**Online Shopping portal**



Project report for

Database Management System (CS201)

April 28, 2016

**Acknowledgment**

Working on this project “Mongo DB” was a source of immense knowledge to us. We would like to express our gratitude to our instructor Mr. Amit Kumar for his guidance and valuable support throughout the course of this project work. We acknowledge with deep sense of gratitude, the encouragement and inspiration received from our faculty members and colleagues. We would also like to thank our parents for their love and support.

Thank you

Kandula Mohan Sai U101114FCS076

Attili Uday U101114FCS053

Dandu Geet Kamal Tej U101114FCS060

Loga Prakash U101114FCS087

M Andrews U101114FBT170

**PROBLEM STATEMENT**

This ER diagram represents an online shopping portal. Online shopping makes things easy and simple. Now a days everyone are interested in getting things online rather going to shop and purchasing them. Imagine a customer who wants to purchase several things in online. To do so he need to proceed with all the below procedures. As a buyer or customer he should have username, customer ID, password, mobile number, email ID and address. He then visits his things of need and also he can contact the dealer services. Services should contain product name, order ID, complaint or feedback and warranty/repairs. A customer should register first to access all the products in website with pre-book. Which consists of email ID, Booking ID, Product ID and Product name. Customer visits the category or category name. Which contains products, sells and offers. Product should be of description, product name, product ID, product image and price. Sells should be of seller with customer ID, product details, seller details, price and bank account details. Offers must contain product ID, offer period, discount% and coupons. Products are to be clearly stated with supplies, availability, payment options and brands. Supplies with manufacturer, manufacturer ID and contact details. Brand consists of brand ID and brand name. Availability of a stock must contain quantity, stock ID, date, product ID. Payment options should consists of cart, payment, order details, delivery and cancellation. Cart has items and quantity selected by customer. Payment has payment method, payment ID, payment details and amount. Order details consists of order ID, shipping details and order date. Delivery consists of estimated delivery date, tracking ID, order ID, address and status. Cancellation consists of order ID, username, reason for cancellation and refund.

**INTRODUCTION**

In the information age, anything we want we can get only in one mouse click. Speed, reliability and accuracy of the computer make it a powerful tool for different purposes. A very important and basic need of today’s modern business world is the quick availability and processing of information using computer. One can easily get the type of required information within a fraction of a second.

The objective of this project is to design the database of an online shopping portal.

