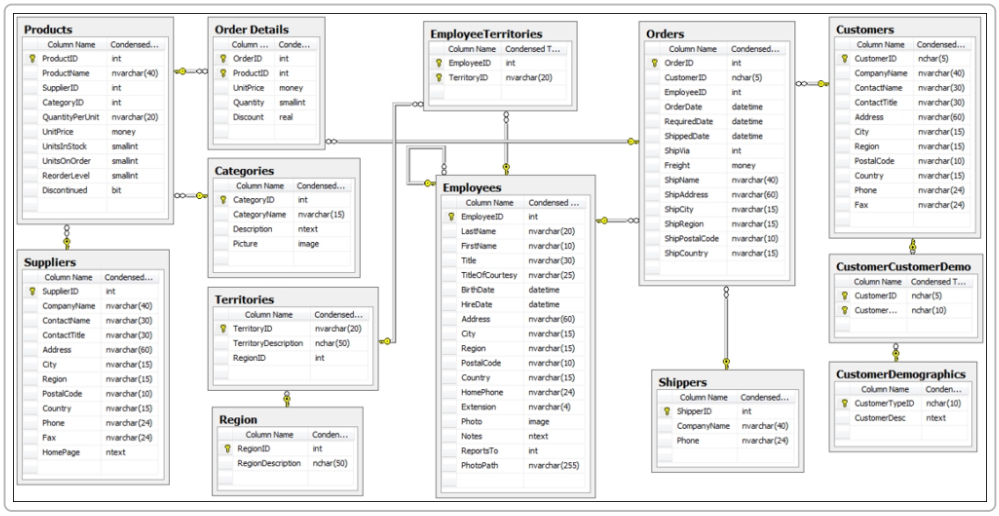
ChatGPT

Northwind Database Schema

Overview: The Northwind sample database contains 14 interrelated tables representing a fictitious trading business. The schema includes tables for orders, customers, employees, products, suppliers, shipping, regions, territories, and linking tables for many-to-many relationships. The entity-relationship diagram below provides a high-level overview of the tables, their columns, and how they connect via primary/foreign keys



. In the diagram, key symbols denote primary keys, and lines indicate foreign key relationships (pointing from the child/foreign key to the parent table). Each table is described in detail in the sections that follow.

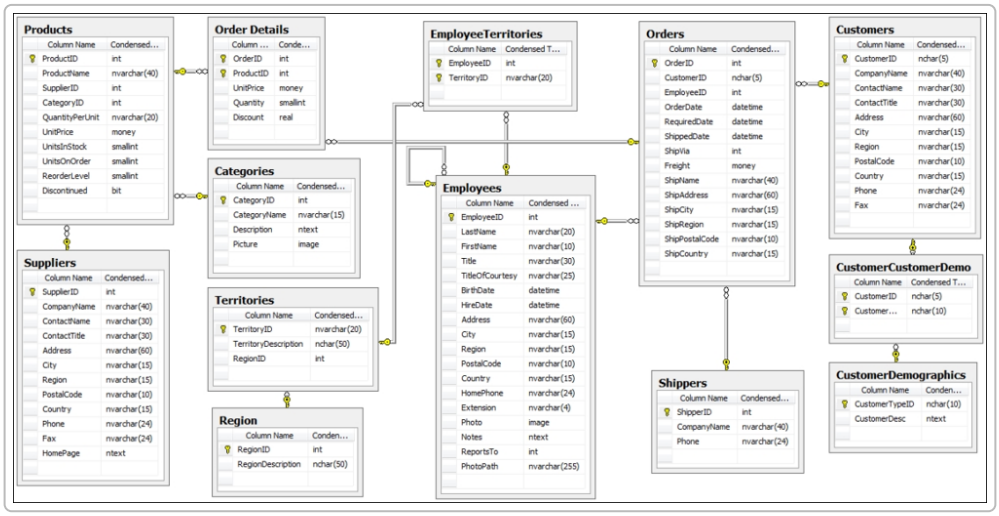


Figure: Entity-Relationship diagram for the Northwind database, illustrating all tables, their key fields, and relationships (one-to-many lines and joining tables for many-to-many relationships).

