A2Z

Online Retail Store

Aayush Ranjan (2021003)
Aishiki Bhattacharya (2021007)

Scope:
Today, due to the extremely fast-paced lives of people, it can be exhausting and time-consuming to visit different stores at different locations. Visiting a market and choosing the right product is cumbersome and may take several hours. Also, due to the recent COVID-19 pandemic, people prefer to get products delivered to their doorstep as quickly as possible.

Therefore, to cater to the needs of the public, we provide complete management solutions to manage information on customers, sellers, delivery agents and products efficiently.

A2Z gives the admins a forum to showcase their products of various categories to customers in a hassle-free and systematic way. Customers can conveniently browse products, manage items in the cart and buy as per their preferred payment method. Delivery agents are assigned orders by the admin, which are supposed to be delivered within the stipulated time to provide an excellent experience to customers. Each of the objects is identified by a unique attribute, i.e., their primary key - Customer_ID, Admin_ID, Delivery_Agent_ID, Order_ID, Product_ID and so on.
This application will enable the admin, delivery agent and customer to interact and coordinate efficiently with each other, resulting in a pleasant experience for all.

Integrity Constraints:

- Each stakeholder, the customer, the admin and the delivery agent will necessarily have a unique ID(primary key) that they're identified with.
- Each Customer can only have one shopping cart assigned to them.
- A cart can have multiple products.
- The admin may have more than one product that belongs to them.
- One product can belong to only one admin.
- Each Product can belong to only one Category.
- Each category may have one or more than one product under it.
- There can be three types of accounts that can be created and accessed. Each type of account has a predefined set of privileges unique to that type of account.
- Customers can only rate those products which are present in their order history.
- Customers can only rate those delivery agents who have served them earlier

- All the attributes regarding price should be positive including zero
- All the attributes regarding discounts should be positive including zero
- Category names have to be unique
- All the important information like, IDs, Names, Contact, Location, etc. have to be NOT NULL
- Discounts can be NULL
- Customer’s and Admin’s Username should be unique
- All the IDs have been made the primary keys of the respective table
- The foreign keys have also been made in accordance with the relational schema

