DBQL BANK

MSIS – 2603 : Database Management Systems

Zexi Zhang (W1189747)
John Antony (W1189757)
Yue Wu (W1188513)
Rohit Jacob (W1187417)

SANTA CLARA UNIVERSITY

DBQL Bank – Final Report

Business Application Description

DBQL or Double Barbeque Query Lollypop (not Database Query Language) Bank is a financial institution that provides a unique banking solution for its customers both in-terms of maintaining a savings/checking account or while applying for a loan. Our bank will be using three core applications for the following three functionalities:

1. Loan origination and disbursement
2. Loan collection, repayment and intimation
3. Maintenance of savings and checking accounts

We will also be monitoring each branches, employees, loan types to understand which of these are generating more business. Monitoring these Key Performance Indicators helps us re-asses our loan products, employee productivity and branch performance.

This complete solution helps our bank run a scalable, responsive and efficient business with clear targets. This will help each branch, and therefore the bank, be a successful business.

Types of Users

1. **Bank Rep**

   A Bank Rep is the face of the bank to our customers. He is the one who assists customers in opening an account, provides a quotation to customers looking for loans, completes a loan application by generating a loan ID and also recommends best options for our customers.

2. **Branch Manager**

   A Branch Manager has the most important role in the branch. He manages the debt to current assets ratio within the bank, he is the final authority for approving loans, he monitors the employee performance and facilitates the premium customers with better interest rates. He also monitors customers with a good credit rating and credit history to promote them to the premium class of customers.

3. **Credit Analyst**

   A Credit Analyst plays an important role of analyzing the credit a customer who has either applied for a loan or a new account. His role is to check the external credit scores (based on the customer's social security), analyze and generate an internal credit score (only for the bank) using information like current salary, credit history and so on.

4. **Collection Agent**

A Collection Agent operates just as his designation suggests. He is responsible for collecting the loan amount pending to be repaid from the customer, he sends out reminders in-case the customer has not paid it past his due date and also sends out a list of customers who have violated payments after the 2nd reminder to the Branch Manager for sending out a legal notice.

5. **Customer**

The Customer is anyone who is looking to either open a bank account or apply for a loan. The customer can also check his eligibility of a loan before he applies for a loan.

Use Cases for the Application

1. **Bank Rep**
    a. A Bank Rep can open or close a customer account.
    b. He can update customer information ranging from SSN to change in address or change in name.
    c. He is responsible for collecting all the relevant documents from the customer applying for a loan and entering this information in the system to generate a loan ID.
    d. It's his responsibility to change the loan status from "Application Completed" stage to "Under Review"
    e. When a customer is applying for a loan, he can provide a choice of payment plans based on loan type.

2. **Branch Manager**
    a. He is responsible of upgrading or downgrading the customers to premium tier based on their credit history and account balance with the bank. This provides higher interest percentage to customers.
    b. He is the final approver of the loan process. He reviews the information updated by the Credit Analyst and the Bank Rep.
    c. He checks outstanding loans and corresponding current status and based on the information, he may or may not decide to send a legal notice for repayment.
    d. He is responsible for hiring or retire employees

3. **Credit Analyst**
    a. Keeps a track record of external and internal credit scores of a customer.
    b. Updates the internal credit score of a new applicant based on his external credit scores and current assets (Income and property)

c. Update external credit score based on outside credit unions
d. Search customer with good external/internal credit score
e. Once he has generated the internal credit score, he is responsible for updating the status.

4. **Collection Agent**
    a. He is responsible for sending a loan repayment reminder to customers who have passed their due date of payment
    b. He sends out the defaulter list to the Branch Manager for sending out Legal Notices.

5. **Customer**
    a. A customer with the bank can update his basic information like phone number, address and so on.
    b. Check account details using his customer ID
    c. Check transaction history for a particular date/time or above particular amount
    d. Apply for a loan online and generate a high-level quote.
    e. He can even check loan status from the application stage to the paid-off stage.

Queries for Users in Business Terms

1. **Bank Rep**
    a. Open a checking and saving account for customer C031.
    b. Process customer loan request LQ001.
    c. Change customer C001's L001 status to PEDING_DOCUMENT
    d. Provide payment plan for client's loan request ID LQ002 based on customer type.

2. **Branch Manager**
    a. Check all the premium customer of branch CA001.
    b. Check loan status of customer L006 and update loan status to Approved.
    c. Search employee E001's quota number.
    d. Retire employee E020.

3. **Credit Analyst**
    a. Update customer's C025's current internal credit score to 92.
    b. Search customers' internal credit score is equal or greater of 85.
    c. Change customer C001 loan L002 status to credit GEN completed. also insert his credit score into credit Gen table.
    d. updated customer C021's external score to 87.

4. **Collection Agent**
    a. Change loan L033 status to LEGAL ACTION

5. **Customer**
    a. Customer C003 check his all transaction history in 2015.
    b. Customer C004 request a loan quote online. Insert a new item in loan request table.
    c. Customer C005 check his loans and status.
    d. Customer C001 update his home address to '1050 Benton St, Santa Clara, CA.

Business Matrix

1. **Bank Rep**
    a. Check the number of new customers on boarded against his target
    b. Check total amount of loan disbursed per month