Orders

- **Description:** The Orders table stores sales orders placed by customers [1]. Each record represents a customer order, including which customer placed it, the employee who took the order, order dates, shipping information, and freight charge [2] [3].
- **Columns:**
- **OrderID** – *INT*, the primary key order identifier (unique order number) [4].
- **CustomerID** – *CHAR(5)*, reference to the customer who placed the order (links to Customers.CustomerID) [5].
- **EmployeeID** – *INT*, reference to the employee who processed the order (links to Employees.EmployeeID) [5].
- **OrderDate** – *DATE/DATETIME*, date the order was placed [4].
- **RequiredDate** – *DATE/DATETIME*, date by which the customer requested the order (required delivery date) [6].
- **ShippedDate** – *DATE/DATETIME*, date the order was shipped (if it has been shipped).
- **ShipVia** – *INT*, reference to the shipping company used (links to Shippers.ShipperID) [7].
- **Freight** – *MONEY/CURRENCY*, freight charge for shipping the order [5].
- **ShipName** – *VARCHAR(40)*, the name of the recipient or shipping destination.
- **ShipAddress** – *VARCHAR(60)*, the street address where the order is shipped [6].
- **ShipCity** – *VARCHAR(15)*, the city for delivery.
- **ShipRegion** – *VARCHAR(15)*, the region/state for delivery (if applicable).
- **ShipPostalCode** – *VARCHAR(10)*, the postal code for delivery.
- **ShipCountry** – *VARCHAR(15)*, the country for delivery.
- **Primary Key:** `OrderID` (unique identifier for orders).
- **Foreign Keys:**
- `CustomerID` → **Customers**(`CustomerID`) – identifies the customer placing the order [8].
- `EmployeeID` → **Employees**(`EmployeeID`) – identifies the employee who handled the order [8].
- `ShipVia` → **Shippers**(`ShipperID`) – identifies the carrier (shipper) used for delivery [8].
- **Relationships:** One **Customer** can have many **Orders** (1-to-many, a customer may place multiple orders) [8]. One **Employee** can handle many **Orders** (1-to-many, an employee may be responsible for multiple orders) [8]. One **Shipper** (shipping company) ships many **Orders** (1-to-many) [8]. Each **Order** can have multiple **Order_Details** line items (1-to-many relationship from Orders to Order_Details) – i.e. an order can contain multiple products [8]. (The many-to-many relationship between Orders and Products is handled via the Order_Details table.)

Order_Details

- **Description:** The Order_Details table (also called “Order Details”) stores the line items for each order [9]. Each record in Order_Details represents a single product line in a specific order, including the product, quantity, price, and any discount applied [10]. This table links orders to products, enabling a many-to-many relationship between **Orders** and **Products** (one order can include many products, and one product can appear in many orders).
- **Columns:**
- **OrderID** – *INT*, foreign key identifying the order (links to Orders.OrderID) [11].
- **ProductID** – *INT*, foreign key identifying the product (links to Products.ProductID) [12].
- **UnitPrice** – *MONEY/CURRENCY*, the price of the product per unit at the time of the order [13].
- **Quantity** – *INT*, the quantity of the product ordered [14].

- **Discount** – *REAL*, the discount applied to that line (stored as a percentage or fraction, e.g. 0.1 for 10% off) [15] [16].
- **Primary Key:** Composite of `OrderID` + `ProductID` (together uniquely identifying a line item) [17].
- **Foreign Keys:**
- `OrderID` → **Orders**(`OrderID`) – associates the line item with a specific order [17].
- `ProductID` → **Products**(`ProductID`) – identifies which product is being ordered on this line [17].
- **Relationships: Order_Details** is a *linking table* between **Orders** and **Products**. For each **Order** (parent), there can be multiple Order_Details records (children) – one for each product in that order (one-to-many from Orders to Order_Details) [17]. Similarly, each **Product** can appear in many Order_Details records (one-to-many from Products to Order_Details) [18]. Through Order_Details, **Orders** and **Products** are related in a many-to-many manner (each order can include multiple products, and each product can be sold in multiple orders).

