### New Database Structure and Tables

#### Tables and Structure:

1. **Customers**
   * customer\_id (Primary Key)
   * first\_name
   * last\_name
   * email (Unique)
   * city
   * join\_date
2. **Orders**
   * order\_id (Primary Key)
   * customer\_id (Foreign Key referencing Customers)
   * order\_date
   * total\_amount
3. **Products**
   * product\_id (Primary Key)
   * product\_name
   * price
   * supplier\_id (Foreign Key referencing Suppliers)
   * stock\_quantity
4. **Suppliers**
   * supplier\_id (Primary Key)
   * supplier\_name
   * contact\_email (Unique)
   * city

**Table Data**

**1. Customers**

| **customer\_id** | **first\_name** | **last\_name** | **email** | **city** | **join\_date** | **phone\_number** |
| --- | --- | --- | --- | --- | --- | --- |
| 1 | John | Doe | john.doe@example.com | New York | 2023-06-15 | 1234567890 |
| 2 | Jane | Smith | jane.smith@example.com | Los Angeles | 2024-02-01 | 2345678901 |
| 3 | Michael | Johnson | michael.johnson@example.com | Chicago | 2023-12-20 | 3456789012 |
| 4 | Sarah | Davis | sarah.davis@example.com | New York | 2022-08-10 | 4567890123 |
| 5 | Emma | Wilson | emma.wilson@example.com | San Francisco | 2023-09-23 | 5678901234 |

**2. Orders**

| **order\_id** | **customer\_id** | **order\_date** | **total\_amount** |
| --- | --- | --- | --- |
| 101 | 1 | 2024-01-05 | 350.00 |
| 102 | 2 | 2024-02-15 | 220.00 |
| 103 | 3 | 2023-12-25 | 450.00 |
| 104 | 1 | 2024-03-01 | 300.00 |
| 105 | 5 | 2023-09-27 | 500.00 |

**3. Products**

| **product\_id** | **product\_name** | **price** | **supplier\_id** | **stock\_quantity** |
| --- | --- | --- | --- | --- |
| 201 | Wireless Earbuds | 150.00 | 301 | 25 |
| 202 | Bluetooth Speaker | 120.00 | 302 | 10 |
| 203 | Laptop Stand | 50.00 | 303 | 100 |
| 204 | Mechanical Keyboard | 200.00 | 301 | 15 |
| 205 | USB-C Hub | 80.00 | 302 | 5 |
| 206 | Noise-Cancelling Headphones | 220.00 | 301 | 8 |
| 207 | Portable Charger | 40.00 | 303 | 60 |

**4. Suppliers**

| **supplier\_id** | **supplier\_name** | **contact\_email** | **city** |
| --- | --- | --- | --- |
| 301 | ABC Suppliers | contact@abc.com | New York |
| 302 | XYZ Distributors | contact@xyz.com | Chicago |
| 303 | Global Tech Supplies | info@globaltech.com | San Francisco |

### Exercise Questions

1. **CREATE TABLE**: Create the **Customers** table with the structure provided. Ensure that the customer\_id is set as the primary key.
2. **CREATE TABLE**: Create the **Orders** table, setting up the foreign key constraint for customer\_id to reference the **Customers** table.
3. **ALTER TABLE**: Add a new column phone\_number to the **Customers** table, ensuring it has a unique constraint.
4. **DELETE**: Delete records from the **Customers** table where the customer’s city is unknown or set to NULL.
5. **UPDATE**: Increase the total\_amount of all orders placed after January 1, 2024, by 10%.
6. **UPDATE**: Write a query to update the stock\_quantity in **Products** for a product supplied by 'ABC Suppliers' to 50.
7. **Constraints**: Modify the **Products** table to add a check constraint that ensures price is greater than 0.
8. **WHERE Clause**: Retrieve all customers from the **Customers** table who joined after July 1, 2023, and reside in the city "New York".
9. **WHERE Clause**: Find all products that have a stock\_quantity less than 10.
10. **DISTINCT**: Retrieve a list of unique cities where suppliers are located.
11. **ORDER BY**: Write a query to retrieve all orders in the **Orders** table, sorted by order\_date in descending order.
12. **ORDER BY**: Retrieve all products sorted by price in ascending order, but list products with the same supplier\_id together.
13. **GROUP BY**: Calculate the total total\_amount spent by each customer, grouping by customer\_id.
14. **GROUP BY**: Retrieve the average price of products provided by each supplier and only include suppliers with at least 3 products.
15. **HAVING Clause**: List suppliers with an average product\_price of more than $200. Use **GROUP BY** and **HAVING** clauses.
16. **UPDATE with WHERE**: Update the price of all products to match the average price of products in their category if their current price is below this average.
17. **Complex Query with GROUP BY and HAVING**: Write a query to find customers who have spent more than the average spending of all customers. Use **GROUP BY** and **HAVING**.
18. **CREATE TABLE with DEFAULT Constraint**: Create a new table called **Reviews** to store product reviews. Include the following columns:

review\_id (Primary Key),

product\_id (Foreign Key referencing Products),

customer\_id (Foreign Key referencing Customers),

rating (an integer with a DEFAULT value of 3),

review\_text.

Ensure that rating has a check constraint to allow only values between 1 and 5.

1. Modify the **Products** table to add a new column, product\_category, with an ENUM constraint to allow only the following categories: 'Electronics', 'Accessories', and 'Home Office'.
2. In the **Orders** table, add a new column quantity with a CHECK constraint to ensure it only allows values greater than 0 and less than or equal to 100.
3. Alter the **Orders** table to set the order\_date column with a DEFAULT value of the current date.
4. Add a payment\_method column to the **Orders** table with an ENUM constraint allowing only 'Credit Card', 'Debit Card', 'PayPal', and 'Cash'. Set the default payment method to 'Credit Card'.
5. Create a new table called **Inventory** to track product stock with columns:

* location\_id (Primary Key),
* product\_id (Primary Key, Foreign Key referencing Products),
* stock\_level (Default value of 0),
* last\_restocked\_date.

Add a CHECK constraint to ensure that stock\_level is never negative.