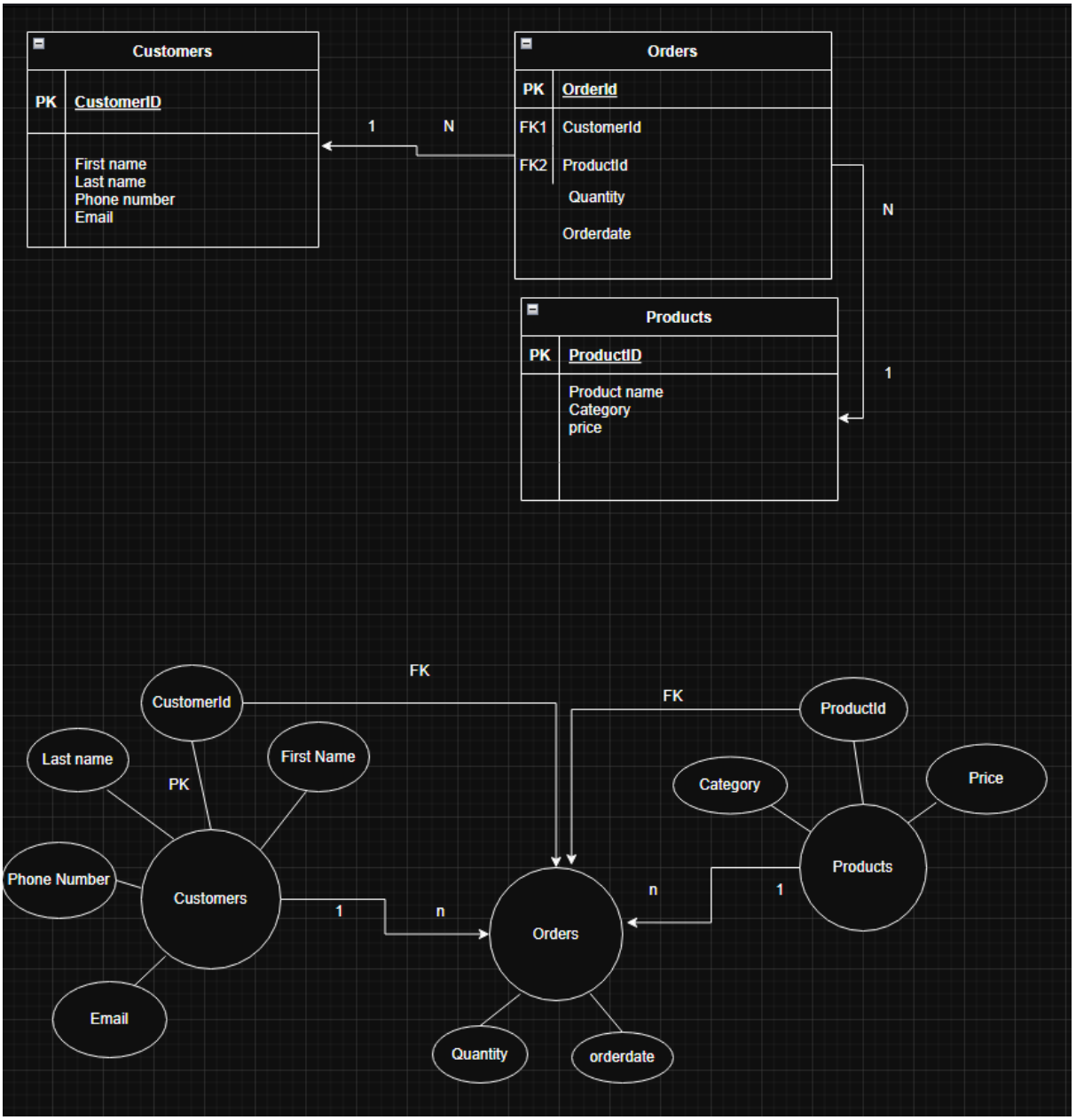

ER diagram

Som visar relationen till alla Tabellerna

Säkerhet

Det finns många anledningar till att säkerhet är viktigt! Om någon får tillgång till din datbas så kan dem deleta alla ordrar och folk, sno din data och mycket mer.
I sådana fall kan du faktiskt säkra dig genoma att ha Hashade lösenord, krypterad data some är känslig och olika nivåer på dina users Som Admin tillgång som kan skriva in och lägga till data och vanlig användare som bara kan läsa data men inte skriva in.

