# Problem Statement 1: Building a Scalable E-Commerce Database with Azure SQL Database

Objective: Design and implement a relational database in Azure SQL for an e-commerce platform to manage customer orders, demonstrating security, scaling, and querying features.  
  
Scenario: You’re tasked with creating an e-commerce database for a small online store. The database should store customer information, products, and orders. Ensure it’s secure, scalable for peak traffic, and has a backup strategy.  
  
Data to Insert:  
  
Customers Table:  
CustomerID 1, Name 'Alice Johnson', Email 'alice@example.com'  
CustomerID 2, Name 'Bob Smith', Email 'bob@example.com'  
CustomerID 3, Name 'Charlie Davis', Email 'charlie@example.com'  
CustomerID 4, Name 'Dana Lee', Email 'dana@example.com'  
CustomerID 5, Name 'Eve Miller', Email 'eve@example.com'  
  
Products Table:  
ProductID 1, Name 'Laptop', Price 999.99  
ProductID 2, Name 'Mouse', Price 29.99  
ProductID 3, Name 'Keyboard', Price 49.99  
ProductID 4, Name 'Monitor', Price 199.99  
ProductID 5, Name 'Headphones', Price 79.99  
  
Orders Table:  
OrderID 1, CustomerID 1, ProductID 1, Quantity 1, OrderDate '2025-10-01'  
OrderID 2, CustomerID 2, ProductID 2, Quantity 2, OrderDate '2025-10-02'  
OrderID 3, CustomerID 3, ProductID 3, Quantity 1, OrderDate '2025-10-03'  
OrderID 4, CustomerID 4, ProductID 4, Quantity 3, OrderDate '2025-10-04'  
OrderID 5, CustomerID 5, ProductID 5, Quantity 1, OrderDate '2025-10-05'  
  
Steps to Solve:  
1. Create the Azure SQL Database.  
2. Secure the Database using firewall rules and Azure AD authentication.  
3. Design and populate the schema using SSMS.  
4. Scale and optimize the performance.  
5. Backup and restore to test recovery.  
6. Connect SSMS and write SQL queries to answer the following assessment questions.

## Assessment Questions:

1. Write a SQL query to display each customer’s name along with the products they ordered, the quantity, and the order date.  
2. Find customers who have ordered more than one quantity of any product.  
3. Write a query to calculate the total amount spent by each customer.  
4. Display the list of products that were ordered on or after October 3, 2025.  
5. Write a query to count how many total orders each customer has placed.  
6. Write a query to find which product has generated the highest total sales value.  
7. Create a view named “CustomerOrderSummary” showing customer names, total quantity ordered, and total spent.  
8. Demonstrate how to back up your database and restore it to a new instance.  
9. Scale your database to higher vCores, rerun a large query, and note any performance improvements.  
10. Document challenges faced during query design, scaling, and restoration.