DATABASE MANAGEMENT SYSTEM

Ecommerce Management DBMS

THAPAR INSTITUTE
OF ENGINEERING & TECHNOLOGY
(Deemed to be University)

MEMBERS

SAM VASISHAT-102116007

NEERAJ SANDHU-102116013

REITIKA KUMAR-102116008

SIDDHI UPADHYAY - 102116024

CONTENTS

- PROJECT DESCRIPTION
- FUNCTIONALITIES
- ENTITY RELATIONSHIP DIAGRAM AND CONSTRAINTS
- CREATING TABLES AND RELATIONS (ER TO TABLES)
- INSERTING RECORDS
- NORMALIZATION
- BASIC QUERIES
- PL/SQL FUNCTIONS

PROJECT DESCRIPTION

An eCommerce management system is a powerful tool that allows businesses to conduct online transactions, manage inventory, track sales, and interact with customers in a streamlined and efficient way. With the rise of online shopping and the increasing importance of eCommerce in today's digital landscape, a well-designed eCommerce management system can be a critical component of a successful business strategy.

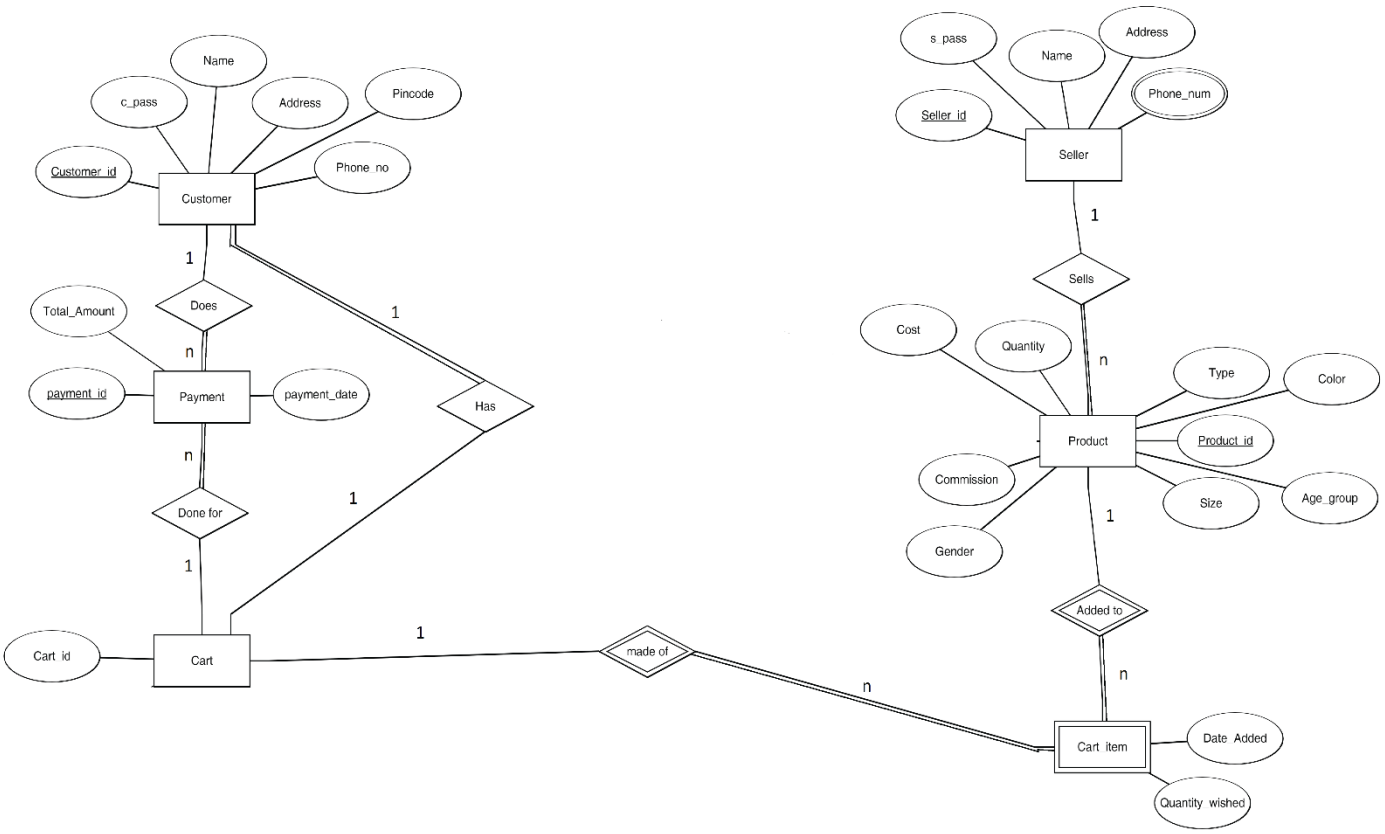
One of the key benefits of an eCommerce management system is its ability to automate many of the processes involved in online transactions. This includes managing customer data, processing payments, and tracking inventory levels. By automating these processes, businesses can save time and reduce errors, while also providing a seamless and hassle-free shopping experience for their customers.

Another important feature of an eCommerce management system is its ability to provide real-time data and analytics about customer behavior, sales trends, and inventory levels. This information can be invaluable for businesses looking to optimize their sales strategies, make data-driven decisions, and stay ahead of the competition.

