Question 7

FIRST VIEW

Tables used in the first view:

CUSTOMER

```
CREATE TABLE CUSTOMER
(
    EMAIL VARCHAR(100) PRIMARY KEY NOT NULL,
    LASTNAME VARCHAR(30),
    FIRSTNAME VARCHAR(30) NOT NULL,
    PASSWORD VARCHAR(30) NOT NULL
)
```

SHOPPINGCART

```
CREATE TABLE SHOPPINGCART
(
    CARTID INT PRIMARY KEY NOT NULL,
    CUSTOMEREMAIL VARCHAR(40),
    ORDERID INT,
    DATECREATED DATE NOT NULL,
    CONSTRAINT SHOPPINGCART_CUSTOMER_EMAIL_FK FOREIGN KEY
(CUSTOMEREMAIL) REFERENCES CUSTOMER (EMAIL),
    CONSTRAINT SHOPPINGCART_ORDER_ORDERID_FK FOREIGN KEY (ORDERID)
REFERENCES ORDER (ORDERID)
)
```

ORDER

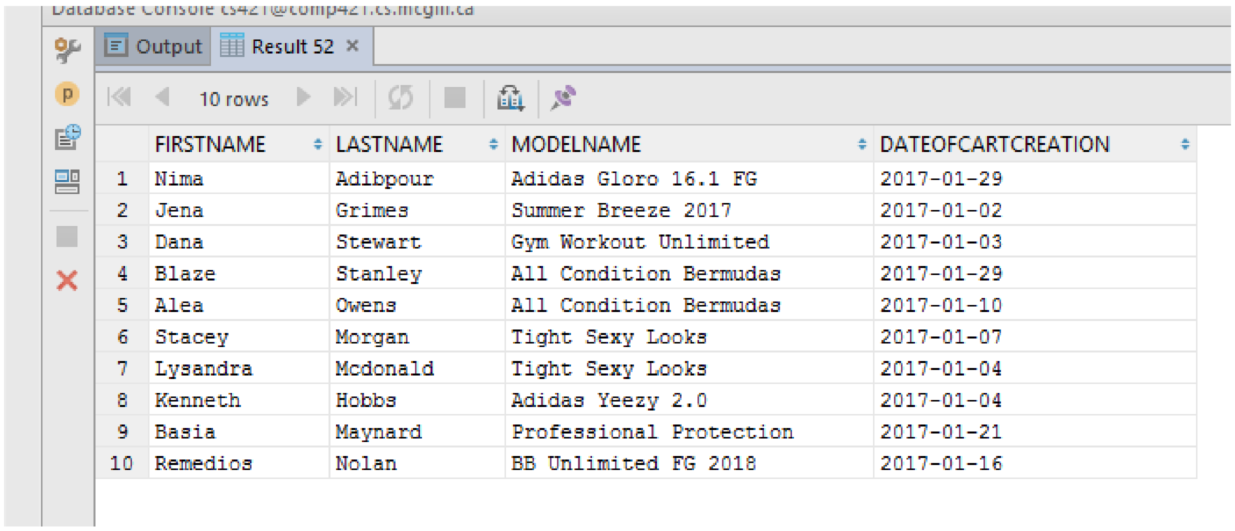
```
CREATE TABLE ORDER
(
    ORDERID INT PRIMARY KEY NOT NULL,
    ORDERTYPE VARCHAR(10) NOT NULL,
    PAYMENTMETHOD VARCHAR(20) NOT NULL,
    ORDERDATE DATE DEFAULT CURRENT DATE NOT NULL,
    FINALAMOUNT FLOAT(53) NOT NULL,
    CUSTOMEREMAIL VARCHAR(100) NOT NULL,
    HANDLEDATE DATE,
    HANDLER INT,
    SHIPPINGID INT,
    BILLINGID INT,
    CARTID INT,
    TRACKINGNUMBER VARCHAR(13),
    CONSTRAINT ORDER_CUSTOMER_EMAIL_FK FOREIGN KEY (CUSTOMEREMAIL)
REFERENCES CUSTOMER (EMAIL),
    CONSTRAINT ORDER_SHIPPINGADDRESS_SA_ID_FK FOREIGN KEY (SHIPPINGID)
REFERENCES SHIPPINGADDRESS (SA_ID),
    CONSTRAINT ORDER_BILLINGADDRESS_BA_ID_FK FOREIGN KEY (BILLINGID)
REFERENCES BILLINGADDRESS (BA_ID),
    CONSTRAINT ORDER_SHOPPINGCART_CARTID_FK FOREIGN KEY (CARTID)
REFERENCES SHOPPINGCART (CARTID)
);
COMMENT ON COLUMN ORDER.ORDERTYPE IS 'PURCHASE or REFUND';
COMMENT ON COLUMN ORDER.PAYMENTMETHOD IS 'VISA/MASTERCARD/AMERICAN
EXPRESS/PAYPAL/INTERAC';
COMMENT ON COLUMN ORDER.SHIPPINGID IS 'Shipping Address ID';
COMMENT ON COLUMN ORDER.BILLINGID IS 'Billing Address ID';
COMMENT ON COLUMN ORDER.CARTID IS 'Shopping Cart ID'
```

