# Group Project – SQL Injectors

**User Requirement Analysis**

Entity and Attributes:

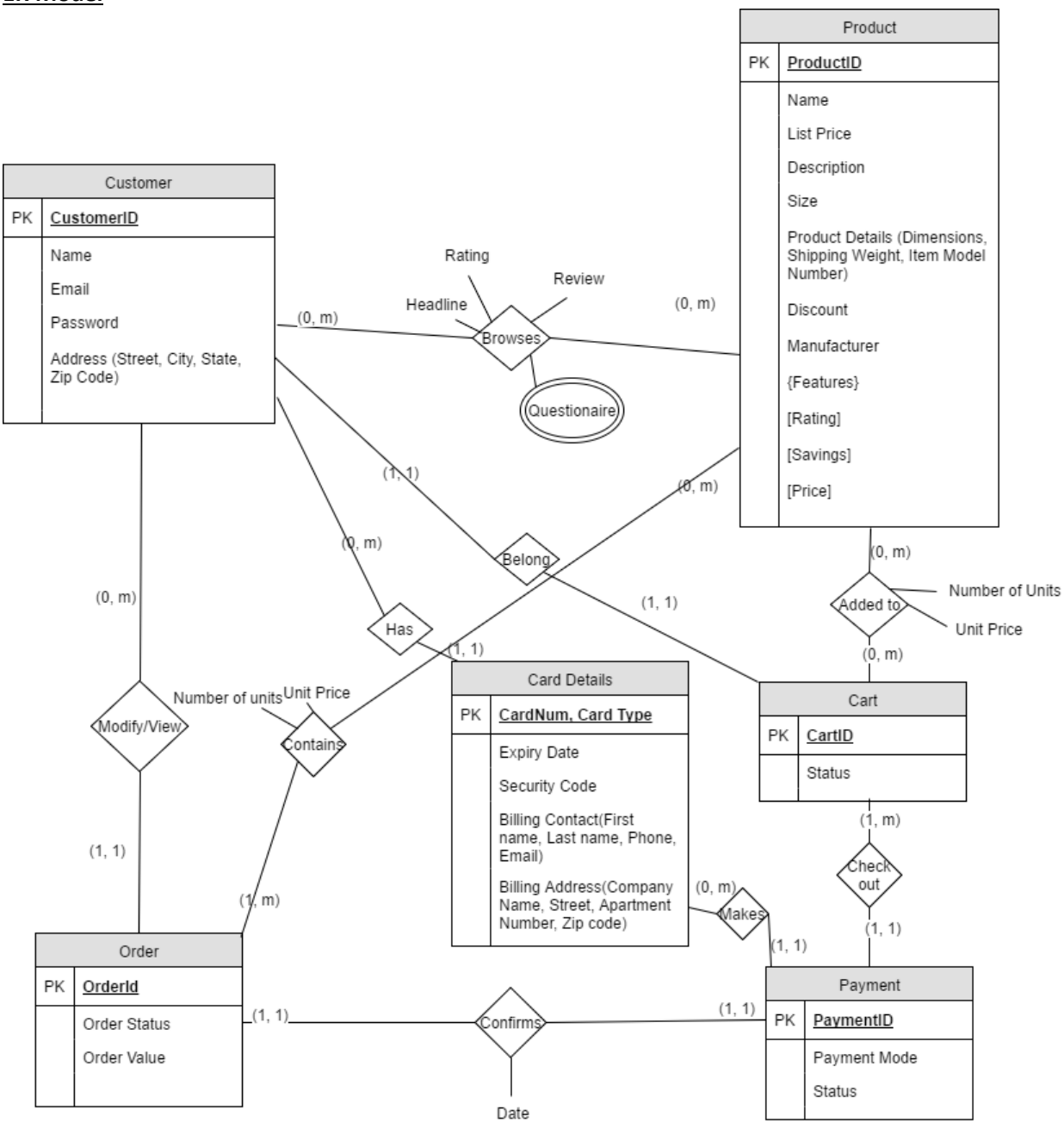
1. Customer – Any person who visits the online retail store PetAtl and/or orders product(s). The attributes of Customer are mentioned below:
    - CustomerID – The unique number/code which identifies each customer
    - Name – The name of the customer
    - Password – The unique password of a particluar customer used to access his account on the online retail store
    - Address – It comprises of Street, State, City, Zip Code for a particular customer
2. Product – The packaged dry dog food products available on the PetAtl online retail store. The attributes of Product are mentioned below:
    - ProductID – The unique number/code which uniquely identifies each product
    - Name – The name of the product
    - List Price – The price of the product
    - Description – It lists down the description of the Product
    - Size – It states the weight of the product
    - Product Details – It consists of product dimensions, shipping weight and item model number
    - Discount – It consists of the discount available on the product in terms of the saving
    - Manufacturer – The company manufacturing the product is the manufacturer
    - Feature – It lists the main features of the product
3. Card Details – This will consist of the card details of a customer. The attributes are:
    - CardNum – This is the card number of the customer
    - CardType – This is the card type of the customer e.g., visa, master, etc.
    - Expiry Date – This is the card expiry date
    - Security Code – This is the code which the Customer needs for making a payment
    - Billing Contact – The first name, last name , phone, email corresponding to the card
    - Billing Address – The company name, street, apartment number, zip code corresponding to the card used to make payment
4. Order – This will include the order information after the customer has bought some products. The attributes are:
    - OrderID – This is a unique number/code which identifies each order made by the customer
    - Order Status – This shows if the order is in-process, in-transition or shipped
    - Order Value – This shows the aggregate value/amount of the order placed by the customer
5. Cart – This represents a virtual shopping cart where a customer puts his/her selected products which he/she may add later. The attributes are
    - CartID – This is a unique number/code which identifies a cart
    - Cart Status – This will show the cart status which is either active or checked-out

