Lab Report Cover Page

Course Number	CPS510
Course Title	Database Systems I
Semester/Year	F2020
Instructor	Dr. A. Abhari
TA Name	Haytham Qushtom

Assignment/Lab Report No.	10
Report Title	Final Documentation with RA

Section No.	8
Group No.	8
Submission Date	December 1, 2020
Due Date	December 1, 2020

Student Name	Student ID	Initial
Tanvir Billah	500829695	T.B.
James Choi	500931983	J.C.
Mahamudul Islam	500963051	M.I.

Project Description

Application: E-Commerce System

Description: Allow users to buy and sell products from any physical location via the internet.

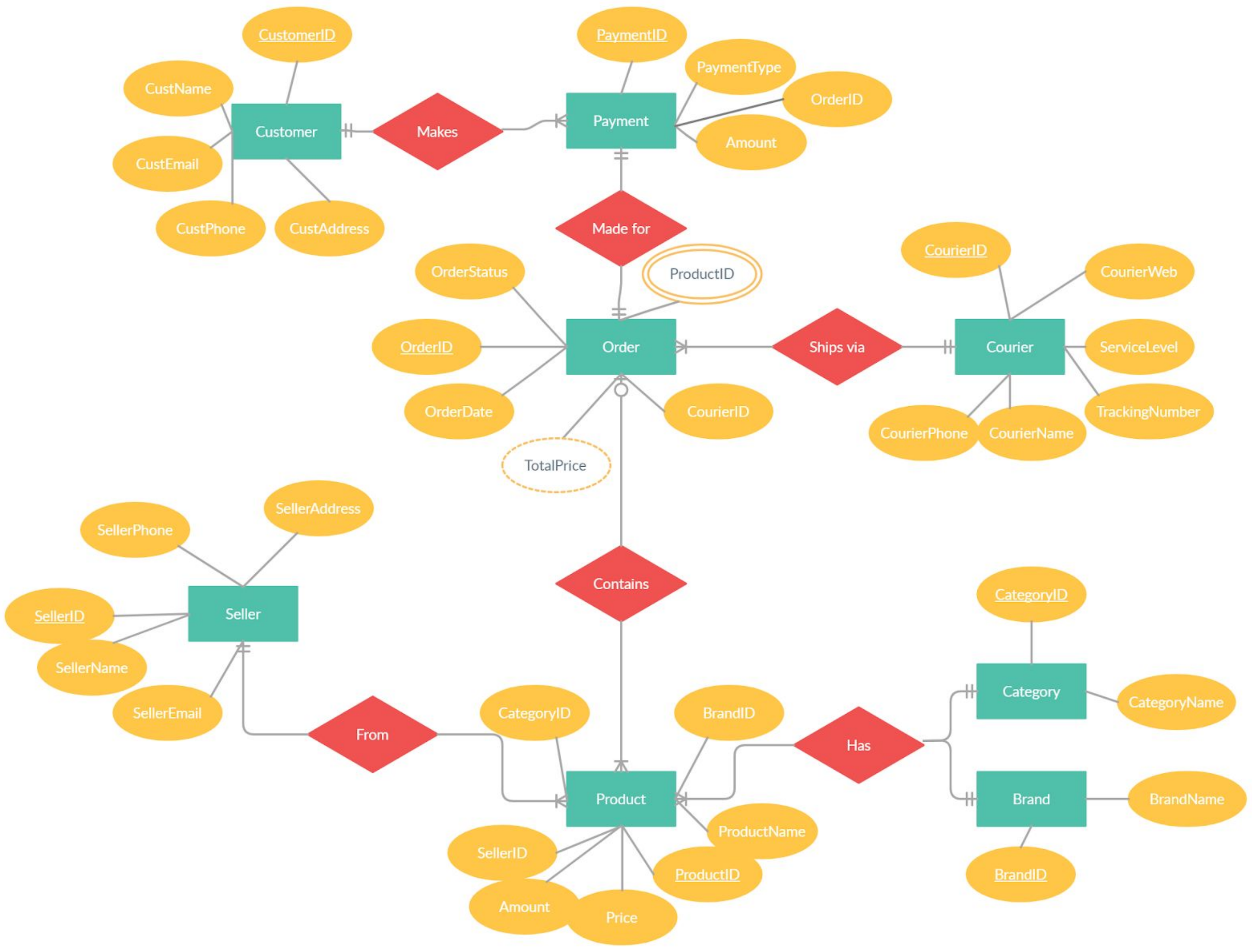
Functions: Users can register new accounts or modify existing account information. Users can browse available products from a seller and then place orders by making a payment. Order, product, and payment information is managed to ensure smooth transactions and deliveries.

Information expected from it: User information (username, password, email, contactID), order information (billing, shipping locations etc) product information (quantity, location, etc)

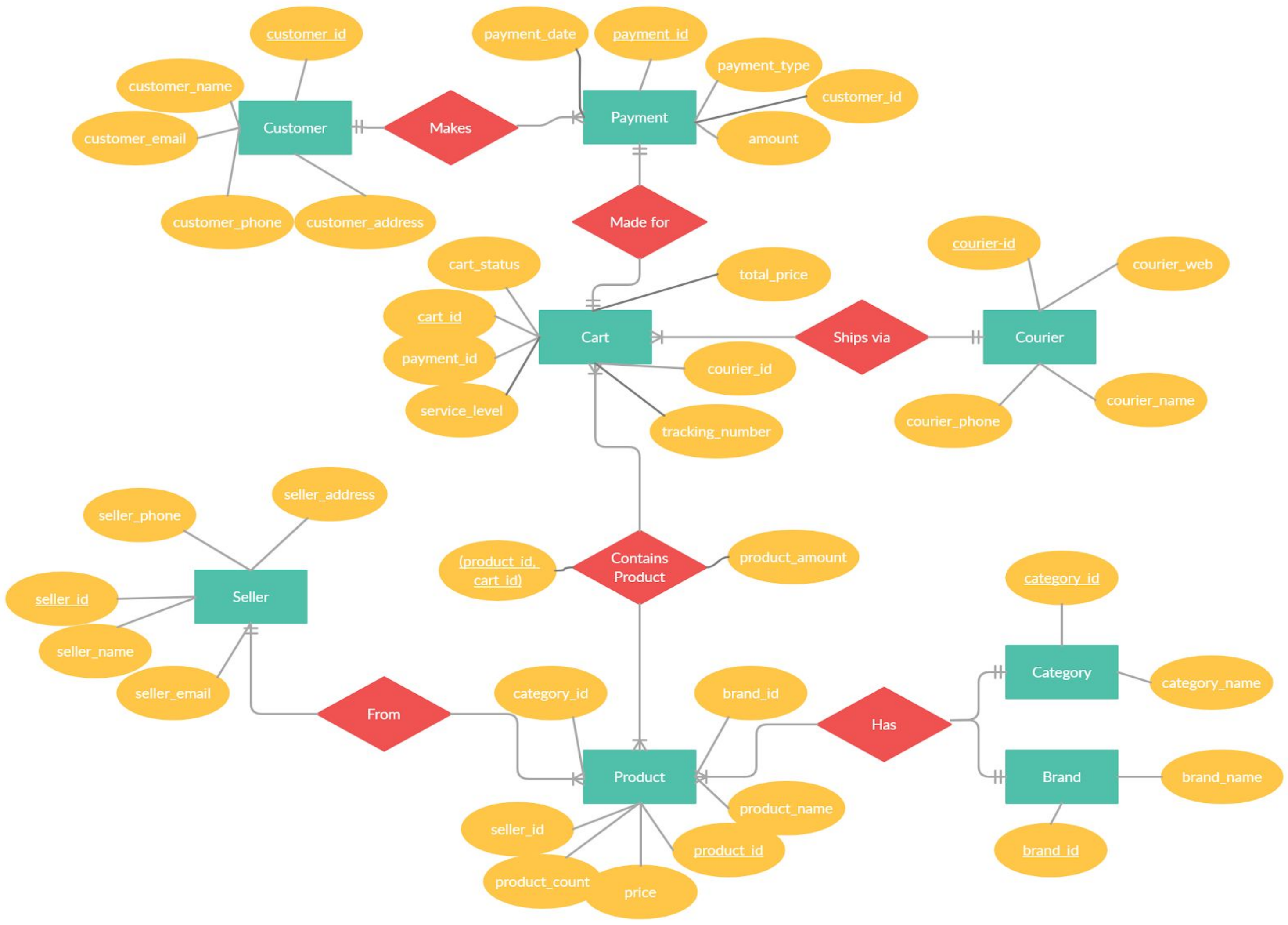
5-10 entities: User, Customer, Order, Product, Seller, Payment, Supplier, Courier, Category

ER Diagram

Initial ER diagram:



ER Diagram revised:



Schema Design

Initial schema/ Create Tables Code:

```
CREATE TABLE customer(
    customer_id NUMBER PRIMARY KEY,
    customer_name VARCHAR2(128) NOT NULL,
    customer_email VARCHAR2(255) NOT NULL ,
    customer_phone VARCHAR2(12),
    customer_address VARCHAR2(255) NOT NULL
    );

CREATE TABLE makes_payment(
    customer_id NUMBER REFERENCES customer(customer_id),
    payment_id NUMBER REFERENCES payment(payment_id),
    PRIMARY KEY (customer_id, payment_id)
    );

CREATE TABLE payment (
    payment_id NUMBER PRIMARY KEY,
    payment_type VARCHAR2(10),
    amount DECIMAL(10,2)
    );

CREATE TABLE payment_for(
    payment_id NUMBER REFERENCES payment(payment_id),
    cart_id NUMBER REFERENCES cart(cart_id),
    PRIMARY KEY (payment_id, cart_id)
    );

CREATE TABLE cart (
    cart_id NUMBER PRIMARY KEY,
    cart_status VARCHAR2(15),
    cart_date DATE,
    total_price DECIMAL(10,2)
    );

CREATE TABLE ships_via(
    courier_id NUMBER REFERENCES courier(courier_id),
    cart_id NUMBER REFERENCES cart(cart_id),
    PRIMARY KEY (courier_id, cart_id)
    );

CREATE TABLE courier (
    courier_id NUMBER PRIMARY KEY,
    courier_web VARCHAR2(100),
    service_level VARCHAR2(25),
    tracking_number VARCHAR2(40),
    courier_name VARCHAR2(50),
    courier_phone VARCHAR2(12)
    );

CREATE TABLE contains_product(
    product_id NUMBER REFERENCES product(product_id),
    cart_id NUMBER REFERENCES cart(cart_id),
    PRIMARY KEY (product_id, cart_id)
    );

CREATE TABLE product (
    product_id NUMBER PRIMARY KEY,
    product_name VARCHAR2(100),
    amount INTEGER,
    price DECIMAL(10,2)
    );

CREATE TABLE product_from(
    product_id NUMBER REFERENCES product(product_id),
    seller_id NUMBER REFERENCES seller(seller_id),
    PRIMARY KEY (product_id, seller_id)
    );

CREATE TABLE seller (
    seller_id NUMBER PRIMARY KEY,
    seller_address VARCHAR2(255),
    seller_phone VARCHAR2(25),
    seller_name VARCHAR2(128),
    seller_email VARCHAR2(254)
    );

CREATE TABLE has_category(
    product_id NUMBER REFERENCES product(product_id),
    category_id NUMBER REFERENCES product_category(category_id),
    PRIMARY KEY (product_id, category_id)
    );

CREATE TABLE product_category (
    category_id NUMBER PRIMARY KEY,
    category_name VARCHAR2(100)
    );
```

```
CREATE TABLE has_brand(
    product_id NUMBER REFERENCES product(product_id),
    brand_id NUMBER REFERENCES brand(brand_id),
    PRIMARY KEY (product_id, brand_id)
    );

CREATE TABLE brand (
    brand_id NUMBER PRIMARY KEY,
    brand_name VARCHAR2(100)
    );
```

Revised Schema Design/ Create Tables Code:

```
CREATE TABLE customer(
    customer_id NUMBER PRIMARY KEY,
    customer_name VARCHAR2(128) NOT NULL,
    customer_email VARCHAR2(255) NOT NULL ,
    customer_phone VARCHAR2(12) NOT NULL,
    customer_address VARCHAR2(255) NOT NULL
    );

CREATE TABLE courier (
    courier_id NUMBER PRIMARY KEY,
    courier_web VARCHAR2(100) NOT NULL,
    courier_name VARCHAR2(50) NOT NULL,
    courier_phone VARCHAR2(12) NOT NULL
    );

CREATE TABLE seller (
    seller_id NUMBER PRIMARY KEY,
    seller_address VARCHAR2(255) NOT NULL,
    seller_phone VARCHAR2(25) NOT NULL,
    seller_name VARCHAR2(128) NOT NULL,
    seller_email VARCHAR2(254) NOT NULL
    );

CREATE TABLE product_category (
    category_id NUMBER PRIMARY KEY,
    category_name VARCHAR2(100) NOT NULL
    );

CREATE TABLE brand (
    brand_id NUMBER PRIMARY KEY,
    brand_name VARCHAR2(100) NOT NULL
    );

CREATE TABLE payment (
    payment_id NUMBER PRIMARY KEY,
    payment_date DATE DEFAULT SYSDATE NOT NULL,
    payment_type VARCHAR2(10) NOT NULL,
    amount DECIMAL(10,2) NOT NULL,
    customer_id NUMBER REFERENCES customer(customer_id) ON DELETE
CASCADE
    );

CREATE TABLE cart (
    cart_id NUMBER PRIMARY KEY,
    cart_status VARCHAR2(15) DEFAULT 'Processing',
    total_price DECIMAL(10,2) NOT NULL,
    courier_id NUMBER REFERENCES courier(courier_id) ON DELETE
CASCADE,
    payment_id NUMBER REFERENCES payment(payment_id) ON DELETE
CASCADE,
    service_level VARCHAR2(25) NOT NULL,
    tracking_number VARCHAR2(40)
    );

CREATE TABLE product (
    product_id NUMBER PRIMARY KEY,
    product_name VARCHAR2(100) NOT NULL,
    product_count NUMBER NOT NULL CHECK (product_count >= 0),
    price DECIMAL(10,2) NOT NULL CHECK (price > 0),
    seller_id NUMBER REFERENCES seller(seller_id) ON DELETE CASCADE,
    brand_id NUMBER REFERENCES brand(brand_id) ON DELETE CASCADE,
    category_id NUMBER REFERENCES product_category(category_id) ON
DELETE CASCADE
    );

CREATE TABLE contains_product(
    product_id NUMBER REFERENCES product(product_id) ON DELETE
CASCADE,
    cart_id NUMBER REFERENCES cart(cart_id) ON DELETE CASCADE,
    product_amount NUMBER CHECK (product_amount > 0),
    PRIMARY KEY (product_id, cart_id)
    );
```

Demo of Designing Views/Simple Queries

```
/*inserting rows of mock data*/
--Customers:
INSERT INTO customer (customer_id, customer_name, customer_email,
customer_phone, customer_address) VALUES (1, 'J Weber',
'jweber@gmail.com', '123456789', '123 main street');
INSERT INTO customer (customer_id, customer_name, customer_email,
customer_phone, customer_address) VALUES (2, 'Ms Feeld',
'feeld@gmail.com', '555156189', '24 oak street');
INSERT INTO customer (customer_id, customer_name, customer_email,
customer_phone, customer_address) VALUES (3, 'Mr Saad',
'saad1@gmail.com', '555999848', '33 welesely rd');
INSERT INTO customer (customer_id, customer_name, customer_email,
customer_phone, customer_address) VALUES (4, 'Cashwan',
'cashwan@gmail.com', '555444888', '18 river rd');
INSERT INTO customer (customer_id, customer_name, customer_email,
customer_phone, customer_address) VALUES (5, 'Betty DeLile',
'bdlile@gmail.com', '555841599', '906 cam ave');

--Couriers:
INSERT INTO courier (courier_id, courier_name, courier_web,
courier_phone ) VALUES (1, 'CanPost', 'canpost.ca', '180055555');
INSERT INTO courier (courier_id, courier_name, courier_web,
courier_phone ) VALUES (2, 'Fedex', 'fedex.com', '18005556515');
INSERT INTO courier (courier_id, courier_name, courier_web,
courier_phone ) VALUES (3, 'UPS', 'ups.com', '18005558954');

--Sellers:
INSERT INTO seller (seller_id, seller_name, seller_address,
seller_phone, seller_email) VALUES (1, 'Max Computers', '123 Max
Ave', '1800555755', 'contact@maxcomp.com');
INSERT INTO seller (seller_id, seller_name, seller_address,
seller_phone, seller_email) VALUES (2, 'Park Vintage', 'Bud St',
'9068526268', 'info@parkvintage.com');
INSERT INTO seller (seller_id, seller_name, seller_address,
seller_phone, seller_email) VALUES (3, 'Min Computers', '321 Min St',
'815784658', 'contact@mincomp.com');
INSERT INTO seller (seller_id, seller_name, seller_address,
seller_phone, seller_email) VALUES (4, 'ToysRUs', '189 Toy Place',
'18885558181', 'cs@toyrus.com');
INSERT INTO seller (seller_id, seller_name, seller_address,
seller_phone, seller_email) VALUES (5, 'ABC Books', '89 John St',
'8159265484', 'abc@yahoo.com');

--Product Categories:
INSERT INTO product_category (category_id, category_name) VALUES (1,
'Computer');
INSERT INTO product_category (category_id, category_name) VALUES (2,
'Clothing');
```