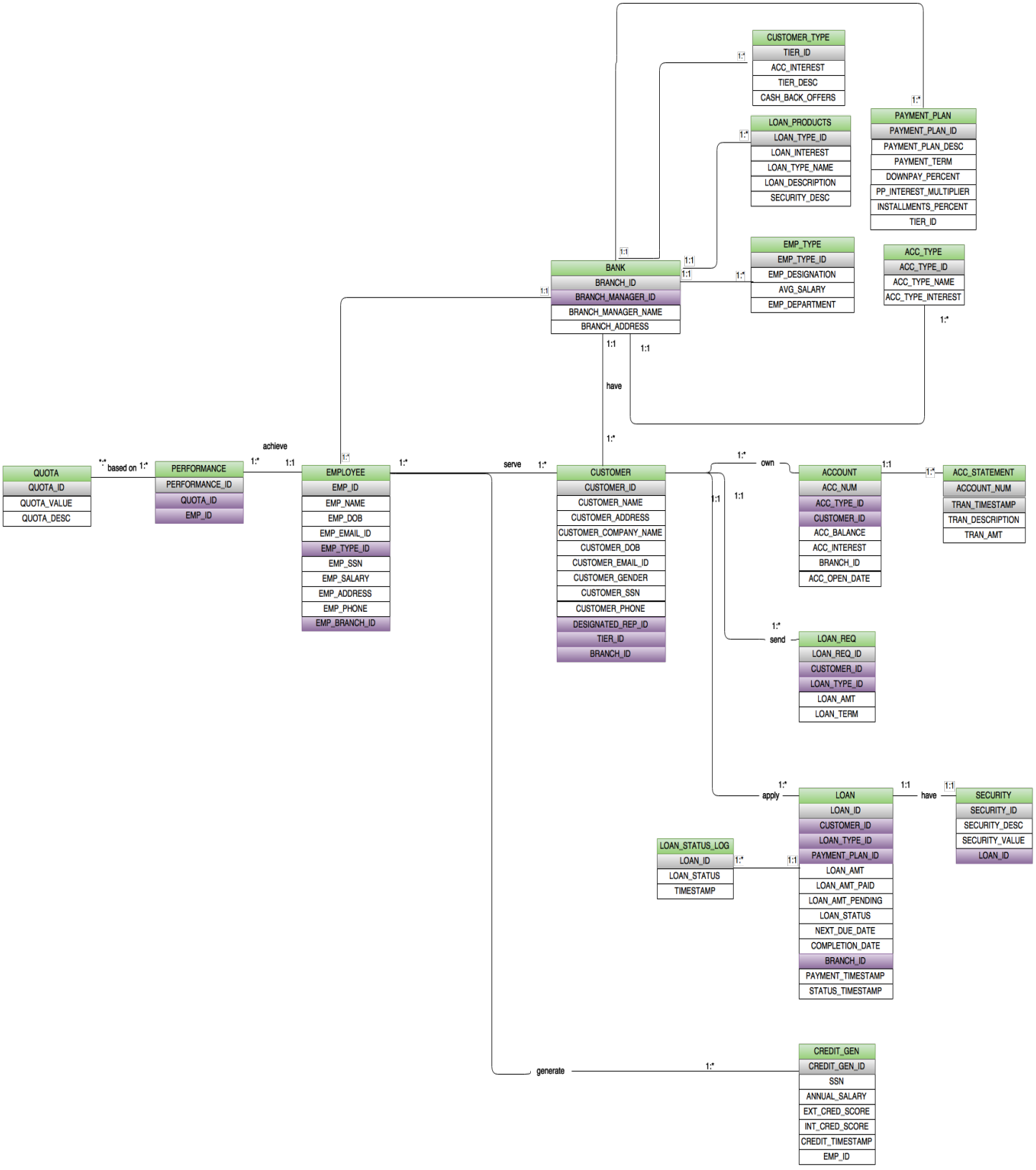
2. **Branch Manager**
    a. Check average amount of savings for all branches

3. **Credit Analyst**
    a. Number of credits generated per month

4. **Collection Agent**
    a. Total loan collection per month

5. **Branch / Bank**
    a. Ratio of Current Assets to the Loan Amount to be collected.
    b. Average amount of Account Balance in each Branch
    c. Total Loans waiting to be Approved (for predicting next month / quarter business)

Logical Schema – UML



Physical Schema

Note: Table Names in **BOLD**, Primary Key in **BOLD and UNDERLINED** and Foreign Keys in *Italics*

1. **CUSTOMER_TYPE** (**TIER_ID**, ACC_INTEREST, TIER_DESC, CASH_BACK_OFFERS)

2. **PAYMENT_PLAN** (**PAYMENT_PLAN_ID**, PAYMENT_PLAN_DESC, PAYMENT_TERM, PP_INTEREST_MULTIPLIER, DOWNPAY_PERCENT, INSTALLMENTS_PERCENT, TIER_ID)

3. **LOAN_PRODUCTS** (**LOAN_TYPE_ID**, LOAN_INTEREST, LOAN_TYPE_NAME, LOAN_DESCRIPTION, SECURITY_DESC)

4. **ACC_TYPE** (**ACC_TYPE_ID**, ACC_TYPE_NAME, ACC_TYPE_INTEREST)

5. **EMP_TYPE** (**EMP_TYPE_ID**, EMP_DESIGNATION, AVG_SALARY, EMP_DEPARTMENT)

6. **CUSTOMER** (**CUSTOMER_ID**, CUSTOMER_NAME, CUSTOMER_ADDRESS, CUSTOMER_COMPANY_NAME, CUSTOMER_DOB, CUSTOMER_EMAIL_ID, CUSTOMER_GENDER, CUSTOMER_SSN, CUSTOMER_PHONE, *DESIGNATED_REP_ID*, *TIER_ID*, *BRANCH_ID*)

7. **EMPLOYEE** (**EMP_ID**, EMP_NAME, EMP_DOB, EMP_EMAIL_ID, *EMP_TYPE_ID*, EMP_SSN, EMP_SALARY, EMP_ADDRESS, EMP_PHONE, *EMP_BRANCH_ID*)

8. **PERFORMANCE** (**PERFORMANCE_ID**, *EMP_ID*, *QUOTA_ID*)

9. **LOAN_REQ** (**LOAN_REQ_ID**, *CUSTOMER_ID*, *LOAN_TYPE_ID*, LOAN_AMT, LOAN_TERM)

10. **SECURITY** (**SECURITY_ID**, SECURITY_TYPE, SECURITY_DESC, SECURITY_VALUE, *LOAN_ID*)

11. **LOAN_STATUS_LOG** (**LOAN_ID**, LOAN_STATUS, TIMESTAMP, LOAN_STATUS_LOG_ID)

12. **LOAN** (**LOAN_ID**, *CUSTOMER_ID*, *LOAN_TYPE_ID*, *PAYMENT_PLAN_ID*, LOAN_AMT, LOAN_AMT_PAID, LOAN_AMT_PENDING, LOAN_STATUS, NEXT_DUE_DATE, COMPLETION_DATE, *BRANCH_ID*, PAYMENT_TIMESTAMP, STATUS_TIMESTAMP)

13. **CREDIT_GEN** (**CREDIT_GEN_ID**, SSN, ANNUAL_SALARY, EXT_CRED_SCORE, INT_CRED_SCORE, CREDIT_TIMESTAMP, EMP_ID)

14. **BANK** (**BRANCH_ID**, BRANCH_MANAGER_ID, BRANCH_MANAGER_NAME, BRANCH_ADDRESS)