6. Payment – This represents the payment made by a customer for an order. The attributes are mentioned below:
    - PaymentID – The unique number associated with a payment
    - Payment Status – This represents the status of the payment which would be successful and unsuccessful

Business Rules:

1. A Customer can browse many Products and one product can be browsed by many Customers. When a customer browses a product, customer can record rating of the product based on his/her experience, can give reviews and can answer to some questions about the product.
2. Many products can be added to one cart. While adding products to the card, number of units of the product should be recorded.
3. One cart belongs to one customer.
4. A customer has many card details by which a customer can make payment. One card belongs to only one customer.
5. One cart can be checked out by only one Payment and one payment can be used to checkout only one cart.
6. Once the payment is done, order can be confirmed. One payment can be used to confirm one order.
7. One order belongs to one customer and one customer can place multiple orders.
8. One Order can contain multiple products.
9. Once the cart is checked out , the cart status changes to checked-out. So, for a customer there will be only one active cart.
10. A customer can modify the cart at any time.
11. A customer can modify or cancel the order once it is placed.

## ER Model


Product
PK
ProductID
Name
List Price
Description
Size
Product Details (Dimensions, Shipping Weight, Item Model Number)
Discount
Manufacturer
{Features}
[Rating]
[Savings]
[Price]
Customer
PK
CustomerID
Name
Email
Password
Address (Street, City, State, Zip Code)
Rating
Review
Headline
Browses
(0, m)
(0, m)
Questionaire
(1, 1)
(0, m)
(0, m)
Belong
(0, m)
Added to
Number of Units
Unit Price
(0, m)
(1, 1)
(0, m)
Has
(1, 1)
Card Details
PK
CardNum, Card Type
Expiry Date
Security Code
Billing Contact(First name, Last name, Phone, Email)
Billing Address(Company Name, Street, Apartment Number, Zip code)
Cart
PK
CartID
Status
Number of units
Unit Price
Modify/View
Contains
(1, 1)
(1, m)
(1, m)
Check out
(0, m)
Makes
(1, 1)
(1, 1)
Order
PK
OrderId
Order Status
Order Value
(1, 1)
Confirms
(1, 1)
Payment
PK
PaymentID
Payment Mode
Status
Date

**ER model Entities**

The entities in the ER model are mapped into respective entity type tables.

1. Customer Entity

   Since the Address is composite attribute they would be included as separate attributes in the table.

   CustomerID -> Primary Key

   Customer (CustomerID, Name, Email, Password, Street, City, State, ZipCode)

2. Order Entity

   OrderID->Primary Key

   Order (OrderID, OrderStatus, OrderValue)

3. Product Entity

   Product Details: The attributes are composite in nature and need to be stored as individual attributes in the table “Product”.
   The multivalued attribute Feature is mapped into a separate multivalued attribute table.

   ProductID -> Primary Key

   Product (ProductID, Name, ListPrice, Description, Size, Dimension, Shipping weight, ItemModelNumber, Discount, Manufacturer, price)

   ProductFeatures (ProductID, Features)

4. Cart Entity

   CartID -> Primary Key

   Cart (CartID, Cart Status)

5. Payment Entity

   PaymentID->Primary Key

   Payment (PaymentID, Payment Status)

6. Card

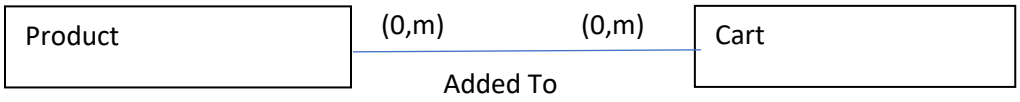
   Billing information: It is a composite attribute so we would require to store individual attributes in the table

   Billing Address: It is a composite attribute so we would require to store individual attributes in the table

