**Section 1 — Conceptual Understanding**

Answer briefly (2-3 sentences each):

1. In your own words, describe each SDLC phase.  
   How do these phases connect in a real project like the one you worked on?

Sdlc phases – initiate, plan, design, develop, test, deploy, maintain

Initiate is where we brain storm the ps and try to solve it, find solutions and how to build the application

Plan – this is where business is analysed, use cases/stories are written the SRS(software requirement specification), BRD(buss req doc), FRD, documentations are written to be clear of what the stakeholders want, and this is passed down to other phases to check and validate

Design, - design the project, system arch is designed | UI/UX of the project is done, prototyping is done

Develop, project is developed using sde, they write code

Test – check if there are any bugs, If all features are working with various modes such as functional testing, Blackbox testing, unit testing, regression and more

Deploy – project is deployed so that it can be reached to users

Maintain- keep checking for any discrepancies and updating the application

1. What’s the purpose of using Git branches and pull requests instead of committing directly to main?

So different people can work on different functions (isolation) and not disrupt each other with errors

1. Why is documentation (README, CONTRIBUTING) important even in small projects?

It helps us to understand the project quicker with a readme, we can get a brief gist about what the project is, how it was built or what tools, libraries and technologies they’ve used.

1. What is the advantage of using virtual environments in Python?

So that we can keep each projects isolated, install only the libraries that are required for that project.

1. How does your data ingestion script ensure that only clean, valid data reaches the database?
2. When writing SQL queries, how do you decide whether to use WHERE or HAVING?

Where – for filtering before any group by

Having – for filtering after group by

1. What’s the purpose of indexing in a database, and when could it slow queries down instead?

**Section 2 — Hands-On Confidence**

Rate yourself from 1 (low) to 5 (high) on each area:

| **Skill Area** | **1** | **2** | **3** | **4** | **5** |
| --- | --- | --- | --- | --- | --- |
| Understanding SDLC phases and Agile flow |  |  |  |  |  |
| Git workflow: branching, commits, merges |  |  |  |  |  |
| Python setup & environment management |  |  |  |  |  |
| Data ingestion using pandas + SQLAlchemy |  |  |  |  |  |
| SQL queries – SELECT, JOIN, GROUP BY, CTE |  |  |  |  |  |
| Translating business questions into SQL |  |  |  |  |  |

Add one sentence below each rating explaining *why* you chose that number.

**Section 3 — Mini Review Challenge**

To check your practical readiness, attempt the following small tasks.  
Do them in your own repo or a scratch workspace — the goal is understanding, not perfection.

1. **Git:**
   * Create a new branch, make a small change in the README, and open a pull request.
   * Merge it cleanly into main.
   * Note anything that confused you during the process.
2. **Python:**
   * Write a quick script to load a new CSV (returns.csv) into a new table in Postgres.
   * Include validation (e.g., no negative values or missing IDs).
3. **SQL:**
   * Write queries for:  
     a. Top 5 products by total revenue.  
     b. Customers who haven’t ordered in the last 30 days.  
     c. Average order value per region.

Save your answers and code in a folder named week1\_review\_challenge/.

**Section 4 — Personal Reflection**

Write short paragraphs for each prompt:

* **Top 3 things I learned this week:**  
  (Be specific — e.g., *how indexing changed query speed*.)
* **The hardest concept for me was:**  
  (Explain what made it hard — syntax, logic, or theory.)
* **How I applied the SDLC mindset to my project:**  
  (Which stage are you strongest in, and which needs work?)
* **If I repeated this week, I’d improve by:**  
  (Think of a workflow, coding habit, or tool you’d use better.)

🧩 1. Top-Spending Customers (using a CTE)

Find the top 3 customers who spent the most money across all orders.

💡 Hint:

Use a CTE to calculate total\_spent per customer (sum of quantity \* unit\_price).