15. **QUOTA** (**QUOTA_ID**, QUOTA_VALUE, QUOTA_DESC)

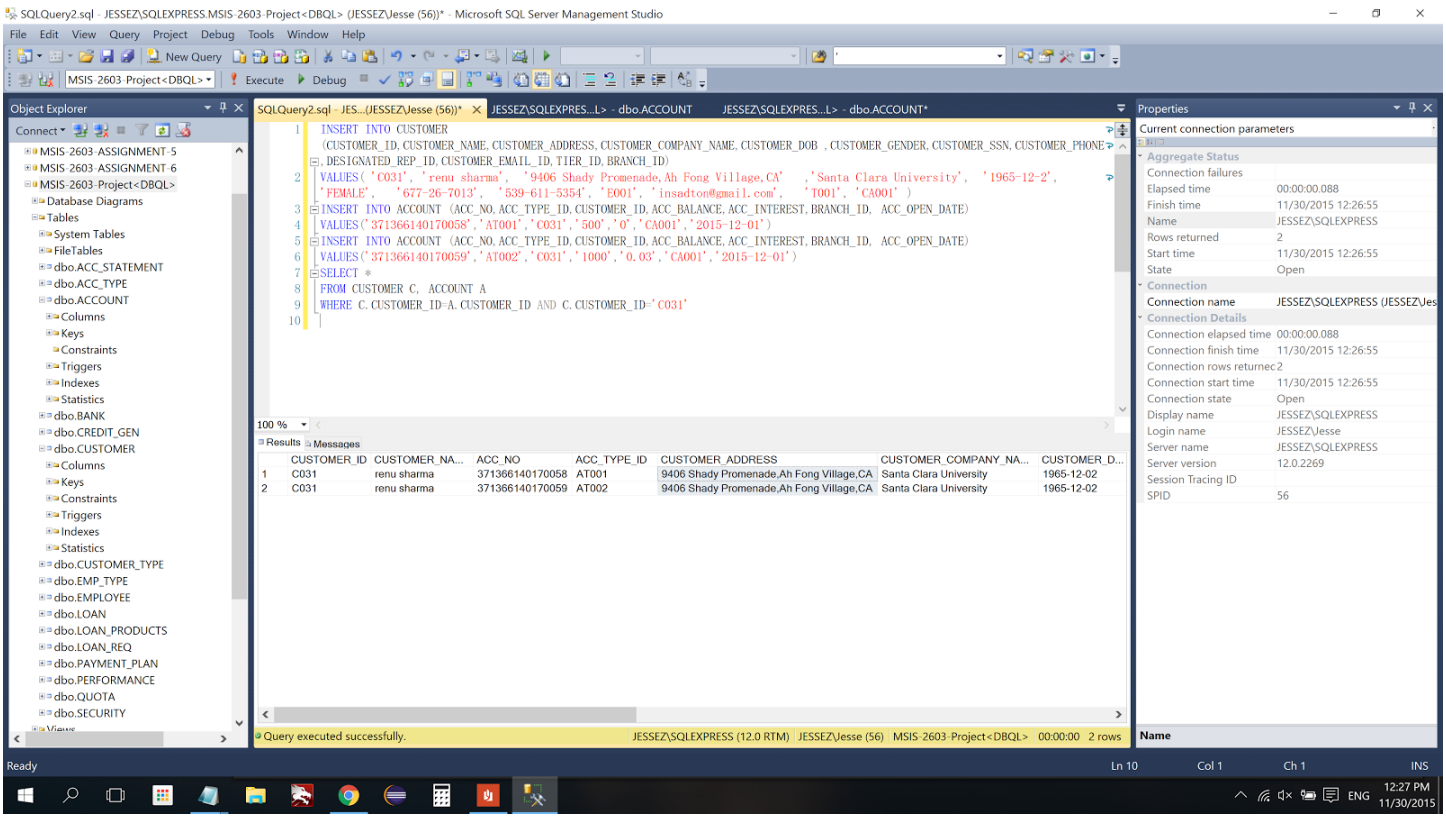
16. **ACC_STATEMENT** (**ACC_NO**, **TRAN_TIMESTAMP**, TRAN_DESC, TRAN_AMT)

17. **ACCOUNT** (**ACC_NO**, *ACC_TYPE_ID*, *CUSTOMER_ID*, ACC_BALANCE, ACC_INTEREST, BRANCH_ID, ACC_OPEN_DATE)