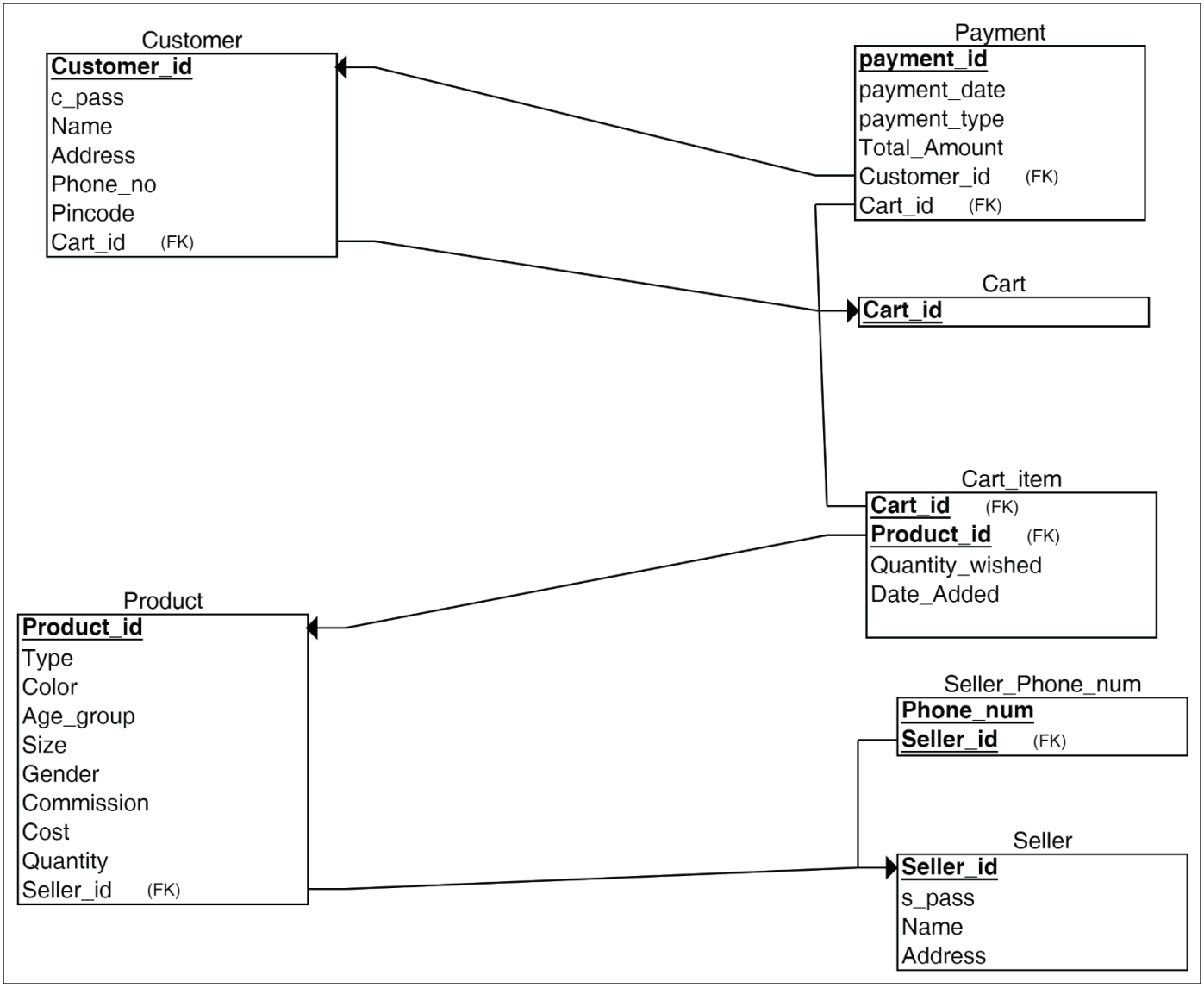
FUNCTIONALITIES

- User registration and login for customers and sellers
- Customers can view and add items to their cart
- Sellers can add, edit, and delete products from their inventory
- Customers can view product details and place orders
- Orders are tracked and managed through the payment system
- Payment system accepts various payment methods
- Sellers can manage their inventory and view sales reports
- Customers can view order history and track the status of their orders
- Customer data, seller data, and product data are stored in respective databasetables
- Data is accessed and managed using SQL queries
- Foreign key constraints are used to ensure data integrity and consistency
- Cursors are not used in this project, but could potentially be added for morecomplex data manipulation tasks.

Entity Relationship diagram



RELATIONAL DATABASE SCHEMA

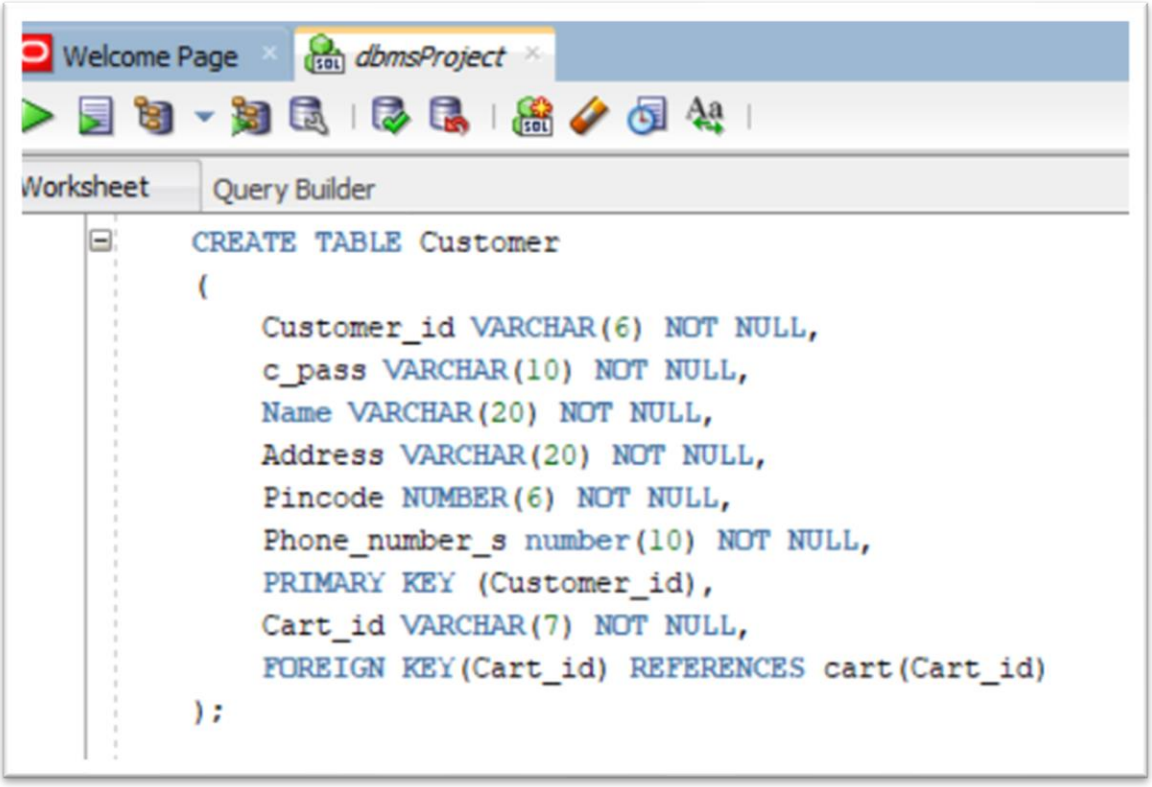


CREATING TABLES AND RELATIONS

[ER TO TABLES]

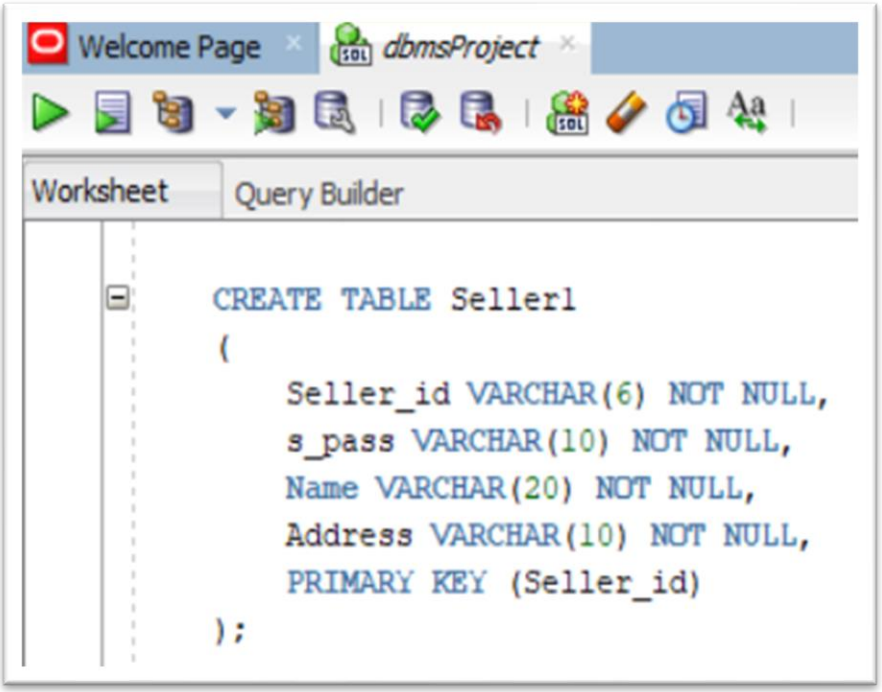
1. Customer Table:

```
CREATE TABLE Customer
  (
    Customer_id VARCHAR(6) NOT NULL,
    c_pass VARCHAR(10) NOT NULL,
    Name VARCHAR(20) NOT NULL,
    Address VARCHAR(20) NOT NULL,
    Pincode NUMBER(6) NOT NULL,
    Phone_number_s number(10) NOT NULL,
    PRIMARY KEY (Customer_id),
    Cart_id VARCHAR(7) NOT NULL,
    FOREIGN KEY(Cart_id) REFERENCES cart(Cart_id)
  );
```



