Module: Deciphering Big Data

Development Team Project: Project Report

Creating the Database Design

On the 10th of November 2024, our team held a meeting where we decided to apply this assignment to the retail sector. We discussed potential variables that would be essential for the database design, such as customer details, product information, orders, and store location. Together, we agreed to individually work on creating a rough outline of what the design should include and how it should be structured. We plan to reconvene on the 14th of November to share our initial ideas, discuss refinements, and collaboratively move forward with the next steps of the project.

Retail databases are rich with data and require well-structured relationships between data entities to support a range of functions, from inventory management to customer interactions. Below is a breakdown of an idea and some key entities, attributes, relationships, and flow considerations:

The Retail System

Assuming our client is a retail business (eg. clothing shop) that sells products both in-store and online, the client would want a database to manage the following:

1. **Inventory and stock**: Products in various categories with suppliers.
2. **Customer information**: Personal information, purchase history, and loyalty points.
3. **Sales and transactions**: Detailed transaction records, including items purchased, prices, and payment details.
4. **Employee information**: Employee roles, schedules, and sales performance.

Core Entities, Attributes, and Relationships

(See video: https://www.youtube.com/watch?v=wMgirP7z4k8

Crow's Foot Notation: https://vertabelo.com/blog/crow-s-foot-notation/)

1. **Customers**
    - *Attributes*: CustomerID (PK), Name, Email, Phone, Address, DateJoined, LoyaltyPoints.
    - *Relationships*: Each customer can place multiple orders (one-to-many relationship with Orders).
    - *Flow*: Data flows from **Customer** to **Order** when a purchase is made, and **Order** information flows back to update the customer's LoyaltyPoints.

2. **Products**
    - *Attributes*: ProductID (PK), ProductName, Category, Price, StockQuantity, SupplierID (FK).
    - *Relationships*: Each product belongs to a **Category** (many-to-one relationship) and has a **Supplier** (many-to-one relationship).
    - *Flow*: Products are linked to **Orders** whenever a sale is made (many-to-many through an intermediate entity, OrderItems).

3. **Orders**
    - *Attributes*: OrderID (PK), OrderDate, TotalAmount, CustomerID (FK), StoreID (FK).
    - *Relationships*: Each **Order** is linked to a single **Customer** (one-to-one) and a **Store** where the order was made or shipped from (many-to-one).
    - *Flow*: **Orders** are placed by **Customers** and contain **OrderItems** (many-to-many relationship with Products).

4. **OrderItems**
    - *Attributes*: OrderItemID (PK), OrderID (FK), ProductID (FK), Quantity, PriceAtPurchase.
    - *Relationships*: This is an intermediary table connecting **Orders** and **Products** in a many-to-many relationship, storing the quantity and price of each product ordered.
    - *Flow*: Data flows between **Orders** and **Products** to track purchased quantities, with **Inventory** adjusting based on sales activity.

5. **Suppliers**
    - *Attributes*: SupplierID (PK), SupplierName, ContactNumber, Email, Address.
    - *Relationships*: Each **Supplier** can supply multiple **Products** (one-to-many relationship).
    - *Flow*: **Products** are linked to **Suppliers**, and reorder requests can be generated if stock levels are low.

6. **Employees**
    - *Attributes*: EmployeeID (PK), Name, Position, StoreID (FK), HireDate, PerformanceScore.

- *Relationships*: Employees are associated with a **Store** (many-to-one) and may manage **Orders** if they process sales.
- *Flow*: Employees contribute to store sales and their performance could be tracked based on completed **Orders**.

7. **Stores**
    - *Attributes*: StoreID (PK), Location, ManagerID (FK), PhoneNumber.
    - *Relationships*: Each **Store** can have multiple **Employees** (one-to-many) and handle multiple **Orders**.
    - *Flow*: Store information is essential for local inventory, order fulfillment, and tracking sales across locations.

Database Management System

For this retail database, we can propose a relational database management system like MySQL, PostgreSQL, or SQL Server as they provide robust support for relational data models, foreign key constraints, indexing for faster queries, and good scalability.