   CardNum, CardType -> Primary Key

   Card( CardNum, CardType, ExpiryDate, SecurityCode, FirstName, LastName, Phone, Email, Company Name, Street, ApartmentNumber, Zip, Code)

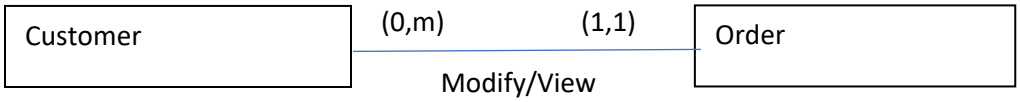
Map Relationship between Product and Cart




Product and Cart has many to many relationship. According to mapping rules, we have to insert new table-CartProduct

CartProducts (CartID, ProductID(fk), Number of Units)

Map Relationship between Customer and Order

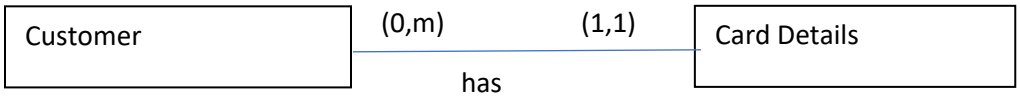



Customer and Order has 1 to many relationship. So here according to mapping rules, we have to insert foreign key of Customer into Order.

Customer(CustomerID, Name, Email, Street, City, Zip Code, State, Password)

Order(OrderID, Order Status, Order Value, CustomerID(fk), Order Date)

Map Relationship between Customer and Card Details

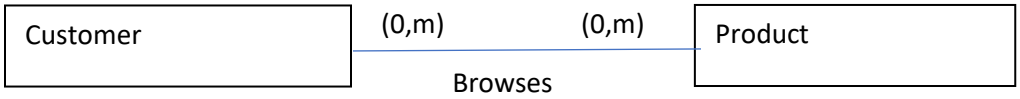



Customer and CardDetails has 1 to many relationship.So here according to mapping rules ,we have to insert foreign key of Customer into Card Details.

Customer(CustomerID, Name, Email, Street, City, Zip Code, State, Password)

CardDetails(CustomerID(fk), Card Type, Card Number, Expiry Date, Security Code, First Name, Last Name, Phone, Email, Company Name, Street, Apartment Number, Zip Code)

Map Relationship between Customer and Product

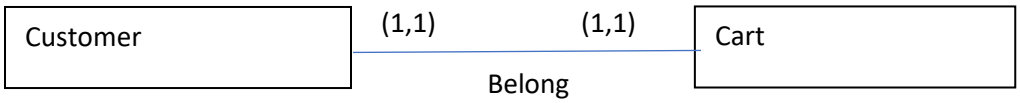



Customer and Product has many to many relationship. According to mapping rules ,we can insert table called CustomerProduct

CustomerProduct(CustomerID(fk), ProductID(fk), Rating, Review)

ProductQuestionaire(Customer_ID(fk), Product_ID(fk), Question, Answer)

Map Relationship between Customer and Cart

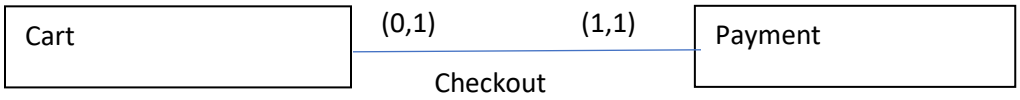



Customer and Cart has 1 to 1 relationship.So here according to mapping rules ,we can insert foreign key of Customer into Cart.

Customer(CustomerID, Name, Email, Street, City, Zip Code, State, Password)

Cart(CartID, CustomerID(fk), Status)

Map Relationship between Cart and Payment

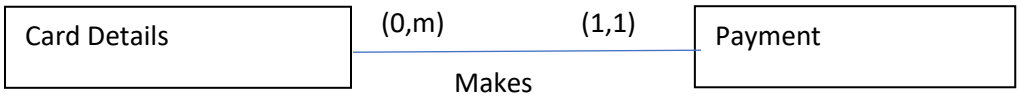



Cart and Payment has 1 to 1 relationship.So here according to mapping rules ,we can insert foreign key of Cart into Paymemt.

Payment(PaymentID, Status, CartID(fk))

Cart(CartID, CustomerID(fk), Status)

Map Relationship between Card details and Payment




There is a one to many relationship between Card Details and Payment. According to mapping rules, the primary key of Card Details is inserted into Payment

Payment(PaymentID, Status, CartID(fk), CardNum(fk), CardType(fk))

Map Relationship between Order and Payment


Order and Payment has 1 to 1 relationship. According to mapping rules ,we can insert primary key of Payment into Order.