Authentication
Handlar mest om vem du är och vad du får för behörighet beroende på vem du är, admin, user or nonUser etc Du får exempelvis inte logga in i en databas/connecta om du inte skall ha tillgång till databasen

#Authorization
Authorization handlar om att kontrollera vad användaren får göra. Skriva i databasen? Bara läsa data? eller ingen tillgång.

SQL VS LINQ

Gjorde svaren i en md fil först så det ser konstigt ut för att jag bara kopierade in i pdf dokumentet

SQL
```
SELECT *
FROM Customers
WHERE FirstName = 'Elvis';
```

v v v

LINQ
```
var ElvisTheMostAwesomeCustomer = db.Customers.Where(c => c.FirstName == "Elvis").ToList();
```
---
* (Select * och Customers) SQL returnerar alla kolumner; i LINQ returneras hela entity-objektet med alla properties.

* (from Customers och db.Customers) SQL-tabellen motsvaras av DbSet i EF och varje rad i tabellen blir ett objekt i LINQ.

* (Where firstname = 'Elvis' och .Where(c => c.FirstName == "Elvis")) SQL-filter översätts till en lambda-funktion som kontrollerar property FirstName på varje objekt.

* (Resultat och Lista med Customers) SQL-resultatet är en tabell med rader och LINQ-resultatet är en lista med objekt som representerar samma rader.

SQL
SELECT ProductName, Category
FROM Products
WHERE ProductName LIKE '%Latte%';

v v v

LINQ
var latteProducts = db.Products.Where(p =>
p.ProductName.Contains("Latte")).Select(p => new { p.ProductName, p.Category
}).ToList();
---
* (SELECT ProductName och Category.Select(p => new { p.ProductName, p.Category })) I SQL väljer du kolumner; i LINQ skapar du ett anonymt objekt med de properties du vill ha.

* (FROM Products och db.Products) Tabell → DbSet → varje rad blir ett objekt.

* (WHERE ProductName LIKE '%Latte%' och .Where(p => p.ProductName.Contains("Latte"))) SQL's LIKE med % → LINQ's Contains för att matcha delsträngar.

* (Resultat och Lista av anonym-objekt med två properties) SQL returnerar en tabell med 2 kolumner och LINQ returnerar lista med objekt med 2 egenskaper.

SQL
SELECT ProductID, SUM(Quantity) AS TotalQuantitySold
FROM Orders
GROUP BY ProductID;

v v v

LINQ
var totalSoldPerProduct = db.Orders.GroupBy(o => o.ProductID).Select(g => new {ProductID = g.Key,TotalQuantitySold = g.Sum(o => o.Quantity)}).ToList();
---
* (GROUP BY ProductID.GroupBy(o => o.ProductID)) SQL grupperar raderna per produkt och LINQ skapar grupper av objekt (IGrouping) där g.Key = ProductID.

* (SUM(Quantity)g.Sum(o => o.Quantity)) Aggregat i SQL → metod på gruppen i LINQ. Varje grupp summerar Quantity-property.

* (SELECT ProductID, SUM(Quantity) och .Select(g => new { ProductID = g.Key, TotalQuantitySold = g.Sum(...) })) SQL returnerar kolumn + aggregat; LINQ skapar ett nytt objekt med samma information.
* (Resultat och Lista av anonym-objekt med ProductID + TotalQuantitySold) SQL-resultatet är tabell med två kolumner; LINQ-resultatet är lista med objekt med två properties.

Jag har börjat att gilla hur man skriver i databaser och LINQ har typ blivit mer klumpig men det är vääl mest för hur lättläst det är med databas queries.

Reflektioner

Överlag har jag börjat gilla att handskas med databaser. Inte jätte krångligt. Syntaxen kommer jag inte alltid ihåg men annars är det inte rocket science. Jag tyckte det var roligt. Sen att jag fortfarande suger på diagram är en annan sak, lite tråkigt MEN viktigt. Upplevt hur man behöver agera med Deletes då man måste gå från foreign keys först och se på alla relationer innan man kan ta bort saker. Sen har det mesta varit smooth sailing. Fick lite problem när jag tog bort en databas och den fortfarande ville koppla den när jag loggade in i sql men löste det snabbt

Kan alltid bli bättre på att ta in info och sätta mig ner under längre tid.

ser fram emot alla andra kurser med.