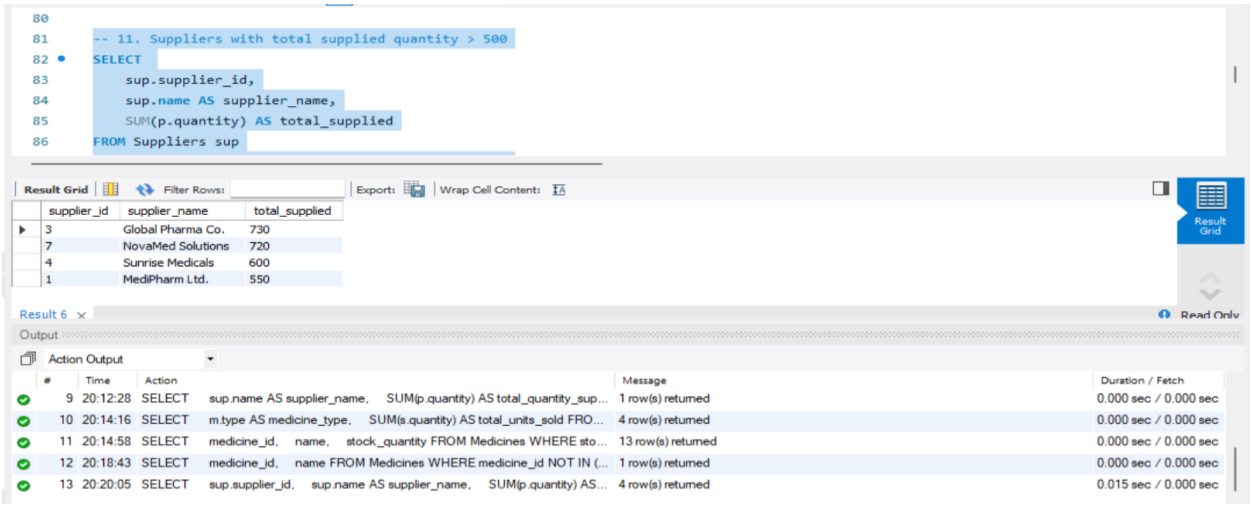


Placement Readiness Assignment

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;
```

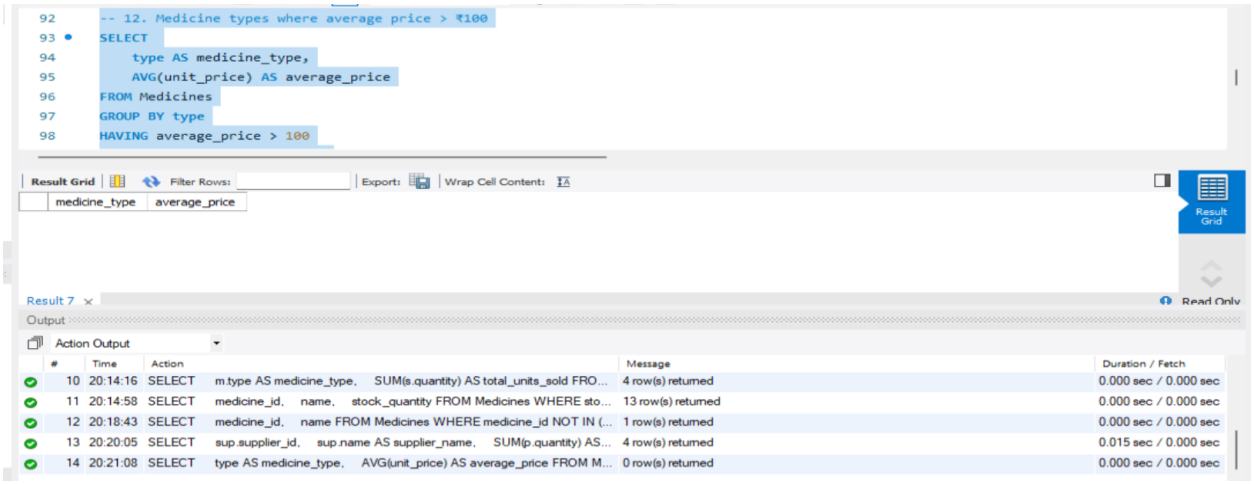


-- 12. Medicine types where average price > ₹100