**ER MODEL**

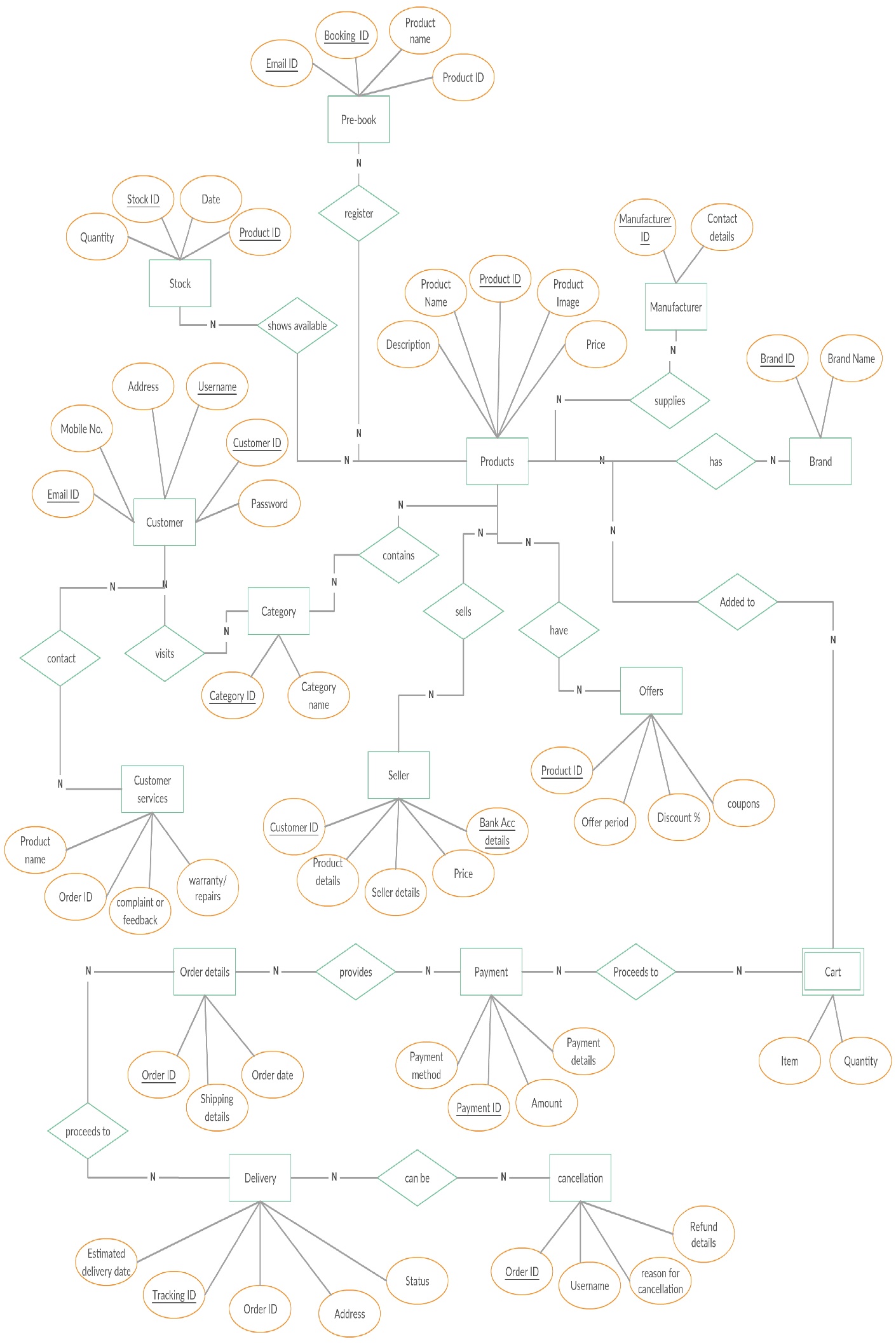
ER model represents information aspects of a business domain or its process requirements in a diagrammatic and systematic form which is implemented in the database. ER model consists of entities which are connected by relationships that express dependencies between them. Each entity has its own set of attributes. The ER model is generally implemented in a database. In relational database, data is stored in form of tables. Each row in the table represents an instance of an entity.

Entities & attributes in our ER model

* Customer (customer id, email id, username, password, mobile no., address)
* Category (category id, category name)
* Products (product id, product name, description, product image, price)
* Manufacturer (manufacturer id, contact details)
* Brand (brand id, brand name)
* Offers (product id, offer period, discount%, coupons)
* Seller (customer id, product details, seller details, price, account details)
* Customer services (order id, product id, complaint/feedback, warranty period)
* Cart (item, quantity)
* Payment (payment id, payment method, payment details, amount)
* Order details (order id, tracking id, shipping details, order date)
* Delivery (tracking id, order id, status, estimated date, address)
* Cancellation (order id, username, reason for cancellation, refund details)
* Stock (product id, stock id, quantity, date)
* Pre-book (booking id, email id, product id, product name)

Relationships between the entities are

* Visits- customer visits category.
* Contact- customer can contact customer services.
* Contains- category contains products.
* Has- products has brands.
* Supplies- product is supplied by a manufacturer.
* Have- products have offers.
* Sells- products are sold by sellers.
* Shows available- stock show availability of the product.
* Register- product is registered for pre-book.
* Added to- product is added to cart if he/she or wants to buy it. Here cart is a weak attribute and is totally dependent on product.
* Proceeds to- customer proceeds for payment after is added to cart.
* Provides- payment provides order details.
* Proceed to- after order is placed, it proceeds for delivery.
* Can be- delivery can be cancelled.



**Database Normalization**

Normalisation is a technique of organizing the data in the database. It is a systematic approach of decomposing tables used for mainly two purposes -

* Eliminating redundant (useless) data and undesirable characteristics like Insertion, Update and Deletion anomalies.
* Ensuring data dependencies are stored.

**Normalization rules**

* First Normal Form - As per 1NF all attributes must be atomic.
* Second Normal Form - As per the 2NF there must not be any partial dependency of any column on primary key and table must be in 1NF.
* Third Normal Form- there should be no transitive functional dependencies i.e. a non-prime attribute should not be determined by another non-prime attribute and also the table should be 2NF.
* BCNF (Boyce and Codd Normal Form)- A 3NF table which does not have multiple overlapping candidate keys is said to be in BCNF. For a table to be in BCNF, following conditions must be satisfied:

R must be in 3NF and, for each functional dependency (X -> Y),

X should be a super key.