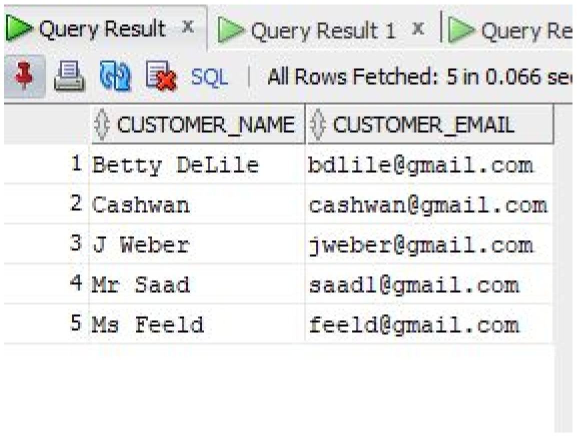
```
INSERT INTO product_category (category_id, category_name) VALUES (3,
'Toys');
INSERT INTO product_category (category_id, category_name) VALUES (4,
'Books');
INSERT INTO product_category (category_id, category_name) VALUES (5,
'Games');

--Brands:
INSERT INTO brand (brand_id, brand_name) VALUES (1, 'Apple');
INSERT INTO brand (brand_id, brand_name) VALUES (2, 'Microsoft');
INSERT INTO brand (brand_id, brand_name) VALUES (3, 'Hot Wheels');
INSERT INTO brand (brand_id, brand_name) VALUES (4, 'Penguin Books');
INSERT INTO brand (brand_id, brand_name) VALUES (5, 'Nintendo');
INSERT INTO brand (brand_id, brand_name) VALUES (6, 'Vintage');


--Products:
INSERT INTO product (product_id, product_name, product_count, price,
brand_id, category_id, seller_id) VALUES (1, 'Apple Monitor', 20,
4000.00, 1, 1, 1);
INSERT INTO product (product_id, product_name, product_count, price,
brand_id, category_id, seller_id) VALUES (2, 'Microsoft Keyboard',
50, 75.00, 2, 1, 3);
INSERT INTO product (product_id, product_name, product_count, price,
brand_id, category_id, seller_id) VALUES (3, 'Hot Wheels Track Set',
15, 55.00, 3, 3, 4);
INSERT INTO product (product_id, product_name, product_count, price,
brand_id, category_id, seller_id) VALUES (4, 'Coffee Table Book', 20,
10.00, 4, 4, 5);
INSERT INTO product (product_id, product_name, product_count, price,
brand_id, category_id, seller_id) VALUES (5, 'Zelda', 40, 70.00, 5,
5, 4);
INSERT INTO product (product_id, product_name, product_count, price,
brand_id, category_id, seller_id) VALUES (6, 'Vintage 80s Military
Jacket M', 10, 25.00, 6, 2, 2);

--Payments/Carts/Contains_Product:
INSERT INTO payment (payment_id, payment_date, payment_type, amount,
customer_id) VALUES (1, '2020-10-5', 'Visa', 4000.00, 1);
INSERT INTO cart (cart_id, cart_status, total_price, courier_id,
payment_id, service_level, tracking_number) VALUES (1, 'Shipped',
4000.00, 1, 1, 'Regular', 'xn151890op');
INSERT INTO contains_product (product_id, cart_id, product_amount)
VALUES (1, 1, 1);

INSERT INTO payment (payment_id, payment_date, payment_type, amount,
customer_id) VALUES (2, '2020-10-20', 'MasterCard', 125.00, 2);
INSERT INTO cart (cart_id, cart_status, total_price, courier_id,
payment_id, service_level, tracking_number) VALUES (2, 'Processing',
125.00, 2, 2, 'Regular', '0605481028700');
INSERT INTO contains_product (product_id, cart_id, product_amount)
VALUES (3, 2, 1);
```

```
INSERT INTO contains_product (product_id, cart_id, product_amount)
VALUES (5, 2, 1);

INSERT INTO payment (payment_id, payment_date, payment_type, amount,
customer_id) VALUES (3, '2020-10-18', 'Interac', 70.00, 3);
INSERT INTO cart (cart_id, cart_status, total_price, courier_id,
payment_id, service_level, tracking_number) VALUES (3, 'Shipped',
70.00, 3, 3, 'Express', 'EX8189156');
INSERT INTO contains_product (product_id, cart_id, product_amount)
VALUES (6, 3, 1);

INSERT INTO payment (payment_id, payment_date, payment_type, amount,
customer_id) VALUES (4, '2020-10-01', 'PayPal', 85.00, 4);
INSERT INTO cart (cart_id, cart_status, total_price, courier_id,
payment_id, service_level, tracking_number) VALUES (4, 'Shipped',
85.00, 1, 4, 'Xpress', 'X81865189EN');
INSERT INTO contains_product (product_id, cart_id, product_amount)
VALUES (2, 4, 1);
INSERT INTO contains_product (product_id, cart_id, product_amount)
VALUES (4, 4, 1);

--more data
INSERT INTO product (product_id, product_name, product_count, price,
brand_id, category_id, seller_id) VALUES (7, 'Vintage Nursery Rhymes
Book', 5, 5.00, 6, 4, 2);
INSERT INTO payment (payment_id, payment_date, payment_type, amount,
customer_id) VALUES (5, '2020-10-28', 'Apple Pay', 60.00, 5);
INSERT INTO cart (cart_id, cart_status, total_price, courier_id,
payment_id, service_level, tracking_number) VALUES (5, 'Shipped',
60.00, 2, 5, 'Xpress', 'WJKX8518');
INSERT INTO contains_product (product_id, cart_id, product_amount)
VALUES (6, 5, 2);
INSERT INTO contains_product (product_id, cart_id, product_amount)
VALUES (4, 5, 1);
/*simple queries*/
SELECT customer_name, customer_email
FROM customer
ORDER BY customer_name;

SELECT courier_name, courier_web, courier_phone
FROM courier
ORDER BY courier_name;

SELECT seller_name, seller_email, seller_phone, seller_address
FROM seller
ORDER BY seller_name;

SELECT category_name
FROM product_category
ORDER BY category_name;

SELECT brand_name
```

```
FROM brand
ORDER BY brand_name;


/*more advanced join queries */
SELECT customer_name, amount, payment_type, payment_date
FROM customer, payment
WHERE customer.customer_id = payment.customer_id
ORDER BY payment_date;

SELECT product_name, product_count, price, seller_name
FROM product, seller
WHERE product.seller_id = seller.seller_id
ORDER BY product_name;

SELECT customer_name, cart_status, tracking_number
FROM customer, cart, payment
WHERE cart.payment_id = payment.payment_id
AND payment.customer_id = customer.customer_id
ORDER BY customer_name;

SELECT customer_name, product_name, product_amount
FROM customer, product, contains_product, cart, payment
WHERE contains_product.product_id = product.product_id
AND contains_product.cart_id = cart.cart_id
AND cart.payment_id = payment.payment_id
AND payment.customer_id = customer.customer_id
ORDER BY customer_name;

--VIEWS
--Potential sale of product if you have a lot of extra inventory,
amount of product is greater than the average product count of all
products
CREATE VIEW products_to_put_on_sale (discounted_productName,
discounted_productCount, original_price) AS
(SELECT product_name, product_count, price
FROM product
WHERE product_count>(
SELECT AVG(product_count)
FROM product));
--Customer that bought the most vintage jackets, could be customized
to show all products top shopper
CREATE VIEW VJ_Number_One_Shopper (Number_One_Customer) AS
SELECT customer_name
FROM customer, payment, cart, contains_product
WHERE product_amount =(
SELECT MAX(product_amount)
FROM contains_product
WHERE product_id = 6)
AND product_id = 6
AND contains_product.cart_id = cart.cart_id
AND cart.payment_id = payment.payment_id
AND payment.customer_id = customer.customer_id;
--Customer's who's orders are still being processed
CREATE VIEW need_to_be_shipped(cust_name, cust_address, track_num,
sell_name, sell_phone)AS
SELECT DISTINCT customer_name, customer_address, tracking_number,
seller_name, seller_phone
FROM customer, payment, cart, contains_product, product, seller
WHERE cart.payment_id = payment.payment_id
AND payment.customer_id = customer.customer_id
AND cart_status = 'Processing'
AND cart.cart_id = contains_product.cart_id
AND contains_product.product_id = product.product_id
AND product.seller_id = seller.seller_id

/*advanced queries for assignment 5 */
--Advanced join query
SELECT DISTINCT product_name, brand_name, category_name, seller_name
FROM product, product_category, brand, seller
WHERE product.brand_id = brand.brand_id
AND product.category_id = product_category.category_id
AND product.seller_id = seller.seller_id
ORDER BY product_name;

--Customers who bought Vintage 80s OR bought coffee table
SELECT customer_name
FROM customer, product, contains_product, cart, payment
WHERE contains_product.product_id = product.product_id
AND contains_product.cart_id = cart.cart_id
AND cart.payment_id = payment.payment_id
AND payment.customer_id = customer.customer_id
AND product_name = 'Vintage 80s Military Jacket M'
UNION
(SELECT customer_name
FROM customer, product, contains_product, cart, payment
WHERE contains_product.product_id = product.product_id
AND contains_product.cart_id = cart.cart_id
AND cart.payment_id = payment.payment_id
AND payment.customer_id = customer.customer_id
AND product_name = 'Coffee Table Book');

--Aggregate function, # of items sold
SELECT product_name, COUNT(*)
FROM contains_product, product
WHERE contains_product.product_id = product.product_id
GROUP BY product_name;

--Products that cost above $20 and $70 or under
SELECT product_name
FROM product
WHERE price BETWEEN 20 AND 70;

--Customers who have their products shipped by couriers other than
```

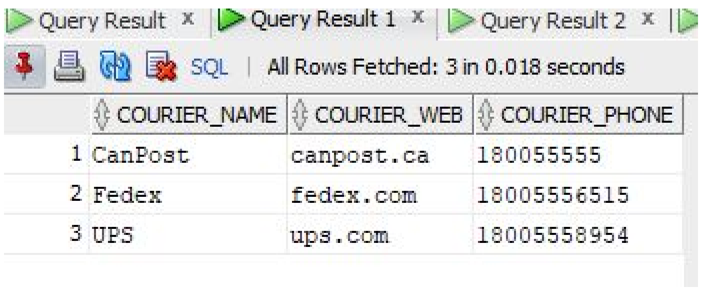
```
Canpost
SELECT customer_name, cart_status
FROM customer, cart, payment
WHERE cart.payment_id = payment.payment_id
AND payment.customer_id = customer.customer_id
AND cart_status = 'Shipped'
AND NOT EXISTS
(SELECT *
FROM courier
WHERE courier_id = 1
AND cart.courier_id = courier.courier_id);
--Products sold by ToysRUs
SELECT product_name
FROM product
WHERE NOT EXISTS
(SELECT *
FROM seller
WHERE seller_id <> 4
AND product.seller_id = seller.seller_id);
```

Query Results:



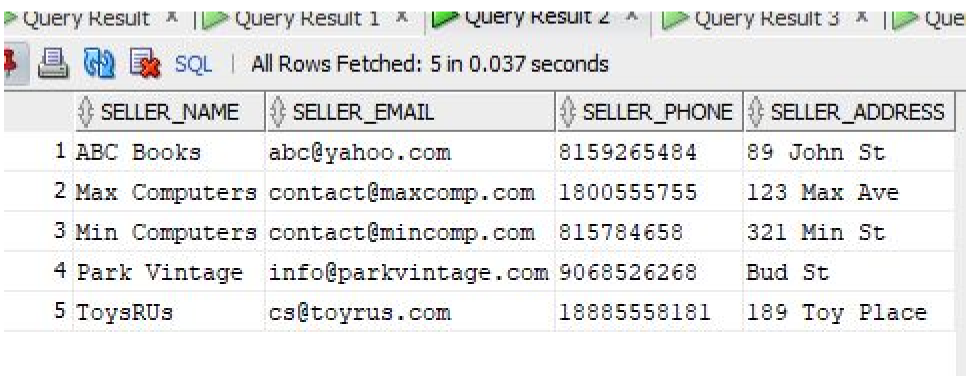
All Rows Fetched: 5 in 0.066 se

	CUSTOMER_NAME	CUSTOMER_EMAIL
1	Betty DeLile	bdlile@gmail.com
2	Cashwan	cashwan@gmail.com
3	J Weber	jweber@gmail.com
4	Mr Saad	saadl@gmail.com
5	Ms Feeld	feeld@gmail.com



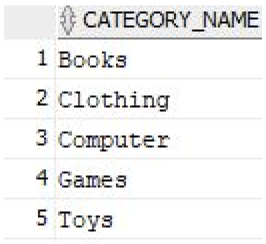
All Rows Fetched: 3 in 0.018 seconds

	COURIER_NAME	COURIER_WEB	COURIER_PHONE
1	CanPost	canpost.ca	180055555
2	Fedex	fedex.com	18005556515
3	UPS	ups.com	18005558954

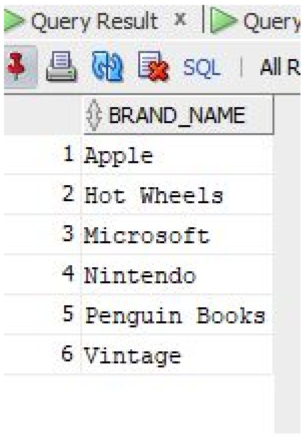


All Rows Fetched: 5 in 0.037 seconds

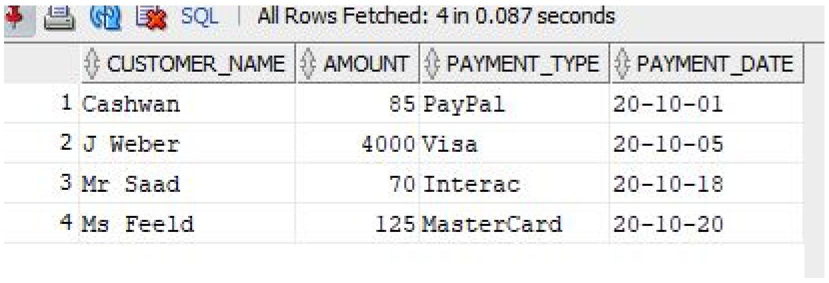
	SELLER_NAME	SELLER_EMAIL	SELLER_PHONE	SELLER_ADDRESS
1	ABC Books	abc@yahoo.com	8159265484	89 John St
2	Max Computers	contact@maxcomp.com	1800555755	123 Max Ave
3	Min Computers	contact@mincomp.com	815784658	321 Min St
4	Park Vintage	info@parkvintage.com	9068526268	Bud St
5	ToysRUs	cs@toyrus.com	18885558181	189 Toy Place



	CATEGORY_NAME
1	Books
2	Clothing
3	Computer
4	Games
5	Toys

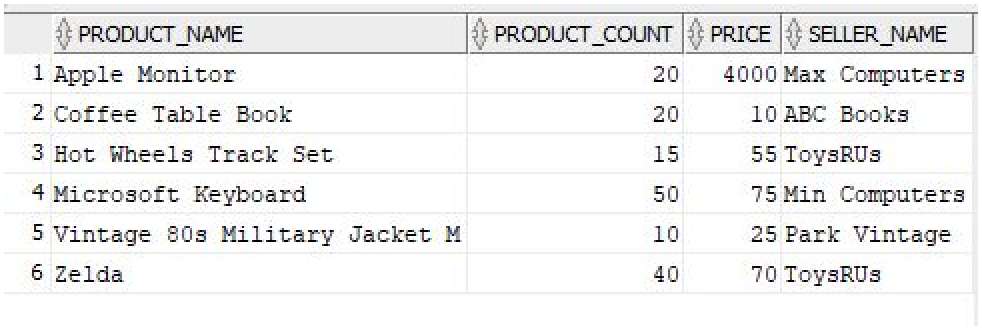


	BRAND_NAME
1	Apple
2	Hot Wheels
3	Microsoft
4	Nintendo
5	Penguin Books
6	Vintage

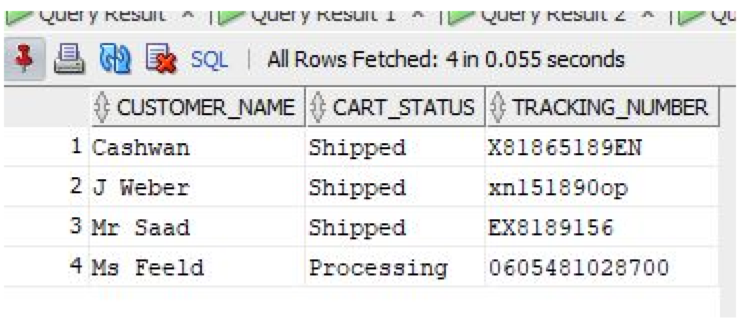


All Rows Fetched: 4 in 0.087 seconds

	CUSTOMER_NAME	AMOUNT	PAYMENT_TYPE	PAYMENT_DATE
1	Cashwan	85	PayPal	20-10-01
2	J Weber	4000	Visa	20-10-05
3	Mr Saad	70	Interac	20-10-18
4	Ms Feeld	125	MasterCard	20-10-20



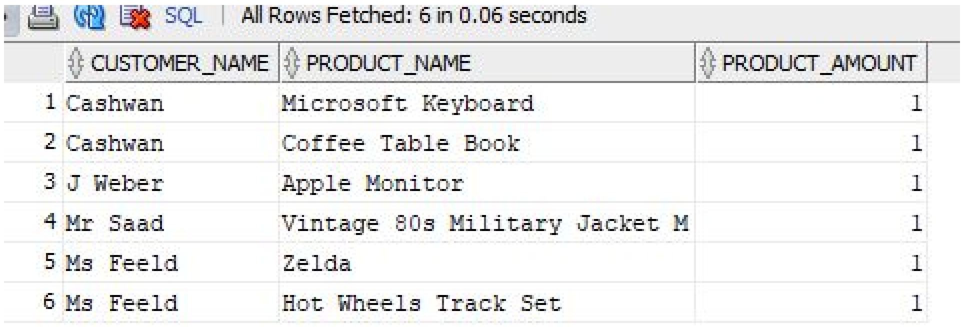
	PRODUCT_NAME	PRODUCT_COUNT	PRICE	SELLER_NAME
1	Apple Monitor	20	4000	Max Computers
2	Coffee Table Book	20	10	ABC Books
3	Hot Wheels Track Set	15	55	ToysRUs
4	Microsoft Keyboard	50	75	Min Computers
5	Vintage 80s Military Jacket M	10	25	Park Vintage
6	Zelda	40	70	ToysRUs



Query Result × | Query Result 1 × | Query Result 2 × | Qu

SQL | All Rows Fetched: 4 in 0.055 seconds

	CUSTOMER_NAME	CART_STATUS	TRACKING_NUMBER
1	Cashwan	Shipped	X81865189EN
2	J Weber	Shipped	xn151890op
3	Mr Saad	Shipped	EX8189156
4	Ms Feeld	Processing	0605481028700



SQL | All Rows Fetched: 6 in 0.06 seconds

	CUSTOMER_NAME	PRODUCT_NAME	PRODUCT_AMOUNT
1	Cashwan	Microsoft Keyboard	1
2	Cashwan	Coffee Table Book	1
3	J Weber	Apple Monitor	1
4	Mr Saad	Vintage 80s Military Jacket M	1
5	Ms Feeld	Zelda	1
6	Ms Feeld	Hot Wheels Track Set	1

UNIX Shell Implementation

Note: Real username and password were used in actual code.

Multiple bash script files were made to execute sqlplus64 commands to connect to the database and execute sql commands and queries within the UNIX shell. The sql commands are the same as the ones used before.

