Here’s a **comprehensive set of SQL questions and solutions** based on the schema you've provided. The schema involves three tables:

* Customers(customer\_id, first\_name, last\_name, age, country)
* Orders(order\_id, item, amount, customer\_id)
* Shippings(shipping\_id, status, customer)

We'll cover **basic to advanced SQL queries**, including **SELECT, JOIN, GROUP BY, HAVING, aggregate functions, subqueries**, and more.

**🔹 1. Basic Queries**

**Q1. Retrieve all customers from the table.**

SELECT \* FROM Customers;

**Q2. Get the full names of customers from India.**

SELECT first\_name || ' ' || last\_name AS full\_name

FROM Customers

WHERE country = 'India';

**🔹 2. Filtering and Conditions**

**Q3. Find customers who are older than 30.**

SELECT \* FROM Customers

WHERE age > 30;

**Q4. List all orders with an amount greater than 1000.**

SELECT \* FROM Orders

WHERE amount > 1000;

**🔹 3. JOIN Operations**

**Q5. Get customer names and the items they ordered.**

SELECT c.first\_name, c.last\_name, o.item

FROM Customers c

JOIN Orders o ON c.customer\_id = o.customer\_id;

**Q6. List all customers and their shipping statuses.**

SELECT c.first\_name, c.last\_name, s.status

FROM Customers c

JOIN Shippings s ON c.customer\_id = s.customer;

**Q7. Get order details with corresponding customer name and shipping status.**

SELECT o.order\_id, o.item, o.amount, c.first\_name, c.last\_name, s.status

FROM Orders o

JOIN Customers c ON o.customer\_id = c.customer\_id

LEFT JOIN Shippings s ON c.customer\_id = s.customer;

**🔹 4. Aggregate Functions**

**Q8. Find the total number of orders placed.**

SELECT COUNT(\*) AS total\_orders FROM Orders;

**Q9. Get the total amount spent by each customer.**

SELECT c.customer\_id, c.first\_name, c.last\_name, SUM(o.amount) AS total\_spent

FROM Customers c

JOIN Orders o ON c.customer\_id = o.customer\_id

GROUP BY c.customer\_id, c.first\_name, c.last\_name;

**Q10. Find the average age of all customers.**

SELECT AVG(age) AS average\_age FROM Customers;

**🔹 5. GROUP BY and HAVING**

**Q11. Get countries with more than 2 customers.**

SELECT country, COUNT(\*) AS num\_customers

FROM Customers

GROUP BY country

HAVING COUNT(\*) > 2;

**Q12. Find customers who placed more than 2 orders.**

SELECT c.customer\_id, c.first\_name, COUNT(\*) AS order\_count

FROM Customers c

JOIN Orders o ON c.customer\_id = o.customer\_id

GROUP BY c.customer\_id, c.first\_name

HAVING COUNT(\*) > 2;

**🔹 6. Subqueries**

**Q13. Get customers who have not placed any orders.**

SELECT \* FROM Customers

WHERE customer\_id NOT IN (SELECT DISTINCT customer\_id FROM Orders);

**Q14. Get the most expensive order's details.**

SELECT \* FROM Orders

WHERE amount = (SELECT MAX(amount) FROM Orders);

**🔹 7. CASE and Conditional Logic**

**Q15. Show customer name along with a label: "Senior" if age > 50, else "Adult".**

SELECT first\_name, last\_name,

CASE

WHEN age > 50 THEN 'Senior'

ELSE 'Adult'

END AS category

FROM Customers;

**🔹 8. UPDATE and DELETE (DML)**

**Q16. Update shipping status to 1 (delivered) for a specific customer.**

UPDATE Shippings

SET status = 1

WHERE customer = 101;

**Q17. Delete all orders where amount is 0.**

DELETE FROM Orders

WHERE amount = 0;

**🔹 9. Sorting and Limiting**

**Q18. Get top 5 customers who spent the most.**

SELECT c.first\_name, c.last\_name, SUM(o.amount) AS total\_spent

FROM Customers c

JOIN Orders o ON c.customer\_id = o.customer\_id

GROUP BY c.customer\_id, c.first\_name, c.last\_name

ORDER BY total\_spent DESC

LIMIT 5;

**🔹 10. Advanced Joins**

**Q19. List all customers including those who haven't ordered or received shipping.**

SELECT c.first\_name, o.item, s.status

FROM Customers c

LEFT JOIN Orders o ON c.customer\_id = o.customer\_id

LEFT JOIN Shippings s ON c.customer\_id = s.customer;

**🔹 Bonus Challenge**

**Q20. For each country, find the average amount spent per customer.**

SELECT c.country, AVG(total\_spent) AS avg\_spent

FROM (

SELECT c.customer\_id, c.country, SUM(o.amount) AS total\_spent

FROM Customers c

JOIN Orders o ON c.customer\_id = o.customer\_id

GROUP BY c.customer\_id, c.country

) AS sub

GROUP BY country;

Would you like me to generate this as a PDF or CSV for a question bank?

Great! Let’s explore **LIMIT** and **OFFSET** in SQL with clear examples based on your Customers, Orders, and Shippings tables.

**🔹 What is LIMIT?**

* LIMIT restricts the number of rows returned by a query.

**🔹 What is OFFSET?**

* OFFSET skips a specified number of rows before starting to return rows.

**✅ Basic Syntax:**

SELECT columns

FROM table

LIMIT number

OFFSET number;

**🔹 Examples using your schema:**

**Q1. Get the first 5 customers from the Customers table:**

SELECT \* FROM Customers

LIMIT 5;

**Q2. Get the next 5 customers after skipping the first 5:**

SELECT \* FROM Customers

LIMIT 5 OFFSET 5;

**Q3. Show the top 3 highest amount orders:**

SELECT \* FROM Orders

ORDER BY amount DESC

LIMIT 3;

**Q4. Paginate the orders list — page 2 with 10 orders per page:**

SELECT \* FROM Orders

ORDER BY order\_id

LIMIT 10 OFFSET 10;

✅ *This skips the first 10 and returns the next 10 (i.e., page 2).*

**Q5. Get the last 5 customers (assuming highest customer\_id means latest):**

SELECT \* FROM Customers

ORDER BY customer\_id DESC

LIMIT 5;

**Bonus Tip: MySQL Alternative Short Syntax**

In MySQL, you can combine LIMIT and OFFSET in a shorter form:

-- Get 10 rows starting from the 6th row

SELECT \* FROM Customers

LIMIT 6, 10;

This is equivalent to:

SELECT \* FROM Customers

LIMIT 10 OFFSET 6;

Would you like me to create a pagination script with page input or convert this into a workbook?