**Customer**

Primary key- customer id

Candidate key- username, email id

Functional dependencies-

Cust\_id->email\_id, username, mob. No., address, password

Email\_id->cust\_id, username, mob. No., address, password

Username-> email\_id, cust\_id, mob. No., address, password

LHs is a superkey in each FDs.

Hence it’s in BCNF.

The normal form chosen for the database is BCNF. We chose BCNF and not a higher normal form as our decompositions were dependency-preserving and we wish to achieve least redundancy in our database to minimize any scope of anomalies and also, since checking functional dependencies other than primary key constraints are difficult in SQL.

**Category**

Primary key- category id

Functional dependencies-

Cat\_id->cat\_name

The normal form chosen for the database is BCNF. We chose BCNF and not a higher normal form as our decompositions were dependency-preserving and we wish to achieve least redundancy in our database to minimize any scope of anomalies and also, since checking functional dependencies other than primary key constraints are difficult in SQL.

**Product**

Primary key- product id

Functional dependencies-

Prod\_id-> prod\_name, description, price

The normal form chosen for the database is BCNF. We chose BCNF and not a higher normal form as our decompositions were dependency-preserving and we wish to achieve least redundancy in our database to minimize any scope of anomalies and also, since checking functional dependencies other than primary key constraints are difficult in SQL.

**Manufacturer**

Primary key- manufacturer id

Functional dependencies-

Manf\_id->contact details

The normal form chosen for the database is BCNF. We chose BCNF and not a higher normal form as our decompositions were dependency-preserving and we wish to achieve least redundancy in our database to minimize any scope of anomalies and also, since checking functional dependencies other than primary key constraints are difficult in SQL.

**Brand**

Primary key- brand id

Functional dependencies-

Brand\_id->brand name

The normal form chosen for the database is BCNF. We chose BCNF and not a higher normal form as our decompositions were dependency-preserving and we wish to achieve least redundancy in our database to minimize any scope of anomalies and also, since checking functional dependencies other than primary key constraints are difficult in SQL.

**Offers**

Primary key- product id, coupon code

Functional dependencies-

offer\_id, coupon code-> offer period, discount%, validity

offer\_id-> offer period, discount%

Coupon code->discount %, validity

Since coupon code is a subset of the primary key the table is not even in 2NF. Hence we will decompose the table as follows –

1. Sales (prod\_id, discount%, offer period)
2. Voucher (coupon code, discount%, validity)

Now it is in BCNF. The normal form chosen for the database is Boyce Codd Normal Form. We chose BCNF and not a higher normal form as our decompositions were dependency-preserving and we wish to achieve least redundancy in our database to minimize any scope of anomalies and also, since checking functional dependencies other than primary key constraints are difficult in SQL.

**Seller**

Primary key- seller id

Functional dependencies-

Sell\_id->sell\_name, sell\_address, product details.

The normal form chosen for the database is BCNF. We chose BCNF and not a higher normal form as our decompositions were dependency-preserving and we wish to achieve least redundancy in our database to minimize any scope of anomalies and also, since checking functional dependencies other than primary key constraints are difficult in SQL.

**Customer** **services**

Primary key- order id

Candidate key- product id

Functional dependencies-

Order\_id->prod\_id, compliant/feedback, warranty

Prod\_id->order\_id

LHS is a superkey in each FDS. Hence it’s in BCNF.

The normal form chosen for the database is BCNF. We chose BCNF and not a higher normal form as our decompositions were dependency-preserving and we wish to achieve least redundancy in our database to minimize any scope of anomalies and also, since checking functional dependencies other than primary key constraints are difficult in SQL.

**Payment**

Primary key- payment id

Functional dependencies-

Pay\_id->pay\_method, pay\_details, amount

The normal form chosen for the database is BCNF. We chose BCNF and not a higher normal form as our decompositions were dependency-preserving and we wish to achieve least redundancy in our database to minimize any scope of anomalies and also, since checking functional dependencies other than primary key constraints are difficult in SQL.

**Order details**

Primary key- order id

Functional dependencies-

Ord\_id->shipping details, order date

The normal form chosen for the database is BCNF. We chose BCNF and not a higher normal form as our decompositions were dependency-preserving and we wish to achieve least redundancy in our database to minimize any scope of anomalies and also, since checking functional dependencies other than primary key constraints are difficult in SQL.