Menu.sh source code:

```
#!/bin/sh
MainMenu()
{

      while [ "$CHOICE" != "START" ]
      do
            clear
            echo
"==================================================================="
            echo "| Oracle All Inclusive Tool|"
            echo "| Main Menu - Select Desired Operation(s):|"
            echo "| <CTRL-Z Anytime to Enter Interactive CMD Prompt>|"
            echo
"-------------------------------------------------------------------"
            echo " $IS_SELECTED1 1) Drop Tables"
            echo " $IS_SELECTED2 2) Create Tables"
            echo " $IS_SELECTED3 3) Populate Tables"
            echo " $IS_SELECTED4 4) Query Tables"
            echo " "
            echo " $IS_SELECTEDE E) End/Exit"
            echo "Choose: "

            read CHOICE

            if [ "$CHOICE" == "0" ]
            then
                  echo "Nothing Here"

            elif [ "$CHOICE" == "1" ]
            then
                  bash drop_tables.sh
                  Pause


            elif [ "$CHOICE" == "2" ]
            then
                  bash create_tables.sh
                  Pause

            elif [ "$CHOICE" == "3" ]
            then
```

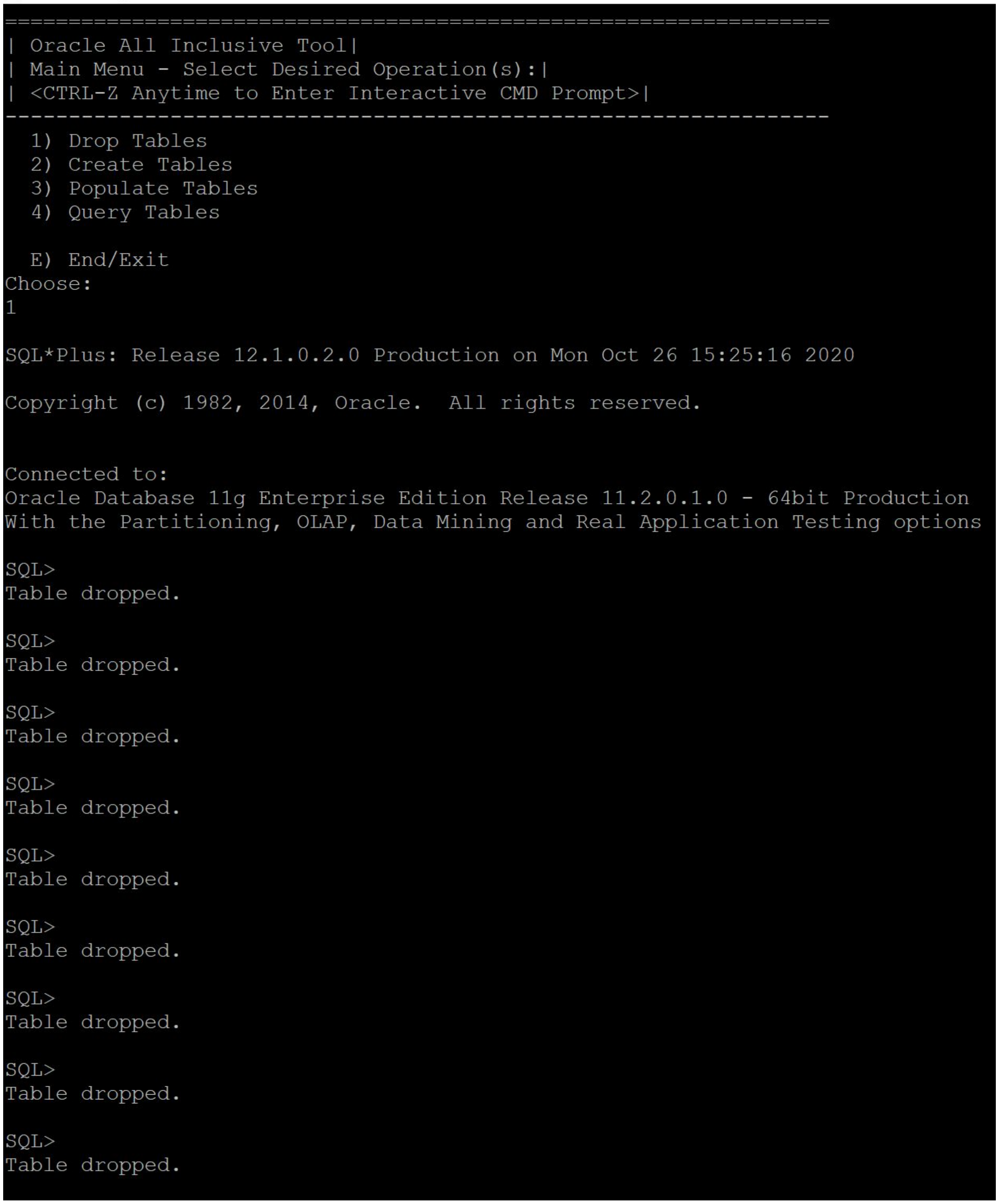
```
                bash populate_tables.sh
                Pause

           elif [ "$CHOICE" == "4" ]
           then
                bash queries.sh
                sleep 5
                Pause

           elif [ "$CHOICE" == "E" ]
           then
                exit
           fi
      done
}

#--COMMENTS BLOCK--
# Main Program
#--COMMENTS BLOCK--

ProgramStart()
{
     StartMessage
     while [ 1 ]
     do
          MainMenu

     done
}

ProgramStart
```

Drop_tables.sh Source Code:

```
#!/bin/sh
#export LD_LIBRARY_PATH=/usr/lib/oracle/12.1/client64/lib
sqlplus64
"username/password@(DESCRIPTION=(ADDRESS=(PROTOCOL=TCP)(Host=oracle.s
cs.ryerson.ca)(Port=1521))(CONNECT_DATA=(SID=orcl)))" <<EOF
DROP TABLE "CUSTOMER" CASCADE CONSTRAINTS;
DROP TABLE "COURIER" CASCADE CONSTRAINTS;
DROP TABLE "SELLER" CASCADE CONSTRAINTS;
DROP TABLE "PRODUCT_CATEGORY" CASCADE CONSTRAINTS;
DROP TABLE "BRAND" CASCADE CONSTRAINTS;
DROP TABLE "PRODUCT" CASCADE CONSTRAINTS;
DROP TABLE "PAYMENT" CASCADE CONSTRAINTS;
DROP TABLE "CART" CASCADE CONSTRAINTS;
DROP TABLE "CONTAINS_PRODUCT" CASCADE CONSTRAINTS;
exit;
EOF
```

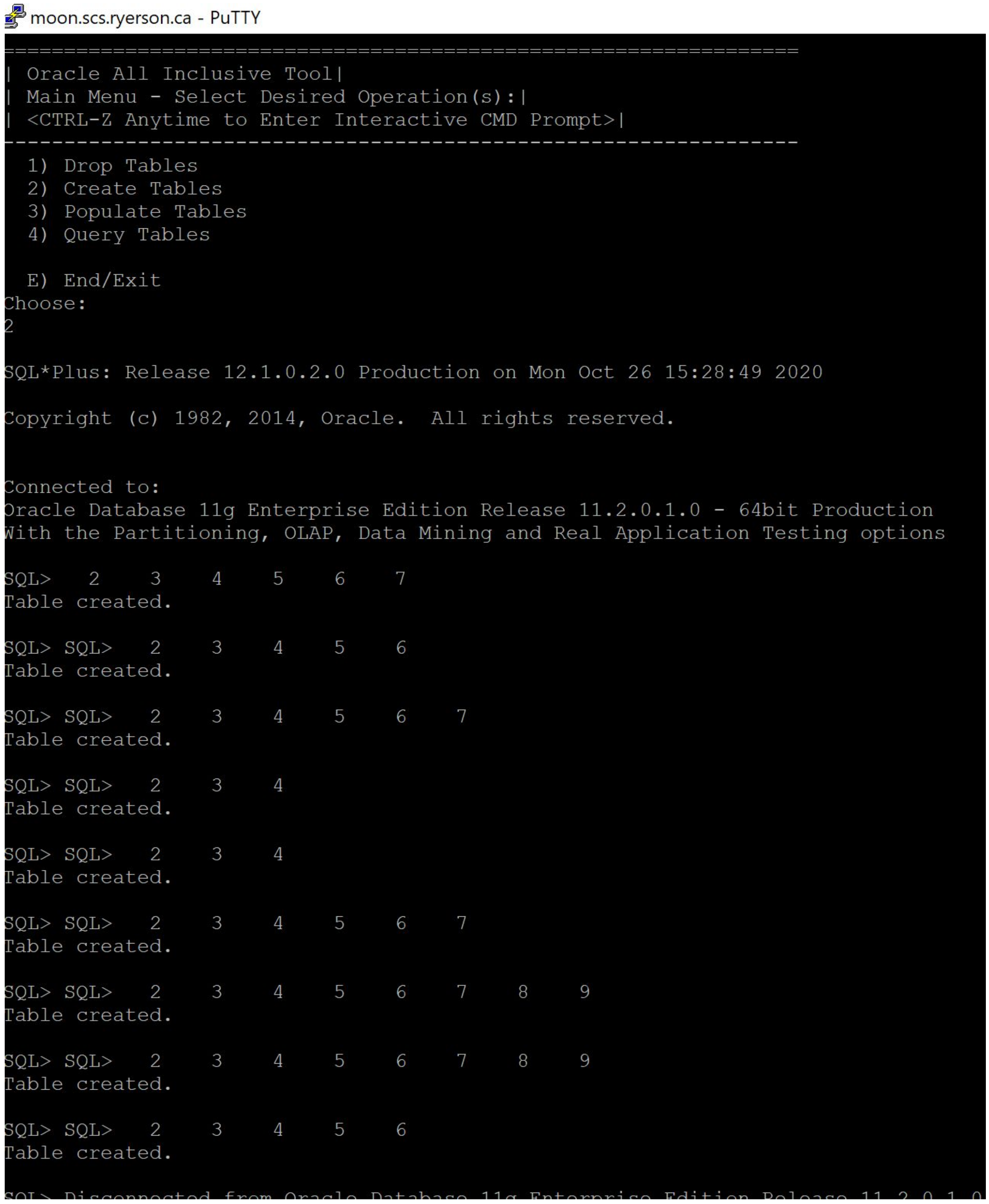
Dropping tables:

moon.scs.ryerson.ca - PuTTY



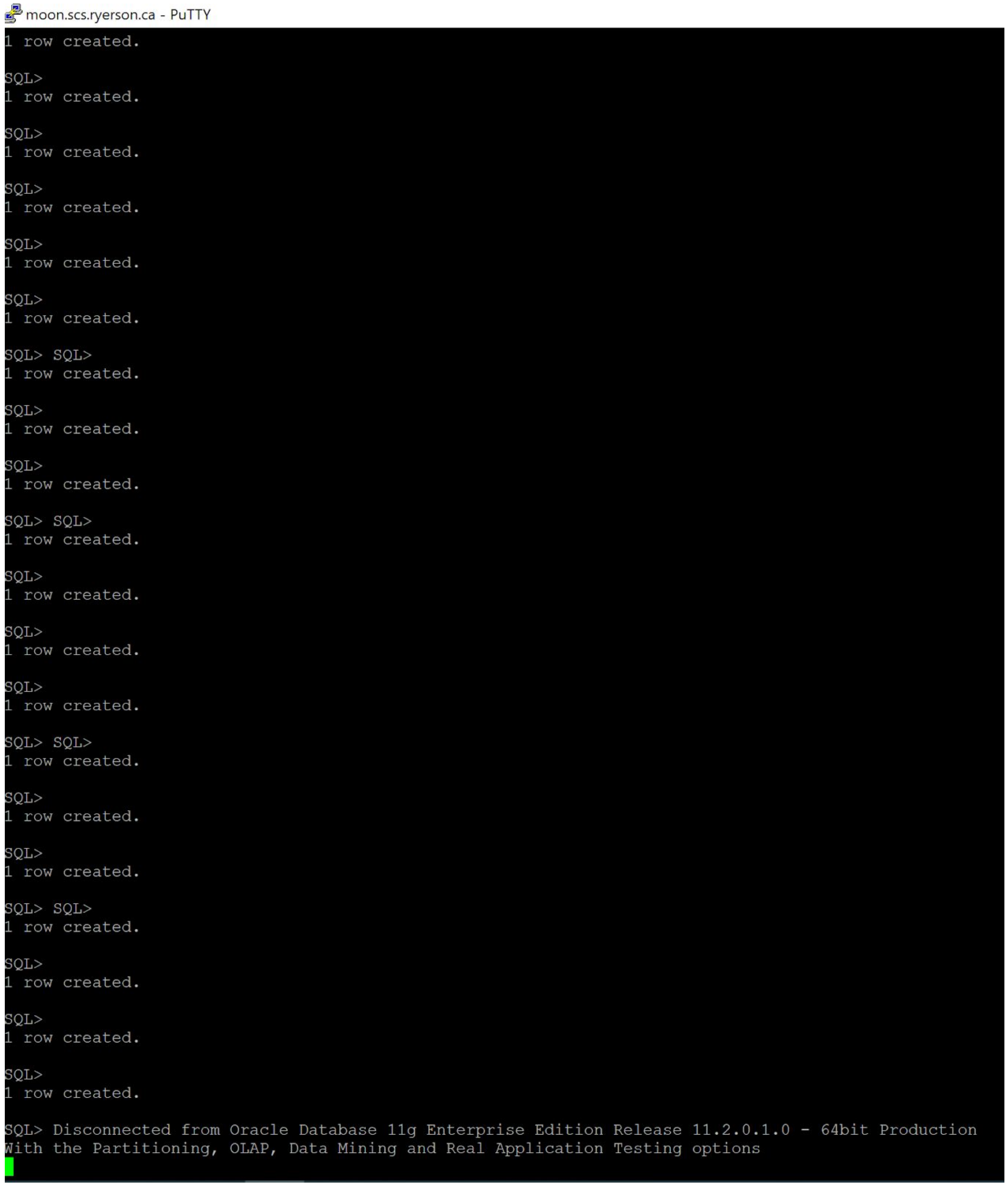
```
=================================================================
| Oracle All Inclusive Tool|
| Main Menu - Select Desired Operation(s):|
| <CTRL-Z Anytime to Enter Interactive CMD Prompt>|
-----------------------------------------------------------------
  1) Drop Tables
  2) Create Tables
  3) Populate Tables
  4) Query Tables

  E) End/Exit
Choose:
1

SQL*Plus: Release 12.1.0.2.0 Production on Mon Oct 26 15:25:16 2020

Copyright (c) 1982, 2014, Oracle.  All rights reserved.


Connected to:
Oracle Database 11g Enterprise Edition Release 11.2.0.1.0 - 64bit Production
With the Partitioning, OLAP, Data Mining and Real Application Testing options

SQL>
Table dropped.

SQL>
Table dropped.

SQL>
Table dropped.

SQL>
Table dropped.

SQL>
Table dropped.

SQL>
Table dropped.

SQL>
Table dropped.

SQL>
Table dropped.

SQL>
Table dropped.
```

Creating tables:


moon.scs.ryerson.ca - PuTTY

```
=================================================================
| Oracle All Inclusive Tool|
| Main Menu - Select Desired Operation(s):|
| <CTRL-Z Anytime to Enter Interactive CMD Prompt>|
-----------------------------------------------------------------
  1) Drop Tables
  2) Create Tables
  3) Populate Tables
  4) Query Tables

  E) End/Exit
Choose:
2

SQL*Plus: Release 12.1.0.2.0 Production on Mon Oct 26 15:28:49 2020

Copyright (c) 1982, 2014, Oracle.  All rights reserved.


Connected to:
Oracle Database 11g Enterprise Edition Release 11.2.0.1.0 - 64bit Production
With the Partitioning, OLAP, Data Mining and Real Application Testing options

SQL>   2    3    4    5    6    7
Table created.

SQL> SQL>   2    3    4    5    6
Table created.

SQL> SQL>   2    3    4    5    6    7
Table created.

SQL> SQL>   2    3    4
Table created.

SQL> SQL>   2    3    4
Table created.

SQL> SQL>   2    3    4    5    6    7
Table created.

SQL> SQL>   2    3    4    5    6    7    8    9
Table created.

SQL> SQL>   2    3    4    5    6    7    8    9
Table created.

SQL> SQL>   2    3    4    5    6
Table created.

SQL> Disconnected from Oracle Database 11g Enterprise Edition Release 11.2.0.1.0
```

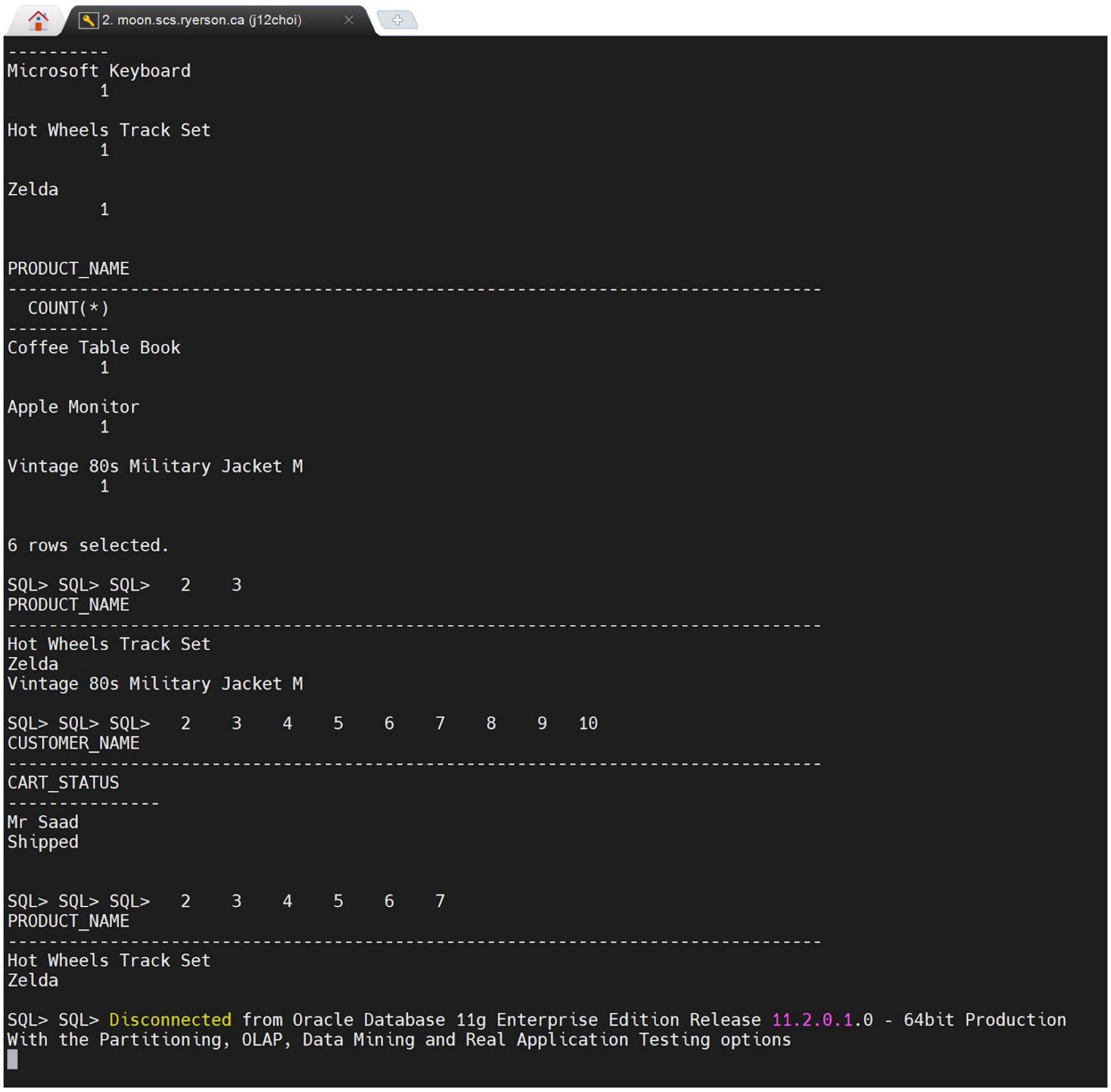
Populating tables:



moon.scs.ryerson.ca - PuTTY

```
1 row created.

SQL>
1 row created.

SQL>
1 row created.

SQL>
1 row created.

SQL>
1 row created.

SQL>
1 row created.

SQL> SQL>
1 row created.

SQL>
1 row created.

SQL>
1 row created.

SQL> SQL>
1 row created.

SQL>
1 row created.

SQL>
1 row created.

SQL>
1 row created.

SQL> SQL>
1 row created.

SQL>
1 row created.

SQL>
1 row created.

SQL> SQL>
1 row created.

SQL>
1 row created.

SQL>
1 row created.

SQL>
1 row created.

SQL> Disconnected from Oracle Database 11g Enterprise Edition Release 11.2.0.1.0 - 64bit Production
With the Partitioning, OLAP, Data Mining and Real Application Testing options
```