Order (OrderID, PaymentID(fk), Order Status, OrderValue, OrderDate, CustomerID(fk))

**Relation Normalization:**

**Customer(<u>CustomerID</u>, Name, Email, Street, City, Zip Code, State, Password)**

It has only one candidate key CustomerID and is selected as the primary key by default.
The non-prime attributes are: Name, Email, Street, City, Zip Code, State, Password
The functional dependencies are:

CustomerID -> Name, Email, Street, City, Zip Code, State, Password

Based on the normalization theory and the above FDs, we conclude that:

- Customer is in 2NF because all non-prime attributes are fully depended on the key.
- Customer is in 3NF because there is no any non-prime attribute that is transitively depended on the key through a non-prime attribute.

**Product(<u>ProductID</u>, Name, Price, Discount, ProductSize, Dimensions, Shipping Weight, Model Number, Description, Image, Manufacturer)**

It has only one candidate key ProductID and is selected as the primary key by default.
The non-prime attributes are: Name, Price, Discount, ProductSize, Dimensions, Shipping Weight, Model Number, Description, Image, Manufacturer

The functional dependencies are:

ProductID -> Name, Price, Discount, Size, Dimensions, Shipping Weight, Model Number, Description, Image, Manufacturer

Based on the normalization theory and the above FDs, we conclude that:

- Product is in 2NF because all non-prime attributes are fully depended on the key.
- Product is in 3NF because there is no any non-prime attribute that is transitively depended on the key through a non-prime attribute.

**ProductFeature(<u>ProductID(fk), Feature</u>)**

It has only one candidate key, (ProductID(fk), Feature), and is the primary key by default. There is no any non-prime attribute because all attributes are part of the key, i.e., prime attributes.

There is no functional dependency between ProductID(fk) and Feature because a product may have multiple feature, and a feature may be shared by multiple Products. That is, the dependencies between them are multivalued dependencies as shown below:

ProductID(fk) ->> Feature
Feature ->> ProductID (fk)

Thus, we conclude that the table is in both 2NF and 3NF because there is no non-prime attribute in the relation by NF definitions.

**Cart(<u>CartID, CustomerID(fk)</u>, Status)**

It has only one candidate key CartID and is selected as the primary key by default.
The non-prime attributes are: Status

The functional dependencies are:

CartID -> CustomerID(fk), Status

Based on the normalization theory and the above FDs, we conclude that:

- Cart is in 2NF because all non-prime attributes are fully depended on the key.
- Cart is in 3NF because there is no any non-prime attribute that is transitively depended on the key through a non-prime attribute.

**CartProducts(<u>CartID, ProductID(fk)</u>, Number of Units)**

It has only one candidate key (CartID, ProductID(fk)) and is selected as the primary key by default.
The non-prime attributes are: Number of Units

The functional dependencies are:

(CartID, ProductID(fk)) -> Number of Units

Based on the normalization theory and the above FDs, we conclude that:

- CartProducts is in 2NF because all non-prime attributes are fully depended on the key.
- CartProducts is in 3NF because there is no any non-prime attribute that is transitively depended on the key through a non-prime attribute.

**Payment(<u>PaymentID</u>, Status, CartID(fk), CardNum(fk), CardType(fk))**

It has only one candidate key PaymentID and is selected as the primary key by default.
The non-prime attributes are: Status, Mode, CartID(fk), CardNum(fk), CardType(fk)

The functional dependencies are:

PaymentID -> Status, Mode, CartID(fk), CardNum(fk), CardType(fk)

Based on the normalization theory and the above FDs, we conclude that:

- Payment is in 2NF because all non-prime attributes are fully depended on the key.

- Payment is in 3NF because there is no any non-prime attribute that is transitively depended on the key through a non-prime attribute.

**Order(<u>OrderID</u>, Order Status, Order Value, Order Date, CustomerID(fk) , PaymentID(fk))**

It has only one candidate key OrderID and is selected as the primary key by default.
The non-prime attributes are: Order Status, Order Value, Order Date, CustomerID(fk) , PaymentID(fk)

The functional dependencies are:

OrderID -> Order Status, Order Value, Order Date, CustomerID(fk) , PaymentID(fk)

Based on the normalization theory and the above FDs, we conclude that:

- Order is in 2NF because all non-prime attributes are fully depended on the key.
- Order is in 3NF because there is no any non-prime attribute that is transitively depended on the key through a non-prime attribute.

**OrderDetails(<u>OrderID</u>, <u>ProductID(fk)</u>, Number of Units, Unit Price)**

It has only one candidate key (OrderID, ProductID(fk)) and is selected as the primary key by default.
The non-prime attributes are: Number of Units, Unit Price

The functional dependencies are:

(OrderID, ProductID(fk)) -> Number of Units, Unit Price

Based on the normalization theory and the above FDs, we conclude that:

- OrderDetails is in 2NF because all non-prime attributes are fully depended on the key.
- OrderDetails is in 3NF because there is no any non-prime attribute that is transitively depended on the key through a non-prime attribute.

