03 SQL Interview Prep - Memberships

DDL & Sample Data

```sql
-- Drop tables if they already exist
DROP TABLE IF EXISTS transactions;
DROP TABLE IF EXISTS memberships;

-- Memberships table
CREATE TABLE memberships (
    membership_id INT PRIMARY KEY AUTO_INCREMENT,
    user_id INT NOT NULL,
    plan_id INT NOT NULL,
    start_date DATE NOT NULL,
    end_date DATE,
    status ENUM('ACTIVE', 'CANCELLED', 'EXPIRED') NOT NULL
);

-- Transactions table
CREATE TABLE transactions (
    transaction_id INT PRIMARY KEY AUTO_INCREMENT,
    membership_id INT NOT NULL,
    amount DECIMAL(10,2) NOT NULL,
    transaction_date DATE NOT NULL,
    payment_status ENUM('SUCCESS', 'FAILED', 'PENDING') NOT NULL,
    FOREIGN KEY (membership_id) REFERENCES memberships(membership_id)
);

-- Insert sample memberships
INSERT INTO memberships (user_id, plan_id, start_date, end_date, status) VALUES
(1, 101, '2024-01-01', '2024-12-31', 'ACTIVE'),
(2, 102, '2024-03-01', '2024-08-31', 'EXPIRED'),
(3, 101, '2024-02-01', '2024-05-31', 'CANCELLED'),
(4, 103, '2024-04-01', NULL, 'ACTIVE'),
(5, 102, '2024-06-01', '2024-11-30', 'ACTIVE'),
(6, 101, '2024-05-15', '2024-09-14', 'EXPIRED'),
(7, 104, '2024-07-01', NULL, 'ACTIVE'),
(8, 101, '2024-01-15', '2024-06-15', 'CANCELLED'),
(9, 103, '2024-08-01', NULL, 'ACTIVE'),
(10, 102, '2024-02-01', '2024-07-31', 'EXPIRED'),
(11, 101, '2024-08-15', NULL, 'ACTIVE'),
(12, 104, '2024-09-01', NULL, 'ACTIVE'),
(13, 102, '2024-01-01', '2024-04-30', 'CANCELLED'),
(14, 103, '2024-05-10', '2024-09-09', 'EXPIRED'),
```

```
(15, 101, '2024-07-01', NULL, 'ACTIVE');

-- Insert sample transactions
INSERT INTO transactions (membership_id, amount, transaction_date, payment_status)
VALUES
(1, 100.00, '2024-01-01', 'SUCCESS'),
(1, 100.00, '2024-02-01', 'SUCCESS'),
(1, 100.00, '2024-08-20', 'FAILED'),
(2, 200.00, '2024-03-01', 'SUCCESS'),
(2, 200.00, '2024-06-01', 'SUCCESS'),
(3, 100.00, '2024-02-01', 'SUCCESS'),
(3, 100.00, '2024-03-01', 'FAILED'),
(4, 300.00, '2024-04-01', 'SUCCESS'),
(4, 300.00, '2024-08-01', 'SUCCESS'),
(5, 200.00, '2024-06-01', 'SUCCESS'),
(5, 200.00, '2024-07-01', 'FAILED'),
(6, 100.00, '2024-05-15', 'SUCCESS'),
(7, 400.00, '2024-07-01', 'SUCCESS'),
(7, 400.00, '2024-08-01', 'SUCCESS'),
(8, 100.00, '2024-01-15', 'SUCCESS'),
(9, 300.00, '2024-08-01', 'PENDING'),
(10, 200.00, '2024-02-01', 'SUCCESS'),
(10, 200.00, '2024-05-01', 'FAILED'),
(11, 100.00, '2024-08-15', 'SUCCESS'),
(12, 400.00, '2024-09-01', 'SUCCESS'),
(13, 200.00, '2024-01-01', 'FAILED'),
(14, 300.00, '2024-05-10', 'SUCCESS'),
(15, 100.00, '2024-07-01', 'SUCCESS'),
(15, 100.00, '2024-08-01', 'FAILED');
```

SQL Practice Questions

Basic (warm-up)

1. **Find all active customers between two dates.**
   Example: active memberships between `'2024-06-01'` and `'2024-08-31'`.
2. **Get the total number of active memberships per plan.**
3. **List all users who have at least one failed payment.**
4. **Find the earliest subscription start date per user.**
5. **Count how many users have cancelled at least once.**

Intermediate

6. **Find users with active subscriptions but no successful transaction in the last 30 days.**
   (tests filtering with joins & nulls)
7. **Calculate the total revenue generated in July 2024.**

8. **Find the average revenue per active user.**
9. **Get the top 3 users who spent the most overall (lifetime revenue).**
10. **Identify churned users (who had memberships before but none active now).**

---

1. Find users with active subscriptions but no successful payment in the last 30 days.

👉 Tests **joins, filtering, date functions, and payment validation.**

```
SELECT m.user_id
FROM memberships m
LEFT JOIN transactions t
  ON m.membership_id = t.membership_id
  AND t.payment_status = 'SUCCESS'
  AND t.transaction_date ≥ CURRENT_DATE - INTERVAL '30 days'
WHERE m.status = 'ACTIVE'
  AND t.transaction_id IS NULL;
```

---

2. Find the total revenue per subscription plan for the last quarter.

👉 Tests **date filtering, grouping, and aggregations.**

```
SELECT m.plan_id,
       SUM(t.amount) AS total_revenue
FROM memberships m
JOIN transactions t
  ON m.membership_id = t.membership_id
WHERE t.payment_status = 'SUCCESS'
  AND t.transaction_date ≥ DATE_TRUNC('quarter', CURRENT_DATE) - INTERVAL '1
quarter'
  AND t.transaction_date < DATE_TRUNC('quarter', CURRENT_DATE)
GROUP BY m.plan_id;
```

---

3. Identify churned users (users who had a subscription but no active membership now).

👉 Tests **set operations or subqueries.**

```
SELECT DISTINCT m.user_id
FROM memberships m
WHERE m.user_id NOT IN (
    SELECT user_id FROM memberships WHERE status = 'ACTIVE'
);
```

4. Find the average lifetime revenue per user.

👉 Tests **window functions and aggregation.**

```sql
SELECT user_id,
       AVG(user_revenue) OVER () AS avg_lifetime_revenue
FROM (
    SELECT m.user_id, SUM(t.amount) AS user_revenue
    FROM memberships m
    JOIN transactions t
      ON m.membership_id = t.membership_id
    WHERE t.payment_status = 'SUCCESS'
    GROUP BY m.user_id
) AS sub;
```

5. Find users who downgraded (moved from higher-priced plan to lower-priced plan).

👉 Tests **self-joins, ordering, and comparisons.**

```sql
SELECT user_id, old.plan_id AS from_plan, new.plan_id AS to_plan
FROM memberships old
JOIN memberships new
  ON old.user_id = new.user_id
  AND new.start_date > old.start_date
JOIN (
    SELECT plan_id, AVG(amount) AS plan_price
    FROM memberships m
    JOIN transactions t ON m.membership_id = t.membership_id
    WHERE t.payment_status = 'SUCCESS'
    GROUP BY plan_id
) p1 ON old.plan_id = p1.plan_id
JOIN (
    SELECT plan_id, AVG(amount) AS plan_price
    FROM memberships m
    JOIN transactions t ON m.membership_id = t.membership_id
    WHERE t.payment_status = 'SUCCESS'
    GROUP BY plan_id
) p2 ON new.plan_id = p2.plan_id
WHERE p1.plan_price > p2.plan_price;
```