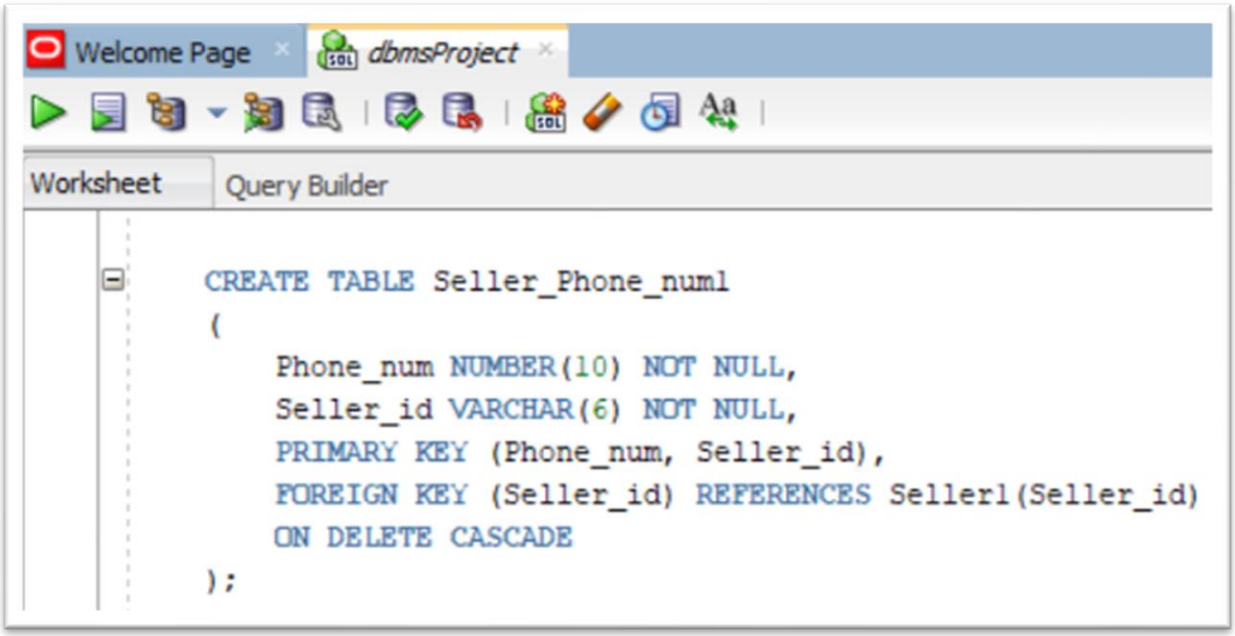
2. Seller Table:

```
CREATE TABLE Seller
    (
        Seller_id VARCHAR(6) NOT NULL,
        s_pass VARCHAR(10) NOT NULL,
        Name VARCHAR(20) NOT NULL,
        Address VARCHAR(10) NOT NULL,
        PRIMARY KEY (Seller_id)
    );
```



3. Seller Phone Number Table:

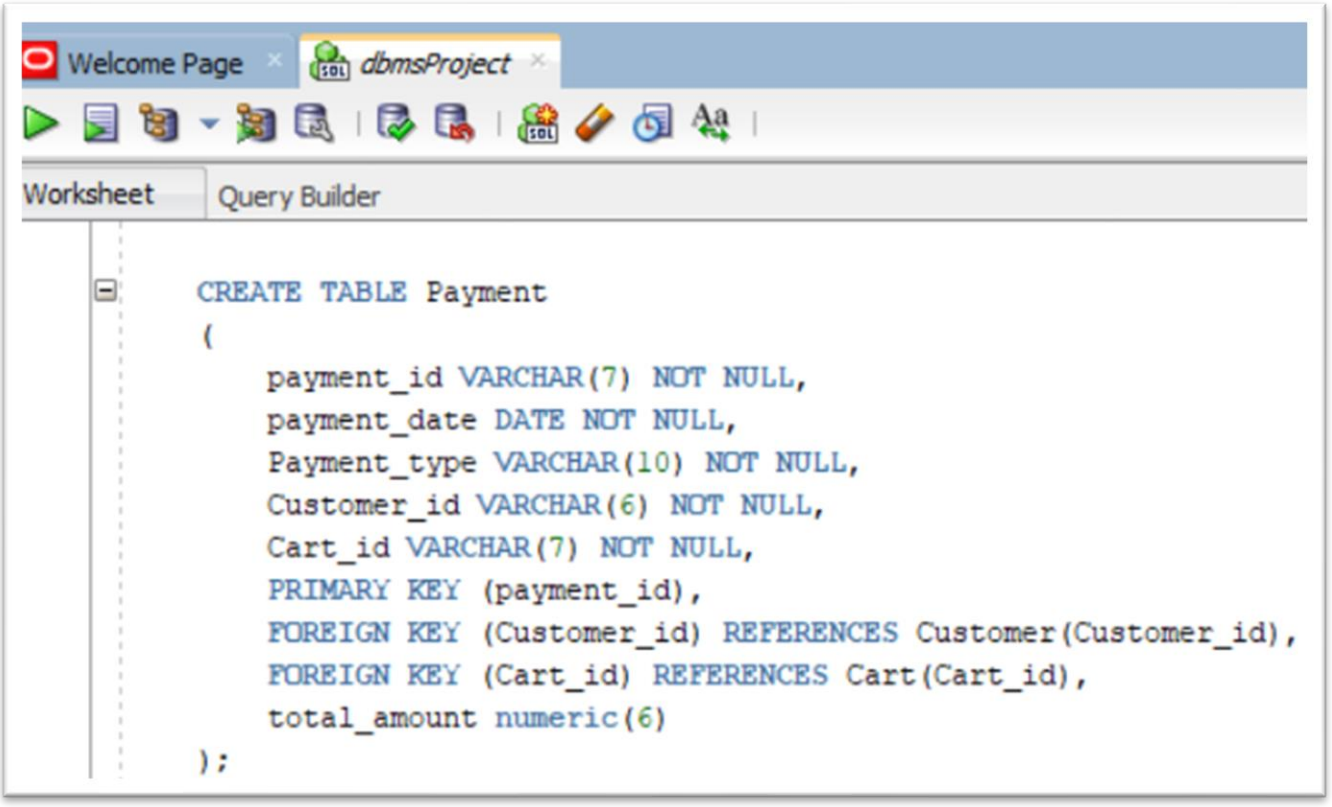
```
CREATE TABLE Seller_Phone_num
    (
        Phone_num NUMBER(10) NOT NULL,
        Seller_id VARCHAR(6) NOT NULL,
        PRIMARY KEY (Phone_num, Seller_id),
        FOREIGN KEY (Seller_id) REFERENCES Seller(Seller_id)
        ON DELETE CASCADE
    );
```



---

4. Payment Table:

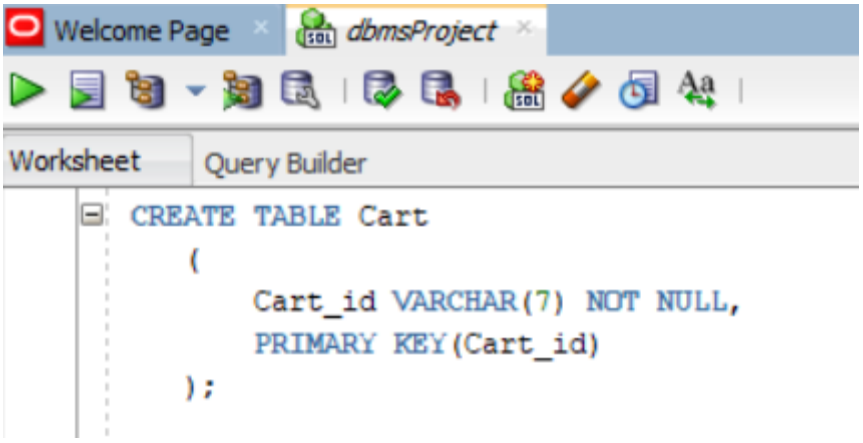
```
CREATE TABLE Payment
    (
        payment_id VARCHAR(7) NOT NULL,
        payment_date DATE NOT NULL,
        Payment_type VARCHAR(10) NOT NULL,
        Customer_id VARCHAR(6) NOT NULL,
        Cart_id VARCHAR(7) NOT NULL,
        PRIMARY KEY (payment_id),
        FOREIGN KEY (Customer_id) REFERENCES Customer(Customer_id),
        FOREIGN KEY (Cart_id) REFERENCES Cart(Cart_id),
        total_amount numeric(6)
    );
```



