Galactica

-

Electronic Commerce System

Database Management System Design Specification

Introduction

We have been hired by the corporation Galactica to help develop their galaxy wide E-commerce system! We are tasked with building a Database Management System (DBMS) that will handle their massive online shopping service. The purpose of this DBMS is to store and manage information related to the purchasing and shipping of products for our customers. In order to guide us during this huge undertaking, we will outline the various aspects of the database management system we are going to be creating. In this report we will discuss the **entities** we will implement and their purpose, the various **attributes** of the entities, and the **relationship** between different entities.

Entities

In total, our system will implement seven entities, these seven entities are:

1. **User**: The user entity will contain all the information about our clients who will use the retail system. This might include information that will be needed to make an order such as, their First and Last name, their billing address, and payment information etc.
2. **Product**: The product entity will contain all the information about the products that it will offer.
3. **Order**: The order entity will contain all the information about orders that the users will place
4. **Review**: The review entity will store all the information about reviews left on products that the users will leave behind
5. **Shopping Cart**: The shopping cart entity will list the products in the shopping carts of all our users
6. **Payment**: This entity's purpose is to track all the information about transactions made by all the users
7. **Shipping**: The shipping entity will keep track of the details of all the products shipped

Attributes

Each of our entities will have various attributes:
* Any attribute underlined will act as the ***primary key***

1. **User**:
    a. *User Id*:
        i. *Simple*

   ii. *Single Valued*
   iii. *Stored*
   iv. *Not null*

b. *First Name*
   i. *Simple*
   ii. *Single Valued*
   iii. *Stored*
   iv. *Not null*

c. *Last Name*
   i. *Simple*
   ii. *Single Valued*
   iii. *Stored*
   iv. *Not null*

d. *Username*: Since usernames are unique to every user, the username can also act as the primary key
   i. *Simple*
   ii. *Single Valued*
   iii. *Stored*
   iv. *Not null*

e. *Email*
   i. *Simple*
   ii. *Single Valued*
   iii. *Stored*
   iv. *Not null*

f. *Password Hash Value: It is not secure to store the password itself, so the password will be hashed first before the database receives it*
   i. *Simple*
   ii. *Single Valued*
   iii. *Stored*
   iv. *Not null*

g. *Shipping Address*
   i. *Composite*
   ii. *Multiple Valued*
   iii. *Stored*
   iv. *Not null*

h. *Phone Number*
   i. *Simple*
   ii. *Single Valued*
   iii. *Stored*
   iv. *Not null*

i. *Payment Information*
    i. *Composite*
    ii. *Multiple Valued*
    iii. *Stored*
    iv. *Not null*

2. Product

a. *<u>Product ID</u>*
    i. *Simple*
    ii. *Single Valued*
    iii. *Stored*
    iv. *Not null*
b. *Product Name*
    i. *Simple*
    ii. *Single Valued*
    iii. *Stored*
    iv. *Not null*
c. *Description*
    i. *Simple*
    ii. *Single Valued*
    iii. *Stored*
    iv. *Not null*
d. *Price*
    i. *Simple*
    ii. *Single Valued*
    iii. *Stored*
    iv. *Not null*
e. *Category*
    i. *Simple*
    ii. *Single Valued*
    iii. *Stored*
    iv. *Not null*
f. *Availability*
    i. *Simple*
    ii. *Single Valued*
    iii. *Stored*
    iv. *Not null*
g. *Image URL*
    i. *Simple*
    ii. *Single Valued*
    iii. *Stored*

iv. *Not null*

3. **Order**
    a. *Order ID*
        i. *Simple*
        ii. *Single Valued*
        iii. *Stored*
        iv. *Not null*
    b. *User ID*
        i. *Simple*
        ii. *Single Valued*
        iii. *Stored*
        iv. *Not null*
    c. *Product IDs: IDs of products ordered*
        i. *Simple*
        ii. *Single Valued*
        iii. *Stored*
        iv. *Not null*
    d. *Shipping ID*
        i. *Simple*
        ii. *Single Valued*
        iii. *Stored*
        iv. *Not null*
    e. *Order Status*
        i. *Simple*
        ii. *Single Valued*
        iii. *Stored*
        iv. *Not null*
4. **Review**
    a. *Review ID*
        i. *Simple*
        ii. *Single Valued*
        iii. *Stored*
        iv. *Not null*
    b. *User ID*
        i. *Simple*
        ii. *Single Valued*
        iii. *Stored*
        iv. *Not null*
    c. *Product ID*
        i. *Simple*

   ii. *Single Valued*
   iii. *Stored*
   iv. *Not null*
- d. *Rating: A rating out of 5*
   i. *Simple*
   ii. *Single Valued*
   iii. *Stored*
   iv. *Not null*
- e. *Review Text*
   i. *Simple*
   ii. *Single Valued*
   iii. *Stored*
   iv. *Not null*
- f. *Timestamp*
   i. *Simple*
   ii. *Single Valued*
   iii. *Stored*
   iv. *Not null*

5. Shopping Cart

- a. *User ID*
   i. *Simple*
   ii. *Single Valued*
   iii. *Stored*
   iv. *Not null*
- b. *Product IDs: IDs of products ordered*
   i. *Simple*
   ii. *Single-valued*
   iii. *Stored*
   iv. *Not null*