Customers

- **Description:** The Customers table stores information about the clients/customers who purchase products from Northwind [1]. It includes the company name, contact person, contact details, and address for each customer [19]. Each customer is identified by a unique CustomerID code (a 5-character identifier) [20].
- **Columns:**
- **CustomerID** – *CHAR(5)*, primary key customer code (unique 5-character identifier for the customer) [20].
- **CompanyName** – *VARCHAR(40)*, the name of the customer's company [21].
- **ContactName** – *VARCHAR(30)*, the name of the primary contact person at the customer.
- **ContactTitle** – *VARCHAR(30)*, the job title of the contact person.
- **Address** – *VARCHAR(60)*, the street address of the customer [22].
- **City** – *VARCHAR(15)*, the city of the address.
- **Region** – *VARCHAR(15)*, the region or state (if applicable) for the address [23].
- **PostalCode** – *VARCHAR(10)*, the postal (ZIP) code.
- **Country** – *VARCHAR(15)*, the country.
- **Phone** – *VARCHAR(24)*, the phone number (including area code).
- **Fax** – *VARCHAR(24)*, the fax number (if any).
- **Primary Key:** `CustomerID` (the unique customer code).
- **Foreign Keys:** *(None)* – the Customers table does not have foreign key columns; it stands alone and is referenced by other tables.
- **Relationships:** One **Customer** can place many **Orders** (Customers:Orders is a one-to-many relationship, with CustomerID as the link) [24]. The Customers table is on the "one" side of that relationship (each order refers to one customer). Additionally, Customers can be associated with zero, one, or multiple **CustomerDemographics** (customer categories) through the **CustomerCustomerDemo** linking table (enabling a many-to-many relationship between customers and customer demographics).

Employees

- **Description:** The Employees table contains details about the Northwind employees (staff), such as their names, titles, and contact/personal information [25]. This includes each employee's name, title,

birth date, hire date, and contact details. The table also features a self-referential hierarchy: an employee may report to another employee (e.g. a supervisor) [26] .

- **Columns:**
- **EmployeeID** – *INT*, primary key for employees (unique ID number for each employee) [27] .
- **LastName** – *VARCHAR(20)*, employee's last name.
- **FirstName** – *VARCHAR(10)*, employee's first name.
- **Title** – *VARCHAR(30)*, job title of the employee (e.g. Sales Representative) [28] .
- **TitleOfCourtesy** – *VARCHAR(25)*, courtesy title (e.g. Mr., Ms., Dr.) used in salutations [28] .
- **BirthDate** – *DATE*, date of birth of the employee.
- **HireDate** – *DATE*, date the employee was hired.
- **Address** – *VARCHAR(60)*, street address of the employee.
- **City** – *VARCHAR(15)*, city of residence.
- **Region** – *VARCHAR(15)*, region or state.
- **PostalCode** – *VARCHAR(10)*, postal code.
- **Country** – *VARCHAR(15)*, country.
- **HomePhone** – *VARCHAR(24)*, home phone number.
- **Extension** – *VARCHAR(4)*, internal phone extension.
- **Photo** – *BLOB/BINARY*, employee photograph (binary data; in some versions a picture or image file) [29] .
- **Notes** – *TEXT*, miscellaneous notes about the employee (e.g. background, additional info).
- **ReportsTo** – *INT*, manager's EmployeeID (foreign key referencing another entry in **Employees**) [26] . This field is null if the employee does not report to anyone (top of hierarchy).
- **PhotoPath** – *VARCHAR(255)*, path or file name of the employee's photo (if stored as a file path in some versions).
- **Primary Key:** `EmployeeID`.
- **Foreign Keys:**
- `ReportsTo` → **Employees**(`EmployeeID`) – a self-referencing foreign key indicating this employee's supervisor/manager (references another row in the same Employees table) [30] .
- **Relationships:** Employees have a **self-referential** one-to-many relationship: one employee (e.g. a manager) can have many direct-report employees (subordinates) who reference them via the ReportsTo field. In other words, the table represents a hierarchy where an employee can oversee multiple employees (Employees.ReportsTo forms this recursive relationship) [26] . Employees also relate to other tables: one **Employee** can be responsible for many **Orders** (1-to-many, as Orders.EmployeeID is a foreign key) [8] . Moreover, Employees are linked to **Territories** through the **EmployeeTerritories** join table, allowing a many-to-many relationship (an employee may cover multiple sales territories, and each territory can have multiple assigned employees) [31] .