Use Cases

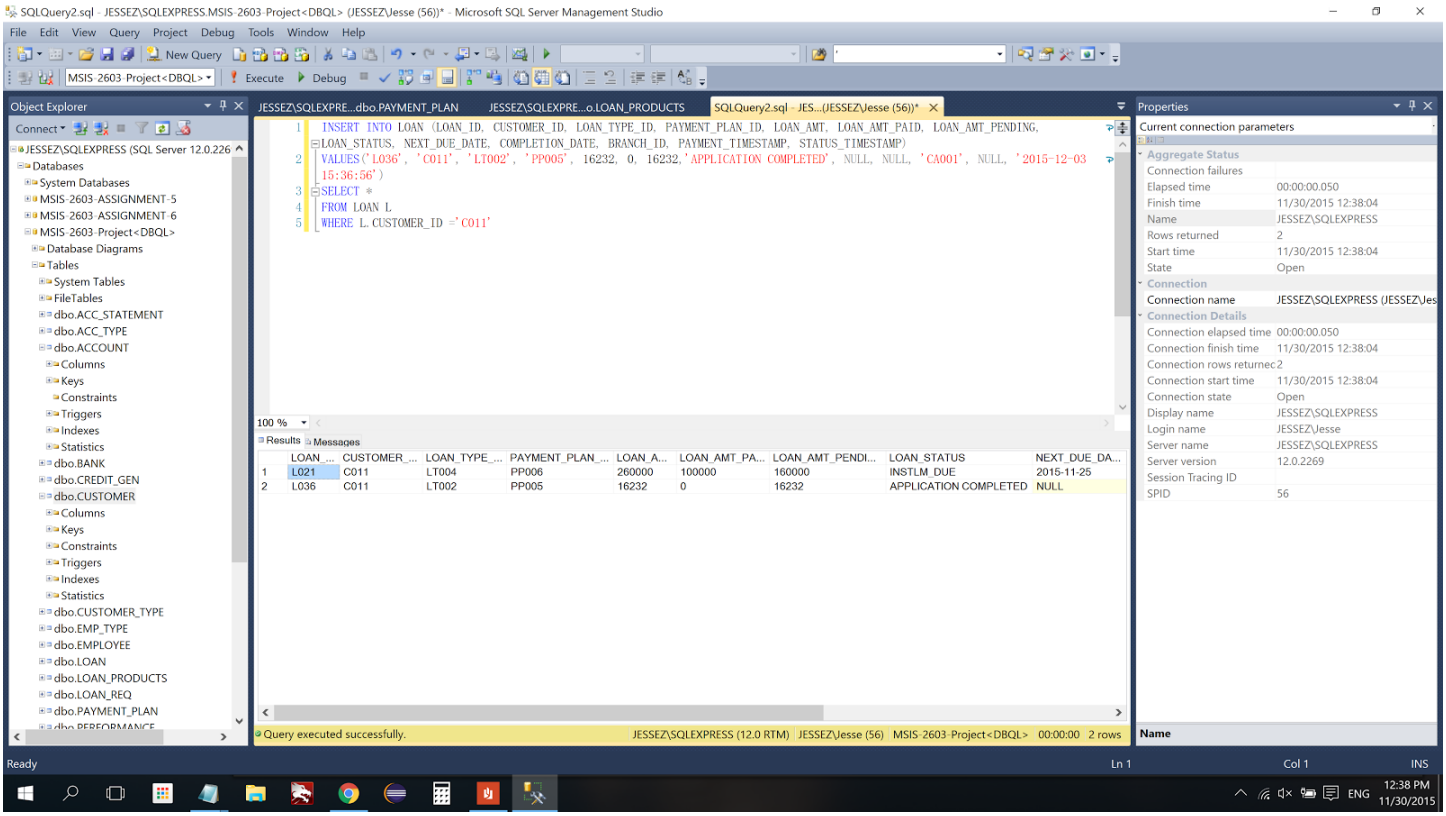
1. **Bank Rep**
    a. Open a checking and saving account for customer C031.

```
INSERT INTO
CUSTOMER(CUSTOMER_ID,CUSTOMER_NAME,CUSTOMER_ADDRESS,CUSTOMER_COMPANY_N
AME,CUSTOMER_DOB
,CUSTOMER_GENDER,CUSTOMER_SSN,CUSTOMER_PHONE,DESIGNATED_REP_ID,CUSTOME
R_EMAIL_ID,TIER_ID,BRANCH_ID)
VALUES(    'C031',    'renu sharma',   '9406 Shady Promenade,Ah Fong
Village,CA'      ,'Santa Clara University', '1965-12-2',     'FEMALE',
      '677-26-7013',   '539-611-5354', 'E001',    'insadton@gmail.com',
      'T001',    'CA001'    )
INSERT INTO ACCOUNT
(ACC_NO,ACC_TYPE_ID,CUSTOMER_ID,ACC_BALANCE,ACC_INTEREST,BRANCH_ID,
ACC_OPEN_DATE)
VALUES('371366140170058','AT001','C031','500','0','CA001','2015-12-
01')
INSERT INTO ACCOUNT
(ACC_NO,ACC_TYPE_ID,CUSTOMER_ID,ACC_BALANCE,ACC_INTEREST,BRANCH_ID,
ACC_OPEN_DATE)
VALUES('371366140170059','AT002','C031','1000','0.03','CA001','2015-
12-01')
SELECT *
FROM CUSTOMER C, ACCOUNT A
WHERE C.CUSTOMER_ID=A.CUSTOMER_ID AND C.CUSTOMER_ID='C031'
```



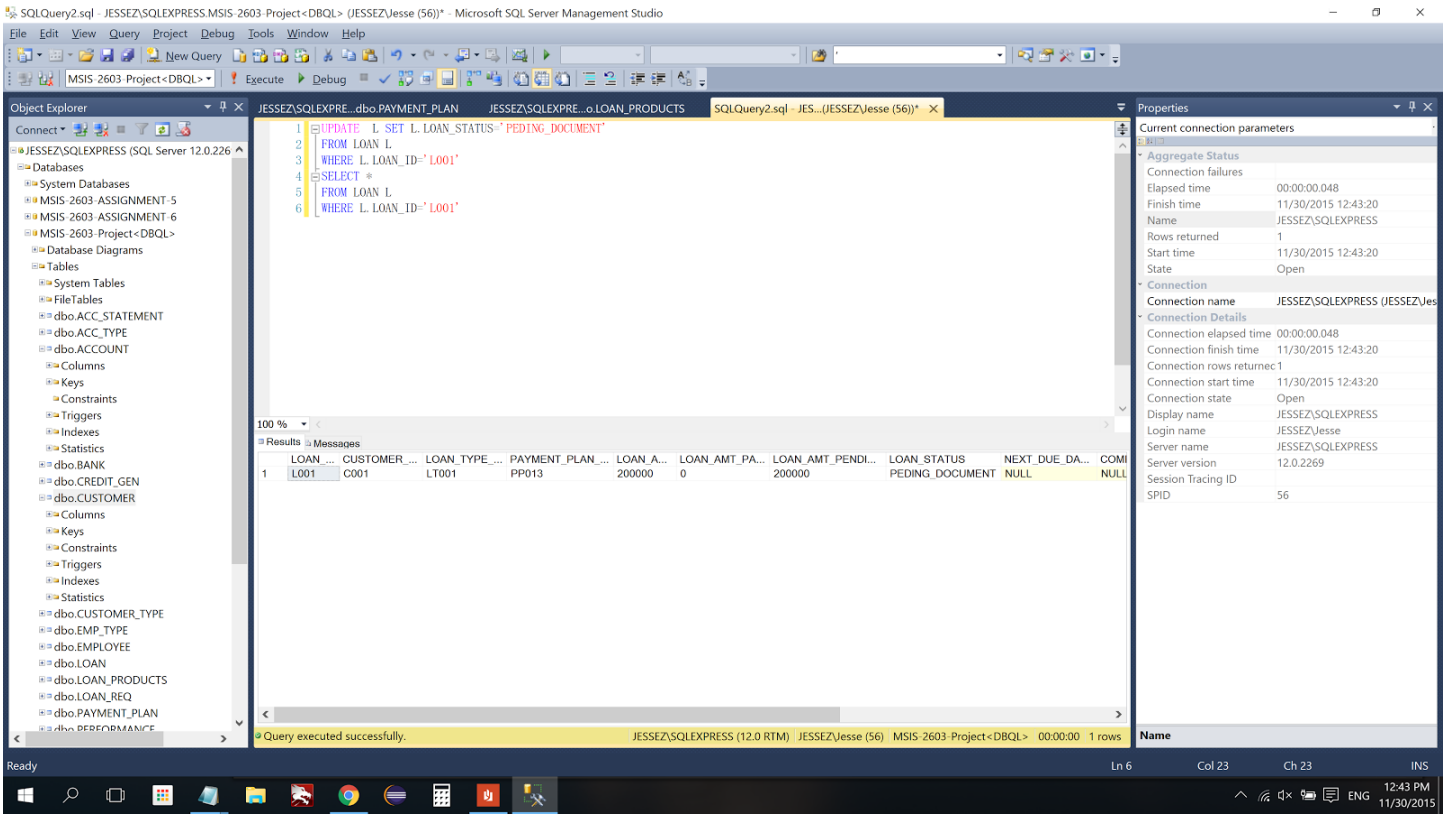
b. Process customer loan request LQ001.

```
INSERT INTO LOAN (LOAN_ID, CUSTOMER_ID, LOAN_TYPE_ID, PAYMENT_PLAN_ID,
LOAN_AMT, LOAN_AMT_PAID, LOAN_AMT_PENDING,  LOAN_STATUS,
NEXT_DUE_DATE, COMPLETION_DATE, BRANCH_ID, PAYMENT_TIMESTAMP,
STATUS_TIMESTAMP)
VALUES('L036', 'C011', 'LT002', 'PP005', 16232, 0, 16232,'APPLICATION
COMPLETED', NULL, NULL, 'CA001', NULL, '2015-12-03 15:36:56')
SELECT *
FROM LOAN L
WHERE L.CUSTOMER_ID ='C011'
```