5. Cart Table:

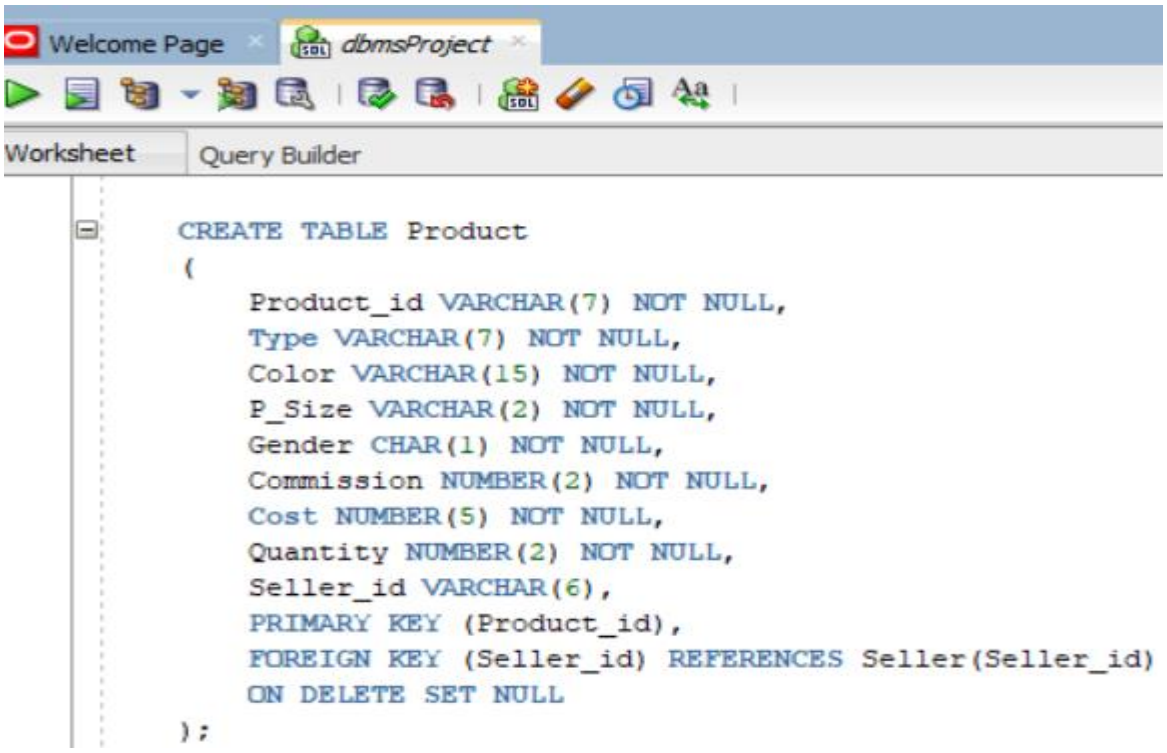
CREATE TABLE Cart

```
(
    Cart_id VARCHAR(7) NOT NULL,
    PRIMARY KEY(Cart_id)
)
```



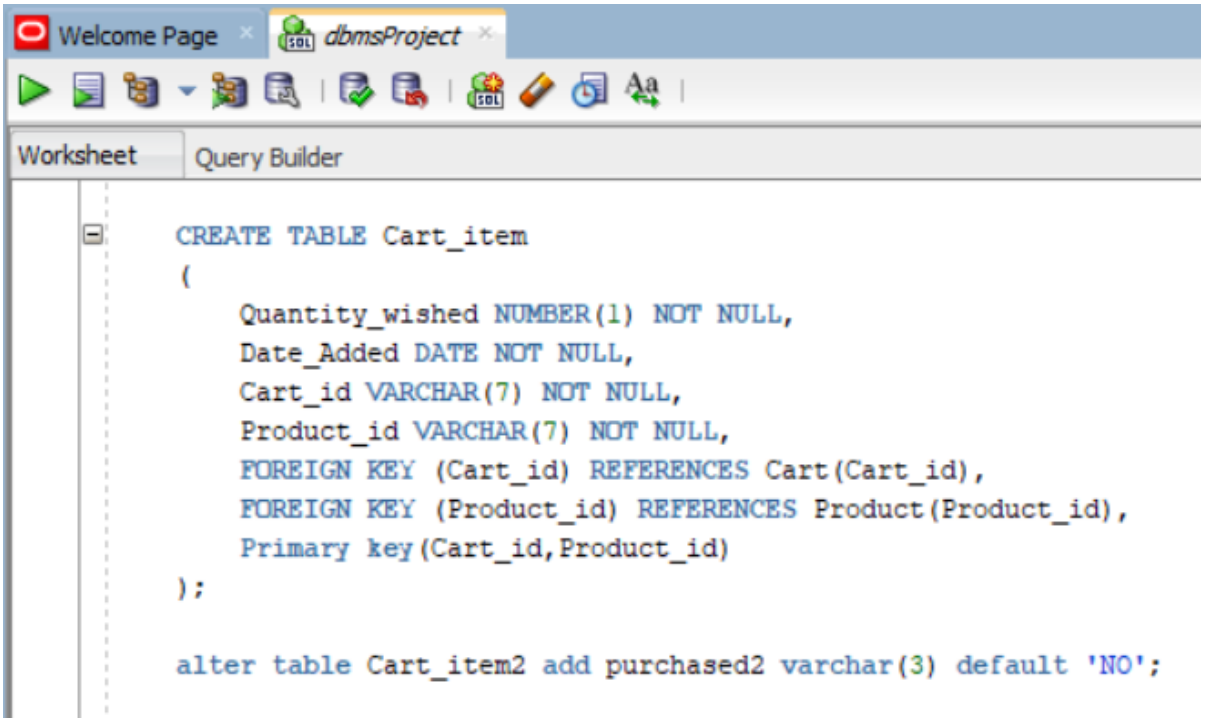
6. Product Table:

```
CREATE TABLE Product
  (
    Product_id VARCHAR(7) NOT NULL,
    Type VARCHAR(7) NOT NULL,
    Color VARCHAR(15) NOT NULL,
    P_Size VARCHAR(2) NOT NULL,
    Gender CHAR(1) NOT NULL,
    Commission NUMBER(2) NOT NULL,
    Cost NUMBER(5) NOT NULL,
    Quantity NUMBER(2) NOT NULL,
    Seller_id VARCHAR(6),
    PRIMARY KEY (Product_id),
    FOREIGN KEY (Seller_id) REFERENCES Seller(Seller_id)
    ON DELETE SET NULL
  );
```



7.Cart Item Table:

```
CREATE TABLE Cart_item
  (
    Quantity_wished NUMBER(1) NOT NULL,
    Date_Added DATE NOT NULL,
    Cart_id VARCHAR(7) NOT NULL,
    Product_id VARCHAR(7) NOT NULL,
    FOREIGN KEY (Cart_id) REFERENCES Cart(Cart_id),
    FOREIGN KEY (Product_id) REFERENCES Product(Product_id),
    Primary key(Cart_id,Product_id)
  );
```



Inserting values into Table:

1.Customer Table:

INSERT INTO Customer (Customer_id, c_pass, Name, Address, Pincode, Phone_number_s, Cart_id)
VALUES ('CUS001', 'password', 'John Doe', '123 Main Street', '123456', '1234567890', 'CART001');
INSERT INTO Customer (Customer_id, c_pass, Name, Address, Pincode, Phone_number_s, Cart_id)
VALUES ('CUS002', 'password', 'Jane Doe', '456 Park Avenue', '789012', '0987654321', 'CART002');
INSERT INTO Customer (Customer_id, c_pass, Name, Address, Pincode, Phone_number_s, Cart_id)
VALUES ('CUS003', 'password', 'Bob Smith', '789 Elm Street', '345678', '4567890123', 'CART003');
INSERT INTO Customer (Customer_id, c_pass, Name, Address, Pincode, Phone_number_s, Cart_id)
VALUES ('CUS004', 'password', 'Alice Johnson', '234 Oak Road', '901234', '7890123456', 'CART004');
INSERT INTO Customer (Customer_id, c_pass, Name, Address, Pincode, Phone_number_s, Cart_id)
VALUES ('CUS005', 'password', 'Sarah Wilson', '567 Pine Street', '567890', '2345678901', 'CART005');
INSERT INTO Customer (Customer_id, c_pass, Name, Address, Pincode, Phone_number_s, Cart_id)
VALUES ('CUS006', 'password', 'Mike Davis', '890 Cedar Lane', '123789', '9012345678', 'CART006');
INSERT INTO Customer (Customer_id, c_pass, Name, Address, Pincode, Phone_number_s, Cart_id)
VALUES ('CUS007', 'password', 'Jessica Brown', '1234 Spruce Street', '890123', '3456789012', 'CART007');
INSERT INTO Customer (Customer_id, c_pass, Name, Address, Pincode, Phone_number_s, Cart_id)
VALUES ('CUS008', 'password', 'David Wilson', '5678 Oak Lane', '456789', '6789012345', 'CART008');
INSERT INTO Customer (Customer_id, c_pass, Name, Address, Pincode, Phone_number_s, Cart_id)
VALUES ('CUS009', 'password', 'Emily Johnson', '9012 Maple Road', '123456', '3456789012', 'CART009');

