Stock Broker DBMS Project

Deliverable 4

Thomas Ciha Jacob Burke Ethan Murphy

Part 1: Queries

Query 1:(Thomas)

// Display the Name and City of all users who have an active account with over 100,000

```
SELECT FName, LName, City, USERID
FROM UserTable JOIN (
SELECT AccountNO
FROM Account
WHERE Balance > 100000 AND ActivityStatus = 'Active'
) USING (AccountNO)
ORDER BY LName;
```

OUTPUT:

```
SQL> SELECT FName, LName, City
  2  FROM UserTable JOIN (
  3  SELECT AccountNO
  4  FROM Account
  5  WHERE Balance > 100000 AND ActivityStatus = 'Active'
  6  ) USING (AccountNO)
  7  ORDER BY LName;

FNAME           LNAME           CITY
--------------- --------------- --------------------
0NNJ94PWg9      03UdJ1pQGv      EtuvRS5IO7xo2Oky4DTT
gHN6oURwDt      068wCZlg6p      YEj5yMIehWhGAnqEk529
I4GrBIMtEZ      06djtnfSMX      v97tfckqWciI61WZlBCr
aUteRkF1zN      06psDbz31W      B4I31UclJljn6N27WB1W
VdtO8V9g4w      086eMccub8      6q8n7AE0Mix45wrET7fz
9j59c5UnqT      09OixiJaw4      8kVm5IGudeFQgc7XqaJC
akPTWF1Yxw      0A4hBXiI8r      CMevzVLuIPa7hRUwGKx2
I8UDnVGJDv      0DVT2lKEp3      R4ku4bRfJ09B7HW4C01t
Ztb4WgVja2      0EIf9Oe85J      mmKUaa5FkLWUrq7CAeXd
wUmxTF0bnj      0EREdTx5P6      vjbxsYoyP9ljW4nKEQQ0
FqpcPthPWY      0EVzAkaj2A      ULqSVBI4KqwJTnmjJfAr

FNAME           LNAME           CITY
--------------- --------------- --------------------
WrGTOE8TTq      0FSVX7HaVn      fgd65dyguvONAVBMIQW7
enVUfo2F5r      0J4Bu348zq      INZ08YVx77MIfxruaz79
kBRFvyAbLs      0JW4kEq5Wb      e3TuE3ngxFofOOMS7TwY
vhvFdc41HZ      0KXeFf8Zbb      Awo0pwUHmkSAHunWQfUj
0QAwokQ7sq      0NIaVIi1QJ      qMIIqdhJ72pv8dyMni5q
If59r0WHfl      0VE0rgHkg0      Y8rVUxe7JoJ4a9LBbe72
lizIs7RZZ4      0VH8gt1sde      cgPfY2QirWjDgEHbKTts
V28onzFFuq      0YX5uCNjDg      MYIt8SS3uajvJnC7ZZA6
Xk5pxGHxOX      0Zzbcu0ueI      wKEbjurbZEF3t9MUX2fD
yaBGlZo4T6      0aQDvGxM6h      87P3htkg7iqE5lVm5guo
VsEHXPzCPn      0aidYia87D      Q7vUZqOPMHIYdeILgBze
```

Query 2: (Thomas)

// Display total value of stocks that user FName = 'BjrrbBLw77' AND Lname = 'QITPh7iEX2' OWNS in their account
// NOTE: This excludes any stocks they have shorted as they obliged to pay X dollars back to the firm
//   where X = Quantity * purchase price they shorted it at.