Products

- **Description:** The Products table stores details of the products that Northwind sells [32] . Each record includes the product's name, the supplier who provides it, the category it belongs to, quantity per unit (package quantity or size), unit price, units in stock, units on order, reorder level, and whether the product is discontinued [33] .
- **Columns:**
- **ProductID** – *INT*, primary key identifier for the product (unique product number) [34] .
- **ProductName** – *VARCHAR(40)*, name of the product.
- **SupplierID** – *INT*, foreign key to the supplier of this product (links to Suppliers.SupplierID) [33] .

- **CategoryID** – *INT*, foreign key to the category this product belongs to (links to Categories.CategoryID) [33] .
- **QuantityPerUnit** – *VARCHAR(20)*, description of the packaging or quantity (e.g. “24 - 12 oz bottles”).
- **UnitPrice** – *MONEY/CURRENCY*, price per unit of the product [35] .
- **UnitsInStock** – *INT*, number of units currently in stock.
- **UnitsOnOrder** – *INT*, number of units currently on order (ordered from supplier but not yet received).
- **ReorderLevel** – *INT*, the stock level at which new orders should be placed (threshold to reorder).
- **Discontinued** – *BOOLEAN* (Yes/No), indicates if the product is discontinued (true if no longer available for sale) [36] [37] .
- **Primary Key:** `ProductID`.
- **Foreign Keys:**
- `SupplierID` → **Suppliers**(`SupplierID`) – identifies the supplier who provides this product [18] .
- `CategoryID` → **Categories**(`CategoryID`) – identifies the category under which this product is listed [18] .
- **Relationships:** Each **Product** belongs to one **Supplier** and one **Category** (many-to-one relationships: many products can be supplied by the same supplier, and many products can be in the same category) [18] . The Products table is on the “many” side of those relationships (Suppliers and Categories are the “one” side). Additionally, a product can appear in many **Order_Details** records (each representing the product being sold in a particular order) [18] . Thus, Products and Orders are indirectly related as a many-to-many (one product can be part of many orders, and one order can include many products) through the Order_Details table.

Categories

- **Description:** The Categories table defines product categories (groups of products), such as “Beverages”, “Condiments”, etc. It stores the category names and descriptions for classifying products [38] . Optionally, it may include a picture representing the category (for example, an image for the category of products) [39] .
- **Columns:**
- **CategoryID** – *INT*, primary key for the category (unique category identifier) [40] .
- **CategoryName** – *VARCHAR(15)*, name of the category (e.g. “Beverages”) [39] .
- **Description** – *TEXT*, description of the category (typically a longer text explaining the kinds of products in the category).
- **Picture** – *BLOB/BINARY*, an image for the category (stored binary picture of the category illustration) [39] .
- **Primary Key:** `CategoryID`.
- **Foreign Keys:** *(None)* – Categories has no foreign keys; it’s an independent lookup table.
- **Relationships:** One **Category** can have many **Products** (1-to-many relationship). The CategoryID is referenced by the Products table to indicate each product’s category [41] . Categories is the parent table in this relationship, with Products on the child side. No other tables reference Categories.

Suppliers

- **Description:** The Suppliers table stores information about product suppliers – the companies that supply (sell) products to Northwind [42] [43] . For each supplier (vendor), it includes the company name, contact name/title, address, country, phone/fax, and a homepage URL [43] [44] .