6. Payment

- a. *Payment ID*
   i. *Simple*
   ii. *Single Valued*
   iii. *Stored*
   iv. *Not null*
- b. *Order ID*
   i. *Simple*
   ii. *Single Valued*
   iii. *Stored*
   iv. *Not null*

c. *Payment Information*
    i. *Composite*
    ii. *Single Valued*
    iii. *Stored*
    iv. *Not null*
d. *Cost*
    i. *Simple*
    ii. *Single Valued*
    iii. *Stored*
    iv. *Not null*
e. *Timestamp*
    i. *Simple*
    ii. *Single Valued*
    iii. *Stored*
    iv. *Not null*

7. Shipping

a. *Shipping ID*
    i. *Simple*
    ii. *Single Valued*
    iii. *Stored*
    iv. *Not null*
b. *Address*
    i. *Simple*
    ii. *Single Valued*
    iii. *Stored*
    iv. *Not null*
c. *Delivery Status: The information about delivery status*
    i. *Composite*
    ii. *Single Valued*
    iii. *Stored*
    iv. *Not null*

Functional Dependencies

USER

USER (userid, username, firstname, lastname, phoneno, email, password)

{userid} -> {username, firstname, lastname, phoneno, email, password}
{username} -> {userid, firstname, lastname, phoneno, email, password}
{email} -> {userid, username, firstname, lastname, phoneno, password}

PAYMENTINFO:

PAYMENTINFO (paymentinfoid, userid, cardname, cardnumber, securitycode, expirydate)

{paymentinfoid} -> {userid, cardname, cardnumber, securitycode, expirydate}
{cardnumber} -> {paymentinfoid, userid, cardname, securitycode, expirydate}

USERADDRESS

USERADDRESS (addressid, userid, streetaddress, city, state, postalcode, country)

{addressid} -> {userid, streetaddress, city, state, postalcode, country}

PRODUCT

PRODUCT (productid, productname, description, price, stockquantity, category)

{productid} -> {productname, description, price, stockquantity, category}

SHOPPINGCART

SHOPPINGCART (userid, productid, quantity)

{userid, productid} -> {quantity}

REVIEW

REVIEW (productid, userid, rating, description, timestamp)

{productid, userid} -> {rating, description, timestamp}

ORDER

ORDER (orderid, userid, totalcost, transactionid, shippingaddress, orderstatus, orderdate)

{orderid} -> {userid, totalcost, transactionid, shippingaddress, orderstatus, orderdate}

ORDER_PRODUCTS

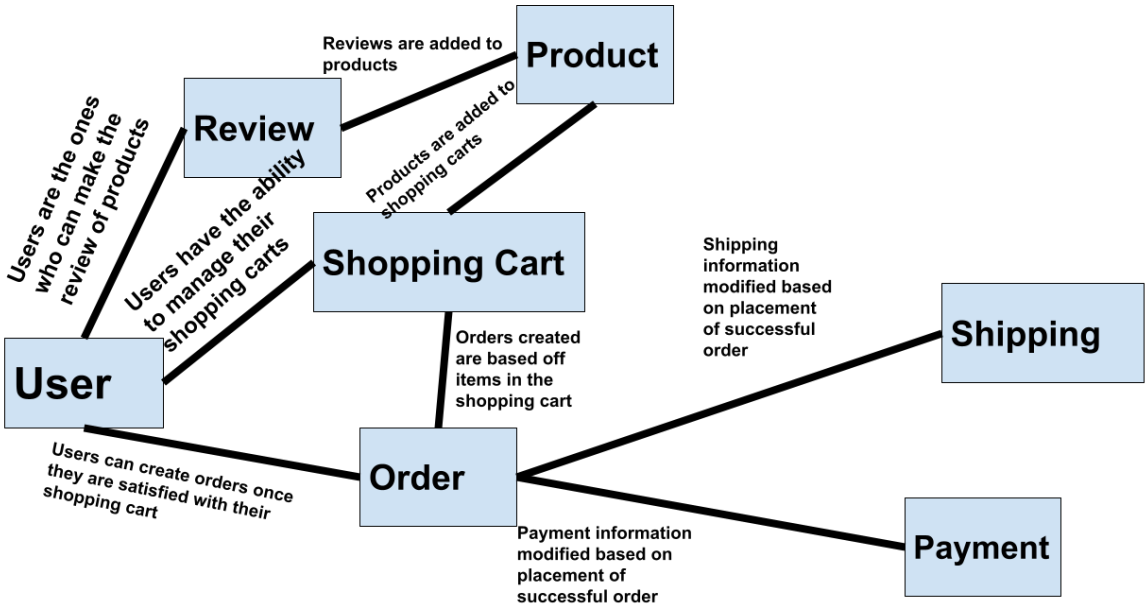
ORDER_PRODUCTS (orderid, productid, quantity)

{orderid, productid} -> {quantity}

Application

Relationships

Due to the interconnecting nature of the various relationships between the entities, a graph has been created to describe the relationship between the products



From this graph, we can already see the importance of the **user**, and the **order** entities in terms of data. This reflects the model of E-commerce systems, as it is designed to help **users** purchase **products** in the form of **orders**

Here are some of the relationships listed out:

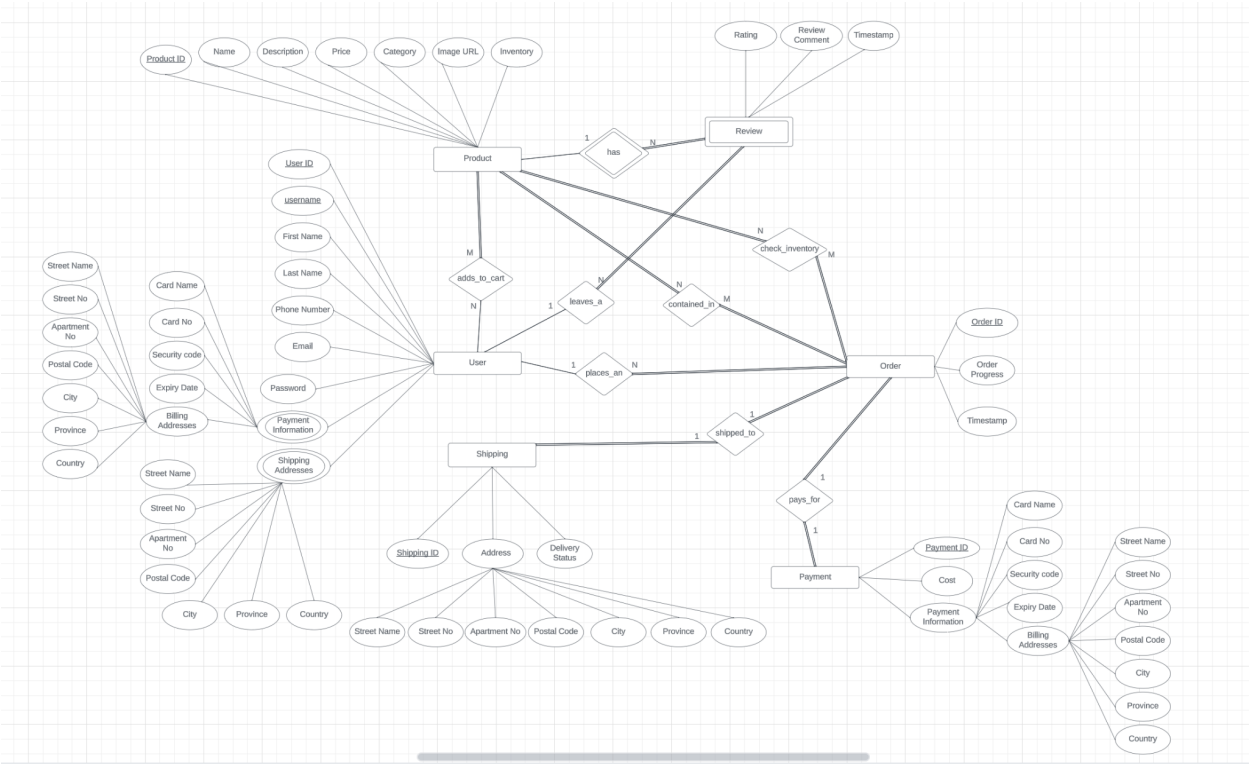
1. ***User :: Review***: Users are the one who can leave reviews of the products
2. ***User :: Shopping Cart***: Users can manage and add to their shopping carts
3. ***User :: Order****: Users can create orders once they are satisfied with the shopping cart*
4. ***Review :: Product****: Reviews are added to products*

5. ***Shopping Cart :: Product****: Products are added to shopping carts*
6. ***Shopping Cart :: Order****: Orders created are based off items in the shopping cart cart*
7. ***Order :: Shipping****: Shipping entries modified based on placement of successful order*
8. ***Order :: Payment****: Payment entries modified based on placement of successful order*

Some Example Sample Queries

- List all orders placed by Zubair (Find Zubair's User ID from Zubair's user entry and then use the User ID to find all the orders placed by that user and list them)
- List all users with 3 or more orders placed (Find at least 3 orders with the same User ID and list those users)
- Add/Delete item to Zubair's shopping cart (Find Zubair's User ID from Zubair's user entry and use that User ID to find Zubair's Shopping Cart entry and then add/delete the new item's Product ID to that Shopping Cart entry)
- Update payment information for Tahshin (Find Tahshin's user entry and update the Payment information)
- List all reviews by Zubair (Find Zubair's User ID from Zubair's user entry and use the User ID to find all the reviews by that user)
- List all products that are available (Find all products that are available and list them)

ER Diagram



Source Code

Source Code

USER

```
CREATE TABLE "USER"
  (      "USERID" NUMBER(*,0),
         "USERNAME" VARCHAR2(255 BYTE) NOT NULL ENABLE,
         "FIRSTNAME" VARCHAR2(255 BYTE) NOT NULL ENABLE,
         "LASTNAME" VARCHAR2(255 BYTE) NOT NULL ENABLE,
         "PHONENO" VARCHAR2(20 BYTE),
         "EMAIL" VARCHAR2(255 BYTE) NOT NULL ENABLE,
         "PASSWORD" VARCHAR2(255 BYTE) NOT NULL ENABLE,
          PRIMARY KEY ("USERID"),
          UNIQUE ("EMAIL") );
```

PAYMENTINFO:

```
CREATE TABLE "PAYMENTINFO"
```

```
  (       "PAYMENTINFOID" NUMBER(*,0),
          "USERID" NUMBER(*,0) NOT NULL ENABLE,
          "CARDNAME" VARCHAR2(255 BYTE) NOT NULL ENABLE,
          "CARDNUMBER" VARCHAR2(16 BYTE) NOT NULL ENABLE,
          "SECURITYCODE" VARCHAR2(4 BYTE) NOT NULL ENABLE,
          "EXPIRYDATE" DATE NOT NULL ENABLE,
           CONSTRAINT "PAYMENTINFO_PK" PRIMARY KEY ("PAYMENTINFOID"),
           CONSTRAINT "PAYMENTINFO_UK1" UNIQUE ("CARDNUMBER")
           FOREIGN KEY ("USERID")
            REFERENCES "USER" ("USERID") ENABLE
  ) ;
```

