Data Pipeline process

Data source:

* Point of sale (asked for email/phone number)
* Customer order/ order items
* Customer creating accounts

Cleaning:

* Recognising and preventing duplicates (if customer creates more than one account)
* Creating a flag for anomalies in PriceAtPurchase, Quantity
* Check that values match the data type for that variable in question

Storage:

* Where is the data housed? (eg. Snowflake, Microsoft servers, SQL server)

User Access

* Employee would have all access
* Customer would have access to their own details, Order, OrderItems, Product

Data Protection (GDPR)

* EmployeeID is necessary to trace back any account activity internally
* After a certain threshold (eg. 3 years of no use), delete the profile (to be inline with GDPR)

Introduction: Project overview, client selection/loyalty scheme, what we are trying to achieve, relational database

Logical Database Design: Entity-relationship diagram (ERD), describe attributes and data types and relationships, primary and foreign keys

Database Model Proposal: Proposing a database management system that fits the store’s needs, discussing how the selected DBMS supports the data structure. Broadly used, ease of user utility, easily integrated with other databases

Data Management Pipeline: Data capture process/source, cleaning, storage, access and manipulation, deletion based on inactivity, anomalies and user input

Conclusion

For our project in Deciphering Big Data, our development team decided to choose a retail store as our client. The retail store wants to migrate their local data to the cloud and implement a new loyalty program. A retail store is complex when it comes to the data they house. For example, you have customers and their demographic data, store locations, products, suppliers, sales records, order numbers, and more. Our plan is to create a cloud database for the retail company in Snowflake that would solve their data needs. Most stores nowadays, when you walk in, have some type of rewards or points program to entice customers to purchase and receive rewards. Our “client” wants to implement that program as well so he can reward customers with discounts and savings as they shop with him. We are going to create a relational database in MySQL Workbench and also implement their loyalty program in the backend as well. This will house all the data in one central location and creating links between the data will be easier and more efficient.

The source of our data for the new loyalty program would be Employee Input. Every Loyalty member will have to have their Name, Email, Phone\_Number, Address, Birthday, inputted into the front-end system by an employee at the store when purchasing a product, or they would be able to sign up online on the retailers website. The information collected in person or online would then be relayed back via an API to the MySQL database and stored as a customer who is signed up for the loyalty program. As the customer purchases products, they will earn rewards which will be generated after certain milestones or events such as birthdays. Once the rewards are generated in the table, the customer will be informed via an email.

**Database Relationships:**

The database for the company will include multiple tables, each linked to others through one-to-one or one-to-many relationships. The overarching database is named Retail\_Store\_Database, with the following key tables: Customer, Order, Store, OrderItems, Employee, Product, and Supplier.

* **Customer Table**:  
  The Customer table houses all customer information and connects to the Order table through a one-to-many relationship, using CustomerID as the primary key in Customer and foreign key in Order.
* **Order Table**:  
  The Order table tracks individual customer purchases and is connected to the Store table via a many-to-one relationship (one store processes many orders), and to the OrderItems table via a one-to-many relationship (one order contains multiple order items). The Order table also includes information about the total amount spent by the customer.
* **OrderItems Table**:  
  The OrderItems table tracks details of items purchased in each order. It connects to the Order table via OrderID and to the Product table via ProductID. This one-to-many relationship allows the store to identify which products were sold and in what quantities.
* **Product Table**:  
  The Product table lists all products available for sale, including details such as product name, category, price, and stock quantity. It connects to the Supplier table via a many-to-one relationship using SupplierID, enabling the store to track inventory back to its suppliers.
* **Store Table**:  
  The Store table tracks physical store locations and is linked to the Employee table via a one-to-many relationship. Each store can have multiple employees assigned to it.
* **Supplier Table**:  
  The Supplier table identifies vendors providing products to the store, allowing efficient inventory management and supplier traceability.

The chosen data types and the reasoning for selecting the data type for each attribute are in the tables in the appendix.

~~TO DELETE BELOW~~

~~The database for the company will be set up in a way where there will be multiple tables. Each table will be linked to another by a one-to-one or one-to-many relationship. The overarching database will be called the Retail\_Store\_Database. The tables, in no particular order, would be Customer, Order, Store, Order\_Items, Employee, Product and Supplier. The customer table would house all the customer information which would also be able to connect to Order table, the relationship between the two would be one to many because one customer can have many orders and the primary key to connect the tables would be through the customerID which is present in both tables.~~

~~The Orders are then connecting to the Store and OrderItems table. It connects to Store via a many-to-one relationship so the order can be tracked to which store it came from, and then store connects to employee to track which employees work at which store also via a one to many relationship on the primary key StoreID because one store will have many employees.~~