- **Columns:**
- **SupplierID** – *INT*, primary key for the supplier (unique supplier identifier) [45] .
- **CompanyName** – *VARCHAR(40)*, the name of the supplier company.
- **ContactName** – *VARCHAR(30)*, name of the contact person at the supplier.
- **ContactTitle** – *VARCHAR(30)*, title of the contact person.
- **Address** – *VARCHAR(60)*, street address of the supplier.
- **City** – *VARCHAR(15)*, city.
- **Region** – *VARCHAR(15)*, region or state.
- **PostalCode** – *VARCHAR(10)*, postal code.
- **Country** – *VARCHAR(15)*, country.
- **Phone** – *VARCHAR(24)*, phone number.
- **Fax** – *VARCHAR(24)*, fax number.
- **HomePage** – *TEXT*, webpage or website URL of the supplier (if any) [45] .
- **Primary Key:** `SupplierID`.
- **Foreign Keys:** *(None)* – the Suppliers table doesn't reference other tables.
- **Relationships:** One **Supplier** can supply many **Products** (1-to-many relationship, as many products can come from the same supplier) [46] . The Products table stores a SupplierID for each product to link it to this table. No other direct relationships (no tables reference suppliers besides Products).

Shippers

- **Description:** The Shippers table (sometimes called Carriers) contains information about shipping companies used to deliver orders [42] . It typically has just a few entries (e.g. "Speedy Express", "United Package", etc.), including the company name and phone number for each shipper [47] .
- **Columns:**
- **ShipperID** – *INT*, primary key for the shipper (unique identifier) [48] .
- **CompanyName** – *VARCHAR(40)*, name of the shipping company.
- **Phone** – *VARCHAR(24)*, contact phone number for the shipper [49] .
- **Primary Key:** `ShipperID`.
- **Foreign Keys:** *(None)* – Shippers is a standalone list of carriers.
- **Relationships:** One **Shipper** can handle many **Orders** (1-to-many relationship). The Orders table's ShipVia field is a foreign key to Shippers, indicating which shipper was used for a given order [8] . No other tables depend on Shippers.

Regions

- **Description:** The Regions table stores a list of sales regions. Each region is an area designation (for example, "Eastern", "Western", "Northern", "Southern") used to categorize territories and possibly for organizing sales reports [50] . This table provides the valid list of region identifiers and their descriptions.
- **Columns:**
- **RegionID** – *INT*, primary key for the region (unique region identifier).
- **RegionDescription** – *CHAR(50)*, name or description of the region (fixed-length, e.g. "Eastern" padded to length) [51] [52] .
- **Primary Key:** `RegionID`.
- **Foreign Keys:** *(None)* – Region is an independent lookup table (no outgoing foreign keys).

- **Relationships:** One **Region** can encompass many **Territories** (1-to-many). The Territories table includes a RegionID foreign key for each territory, linking it to this Region table [53] . Regions is the parent in this relationship (each territory is assigned to one region). No other tables directly reference Region.

Territories

- **Description:** The Territories table contains sales territories, which are geographic areas or districts assigned to employees. Each territory has an ID code and a territory description (e.g. a region of a country or city area) [50] . Territories are each associated with one of the defined Regions.
- **Columns:**
- **TerritoryID** – *VARCHAR(20)*, primary key code of the territory (unique territory identifier, could be an alphanumeric code) [54] .
- **TerritoryDescription** – *CHAR(50)*, description of the territory (e.g. territory name or description, stored as a fixed-length text) [54] .
- **RegionID** – *INT*, foreign key linking to the Region table (indicates which Region this territory falls under) [54] .
- **Primary Key:** `TerritoryID`.
- **Foreign Keys:**
- `RegionID` → **Regions**(`RegionID`) – associates the territory with its region [53] .
- **Relationships:** Each **Territory** belongs to exactly one **Region** (many territories to one region) [53] . A **Territory** can also be associated with multiple **Employees** (and vice versa) through the **EmployeeTerritories** join table. In the join table, each territory can appear in many records (one per employee assignment), meaning a territory can have many employees assigned (many-to-many between Territories and Employees via the linking table).