//step1: get the account number of user:

```
SELECT AccountNo
FROM UserTable
WHERE FName = 'BjrrbBLw77' AND Lname = 'QITPh7iEX2'
```

//step2: use that account no to determine the Ticker and Quantity of stocks they OWN (meaning Quantity > 0)

```
SELECT Ticker, Quantity
FROM ContainsStock JOIN (
SELECT AccountNo
FROM UserTable
WHERE FName = 'BjrrbBLw77' AND Lname = 'QITPh7iEX2'
) USING (AccountNo)
WHERE Quantity > 0;
```

// Step 3: use the ticker of all stocks that account number owns to determine the market value of those shares

```
SELECT Quantity * Price AS MarketValue
FROM StockTable JOIN (
        SELECT Ticker, Quantity
        FROM ContainsStock JOIN (
                SELECT AccountNo
                FROM UserTable
                WHERE FName = 'BjrrbBLw77' AND Lname = 'QITPh7iEX2'
        ) USING (AccountNo)
        WHERE Quantity > 0
) USING (Ticker);
```

OUTPUT:

```
SQL> SELECT Quantity * Price AS MarketValue
  2  FROM StockTable JOIN (
  3  SELECT Ticker, Quantity
```

```
  4  FROM ContainsStock JOIN (
  5  SELECT AccountNo
  6  FROM UserTable
  7  WHERE FName = 'BjrrbBLw77' AND Lname = 'QITPh7iEX2'
  8  ) USING (AccountNo)
  9  WHERE Quantity > 0
 10  ) USING (Ticker);

MARKETVALUE
-----------
     482600
```

Query 3:

Select all Users with a Filled Optionorder with size over 50 , that have an accountNumber greater than 5000 ordered by Last Name (ETHAN)

```
Select UserId,LNAme,FName,accountNO
From Usertable natural join optionorder
where OOsize > 50 and status = 'Filled'
AND AccountNO > 5000
Order By LName,FNAme
;
```

<u>OUTPUT</u>

```
SQL> Select UserId,LNAme,FName,accountNO
  2  From Usertable natural join optionorder
  3  where OOsize > 50 and status = 'Filled'
  4  AND AccountNO > 5000
  5  Order By LName,FNAme
  6  ;

USERID                   LNAME           FNAME            ACCOUNTNO
------------------------ --------------- --------------- ----------
77794739                 fRn8vcFW57      BKzCqmWbra            9988
36123259                 xG4zn9Ikd1      6sG6B2SvZk            9757
```

Query 4:

Selecting first names, last names, and the addresses of users currently working with a professional account status, but have had their accounts frozen.

Query: SQL> SELECT LNAME, FNAME, ADDRESS FROM USERTABLE JOIN (SELECT Accountno FROM ACCOUNT WHERE ACCOUNTSTATUS = 'Prof' AND ACTIVITYSTATUS = 'Frozen') USING (ACCOUNTNO) ORDER BY FNAME;