```
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;
```

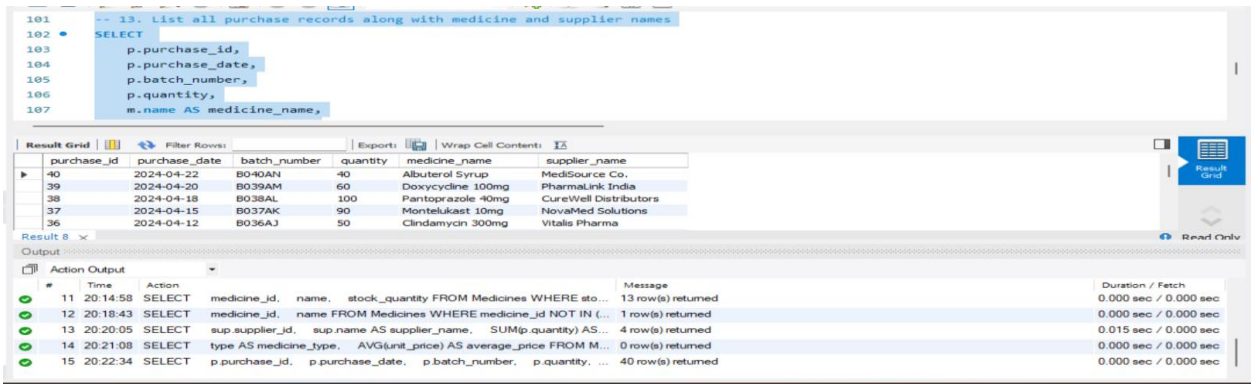
Placement Readiness Assignment

SQL



-- 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;
```

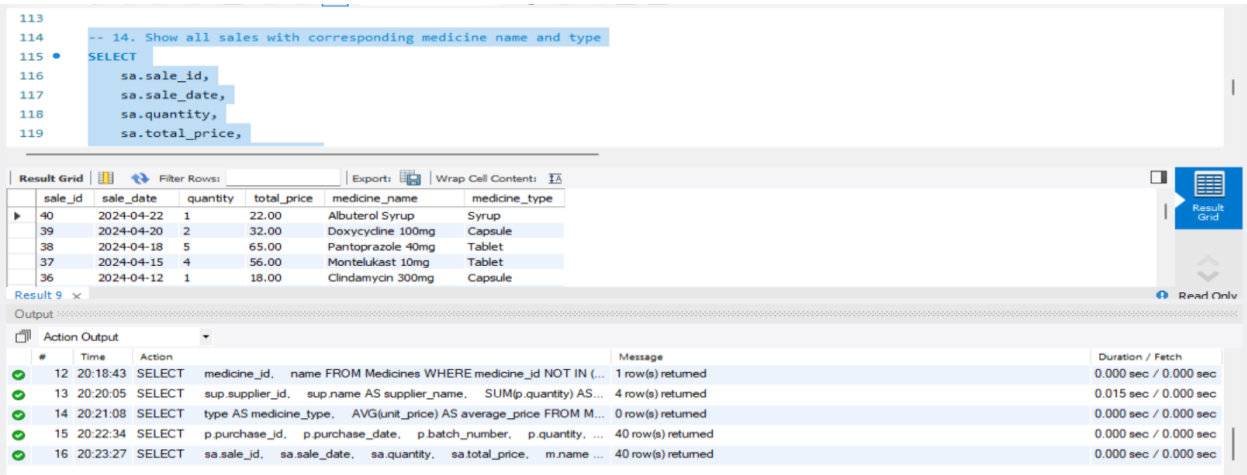


Placement Readiness Assignment

SQL

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

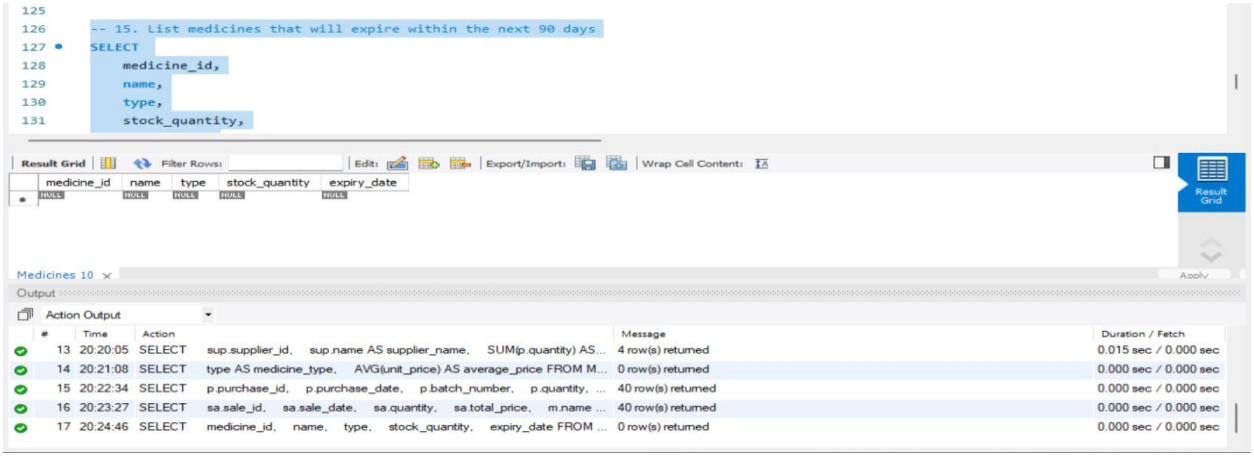
```
SELECT
    sa.sale_id,
    sa.sale_date,
    sa.quantity,
    sa.total_price,
    m.name AS medicine_name,
    m.type AS medicine_type
FROM Sales sa
JOIN Medicines m ON sa.medicine_id = m.medicine_id
ORDER BY sa.sale_date DESC;
```