USERADDRESS

```
CREATE TABLE "USERADDRESS"
  (       "ADDRESSID" NUMBER(*,0),
          "USERID" NUMBER(*,0) NOT NULL ENABLE,
          "STREETADDRESS" VARCHAR2(255 BYTE) NOT NULL ENABLE,
          "CITY" VARCHAR2(100 BYTE) NOT NULL ENABLE,
          "STATE" VARCHAR2(50 BYTE),
          "POSTALCODE" VARCHAR2(20 BYTE) NOT NULL ENABLE,
          "COUNTRY" VARCHAR2(100 BYTE) NOT NULL ENABLE,
           PRIMARY KEY ("ADDRESSID"),
           FOREIGN KEY ("USERID")
            REFERENCES "USER" ("USERID") ENABLE
  ) ;
```

PRODUCT

```
CREATE TABLE "PRODUCT"
  (       "PRODUCTID" NUMBER(*,0),
          "PRODUCTNAME" VARCHAR2(255 BYTE) NOT NULL ENABLE,
          "DESCRIPTION" CLOB,
          "PRICE" NUMBER(10,2) NOT NULL ENABLE,
          "STOCKQUANTITY" NUMBER(10,0) NOT NULL ENABLE,
          "CATEGORY" VARCHAR2(50 BYTE),
           PRIMARY KEY ("PRODUCTID")
  ) ;
```

SHOPPINGCART

```
CREATE TABLE "SHOPPINGCART"
```

```
(       "USERID" NUMBER,
        "PRODUCTID" NUMBER,
        "QUANTITY" NUMBER NOT NULL ENABLE,
         PRIMARY KEY ("USERID", "PRODUCTID"),
         FOREIGN KEY ("USERID")
          REFERENCES "USER" ("USERID") ENABLE,
         FOREIGN KEY ("PRODUCTID")
          REFERENCES "PRODUCT" ("PRODUCTID") ENABLE
) ;
```

REVIEW

```
CREATE TABLE "REVIEW"
(       "PRODUCTID" NUMBER NOT NULL ENABLE,
        "USERID" NUMBER NOT NULL ENABLE,
        "RATING" NUMBER(2,1) NOT NULL ENABLE,
        "DESCRIPTION" CLOB,
        "TIMESTAMP" TIMESTAMP (6) DEFAULT CURRENT_TIMESTAMP NOT NULL,
         CHECK (Rating >= 1 AND Rating <= 5) ENABLE,
         CONSTRAINT "REVIEW_PK" PRIMARY KEY ("PRODUCTID", "USERID"),
         CONSTRAINT "SYS_C001550472" FOREIGN KEY ("PRODUCTID")
          REFERENCES "PRODUCT" ("PRODUCTID") ON DELETE CASCADE ENABLE,
         CONSTRAINT "SYS_C001550473" FOREIGN KEY ("USERID")
          REFERENCES "USER" ("USERID") ON DELETE CASCADE ENABLE) ;
```

ORDER

```
CREATE TABLE “ORDER"
(       "ORDERID" NUMBER(*,0),
        "USERID" NUMBER(*,0) NOT NULL ENABLE,
        "TOTALCOST" NUMBER(10,2) NOT NULL ENABLE,
        "TRANSACTIONID" NUMBER(*,0) NOT NULL ENABLE,
        "SHIPPINGADDRESSID" NUMBER(*,0) NOT NULL ENABLE,
        "ORDERSTATUS" VARCHAR2(20 BYTE) NOT NULL ENABLE,
        "TIMESTAMP" TIMESTAMP (6) NOT NULL ENABLE,
         CHECK (
        "ORDERSTATUS" IN (
           'Processing',
           'Shipped',
           'Delivered',
           'Canceled'
```

```
        )
    ) ENABLE,
        PRIMARY KEY ("ORDERID"),
        FOREIGN KEY ("USERID")
         REFERENCES "USER" ("USERID") ENABLE,
        FOREIGN KEY ("TRANSACTIONID")
         REFERENCES "PAYMENTINFO" ("PAYMENTINFOID") ENABLE,
        FOREIGN KEY ("SHIPPINGADDRESSID")
         REFERENCES "USERADDRESS" ("ADDRESSID") ENABLE
  );
```

ORDER_PRODUCTS

```
CREATE TABLE "ORDER_PRODUCTS"
  (       "ORDERID" NUMBER NOT NULL ENABLE,
          "PRODUCTID" NUMBER NOT NULL ENABLE,
          "QUANTITY" NUMBER NOT NULL ENABLE,
           CONSTRAINT "SYS_C001550501" PRIMARY KEY ("PRODUCTID", "ORDERID")
ENABLE,
          FOREIGN KEY ("PRODUCTID")
           REFERENCES "PRODUCT" ("PRODUCTID") ENABLE,
          FOREIGN KEY ("ORDERID")
           REFERENCES "ORDER" ("ORDERID") ENABLE
  );
```

Simple Queries

1) **USER table**

Query: Retrieve the distinct lastnames and ordered by the ascending ordr of the lastnames alphabetically

```
SELECT DISTINCT LASTNAME
FROM "USER"
ORDER BY LASTNAME ASC;
```

Result:

```
LASTNAME
-------------------------
Asad
```

Gerrard
Rahman
Saha
Shahriar
Shibly
Tapu
def

8 rows selected.

