Technical Deep-Dive Q&A: Customer360 SQL Project (Snowflake)

Q1: How did you design the schema for this project?

A: I used a normalized star-schema style layout. The central table is `customers`, which links to `orders`, `logins`, and `support_tickets` via foreign keys. This approach supports scalable joins, simplifies analytical queries, and reflects real-world user activity domains.

Q2: How did you handle customers who had no logins or orders?

A: I used LEFT JOINs to ensure that customers without any related activity still appeared in the result sets. For example, in the churn risk and funnel stage queries, I used CASE WHEN and EXISTS to safely identify presence or absence of activity.

Q3: What challenges can arise with aggregating across multiple joins?

A: Aggregating after multiple LEFT JOINs can cause row multiplication if not handled properly. For instance, a customer with multiple logins and multiple orders can produce a Cartesian explosion. I avoided this by:

- Using aggregate functions like MAX, MIN, and COUNT carefully
- Performing grouping at the customer level
- Using EXISTS subqueries instead of joins in some cases (e.g., funnel stages)

Q4: Why did you use DATEDIFF instead of filtering directly by date?

A: DATEDIFF lets me compute the time elapsed from key dates (e.g., login_time, order_date) to CURRENT_DATE. This is more flexible than static comparisons and helps track thresholds like 30/60-day inactivity windows for churn detection.

Q5: How would this scale in a production environment?

A: In production, I would:

- Partition or cluster large tables like orders by customer_id or order_date
- Materialize heavy queries as views or tables
- Schedule incremental loads with tools like dbt or Airflow
- Implement role-based access control and masking policies in Snowflake

Q6: How did you simulate realistic data?

A: I manually inserted realistic but varied data covering all scenarios:

- Customers with no orders or logins
- Multiple logins per user
- Orders with different statuses (shipped, cancelled)
- Support tickets spread across customers

This enabled me to test conditional logic (e.g., COUNT only "cancelled" orders).

Q7: How would you test or validate these queries?

A: I used controlled sample data so I could predict the correct output. To validate:

- I cross-verified COUNTs and DATEDIFF results manually
- I incrementally built and tested each filter and aggregation step
- I used LIMIT, ORDER BY, and subqueries to verify intermediate logic

Q8: Can you integrate this with visualization tools?

A: Yes. Snowflake connects easily with BI tools like Power BI, Tableau, or Sigma. I can expose these SQL queries as views and connect them to dashboards that visualize:

- Churn over time
- Funnel stage conversion rates
- High-value customer cohorts