Queries:



2. moon.scs.ryerson.ca (j12choi)

```
----------
Microsoft Keyboard
         1

Hot Wheels Track Set
         1

Zelda
         1


PRODUCT_NAME
--------------------------------------------------------------------------------
  COUNT(*)
----------
Coffee Table Book
         1

Apple Monitor
         1

Vintage 80s Military Jacket M
         1


6 rows selected.

SQL> SQL> SQL>   2    3
PRODUCT_NAME
--------------------------------------------------------------------------------
Hot Wheels Track Set
Zelda
Vintage 80s Military Jacket M

SQL> SQL> SQL>   2    3    4    5    6    7    8    9   10
CUSTOMER_NAME
--------------------------------------------------------------------------------
CART_STATUS
---------------
Mr Saad
Shipped


SQL> SQL> SQL>   2    3    4    5    6    7
PRODUCT_NAME
--------------------------------------------------------------------------------
Hot Wheels Track Set
Zelda

SQL> SQL> Disconnected from Oracle Database 11g Enterprise Edition Release 11.2.0.1.0 - 64bit Production
With the Partitioning, OLAP, Data Mining and Real Application Testing options
```

Functional Dependencies/Normalization:

Customer:
Customer_id → customer_name, customer_email, customer_phone, customer_address

Courier:
Courier_id → courier_web, courier_name, courier_phone

Seller:
Seller_id → seller_address, seller_phone, seller_name, seller_email

Product_category:
Category_id → category_name

Brand:
Brand_id → brand_name

Payment:
Payment_id → payment_date, payment_type, amount, customer_id

Cart:
Cart_id → cart_status, total_price, courier_id, payment_id, service_level, tracking_number

Product:
Product_id → product_name, product_count, price, seller_id, brand_id, category_id

Contains_product:
(product_id, cart_id) → amount

Deriving 3NF

1st

Customer:

Customer_id → customer_name, customer_email, customer_phone, customer_address

2NF

Cust_Email:

Customer_email → customer_name, customer_address, customer_phone

Cust_ID:

Customer_id →customer_email, customer_phone

3NF

Cust_Email2:

Customer_email → customer_name, customer_address

Cust_Phone2:

Customer_phone →customer_email

Cust_ID2:

Customer_id →customer_phone

2nd

Courier:

Courier_id → courier_web, courier_name, courier_phone

2NF

Cou_ID**:**

Courier_id→ courier_name, courier_phone

Cou_ID2:

Courier_id→ courier_web

3NF

Cou_ID3

Courier_id→ courier_name

Cou_Phone

Courier_phone→ courier_name

Cou_ID2:

Courier_id→ courier_web

3rd

Seller:

Seller_id → seller_address, seller_phone, seller_name, seller_email

2NF

Sell_ID1:

Seller_id → seller_address, seller_phone, seller_name

Sell_Email:

Seller_email → seller_id

3NF

Sell_ID2:

Seller_id → seller_address, seller_name

Sell_Phone:

Seller_phone → Seller_id

Sell_Email:

Seller_email → seller_id

Others:

Product_category:

Category_id → category_name

Brand:

Brand_id → brand_name

Payment:

Payment_id → payment_date, payment_type, amount, customer_id

Cart:

Cart_id → cart_status, total_price, courier_id, payment_id, service_level, tracking_number

Product:

Product_id → product_name, product_count, price, seller_id, brand_id, category_id

Contains_product:

(product_id, cart_id) → amount

Deriving BCNF

1st

Customer:

customer_id → customer_name, customer_email, customer_phone, customer_address

2NF

Cust_Email:

customer_email → customer_name, customer_address, customer_phone

Cust_ID:

customer_id →customer_email, customer_phone

3NF

Cust_Email2:

customer_email → customer_name, customer_address

Cust_Phone2:

customer_phone →customer_email

Cust_ID2:

customer_id →customer_phone

BCNF

Steps:

Functional Dependencies are in the desired form.

Here customer_name can also determine customer_address despite being a non-prime attribute which is not allowed in BCNF.

Therefore, these two attributes cannot be together.

So, creating a new attribute and two separate table will make it in BCNF.

Cust_Email3

customer_email → address_id

Cust_Add:

address_id → customer_address, customer_name

2nd

Courier:

courier_id → courier_web, courier_name, courier_phone

2NF

Cou_ID**:**

courier_id→ courier_name, courier_phone

Cou_ID2:

courier_id→ courier_web

3NF

Cou_ID3

courier_id→ courier_name

Cou_Phone

courier_phone→ courier_name

Cou_ID2:

courier_id→ courier_web

3rd

Seller:

seller_id → seller_address, seller_phone, seller_name, seller_email

2NF

Sell_ID1:

seller_id → seller_address, seller_phone, seller_name

Sell_Email:

seller_email → seller_id

3NF

Sell_ID2:

seller_id → seller_address, seller_name

Sell_Phone:

seller_phone → Seller_id

Sell_Email:

seller_email → seller_id

BCNF

Steps:

Functional Dependencies are in the desired form.

Here seller_name can also determine seller_address despite being a non-prime attribute which is not allowed in BCNF.

Therefore, these two attributes cannot be together.

So, creating a new attribute and two separate table will make it in BCNF.

Sell_Email2

seller_email → selladdress_id

Sell_Add:

selladdress_id → seller_address, seller_name

Others:

Product_category:

category_id → category_name

Brand:

brand_id → brand_name

Payment:

payment_id → payment_date, payment_type, amount, customer_id

Cart:

cart_id → cart_status, total_price, courier_id, payment_id, service_level, tracking_number

Product:

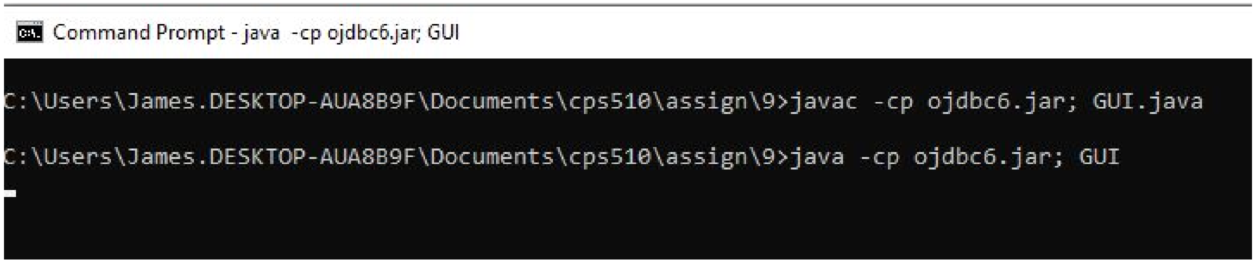
product_id → product_name, product_count, price, seller_id, brand_id, category_id

Contains_product:

(product_id, cart_id) → amount

Java Based User Interface:
Java GUI Screenshots:

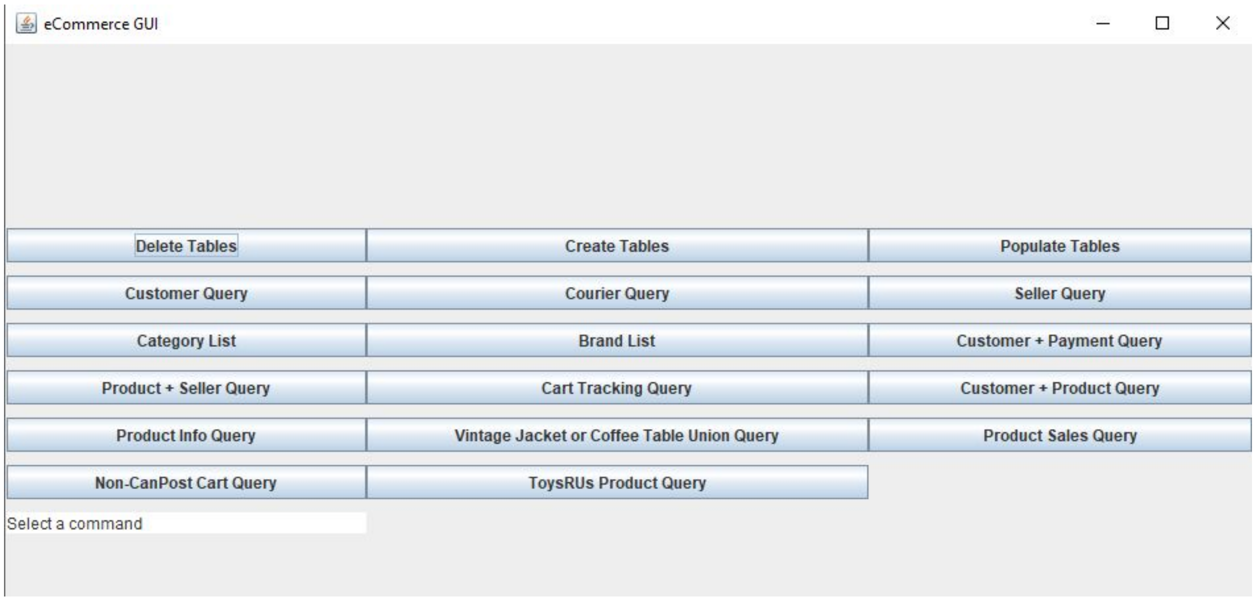
Compiling and running using ojdbc6.jar:



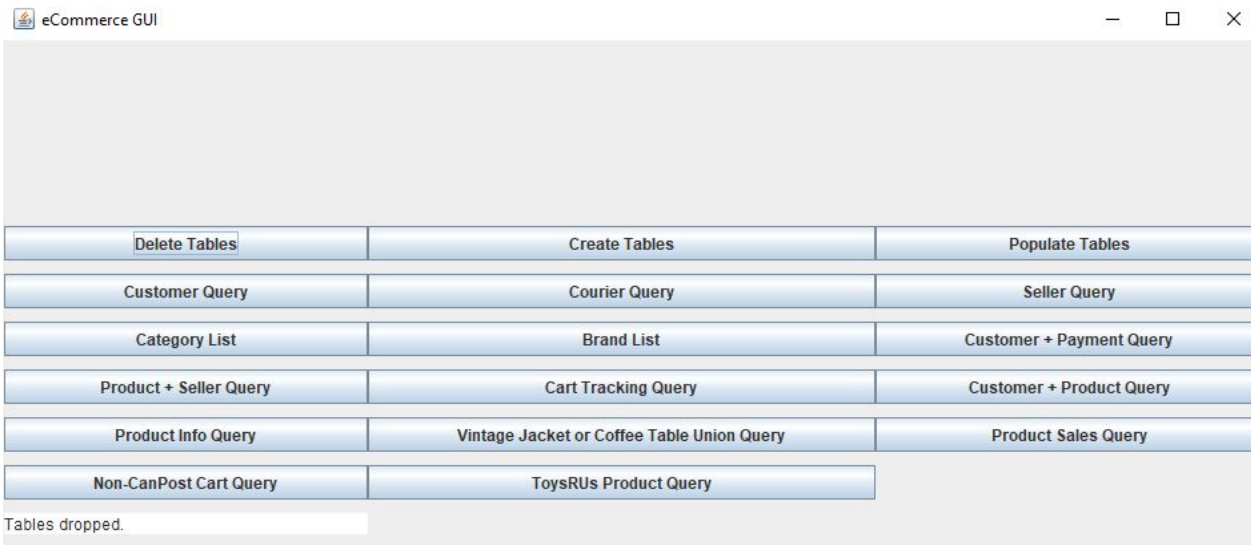
Command Prompt - java -cp ojdbc6.jar; GUI

```
C:\Users\James.DESKTOP-AUA8B9F\Documents\cps510\assign\9>javac -cp ojdbc6.jar; GUI.java

C:\Users\James.DESKTOP-AUA8B9F\Documents\cps510\assign\9>java -cp ojdbc6.jar; GUI
```

Java program launched:

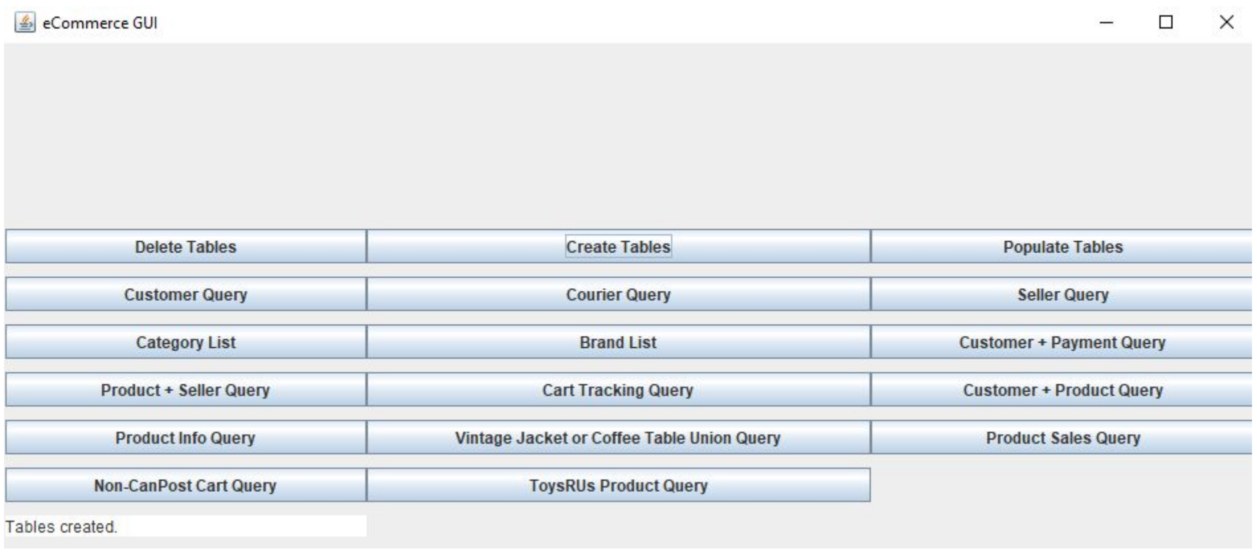


Tables dropped:



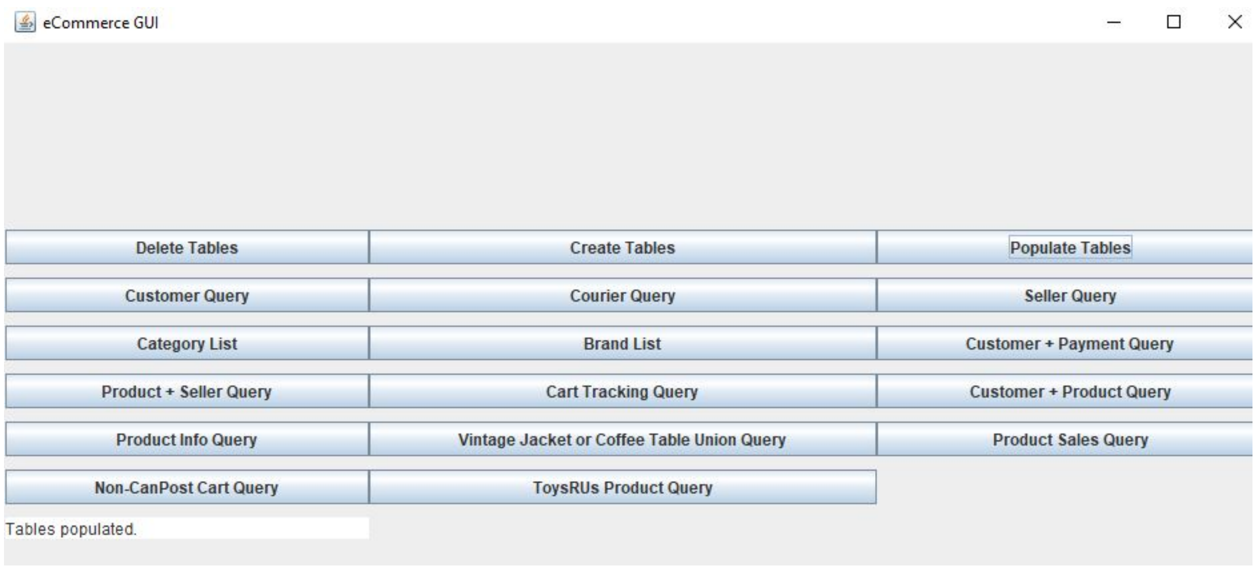
Tables dropped.