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

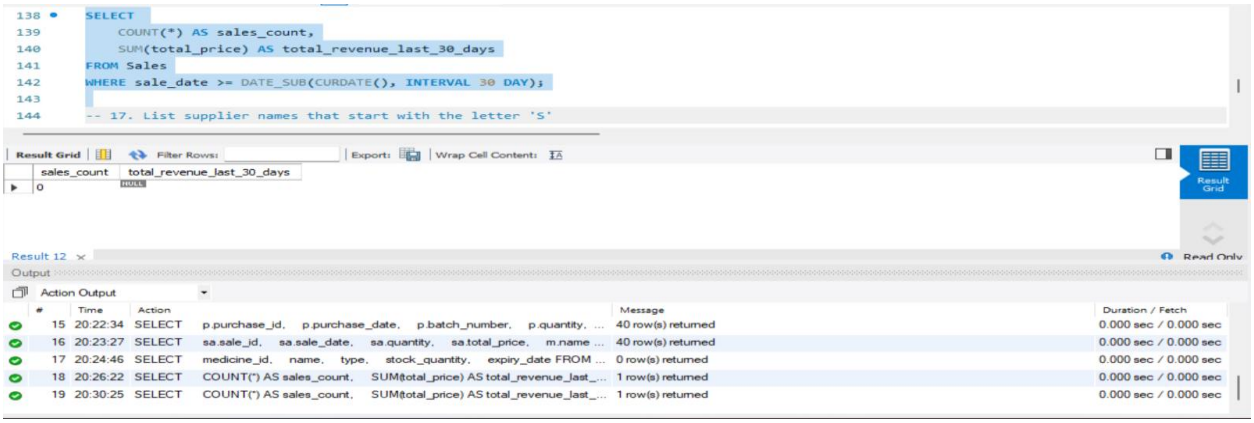
```
SELECT
    medicine_id,
    name,
    type,
```

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



-- 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);
```

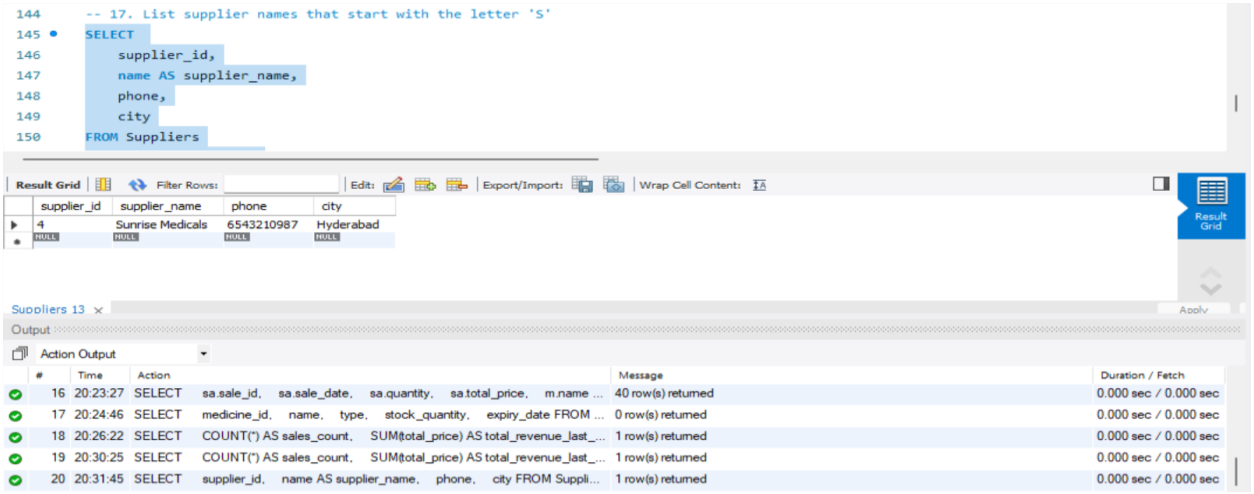


Placement Readiness Assignment

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%';
```