2) PRODUCT table

Query: Retrieve the count of product under each category and order the result by Product Count in ascending order

```
SELECT  CATEGORY, COUNT(*) AS PRODUCT_COUNT
FROM "PRODUCT"
GROUP BY CATEGORY
ORDER BY PRODUCT_COUNT ASC;
```

Result:

CATEGORY	PRODUCT_COUNT
Headphones	1
Tablets	1
Monitor	1
Shoes	1
Laptop	2
Phone	4

6 rows selected.

3) USERADDRESS table

Query: Retrieve the count of users in each city and order the result by ascending user count

```
SELECT CITY, COUNT(USERID) AS USER_COUNT
FROM "USERADDRESS"
GROUP BY CITY
ORDER BY USER_COUNT;
```

Result:

CITY	USER_COUNT
Cleveland	1
Paris	1
Mama	1
Mountain View	2
Toronto	2
Scarborough	3

6 rows selected.

4) **PAYMENTINFO Table:**

Query: Retrieve the count of cardholders whose cards expires this year, grouped by card name and ordered by count

```
SELECT CARDNAME, COUNT(*) AS CardCount
FROM "PAYMENTINFO"
WHERE EXPIRYDATE < TO_DATE('2023-12-31', 'YYYY-MM-DD')
GROUP BY CARDNAME
ORDER BY CardCount DESC;
```

Result:

CARDNAME	CardCount
Tahshin Shahriar	2
Zubair Shibly	2
Fardin Rahman	1
Steven Gerrard	1

5) **SHOPPINGCART Table:**

QUERY: -- Retrieve the total quantity of products in the shopping cart for each user, ordered by TOTAL_QUANTITY

```
SELECT USERID, SUM(QUANTITY) AS TOTAL_QUANTITY
```

```
FROM "SHOPPINGCART"
GROUP BY USERID
ORDER BY TOTAL_QUANTITY;
```

RESULT:

```
   USERID TOTAL_QUANTITY
---------- --------------
         8              1
         6              3
         2              6
         4              6
         1              8
         3             20
```

6 rows selected.

6) <u>REVIEW Table:</u>

QUERY: -- Retrieve the average rating for each product, ordered by product ID

```
SELECT PRODUCTID, AVG(RATING) AS AVERAGE_RATING
FROM "REVIEW"
GROUP BY PRODUCTID
ORDER BY PRODUCTID;
```

RESULT:

```
 PRODUCTID AVERAGE_RATING
---------- --------------
         1              5
         2              4
         3              2
         4              1
         5              5
         6              4
         8              5
```

7 rows selected.

7) <u>ORDER Table:</u>

QUERY: -- Retrieve the total cost of orders for each user, ordered by total cost in descending order

```
SELECT USERID, SUM(TOTALCOST) AS TOTAL_COST
FROM "ORDER"
GROUP BY USERID
ORDER BY TOTAL_COST DESC;
```

RESULT:

```
 USERID TOTAL_COST
---------- ----------
         8    4501000
         3     123413
         2       1690
         1        123
```

8) <u>ORDER_PRODUCTS Table:</u>

QUERY- Retrieve the average product quantity for each product in an order, ordered by PRODUCTID

```
SELECT PRODUCTID, AVG(QUANTITY) AS AVERAGE_QUANTITY
FROM "ORDER_PRODUCTS"
GROUP BY PRODUCTID
ORDER BY PRODUCTID;
```

RESULT:

```
 PRODUCTID AVERAGE_QUANTITY
---------- ----------------
         1                3
         2                3
         3                2
         4                6
         6                3
         7                1
         8                2
        10                4
```

8 rows selected.

Queries

1. **Query:** Retrieve the Users that has an Order of over $500 and displays their First names and which City they are from:

```
SELECT DISTINCT U.USERID, U.FIRSTNAME, S.CITY, O.TOTALCOST
FROM "ZSHIBLY"."USER" U, "ZSHIBLY"."ORDER" O,
"ZSHIBLY"."USERADDRESS" S
WHERE
U.USERID = O.USERID
AND
U.USERID = S.USERID
AND
TOTALCOST >= 500
ORDER BY TOTALCOST;
```

Result:

USERID	FIRSTNAME	CITY	TOTALCOST
8	Steven	Mountain View	1000
2	Tahshin	Scarborough	1234
3	Fardin	Toronto	123413
8	Steven	Mountain View	4500000

2. **QUERY**: Retrieve the products that a specific User has added to their cart and displays the User's fullname, Product names, and the Quantity of products added to the shopping cart

```
SELECT U.FIRSTNAME, U.LASTNAME, P.PRODUCTNAME,
S.QUANTITY AS SHOPPING_CART_QUANTITY
FROM "ZSHIBLY"."PRODUCT" P, "ZSHIBLY"."USER" U,
"ZSHIBLY"."SHOPPINGCART" S
WHERE P.PRODUCTID = S.PRODUCTID
AND U.USERID = S.USERID
AND U.FIRSTNAME = 'Tahamid';
```

RESULT:

FIRSTNAME	LASTNAME	PRODUCTNAME	Shopping_Cart_Quantity
Tahamid	Tapu	Iphone 15	3
Tahamid	Tapu	Iphone 12	4
Tahamid	Tapu	Macbook Air	1