Script Output × | Query Result ×

SQL | All Rows Fetched: 9 in 0.002 seconds

	CUSTOMER_ID	C_PASS	NAME	ADDRESS	PINCODE	PHONE_NUMBER_S	CART_ID
1	CUS001	password	John Doe	123 Main Street	123456	1234567890	CART001
2	CUS002	password	Jane Doe	456 Park Avenue	789012	987654321	CART002
3	CUS003	password	Bob Smith	789 Elm Street	345678	4567890123	CART003
4	CUS004	password	Alice Johnson	234 Oak Road	901234	7890123456	CART004
5	CUS005	password	Sarah Wilson	567 Pine Street	567890	2345678901	CART005
6	CUS006	password	Mike Davis	890 Cedar Lane	123789	9012345678	CART006
7	CUS007	password	Jessica Brown	1234 Spruce Street	890123	3456789012	CART007
8	CUS008	password	David Wilson	5678 Oak Lane	456789	6789012345	CART008
9	CUS009	password	Emily Johnson	9012 Maple Road	123456	3456789012	CART009

2. Seller Table:

INSERT INTO Seller1 (Seller_id, s_pass, Name, Address) VALUES ('S10001', 'sellerpass1', 'John Doe', '123 Main St.');
INSERT INTO Seller1 (Seller_id, s_pass, Name, Address) VALUES ('S10002', 'sellerpass2', 'Jane Smith', '456 Oak St.');
INSERT INTO Seller1 (Seller_id, s_pass, Name, Address) VALUES ('S10003', 'sellerpass3', 'Bob Johnson', '789 Maple Ave.');
INSERT INTO Seller1 (Seller_id, s_pass, Name, Address) VALUES ('S10004', 'sellerpass4', 'Sarah Lee', '567 Pine St.');
INSERT INTO Seller1 (Seller_id, s_pass, Name, Address) VALUES ('S10005', 'sellerpass5', 'Michael Jordan', '321 Elm St.');
INSERT INTO Seller1 (Seller_id, s_pass, Name, Address) VALUES ('S10006', 'sellerpass6', 'Taylor Swift', '999 Broadway');
INSERT INTO Seller1 (Seller_id, s_pass, Name, Address) VALUES ('S10007', 'sellerpass7', 'Elon Musk', '111 Rocket Rd.');
INSERT INTO Seller1 (Seller_id, s_pass, Name, Address) VALUES ('S10008', 'sellerpass8', 'Bill Gates', '456 Microsoft Way');
INSERT INTO Seller1 (Seller_id, s_pass, Name, Address) VALUES ('S10009', 'sellerpass9', 'Jeff Bezos', '789 Amazon St.');

Script Output × Query Result ×

SQL | All Rows Fetched: 18 in 0.004 seconds

	SELLER_ID	S_PASS	NAME	ADDRESS
1	S10001	sellerpass1	John Doe	123-Main
2	S10002	sellerpass2	Jane Smith	456-Oak
3	S10003	sellerpass3	Bob Johnson	789-Maple
4	S10004	sellerpass4	Sarah Lee	567-Pine
5	S10005	sellerpass5	Michael Jordan	321-Elm
6	S10009	sellerpass9	Jeff Bezos	789 Amazon St.
7	S10006	sellerpass6	Taylor Swift	999 Broadway
8	S10007	sellerpass7	Elon Musk	111 Rocket Rd.
9	S10008	sellerpass8	Bill Gates	456 Microsoft Way
10	S10011	sellerpass1	John Doe	123 Main St.
11	S10012	sellerpass2	Jane Smith	456 Oak St.
12	S10013	sellerpass3	Bob Johnson	789 Maple Ave.
13	S10014	sellerpass4	Sarah Lee	567 Pine St.
14	S10015	sellerpass5	Michael Jordan	321 Elm St.
15	S10016	sellerpass6	Taylor Swift	999 Broadway
16	S10017	sellerpass7	Elon Musk	111 Rocket Rd.
17	S10018	sellerpass8	Bill Gates	456 Microsoft Way
18	S10019	sellerpass9	Jeff Bezos	789 Amazon St.

3. Seller phone num:

INSERT INTO Seller_Phone_num1 (Phone_num, Seller_id) VALUES (1234567890, 'S10001');
INSERT INTO Seller_Phone_num1 (Phone_num, Seller_id) VALUES (2345678901, 'S10002');
INSERT INTO Seller_Phone_num1 (Phone_num, Seller_id) VALUES (3456789012, 'S10003');
INSERT INTO Seller_Phone_num1 (Phone_num, Seller_id) VALUES (4567890123, 'S10004');
INSERT INTO Seller_Phone_num1 (Phone_num, Seller_id) VALUES (5678901234, 'S10005');
INSERT INTO Seller_Phone_num1 (Phone_num, Seller_id) VALUES (6789012345, 'S10006');
INSERT INTO Seller_Phone_num1 (Phone_num, Seller_id) VALUES (7890123456, 'S10007');
INSERT INTO Seller_Phone_num1 (Phone_num, Seller_id) VALUES (8901234567,

'S10008');
INSERT INTO Seller_Phone_num1 (Phone_num, Seller_id) VALUES (9012345678, 'S10009');

Script Output × Query Result ×

SQL | All Rows Fetched: 9 in 0.003 seconds

	PHONE_NUM	SELLER_ID
1	1234567890	S10011
2	2345678901	S10012
3	3456789012	S10013
4	4567890123	S10014
5	5678901234	S10015
6	6789012345	S10016
7	7890123456	S10017
8	8901234567	S10018
9	9012345678	S10019

4.Payments Table:

INSERT INTO Payment0 (payment_id, payment_date, Payment_type, Customer_id, Cart_id, total_amount) VALUES ('PAY001', '2023-04-01', 'Credit', 'CUS001', 'CART001', 50.00);
INSERT INTO Payment0 (payment_id, payment_date, Payment_type, Customer_id, Cart_id, total_amount) VALUES ('PAY002', '2023-04-02', 'Debit', 'CUS002', 'CART002', 100.00);
INSERT INTO Payment0 (payment_id, payment_date, Payment_type, Customer_id, Cart_id, total_amount) VALUES ('PAY003', '2023-04-03', 'Cash', 'CUS003', 'CART003', 75.00);
INSERT INTO Payment0 (payment_id, payment_date, Payment_type, Customer_id, Cart_id, total_amount) VALUES ('PAY004', '2023-04-04', 'Credit', 'CUS004', 'CART004', 25.00);
INSERT INTO Payment0 (payment_id, payment_date, Payment_type, Customer_id, Cart_id, total_amount) VALUES ('PAY005', '2023-04-05', 'Debit', 'CUS005', 'CART005', 200.00);

INSERT INTO Payment0 (payment_id, payment_date, Payment_type, Customer_id, Cart_id, total_amount) VALUES ('PAY006', '2023-04-06', 'Cash', 'CUS006', 'CART006', 150.00);
INSERT INTO Payment0 (payment_id, payment_date, Payment_type, Customer_id, Cart_id, total_amount) VALUES ('PAY007', '2023-04-07', 'Credit', 'CUS007', 'CART007', 75.00);