Output:

```
SQL>  SELECT LNAME, FNAME, ADDRESS FROM USERTABLE JOIN (SELECT Accountno FROM ACCOUNT WHERE
ACCOUNTSTATUS = 'Prof' AND ACTIVITYSTATUS = 'Frozen' ) USING (ACCOUNTNO) ORDER BY FNAME;

LNAME          FNAME          ADDRESS
-------------- -------------- ------------------------------
5TC8Oi2wmQ     0127FutuNM     Hauqb8nP1ZzuCthZBZSWGNgs2S
XtGoCuqcRX     01x5R9NhwK     wJ8dMWZEO9BATKvXdJaw0zOGgJ
yVywyn1Tqx     03aKTJaQek     jbMbaQHR0vEseOgAnc1WD49jak
aK4hYDJKN0     0AAjVYE42S     yCtyY5IsQDyQggteHdBkoc2jFU
a5NaiRLOdQ     0D4p8wawqN     DIUZ84jc3WLX6u0nfKmK3e61XO
wObV2XFYdq     0Dxi2aVxZD     VhhsGzClFMxffYPkcaXZG7KIBI
GMdx3QS0ID     0H5ZclL7mk     5xgjx4RSY9pfcAMeq8wKp3yv1d
0zf8ILUSrD     0HCDgaCf1N     kV7IoQM16DRIpoEy3q7NpxQVze
p5GVuMEZnp     0IRUpkJfeo     byhQIBxODtOzfaK8DZ9cA6EDZT
dnZ3qO6IG6     0L1phNpi7P     tmQcRuARL1bLEOvVvH1qqbcuJ2
47pt8MVrcO     0OcEsbhgL9     6hSdOFMXYIYcSWBIVF0EUVFFii

LNAME          FNAME          ADDRESS
-------------- -------------- ------------------------------
RFFzpemmM3     0QBFCVvab3     MBkEQRbOBLtdixu8IZMbCydYL6
i6SJxUTYzK     0Qm9JZx8SI     cwa75KVkyw0xA5ur6ZZWX0zgWW
JUDshvN7TI     0SYMrfYyUY     MC7MDiHFKKZH9YvcyMaFcSSdI2
qorIdVFHk8     0VhgG3tIjJ     dyBNgRkVQkm8yEKXZg3oMXx4zr
Z73w5GsdIa     0Y53Iv2jak     I58rtYDP6p9TsnXDMVjf6dZWAZ
FRCq4iaoaB     0ZVaDzGwDq     2p3ddPM73hCZMkT2n4pGf92twG
acNUD263VC     0b0JuNZzSW     I51XXXgDUvuBy6d5AvIbMQVSYB
iF8R6QGSUh     0cTirqUsrf     oVovnDpaSLijrGnknekd4UEfxf
HNPk86Xi9r     0cozXxLaYs     3xsIKIyIGfkvEAi8d63SJif8sW
7wLCQW68tS     0ga8QHcFMI     GqdXTnIIfckvQ6vwaInsHgrb06
hmibdtEjME     0jPduXIAIs     4xGW8Ccw27RsMijC4LkjD9oi3Z
```

Query 5:

DIsplay the Price Of Both optionOrders and stockorders then compute the average total cost of an order for each.

```
Select  OrderNUm , 'OptionOrder' As Type ,OOSize ,Price, (OOsize*Price) as OrderPrice
from optionorder
Group By OrderNUm,OOsize,Price
UNION
```

```
Select orderNUm,'StockOrder' AS Type, SOsize ,Price, (SOSIZE*Price) as Orderprice
From stockorder
Group by orderNum,SOsize,Price
order by type;
Select 'OptionOrder' AS type,(SUM(Price)+SUM(OOsize))/count(Price)
 AS AverageOrdercost
 From Optionorder
 UNION
 Select 'StockOrder' AS type,(SUM(Price)+SUM(SOsize))/count(price)
AS AverageOrderCost
From stockorder;
```

Output

```
SQL> Select  OrderNUm , 'OptionOrder' As Type ,OOSize ,Price, (OOsize*Price)
OrderPrice
  2  from optionorder
  3  Group By OrderNUm,OOsize,Price
  4  UNION
  5  Select orderNUm,'StockOrder' AS Type, SOsize ,Price, (SOSIZE*Price) as O
rprice
  6  from stockorder
  7  Group by orderNum,SOsize,Price
  8  order by type;

  ORDERNUM TYPE            OOSIZE      PRICE ORDERPRICE
---------- ----------- ---------- ---------- ----------
       323 OptionOrder        550         51      28050
      5646 OptionOrder        100         33       3300
      9999 OptionOrder         55         41       2255
     33333 OptionOrder       3500         13      45500
     41683 OptionOrder        550          3       1650
    364636 OptionOrder        250         33       8250
         0 StockOrder         500         87      43500
    123573 StockOrder         200         35       7000
    345162 StockOrder         100         51       5100
   5782347 StockOrder         200          3        600

10 rows selected.

SQL> Select 'OptionOrder' AS type,(SUM(Price)+SUM(OOsize))/count(Price)
  2   AS AverageOrdercost
  3   From Optionorder
  4   UNION
  5   Select 'StockOrder' AS type,(SUM(Price)+SUM(SOsize))/count(price)
  6  AS AverageOrderCost
  7  From stockorder;

TYPE        AVERAGEORDERCOST
----------- ----------------
OptionOrder       863.166667
StockOrder               294
```

Incomplete

Query 6:

Finding the first name, last name, city, and account number of users who have current options assets in their account where the option’s purchase price is less than it’s current strike price.