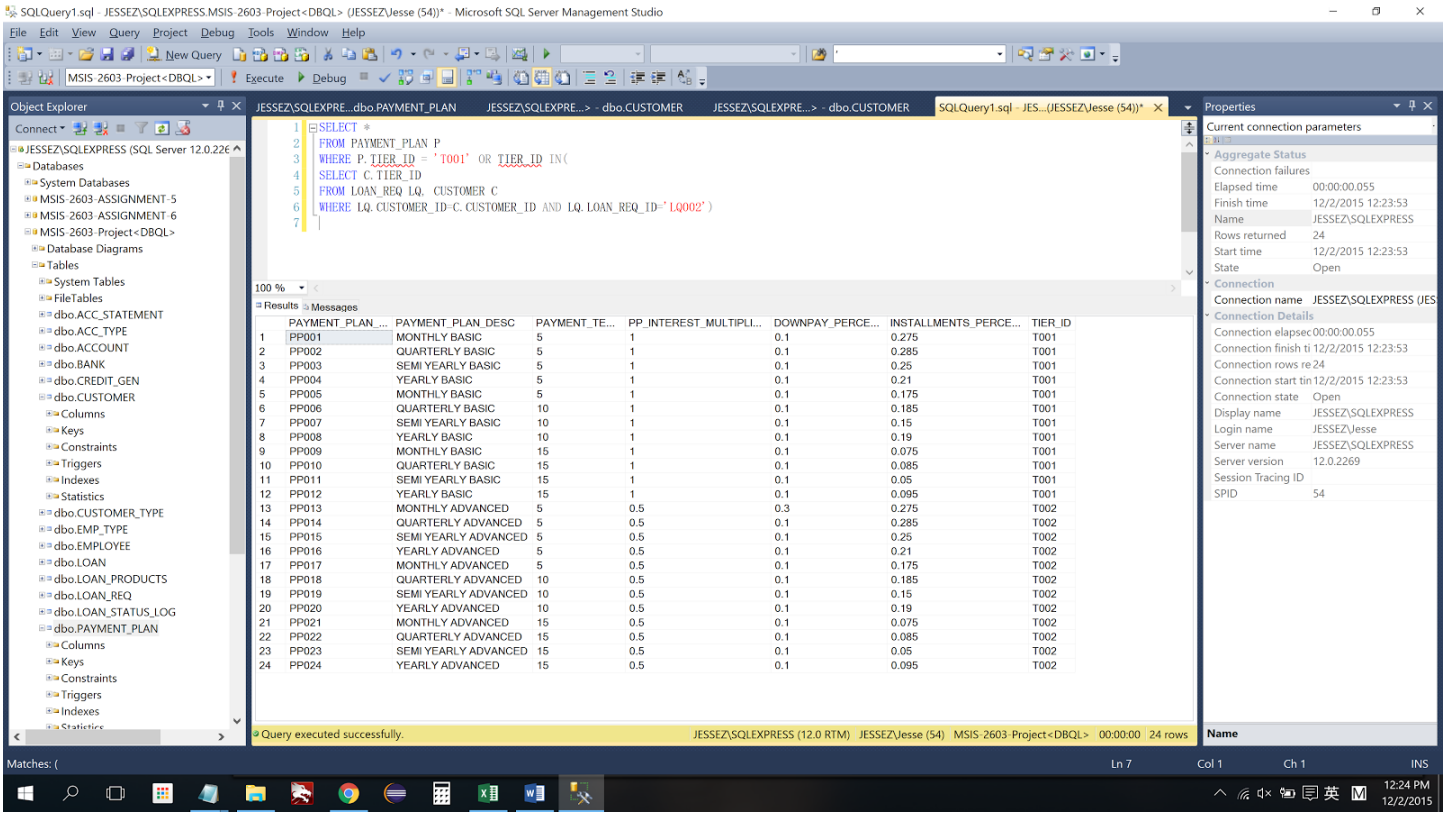
c. Change customer C001's L001 status to PEDING_DOCUMENT

```
UPDATE  L SET L.LOAN_STATUS='PEDING_DOCUMENT'
FROM LOAN L
WHERE L.LOAN_ID='L001'
SELECT *
FROM LOAN L
WHERE L.LOAN_ID='L001'
```



d. Provide payment plan for client's loan request ID LQ002 based on customer type.