Tables created:



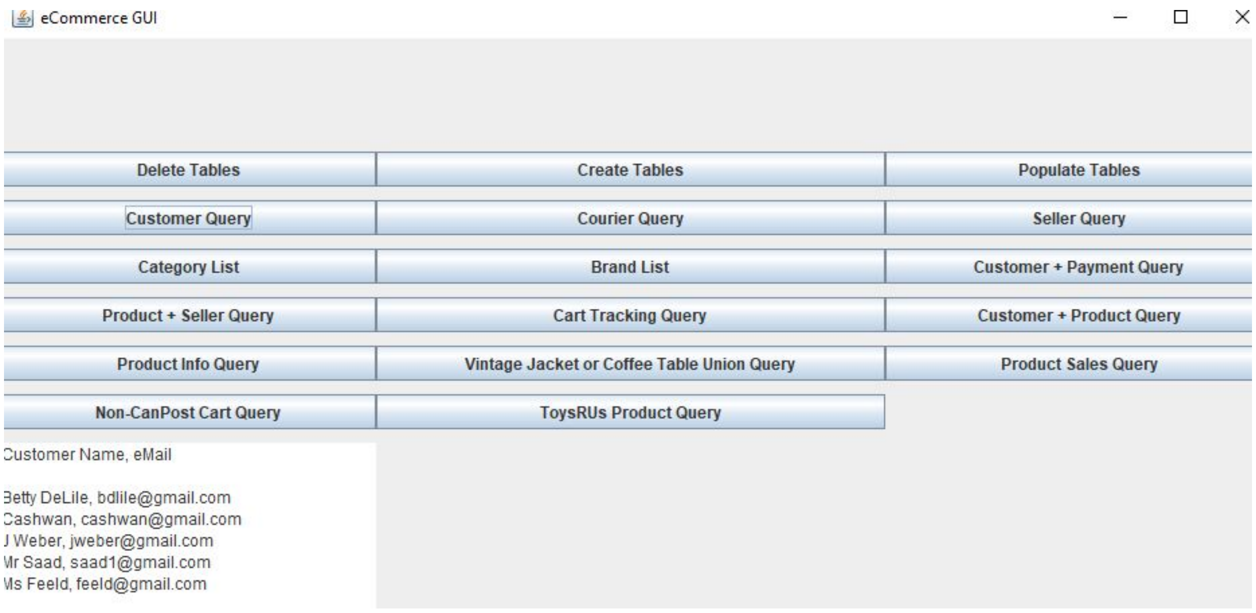
Tables created.

Tables populated:

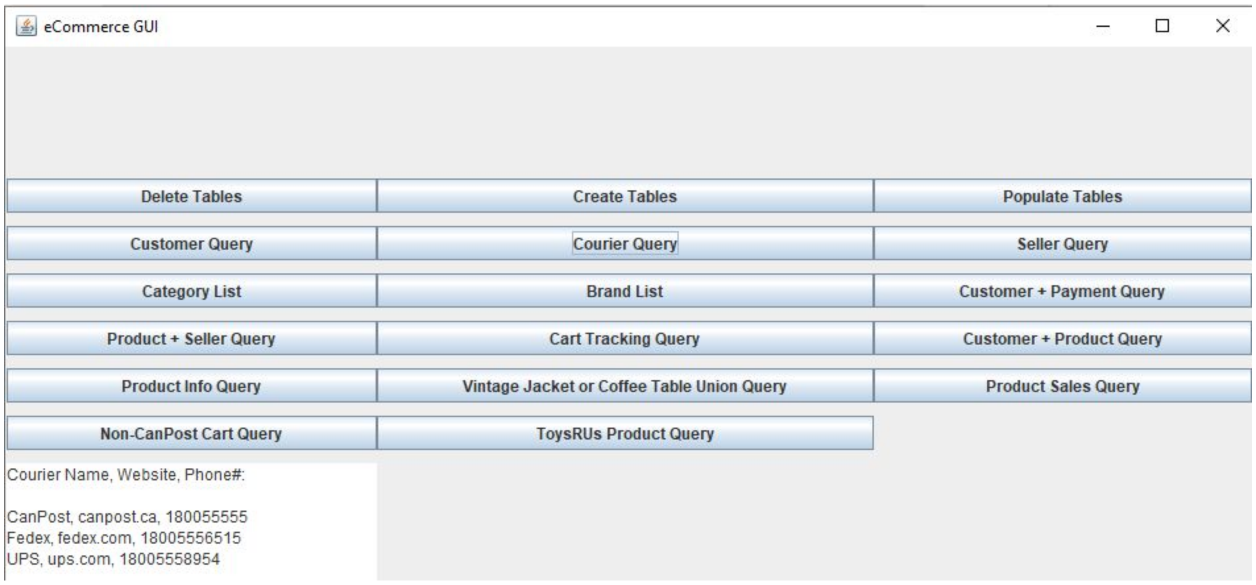


Tables populated.

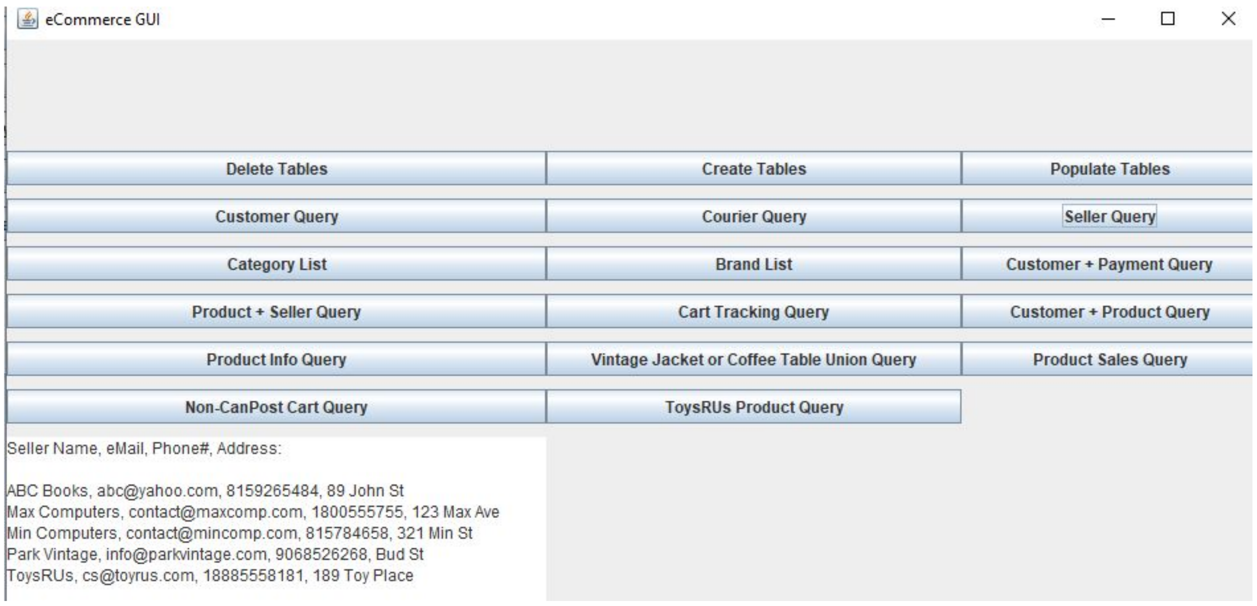
Queries:



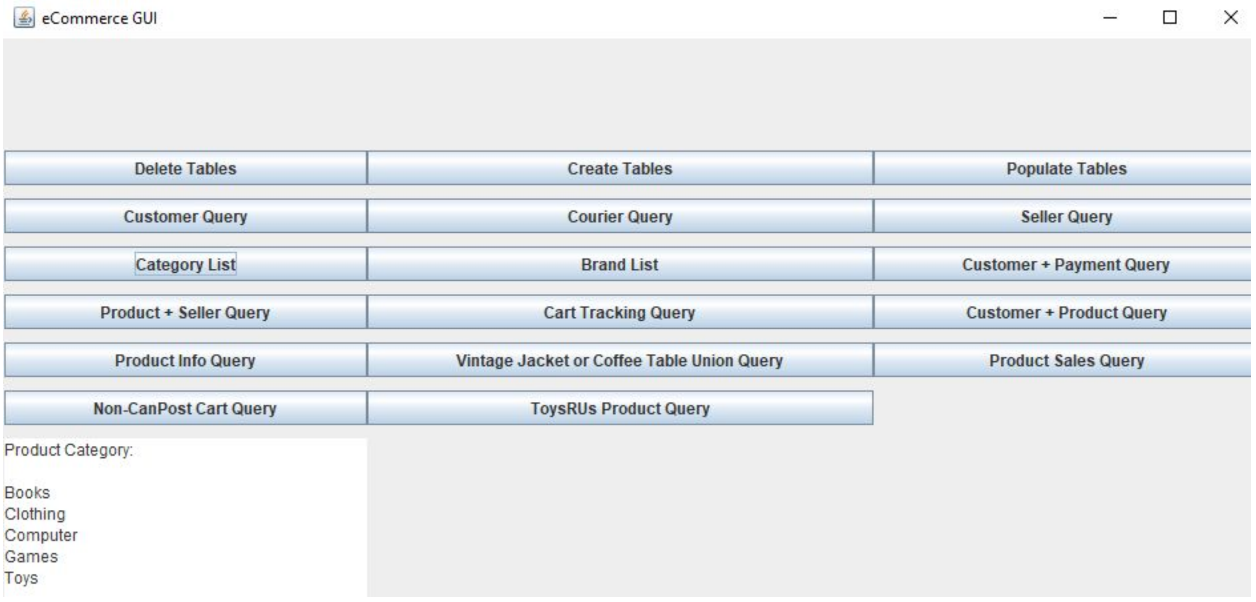
```
Customer Name, eMail

Betty DeLile, bdlile@gmail.com
Cashwan, cashwan@gmail.com
J Weber, jweber@gmail.com
Mr Saad, saad1@gmail.com
Ms Feeld, feeld@gmail.com
```



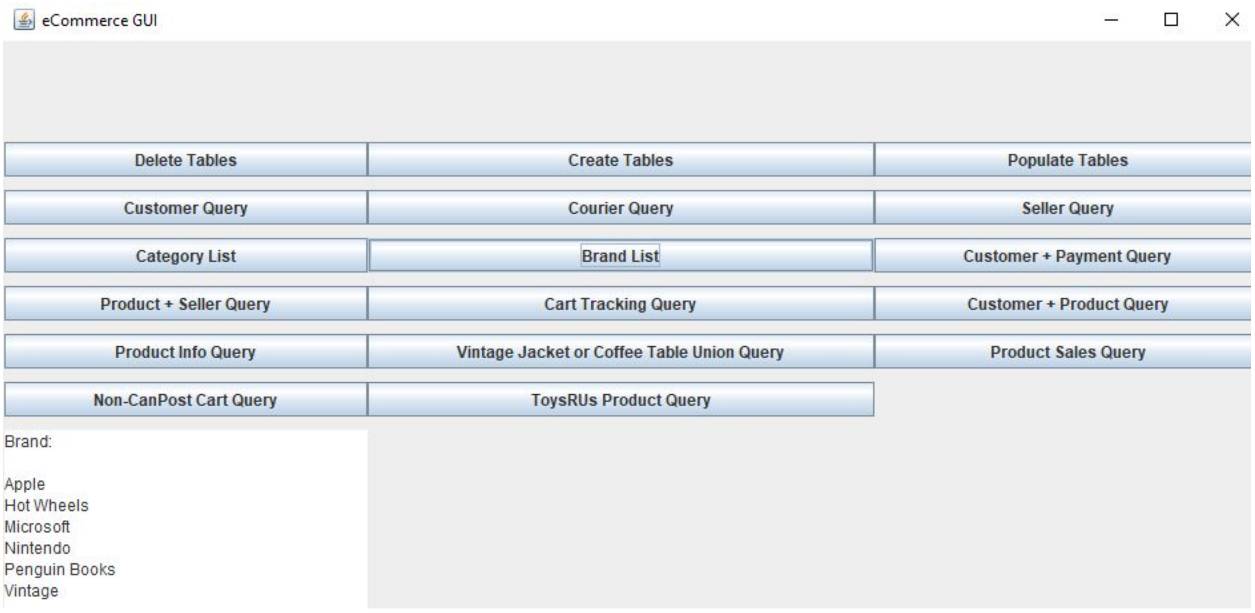
```
Courier Name, Website, Phone#:

CanPost, canpost.ca, 180055555
Fedex, fedex.com, 18005556515
UPS, ups.com, 18005558954
```



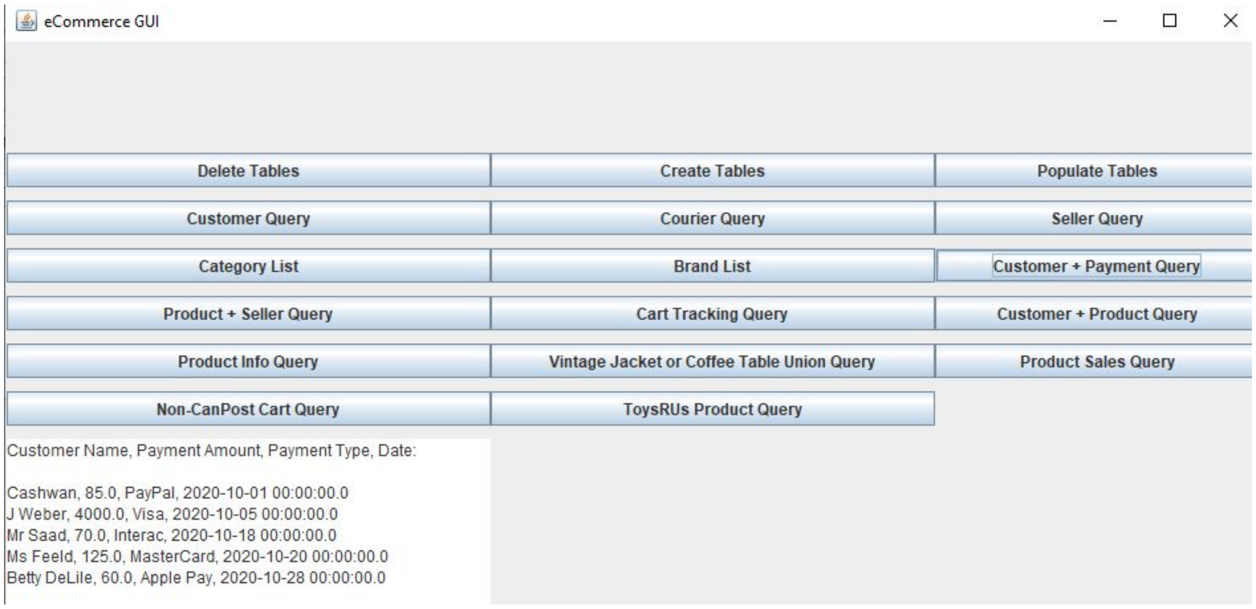
```
Seller Name, eMail, Phone#, Address:

ABC Books, abc@yahoo.com, 8159265484, 89 John St
Max Computers, contact@maxcomp.com, 1800555755, 123 Max Ave
Min Computers, contact@mincomp.com, 815784658, 321 Min St
Park Vintage, info@parkvintage.com, 9068526268, Bud St
ToysRUs, cs@toyrus.com, 18885558181, 189 Toy Place
```



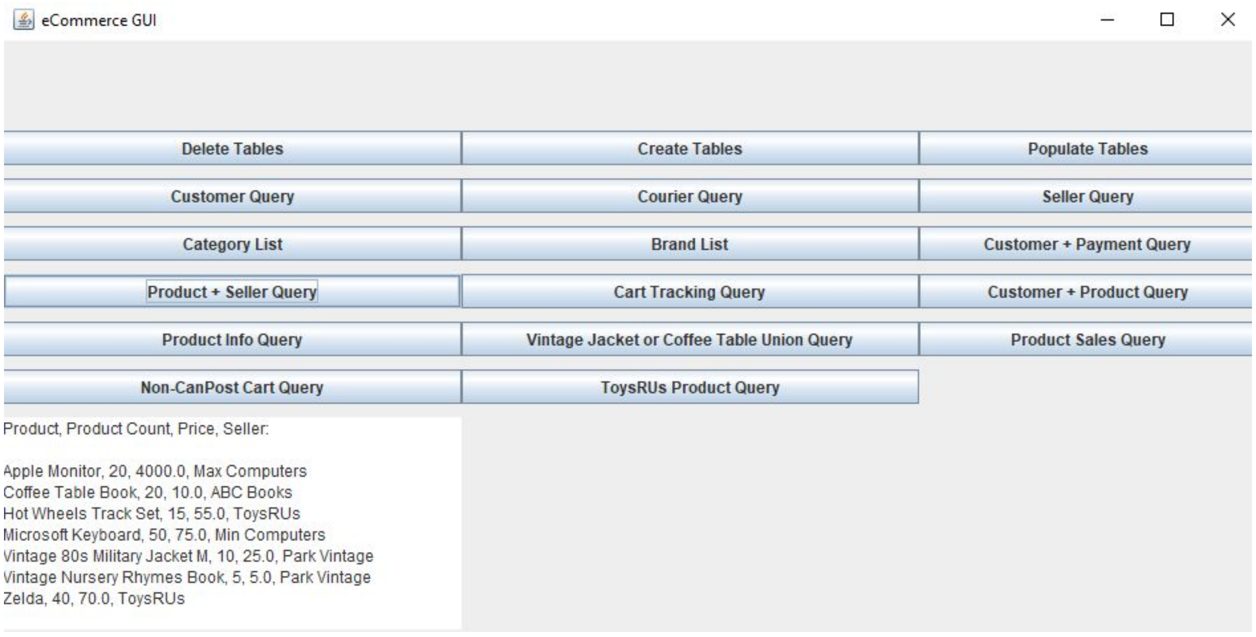
```
Product Category:

Books
Clothing
Computer
Games
Toys
```



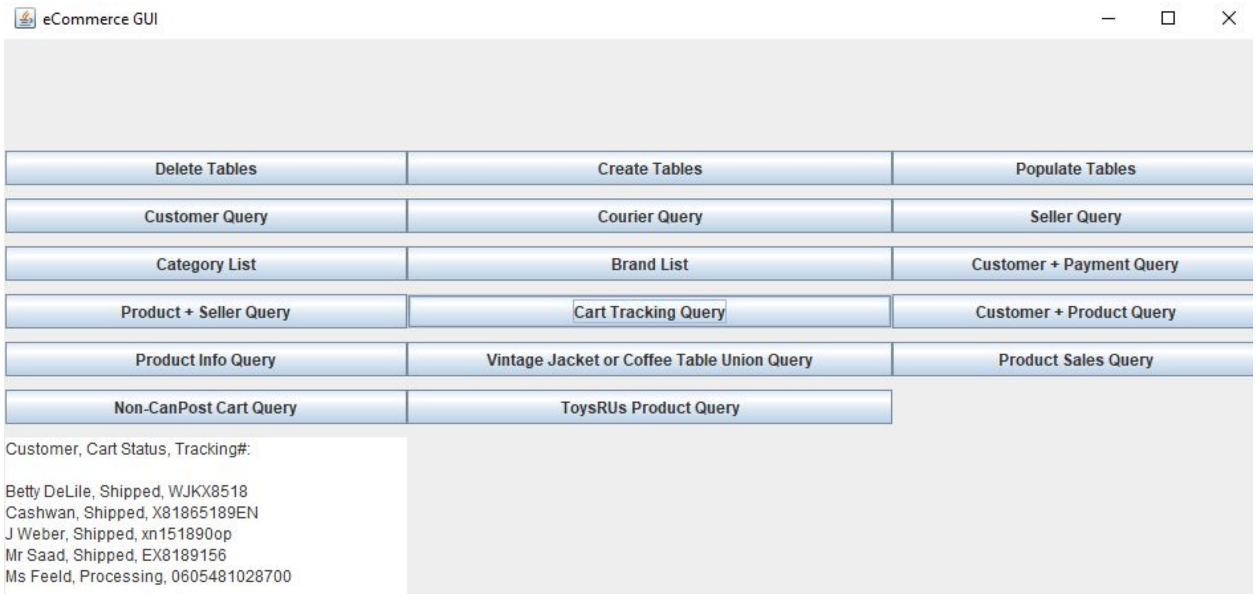
```
Brand:

Apple
Hot Wheels
Microsoft
Nintendo
Penguin Books
Vintage
```