Query:

```
SQL> Select fname, lname, city, accountno from usertable Join
  2  (select accountno from account join(select accountno, purchaseprice, strike
  3  from containsoption where purchaseprice < strike)
  4  using (accountno)) using (accountno) Order by Lname;
```

Output:

```
SQL> Select fname, lname, city, accountno from usertable Join
  2  (select accountno from account join(select accountno, purchaseprice, strike
  3  from containsoption where purchaseprice < strike)
  4  using (accountno)) using (accountno) Order by Lname;

FNAME          LNAME          CITY                           ACCOUNTNO
-------------- -------------- -------------------- ----------
FYr5RS9e5u     0hwmWapmsY     NYTWtIwAPBTUz9NWEE70       654
zsQNehhu2d     GfnmBtujwu     v4UXoqmzYswRCPWSd6bo         0
```

Part 2: Data Modification

Query 1:(Thomas)

deleting a set of tuples that is more than one but less than all the tuples in a relation

// suppose the firm is in financial desperation and is forced to cut off all clients with a balance below
// $10,000 since they do not pay enough in commissions

//must delete users information as well as account information and also any other information that could be tied to them

Instead of this:

```
DELETE FROM UserTable
WHERE ACCOUNTNO IN(
SELECT AccountNo FROM Account
```

```
WHERE Balance < 10000);

DELETE FROM ContainsStock
WHERE ACCOUNTNO IN(
SELECT AccountNo FROM Account
WHERE Balance < 10000);

DELETE FROM ContainsOption
WHERE ACCOUNTNO IN(
SELECT AccountNo FROM Account
WHERE Balance < 10000);

DELETE FROM OptionOrder
WHERE ACCOUNTNO IN(
SELECT AccountNo FROM Account
WHERE Balance < 10000);

DELETE FROM StockOrder
WHERE ACCOUNTNO IN(
SELECT AccountNo FROM Account
WHERE Balance < 10000);

DELETE FROM Account
WHERE Balance < 10000;
```

NOTE: I deleted this table, then realized a better way of doing it (disclosed below)

```
SQL> DELETE FROM UserTable
  2  WHERE ACCOUNTNO IN(
  3  SELECT AccountNo FROM Account
  4  WHERE Balance < 10000);

422 rows deleted.
```

We can go back and add FOREIGN KEY constraints with ON DELETE CASCADE OPTION

```
SQL> ALTER TABLE StockOrder
  2  DROP CONSTRAINT FK_AccountNum;

Table altered.

SQL> ALTER TABLE StockOrder
  2  ADD CONSTRAINT FK_AccountNum
  3  FOREIGN KEY (AccountNum) REFERENCES Account(AccountNo) ON DELETE CASCADE;

Table altered.

SQL> ALTER TABLE OptionOrder
  2  DROP CONSTRAINT FK_OptionAccountNum;

Table altered.

SQL>
SQL> ALTER TABLE OptionOrder
  2  ADD CONSTRAINT FK_OptionAccountNum
  3  FOREIGN KEY (AccountNum) REFERENCES Account(AccountNo) ON DELETE CASCADE;
```

```
Table altered.


SQL> ALTER TABLE ContainsStock
  2  DROP CONSTRAINT FK_AccountNO2;

Table altered.

SQL> ALTER TABLE ContainsStock
  2  ADD CONSTRAINT FK_AccountNO2
  3  FOREIGN KEY (AccountNO) REFERENCES Account(AccountNo) ON DELETE CASCADE;

Table altered.


SQL> ALTER TABLE ContainsOption
  2  DROP CONSTRAINT FK_AccountNO3;

Table altered.

SQL> ALTER TABLE ContainsOption
  2  ADD CONSTRAINT FK_AccountNO3
  3  FOREIGN KEY (AccountNo) REFERENCES Account(AccountNo) ON DELETE CASCADE;

Table altered.


SQL> ALTER TABLE UserTable
  2  DROP CONSTRAINT fk_Accno;

Table altered.

SQL> ALTER TABLE UserTable
  2  ADD CONSTRAINT fk_Accno
  3  FOREIGN KEY (AccountNo) REFERENCES Account(AccountNo) ON DELETE CASCADE;

Table altered.
```

