Challenge: SQL Aggregate Functions and Data Constraints

Follow the instructions below to complete the challenge using a different database setup and sample data.

Instructions:

1. Database Setup:

- Use the following SQL script to create a new database named `SalesDB` and set up the necessary tables.

```
-- Create SalesDB database
CREATE DATABASE SalesDB;
USE SalesDB;

-- Create Products table
CREATE TABLE Products (
	product_id INT AUTO_INCREMENT,
	product_name VARCHAR(100) NOT NULL,
	category VARCHAR(50),
	price DECIMAL(10, 2) NOT NULL,
	PRIMARY KEY (product_id)
);

-- Insert sample data into Products table
INSERT INTO Products (product_name, category, price) VALUES
('Laptop', 'Electronics', 1200.00),
('Smartphone', 'Electronics', 800.00),
('Headphones', 'Electronics', 100.00),
('Shirt', 'Clothing', 25.00),
('Jeans', 'Clothing', 40.00),
('Sneakers', 'Footwear', 50.00),
('Watch', 'Accessories', 150.00),
('Backpack', 'Accessories', 80.00);

-- Create Sales table
CREATE TABLE Sales (
	sale_id INT AUTO_INCREMENT,
	product_id INT,
	quantity INT,
	sale_date DATE,
	PRIMARY KEY (sale_id),
	FOREIGN KEY (product_id) REFERENCES Products(product_id)
);
```

```
-- Insert sample data into Sales table
INSERT INTO Sales (product_id, quantity, sale_date) VALUES
(1, 2, '2024-06-01'),
(2, 1, '2024-06-02'),
(3, 3, '2024-06-03'),
(4, 5, '2024-06-04'),
(5, 4, '2024-06-05'),
(6, 2, '2024-06-06'),
(7, 1, '2024-06-07'),
(8, 3, '2024-06-08');
```

- Write SQL queries to answer the following questions:
    1. Calculate the total revenue generated from sales (`total_revenue`).
    2. Determine the average price of products sold (`average_price`).
    3. Find the maximum quantity of a single product sold (`max_quantity`).
    4. Identify the product with the highest total revenue (`top_product`).

- Create a new table named `Customers` to demonstrate various data constraints.
- This table should contain the following columns:

customer_id: A unique identifier for each customer, automatically generated by the database.
customer_name: The name of the customer, which must be provided..
email: The email address of the customer, which must be unique. This ensures that each email address is associated with only one customer, preventing duplicates.

- Add data to the `Customers` table, enforcing the necessary constraints.
- Attempt to insert data that violates each constraint and document the reasons for failure.

Composite Key Use Case:
- Write a SQL query to retrieve the total quantity of products sold for each category.
- Ensure that you join the `Sales` table with the `Products` table to gather relevant information.

Write Queries to answer the Following Questions:
1. What is the total revenue generated from all sales?
2. What is the average price of products sold?
3. Which product had the highest total revenue?
4. How many customers are there in the database?
5. What is the total quantity of products sold for the "Electronics" category?

- Push your SQL script file (`.sql`) to GitHub for review.
- Include comments or documentation to explain each step and query in your script.

Challenge Tips:
- Check your SQL syntax and query structure.
- Test each query thoroughly to ensure accuracy.
- Utilize aggregate functions effectively to derive meaningful insights.
- Pay attention to data constraints and handle potential errors appropriately.
- Use aliases and descriptive column names for clarity.

Best of luck!

Complete the challenge by following the instructions and tackling the test questions. Don't hesitate to seek assistance or collaborate with peers if needed.
Happy querying!