CARTDETAILS

```
CREATE TABLE CARTDETAILS
(
    CARTID INT NOT NULL,
    COLOR VARCHAR(15) NOT NULL,
    SIZE VARCHAR(10) NOT NULL,
    MODELNAME VARCHAR(30) NOT NULL,
    QUANTITY INT DEFAULT 1 NOT NULL,
    CONSTRAINT CARTDETAILS_CARTID_COLOR_SIZE_MODELNAME_PK PRIMARY KEY
(CARTID, COLOR, SIZE, MODELNAME),
    CONSTRAINT CARTDETAILS_SHOPPINGCART_CARTID_FK FOREIGN KEY (CARTID)
REFERENCES SHOPPINGCART (CARTID),
    CONSTRAINT CARTDETAILS_CLOTHINGUNIT_COLOR_SIZE_MODELNAME_FK
FOREIGN KEY (SIZE) REFERENCES CLOTHINGUNIT (SIZE)
)
```

This view is named “YellowBuyers” and it represents every customer who purchased one or more yellow clothing units. It shows the customers’ full names, the names of the clothing models bought and the dates each customer created his/her shopping cart.

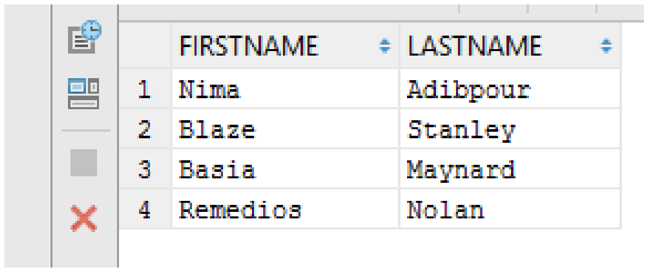
```
CREATE VIEW YellowBuyers (firstname, lastname, modelname, dateofcartcreation)
  AS SELECT C.firstname, C.lastname, D.modelname, S.datecreated
  FROM CUSTOMER C, SHOPPINGCART S, CARTDETAILS D, ORDER O
  WHERE C.email = S.customeremail and S.CARTID = D.CARTID
        and D.CARTID = O.CARTID and O.ordertype = 'PURCHASE'
        and D.color = 'YELLOW';
```



	FIRSTNAME	LASTNAME	MODELNAME	DATEOFCARTCREATION
1	Nima	Adibpour	Adidas Gloro 16.1 FG	2017-01-29
2	Jena	Grimes	Summer Breeze 2017	2017-01-02
3	Dana	Stewart	Gym Workout Unlimited	2017-01-03
4	Blaze	Stanley	All Condition Bermudas	2017-01-29
5	Alea	Owens	All Condition Bermudas	2017-01-10
6	Stacey	Morgan	Tight Sexy Looks	2017-01-07
7	Lysandra	Mcdonald	Tight Sexy Looks	2017-01-04
8	Kenneth	Hobbs	Adidas Yeezy 2.0	2017-01-04
9	Basia	Maynard	Professional Protection	2017-01-21
10	Remedios	Nolan	BB Unlimited FG 2018	2017-01-16

Query: Display the full names of the customers who created their shopping cart after January 10th, 2017

```sql
SELECT FIRSTNAME,LASTNAME
  FROM YellowBuyers
    WHERE dateofcartcreation > '2017-01-10';
```



	FIRSTNAME	LASTNAME
1	Nima	Adibpour
2	Blaze	Stanley
3	Basia	Maynard
4	Remedios	Nolan

SECOND VIEW

Tables used in the second view:

UNITSTOCKING

```
CREATE TABLE UNITSTOCKING
(
    MODELNAME VARCHAR(30) NOT NULL,
    QUANTITYAVAILABLE INT NOT NULL,
    LOCATION VARCHAR(20) NOT NULL,
    COLOR VARCHAR(15) NOT NULL,
    SIZE VARCHAR(10) NOT NULL,
    CONSTRAINT UNITSTOCKING_COLOR_SIZE_PK PRIMARY KEY (COLOR,
MODELNAME, SIZE, LOCATION),
    CONSTRAINT UNITSTOCKING_WAREHOUSE_LOCATION_FK FOREIGN KEY
(LOCATION) REFERENCES WAREHOUSE (LOCATION)
);
COMMENT ON COLUMN UNITSTOCKING.QUANTITYAVAILABLE IS 'Current Stock'
```

CLOTHINGCATEGORY

```
CREATE TABLE CLOTHINGCATEGORY
(
    CATID INT PRIMARY KEY NOT NULL,
    CATNAME VARCHAR(50) NOT NULL,
    CATTYPE VARCHAR(20) NOT NULL,
    CATEGORYDISCOUNT INT DEFAULT 0 NOT NULL
);
COMMENT ON COLUMN CLOTHINGCATEGORY.CATEGORYDISCOUNT IS 'Displayed as
percentage'
```

CLOTHINGMODEL

```
CREATE TABLE CLOTHINGMODEL
(
    MODELNAME VARCHAR(30) PRIMARY KEY NOT NULL,
    SEX VARCHAR(1) NOT NULL,
    AGERANGE VARCHAR(10),
    MODELDISCOUNT INT DEFAULT 0,
    PRICE FLOAT(53) NOT NULL,
    CATID INT NOT NULL,
    CONSTRAINT CLOTHINGMODEL_CLOTHINGCATEGORY_CATID_FK FOREIGN KEY
(CATID) REFERENCES CLOTHINGCATEGORY (CATID)
);
COMMENT ON COLUMN CLOTHINGMODEL.MODELDISCOUNT IS 'Percentage Values'
```

BILLINGADDRESS

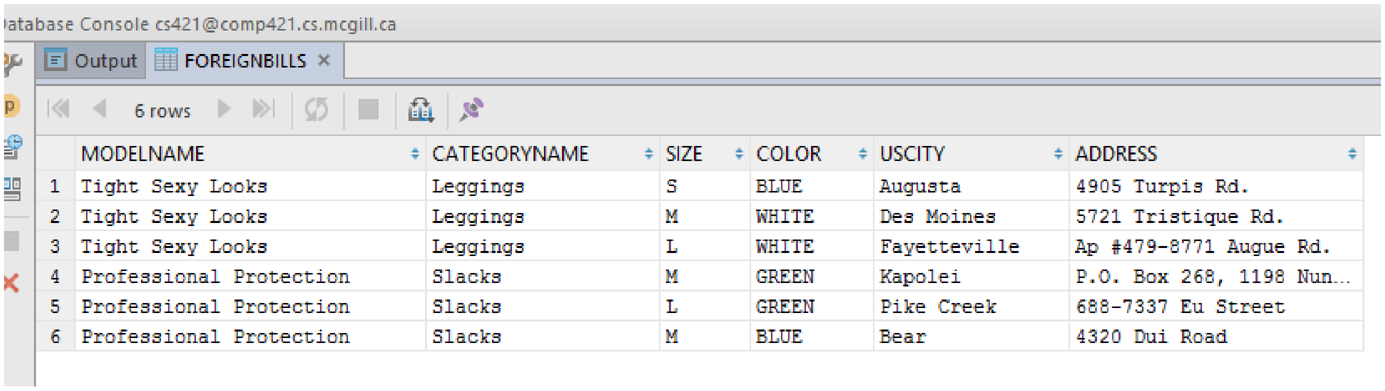
```
CREATE TABLE BILLINGADDRESS
(
    STREETADDRESS VARCHAR(50) NOT NULL,
    CITY VARCHAR(20) NOT NULL,
    "Postal/Zip" VARCHAR(15) NOT NULL,
    COUNTRY VARCHAR(15) NOT NULL,
    BA_ID INT PRIMARY KEY NOT NULL
)
```