**CardDetails(<u>CustomerID(fk)</u>, <u>CardType</u>, <u>CardNum</u>, Expiry Date, Security Code, First Name, Last Name, Phone, Email, Company Name, Street, Apartment Number, Zip Code)**

It has only one candidate key (CustomerID(fk), CardType, CardNum) and is selected as the primary key by default.
The non-prime attributes are: Expiry Date, Security Code, First Name, Last Name, Phone, Email, Company Name, Street, Apartment Number, Zip Code

The functional dependencies are:

(CustomerID(fk), CardType, CardNum) -> Expiry Date, Security Code, First Name, Last Name, Phone, Email, Company Name, Street, Apartment Number, Zip Code

Based on the normalization theory and the above FDs, we conclude that:

- CardDetails is in 2NF because all non-prime attributes are fully depended on the key.
- CardDetails is in 3NF because there is no any non-prime attribute that is transitively depended on the key through a non-prime attribute.

**CustomerProduct(CustomerID(fk), ProductID(fk), Rating, Headline, Review)**

It has only one candidate key (CustomerID(fk), ProductID(fk)) and is selected as the primary key by default.
The non-prime attributes are: Rating, Review, Headline

The functional dependencies are:

(CustomerID(fk), ProductID(fk)) -> Rating, Review, Headline

Based on the normalization theory and the above FDs, we conclude that:

- CustomerProduct is in 2NF because all non-prime attributes are fully depended on the key.
- CustomerProduct is in 3NF because there is no any non-prime attribute that is transitively depended on the key through a non-prime attribute.

**ProductQuestionnaire(CustomerID(fk), ProductID(fk), Question, Answer)**

It has only one candidate key (CustomerID(fk), ProductID(fk), Question) and is selected as the primary key by default.
The non-prime attributes are: Answer

The functional dependencies are:

(CustomerID(fk), ProductID(fk), Question) -> Answer

Based on the normalization theory and the above FDs, we conclude that:

- ProductQuestionnaire is in 2NF because all non-prime attributes are fully depended on the key.
- ProductQuestionnaire is in 3NF because there is no any non-prime attribute that is transitively depended on the key through a non-prime attribute.

Replacement of composite keys:

CartProducts(CartID, ProductID(fk), Number of Units)

OrderDetails(OrderID, ProductID(fk), Number of Units, Unit Price)

CardDetails(CustomerID(fk), Card Type, Card Number, Expiry Date, Security Code, First Name, Last Name, Phone, Email, Company Name, Street, Apartment Number, Zip Code)

CustomerProduct(CustomerID(fk), ProductID(fk), Rating, Headline, Review)

ProductQuestionnaire(CustomerID(fk), ProductID(fk), Question, Answer)

ProductFeature(ProductID(fk), Feature)

Cart(CartID, CustomerID(fk), Status)

The composite keys are resulted from multivalued attribute, many-to-many relationships as shown in the ER mapping.

Following the best practice, the composite keys are being replaced by surrogate keys.

The final set of tables are:

Customer(CustomerID, Name, Email, Street, City, Zip Code, State, Password)

Product(ProductID, Name, Price, Discount, Size, Dimensions, Shipping Weight, Model Number, Description, Image, Manufacturer)

ProductFeature(ProductFeatureID, ProductID(fk), Feature)

Cart(CartID, CustomerID(fk), Status)

CartProducts(CartProductsID, CartID(fk), ProductID(fk), Number of Units)

Payment(PaymentID, Status, CartID(fk), CardDetailsID(fk))

Order(OrderID, Order Status, Order Value, Order Date, CustomerID(fk), PaymentID(fk))

OrderDetails(OrderDetailsID, OrderID, ProductID(fk), Number of Units, Unit Price)

CardDetails(CardDetailsID, CustomerID(fk), Card Type, Card Number, Expiry Date, Security Code, First Name, Last Name, Phone, Email, Company Name, Street, Apartment Number, Zip Code)

CustomerProduct(CustomerProductID, CustomerID(fk), ProductID(fk), Rating, Headline, Review)

ProductQuestionnaire(CustomerProductID, CustomerID(fk), ProductID(fk), Question, Answer)

# Physical Design

## Table Structure Definition and SQL Statement

The following represents the structure of the tables we have created including attribute name, type, key, referential constraint, etc. Please note that we have indicated 'ON UPDATE CASCADE' and 'ON DELETE CASCADE' in the table structures but in the SQL statements 'ON UPDATE CASCADE' is not mentioned as we are running our queries on Oracle which does not accept update cascades.