**Delivery**

Primary key- tracking id

Candidate key- order id

Functional dependencies-

Track\_id ->ord\_id, name, address, status, estimated date

The normal form chosen for the database is BCNF. We chose BCNF and not a higher normal form as our decompositions were dependency-preserving and we wish to achieve least redundancy in our database to minimize any scope of anomalies and also, since checking functional dependencies other than primary key constraints are difficult in SQL.

**Cancellation**

Primary key- order id

Functional dependencies-

Ord\_id-> username, reason for cancellation, refund details

The normal form chosen for the database is BCNF. We chose BCNF and not a higher normal form as our decompositions were dependency-preserving and we wish to achieve least redundancy in our database to minimize any scope of anomalies and also, since checking functional dependencies other than primary key constraints are difficult in SQL.

**Stock**

Primary key- stock id

candidate key- product id

Functional dependencies-

Stock\_id-> prod\_id, quantity, date.

The normal form chosen for the database is BCNF. We chose BCNF and not a higher normal form as our decompositions were dependency-preserving and we wish to achieve least redundancy in our database to minimize any scope of anomalies and also, since checking functional dependencies other than primary key constraints are difficult in SQL.

**Pre-book**

Primary key- booking id

Candidate key- email id

Functional dependencies-

book\_id-> email\_id, prod\_id, product name

The normal form chosen for the database is BCNF. We chose BCNF and not a higher normal form as our decompositions were dependency-preserving and we wish to achieve least redundancy in our database to minimize any scope of anomalies and also, since checking functional dependencies other than primary key constraints are difficult in SQL.

**Cart** Primary key- item\_id

Functional dependencies - None Hence, it’s already in BCNF

**Conversion of ER to Relational Model**

All entities are converted to tables as per following rules-

One to many relationships- The ‘many’ side has total participation and the key from the ‘one’ side acts as a foreign key in the ‘many’ side table.

1. Customer visits category

Primary key- cat\_id

Foreign key- cust\_id

(cat\_id, cat\_name, cust\_id)

1. Category contains product

Primary key- prod\_id

Foreign key- cat\_id

(prod\_id, prod\_name, description, price, cat\_id)

1. Stock shows availability of products

Primary key-prod\_id

Foreign key- stock\_id

(prod\_id, prod\_name, description, price, stock\_id)

1. Products can be pre-ordered

Primary key- prod\_id

Foreign key- book\_id

(prod\_id, prod\_name, description, price, book\_id)

1. Seller sells product

Primary key- prod\_id

Foreign key- sell\_id

(prod\_id, prod\_name, description, price, sell\_id)

1. Manufacturer supplies product

Primary key- prod\_id

Foreign key- manf\_id

(prod\_id, prod\_name, description, price, manf\_id)

1. Customer contacts customer services

Primary key- cust\_id

Foreign key- order\_id

(cust\_id, email\_id, username, mob. No., address, password)

One to one relationships- The key from any of the ‘one’ side can act as a foreign key in other ‘one’ side, whichever is relevant.

1. Product has offer

Primary key- Coupon code in voucher

Offer\_id in sales

Foreign key- prod\_id in both tables

(coupon code, discount%, validity, prod\_id)

1. Payment provides order details

Primary key- order\_id

Foreign key- pay\_id

(order\_id, pay\_id, shipping details, order date)

1. Delivery of product after receiving order details

Primary key- track\_id

Foreign key- order\_id

(track\_id, order\_id, address, estimated date, status)

1. Delivery can be cancelled

Primary key- order\_id

Foreign key- track\_id

(order\_id, track\_id, username, reason for cancellation, refund details).

Weak entities- The primary key of the strong entity acts as a foreign key in weak entity table.

