**Exercise 01: Creating a View**

Consider the Northwind database, and suppose you want to create a view that provides information about customers and their orders. The view should include the following columns:

* Customer ID
* Company Name
* Contact Name
* Order ID
* Order Date
* Shipped Date

Create a view named **CustomerOrderView** to achieve this.

**Hint:** You will need to join the **Customers** and **Orders** tables.

**Exercise 02: Creating a SalesSummaryView**

Consider the Northwind database, and create a view named **SalesSummaryView** that includes the following columns:

* Product ID
* Product Name
* Category Name
* Total Quantity Sold
* Total Sales Amount

**Hint:** You'll need to join the **Products**, **OrderDetails**, and **Categories** tables. Additionally, you should use aggregate functions to calculate the total quantity sold and total sales amount.

**Exercise 03: Creating an EmployeeTerritoryView**

Consider the Northwind database, and create a view named **EmployeeTerritoryView** that includes the following columns:

* Employee ID
* Employee Name (concatenation of FirstName and LastName)
* Territory ID
* Territory Description

Create the view to include only those employees who have territories assigned.

**Hint:** You'll need to join the **Employees**, **EmployeeTerritories**, and **Territories** tables. Use the **IS NOT NULL** condition to filter out employees without territories.

**Exercise 04: Creating a SupplierProductView**

Consider the Northwind database and create a view named **SupplierProductView** that includes the following columns:

* Supplier ID
* Supplier Name
* Product ID
* Product Name
* Category Name

Create the view to include only those products that have an associated supplier.

**Hint:** You'll need to join the **Suppliers**, **Products**, and **Categories** tables. Use the **IS NOT NULL** condition to filter out products without a supplier.

**Exercise 05: Creating a HighValueCustomersView**

Consider the Northwind database and create a view named **HighValueCustomersView** that includes the following columns:

* Customer ID
* Company Name
* Contact Name
* Total Order Amount

Create the view to include only those customers whose total order amount is greater than a specified threshold (e.g., $10,000).

**Hint:** You'll need to join the **Customers, Orders** and **OrderDetails** tables. Use aggregate functions to calculate the total order amount and a **HAVING** clause to filter customers based on the total order amount.

**Exercise 06: Creating an EmployeeOrderView**

Consider the Northwind database and create a view named **EmployeeOrderView** that includes the following columns:

* Employee ID
* Employee Name (concatenation of FirstName and LastName)
* Order ID
* Order Date
* Ship City

Create the view to include only those orders where the employee responsible for the order is known.

**Hint:** You'll need to join the **Employees** and **Orders** tables. Use the **IS NOT NULL** condition to filter out orders without an associated employee.

**Exercise 07: Creating a DiscontinuedProductView**

Consider the Northwind database and create a view named **DiscontinuedProductView** that includes the following columns:

* Product ID
* Product Name
* Supplier ID
* Supplier Name
* Category Name

Create the view to include only those products that are discontinued.

**Hint:** You'll need to join the **Products**, **Suppliers**, and **Categories** tables. Use the **WHERE** clause to filter out only discontinued products.

**Exercise 08: Creating a EmployeeHierarchyView**

Consider the Northwind database and create a view named **EmployeeHierarchyView** that includes the following columns:

* Employee ID
* Employee Name (concatenation of FirstName and LastName)
* Manager ID
* Manager Name

Create the view to represent the hierarchy of employees and their managers.

**Hint:** You'll need to join the **Employees** table with itself using aliases to represent employees and their managers.

**Exercise 09: Creating a ProductSalesView**

Consider the Northwind database and create a view named **ProductSalesView** that includes the following columns:

* Product ID
* Product Name
* Total Sales Amount

Create the view to display the total sales amount for each product.

**Hint:** You'll need to join the **Products** and **OrderDetails** tables. Use aggregate functions to calculate the total sales amount.

**Exercise 10: Creating a CustomerRegionView**

Consider the Northwind database and create a view named **CustomerRegionView** that includes the following columns:

* Customer ID
* Company Name
* Region

Create the view to display the region for each customer.