EmployeeTerritories

- **Description:** The EmployeeTerritories table links **Employees** to **Territories**, assigning which territories each employee is responsible for. It is a junction table implementing a many-to-many relationship between employees and territories [31] . Each record indicates that a given employee works with a given territory (i.e. an assignment of an employee to a territory).
- **Columns:**
- **EmployeeID** – *INT*, foreign key to an employee (links to Employees.EmployeeID) [55] .
- **TerritoryID** – *VARCHAR(20)*, foreign key to a territory (links to Territories.TerritoryID) [55] .
- **Primary Key:** Composite of `EmployeeID` + `TerritoryID` (each combination of employee and territory is unique) [55] .
- **Foreign Keys:**
- `EmployeeID` → **Employees**(`EmployeeID`) – references the employee in the assignment [55] .
- `TerritoryID` → **Territories**(`TerritoryID`) – references the territory in the assignment [55] .
- **Relationships: EmployeeTerritories** is a pure linking table. By itself it has a many-to-one relation from EmployeeTerritories to Employees (many assignments for one employee) and many-to-one from EmployeeTerritories to Territories (many assignments for one territory). These two links together mean **Employees** and **Territories** have a many-to-many relationship through this table [31] . One employee can be linked to multiple territories (1-to-many from Employees to EmployeeTerritories), and one territory can be linked to multiple employees (1-to-many from

Territories to EmployeeTerritories). This allows, for example, assigning multiple sales regions to a single employee and also multiple employees per territory.

CustomerDemographics

- **Description:** The CustomerDemographics table holds categories or demographic segments of customers (e.g. market segments). Each record represents a customer demographic type (for example, “Retail”, “Wholesale”, “Geographic Region”, or other classifications that might describe a group of customers). In the Northwind sample data this table is defined but often remains empty (it’s provided for demonstration of a many-to-many classification) [56] [57] .
- **Columns:**
- **CustomerTypeID** – *CHAR(10)*, primary key code of the customer demographic category (identifier for the demographic type) [58] .
- **CustomerDesc** – *TEXT*, description of the customer demographic (explanation of the category).
- **Primary Key:** `CustomerTypeID`.
- **Foreign Keys:** *(None)* – this table is not a child of any other table.
- **Relationships:** Many **Customers** can be associated with a given **CustomerDemographics** entry, and each **CustomerDemographics** category can apply to many customers — this many-to-many relationship is implemented via the CustomerCustomerDemo table (linking table). The **CustomerDemographics** table is on the “one” side of a one-to-many relationship with **CustomerCustomerDemo**, and through that link it relates to Customers.

CustomerCustomerDemo

- **Description:** The CustomerCustomerDemo table is a junction table linking **Customers** with **CustomerDemographics**. It assigns customers to demographic categories, allowing each customer to have zero or more demographic classifications and each customer demographic category to include any number of customers (many-to-many relationship between Customers and CustomerDemographics) [59] . Each record in this table indicates that a particular customer belongs to a particular customer demographic group.
- **Columns:**
- **CustomerID** – *CHAR(5)*, foreign key to a customer (links to Customers.CustomerID) [59] .
- **CustomerTypeID** – *CHAR(10)*, foreign key to a customer demographic type (links to CustomerDemographics.CustomerTypeID) [59] .
- **Primary Key:** Composite of `CustomerID` + `CustomerTypeID` (each combination of customer and demographic category is unique) [59] .
- **Foreign Keys:**
- `CustomerID` → **Customers**(`CustomerID`) – references the customer in the association [59] .
- `CustomerTypeID` → **CustomerDemographics**(`CustomerTypeID`) – references the demographic category in the association [59] .
- **Relationships: CustomerCustomerDemo** functions purely as a linking table. It has a many-to-one relationship to Customers (one customer can have multiple entries in this table, one per demographic type) and likewise many-to-one to CustomerDemographics (one demographic type can link to multiple customers through multiple records). Together, this creates a many-to-many relationship between **Customers** and **CustomerDemographics**. For example, a single customer can belong to several demographic categories (multiple rows in CustomerCustomerDemo with the same

CustomerID but different CustomerTypeID), and a single demographic category can include many different customers.