ORDER
Refer to First View

CARTDETAILS
Refer to First View

The view is named “ForeignBills” and it represents all the legwear clothes that were produced in the Paris warehouse but billed at an address in the United States. The view shows the name of the clothing model, its legwear category, the size and color of the model the customer ordered and the address of the customer.

```
CREATE VIEW ForeignBills (modelname, categoryname, size, color, USCity,
address)
  AS SELECT U.modelname, C.catname, D.size, D.color, B.city, B.streetaddress
  FROM UNITSTOCKING U, CLOTHINGCATEGORY C, CLOTHINGMODEL M, BILLINGADDRESS B,
ORDER O, CARTDETAILS D
  WHERE C.cattype = 'LEGWEAR' and C.catid = M.catid and M.modelname =
D.modelname and D.modelname = U.modelname
    and D.size = U.size and D.color = U.color
        and D.cartid = O.cartid and O.billingid = B.ba_id and B.country =
'United States' and U.location = 'PARIS';
```


atabase Console cs421@comp421.cs.mcgill.ca

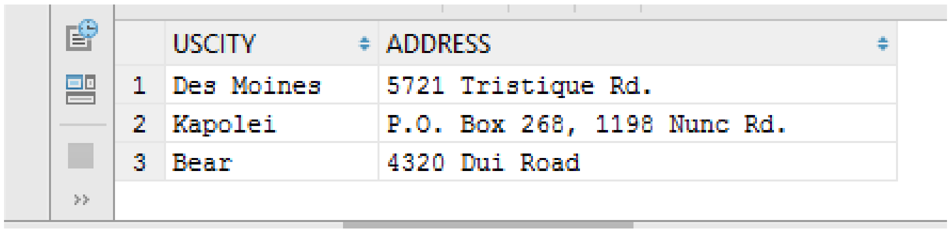
Output | FOREIGNBILLS

6 rows

	MODELNAME	CATEGORYNAME	SIZE	COLOR	USCITY	ADDRESS
1	Tight Sexy Looks	Leggings	S	BLUE	Augusta	4905 Turpis Rd.
2	Tight Sexy Looks	Leggings	M	WHITE	Des Moines	5721 Tristique Rd.
3	Tight Sexy Looks	Leggings	L	WHITE	Fayetteville	Ap #479-8771 Augue Rd.
4	Professional Protection	Slacks	M	GREEN	Kapolei	P.O. Box 268, 1198 Nun…
5	Professional Protection	Slacks	L	GREEN	Pike Creek	688-7337 Eu Street
6	Professional Protection	Slacks	M	BLUE	Bear	4320 Dui Road

Query: Display the full address of the customers with medium sized legwear clothes.

```sql
SELECT USCITY,ADDRESS
  FROM ForeignBills
    WHERE size = 'M';
```



	USCITY	ADDRESS
1	Des Moines	5721 Tristique Rd.
2	Kapolei	P.O. Box 268, 1198 Nunc Rd.
3	Bear	4320 Dui Road

UPDATE STATEMENTS

Let's try running the following SQL UPDATE statement on the view "YellowBuyers":

```sql
UPDATE YellowBuyers
SET MODELNAME='Summer Breeze 2017'
WHERE FIRSTNAME='Blaze' OR LASTNAME='Morgan';
```

We expect this to replace the actual clothing models purchased by Blaze Stanley and Stacey Morgan by the model "Summer Breeze 2017". However, the following message gets displayed once the statement is ran:

"[42807][-150] The target fullselect, view, typed table, materialized query table, range-clustered table, or staging table in the INSERT, DELETE, UPDATE, MERGE, or TRUNCATE statement is a target for which the requested operation is not permitted.. SQLCODE=-150, SQLSTATE=42807, DRIVER=4.7.85"

Now, let’s try running the following SQL UPDATE statement on the view “ForeignBills”:

```
UPDATE ForeignBills
SET USCITY='Montreal'
WHERE MODELNAME='Tight Sexy Looks';
```

We expect this to replace all the actual US city entries where the name of the clothing model is ‘Tight Sexy Looks’ with ‘Montreal’. However, the exact same message as the previous one gets displayed.

None of the two views are updatable!
This makes sense because in order for a view to be updated, the base tables used to create the view also need to be updated. Let’s take for instance the view “YellowBuyers”. The names of the clothing models associated with the names of the customers were obtained by getting the e-mail address of each customer from table “CUSTOMER”, which is also a foreign key in table “SHOPPING CART”. Thus, the id of the shopping cart, which is also a foreign key in table “CARTDETAILS”, can be associated with the name of the clothing model from “CARTDETAILS”. In brief, the information stored in “YellowBuyers” comes from the retrieval of different foreign keys of different tables. Updating the view in this instance would mean updating every entry inside the involved foreign key columns of every base tables. This is a very similar situation with “ForeignBills” since it was generated based on multiple tables.
For DB2 or PostgreSQL to allow updating a view, the view must be defined based on one and only one table. Furthermore, the view must include the primary key of the table based upon which the view was created. Indeed, the view must be able to perform a UPDATE command on itself without affecting any other table.