Then, select the top 3 from that CTE using ORDER BY total\_spent DESC LIMIT 3

with total\_spent as (

select

orders.customer\_id,

sum(order\_items.quantity \* order\_items.unit\_price) as total\_spent

from order\_items

join orders on orders.order\_id = order\_items.order\_id

group by orders.customer\_id

)

select \* from total\_spent order by total\_spent desc limit 3;

🧩 2. High-Value Orders (using a subquery)

List all orders whose total value is above the average order value.

💡 Hint:

Inner query → compute the average total\_price across all orders.

Outer query → filter orders whose total is greater than that average.

🧩 SQL Practice — Joins, Group By, and Aggregation

1. Customer Orders Summary

Retrieve each customer’s name and the total number of orders they have placed.

select customers.name, count(orders.order\_id) from orders

join customers on customers.customer\_id = orders.customer\_id

group by customers.name;

//tryno

2. Total Quantity per Product

For each product, calculate the total quantity sold across all orders.

select products.product\_id, products.product\_name, sum(order\_items.quantity) from products

join order\_items on products.product\_id = order\_items.product\_id

group by products.product\_name, products.product\_id ;

3. High-Spending Customers

List all customers who have spent more than 5,000 in total purchases.

customers,orders,order\_items

select customers.name, order\_items.total\_price from orders

join customers on orders.customer\_id = customers.customer\_id

join order\_items on order\_items.order\_id = orders.order\_id

where total\_price > 1000

group by customers.name , order\_items.total\_price ;

4. Revenue by Product Category

Show the total revenue generated for each product category, sorted from highest to lowest.

select products.category, sum (quantity\*unit\_price ) as revenue from products

join order\_items on products.product\_id = order\_items.product\_id

group by products.category;

5. Average Order Value per Customer

Calculate the average value of each customer’s orders.

6. Unordered Products

Find all products that have never been ordered.

order\_items, products

select products.product\_name

from products left join order\_items on products.product\_id = order\_items.product\_id

where order\_id is null;

7. Customers Without Orders

Display customers who haven’t placed any orders yet.

select customers.name

from customers left join orders on customers.customer\_id = orders.customer\_id

where order\_id is null;

8. Most Popular Product

Identify the product that has been ordered the most (by total quantity sold).

select products.product\_name, sum(order\_items.quantity) as quant from products

join order\_items on order\_items.product\_id = products.product\_id

group by product\_name

order by quant desc limit 1;

9. Top 5 Customers by Spending

Find the 5 customers who have spent the most money overall.

customers, orders, transactions

select customers.name, sum(transactions.amount) as total\_spent

from orders join customers on customers.customer\_id = orders.customer\_id

join transactions on transactions.order\_id = orders.order\_id

group by customers.name

order by total\_spent desc limit 5;

10. Monthly Revenue Trend

Calculate the total revenue for each month based on order dates.

-- Example implementation (assumes an `orders.order\_date` column exists):

select

date\_trunc('month', orders.order\_date) as month,

sum(order\_items.quantity \* order\_items.unit\_price) as revenue

from orders

join order\_items on orders.order\_id = order\_items.order\_id

group by date\_trunc('month', orders.order\_date)

order by month;

11. Category with Highest Average Price

For each category, compute the average product price and identify the highest one.

12. Customer–Category Combination

Find which product category each customer buys from the most.

13. Orders with Multiple Products

List all orders that contain more than one product.

14. Products Bought by a Specific Customer

Show all products purchased by the customer named “Maria White”.

15. Top-Selling Category by Revenue

Determine which product category generates the highest total revenue.

16. Inactive Customers in the Last 3 Months

List customers who have not made any purchases in the last three months.

17. Repeat Buyers

Identify customers who have placed more than one order.

18. Average Quantity per Order by Category

For each category, calculate the average quantity of products ordered per order.

19. Customer Count per City

Display the number of customers living in each city.

20. Products Purchased by All Customers

Find products that have been purchased by every customer at least once.