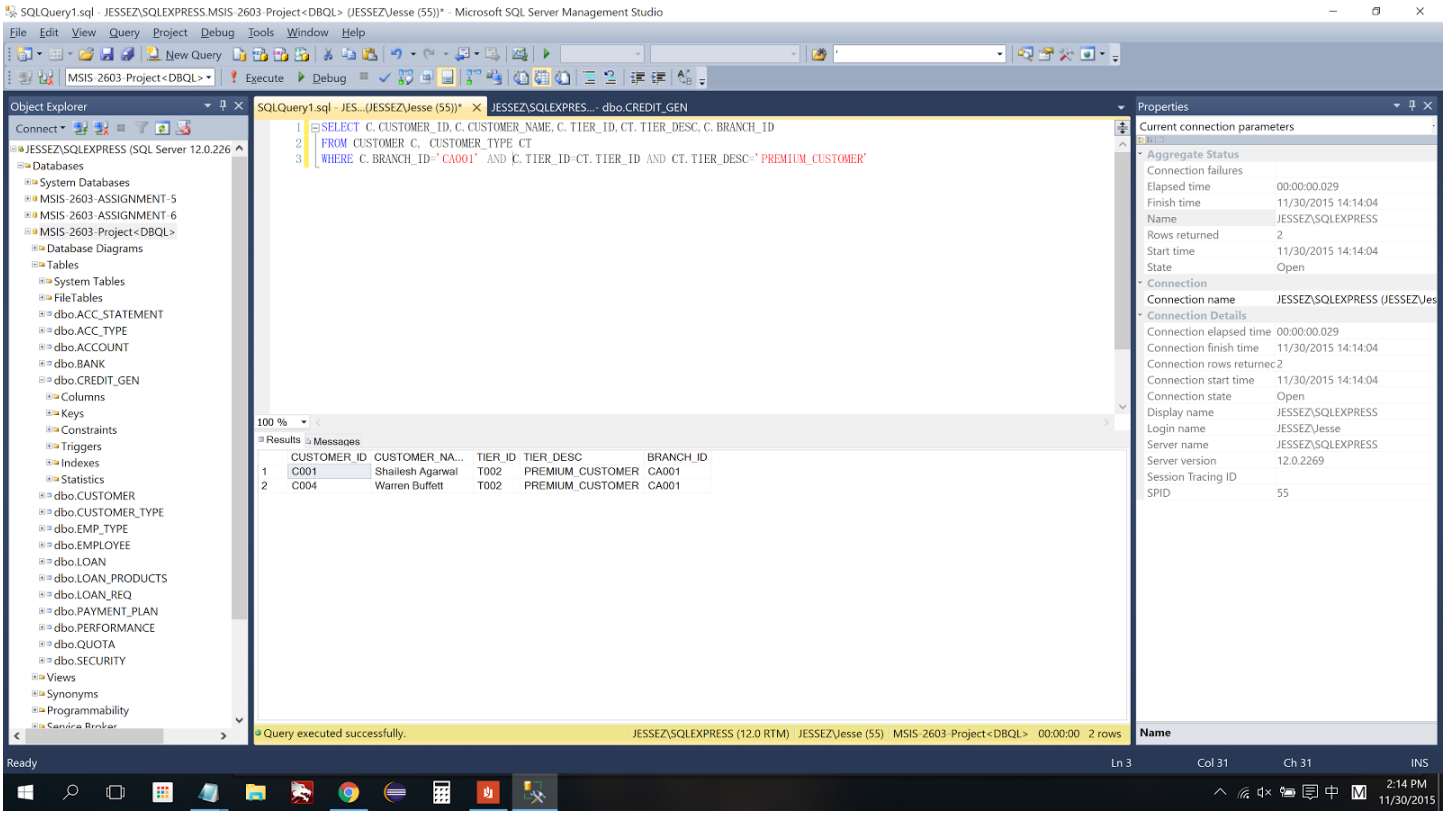
```
SELECT *
FROM PAYMENT_PLAN P
WHERE P.TIER_ID = 'T001' OR TIER_ID IN(
SELECT C.TIER_ID
FROM LOAN_REQ LQ, CUSTOMER C
WHERE LQ.CUSTOMER_ID=C.CUSTOMER_ID AND LQ.LOAN_REQ_ID='LQ002')
```



	PAYMENT_PLAN_...	PAYMENT_PLAN_DESC	PAYMENT_TE...	PP_INTEREST_MULTIPLI...	DOWNPAY_PERCE...	INSTALLMENTS_PERCE...	TIER_ID
1	PP001	MONTHLY BASIC	5	1	0.1	0.275	T001
2	PP002	QUARTERLY BASIC	5	1	0.1	0.285	T001
3	PP003	SEMI YEARLY BASIC	5	1	0.1	0.25	T001
4	PP004	YEARLY BASIC	5	1	0.1	0.21	T001
5	PP005	MONTHLY BASIC	5	1	0.1	0.175	T001
6	PP006	QUARTERLY BASIC	10	1	0.1	0.185	T001
7	PP007	SEMI YEARLY BASIC	10	1	0.1	0.15	T001
8	PP008	YEARLY BASIC	10	1	0.1	0.19	T001
9	PP009	MONTHLY BASIC	15	1	0.1	0.075	T001
10	PP010	QUARTERLY BASIC	15	1	0.1	0.085	T001
11	PP011	SEMI YEARLY BASIC	15	1	0.1	0.05	T001
12	PP012	YEARLY BASIC	15	1	0.1	0.095	T001
13	PP013	MONTHLY ADVANCED	5	0.5	0.3	0.275	T002
14	PP014	QUARTERLY ADVANCED	5	0.5	0.1	0.285	T002
15	PP015	SEMI YEARLY ADVANCED	5	0.5	0.1	0.25	T002
16	PP016	YEARLY ADVANCED	5	0.5	0.1	0.21	T002
17	PP017	MONTHLY ADVANCED	5	0.5	0.1	0.175	T002
18	PP018	QUARTERLY ADVANCED	10	0.5	0.1	0.185	T002
19	PP019	SEMI YEARLY ADVANCED	10	0.5	0.1	0.15	T002
20	PP020	YEARLY ADVANCED	10	0.5	0.1	0.19	T002
21	PP021	MONTHLY ADVANCED	15	0.5	0.1	0.075	T002
22	PP022	QUARTERLY ADVANCED	15	0.5	0.1	0.085	T002
23	PP023	SEMI YEARLY ADVANCED	15	0.5	0.1	0.05	T002
24	PP024	YEARLY ADVANCED	15	0.5	0.1	0.095	T002