Now, with these added constraints, we can complete the deletion more simply.

```
SQL> DELETE FROM Account
  2  WHERE BALANCE <10000;

422 rows deleted.
```

Proof of CASCADE working:

```
SQL> SELECT AccountNo FROM Account JOIN ContainsStock USING (AccountNo) WHERE Balance < 10000;
no rows selected
```

Query 2:(Thomas)

updating several tuples at once and inserting the result of a query

// change the account status of all active, "Unprofessional" accounts that are of the type 'Margin' to Professional

```
SQL> UPDATE Account
```

```
  2  SET AccountStatus = 'Prof'
  3  WHERE AccountNo IN (
  4  SELECT AccountNo FROM Account
  5  WHERE AccountStatus = 'UnProf' AND AccountType = 'Margin' AND ActivityStatus IN (
  6  SELECT ActivityStatus FROM Account
  7  GROUP BY ActivityStatus HAVING ActivityStatus = 'Active'));

614 rows updated.
```

Proof of Update:

```
SQL> SELECT * FROM Account
  2  WHERE AccountStatus = 'UnProf' AND AccountType = 'Margin' AND ActivityStatus IN (
  3  SELECT ActivityStatus FROM Account
  4  GROUP BY ActivityStatus HAVING ActivityStatus = 'Active');

no rows selected
```

Query 3:

updating several tuples at once and inserting the result of a query

// Updating the Order Action status to ‘Sell’ on any Options in the OptionOrder that have a current strike price that is greater than it’s last purchase price (Jacob).

```
SQL> Update OptionOrder SET OrderAction = 'Sell'
  2      where Ordernum IN (Select OrderNum from OptionOrder
  3     where strike > price);

2 rows updated.
```

Proof of Update:

```
Select Ordernum, OrderAction, Strike, Price from OptionOrder Where OrderAction = 'Sell' AND strike < price;

no rows selected
```

Query 4:

Update any Filled stock order with an action to buy from AON to GTC (ETHAN)

```
UPDATE Stockorder
SET Term = 'GTC'
WHERE TERM IN(
Select Term From stockorder
WHERE Term = 'AON' AND Status = 'Filled' AND OrderAction = 'Buy'
)
;
```

Output

```
SQL> UPDATE Stockorder
  2  SET Term = 'GTC'
  3  WHERE TERM IN(
  4  Select Term From stockorder
  5  WHERE Term = 'AON' AND Status = 'Filled' AND OrderAction = 'Buy'
  6  )
  7  ;

2 rows updated.
```

PROOF OF UPDATE:

```
SQL> select * from stockorder
  2  where Term = 'AON' AND OrderAction = 'Buy'
  3  ;

no rows selected
```

Query 5:

Deleting the stock data in which the price of the stock is less than $10.00 (Jacob).

```
SQL> Delete from ContainsStock
  2  Where Ticker IN (Select Ticker From StockTable Where Price < 10);

9 rows deleted.
```

Proof of delete

```
SQL> select AccountNo from containsstock Join (Select ticker from stocktable where price < 10) using (ticker);

no rows selected
```

Query 6:

Remove all Users from the UserTable with standard account types who have not placed an option order(ETHAN)

```
DELETE FROM USERTABLE
WHERE AccountNo IN(
Select AccountNO from account
WHERE AccountType = 'Standard')
AND AccountNo NOT IN(Select AccountNum
from optionorder);
```

OUTPUT

```
SQL>
SQL> DELETE FROM USERTABLE
  2  WHERE AccountNo IN(
  3  Select AccountNO from account
  4  WHERE AccountType = 'Standard')
  5  AND AccountNo NOT IN(Select AccountNum
  6  from optionorder);

2418 rows deleted.
```

Part 3: Creation of Useful Views

View 1: (Thomas)

```
SQL> CREATE VIEW Account_Stock_Unrealized_Gains AS
  2  SELECT AccountNo, PurchasePrice * Quantity AS Stock_Initial_value, Price * Quantity AS Stock_curr_Value,
  3  (((Price * Quantity) / (PurchasePrice * Quantity)) - 1) * 100 AS Unrealized_Percent_Gain_Or_Loss,
  4  (Price * Quantity) - (PurchasePrice *Quantity) AS Unrealized_Dollar_Gain_Or_Loss
  5  FROM StockTable JOIN (
  6  SELECT Ticker, Quantity, AccountNo, PurchasePrice
  7  FROM ContainsStock
  8  WHERE Quantity > 0)
  9  USING (Ticker);

View created.

SQL> SELECT * FROM Account_stock_unrealized_gains;

 ACCOUNTNO STOCK_INITIAL_VALUE STOCK_CURR_VALUE UNREALIZED_PERCENT_GAIN_OR_LOSS
UNREALIZED_DOLLAR_GAIN_OR_LOSS
```

```
---------- ------------------ ---------------- ------------------------------ ------------------------------
      2222              1245        27412.5                    2101.80723                     26167.5
      3434             88560         166545                    88.0589431                       77985
      1902            1111.5         2380.5                     114.17004                        1269
       201           8017.75        2095.21                    -73.867856                    -5922.54
        65            2101.5          842.4                    -59.914347                     -1259.1
       123              5175       11318.85                    118.721739                     6143.85
      8211            152425         9439.5                    -93.807118                   -142985.5
      6001         20828.825      20828.825                             0                           0
      4286            345.75         345.75                             0                           0
      5555              7275           7239                    -.49484536                         -36
      3000             10808          10888                    .74019245                           80

 ACCOUNTNO STOCK_INITIAL_VALUE STOCK_CURR_VALUE UNREALIZED_PERCENT_GAIN_OR_LOSS
UNREALIZED_DOLLAR_GAIN_OR_LOSS
---------- ------------------ ---------------- ------------------------------ ------------------------------
      3250             89683          89683                             0                           0
      9988               497            497                             0                           0
      5671            1759.2            126                    -92.837653                     -1633.2
      6969            4241.5         4241.5                             0                           0
      9211            1522.5            154                    -89.885057                     -1368.5
      9612               141            141                             0                           0
      2343           2121.75        2121.75                             0                           0
      4386              2625        12302.5                    368.666667                      9677.5

19 rows selected.


SQL> SELECT * FROM Account_Stock_Unrealized_Gains WHERE UNREALIZED_PERCENT_GAIN_OR_LOSS > 0;

 ACCOUNTNO STOCK_INITIAL_VALUE STOCK_CURR_VALUE UNREALIZED_PERCENT_GAIN_OR_LOSS
UNREALIZED_DOLLAR_GAIN_OR_LOSS
---------- ------------------ ---------------- ------------------------------ ------------------------------
      2222              1245        27412.5                    2101.80723                     26167.5
      3434             88560         166545                    88.0589431                       77985
      1902            1111.5         2380.5                     114.17004                        1269
       123              5175       11318.85                    118.721739                     6143.85
      3000             10808          10888                    .74019245                           80
      4386              2625        12302.5                    368.666667                      9677.5

6 rows selected.
```

Updating View:

```
SQL> INSERT INTO Account_Stock_Unrealized_gains VALUES (1123123, 10, 10, 10, 10);
INSERT INTO Account_Stock_Unrealized_gains VALUES (1123123, 10, 10, 10, 10)
*
ERROR at line 1:
ORA-01779: cannot modify a column which maps to a non key-preserved table
```

This view is not updatable because it is partially contrived from the ContainsStock table which imposes a foreign key constraint on the account number.