1. Product is added to cart

Foreign key- product id

(prod\_id, item, quantity)

**Table Schema**

create table customer details (

cust\_id int not null,

username text not null,

email\_id text not null,

mob.no. int not null,

caddress text,

upassword text not null,

primary key(customer\_Id)

);

|  |  |  |  |  |  |
| --- | --- | --- | --- | --- | --- |
| Cust\_id | username | Email\_id | Mob.no. | Caddress | upassword |

create table customer\_services (

prod\_id int not null,

order\_id int not null,

feedback text,

warranty\_period datetime not null,

primary key (order\_id),

foreign key(cust\_id) references customer details (cust\_id)

);

|  |  |  |  |
| --- | --- | --- | --- |
| cust\_id | Order\_id | Feedback | Warranty period |

create table category (

cat\_id int not null,

product\_id int not null,

cat\_name text not null,

primary key (cat\_id)

foreign key (prod\_id) references product (prod\_id)

);

|  |  |  |
| --- | --- | --- |
| Prod\_id | cat\_id | Cat\_name |

create table products (

prod\_id int not null,

prod\_name text not null,

prod\_description text,

price int not null,

primary key (product\_id)

);

|  |  |  |  |
| --- | --- | --- | --- |
| Prod\_id | Prod\_name | Prod\_description | Price |

create table brands (

brand\_id int not null,

product\_id int not null,

brand\_name text not null,

primary key (brand\_id)

foreign key (prod\_id) references product (prod\_id)

);

|  |  |  |
| --- | --- | --- |
| Prod\_id | Brand\_id | Brand\_name |

create table seller (

sell\_id int not null,

product\_id int not null,

sell\_name text not null,

sell\_address text,

product\_details text not null,

primary key (seller\_id)

foreign key (prod\_id) references product (prod\_id)

);

|  |  |  |  |
| --- | --- | --- | --- |
| Prod\_id | Sell\_id | Sell\_name | Sell\_address |

create table sales (

prod\_id int not null,

offer\_id int not null,

offer\_period datetime,

discount int not null,

primary key(offer\_id),

foreign key(prod\_id) references products (prod\_id)

);

|  |  |  |  |
| --- | --- | --- | --- |
| Prod\_id | Offer\_id | Offer\_period | Discount |

create table vouchers (

prod\_id int not null,

coupon\_code int not null,

offer\_period datetime,

discount int not null,

primary key(offer\_id),

foreign key(prod\_id) references products (prod\_id)

);

|  |  |  |  |
| --- | --- | --- | --- |
| Prod\_id | Coupon\_code | Offer\_period | Discount |

create table payment (

pay\_id int not null,

pay\_method text not null,

pay\_details text not null,

order\_id int not null,

amount int not null,

primary key (pay\_id),

foreign key(order\_id) references customer\_services (order\_id)

);

|  |  |  |  |  |
| --- | --- | --- | --- | --- |
| Pay\_id | Order\_id | Pay\_method | Pay\_details | Amount |

create table delivery (

track\_id int not null,

order\_id int not null,

status text not null,

estimated\_date datetime not null,

dname text not null,

daddress text not null,

primary key (track\_id),

foreign key(order\_id) references customer\_services (order\_id)

);

|  |  |  |  |  |  |
| --- | --- | --- | --- | --- | --- |
| Track\_id | Order\_id | status | Estimated date | Dname | Daddress |

create table cancellation (

order\_id int not null,

username text not null,

reason text not null,

refund int,

primary key(username),

foreign key(order\_id) references customer\_services (order\_id)

);

|  |  |  |  |
| --- | --- | --- | --- |
| Order\_id | username | Reason | Refund |

create table manufacturer (

manf\_id int not null,

product\_id int not null,

contact\_details int not null,

primary key (manf\_id)

foreign key(prod\_id) references products (prod\_id)

);

|  |  |  |
| --- | --- | --- |
| Manf\_id | Prod\_id | Contact details |

create table stock (

stock\_id int not null,

product\_id int not null,

sdate datetime not null,

quantity int not null,

primary key (stock\_id),

foreign key(product\_id) references products (product\_id)

);

|  |  |  |  |
| --- | --- | --- | --- |
| Stock\_id | Product\_id | sdate | Quantity |

create table cart (

item text not null,

quantity int not null,

prod\_id int not null,

foreign key(product\_id) references products(product\_id)

);

|  |  |  |
| --- | --- | --- |
| Item | Quantity | Prod\_id |

create table prebook (

email\_id text not null,

book\_id int not null,

prod\_id int not null,

prod\_name text not null,

primary key (booking\_id),

foreign key(product\_id) references products (product\_id)

);

|  |  |  |  |
| --- | --- | --- | --- |
| Email\_id | Book\_id | Prod\_id | Prod\_name |

create table order\_details (

order\_id int not null,

shipping\_details text not null,

order\_date datetime not null,

primary key (order\_id)

foreign key (pay\_id) references products (pay\_id)

);

|  |  |  |
| --- | --- | --- |
| Order\_id | Shipping details | Order\_date |