INSERT INTO Payment0 (payment_id, payment_date, Payment_type, Customer_id, Cart_id, total_amount) VALUES ('PAY008', '2023-04-08', 'Debit', 'CUS008', 'CART008', 50.00);
INSERT INTO Payment0 (payment_id, payment_date, Payment_type, Customer_id, Cart_id, total_amount) VALUES ('PAY009', '2023-04-09', 'Cash', 'CUS001', 'CART009', 125.00);

Script Output | Query Result

SQL | All Rows Fetched: 9 in 0.003 seconds

	PAYMENT_ID	PAYMENT_DATE	PAYMENT_TYPE	CUSTOMER_ID	CART_ID	TOTAL_AMOUNT
1	PAY001	25-APR-23	Credit	CUS001	CART001	50
2	PAY002	02-APR-23	Debit	CUS002	CART002	100
3	PAY003	03-APR-23	Cash	CUS003	CART003	75
4	PAY004	04-APR-23	Credit	CUS004	CART004	25
5	PAY005	05-APR-23	Debit	CUS005	CART005	200
6	PAY006	06-APR-23	Cash	CUS006	CART006	150
7	PAY007	07-APR-23	Credit	CUS007	CART007	75
8	PAY008	08-APR-23	Debit	CUS008	CART008	50
9	PAY009	09-APR-23	Cash	CUS001	CART009	125

5. Cart Table:

INSERT INTO Cart (Cart_id) VALUES ('CART001');
INSERT INTO Cart (Cart_id) VALUES ('CART002');
INSERT INTO Cart (Cart_id) VALUES ('CART003');
INSERT INTO Cart (Cart_id) VALUES ('CART004');
INSERT INTO Cart (Cart_id) VALUES ('CART005');
INSERT INTO Cart (Cart_id) VALUES ('CART006');
INSERT INTO Cart (Cart_id) VALUES ('CART007');
INSERT INTO Cart (Cart_id) VALUES ('CART008');
INSERT INTO Cart (Cart_id) VALUES (‘CART009’);

Script Output × Query Result ×

SQL | All Rows Fetched: 11 in 0.004 seconds

	CART_ID
1	crtl011
2	crtl000
3	CART001
4	CART002
5	CART003
6	CART004
7	CART005
8	CART006
9	CART007
10	CART008
11	CART009

6. Product table:

INSERT INTO Product (Product_id, Type, Color, P_Size, Gender, Commission, Cost, Quantity, Seller_id)
VALUES ('PROD001', 'Shirt', 'Blue', 'Medium', 'Male', 10.00, 25.00, 50, 'SELL001');

INSERT INTO Product (Product_id, Type, Color, P_Size, Gender, Commission, Cost, Quantity, Seller_id)
VALUES ('PROD002', 'Dress', 'Black', 'Small', 'Female', 15.00, 50.00, 25, 'SELL002');

INSERT INTO Product (Product_id, Type, Color, P_Size, Gender, Commission, Cost, Quantity, Seller_id)
VALUES ('PROD003', 'Shoes', 'Red', 'Large', 'Unisex', 5.00, 75.00, 10, 'SELL003');

INSERT INTO Product (Product_id, Type, Color, P_Size, Gender, Commission, Cost, Quantity, Seller_id)
VALUES ('PROD004', 'Jeans', 'Gray', 'Medium', 'Male', 12.50, 40.00, 30, 'SELL002');

INSERT INTO Product (Product_id, Type, Color, P_Size, Gender, Commission, Cost, Quantity, Seller_id)
VALUES ('PROD005', 'T-Shirt', 'Green', 'Large', 'Male', 7.50, 20.00, 100, 'SELL001');

INSERT INTO Product (Product_id, Type, Color, P_Size, Gender, Commission, Cost,

Quantity, Seller_id)

VALUES ('PROD006', 'Sweater', 'White', 'Small', 'Female', 10.00, 60.00, 15, 'SELL004');

INSERT INTO Product (Product_id, Type, Color, P_Size, Gender, Commission, Cost, Quantity, Seller_id)
VALUES ('PROD007', 'Skirt', 'Yellow', 'Medium', 'Female', 20.00, 35.00, 20, 'SELL003');

INSERT INTO Product (Product_id, Type, Color, P_Size, Gender, Commission, Cost, Quantity, Seller_id)
VALUES ('PROD008', 'Jacket', 'Black', 'Large', 'Unisex', 15.00, 100.00, 5, 'SELL005');

INSERT INTO Product (Product_id, Type, Color, P_Size, Gender, Commission, Cost, Quantity, Seller_id)
VALUES ('PROD009', 'Pants', 'Brown', 'Small', 'Male', 7.50, 45.00, 20, 'SELL006');

Script Output × Query Result ×

SQL | All Rows Fetched: 8 in 0.004 seconds

	PRODUCT_ID	TYPE	COLOR	P_SIZE	GENDER	COMMISSION	COST	QUANTITY	SELLER_ID
1	PROD001	Shirt	Blue	Medium	Male	10	25	50	S10011
2	PROD002	Dress	Black	Small	Female	15	50	25	S10012
3	PROD003	Shoes	Red	Large	Unisex	5	75	10	S10013
4	PROD004	Jeans	Gray	Medium	Male	13	40	30	S10012
5	PROD006	Sweater	White	Small	Female	10	60	15	S10014
6	PROD007	Skirt	Yellow	Medium	Female	20	35	20	S10013
7	PROD008	Jacket	Black	Large	Unisex	15	100	5	S10015
8	PROD009	Pants	Brown	Small	Male	8	45	20	S10016

7.Cart_item table:

INSERT INTO Cart_item0 (Quantity_wished, Date_Added, Cart_id, Product_id) VALUES (1, '2023-04-25', 'CART001', 'PROD001');
INSERT INTO Cart_item0 (Quantity_wished, Date_Added, Cart_id, Product_id) VALUES (2, '2023-04-24', 'CART001', 'PROD002');
INSERT INTO Cart_item0 (Quantity_wished, Date_Added, Cart_id, Product_id) VALUES (1, '2023-04-23', 'CART002', 'PROD003');
INSERT INTO Cart_item0 (Quantity_wished, Date_Added, Cart_id, Product_id) VALUES (3, '2023-04-22', 'CART002', 'PROD004');
INSERT INTO Cart_item0 (Quantity_wished, Date_Added, Cart_id, Product_id) VALUES (2, '2023-04-21', 'CART003', 'PROD005');
INSERT INTO Cart_item0 (Quantity_wished, Date_Added, Cart_id, Product_id) VALUES (1, '2023-04-20', 'CART003', 'PROD006');
INSERT INTO Cart_item0 (Quantity_wished, Date_Added, Cart_id, Product_id) VALUES (1, '2023-04-19', 'CART004', 'PROD007');
INSERT INTO Cart_item0 (Quantity_wished, Date_Added, Cart_id, Product_id) VALUES (2, '2023-04-18', 'CART004', 'PROD008');
INSERT INTO Cart_item0 (Quantity_wished, Date_Added, Cart_id, Product_id) VALUES (1, '2023-04-17', 'CART004', 'PROD009');

Script Output × Query Result ×

SQL | All Rows Fetched: 16 in 0.003 seconds

	QUANTITY_WISHED	DATE_ADDED	CART_ID	PRODUCT_ID
1	1	25-APR-23	CART001	PROD001
2	2	24-APR-23	CART001	PROD002
3	1	23-APR-23	CART002	PROD003
4	3	22-APR-23	CART002	PROD004
5	1	20-APR-23	CART003	PROD006
6	1	19-APR-23	CART004	PROD007
7	2	18-APR-23	CART004	PROD008
8	1	25-APR-23	CART003	PROD001
9	2	24-APR-23	CART003	PROD003
10	1	23-APR-23	CART004	PROD002
11	3	22-APR-23	CART005	PROD004
12	2	21-APR-23	CART001	PROD003
13	1	20-APR-23	CART006	PROD006
14	1	19-APR-23	CART008	PROD007
15	2	18-APR-23	CART009	PROD008
16	1	17-APR-23	CART007	PROD009

Normalisation of tables:

Customer table:

Script Output × Query Result ×

SQL | All Rows Fetched: 8 in 0.004 seconds

	PRODUCT_ID	TYPE	COLOR	P_SIZE	GENDER	COMMISSION	COST	QUANTITY	SELLER_ID
1	PROD001	Shirt	Blue	Medium	Male	10	25	50	S10011
2	PROD002	Dress	Black	Small	Female	15	50	25	S10012
3	PROD003	Shoes	Red	Large	Unisex	5	75	10	S10013
4	PROD004	Jeans	Gray	Medium	Male	13	40	30	S10012
5	PROD006	Sweater	White	Small	Female	10	60	15	S10014
6	PROD007	Skirt	Yellow	Medium	Female	20	35	20	S10013
7	PROD008	Jacket	Black	Large	Unisex	15	100	5	S10015
8	PROD009	Pants	Brown	Small	Male	8	45	20	S10016

1NF: A relation R is in first normal form (1NF) if and only if all underlying domains contain atomic values only. Since the CUSTOMER table contains only single value in each cell, therefore it is in 1NF.