View 2:(Ethan)

Display the total sales in dollars of both options and stocks, as well as the largest order bought
From both

```
SQL> Create VieW Totalsales AS
  2  Select 'OptionOrders' AS Type, SUM(OOsize*Price) AS totalsales,MAX(OOsize*P
rice) AS LargestBuy
  3  from optionorder
  4  UNION
  5  Select 'StockOrders' AS TYpe,SUM(SOsize*Price) AS totalsales,MAX(SOsize*Pri
ce) AS LargestBuy
  6  from stockorder;

View created.


SQL> select * from totalsales;

TYPE         TOTALSALES LARGESTBUY
------------ ---------- ----------
OptionOrders      89005      45500
StockOrders       56200      43500
```

It doesn’t make sense to insert a tuple into this view as it aggregates all of the “total sales” and largest purchases into a compact table.

View 3: (Thomas)

```
SQL> CREATE VIEW Premium_Clients AS
  2  SELECT UserID, AccountNo, Fname, LName, City
  3  FROM UserTable
  4  WHERE AccountNo IN (
  5  SELECT AccountNo
  6  FROM Account
  7  WHERE BALANCE > 220000 AND AccountStatus = 'Prof' AND ActivityStatus = 'Active');

View created.


SQL> SELECT * FROM Premium_Clients WHERE AccountNo BETWEEN 1000 AND 1100;

USERID    ACCOUNTNO FNAME           LNAME           CITY
-------- ---------- --------------- --------------- --------------------
47824477       1054 frLtswngbm      MOfFYoY5a2      evsqEOFfwwyETY2geHNI
78886852       1067 nCAgsiMmy3      onObVSpgv0      navV51kc0wHzH77aj1Yu
79619892       1074 75oN4yItSs      mK5p70aqpN      abSt43X3wC4CIYt3B3sA
15167777       1082 rQ3V8sQCXW      XrUTOniYq3      NkxSPrHBEzQoA2pwNedS
89947129       1092 NIGY6S7v0U      ruPDL6wgDI      pYTDJoaO4SJnBDz2k37V
78839393       1095 bANfhECXrZ      8pUOe9JlvJ      6fOzNCVriT5hUzP4UTx2
```

```
6 rows selected.

SQL> INSERT INTO Premium_Clients VALUES ('99999', 1870, 'Joe', 'Shmoe', 'Atlanta');
1 row created.

SQL> SELECT * FROM Premium_Clients WHERE FName = 'Joe';
no rows selected
```

This has occurred because the premium clients view will display the user information of clients whose balance is above 220,000. Although the insertion of Joe Shmoe appeared to be successful, this tuple does not show up in the view because we have not specified any balance information for them.

```
SQL> SELECT * FROM UserTable WHERE FNAME = 'Joe';

USERID   ACCOUNTNO FNAME          LNAME          ADDRESS                        SIN        PHONENO    CITY
-------- ---------- -------------- -------------- ------------------------------ ---------- ---------- --------------------
99999          1870 Joe            Shmoe                                                              Atlanta
```

The insertion instead placed Joe Shmoe into the Usertable.

Work Outline:

Thomas:

- Queries 1 & 2 part 1 and part 2
- View 1 & 3 part 3

Ethan:

- Queries 3 & 4 for part1, 4 & 6 for part 2
- View 2 for part 3

Jacob:

- Queries 5 & 6 part 1
- Queries 3 & 5 part 2