**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.

**Answers:**

-- Creating the view

CREATE VIEW CustomerOrderView AS

SELECT

c.CustomerID,

c.CompanyName,

c.ContactName,

o.OrderID,

o.OrderDate,

o.ShippedDate

FROM

Customers c

JOIN

Orders o ON c.CustomerID = o.CustomerID;

**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.

**Answers:**

-- Creating the view

CREATE VIEW SalesSummaryView AS

SELECT

p.ProductID,

p.ProductName,

c.CategoryName,

SUM(od.Quantity) AS TotalQuantitySold,

SUM(od.Quantity \* od.UnitPrice) AS TotalSalesAmount

FROM

Products p

JOIN

Categories c ON p.CategoryID = c.CategoryID

JOIN

OrderDetails od ON p.ProductID = od.ProductID

GROUP BY

p.ProductID, p.ProductName, c.CategoryName;

Now, let's test the view by selecting data from it:

-- Selecting data from the view

SELECT \* FROM SalesSummaryView;

**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.

**Answers:**

-- Creating the view

CREATE VIEW EmployeeTerritoryView AS

SELECT

e.EmployeeID,

CONCAT(e.FirstName, ' ', e.LastName) AS EmployeeName,

et.TerritoryID,

t.TerritoryDescription

FROM

Employees e

JOIN

EmployeeTerritories et ON e.EmployeeID = et.EmployeeID

JOIN

Territories t ON et.TerritoryID = t.TerritoryID

WHERE

et.TerritoryID IS NOT NULL;

Now, let's test the view by selecting data from it:

-- Selecting data from the view

SELECT \* FROM EmployeeTerritoryView;

**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.

**Answers:**

-- Creating the view

CREATE VIEW SupplierProductView AS

SELECT

s.SupplierID,

s.SupplierName,

p.ProductID,

p.ProductName,

c.CategoryName

FROM

Suppliers s

JOIN

Products p ON s.SupplierID = p.SupplierID

JOIN

Categories c ON p.CategoryID = c.CategoryID

WHERE

p.SupplierID IS NOT NULL;

Now, let's test the view by selecting data from it:

-- Selecting data from the view

SELECT \* FROM SupplierProductView;

**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** and **Orders** tables. Use aggregate functions to calculate the total order amount and a **HAVING** clause to filter customers based on the total order amount.

**Answers:**

-- Creating the view

CREATE VIEW HighValueCustomersView AS

SELECT

c.CustomerID,

c.CompanyName,

c.ContactName,

SUM(od.Quantity \* od.UnitPrice) AS TotalOrderAmount

FROM

Customers c

JOIN

Orders o ON c.CustomerID = o.CustomerID

JOIN

OrderDetails od ON o.OrderID = od.OrderID

GROUP BY

c.CustomerID, c.CompanyName, c.ContactName

HAVING

TotalOrderAmount > 10000;

Now, let's test the view by selecting data from it:

-- Selecting data from the view

SELECT \* FROM HighValueCustomersView;

**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.

**Answers:**

-- Creating the view

CREATE VIEW EmployeeOrderView AS

SELECT

e.EmployeeID,

CONCAT(e.FirstName, ' ', e.LastName) AS EmployeeName,

o.OrderID,

o.OrderDate,

o.ShipCity

FROM

Employees e

JOIN

Orders o ON e.EmployeeID = o.EmployeeID

WHERE

e.EmployeeID IS NOT NULL;

**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.

**Answers:**

-- Creating the view

CREATE VIEW DiscontinuedProductView AS

SELECT

p.ProductID,

p.ProductName,

p.SupplierID,

s.SupplierName,

c.CategoryName

FROM

Products p

JOIN

Suppliers s ON p.SupplierID = s.SupplierID

JOIN

Categories c ON p.CategoryID = c.CategoryID

WHERE

p.Discontinued = 1;

**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.

**Answers:**

-- Creating the view

CREATE VIEW EmployeeHierarchyView AS

SELECT

e.EmployeeID,

CONCAT(e.FirstName, ' ', e.LastName) AS EmployeeName,

e.ReportsTo AS ManagerID,

CONCAT(m.FirstName, ' ', m.LastName) AS ManagerName

FROM

Employees e

LEFT JOIN

Employees m ON e.ReportsTo = m.EmployeeID;

**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.

**Answers:**

-- Creating the view

CREATE VIEW ProductSalesView AS

SELECT

p.ProductID,

p.ProductName,

SUM(od.Quantity \* od.UnitPrice) AS TotalSalesAmount

FROM

Products p

JOIN

OrderDetails od ON p.ProductID = od.ProductID

GROUP BY

p.ProductID, p.ProductName;