1. Customer Table

   Customer (CustomerID, Name, Email, Street, City, ZipCode, State, Password)

| Attribute Name | Type | Key | Nullable | Referential Constraint |
|---|---|---|---|---|
| CustomerID | Numeric (4) | PRIMARY | NOT NULL | |
| Name | Varchar (30) | | NOT NULL | |
| Email | Varchar (30) | | NOT NULL | |
| Street | Varchar (30) | | NULL | |
| City | Varchar (30) | | NULL | |
| ZipCode | Numeric (6) | | NULL | |
| State | Varchar (20) | | NULL | |
| Password | Varchar (30) | | NOT NULL | |

```
Create TABLE Customer (
CustomerID numeric (4) NOT NULL,
Name varchar (30) NOT NULL,
Email varchar (30) NOT NULL,
Street varchar (30) NULL,
City varchar (30) NULL,
ZipCode numeric (6) NULL,
State varchar (20) NULL,
Password varchar (30) NOT NULL,
CONSTRAINT Customer_Pk PRIMARY KEY (CustomerID)
);
```

2. Product (ProductID, Name, Price, Discount, Size, Dimensions, Shipping Weight, Model Number, Description, Image, Manufacturer)

| Attribute Name | Type | Key | Nullable | Referential Constraint |
|---|---|---|---|---|
| ProductID | Numeric (4) | PRIMARY | NOT NULL | |
| Name | Varchar (30) | | NOT NULL | |
| Price | Numeric (10) | | NOT NULL | |
| Discount | Numeric (10) | | NULL | |
| Size | Varchar (30) | | NOT NULL | |
| Dimensions | Varchar (30) | | NOT NULL | |
| Shipping Weight | Numeric (5) | | NOT NULL | |
| Model Number | Varchar (30) | | NOT NULL | |
| Description | Varchar (100) | | NOT NULL | |
| Image | Varchar (50) | | NOT NULL | |
| Manufacturer | Varchar (30) | | NULL | |

```
Create TABLE Product (
ProductID numeric (4) NOT NULL,
ProductName varchar (100) NOT NULL,
Price_inDollars numeric (10) NOT NULL ,
Discount_inDollars varchar (50) NOT NULL,
ProductSize_inPounds varchar (50)NOT NULL,
Dimensions_inInches varchar (30) NOT NULL,
ShippingWeight_inPounds varchar (50) NOT NULL,
ModelNumber varchar (30) NOT NULL,
Description varchar (1000) NOT NULL,
Image varchar (50) NOT NULL,
Manufacturer varchar (50) NOT NULL,
CONSTRAINT Product5_Pk PRIMARY KEY (ProductID)
);
```

3. ProductFeature (ProductFeatureID, ProductID(fk), Feature)

| Attribute Name | Type | Key | Nullable | Referential Constraint |
|---|---|---|---|---|
| ProductFeature ID | numeric (4) | PRIMARY | NOT NULL | |
| ProductId | numeric (4) | ForeignKey Referencing Product.ProductID | NOT NULL | ON DELETE CASCADE ON UPDATE CASCADE |
| Feature | varchar (500) | | NOT NULL | |

```
CREATE TABLE ProductFeature (
ProductFeatureID numeric (4)    NOT NULL,
Feature          varchar (500)  NOT NULL,
ProductID        numeric (4)    NOT NULL,
CONSTRAINT productfeature_pk PRIMARY KEY (ProductFeatureID),
CONSTRAINT product_fk FOREIGN KEY (ProductID)
REFERENCES Product (ProductID)
ON DELETE CASCADE
);
```

4. Cart (CartID, CustomerID(fk), Status)

| Attribute Name | Type | Key | Nullable | Referential Constraint |
|---|---|---|---|---|
| CartID | numeric (4) | PRIMARY | NOT NULL | |
| CustomerId | numeric (4) | ForeignKey Referencing Customer.CustomerID | NOT NULL | ON DELETE CASCADE ON UPDATE CASCADE |
| Status | varchar (10) | | NOT NULL | |

```
CREATE TABLE Cart (
```

CartID numeric (4) NOT NULL,

Status varchar (10) NOT NULL,

CustomerID numeric (4) NOT NULL,

CONSTRAINT cart_pk PRIMARY KEY (CartID),

CONSTRAINT customer_fk FOREIGN KEY (CustomerID)

REFERENCES Customer (CustomerID)

ON DELETE CASCADE

);

5. CartProducts (CartProductsID, CartID(fk), ProductID(fk), Number of Units)

| Attribute Name | Type | Key | Nullable | Referential Constraint |
|---|---|---|---|---|
| CartProductsID | numeric (4) | PRIMARY | NOT NULL | |
| CartID | numeric (4) | ForeignKey Referencing Cart.CartID | NOT NULL | ON DELETE CASCADE ON UPDATE CASCADE |
| ProductID | numeric (4) | ForeignKey Referencing Product. ProductID | NOT NULL | ON UPDATE CASCADE |
| NumberOfUnits | integer | | NOT NULL | |