DDL

```
drop table if exists PAYMENT;
drop table if exists CONTAINS;
drop table if exists SUPPLIES;
drop table if exists RATES;
drop table if exists REVIEW;
drop table if exists `RETURN`;
drop table if exists DELIVERY_AGENT;
drop table if exists `ORDER`;
drop table if exists CART;
drop table if exists CUSTOMER_QUERY;
drop table if exists CUSTOMER;
drop table if exists PRODUCT;
drop table if exists CATEGORY;
drop table if exists ADMIN;
drop table if exists AGENT_CONTACT;
drop table if exists CUSTOMER_CONTACT;
drop table if exists AGENT_CONTACT;
drop table if exists CART_PRODUCT;

create table if not exists ADMIN (
        Admin_Id Integer NOT NULL UNIQUE,
        Name VARCHAR(50) NOT NULL,
        Company_Name VARCHAR(50) NOT NULL,
        Username VARCHAR(50) NOT NULL UNIQUE,
        Password VARCHAR(100) NOT NULL,
        Contact VARCHAR(50) NOT NULL UNIQUE,
   primary key(Admin_Id)
);

create table if not exists CUSTOMER (
        Customer_Id Integer NOT NULL UNIQUE,
        Name VARCHAR(50) NOT NULL,
        Username VARCHAR(50) NOT NULL UNIQUE,
        Password VARCHAR(50) NOT NULL,
        Address VARCHAR(50) NOT NULL,
        Membership VARCHAR(6),
        Wallet Integer NOT NULL,
   primary key(Customer_Id),
   constraint cust_wallet check(Wallet>=0)
);

create table if not exists CATEGORY (
        Category_Id Integer NOT NULL UNIQUE,
        Name VARCHAR(50) NOT NULL UNIQUE,
        No_Of_Products Integer NOT NULL,
   primary key(Category_Id),
   constraint categ_num check(No_Of_Products>=0)
);

create table if not exists PRODUCT (
        Product_Id Integer NOT NULL UNIQUE,
        Name VARCHAR(50) NOT NULL,
        Discount Integer,
        Price Integer NOT NULL,
        Stock Integer NOT NULL,
        About VARCHAR(100),
        Category_Id Integer NOT NULL,
   primary key(Product_Id),
   foreign key(Category_Id) references CATEGORY(Category_Id) on delete cascade,
   constraint prod_stock check(Stock>=0),
   constraint prod_price check(Price>0),
   constraint prod_disc check(Discount>=0)
);

create table if not exists CUSTOMER_QUERY (
        Query_Id Integer NOT NULL UNIQUE,
        Description VARCHAR(26) NOT NULL,
        Status VARCHAR(8) NOT NULL,
        Customer_Id Integer NOT NULL,
        Admin_Id Integer NOT NULL,
   primary key(Query_Id),
   foreign key(Customer_Id) references CUSTOMER(Customer_Id) on delete cascade,
   foreign key(Admin_Id) references ADMIN(Admin_Id) on delete cascade
);
```

```
create table if not exists CART (
        Cart_Id Integer NOT NULL UNIQUE,
   No_Of_Product Integer NOT NULL,
   Customer_Id Integer NOT NULL,
   foreign key(Customer_Id) references CUSTOMER(Customer_Id) on delete cascade,
   constraint cart_qty check(No_Of_Product>=0)
);

create table if not exists `ORDER` (
        Order_Id Integer NOT NULL UNIQUE,
        Cart_Id Integer NOT NULL,
        Customer_Id Integer NOT NULL,
        Admin_Id Integer NOT NULL,
        Order_Date DATETIME NOT NULL,
        Total_Price Integer NOT NULL,
        Total_Discount Integer,
        Location VARCHAR(50) NOT NULL,
   primary key(Order_Id),
   foreign key(Customer_Id) references CUSTOMER(Customer_Id) on delete cascade,
        foreign key(Admin_Id) references ADMIN(Admin_Id) on delete cascade,
   constraint order_price check(Total_Price>=0),
   constraint order_disc check(Total_Discount>=0)
);

create table if not exists DELIVERY_AGENT (
        Agent_Id Integer NOT NULL UNIQUE,
        Order_Id Integer NOT NULL,
        Admin_Id Integer NOT NULL,
        Name VARCHAR(50) NOT NULL,
        Status VARCHAR(9) NOT NULL,
        Rating DECIMAL(2,1) NOT NULL,
   primary key(Agent_Id),
   foreign key(Order_Id) references `ORDER`(Order_Id) on delete cascade,
   foreign key(Admin_Id) references Admin(Admin_Id) on delete cascade,
   constraint agent_rate check(Rating>=1 and Rating<=5)
);

create table if not exists `RETURN` (
        Order_Id Integer NOT NULL UNIQUE,
        Status VARCHAR(9) NOT NULL,
        Refund_Amount Integer NOT NULL,
        Customer_Id Integer NOT NULL,
        Agent_Id Integer NOT NULL,
            Contact VARCHAR(50) NOT NULL,
    foreign key(Order_Id) references `ORDER`(Order_Id),
    foreign key(Customer_Id) references CUSTOMER(Customer_Id) on delete cascade,
    foreign key(Agent_Id) references DELIVERY_AGENT(Agent_Id) on delete cascade,
    constraint return_refund check(Refund_Amount>=0)
);

create table if not exists PAYMENT (
            Payment_Id Integer NOT NULL UNIQUE,
            Order_Id Integer NOT NULL,
            Price Integer NOT NULL,
            Status VARCHAR(10) NOT NULL,
            Payment_Method VARCHAR(11) NOT NULL,
            Payment_Date DATETIME NOT NULL,
            Customer_Id Integer NOT NULL,
    primary key(Payment_Id),
    foreign key(Order_Id) references `ORDER`(Order_Id) on delete cascade,
    foreign key(Customer_Id) references CUSTOMER(Customer_Id) on delete cascade,
    constraint payment_price check(Price>=0)
);

create table if not exists REVIEW(
            Product_Id      Integer NOT NULL,
            Customer_Id Integer NOT NULL,
            Stars DECIMAL(2,1) NOT NULL,
            Description VARCHAR(15),
    foreign key(Product_Id) references PRODUCT(Product_Id) on delete cascade,
    foreign key(Customer_Id) references CUSTOMER(Customer_Id) on delete cascade,
    constraint review_stars check(Stars>=1 and Stars<=5)
);

create table SUPPLIES (
            Admin_Id Integer NOT NULL,
            Product_Id Integer NOT NULL,
    foreign key(Admin_Id) references ADMIN(Admin_Id) on delete cascade,
    foreign key(Product_Id) references PRODUCT(Product_Id) on delete cascade
);

create table if not exists RATES (
            Customer_Id Integer NOT NULL,
            Agent_Id Integer NOT NULL,
    foreign key(Customer_Id) references CUSTOMER(Customer_Id) on delete cascade,
    foreign key(Agent_Id) references DELIVERY_AGENT(Agent_Id) on delete cascade
```

```
);

create table if not exists CONTAINS (
        Customer_Id Integer NOT NULL,
        Cart_Id Integer NOT NULL,
        Product_Id Integer NOT NULL,
        foreign key(Customer_Id) references CUSTOMER(Customer_Id) on delete cascade,
   foreign key(Product_Id) references PRODUCT(Product_Id) on delete cascade
);

create table if not exists ADMIN_CONTACT (
        Admin_Id Integer NOT NULL,
        Contact VARCHAR(50) NOT NULL,
   foreign key(Admin_Id) references ADMIN(Admin_Id)
);

create table if not exists CUSTOMER_CONTACT (
        Customer_Id Integer NOT NULL,
        Contact VARCHAR(50) NOT NULL,
   foreign key(Customer_Id) references CUSTOMER(Customer_Id)
);

create table if not exists AGENT_CONTACT (
        Agent_Id Integer NOT NULL,
        Contact VARCHAR(50) NOT NULL,
   foreign key(Agent_Id) references DELILVERY_AGENT(Agent_Id)
);

create table if not exists CART_PRODUCT (
        Cart_Id Integer NOT NULL,
        Customer_Id Integer NOT NULL,
   Product_Id Integer NOT NULL,
   foreign key(Customer_Id) references CUSTOMER(Customer_Id),
   foreign key(Product_Id) references PRODUCT(Product_Id)
);
```

INDEXES

```
DROP INDEX categName on CATEGORY;
DROP INDEX custUserName on CUSTOMER;
DROP INDEX adminUserName on ADMIN;
DROP INDEX customerName on CUSTOMER;
```

DROP INDEX adminName on ADMIN;
DROP INDEX prodName on PRODUCT;

create unique index categName on CATEGORY(Name);
- To allow quick access using category names as they are unique

create unique index custUserName on CUSTOMER(Username);
- To allow quick access using customer's username as they are unique

create unique index adminUserName on ADMIN(Username);
- To allow quick access using admin's username as they are unique

create index customerName on CUSTOMER(Name);
- To allow quick access using customer's name which are not unique but an important information

create index adminName on ADMIN(Name);
- To allow quick access using admin's name which are not unique but an important information

create index prodName on PRODUCT(Name);
- To allow quick access using product's name which are not unique but an important information