~~The second leg then connects Orders to OrderItems via a one-to-many relation because one person can have many orders and we can find out which products were purchased during the orders they placed. OrderItems table is a table for the shop to know which products are sold, It would also help define how much of an item one person had purchased and help for inventory management. The OrderItems table contains the ProductID which is then connected to the Products table through a many-to-many relationship. This Product table will connect to the supplier table as well and the benefit of that is the Products will be able to be traced back all the way to the supplier through the database and its relations, this will be many products to many one supplier~~

Appendix:

A diagram of a computer

Description automatically generated

Appendix Continued:

Code to Create tables in SQL:

A screenshot of a computer program

Description automatically generatedA screenshot of a computer program

Description automatically generated

Appendix Continued:

Tables Explaining Data Types and Reasoning:

**Customer Table**

| **Attribute** | **Data Type** | **Reason** |
| --- | --- | --- |
| CustomerID | INT | A unique integer identifier for each customer, efficient for indexing and queries. |
| Name | VARCHAR | Flexible text format to store customer names of varying lengths (e.g., "John Doe"). |
| Email | VARCHAR | Emails are alphanumeric and variable-length; VARCHAR ensures storage efficiency. |
| Phone | VARCHAR | Phone numbers often include formatting characters (e.g., +1, -), making VARCHAR suitable. |
| Address | TEXT | Addresses can be lengthy and vary significantly, so TEXT provides sufficient flexibility. |
| DOB | DATE | A specific date format is needed to store the customer's birth date. |
| DateJoined | DATE | Captures the date the customer joined the loyalty program for tracking membership history. |
| LoyaltyPoints | INT | An integer value to store points earned by customers in the loyalty program. |

**Order Table**

| **Attribute** | **Data Type** | **Reason** |
| --- | --- | --- |
| OrderID | INT | Unique identifier for each order, optimized for indexing and relational links. |
| CustomerID | INT | Foreign key linking to the Customer table. Matches the data type of CustomerID. |
| StoreID | INT | Foreign key linking to the Store table. Matches the data type of StoreID. |
| OrderDate | DATE | Tracks the specific date the order was placed. |
| TotalAmount | DECIMAL | Stores the total cost of the order, including decimals for accuracy in monetary values. |

**Store Table**

| **Attribute** | **Data Type** | **Reason** |
| --- | --- | --- |
| StoreID | INT | Unique identifier for each store, suitable for indexing. |
| Location | VARCHAR | Text format to store store locations (e.g., city names or addresses). |
| Phone | VARCHAR | Allows storage of phone numbers with varying formats. |

**OrderItems Table**

| **Attribute** | **Data Type** | **Reason** |
| --- | --- | --- |
| OrderItemID | INT | Unique identifier for each order item, optimized for indexing. |
| OrderID | INT | Foreign key linking to the Order table. Matches the data type of OrderID. |
| ProductID | INT | Foreign key linking to the Product table. Matches the data type of ProductID. |
| Quantity | INT | Integer to store the number of items purchased. |
| PriceAtPurchase | DECIMAL | Captures the price of the product at the time of purchase, with decimal precision. |

**Employee Table**

| **Attribute** | **Data Type** | **Reason** |
| --- | --- | --- |
| EmployeeID | INT | Unique identifier for each employee, efficient for indexing and queries. |
| StoreID | INT | Foreign key linking to the Store table. Matches the data type of StoreID. |
| EmployeeName | VARCHAR | Stores employee names of varying lengths. |
| Position | VARCHAR | Tracks employee job titles (e.g., "Manager"). |
| HireDate | DATE | Tracks when the employee was hired. |
| PerformanceScore | INT | Integer used for performance tracking or evaluations. |

**Product Table**

| **Attribute** | **Data Type** | **Reason** |
| --- | --- | --- |
| ProductID | INT | Unique identifier for each product, optimized for indexing. |
| SupplierID | INT | Foreign key linking to the Supplier table. Matches the data type of SupplierID. |
| ProductName | VARCHAR | Stores product names, allowing flexibility for different lengths. |
| Category | VARCHAR | Tracks product categories (e.g., "Electronics"). |
| Price | DECIMAL | Stores product prices with decimal precision for accuracy. |
| StockQuantity | INT | Tracks the quantity of the product in stock. |

**Supplier Table**

| **Attribute** | **Data Type** | **Reason** |
| --- | --- | --- |
| SupplierID | INT | Unique identifier for each supplier. |
| SupplierName | VARCHAR | Stores supplier names with varying lengths. |
| ContactNumber | VARCHAR | Allows flexibility for phone numbers with varying formats. |
| Email | VARCHAR | Suitable for storing email addresses. |
| Address | TEXT | Provides flexibility for lengthy addresses. |