CREATE TABLE CartProducts (

CartProductsID numeric (4) NOT NULL,

CartID numeric (4) NOT NULL,

ProductID numeric (4) NOT NULL,

NumberOfUnits integer NOT NULL,

CONSTRAINT cartproductsis_pk PRIMARY KEY (CartProductsID),

CONSTRAINT cartid_fk FOREIGN KEY (CartID)

REFERENCES Cart (CartID),

CONSTRAINT productid_fk FOREIGN KEY (ProductID)

REFERENCES Product (ProductID)

ON DELETE CASCADE

);

6. Payment (PaymentID, Status, CartID(fk), CardDetailId(fk))

| Attribute Name | Type | Key | Nullable | Referential Constraint |
|---|---|---|---|---|
| PaymentID | Numeric (4) | PRIMARY | NOT NULL | |
| Status | Varchar (30) | | NOT NULL | |
| CartID | Numeric (4) | ForeignKey Referencing Cart.CartID | NOT NULL | ON UPDATE CASCADE |
| CardDetailId | Numeric (4) | ForeignKey Referencing CardDetails. CardDetailId | NOT NULL | ON UPDATE CASCADE |

Create TABLE Payment (

PaymentID numeric(4) NOT NULL,

Status varchar(30) NOT NULL,

CartID numeric(4) NOT NULL,

CardDetailId numeric(4) NOT NULL,

CONSTRAINT Payment_pk PRIMARY KEY (PaymentID),

CONSTRAINT Payment_Cart_fk FOREIGN KEY (CartID)

REFERENCES Cart (CartID)

ON DELETE CASCADE,

CONSTRAINT Payment_CardDetails_fk FOREIGN KEY (CardDetailId)

REFERENCES CardDetails (CardDetailId)

ON DELETE CASCADE

);

7. Order(OrderID, Order Status, Order Value, Order Date, CustomerID(fk), PaymentID(fk))

| Attribute Name | Type | Key | Nullable | Referential Constraint |
|---|---|---|---|---|
| OrderID | Numeric (4) | PRIMARY | NOT NULL | |
| Order Status | varchar(30) | | NOT NULL | |
| Order Value | Numeric (10) | | NOT NULL | |
| Order Date | Date | | NOT NULL | Set default to current date time |
| CustomerID | Numeric (4) | ForeignKey Referencing Customer.CustomerID | NOT NULL | ON DELETE CASCADE ON UPDATE CASCADE |
| PaymentID | Numeric (4) | ForeignKey Referencing Payment.PaymentID | NOT NULL | ON DELETE CASCADE ON UPDATE CASCADE |

```
Create TABLE Orders (
OrderID numeric(4) NOT NULL,
CUSTOMERID numeric (4) NOT NULL,
PaymentID numeric (4) NOT NULL,
OrderStatus varchar(30) NOT NULL,
OrderValue numeric(10) NOT NULL,
OrderDate DATE DEFAULT SysDate NOT NULL,
CONSTRAINT Order_pk PRIMARY KEY (OrderID),
CONSTRAINT Order_Customer_fk FOREIGN KEY (CustomerID)
REFERENCES Customer (CustomerID)
ON DELETE CASCADE,
CONSTRAINT Order_Payment_fk FOREIGN KEY (PaymentID)
REFERENCES Payment (PaymentID)
ON DELETE CASCADE
);
```

8. OrderDetails (OrderDetailsID, OrderId,ProductId(fk),NumberOfUnits,Unitprice)

| Attribute Name | Type | Key | Nullable | Referential Constraint |
|---|---|---|---|---|
| OrderDetailsID | numeric (4) | PRIMARY | NOT NULL | |
| OrderId | numeric (4) | ForeignKey Referencing Order.OrderiD | NOT NULL | ON DELETE CASCADE<br>ON UPDATE CASCADE |
| ProductId | Numeric (4) | ForeignKey Referencing Order.OrderiD | NOT NULL | ON UPDATE CASCADE |
| NumberOfUnits | Integer | | NOT NULL | |
| Unitprice | varchar (10) | | NOT NULL | |

```
CREATE TABLE OrderDetails (
OrderDetailsID numeric (4) NOT NULL,
OrderID numeric (4) NOT NULL,
ProductId numeric(4) NOT NULL,
NumberOfUnits integer NOT NULL,
Unitprice  varchar(10) NOT Null,
CONSTRAINT orderdetails_pk PRIMARY KEY (OrderDetailsID),
CONSTRAINT order_orderdetails_fk FOREIGN KEY (OrderID)
REFERENCES Orders(OrderID)
On Delete Cascade,
CONSTRAINT order_productid_fk FOREIGN KEY (ProductID)
REFERENCES Product (ProductID)
);
```