 2NF: A relation R is in second normal form (2NF) if and only if it is in 1NF and every non-primary key attribute is fully dependent on the primary key. Since the above table is in 1NF and contains only a single primary key i.e. Customer_id and there is no partial dependency, therefore it is in 2NF.

 3NF: A relation R is in third normal form (3NF) if and only if it is in 2NF and every non-key attribute is non-transitively dependent on the primary key. Since the above table does not have any transitive dependencies, therefore it is in 3NF.

Seller and Phone_num:

	PHONE_NUM	SELLER_ID
1	1234567890	S10011
2	2345678901	S10012
3	3456789012	S10013
4	4567890123	S10014
5	5678901234	S10015
6	6789012345	S10016
7	7890123456	S10017
8	8901234567	S10018
9	9012345678	S10019

Script Output × Query Result ×

SQL | All Rows Fetched: 18 in 0.004 seconds

	SELLER_ID	S_PASS	NAME	ADDRESS
1	S10001	sellerpass1	John Doe	123-Main
2	S10002	sellerpass2	Jane Smith	456-Oak
3	S10003	sellerpass3	Bob Johnson	789-Maple
4	S10004	sellerpass4	Sarah Lee	567-Pine
5	S10005	sellerpass5	Michael Jordan	321-Elm
6	S10009	sellerpass9	Jeff Bezos	789 Amazon St.
7	S10006	sellerpass6	Taylor Swift	999 Broadway
8	S10007	sellerpass7	Elon Musk	111 Rocket Rd.
9	S10008	sellerpass8	Bill Gates	456 Microsoft Way
10	S10011	sellerpass1	John Doe	123 Main St.
11	S10012	sellerpass2	Jane Smith	456 Oak St.
12	S10013	sellerpass3	Bob Johnson	789 Maple Ave.
13	S10014	sellerpass4	Sarah Lee	567 Pine St.
14	S10015	sellerpass5	Michael Jordan	321 Elm St.
15	S10016	sellerpass6	Taylor Swift	999 Broadway
16	S10017	sellerpass7	Elon Musk	111 Rocket Rd.
17	S10018	sellerpass8	Bill Gates	456 Microsoft Way
18	S10019	sellerpass9	Jeff Bezos	789 Amazon St.

1NF: A relation R is in first normal form (1NF) if and only if all underlying domains contain atomic values only. Since the Seller table contains only single value in each cell, therefore it is in 1NF.

 2NF: A relation R is in second normal form (2NF) if and only if it is in 1NF and every non-primary key attribute is fully dependent on the primary key. Since the above table is in 1NF and contains only a single primary key and there is no partial dependency, therefore it is in 2NF.

 3NF: A relation R is in third normal form (3NF) if and only if it is in 2NF and every non-key attribute is non-transitively dependent on the primary key. Since the above table does not have any transitive dependencies, therefore it is in 3NF.

Note:
we have broken down our seller table into two tables: Seller_id and Phone_num. if we hadn't broken down the table, then original table would have had columns: Seller_id, s_pass, Name, Address, Phone_num. but we could insert two values of phone number in the table i. e. insert into Seller values('sid1001', '457676', 'rajat','block-b78', '8796687567 9786778766');
could have been inserted but this results in multivalued attribute at Phone_num. therefore, seperate table has been created containing column values as 'Seller_id', 'phone_num' and Seller_id is a foreign key that references the seller_id in table: Seller.

Cart_item:

	QUANTITY_WISHED	DATE_ADDED	CART_ID	PRODUCT_ID
1	1	25-APR-23	CART001	PROD001
2	2	24-APR-23	CART001	PROD002
3	1	23-APR-23	CART002	PROD003
4	3	22-APR-23	CART002	PROD004
5	1	20-APR-23	CART003	PROD006
6	1	19-APR-23	CART004	PROD007
7	2	18-APR-23	CART004	PROD008
8	1	25-APR-23	CART003	PROD001
9	2	24-APR-23	CART003	PROD003
10	1	23-APR-23	CART004	PROD002
11	3	22-APR-23	CART005	PROD004
12	2	21-APR-23	CART001	PROD003
13	1	20-APR-23	CART006	PROD006
14	1	19-APR-23	CART008	PROD007
15	2	18-APR-23	CART009	PROD008
16	1	17-APR-23	CART007	PROD009

1NF: A relation R is in first normal form (1NF) if and only if all underlying domains contain atomic values only. Since the Cart_item table contains only single value in each cell, therefore it is in 1NF.

 2NF: A relation R is in second normal form (2NF) if and only if it is in 1NF and every non-primary key attribute is fully dependent on the primary key. The above table is in 1NF. The primary key of the table is a combination of Cart_id+Product_id. Now, for the table to be in 2NF, other columns should be separately dependent of the combination of the Cart_id+Product_id, which is exactly what is happening in out table. Therfore it is in 2NF.

 3NF: A relation R is in third normal form (3NF) if and only if it is in 2NF and every non-key attribute is non-transitively dependent on the primary key. Since the above table does not have any transitive dependencies, therefore it is in 3NF.

Product:

Script Output x | Query Result x

SQL | All Rows Fetched: 8 in 0.004 seconds

	PRODUCT_ID	TYPE	COLOR	P_SIZE	GENDER	COMMISSION	COST	QUANTITY	SELLER_ID
1	PROD001	Shirt	Blue	Medium	Male	10	25	50	S10011
2	PROD002	Dress	Black	Small	Female	15	50	25	S10012
3	PROD003	Shoes	Red	Large	Unisex	5	75	10	S10013
4	PROD004	Jeans	Gray	Medium	Male	13	40	30	S10012
5	PROD006	Sweater	White	Small	Female	10	60	15	S10014
6	PROD007	Skirt	Yellow	Medium	Female	20	35	20	S10013
7	PROD008	Jacket	Black	Large	Unisex	15	100	5	S10015
8	PROD009	Pants	Brown	Small	Male	8	45	20	S10016

1NF: A relation R is in first normal form (1NF) if and only if all underlying domains contain atomic values only. Since the Product table contains only single value in each cell, therefore it is in 1NF.

 2NF: A relation R is in second normal form (2NF) if and only if it is in 1NF and every non-primary key attribute is fully dependent on the primary key. Since the above table is in 1NF and contains only a single primary key and there is no partial dependency, therefore it is in 2NF.

 3NF: A relation R is in third normal form (3NF) if and only if it is in 2NF and every non-key attribute is non-transitively dependent on the primary key. Since the above table does not have any transitive dependencies, therefore it is in 3NF.

Cart:

	CART_ID
1	crt1011
2	crt1000
3	CART001
4	CART002
5	CART003
6	CART004
7	CART005
8	CART006
9	CART007
10	CART008
11	CART009

1NF: A relation R is in first normal form (1NF) if and only if all underlying domains contain atomic values only. Since the Product table contains only single value in each cell, therefore it is in 1NF.

 2NF: A relation R is in second normal form (2NF) if and only if it is in 1NF and every non-primary key attribute is fully dependent on the primary key. Since the above table is in 1NF and contains only a single primary key and there is no partial dependency, therefore it is in 2NF.