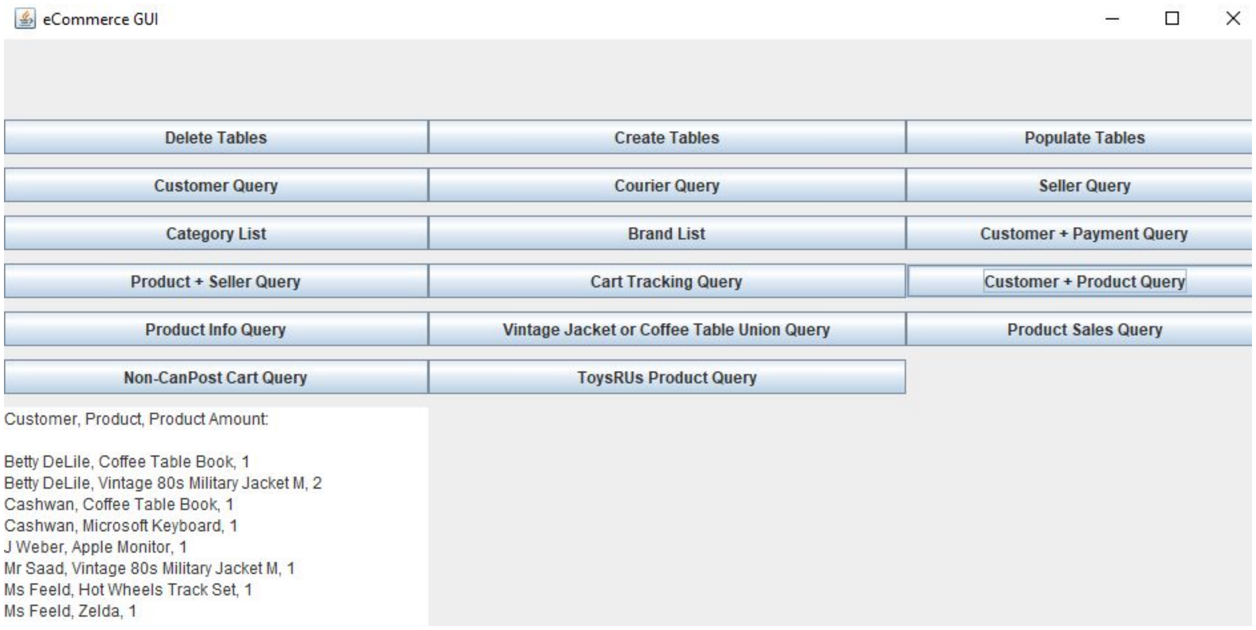
```
Customer Name, Payment Amount, Payment Type, Date:

Cashwan, 85.0, PayPal, 2020-10-01 00:00:00.0
J Weber, 4000.0, Visa, 2020-10-05 00:00:00.0
Mr Saad, 70.0, Interac, 2020-10-18 00:00:00.0
Ms Feeld, 125.0, MasterCard, 2020-10-20 00:00:00.0
Betty DeLile, 60.0, Apple Pay, 2020-10-28 00:00:00.0
```



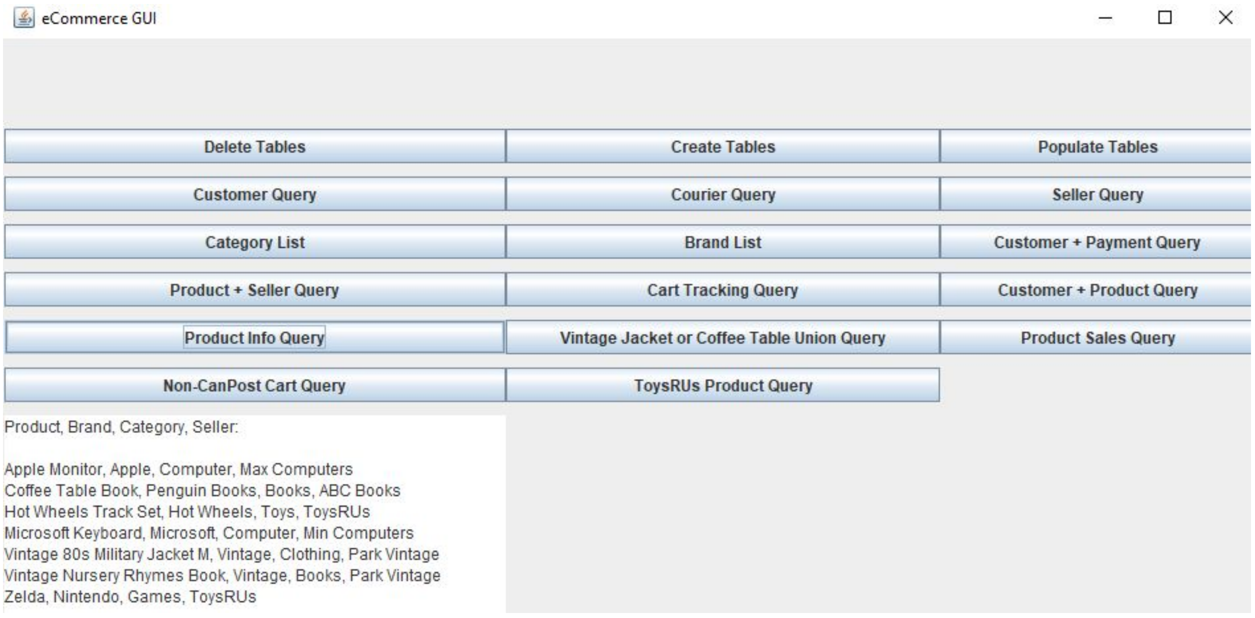
```
Product, Product Count, Price, Seller:

Apple Monitor, 20, 4000.0, Max Computers
Coffee Table Book, 20, 10.0, ABC Books
Hot Wheels Track Set, 15, 55.0, ToysRUs
Microsoft Keyboard, 50, 75.0, Min Computers
Vintage 80s Military Jacket M, 10, 25.0, Park Vintage
Vintage Nursery Rhymes Book, 5, 5.0, Park Vintage
Zelda, 40, 70.0, ToysRUs
```



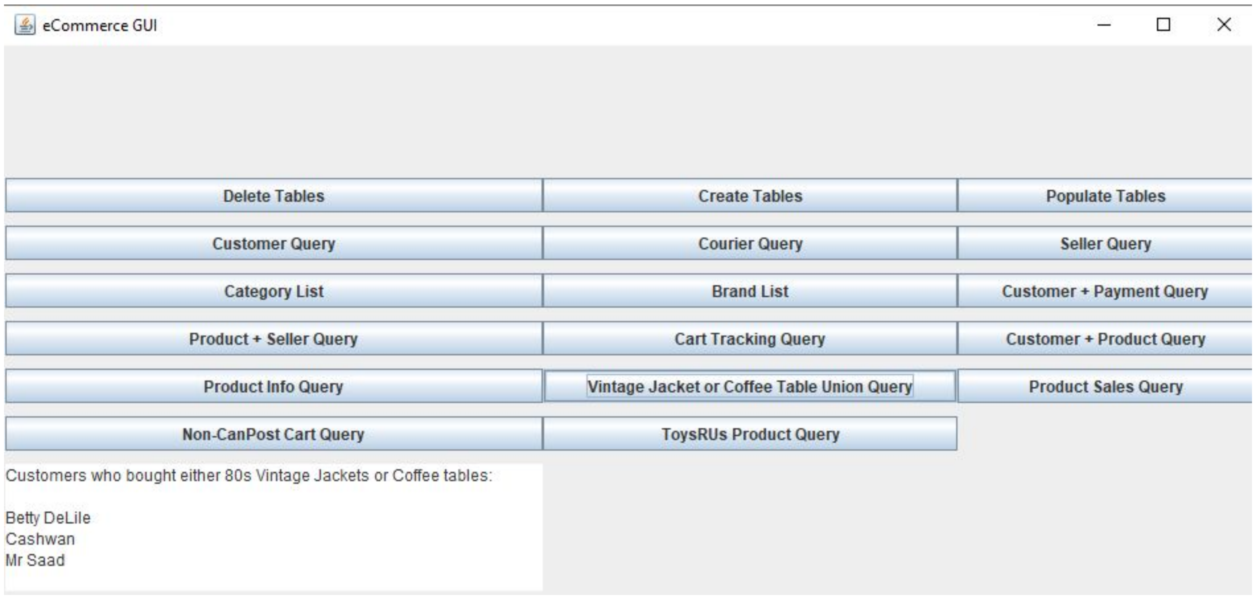
```
Customer, Cart Status, Tracking#:

Betty DeLile, Shipped, WJKX8518
Cashwan, Shipped, X81865189EN
J Weber, Shipped, xn151890op
Mr Saad, Shipped, EX8189156
Ms Feeld, Processing, 0605481028700
```



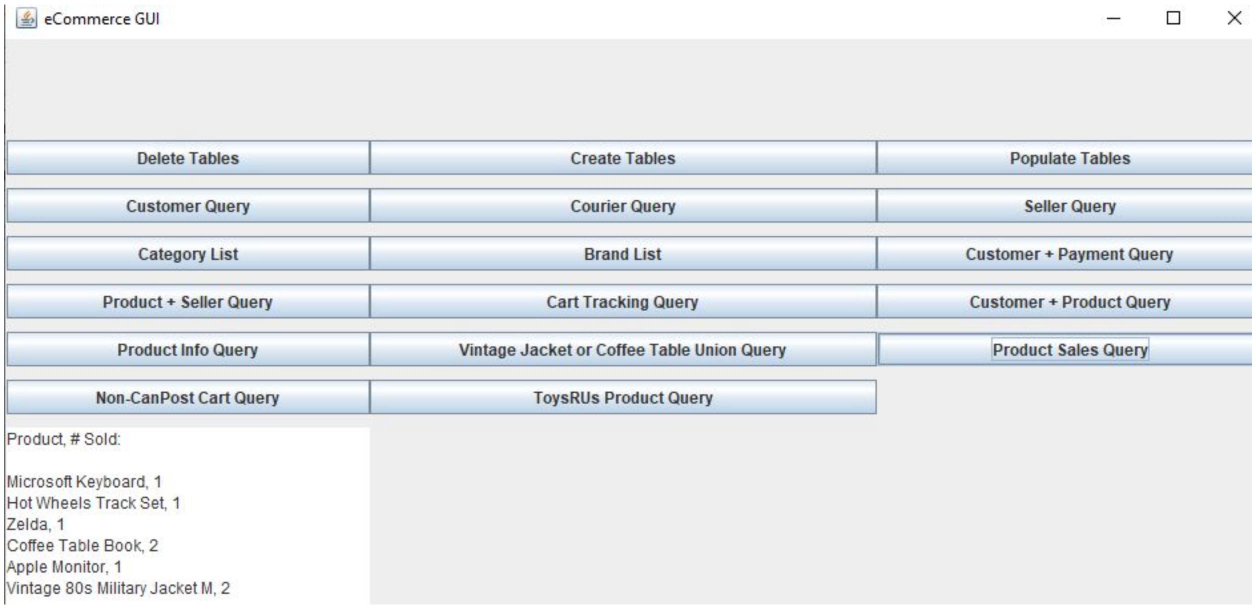
```
Customer, Product, Product Amount:

Betty DeLile, Coffee Table Book, 1
Betty DeLile, Vintage 80s Military Jacket M, 2
Cashwan, Coffee Table Book, 1
Cashwan, Microsoft Keyboard, 1
J Weber, Apple Monitor, 1
Mr Saad, Vintage 80s Military Jacket M, 1
Ms Feeld, Hot Wheels Track Set, 1
Ms Feeld, Zelda, 1
```



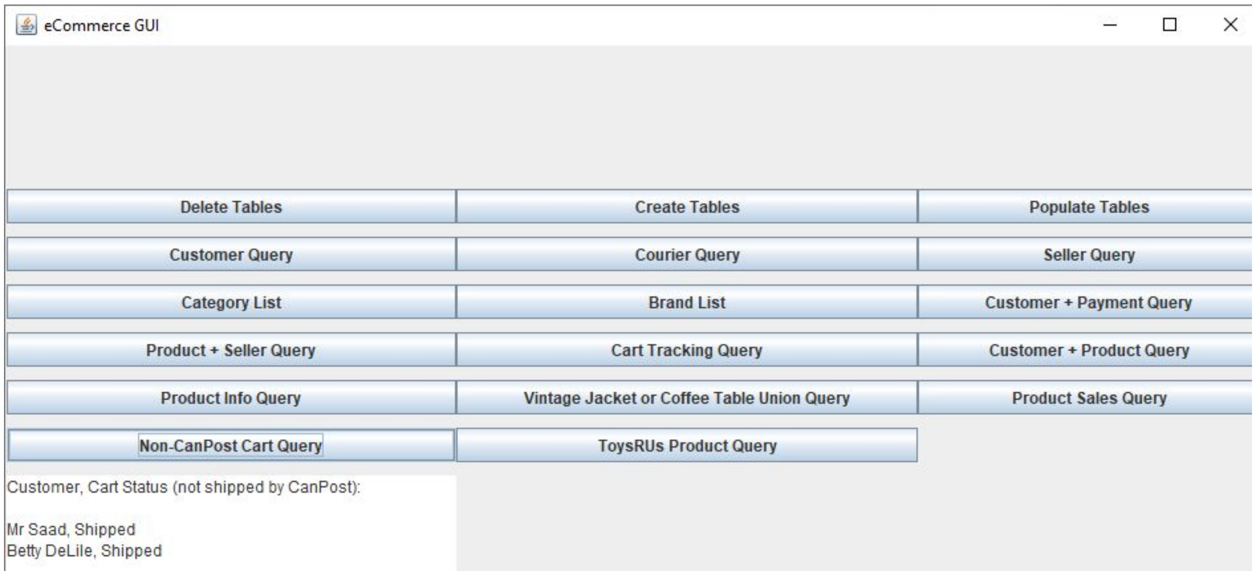
```
Product, Brand, Category, Seller:

Apple Monitor, Apple, Computer, Max Computers
Coffee Table Book, Penguin Books, Books, ABC Books
Hot Wheels Track Set, Hot Wheels, Toys, ToysRUs
Microsoft Keyboard, Microsoft, Computer, Min Computers
Vintage 80s Military Jacket M, Vintage, Clothing, Park Vintage
Vintage Nursery Rhymes Book, Vintage, Books, Park Vintage
Zelda, Nintendo, Games, ToysRUs
```



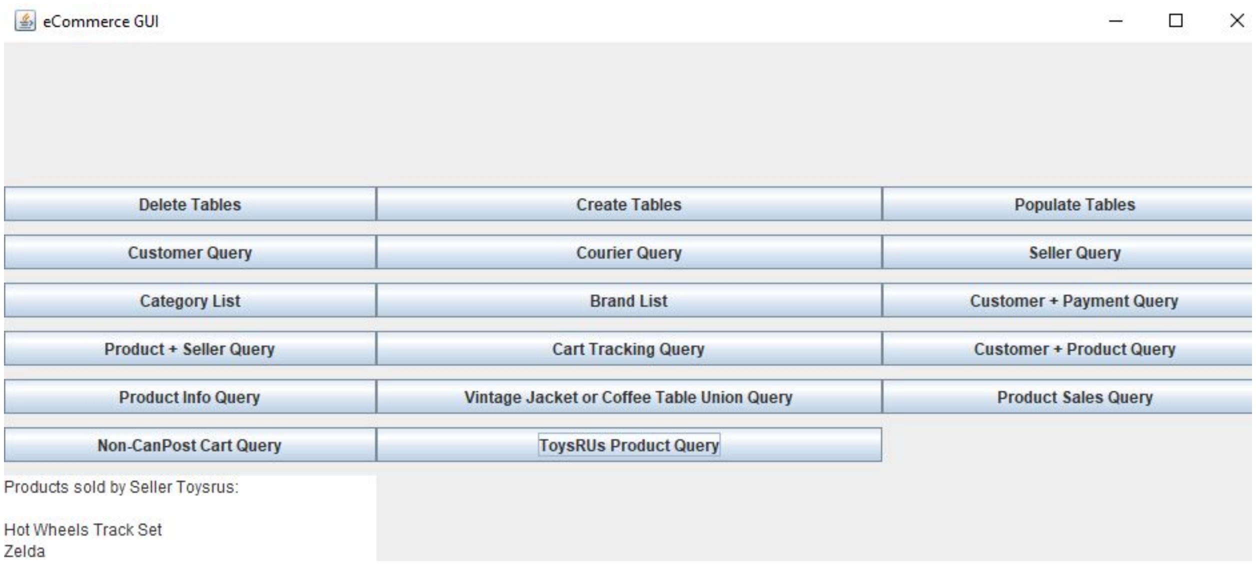
```
Customers who bought either 80s Vintage Jackets or Coffee tables:

Betty DeLile
Cashwan
Mr Saad
```



```
Product, # Sold:

Microsoft Keyboard, 1
Hot Wheels Track Set, 1
Zelda, 1
Coffee Table Book, 2
Apple Monitor, 1
Vintage 80s Military Jacket M, 2
```



```
Customer, Cart Status (not shipped by CanPost):

Mr Saad, Shipped
Betty DeLile, Shipped
```



```
Products sold by Seller Toysrus:

Hot Wheels Track Set
Zelda
```

