Flipkart Clone Database Report

**Introduction**

Databases are vital tools for storing, managing, and retrieving information. They are also critical for building an e-commerce system. A well-structured database powers e-commerce and manages all the interactions within the system.

A good e-commerce database design includes:

* **Simple, functional database structure:**The database table structure is simple but covers all the required functionality without compromising the user experience.
* **High performance:**Database queries execute quickly to facilitate live customer interactions and support a frictionless shopping experience. Therefore, the selected database should have good indexing and performance optimization options.
* **High availability and scalability:**A good database design is highly available with automatic snapshots and enables automatic scaling to support future platform growth as well as sudden traffic spikes.

Database Diagram



Database Scope

The main consideration when designing the database is identifying the functionalities offered by the e-commerce platform. These functionalities can be further divided into core functions and additional functions.

**Core functions** are the functions necessary for facilitating the day-to-day operations of the e-commerce platform, including user management, product and inventory management, shopping cart function, payment management, and shipping/logistics management.

**Additional functions** are the nice-to-have functions for the e-commerce platform that enhance the user experience for both end-users (customers) and administrators (the business). Additional functions include marketing functions, help desk and support, advanced analytics, and third-party integrations.

**Core functions**

 The table fields and indexes depending on the design of the overall platform. It contains three separate sections for user management, product management, and shopping process. Let's have a closer look at each section.

**Customer (End-User) Management**

We have created a user table that contains all the user details along with **user\_payment** and **user\_address** tables to store multiple addresses and payment details of users. This structure offers more granular control over data while eliminating duplicate records.

****

Another way to manage users is by creating two separate user tables for end-users and administrators and assigning relationships according to their requirements



Customer (End-users) Management Tables Description

**Customer Table**

Description :- contains various field giving information about customer.

|  |  |
| --- | --- |
| CustoemrId | Unique index of customer |
| FirstName | First name of the customer |
| MiddleName | Middle name of customer |
| LastName | Last name of customer |
| PhoneNumber | Phone number of customer |
| UserName | A unique string contain login credentials for customer |
| Email | Email of customer |
| Active | Contain bool value of customer active or not |
| Registerd At | Time of registration of customer |
| Created At | Time Stamp of creation of record |
| ModifedAt | Time stamp of last Modification of record |

Customer Address Table

Description: - Container a one (customer) to many (addresses) relation between table customer and and table address

Customer Payment Detail Table

Description: - Container a one (customer) to many (payments)

Relation between customer and customer Payment.

In payment table field contain payment related columns like

Credit card or UPI id information

**Product Management**

Managing products is not simply about maintaining a list of products. We also have to manage the inventory, discounts, categories, and other attributes of the products. So always focus on simplifying the data structure while reducing duplicates. In the following table structure, the main product table contains information about the products.

There are two other separate tables called discount, **product\_inventory**, and **product\_category** that are connected to it through database relationships. This approach provides the greatest level of flexibility to the database.

For instance, we can simply query the **product\_inventory** table to check for inventory without going through all the data associated with other related tables. This is also a good place to utilize indexes to increase the performance of the database.



Product Management Tables

**Product Table**

Description: - Contains field containing information about product and relational key to other product management table

|  |  |
| --- | --- |
| ProductID | Unique indexer for each product |
| ProductTitle | Name of Product |
| ProductDescription | Brief detail about product |
| BrandID | Brand table key realtion indexer |
| CategoryId | Category table key realtion indexer |
| SubCatgoryId | SubCategory table key realtion index |
| MRP | Max Retail Price of Product |
| Price | Actual Price of Product |
| UnitWeight | Weight of product for logistic purpose |
| DiscountId | Discount table key relation indexer |
| IsActive | Is Product is listed and currently is sale or not |
| CreatedAT | Time Stamp |
| ModifeidAt | Time Stamp |

**Category Table**

Description: - Contain Category information on which product are classified or categorised

Ex: boat headphone comers under category Electronics

**SubCategory Table**

Description: - Conainer SubCategory information on which products of same category are further classified

Ex: boat headphone comes under subcategory headphone&Headset

**Brand Table**: Container information field of product brand

Details.

**Discount Table**: Container information regarding sale and offer on products

**Inventory table**: Container information regarding current stock of product as product can have different stock units at different warehouses it’s better to separate inventory table to for future scope.

**Seller**: Contains information regarding seller or supplier like contact number address and etc.

**Product seller**: is relation table between product and seller as each product can have multiple seller and each seller can have multiple product to sell.

**Shopping process**

This is the most critical and complex part when it comes to designing the database. The shopping process will guide a user to search the products, add the desired products to the shopping cart, and finally complete the transaction using a payment provider.

The heart of the e-commerce process connects users with products. A good chunk of design effort should be exhausted to streamline the shopping process.

In the example above, there are **shopping\_cart** and **cart\_item** as temporary data stores that only store the shopping session information of the current user until the order is confirmed and the data is moved to permanent storage tables with the payment details (**order\_details**, **order\_items**, and **payment\_details**).