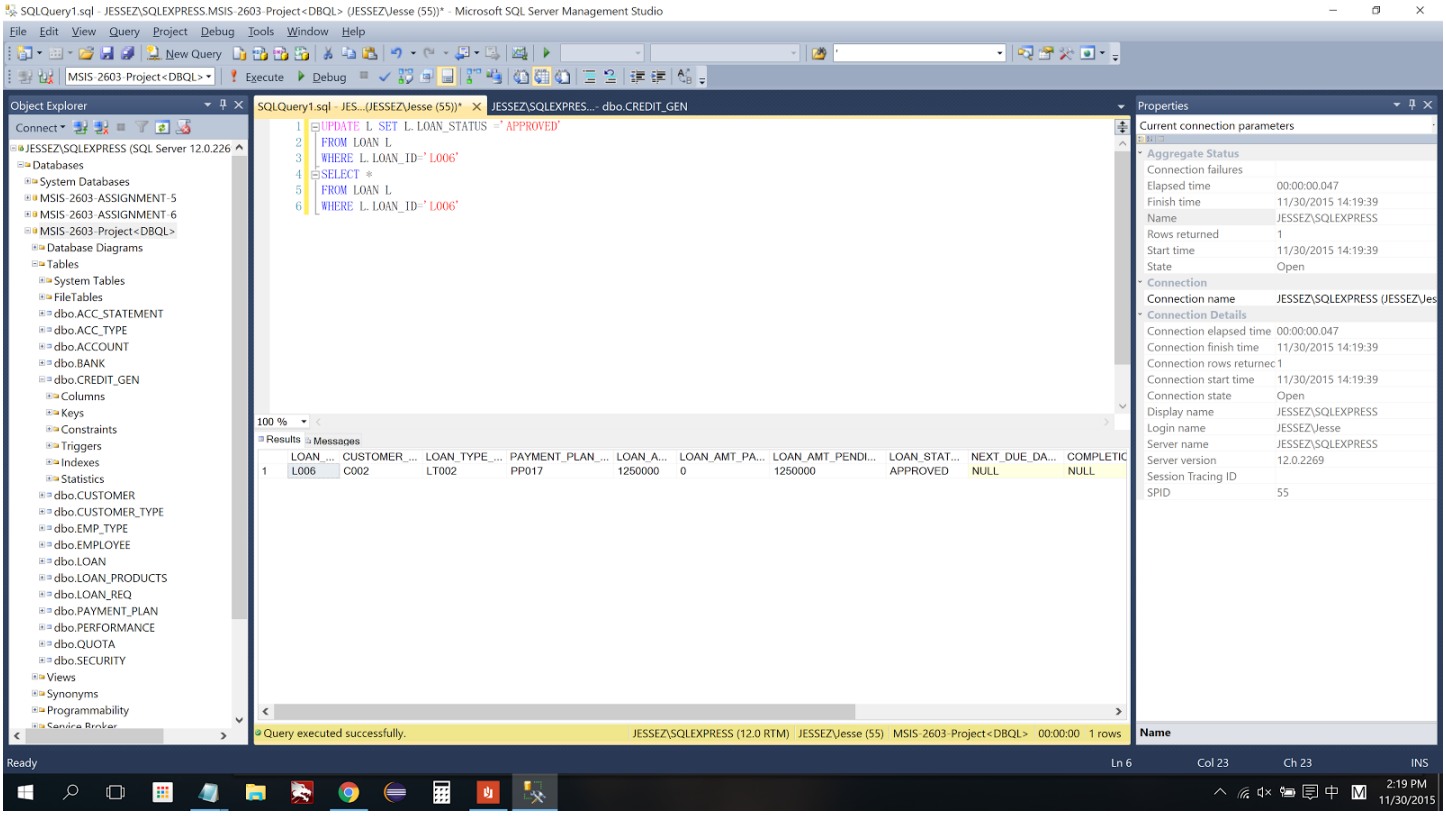
2. **Branch Manager**
    a. Check all the premium customer of branch CA001.

```
SELECT
C.CUSTOMER_ID,C.CUSTOMER_NAME,C.TIER_ID,CT.TIER_DESC,C.BRANCH_ID
FROM CUSTOMER C, CUSTOMER_TYPE CT
WHERE C.BRANCH_ID='CA001' AND C.TIER_ID=CT.TIER_ID AND
CT.TIER_DESC='PREMIUM_CUSTOMER'
```