Source Code (note: real username and password used to access database):

```
import java.awt.*;
import java.awt.event.ActionEvent;
import java.awt.event.ActionListener;
import java.sql.Connection;
import java.sql.DriverManager;
import java.sql.SQLException;
import java.sql.Statement;
import java.sql.ResultSet;
import javax.swing.*;


/**
* This program is a graphical user interface for interacting with an
eCommerce database that is running on Oracle
*
*/
public class GUI {
   //variables to set up the grid layout for the GUI
   final static boolean shouldFill = true;
   final static boolean shouldWeightX = true;
   final static boolean RIGHT_TO_LEFT = false;


   public static void addComponentsToPane(Container pane) {
       if (RIGHT_TO_LEFT) {

pane.setComponentOrientation(ComponentOrientation.RIGHT_TO_LEFT);
       }

       //connecting to the Ryerson Oracle 11g database
       String dbURL1 =
"jdbc:oracle:thin:username/password@oracle.scs.ryerson.ca:1521:orcl";


       //setting up the elements to be used in the frame (button and
textarea)
       JButton button;
       pane.setLayout(new GridBagLayout());
       GridBagConstraints c = new GridBagConstraints();

       //This text area will display the relevant information when
each button is pressed
       JTextArea textArea = new JTextArea("Select a command");
```

```
        c.fill = GridBagConstraints.HORIZONTAL;
        c.anchor = GridBagConstraints.PAGE_END; //bottom of space
        c.insets = new Insets(10,0,0,0);  //top padding
        c.gridy = 7;
        pane.add(textArea, c);

        if (shouldFill) {
            //natural height, maximum width
            c.fill = GridBagConstraints.HORIZONTAL;
        }

        //each button, when pressed, will attempt to connect to the
oracle database and execute the appropriate sql commands or queries
        //button for deleting tables
        button = new JButton("Delete Tables");
        if (shouldWeightX) {
            c.weightx = 0.5;
        }
        c.fill = GridBagConstraints.HORIZONTAL;
        c.gridx = 0;
        c.gridy = 0;
        pane.add(button, c);
        button.addActionListener(new ActionListener() {
            public void actionPerformed(ActionEvent e) {
                try{
                    Connection conn =
DriverManager.getConnection(dbURL1);
                    Statement stmt = conn.createStatement();
                    stmt.executeUpdate("DROP TABLE CART CASCADE
CONSTRAINTS ");
                    stmt.executeUpdate("DROP TABLE SELLER CASCADE
CONSTRAINTS ");
                    stmt.executeUpdate("DROP TABLE COURIER CASCADE
CONSTRAINTS ");
                    stmt.executeUpdate("DROP TABLE PAYMENT CASCADE
CONSTRAINTS ");
                    stmt.executeUpdate("DROP TABLE PRODUCT CASCADE
CONSTRAINTS ");
                    stmt.executeUpdate("DROP TABLE BRAND CASCADE
CONSTRAINTS ");
                    stmt.executeUpdate("DROP TABLE PRODUCT_CATEGORY
CASCADE CONSTRAINTS ");
                    stmt.executeUpdate("DROP TABLE CONTAINS_PRODUCT
CASCADE CONSTRAINTS ");
```

```
                    stmt.executeUpdate("DROP TABLE CUSTOMER CASCADE
CONSTRAINTS ");
                    textArea.setText("Tables dropped.");
                } catch (SQLException err) {
                    textArea.setText("Error dropping tables.");
                    err.printStackTrace();
                }
            }
        });

        //button for creating tables
        button = new JButton("Create Tables");
        c.fill = GridBagConstraints.HORIZONTAL;
        c.weightx = 0.5;
        c.gridx = 1;
        c.gridy = 0;
        pane.add(button, c);
        button.addActionListener(new ActionListener() {
            public void actionPerformed(ActionEvent e) {
                try{
                    Connection conn =
DriverManager.getConnection(dbURL1);
                    Statement stmt = conn.createStatement();
                    String sql = "CREATE TABLE customer( " +
                                 "customer_id NUMBER PRIMARY KEY," +
                                 "customer_name VARCHAR2(128) NOT
NULL," +
                                 "customer_email VARCHAR2(255) NOT
NULL ," +
                                 "customer_phone VARCHAR2(12) NOT
NULL," +
                                 "customer_address VARCHAR2(255) NOT
NULL)";
                    stmt.executeUpdate(sql);
                    sql = "CREATE TABLE courier (" +
                            "courier_id NUMBER PRIMARY KEY," +
                            "courier_web VARCHAR2(100) NOT NULL," +
                            "courier_name VARCHAR2(50) NOT NULL," +
                            "courier_phone VARCHAR2(12) NOT NULL)";
                    stmt.executeUpdate(sql);
                    sql = "CREATE TABLE seller (" +
                            "seller_id NUMBER PRIMARY KEY," +
                            "seller_address VARCHAR2(255) NOT NULL," +
                            "seller_phone VARCHAR2(25) NOT NULL," +
                            "seller_name VARCHAR2(128) NOT NULL," +
                            "seller_email VARCHAR2(254) NOT NULL)";
                    stmt.executeUpdate(sql);
                    sql = "CREATE TABLE product_category (" +
                            "category_id NUMBER PRIMARY KEY," +
                            "category_name VARCHAR2(100) NOT NULL)";
                    stmt.executeUpdate(sql);
                    sql = "CREATE TABLE brand (" +
                            "brand_id NUMBER PRIMARY KEY," +
                            "brand_name VARCHAR2(100) NOT NULL)";
                    stmt.executeUpdate(sql);
                    sql = "CREATE TABLE payment (" +
                            "payment_id NUMBER PRIMARY KEY," +
                            "payment_date DATE DEFAULT SYSDATE NOT
 NULL," +
                            "payment_type VARCHAR2(10) NOT NULL," +
                            "amount DECIMAL(10,2) NOT NULL," +
                            "customer_id NUMBER REFERENCES
 customer(customer_id) ON DELETE CASCADE )";
                    stmt.executeUpdate(sql);
                    sql = "CREATE TABLE cart (" +
                            "cart_id NUMBER PRIMARY KEY," +
                            "cart_status VARCHAR2(15) DEFAULT
 'Processing'," +
                            "total_price DECIMAL(10,2) NOT NULL," +
                            "courier_id NUMBER REFERENCES
 courier(courier_id) ON DELETE CASCADE," +
                            "payment_id NUMBER REFERENCES
 payment(payment_id) ON DELETE CASCADE," +
                            "service_level VARCHAR2(25) NOT NULL," +
                            "tracking_number VARCHAR2(40))";
                    stmt.executeUpdate(sql);
                    sql = "CREATE TABLE product (" +
                            "product_id NUMBER PRIMARY KEY," +
                            "product_name VARCHAR2(100) NOT NULL," +
                            "product_count NUMBER NOT NULL CHECK
 (product_count >= 0)," +
                            "price DECIMAL(10,2) NOT NULL CHECK (price
 > 0)," +
                            "seller_id NUMBER REFERENCES
 seller(seller_id) ON DELETE CASCADE," +
                            "brand_id NUMBER REFERENCES
 brand(brand_id) ON DELETE CASCADE," +
                            "category_id NUMBER REFERENCES
 product_category(category_id) ON DELETE CASCADE)";
                    stmt.executeUpdate(sql);
```

```
                    sql = "CREATE TABLE contains_product(" +
                            "product_id NUMBER REFERENCES
product(product_id) ON DELETE CASCADE," +
                            "cart_id NUMBER REFERENCES cart(cart_id)
ON DELETE CASCADE," +
                            "product_amount NUMBER CHECK
(product_amount > 0)," +
                            "PRIMARY KEY (product_id, cart_id))";
                    stmt.executeUpdate(sql);
                    textArea.setText("Tables created.");
                } catch (SQLException err) {
                    textArea.setText("Error creating tables.");
                    err.printStackTrace();
                }
            }
        });

        //button for populating tables
        button = new JButton("Populate Tables");
        c.fill = GridBagConstraints.HORIZONTAL;
        c.weightx = 0.5;
        c.gridx = 2;
        c.gridy = 0;
        pane.add(button, c);
        button.addActionListener(new ActionListener() {
            public void actionPerformed(ActionEvent e) {
                try{
                    Connection conn =
DriverManager.getConnection(dbURL1);
                    Statement stmt = conn.createStatement();
                    stmt.executeUpdate("INSERT INTO customer
(customer_id, customer_name, customer_email, customer_phone,
customer_address) VALUES (1, 'J Weber', 'jweber@gmail.com',
'123456789', '123 main street')");
                    stmt.executeUpdate("INSERT INTO customer
(customer_id, customer_name, customer_email, customer_phone,
customer_address) VALUES (2, 'Ms Feeld', 'feeld@gmail.com',
'555156189', '24 oak street')");
                    stmt.executeUpdate("INSERT INTO customer
(customer_id, customer_name, customer_email, customer_phone,
customer_address) VALUES (3, 'Mr Saad', 'saad1@gmail.com',
'555999848', '33 welesely rd')");
                    stmt.executeUpdate("INSERT INTO customer
(customer_id, customer_name, customer_email, customer_phone,
customer_address) VALUES (4, 'Cashwan', 'cashwan@gmail.com',
'555444888', '18 river rd')");
                    stmt.executeUpdate("INSERT INTO customer
(customer_id, customer_name, customer_email, customer_phone,
customer_address) VALUES (5, 'Betty DeLile', 'bdlile@gmail.com',
'555841599', '906 cam ave')");
                    stmt.executeUpdate("INSERT INTO courier
(courier_id, courier_name, courier_web, courier_phone ) VALUES (1,
'CanPost', 'canpost.ca', '180055555')");
                    stmt.executeUpdate("INSERT INTO courier
(courier_id, courier_name, courier_web, courier_phone ) VALUES (2,
'Fedex', 'fedex.com', '18005556515')");
                    stmt.executeUpdate("INSERT INTO courier
(courier_id, courier_name, courier_web, courier_phone ) VALUES (3,
'UPS', 'ups.com', '18005558954')");
                    stmt.executeUpdate("INSERT INTO seller (seller_id,
seller_name, seller_address, seller_phone, seller_email) VALUES (1,
'Max Computers', '123 Max Ave', '1800555755',
'contact@maxcomp.com')");
                    stmt.executeUpdate("INSERT INTO seller (seller_id,
seller_name, seller_address, seller_phone, seller_email) VALUES (2,
'Park Vintage', 'Bud St', '9068526268', 'info@parkvintage.com')");
                    stmt.executeUpdate("INSERT INTO seller (seller_id,
seller_name, seller_address, seller_phone, seller_email) VALUES (3,
'Min Computers', '321 Min St', '815784658', 'contact@mincomp.com')");
                    stmt.executeUpdate("INSERT INTO seller (seller_id,
seller_name, seller_address, seller_phone, seller_email) VALUES (4,
'ToysRUs', '189 Toy Place', '18885558181', 'cs@toyrus.com')");
                    stmt.executeUpdate("INSERT INTO seller (seller_id,
seller_name, seller_address, seller_phone, seller_email) VALUES (5,
'ABC Books', '89 John St', '8159265484', 'abc@yahoo.com')");
                    stmt.executeUpdate("INSERT INTO product_category
(category_id, category_name) VALUES (1, 'Computer')");
                    stmt.executeUpdate("INSERT INTO product_category
(category_id, category_name) VALUES (2, 'Clothing')");
                    stmt.executeUpdate("INSERT INTO product_category
(category_id, category_name) VALUES (3, 'Toys')");
                    stmt.executeUpdate("INSERT INTO product_category
(category_id, category_name) VALUES (4, 'Books')");
                    stmt.executeUpdate("INSERT INTO product_category
(category_id, category_name) VALUES (5, 'Games')");
                    stmt.executeUpdate("INSERT INTO brand (brand_id,
brand_name) VALUES (1, 'Apple')");
                    stmt.executeUpdate("INSERT INTO brand (brand_id,
brand_name) VALUES (2, 'Microsoft')");
```

```
                   stmt.executeUpdate("INSERT INTO brand (brand_id,
brand_name) VALUES (3, 'Hot Wheels')");
                   stmt.executeUpdate("INSERT INTO brand (brand_id,
brand_name) VALUES (4, 'Penguin Books')");
                   stmt.executeUpdate("INSERT INTO brand (brand_id,
brand_name) VALUES (5, 'Nintendo')");
                   stmt.executeUpdate("INSERT INTO brand (brand_id,
brand_name) VALUES (6, 'Vintage')");
                   stmt.executeUpdate("INSERT INTO product
(product_id, product_name, product_count, price, brand_id,
category_id, seller_id) VALUES (1, 'Apple Monitor', 20, 4000.00, 1,
1, 1)");
                   stmt.executeUpdate("INSERT INTO product
(product_id, product_name, product_count, price, brand_id,
category_id, seller_id) VALUES (2, 'Microsoft Keyboard', 50, 75.00,
2, 1, 3)");
                   stmt.executeUpdate("INSERT INTO product
(product_id, product_name, product_count, price, brand_id,
category_id, seller_id) VALUES (3, 'Hot Wheels Track Set', 15, 55.00,
3, 3, 4)");
                   stmt.executeUpdate("INSERT INTO product
(product_id, product_name, product_count, price, brand_id,
category_id, seller_id) VALUES (4, 'Coffee Table Book', 20, 10.00, 4,
4, 5)");
                   stmt.executeUpdate("INSERT INTO product
(product_id, product_name, product_count, price, brand_id,
category_id, seller_id) VALUES (5, 'Zelda', 40, 70.00, 5, 5, 4)");
                   stmt.executeUpdate("INSERT INTO product
(product_id, product_name, product_count, price, brand_id,
category_id, seller_id) VALUES (6, 'Vintage 80s Military Jacket M',
10, 25.00, 6, 2, 2)");
                   stmt.executeUpdate("INSERT INTO payment
(payment_id, payment_date, payment_type, amount, customer_id) VALUES
(1, '2020-10-5', 'Visa', 4000.00, 1)");
                   stmt.executeUpdate("INSERT INTO cart (cart_id,
cart_status, total_price, courier_id, payment_id, service_level,
tracking_number) VALUES (1, 'Shipped', 4000.00, 1, 1, 'Regular',
'xn151890op')");
                   stmt.executeUpdate("INSERT INTO contains_product
(product_id, cart_id, product_amount) VALUES (1, 1, 1)");
                   stmt.executeUpdate("INSERT INTO payment
(payment_id, payment_date, payment_type, amount, customer_id) VALUES
(2, '2020-10-20', 'MasterCard', 125.00, 2)");
                   stmt.executeUpdate("INSERT INTO cart (cart_id,
cart_status, total_price, courier_id, payment_id, service_level,
tracking_number) VALUES (2, 'Processing', 125.00, 2, 2, 'Regular',
'0605481028700')");
                    stmt.executeUpdate("INSERT INTO contains_product
(product_id, cart_id, product_amount) VALUES (3, 2, 1)");
                    stmt.executeUpdate("INSERT INTO contains_product
(product_id, cart_id, product_amount) VALUES (5, 2, 1)");
                    stmt.executeUpdate("INSERT INTO payment
(payment_id, payment_date, payment_type, amount, customer_id) VALUES
(3, '2020-10-18', 'Interac', 70.00, 3)");
                    stmt.executeUpdate("INSERT INTO cart (cart_id,
cart_status, total_price, courier_id, payment_id, service_level,
tracking_number) VALUES (3, 'Shipped', 70.00, 3, 3, 'Express',
'EX8189156')");
                    stmt.executeUpdate("INSERT INTO contains_product
(product_id, cart_id, product_amount) VALUES (6, 3, 1)");
                    stmt.executeUpdate("INSERT INTO payment
(payment_id, payment_date, payment_type, amount, customer_id) VALUES
(4, '2020-10-01', 'PayPal', 85.00, 4)");
                    stmt.executeUpdate("INSERT INTO cart (cart_id,
cart_status, total_price, courier_id, payment_id, service_level,
tracking_number) VALUES (4, 'Shipped', 85.00, 1, 4, 'Xpress',
'X81865189EN')");
                    stmt.executeUpdate("INSERT INTO contains_product
(product_id, cart_id, product_amount) VALUES (2, 4, 1)");
                    stmt.executeUpdate("INSERT INTO contains_product
(product_id, cart_id, product_amount) VALUES (4, 4, 1)");
                    stmt.executeUpdate("INSERT INTO product
(product_id, product_name, product_count, price, brand_id,
category_id, seller_id) VALUES (7, 'Vintage Nursery Rhymes Book', 5,
5.00, 6, 4, 2)");
                    stmt.executeUpdate("INSERT INTO payment
(payment_id, payment_date, payment_type, amount, customer_id) VALUES
(5, '2020-10-28', 'Apple Pay', 60.00, 5)");
                    stmt.executeUpdate("INSERT INTO cart (cart_id,
cart_status, total_price, courier_id, payment_id, service_level,
tracking_number) VALUES (5, 'Shipped', 60.00, 2, 5, 'Xpress',
'WJKX8518')");
                    stmt.executeUpdate("INSERT INTO contains_product
(product_id, cart_id, product_amount) VALUES (6, 5, 2)");
                    stmt.executeUpdate("INSERT INTO contains_product
(product_id, cart_id, product_amount) VALUES (4, 5, 1)");
                    textArea.setText("Tables populated.");
                } catch (SQLException err) {
                    textArea.setText("Error populating tables.");
                    err.printStackTrace();
                }
            }
        });

        //button for query of customer names and emails
        button = new JButton("Customer Query");
        c.fill = GridBagConstraints.HORIZONTAL;
        c.weightx = 0.0;
        c.gridwidth = 1;
        c.gridx = 0;
        c.gridy = 1;
        pane.add(button, c);
        button.addActionListener(new ActionListener() {
            public void actionPerformed(ActionEvent e) {
                try{
                    String query = "SELECT customer_name,
customer_email FROM customer ORDER BY customer_name";
                    Connection conn =
DriverManager.getConnection(dbURL1);
                    Statement stmt = conn.createStatement();
                    ResultSet rs = stmt.executeQuery(query);
                    textArea.setText("Customer Name, eMail\n\n");
                    while (rs.next()) {
                        String name = rs.getString("customer_name");
                        String email = rs.getString("customer_email");
                        textArea.setText(textArea.getText() + name +
", " + email + "\n");
                    }

                } catch (SQLException err) {
                    textArea.setText("Error with query.");
                    err.printStackTrace();
                }
            }
        });

        //button for query of courier information
        button = new JButton("Courier Query");
        c.fill = GridBagConstraints.HORIZONTAL;
        c.weightx = 0.0;
        c.gridwidth = 1;
        c.gridx = 1;
        c.gridy = 1;
        pane.add(button, c);
        button.addActionListener(new ActionListener() {
            public void actionPerformed(ActionEvent e) {
                try{
                    String query = "SELECT courier_name, courier_web,
courier_phone FROM courier ORDER BY courier_name";
                    Connection conn =
DriverManager.getConnection(dbURL1);
                    Statement stmt = conn.createStatement();
                    ResultSet rs = stmt.executeQuery(query);
                    textArea.setText("Courier Name, Website,
Phone#:\n\n");
                    while (rs.next()) {
                        String name = rs.getString("courier_name");
                        String web = rs.getString("courier_web");
                        String phone = rs.getString("courier_phone");
                        textArea.setText(textArea.getText() + name +
", " + web + ", " + phone + "\n");
                    }

                } catch (SQLException err) {
                    textArea.setText("Error with query.");
                    err.printStackTrace();
                }
            }
        });

        //button for query of seller information
        button = new JButton("Seller Query");
        c.fill = GridBagConstraints.HORIZONTAL;
        c.weightx = 0.0;
        c.gridwidth = 1;
        c.gridx = 2;
        c.gridy = 1;
        pane.add(button, c);
        button.addActionListener(new ActionListener() {
            public void actionPerformed(ActionEvent e) {
                try{
                    String query = "SELECT seller_name, seller_email,
seller_phone, seller_address FROM seller ORDER BY seller_name";
                    Connection conn =
DriverManager.getConnection(dbURL1);
                    Statement stmt = conn.createStatement();
                    ResultSet rs = stmt.executeQuery(query);
                    textArea.setText("Seller Name, eMail, Phone#,
Address:\n\n");
                    while (rs.next()) {
                        String name = rs.getString("seller_name");
                        String email = rs.getString("seller_email");
                        String phone = rs.getString("seller_phone");
                        String address = 
rs.getString("seller_address");
                        textArea.setText(textArea.getText() + name + 
", " + email + ", " + phone + ", " + address + "\n");
                    }

                } catch (SQLException err) {
                    textArea.setText("Error with query.");
                    err.printStackTrace();
                }
            }
        });

        //button for query of categories
        button = new JButton("Category List");
        c.fill = GridBagConstraints.HORIZONTAL;
        c.weightx = 0.0;
        c.gridwidth = 1;
        c.gridx = 0;
        c.gridy = 2;
        pane.add(button, c);
        button.addActionListener(new ActionListener() {
            public void actionPerformed(ActionEvent e) {
                try{
                    String query = "SELECT category_name FROM 
product_category ORDER BY category_name";
                    Connection conn = 
DriverManager.getConnection(dbURL1);
                    Statement stmt = conn.createStatement();
                    ResultSet rs = stmt.executeQuery(query);
                    textArea.setText("Product Category:\n\n");
                    while (rs.next()) {
                        String name = rs.getString("category_name");
                        textArea.setText(textArea.getText() + name + 
"\n");
                    }

                } catch (SQLException err) {
                    textArea.setText("Error with query.");
                    err.printStackTrace();
                }
            }
        });

        //button for query of brands
        button = new JButton("Brand List");
        c.fill = GridBagConstraints.HORIZONTAL;
        c.weightx = 0.0;
        c.gridwidth = 1;
        c.gridx = 1;
        c.gridy = 2;
        pane.add(button, c);
        button.addActionListener(new ActionListener() {
            public void actionPerformed(ActionEvent e) {
                try{
                    String query = "SELECT brand_name FROM brand ORDER
BY brand_name";
                    Connection conn =
DriverManager.getConnection(dbURL1);
                    Statement stmt = conn.createStatement();
                    ResultSet rs = stmt.executeQuery(query);
                    textArea.setText("Brand:\n\n");
                    while (rs.next()) {
                        String name = rs.getString("brand_name");
                        textArea.setText(textArea.getText() + name +
"\n");
                    }

                } catch (SQLException err) {
                    textArea.setText("Error with query.");
                    err.printStackTrace();
                }
            }
        });

        //button for query of customers and their payments made
        button = new JButton("Customer + Payment Query");
        c.fill = GridBagConstraints.HORIZONTAL;
        c.weightx = 0.0;
        c.gridwidth = 1;
        c.gridx = 2;
        c.gridy = 2;
        pane.add(button, c);
        button.addActionListener(new ActionListener() {
            public void actionPerformed(ActionEvent e) {
                try{
                    String query = "SELECT customer_name, amount,
payment_type, payment_date FROM customer, payment WHERE
customer.customer_id = payment.customer_id ORDER BY payment_date";
                    Connection conn =
DriverManager.getConnection(dbURL1);
                    Statement stmt = conn.createStatement();
                    ResultSet rs = stmt.executeQuery(query);
                    textArea.setText("Customer Name, Payment Amount,
Payment Type, Date:\n\n");
                    while (rs.next()) {
                        String name = rs.getString("customer_name");
                        double amount = rs.getDouble("amount");
                        String type = rs.getString("payment_type");
                        String date = rs.getString("payment_date");
                        textArea.setText(textArea.getText() + name +
", " + amount + ", " + type + ", " + date + "\n");
                    }

                } catch (SQLException err) {
                    textArea.setText("Error with query.");
                    err.printStackTrace();
                }
            }
        });

        //button for sellers and their products
        button = new JButton("Product + Seller Query");
        c.fill = GridBagConstraints.HORIZONTAL;
        c.weightx = 0.0;
        c.gridwidth = 1;
        c.gridx = 0;
        c.gridy = 3;
        pane.add(button, c);
        button.addActionListener(new ActionListener() {
            public void actionPerformed(ActionEvent e) {
                try{
                    String query = "SELECT product_name,
product_count, price, seller_name FROM product, seller WHERE
product.seller_id = seller.seller_id ORDER BY product_name";
                    Connection conn =
DriverManager.getConnection(dbURL1);
                    Statement stmt = conn.createStatement();
                    ResultSet rs = stmt.executeQuery(query);
                    textArea.setText("Product, Product Count, Price,
Seller:\n\n");
```

```
                   while (rs.next()) {
                       String name = rs.getString("product_name");
                       int count = rs.getInt("product_count");
                       double price = rs.getDouble("price");
                       String seller = rs.getString("seller_name");
                       textArea.setText(textArea.getText() + name +
", " + count + ", " + price + ", " + seller + "\n");
                   }

               } catch (SQLException err) {
                   textArea.setText("Error with query.");
                   err.printStackTrace();
               }
           }
       });

       //button for query of customer carts and tracking numbers
       button = new JButton("Cart Tracking Query");
       c.fill = GridBagConstraints.HORIZONTAL;
       c.weightx = 0.0;
       c.gridwidth = 1;
       c.gridx = 1;
       c.gridy = 3;
       pane.add(button, c);
       button.addActionListener(new ActionListener() {
           public void actionPerformed(ActionEvent e) {
               try{
                   String query = "SELECT customer_name, cart_status,
tracking_number FROM customer, cart, payment WHERE cart.payment_id =
payment.payment_id AND payment.customer_id = customer.customer_id
ORDER BY customer_name";
                   Connection conn =
DriverManager.getConnection(dbURL1);
                   Statement stmt = conn.createStatement();
                   ResultSet rs = stmt.executeQuery(query);
                   textArea.setText("Customer, Cart Status,
Tracking#:\n\n");
                   while (rs.next()) {
                       String name = rs.getString("customer_name");
                       String status = rs.getString("cart_status");
                       String track =
rs.getString("tracking_number");
                       textArea.setText(textArea.getText() + name +
", " + status + ", " + track + "\n");
                   }
```

```
                } catch (SQLException err) {
                    textArea.setText("Error with query.");
                    err.printStackTrace();
                }
            }
        });

        //button for query of products ordered by customer
        button = new JButton("Customer + Product Query");
        c.fill = GridBagConstraints.HORIZONTAL;
        c.weightx = 0.0;
        c.gridwidth = 1;
        c.gridx = 2;
        c.gridy = 3;
        pane.add(button, c);
        button.addActionListener(new ActionListener() {
            public void actionPerformed(ActionEvent e) {
                try{
                    String query = "SELECT customer_name,
product_name, product_amount FROM customer, product,
contains_product, cart, payment WHERE contains_product.product_id =
product.product_id AND contains_product.cart_id = cart.cart_id AND
cart.payment_id = payment.payment_id AND payment.customer_id =
customer.customer_id ORDER BY customer_name";
                    Connection conn =
DriverManager.getConnection(dbURL1);
                    Statement stmt = conn.createStatement();
                    ResultSet rs = stmt.executeQuery(query);
                    textArea.setText("Customer, Product, Product
Amount: \n\n");
                    while (rs.next()) {
                        String name = rs.getString("customer_name");
                        String product = rs.getString("product_name");
                        int amount = rs.getInt("product_amount");
                        textArea.setText(textArea.getText() + name +
", " + product + ", " + amount + "\n");
                    }

                } catch (SQLException err) {
                    textArea.setText("Error with query.");
                    err.printStackTrace();
                }
            }
        });
```

```
       //button for query of products and related brand, category and
seller
       button = new JButton("Product Info Query");
       c.fill = GridBagConstraints.HORIZONTAL;
       c.weightx = 0.0;
       c.gridwidth = 1;
       c.gridx = 0;
       c.gridy = 4;
       pane.add(button, c);
       button.addActionListener(new ActionListener() {
           public void actionPerformed(ActionEvent e) {
               try{
                   String query = "SELECT DISTINCT product_name,
brand_name, category_name, seller_name FROM product,
product_category, brand, seller WHERE product.brand_id =
brand.brand_id AND product.category_id = product_category.category_id
AND product.seller_id = seller.seller_id ORDER BY product_name";
                   Connection conn =
DriverManager.getConnection(dbURL1);
                   Statement stmt = conn.createStatement();
                   ResultSet rs = stmt.executeQuery(query);
                   textArea.setText("Product, Brand, Category,
Seller: \n\n");
                   while (rs.next()) {
                       String name = rs.getString("product_name");
                       String brand = rs.getString("brand_name");
                       String category =
rs.getString("category_name");
                       String seller = rs.getString("seller_name");
                       textArea.setText(textArea.getText() + name +
", " + brand + ", " + category + ", " + seller + "\n");
                   }

               } catch (SQLException err) {
                   textArea.setText("Error with query.");
                   err.printStackTrace();
               }
           }
       });

       //button for query of customers who ordered 80s vintage jacket
of coffee table
       button = new JButton("Vintage Jacket or Coffee Table Union
Query");
```

```
        c.fill = GridBagConstraints.HORIZONTAL;
        c.weightx = 0.0;
        c.gridwidth = 1;
        c.gridx = 1;
        c.gridy = 4;
        pane.add(button, c);
        button.addActionListener(new ActionListener() {
            public void actionPerformed(ActionEvent e) {
                try{

                    String query = "SELECT customer_name FROM
customer, product, contains_product, cart, payment WHERE
contains_product.product_id = product.product_id AND
contains_product.cart_id = cart.cart_id AND cart.payment_id =
payment.payment_id AND payment.customer_id = customer.customer_id AND
product_name = 'Vintage 80s Military Jacket M' UNION (SELECT
customer_name FROM customer, product, contains_product, cart, payment
WHERE contains_product.product_id = product.product_id AND
contains_product.cart_id = cart.cart_id AND cart.payment_id =
payment.payment_id AND payment.customer_id = customer.customer_id AND
product_name = 'Coffee Table Book')";
                    Connection conn =
DriverManager.getConnection(dbURL1);
                    Statement stmt = conn.createStatement();
                    ResultSet rs = stmt.executeQuery(query);
                    textArea.setText("Customers who bought either 80s
Vintage Jackets or Coffee tables: \n\n");
                    while (rs.next()) {
                        String name = rs.getString("customer_name");
                        textArea.setText(textArea.getText() + name +
"\n");
                    }

                } catch (SQLException err) {
                    textArea.setText("Error with query.");
                    err.printStackTrace();
                }
            }
        });

        //button for query of products and number of sales of product
        button = new JButton("Product Sales Query");
        c.fill = GridBagConstraints.HORIZONTAL;
        c.weightx = 0.0;
        c.gridwidth = 1;
        c.gridx = 2;
        c.gridy = 4;
        pane.add(button, c);
        button.addActionListener(new ActionListener() {
            public void actionPerformed(ActionEvent e) {
                try{

                    String query = "SELECT product_name, COUNT(*) FROM
contains_product, product WHERE contains_product.product_id =
product.product_id GROUP BY product_name";
                    Connection conn =
DriverManager.getConnection(dbURL1);
                    Statement stmt = conn.createStatement();
                    ResultSet rs = stmt.executeQuery(query);
                    textArea.setText("Product, # Sold: \n\n");
                    while (rs.next()) {
                        String name = rs.getString("product_name");
                        int count = rs.getInt("COUNT(*)");
                        textArea.setText(textArea.getText() + name +
", " + count + "\n");
                    }

                } catch (SQLException err) {
                    textArea.setText("Error with query.");
                    err.printStackTrace();
                }
            }
        });

        //button for query of carts ordered by couriers that are not
CanPost
        button = new JButton("Non-CanPost Cart Query");
        c.fill = GridBagConstraints.HORIZONTAL;
        c.weightx = 0.0;
        c.gridwidth = 1;
        c.gridx = 0;
        c.gridy = 5;
        pane.add(button, c);
        button.addActionListener(new ActionListener() {
            public void actionPerformed(ActionEvent e) {
                try{

                    String query = "SELECT customer_name, cart_status
FROM customer, cart, payment WHERE cart.payment_id =
payment.payment_id AND payment.customer_id = customer.customer_id AND
```

```
cart_status = 'Shipped' AND NOT EXISTS (SELECT * FROM courier WHERE
courier_id = 1 AND cart.courier_id = courier.courier_id)";
                   Connection conn =
DriverManager.getConnection(dbURL1);
                   Statement stmt = conn.createStatement();
                   ResultSet rs = stmt.executeQuery(query);
                   textArea.setText("Customer, Cart Status (not
shipped by CanPost): \n\n");
                   while (rs.next()) {
                       String name = rs.getString("customer_name");
                       String status = rs.getString("cart_status");
                       textArea.setText(textArea.getText() + name +
", " + status + "\n");
                   }

               } catch (SQLException err) {
                   textArea.setText("Error with query.");
                   err.printStackTrace();
               }
           }
       });

       //button for query of products sold by ToysRUs
       button = new JButton("ToysRUs Product Query");
       c.fill = GridBagConstraints.HORIZONTAL;
       c.weightx = 0.0;
       c.gridwidth = 1;
       c.gridx = 1;
       c.gridy = 5;
       pane.add(button, c);
       button.addActionListener(new ActionListener() {
           public void actionPerformed(ActionEvent e) {
               try{

                   String query = "SELECT product_name FROM product
WHERE NOT EXISTS (SELECT * FROM seller WHERE seller_id <> 4 AND
product.seller_id = seller.seller_id)";
                   Connection conn =
DriverManager.getConnection(dbURL1);
                   Statement stmt = conn.createStatement();
                   ResultSet rs = stmt.executeQuery(query);
                   textArea.setText("Products sold by Seller Toysrus:
\n\n");
                   while (rs.next()) {
                       String name = rs.getString("product_name");
```

```
                        textArea.setText(textArea.getText() + name +
"\n");
                    }

                } catch (SQLException err) {
                    textArea.setText("Error with query.");
                    err.printStackTrace();
                }
            }
        });



    }

    //setup JFrame
    private static void createAndShowGUI() {
        //Create and set up the window.
        JFrame frame = new JFrame("eCommerce GUI");
        frame.setDefaultCloseOperation(JFrame.EXIT_ON_CLOSE);
        frame.setSize(600, 1000);

        //Set up the content pane.
        addComponentsToPane(frame.getContentPane());

        //Display the window.
        frame.pack();
        frame.setExtendedState(JFrame.MAXIMIZED_BOTH);
        frame.setVisible(true);
    }

    //main method calls
    public static void main(String[] args) {
        javax.swing.SwingUtilities.invokeLater(new Runnable() {
            public void run() {
                createAndShowGUI();
            }
        });
    }
}
```

