Retailer Sales Company

Sudheer Daggubati

GIT HUB URL: https://github.com/sudheer34/Mycode.git

November 13’ 2022

A department store's general operation and function depend heavily on the administration and management of the store. There will be many products and categories, especially in the case of larger stores. Therefore, efficient management and good resource management are essential for success and smooth operation. So, to start our company that makes toys for kids and sells them on our website and mobile app, we must have a plan to develop and start our company in the city. Gradually we have plan to open more and more branches of it to enhance the company size and generate more revenue, we are planning to open our company into Mexico state. The initial size of our company is about 7-10 employees and next we will increase and enhance our company with time. We will be selling toy products of named cartoons or characters to make kids indulge into them. Like ben10, Pokémon, Avengers, Superman, Batman, Animal Kingdom, Doraemon etc. As far as annual sale is concerned, we are planning to generate 1 million dollars.

Target is the one retailer company which we found like our plan. They have categorized the popular and exciting face characters of the kids to sell their clothes. We are doing the same, the different thing is that we are making their toys to sell. Target has generated 106$ billion in 2022 and has 1938 stores around the US. They employee 400,000+ members in their company. (<https://corporate.target.com/about>)

For our system we are going to using the 3-tier architecture like below. This is the generic architecture from which we have our client interaction with presentation layer (UI layer) which will send request to backend business logic which follows the data retrieval from the data component layer and make it in some presentable state to present back it on presentation layer.

Diagram

Description automatically generated

For our business we need to store inventory, prices, quantities, product names, etc. So, for this purpose we may need following data types:

varchar, int, number, bool and varchar

To translate data from one system to another system we can use Celigo, Cleo or Cyclr etc.

Customers of these are: Amazon, Amazon web services, Braintree, asana, 3dcart, ADP etc. (https://solutionsreview.com/data-integration/the-best-data-transformation-tools-and-software/)

The common data formats that computer systems use to transfer product data between system are CSV and JSON and XML.

Entities:

**Customer:** Describes the information of customer. Customer\_ID, Name, Email, Address.

**Product:** Describes the information of products. Product\_ID, Name, Price.

**Stock:** Describes the information of stock in hand. Stock\_ID, Product\_ID, Available\_Quantity.

**Sales:** Describes the information of sales record. Sales\_ID, Product\_ID, Customer\_ID, Quantity, Total\_Amount.

**Payment:** Describes the information of payments and bills. Payment\_ID, Sales\_ID, Payment\_Method, Date.

Tables:

|  |  |
| --- | --- |
| **Customer** | |
| Customer\_ID | Integer |
| Name | Varchar(255) |
| Email | Varchar(255) |
| Address | Varchar(255) |

|  |  |
| --- | --- |
| **Product** | |
| Product\_ID | Integer |
| Name | Varchar(255) |
| Price | BIGINT |

|  |  |
| --- | --- |
| **Sales** | |
| Sales\_ID | Integer |
| Product\_ID | Integer |
| Customer\_ID | Integer |
| Quantity | BIGINT |
| Total\_Amount | BIGINT |

|  |  |
| --- | --- |
| **Payment** | |
| Payment\_ID | Integer |
| Sales\_ID | Integer |
| Payment\_Method | String |
| Date | date |

|  |  |
| --- | --- |
| **Stock** | |
| Stock\_ID | Integer |
| Product\_ID | Integer |
| Available\_Quantity | BIGINT |

UMLet:

Diagram

Description automatically generated

Crow Foot’s Notation:

Diagram

Description automatically generated

Relationships between tables:

Product-Sales: Product ßà Sales (one to many relationship)

Customer-Sales: Customer ßà Sales (one to many relationship).

Product-Stock: Product ßà Stock (one to one relationship).

Sales-Payment: Sales ßà Payment (one to one relationship).