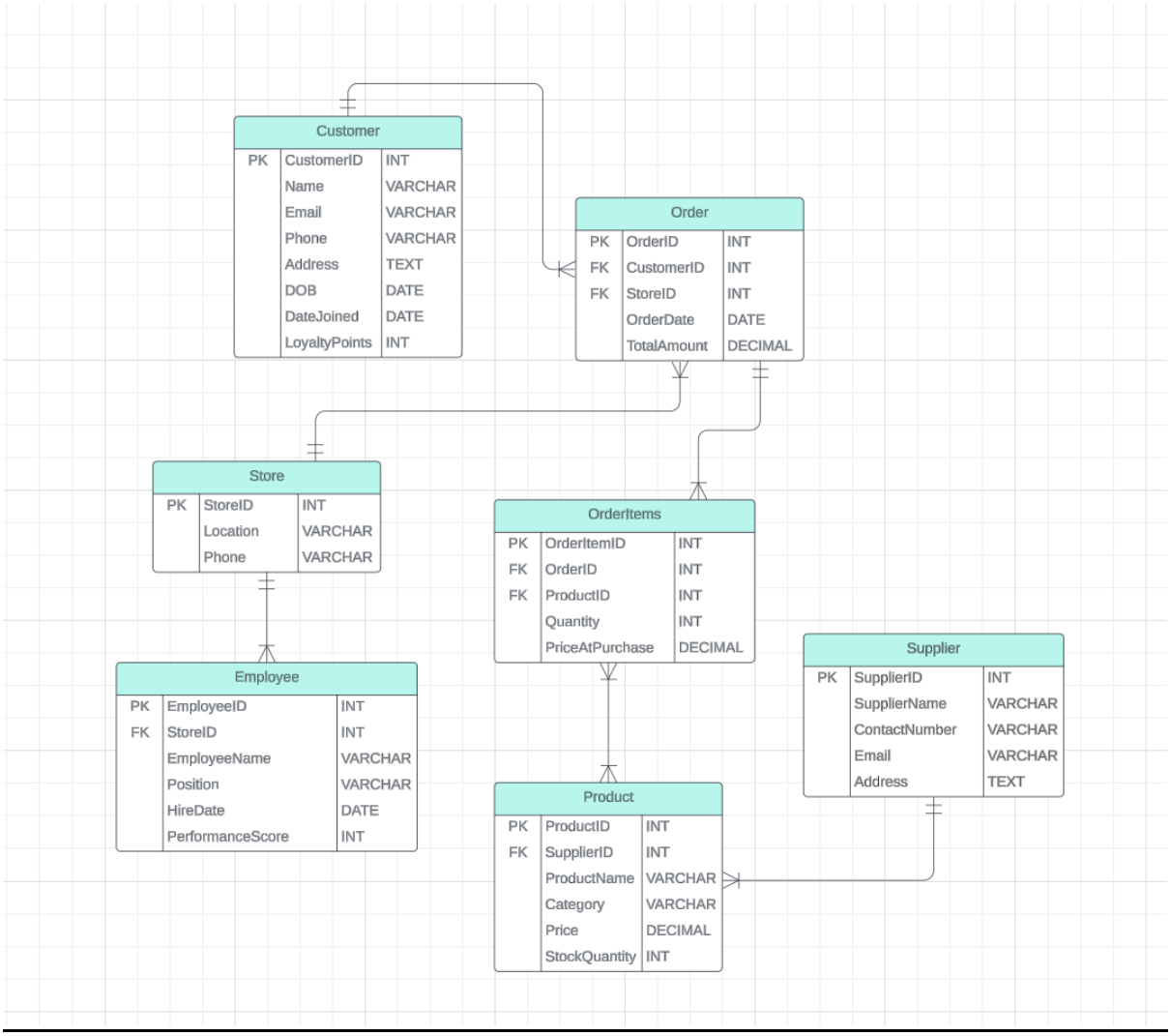
Some considerations:

- **Storage**: Estimate based on product range, transaction volume, and historical data retention needs.
- **User Access**: Define user roles for administrative access, customer service, and analytics to limit access by role.
- **Data Retrieval**: Ensure that your DBMS supports efficient querying, especially for analytical queries on sales trends, inventory management, and customer segmentation.

Data Management Pipeline

1. **Data Capturing**: Data sources may include online orders, in-store POS systems, and customer interactions.
2. **Data Cleaning**:
    - Deduplication: Removing duplicate customer records or products.
    - Normalization: Ensuring consistent formatting for text fields (e.g., phone numbers, addresses).
    - Validation: Checking for mandatory fields (e.g., OrderDate, TotalAmount) to avoid incomplete records.
3. **ETL (Extract, Transform, Load)**: If data comes from various sources, set up a pipeline to consolidate it in the database.

Entity Relationship Diagrams using Crow's Foot Notation



https://lucid.app/lucidchart/4a0ef786-8cb3-4629-89c7-a43ba85891c4/edit?viewport_loc=-1820%2C-492%2C1361%2C1509%2C0_0&invitationId=inv_0834ce91-a897-4dac-a684-db8a8b569efd