9. CardDetails(CardDetailID,CustomerID(fk),CardType,CardNumber,ExpiryDate,SecurityCode,FirstName,LastName,Phone,Email,CopmanyName,Street,ApartmentNumber,ZipCode)

| Attribute Name | Type | Key | Nullable | Referential Constraint |
|---|---|---|---|---|
| CardDetailID | numeric (4) | PRIMARY | NOT NULL | |
| CustomerID | numeric (4) | ForeignKey Referencing Order.OrderID | NOT NULL | ON DELETE CASCADE ON UPDATE CASCADE |
| CardType | varchar (25) | | NOT NULL | |
| CardNumber | Numeric (24) | | NOT NULL | |
| ExpiryDate | DateTime | | NOT NULL | |
| SecurityCode | Numeric (24) | | NOT NULL | |
| FirstName | varchar (100) | | NOT NULL | |
| LastName | varchar (25) | | NOT NULL | |
| Phone | varchar (100) | | NOT NULL | |
| Email | varchar (100) | | NOT NULL | |
| CompanyName | varchar (100) | | Null | |
| Street | varchar (100) | | NOT NULL | |
| ApartmentNumber | Numeric (24) | | NOT NULL | |
| ZipCode | Numeric (24) | | NOT NULL | |

CREATE TABLE CardDetails (

CardDetailId numeric (4) NOT NULL,

CustomerId numeric (4) NOT NULL,

CardType varchar2(25) NOT NULL,

CardNumber numeric(24) NOT NULL,

ExpiryDate Date NOT Null,

SecurityCode numeric(24) NOT NULL,

FirstName varchar (100) NOT NULL,

LastName varchar (100) NOT NULL,

Phone varchar (25) NOT NULL,

Email varchar (100) NOT NULL,

CompanyName  varchar2 (100) Null,

Street varchar (100) NOT NULL,

ApartmentNumber numeric (24) NOT NULL,

ZipCode numeric(4) NOT NULL,

CONSTRAINT carddetail_pk PRIMARY KEY (CardDetailId),

CONSTRAINT customer_carddetails_fk FOREIGN KEY (CustomerId)

REFERENCES Customer(CustomerId)

ON DELETE CASCADE

);

10. CustomerProduct(CustomerProductID, CustomerID(fk), ProductID(fk), Rating, Headline Review)

| Attribute Name | Type | Key | Nullable | Referential Constraint |
|---|---|---|---|---|
| CustomerProductID | numeric (4) | PRIMARY | NOT NULL | |
| CustomerID | numeric (4) | ForeignKey Referencing Customer.CustomerID | NOT NULL | ON UPDATE CASCADE |
| ProductID | numeric (4) | ForeignKey Referencing Product.ProductID | NOT NULL | ON DELETE CASCADE ON UPDATE CASCADE |
| Rating | numeric (1) | | NULL | |
| Headline | varchar(100) | | NULL | |

| | | | | |
|---|---|---|---|---|
| Review | Varchar(500) | | NULL | |

```
CREATE TABLE CustomerProduct(
CustomerProductID numeric (4) NOT NULL,
CustomerID numeric (4) NOT NULL,
ProductID numeric (4)  NOT NULL,
Rating numeric(1),
Headline varchar(100),
Review varchar(500),
CONSTRAINT customerproduct_pk PRIMARY KEY (CustomerProductID),
CONSTRAINT customer_prod1_fk FOREIGN KEY (CustomerID)
REFERENCES Customer (CustomerID),
CONSTRAINT product1_fk FOREIGN KEY (ProductID)
REFERENCES Product (ProductID)
ON delete CASCADE
);
```

11. ProductQuestionnaire(CustomerProductID, CustomerID(fk), ProductID(fk), Question, Answer)

| Attribute Name | Type | Key | Nullable | Referential Constraint |
|---|---|---|---|---|
| CustomerProductID | numeric (4) | PRIMARY | NOT NULL | |
| CustomerID | numeric (4) | ForeignKey Referencing Customer.CustomerID | NOT NULL | ON UPDATE CASCADE |
| ProductID | numeric (4) | ForeignKey Referencing Product.ProductID | NOT NULL | ON DELETE CASCADE<br>ON UPDATE CASCADE |
| Question | varchar(240) | | NULL | |
| Answer | varchar (240) | | NULL | |

```
CREATE TABLE ProductQuestionnaire (
CustomerProductID numeric (4) NOT NULL,
CustomerID numeric (4) NOT NULL,
ProductID numeric (4)    NOT NULL,
Question varchar(240),
Answer varchar(240),
CONSTRAINT customerproduct2_pk PRIMARY KEY (CustomerProductID),
CONSTRAINT customer2_fk FOREIGN KEY (CustomerID)
REFERENCES Customer (CustomerID),
CONSTRAINT product2_fk FOREIGN KEY (ProductID)
REFERENCES Product (ProductID)
ON DELETE CASCADE
);
```