 3NF: A relation R is in third normal form (3NF) if and only if it is in 2NF and every non-key attribute is non-transitively dependent on the primary key. Since the above table does not have any transitive dependencies, therefore it is in 3NF.

Payment:

	PAYMENT_ID	PAYMENT_DATE	PAYMENT_TYPE	CUSTOMER_ID	CART_ID	TOTAL_AMOUNT
1	PAY001	25-APR-23	Credit	CUS001	CART001	50
2	PAY002	02-APR-23	Debit	CUS002	CART002	100
3	PAY003	03-APR-23	Cash	CUS003	CART003	75
4	PAY004	04-APR-23	Credit	CUS004	CART004	25
5	PAY005	05-APR-23	Debit	CUS005	CART005	200
6	PAY006	06-APR-23	Cash	CUS006	CART006	150
7	PAY007	07-APR-23	Credit	CUS007	CART007	75
8	PAY008	08-APR-23	Debit	CUS008	CART008	50
9	PAY009	09-APR-23	Cash	CUS001	CART009	125

1NF: A relation R is in first normal form (1NF) if and only if all underlying domains contain atomic values only. Since the Payment table contains only single value in each cell, therefore it is in 1NF.

 2NF: A relation R is in second normal form (2NF) if and only if it is in 1NF and every non-primary key attribute is fully dependent on the primary key. Since the above table is in 1NF and contains only a single primary key and there is no partial dependency, therefore it is in 2NF.

 3NF: A relation R is in third normal form (3NF) if and only if it is in 2NF and every non-key attribute is non-transitively dependent on the primary key. Since the above table does not have any transitive dependencies, therefore it is in 3NF.

-

• Basic Queries:

- If the customer wants to see details of product present in the cart:

```
select * from Product  where Product_id in(
        select product_id from Cart_item0 where (Cart_id in (
            select Cart_id from Customer where Customer_id='CUS001'
        )));
```

Script Output x | Query Result x

SQL | All Rows Fetched: 3 in 0.013 seconds

	PRODUCT_ID	TYPE	COLOR	P_SIZE	GENDER	COMMISSION	COST	QUANTITY	SELLER_ID
1	PROD001	Shirt	Blue	Medium	Male	10	25	50	S10011
2	PROD002	Dress	Black	Small	Female	15	50	25	S10012
3	PROD003	Shoes	Red	Large	Unisex	5	75	10	S10013

- If a customer wants to see order history:

```
select product_id,Quantity_wished from Cart_item where (purchased='Y' and
Cart_id in (select Cart_id from customer where Customer_id='cid101'));
```

Script Output x | Query Result x

SQL | All Rows Fetched: 3 in 0.004 seconds

	PRODUCT_ID	QUANTITY_WISHED
1	PROD001	1
2	PROD002	2
3	PROD003	2

- Customer wants to see filtered products on the basis of size , gender,type:

select Product_id, Color, Cost, Seller_id from Product2 where (Type='Pants' and P_size='Small' and Gender='Male' and Quantity>0)

Script Output × Query Result ×

SQL | All Rows Fetched: 1 in 0.004 seconds

	PRODUCT_ID	COLOR	COST	SELLER_ID
1	PROD009	Brown	45	S10016

- if customer wants to modify the cart:

delete from cart_item0 where (product_id='PROD002' and Cart_id in (select Cart_id from Customer where Customer_id='CUS001'));

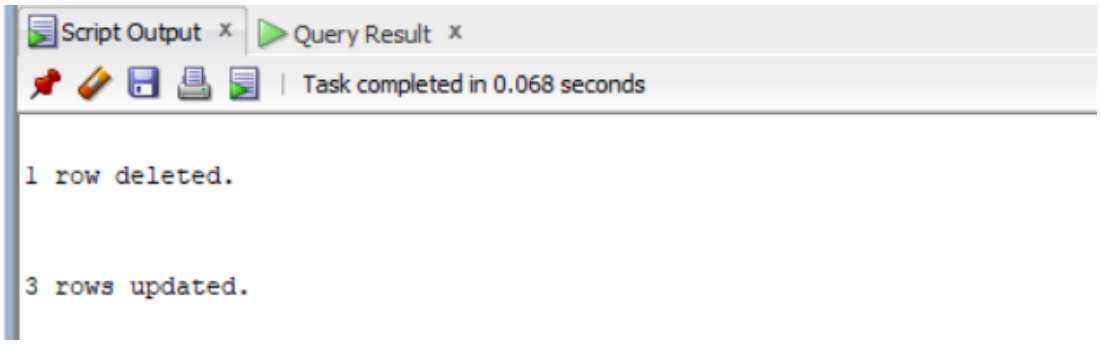
Script Output × Query Result ×

Task completed in 0.083 seconds

```
1 row deleted.
```

- If a seller stops selling his product:

```
delete from seller where seller_id = 'sid100';
update product set quantity = 00 where seller_id is NULL;
```



- ## PL/SQL Functions:

```
create or replace procedure cost_filter(c in number,t in varchar)
   is
   cs product.cost%type;
   ty  product.type%type;
   id product.product_id%type;
   cursor cf is
   select product_id,cost,type from product where cost<c and type=t;
   begin
   open cf;
   loop
   fetch cf into  id,cs,ty;
   exit when cf%notfound;
   dbms_output.put_line('Product' || id || 'has cost ' || cs || ' and the type is' || ty);
   end loop;
   close cf;
   exception
   when no_data_found then
   dbms_output.put_line('Sorry no such products exist');
  end;
```

```
Statement processed.
ProductPROD001has cost 25 and the type isShirt
```

Script Output x Query Result x

Task completed in 0.102 seconds

```
Procedure COST_FILTER compiled


Procedure COST_FILTER compiled
```

- ## Function which returns total number of products which a particular seller sells:

```
create or replace function totalProducts(sId in varchar)
return number
is
total number(2):=0;
begin
select count(*) into total
from product
where seller_id=sId;
return total;
end;
```

- ## Function execution:

Script Output x | Query Result x

Task completed in 0.076 seconds

```
Function TOTALPRODUCTS compiled


Function TOTALPRODUCTS compiled
```

TOTALPRODUCTS('S10001')
1

- ## Procedure which returns the total quantity of product with the given ID:

```
create or replace procedure prod_details(p_id in varchar)
  is
  quan number(2);
  begin
  select quantity into quan from product where product_id=p_id;
  exception
  when no_data_found then
  dbms_output.put_line('Sorry no such product exist !!');
  end;
```


```
Procedure PROD_DETAILS compiled
```

• Triggers

- ## Function to count number of cart items:

```
create OR REPLACE FUNCTION numCartId(cd IN VARCHAR)
RETURN NUMBER
IS
   total NUMBER(2) := 0;
BEGIN
   SELECT COUNT(*) INTO total
   FROM cart_item
   WHERE cart_id = cd;

   RETURN total;
END;
```

```
CREATE OR REPLACE TRIGGER before_customer
BEFORE INSERT ON customer
FOR EACH ROWDECLARE
c VARCHAR(10);n
NUMBER(2);
BEGIN
c := :new.cart_id; n :=
numCartId(c);

IF n > 0 THEN
dbms_output.put_line('Sorry');END IF;

    INSERT INTO cart VALUES (c);
END;
```

```
Trigger created.
```

- Trigger to update the total amount of user everytime he adds something to payment table:

```
create or replace function total_cost(cId in varchar)
    return number
    is
    total number(2) :=0;
    begin
    select sum(cost) into total from product,cart_item where
product.product_id=cart_item.product_id and cart_id=cId;
    return total;
    end;
```

```
Function created.
```

```
create or replace trigger before_pay_up
before insert
   on
   payment
   for each row
   declare
   total
   number(3);
      begin
      total
   :=total_cost(
   :new.cart_id
   );
      insert into
   payment
   values(:new
   .payment_id
   ,:new.paym
   ent_date,:ne
   w.payment_
   type,:new.c
   ustomer_id,:
   new.cart_id,
   total);
      end;;
```

```
Trigger created.
```