US_States (optional)

- **Description:** The US_States table, if present, provides a reference list of U.S. states and territories. This is a supplemental lookup table that includes state names, abbreviations, and a regional grouping. It is not central to the main schema and is not referenced by other tables (often used for demo or form lookups in certain versions of the database).
- **Columns:**
- **StateID** – *INT*, primary key for the state (unique ID, e.g. 1–50 for states, plus perhaps territories) [60] .
- **StateName** – *VARCHAR(100)*, full name of the state (e.g. "Alabama", "Alaska").
- **StateAbbr** – *VARCHAR(2)*, two-letter state abbreviation (e.g. "AL", "AK").
- **StateRegion** – *VARCHAR(50)*, the region of the country the state is in (e.g. "Northwest", "Southeast").
- **Primary Key:** `StateID`.
- **Foreign Keys:** *(None)* – US_States is a standalone reference table.
- **Relationships:** This table is not linked via foreign keys to the rest of the Northwind schema. It may be used for address data consistency or not used at all, depending on the database implementation. (In many Northwind implementations, the *Region* field in addresses is free text and not linked to US_States; thus US_States is purely informational.)

Summary of Table Relationships

The following table summarizes the relationships between the key tables in the Northwind database schema, indicating foreign key links and cardinality:

Tables (Primary Key → Foreign Key)	Relationship	Details
Customers (CustomerID) → **Orders** (CustomerID)	One-to-Many	Each customer can have many orders [8] . (An order references one customer.)
Employees (EmployeeID) → **Orders** (EmployeeID)	One-to-Many	Each employee can process many orders [8] . (An order is handled by one employee.)
Shippers (ShipperID) → **Orders** (ShipVia)	One-to-Many	Each shipper can ship many orders [8] . (An order uses one shipper.)
Orders (OrderID) → **Order_Details** (OrderID)	One-to-Many	Each order can have multiple order detail line items [8] .
Products (ProductID) → **Order_Details** (ProductID)	One-to-Many	Each product can appear in many order details (many orders) [18] .
Categories (CategoryID) → **Products** (CategoryID)	One-to-Many	Each category includes many products [41] .
Suppliers (SupplierID) → **Products** (SupplierID)	One-to-Many	Each supplier provides many products [46] .

Tables (Primary Key → Foreign Key)	Relationship	Details
Region (RegionID) → **Territories** (RegionID)	One-to-Many	Each region encompasses many territories [53] .
Employees (EmployeeID) → **Employees** (ReportsTo)	One-to-Many (self)	Each manager (employee) can have many direct-report employees [26] . (Employee hierarchy via ReportsTo.)
Employees ↔ Territories (via **EmployeeTerritories**)	Many-to-Many	An employee can be assigned to multiple territories and each territory can have multiple employees [31] . (Link table EmployeeTerritories with EmployeeID & TerritoryID.)
Customers ↔ CustomerDemographics (via **CustomerCustomerDemo**)	Many-to-Many	A customer can belong to multiple demographic categories, and each category can include multiple customers [59] . (Link table CustomerCustomerDemo.)
Orders ↔ Products (via **Order_Details**)	Many-to-Many	An order can contain multiple products and a product can appear in multiple orders [17] . (Link table Order_Details.)

[1] [31] [50] SQL Sample Database
https://www.zentut.com/sql-tutorial/sql-sample-database/

[2] [3] [4] [5] [6] [7] [8] [9] [10] [11] [12] [13] [14] [15] [16] [17] [18] [19] [20] [21] [22] [23] [24] [25] [26] [27] [28] [29] [32] [33] [34] [35] [36] [37] [38] [39] [40] [41] [42] [43] [44] [45] [46] [47] [48] [49] Northwind Database
https://techwriter.me/downloads/samples/Database/Access2003Northwind.pdf

[30] [51] [52] [53] [54] [55] [58] [59] [60] raw.githubusercontent.com
https://raw.githubusercontent.com/yugabyte/yugabyte-db/master/sample/northwind_ddl.sql

[56] [57] Is there data in northwind's customerDemographics table? - Microsoft Q&A
https://learn.microsoft.com/en-us/answers/questions/252925/is-there-data-in-northwinds-customerdemographics-t