**Hint:** You'll need to join the **Customers** and **Region** tables.

**Exercise 11: Creating a CustomerProductView**

Consider the Northwind database and create a view named **CustomerProductView** that includes the following columns:

* Customer ID
* Company Name
* Product ID
* Product Name
* Quantity Ordered

Create the view to display a list of products ordered by each customer.

**Hint:** You'll need to join the **Customers**, **Orders**, **OrderDetails**, and **Products** tables.

**Exercise 12: Creating a EmployeeSalesView**

Consider the Northwind database and create a view named **EmployeeSalesView** that includes the following columns:

* Employee ID
* Employee Name (concatenation of FirstName and LastName)
* Total Sales Amount

Create the view to display the total sales amount for each employee.

**Hint:** You'll need to join the **Employees**, **Orders**, and **OrderDetails** tables. Use aggregate functions to calculate the total sales amount.

**Exercise 13: Creating a CategoryProductCountView**

Consider the Northwind database and create a view named **CategoryProductCountView** that includes the following columns:

* Category ID
* Category Name
* Number of Products

Create the view to display the count of products in each category.

**Hint:** You'll need to join the **Categories** and **Products** tables. Use the **COUNT** function to calculate the number of products.

**Exercise 14: Creating a CustomerEmployeeView**

Consider the Northwind database and create a view named **CustomerEmployeeView** that includes the following columns:

* Customer ID
* Company Name
* Employee ID
* Employee Name (concatenation of FirstName and LastName)

Create the view to display the employee responsible for each customer.

**Hint:** You'll need to join the **Customers** and **Employees** tables.

**Exercise 15: Creating a ProductOrderHistoryView**

Consider the Northwind database and create a view named **ProductOrderHistoryView** that includes the following columns:

* Product ID
* Product Name
* Order ID
* Order Date
* Quantity Ordered

Create the view to display the order history for each product.

**Hint:** You'll need to join the **Products**, **OrderDetails**, and **Orders** tables.

**Exercise 16: Creating a HighValueProductsView**

Consider the Northwind database and create a view named **HighValueProductsView** that includes the following columns:

* Product ID
* Product Name
* Total Sales Amount

Create the view to display products with a total sales amount exceeding $5,000.

**Hint:** You'll need to join the **Products** and **OrderDetails** tables. Use aggregate functions and a **HAVING** clause to filter products based on the total sales amount.

**Exercise 17: Creating a EmployeeTerritoryCountView**

Consider the Northwind database and create a view named **EmployeeTerritoryCountView** that includes the following columns:

* Employee ID
* Employee Name (concatenation of FirstName and LastName)
* Number of Territories

Create the view to display the count of territories assigned to each employee.

**Hint:** You'll need to join the **Employees** and **EmployeeTerritories** tables. Use the **COUNT** function to calculate the number of territories.

**Exercise 18: Creating a CategoryRevenueView**

Consider the Northwind database and create a view named **CategoryRevenueView** that includes the following columns:

* Category ID
* Category Name
* Total Revenue

Create the view to display the total revenue for each category.

**Hint:** You'll need to join the **Categories**, **Products**, **OrderDetails**, and **Orders** tables. Use aggregate functions to calculate the total revenue.

**Exercise 19: Creating a EmployeeAverageOrderView**

Consider the Northwind database and create a view named **EmployeeAverageOrderView** that includes the following columns:

* Employee ID
* Employee Name (concatenation of FirstName and LastName)
* Average Order Amount

Create the view to display the average order amount for each employee.

**Hint:** You'll need to join the **Employees**, **Orders**, and **OrderDetails** tables. Use aggregate functions to calculate the average order amount.

**Exercise 20: Creating a SupplierCountryView**

Consider the Northwind database and create a view named **SupplierCountryView** that includes the following columns:

* Supplier ID
* Supplier Name
* Country

Create the view to display the country for each supplier.

**Hint:** You'll need to join the **Suppliers** and **Countries** tables.