Relational Algebra:

$\Pi_{\text{CUSTOMER\_NAME, CUSTOMER\_EMAIL}}(\text{CUSTOMER})$

$\Pi_{\text{COURIER\_NAME, COURIER\_WEB, COURIER\_PHONE}}(\text{COURIER})$

$\Pi_{\text{SELLER\_NAME, SELLER\_EMAIL, COURIER\_PHONE, SELLER\_ADDRESS}}(\text{SELLER})$

$\Pi_{\text{CATEGORY\_NAME}}(\text{CATEGORY})$

$\Pi_{\text{BRAND\_NAME}}(\text{BRAND})$

$\Pi_{\text{CUSTOMER\_NAME, AMOUNT, PAYMENT\_TYPE, PAYMENT\_DATE}}(\text{CUSTOMER} \bowtie \text{PAYMENT})$

$\Pi_{\text{PRODUCT\_NAME, PRODUCT\_COUNT, PRICE, SELLER\_NAME}}(\text{PRODUCT} \bowtie \text{SELLER})$

$\Pi_{\text{CUSTOMER\_NAME, CART\_STATUS, TRACKING\_NUMBER}}(\text{CUSTOMER} \bowtie \text{CART} \bowtie \text{PAYMENT})$

$\Pi_{\text{CUSTOMER\_NAME, PRODUCT\_NAME, PRODUCT\_AMOUNT}}$
$(\text{CUSTOMER} \bowtie \text{PRODUCT} \bowtie \text{CONTAINS\_PRODUCT} \bowtie \text{CART} \bowtie \text{PAYMENT})$

$\Pi_{\text{PRODUCT\_NAME, BRAND\_NAME, CATEGORY\_NAME, SELLER\_NAME}}$
$(\text{PRODUCT} \bowtie \text{PRODUCT\_CATEGORY} \bowtie \text{BRAND} \bowtie \text{SELLER})$

$\text{BoughtJacket} \leftarrow \Pi_{\text{CUSTOMER\_NAME, PRODUCT\_NAME, PRODUCT\_AMOUNT}} (\sigma_{(\text{PRODUCT\_NAME = Vintage 80s Military Jacket M})}$
$(\text{CUSTOMER} \bowtie \text{PRODUCT CONTAINS\_PRODUCT} \bowtie \text{CART} \bowtie \text{PAYMENT}))$
$\text{BoughtBook} \leftarrow \Pi_{\text{CUSTOMER\_NAME, PRODUCT\_NAME, PRODUCT\_AMOUNT}} (\sigma_{(\text{PRODUCT\_NAME = Coffee Table Book})}$
$(\text{CUSTOMER} \bowtie \text{PRODUCT} \bowtie \text{CONTAINS\_PRODUCT} \bowtie \text{CART} \bowtie \text{PAYMENT}))$
$\text{JacketOrBook} \leftarrow \text{BoughtJacket U BoughtBook}$

${}_{\text{PRODUCT\_NAME}}\mathcal{F}_{\text{COUNT PRODUCT\_ID}}\ (\text{CONTAINS\_PRODUCT} \bowtie \text{PRODUCT})$

$\Pi_{\text{PRODUCT\_NAME}}(\sigma_{(\text{PRICE} > 20 \text{ AND PRICE} \leq 70)}\ (\text{PRODUCT})$

$\text{Shipped} \leftarrow \sigma_{(\text{CART\_STATUS = Shipped})}\ (\text{CUSTOMER} \bowtie \text{CART} \bowtie \text{PAYMENT})$
$\text{CanPost} \leftarrow \sigma_{(\text{COURIER\_ID} = 1)}\ (\text{COURIER} \bowtie \text{CART})$
$\text{ShippedNotCanPost} \leftarrow \Pi_{\text{CUSTOMER\_NAME, CART\_STATUS}}(\text{Shipped - CanPost})$

$\text{ToysRUs} \leftarrow \sigma_{(\text{SELLER\_ID} = 4)}\ (\text{SELLER} \bowtie \text{PRODUCT})$
$\text{ProdToysR} \leftarrow \Pi_{\text{PRODUCT\_NAME}}(\text{PRODUCT} \cap \text{ToysRUs})$