**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.

**Answers:**

-- Creating the view

CREATE VIEW CustomerRegionView AS

SELECT

c.CustomerID,

c.CompanyName,

r.RegionDescription AS Region

FROM

Customers c

JOIN

Orders o ON c.CustomerID = o.CustomerID

JOIN

Employees e ON o.EmployeeID = e.EmployeeID

JOIN

EmployeeTerritories et ON e.EmployeeID = et.EmployeeID

JOIN

Territories t ON et.TerritoryID = t.TerritoryID

JOIN

Region r ON t.RegionID = r.RegionID;

**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.

**Answers:**

-- Creating the view

CREATE VIEW CustomerProductView AS

SELECT

c.CustomerID,

c.CompanyName,

od.ProductID,

p.ProductName,

od.Quantity

FROM

Customers c

JOIN

Orders o ON c.CustomerID = o.CustomerID

JOIN

OrderDetails od ON o.OrderID = od.OrderID

JOIN

Products p ON od.ProductID = p.ProductID;

**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.

**Answers:**

-- Creating the view

CREATE VIEW EmployeeSalesView AS

SELECT

e.EmployeeID,

CONCAT(e.FirstName, ' ', e.LastName) AS EmployeeName,

SUM(od.Quantity \* od.UnitPrice) AS TotalSalesAmount

FROM

Employees e

JOIN

Orders o ON e.EmployeeID = o.EmployeeID

JOIN

OrderDetails od ON o.OrderID = od.OrderID

GROUP BY

e.EmployeeID, EmployeeName;

**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.

**Answers:**

-- Creating the view

CREATE VIEW CategoryProductCountView AS

SELECT

c.CategoryID,

c.CategoryName,

COUNT(p.ProductID) AS NumberOfProducts

FROM

Categories c

LEFT JOIN

Products p ON c.CategoryID = p.CategoryID

GROUP BY

c.CategoryID, c.CategoryName;

**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.

**Answers:**

-- Creating the view

CREATE VIEW CustomerEmployeeView AS

SELECT

c.CustomerID,

c.CompanyName,

e.EmployeeID,

CONCAT(e.FirstName, ' ', e.LastName) AS EmployeeName

FROM

Customers c

LEFT JOIN

Orders o ON c.CustomerID = o.CustomerID

LEFT JOIN

Employees e ON o.EmployeeID = e.EmployeeID;

**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.

**Answers:**

-- Creating the view

CREATE VIEW ProductOrderHistoryView AS

SELECT

p.ProductID,

p.ProductName,

od.OrderID,

o.OrderDate,

od.Quantity

FROM

Products p

JOIN

OrderDetails od ON p.ProductID = od.ProductID

JOIN

Orders o ON od.OrderID = o.OrderID;

**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.

**Answers:**

-- Creating the view

CREATE VIEW HighValueProductsView AS

SELECT

p.ProductID,

p.ProductName,

SUM(od.Quantity \* od.UnitPrice) AS TotalSalesAmount

FROM

Products p

JOIN

OrderDetails od ON p.ProductID = od.ProductID

GROUP BY

p.ProductID, p.ProductName

HAVING

TotalSalesAmount > 5000;

**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.

**Answers:**

-- Creating the view

CREATE VIEW EmployeeTerritoryCountView AS

SELECT

e.EmployeeID,

CONCAT(e.FirstName, ' ', e.LastName) AS EmployeeName,

COUNT(et.TerritoryID) AS NumberOfTerritories

FROM

Employees e

LEFT JOIN

EmployeeTerritories et ON e.EmployeeID = et.EmployeeID

GROUP BY

e.EmployeeID, EmployeeName;

**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.

**Answers:**

-- Creating the view

CREATE VIEW CategoryRevenueView AS

SELECT

c.CategoryID,

c.CategoryName,

SUM(od.Quantity \* od.UnitPrice) AS TotalRevenue

FROM

Categories c

JOIN

Products p ON c.CategoryID = p.CategoryID

JOIN

OrderDetails od ON p.ProductID = od.ProductID

JOIN

Orders o ON od.OrderID = o.OrderID

GROUP BY

c.CategoryID, c.CategoryName;

**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.

**Answers:**

-- Creating the view

CREATE VIEW EmployeeAverageOrderView AS

SELECT

e.EmployeeID,

CONCAT(e.FirstName, ' ', e.LastName) AS EmployeeName,

AVG(od.Quantity \* od.UnitPrice) AS AverageOrderAmount

FROM

Employees e

JOIN

Orders o ON e.EmployeeID = o.EmployeeID

JOIN

OrderDetails od ON o.OrderID = od.OrderID

GROUP BY

e.EmployeeID, EmployeeName;

**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.

**Answers:**

-- Creating the view

CREATE VIEW SupplierCountryView AS

SELECT

s.SupplierID,

s.SupplierName,

c.Country

FROM

Suppliers s

JOIN

Countries c ON s.Country = c.Country;