3. **QUERY:** Retrieve the First names of the Users who left Reviews on Products that they Ordered:

```
SELECT U.FIRSTNAME, P.PRODUCTNAME, R.DESCRIPTION
FROM "ZSHIBLY"."USER" U, "ZSHIBLY"."ORDER" O,
"ZSHIBLY"."ORDER_PRODUCTS" OP, "ZSHIBLY"."REVIEW" R,
"ZSHIBLY"."PRODUCT" P
WHERE
U.USERID = O.USERID
AND
O.USERID = R.USERID
AND
O.ORDERID = OP.ORDERID
AND
OP.PRODUCTID = P.PRODUCTID
AND
P.PRODUCTID = R.PRODUCTID
ORDER BY P.PRODUCTNAME;
```

Result:

FIRSTNAME	PRODUCTNAME	DESCRIPTION
Tahamid	Iphone 12	Worst purchase ever
Steven	Iphone 14	just wow
Tahamid	Macbook Air	Good product, I wish it was less

Fardin	Nike Air Force	expensive Amazing product!

4. **QUERY:** Retrieve the Name and Address of a User whose Order is ‘Processing’ to be shipped to a specific City, for example ‘Scarborough’

```
SELECT DISTINCT O.ORDERID, U.FIRSTNAME, U.LASTNAME,
A.STREETADDRESS, A.CITY, A.STATE
FROM "ZSHIBLY"."USER" U, "ZSHIBLY"."ORDER" O,
"ZSHIBLY"."USERADDRESS" A
WHERE
O.USERID = U.USERID
AND
O.SHIPPINGADDRESSID = A.ADDRESSID
AND
O.ORDERSTATUS = 'Processing'
AND
A.CITY = 'Scarborough'
ORDER BY O.ORDERID;
```

Queries

1. **Query:** Retrieve the Users that has ordered a product and left a review with ratings and descriptions

```
SELECT U.USERID, U.FIRSTNAME, U.LASTNAME
FROM "ZSHIBLY"."USER" U
WHERE EXISTS(
SELECT O.USERID
FROM "ZSHIBLY"."ORDER" O,"ZSHIBLY"."REVIEW" R
WHERE O.USERID = R.USERID
AND
R.DESCRIPTION IS NOT NULL
AND
O.USERID = U.USERID
)
ORDER BY U.USERID;
```

Result:

	USERID	FIRSTNAME	LASTNAME
1	1	Tahhamid	Tapu
2	3	Fardin	Rahman
3	8	Steven	Gerrard

2. **QUERY**: Retrieve the Users who added a product in the shopping cart but didn't order it

```
SELECT DISTINCT U.USERID, U.FIRSTNAME
FROM "ZSHIBLY"."USER" U,"ZSHIBLY"."SHOPPINGCART" S
WHERE U.USERID = S.USERID
AND
NOT EXISTS
(SELECT O.USERID
   FROM "ZSHIBLY"."ORDER" O, "ZSHIBLY"."SHOPPINGCART" S
   WHERE
   O.USERID = S.USERID
   AND
   U.USERID = O.USERID
  )
ORDER BY U.USERID;
```

RESULT:

	USERID	FIRSTNAME
1	4	Tauseef
2	6	ed2ed

3. **QUERY:** Retrieve products that has never been ordered or ordered only once

```
SELECT P.PRODUCTID, P.PRODUCTNAME
```

```
FROM "ZSHIBLY"."PRODUCT" P
WHERE NOT EXISTS (
   SELECT *
   FROM "ZSHIBLY"."ORDER_PRODUCTS" OP
   WHERE OP.PRODUCTID = P.PRODUCTID
)
UNION
SELECT P.PRODUCTID, P.PRODUCTNAME
FROM "ZSHIBLY"."PRODUCT" P
WHERE P.PRODUCTID IN (
   SELECT OP.PRODUCTID
   FROM "ZSHIBLY"."ORDER_PRODUCTS" OP
   GROUP BY OP.PRODUCTID
   HAVING COUNT(OP.ORDERID) = 1
);
```

Result:

	PRODUCTID	PRODUCTNAME
1	2	Iphone 14
2	3	Iphone 13
3	5	Nike Air Force
4	7	Macbook Pro
5	8	Ipad air
6	9	Samsung Monitor
7	10	Airpods pro

4. **QUERY:** Retrieve the products that does not have any reviews

```
(SELECT P.PRODUCTID
FROM "ZSHIBLY"."PRODUCT" P)
MINUS
(SELECT R.PRODUCTID
FROM "ZSHIBLY"."REVIEW" R, "ZSHIBLY"."PRODUCT" P
WHERE R.PRODUCTID=P.PRODUCTID);
```

Result:

	PRODUCTID
1	7
2	8
3	9
4	10

5. **Query:** Retrieve the first and last name of the users whose postal code starts with M:

```
SELECT U.FIRSTNAME, U.LASTNAME
FROM "ZSHIBLY"."USER" U, "ZSHIBLY"."USERADDRESS" UA
WHERE U.USERID = UA.USERID
AND
UA.POSTALCODE LIKE 'M%';
```

Result:

	FIRSTNAME	LASTNAME
1	Tahshin	Shahriar
2	Tahshin	Shahriar

6. **Query:** Retrieve the products that has an average rating between 1 to 3:

```
SELECT P.PRODUCTNAME, R.PRODUCTID, AVG(R.RATING) AS
AVERAGE_RATING
FROM "ZSHIBLY"."PRODUCT" P, "ZSHIBLY"."REVIEW" R
WHERE P.PRODUCTID = R.PRODUCTID
GROUP BY P.PRODUCTNAME,R.PRODUCTID
HAVING AVG(R.RATING) BETWEEN 1 AND 3
ORDER BY AVG(R.RATING);
```

Result:

	PRODUCTNAME	PRODUCTID	AVERAGE_RATING
1	Iphone 12	4	1
2	Iphone 13	3	2

Result:

```
 ORDERID FIRSTNAME LASTNAME  STREET ADDRESS   CITY           STATE
---------- -------------------- ---------------- ------------ --------------   --------------     -----------
```

1 Tahamid Tapu 666 Warden Avenue Scarborough Ontario

VIEWs

VIEW 1: VIEW UserPaymentView DISPLAYS INFORMATION ABOUT USER, USER'S PAYMENT INFO, AND THEIR ADDRESS

```
CREATE VIEW UserPaymentView AS
SELECT DISTINCT
   U.USERID,
   U.USERNAME,
   U.FIRSTNAME,
   U.LASTNAME,
   U.EMAIL,
   P.CARDNAME,
   P.CARDNUMBER,
   S.CITY,
   S.COUNTRY
FROM "ZSHIBLY"."USER" U, "ZSHIBLY"."PAYMENTINFO"
P,"ZSHIBLY"."USERADDRESS" S
WHERE
U.USERID = S.USERID
AND
U.USERID = P.USERID
AND
S.USERID = P.USERID
ORDER BY U.USERID;
```

Result:

	USERID	USERNAME	FIRSTNAME	LASTNAME	EMAIL	CARDNAME	CARDNUMBER	CITY	COUNTRY
1	1	stapu	Tahhamid	Tapu	tahamid	Tahamid Tapu	123456	Scarborough	Canada
2	2	tshahria	Tahshin	Shahriar	tahshin890@gmail.com	Tahshin Shahriar	234423	Scarborough	Canada
3	2	tshahria	Tahshin	Shahriar	tahshin890@gmail.com	Tahshin Shahriar	32323	Scarborough	Canada
4	2	tshahria	Tahshin	Shahriar	tahshin890@gmail.com	Tahshin Shahriar	654321	Scarborough	Canada
5	3	fardinr	Fardin	Rahman	fardinr24@gmail.com	Fardin Rahman	673523	Toronto	Canada
6	8	stevie_g	Steven	Gerrard	gerrard@liverpool.com	Steven Gerrard	848484	Mountain View	USA
7	9	zshibly	Zubair	Shibly	zshibly@torontomu.ca	Zubair Shibly	1154257	Mountain View	USA
8	9	zshibly	Zubair	Shibly	zshibly@torontomu.ca	Zubair Shibly	424242	Mountain View	USA
9	9	zshibly	Zubair	Shibly	zshibly@torontomu.ca	Zubair Shibly	888888	Mountain View	USA

VIEW 2: OrderProductView DISPLAYS INFORMATION ABOUT THE PRODUCTS ORDERED AND THE ORDER STATUS

```
CREATE VIEW OrderProductView AS
SELECT
   O.ORDERID,
   P.PRODUCTNAME,
   P.PRICE,
   OP.QUANTITY,
   O.ORDERSTATUS
FROM "ZSHIBLY"."ORDER" O, "ZSHIBLY"."PRODUCT" P,
"ZSHIBLY"."ORDER_PRODUCTS" OP
WHERE
O.ORDERID = OP.ORDERID
AND
OP.PRODUCTID = P.PRODUCTID;
```

Result:

	ORDERID	PRODUCTNAME	PRICE	QUANTITY	ORDERSTATUS
1	1	Iphone 12	1100	2	Processing
2	1	Airpods pro	200	4	Processing
3	1	Macbook Air	1100	7	Processing
4	2	Iphone 13	1300	2	Shipped
5	2	Iphone 12	1100	6	Shipped
6	3	Iphone 15	1500	3	Canceled
7	4	Iphone 14	1400	3	Delivered
8	5	Iphone 15	1500	2	Delivered
9	5	Ipad air	800	2	Delivered
10	5	Macbook Air	1100	3	Delivered
11	5	Macbook Pro	1200	1	Delivered
12	6	Iphone 15	1500	3	Processing
13	6	Nike Air Force	140	2	Processing

VIEW 3: OrderInfoView DISPLAYS THE INFORMATION ABOUT USERS THAT HAS AN ORDER, THEIR DELIVERY ADDRESS, AND STATUS OF THE DELIVERY

```
CREATE VIEW OrderInfoView AS
SELECT
   O.ORDERID,
   U.FIRSTNAME,
   O.ORDERDATE,
   O.ORDERSTATUS,
   O.TOTALCOST,
   A.STREETADDRESS,
   A.CITY,
   A.POSTALCODE,
   A.COUNTRY
FROM "ZSHIBLY"."ORDER" O, "ZSHIBLY"."USER" U,"ZSHIBLY"."USERADDRESS" A
WHERE
O.USERID = U.USERID
```

```
AND
O.USERID = U.USERID
AND
O.SHIPPINGADDRESSID = A.ADDRESSID
ORDER BY O.ORDERID;
```

Result:

	ORDERID	FIRSTNAME	ORDERDATE	ORDERSTATUS	TOTALCOST	STREETADDRESS	CITY	POSTALCODE	COUNTRY
1	1	Tahhamid	23-10-03 20:20:50.160067000	Processing	123	666 Warden Avenue	Scarborough	L0L 6L6	Canada
2	2	Tahshin	23-10-12 20:20:29.460239000	Shipped	456	69 Eglintion Avenue East	Scarborough	M3M 3S5	Canada
3	3	Tahshin	23-10-12 20:59:15.890374000	Canceled	1234	128 Kennedy Road	Scarborough	M2L 2S3	Canada
4	4	Steven	23-10-03 20:20:34.382195000	Delivered	1000	99 Apple Drive	Mountain View	696969	USA
5	5	Steven	23-10-14 16:20:38.269894000	Delivered	4500000	99 Apple Drive	Mountain View	696969	USA
6	6	Fardin	23-10-30 20:59:51.388116000	Processing	123413	30 Victoria Street	Toronto	L2L 2L2	Canada

Shell code for DB UI:

```
#!/bin/bash

# Function to execute SQL query
run_query() {
  local query="$1"
  sqlplus64
"zshibly/02047333@(DESCRIPTION=(ADDRESS=(PROTOCOL=TCP)(Host=oracle.scs.ryer
son.ca)(Port=1521))(CONNECT_DATA=(SID=orcl)))" <<EOF > tempfile
  SET PAGESIZE 1000
  SET LINESIZE 200
  $query
  exit;
EOF
  sed -n '/^SQL>/,/^Disconnected/p' tempfile | head -n -2
  rm tempfile
}

# Main menu
while true; do
  clear
  echo "Galactica Database Main Menu"
  echo "----------------------------"
  echo "1. Execute SQL Query"
  echo "2. Create Table"
  echo "3. Insert Data"
  echo "4. Drop Table"
```

```
  echo "5. Exit"

  read -p "Enter option: " option

  case $option in
    1)
      clear
      read -p "Enter SQL query: " sql_query
      echo -e "\n"
      run_query "$sql_query"
      read -n 1 -s -r -p "Press any key to continue..."
      ;;
    2)
      clear
      read -p "Enter table name: " table_name
      echo
      read -p "Add first column name: " first_column_name

      query_str='CREATE TABLE "ZSHIBLY"."$table_name"" (""$first_column_name"

      while true; do
        while true; do
          echo -e "\nWhat is the data type?\nEnter the number corresponding to the data
type.\n1. NUMBER\n2. VARCHAR2\n3. DATE\n4. CLOB\n"
          read -n 1 -p "Enter your selection: " data_type_option
          case $data_type_option in
            1)
              query_str+=" NUMBER"
              break
              ;;
            2)
              query_str+=" VARCHAR2"
              break
              ;;
            3)
              query_str+=" DATE"
              break
              ;;
            4)
              query_str+=" CLOB"
              break
              ;;
            *)
              echo "Invalid option. Please try again."
```

```
      read -n 1 -s -r -p "Press any key to continue..."
      ;;
  esac
done

while true; do
  echo -e "\n"
  read -n 1 -p "Is it a primary key? y/n: " primary_key_option
  case $primary_key_option in
    y)
      query_str+=" PRIMARY KEY"
      break
      ;;
    n)
      break
      ;;
    *)
      echo "Invalid option. Please try again."
      read -n 1 -s -r -p "Press any key to continue..."
      ;;
  esac
done

while true; do
  echo -e "\n"
  read -n 1 -p "Can it be a NULL? y/n: " null_option
  case $null_option in
    y)
      break
      ;;
    n)
      query_str+=" NOT NULL"
      break
      ;;
    *)
      echo "Invalid option. Please try again."
      read -n 1 -s -r -p "Press any key to continue..."
      ;;
  esac
done
echo -e "\n"
read -n 1 -p "Do you want to add more columns? y/n: " more_columns_option
echo -e "\n"
case $more_columns_option in
```

```
        y)
          clear
          read -p "Name of new column: " column_name
          query_str+=", $column_name"
          ;;
        n)
          query_str+=");"
          break
          ;;
        *)
          echo "Invalid option. Please try again."
          read -n 1 -s -r -p "Press any key to continue..."
          ;;
      esac
    done
    echo -e "\n"
    run_query "$query_str"
    read -n 1 -s -r -p "Press any key to continue..."
    ;;
  3)
    clear
    read -p "Enter table name: " table_name
    echo -e "\n"
    echo -e "$table_name"" table has the following columns:\n"
    run_query 'DESCRIBE "ZSHIBLY".'"$table_name"
    echo -e "\n"
    query_str="INSERT INTO """ZSHIBLY".'"$table_name"" ("
    read -p "Enter column names separated by commas: " column_names
    echo -e "\n"
    query_str+="$column_names"") VALUES ("
    read -p "Enter values separated by commas (values need to be in the same order in
which the columns were listed): " values
    query_str+="$values"");"
    echo -e "\n"
    run_query "$query_str"
    read -n 1 -s -r -p "Press any key to continue..."
    ;;
  4)
    clear
    read -p "Enter table name to be dropped: " table_name
    sql_query="DROP TABLE """ZSHIBLY".'"$table_name""";"
    echo -e "\n"
    run_query "$sql_query"
    read -n 1 -s -r -p "Press any key to continue..."
```

```
    ;;
  5)
    clear
    echo "Goodbye!"
    exit 0
    ;;
  *)
    echo "Invalid option. Please try again."
    read -n 1 -s -r -p "Press any key to continue..."
    ;;

 esac
done
```