-- 18. Convert all medicine names to uppercase

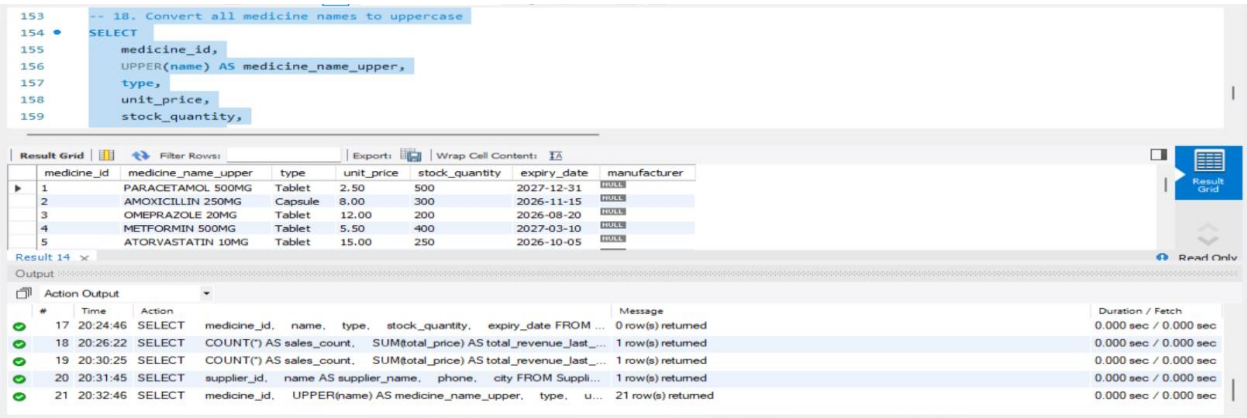
```
SELECT
    medicine_id,
    UPPER(name) AS medicine_name_upper,
    type,
    unit_price,
    stock_quantity,
    expiry_date,
```

Placement Readiness Assignment

SQL

manufacturer

FROM Medicines;



-- 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

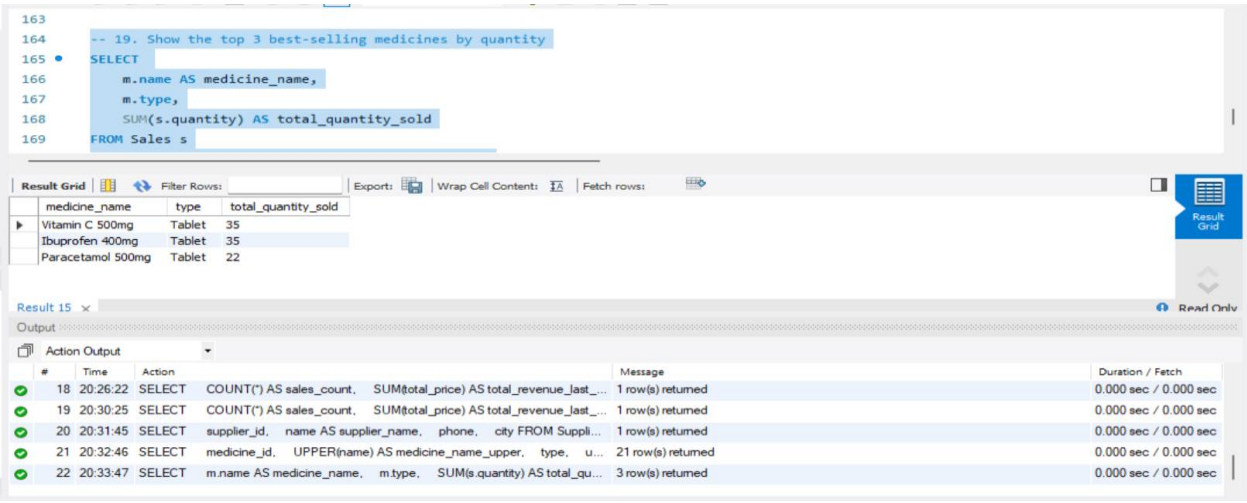
FROM Sales s

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;



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
);
```