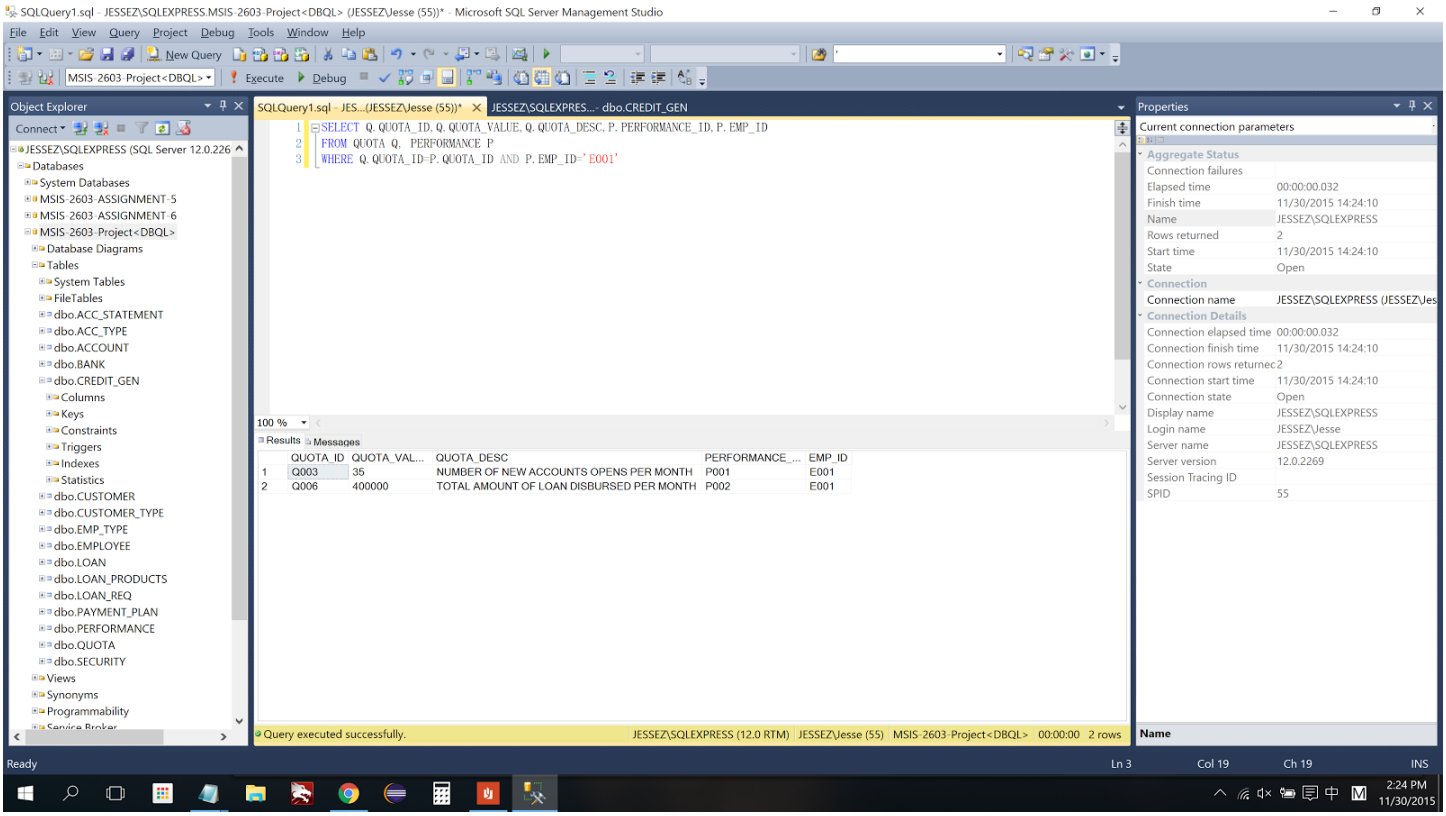
b. Check loan status of customer L006 and update loan status to Approved.

```
UPDATE L SET L.LOAN_STATUS ='APPROVED'
FROM LOAN L
WHERE L.LOAN_ID='L006'
SELECT *
FROM LOAN L
WHERE L.LOAN_ID='L006'
```



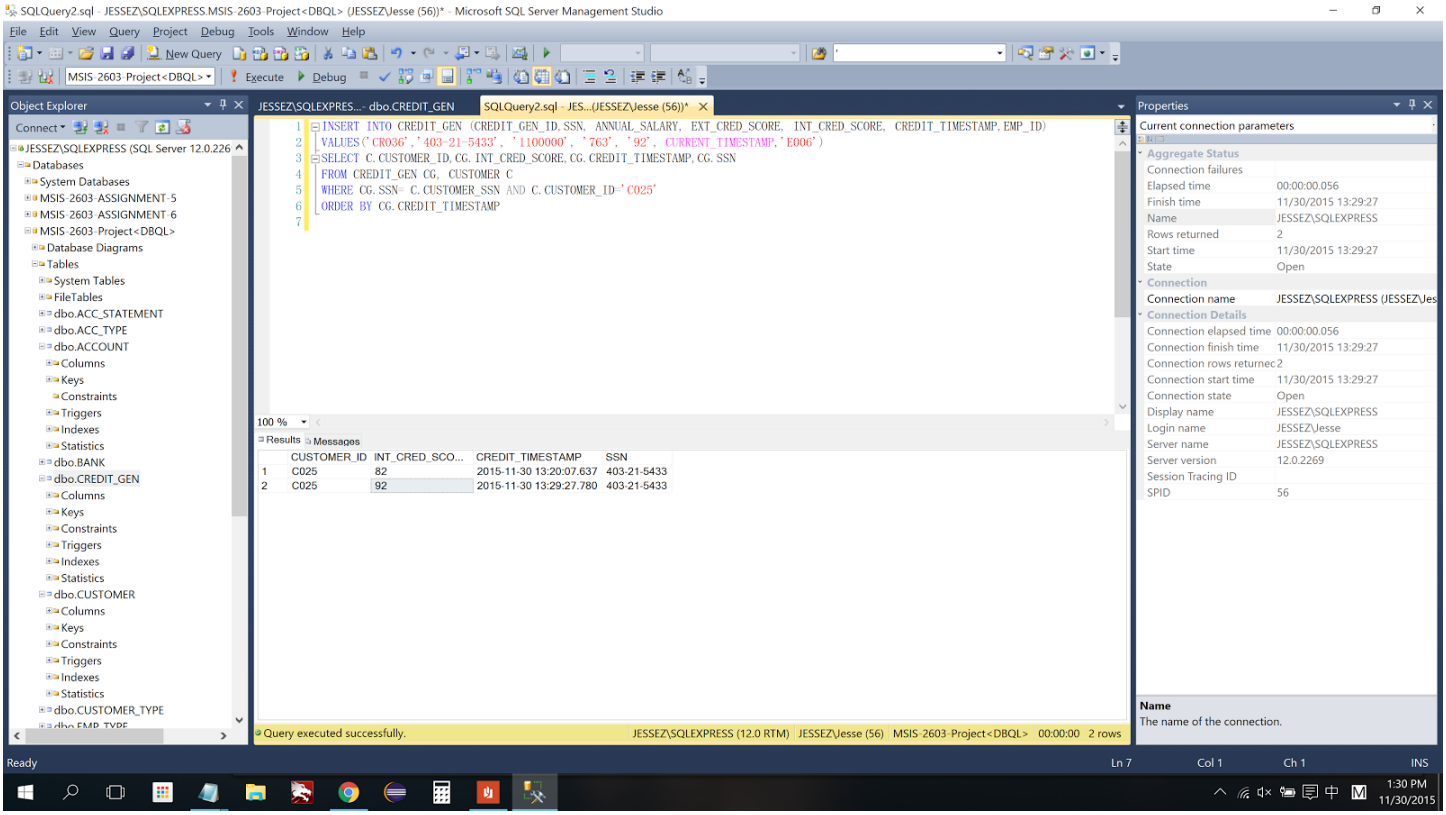
c. Search employee E001's quota number.

```
SELECT Q.QUOTA_ID,Q.QUOTA_VALUE,Q.QUOTA_DESC,P.PERFORMANCE_ID,P.EMP_ID
FROM QUOTA Q, PERFORMANCE P
WHERE Q.QUOTA_ID=P.QUOTA_ID AND P.EMP_ID='E001'
```



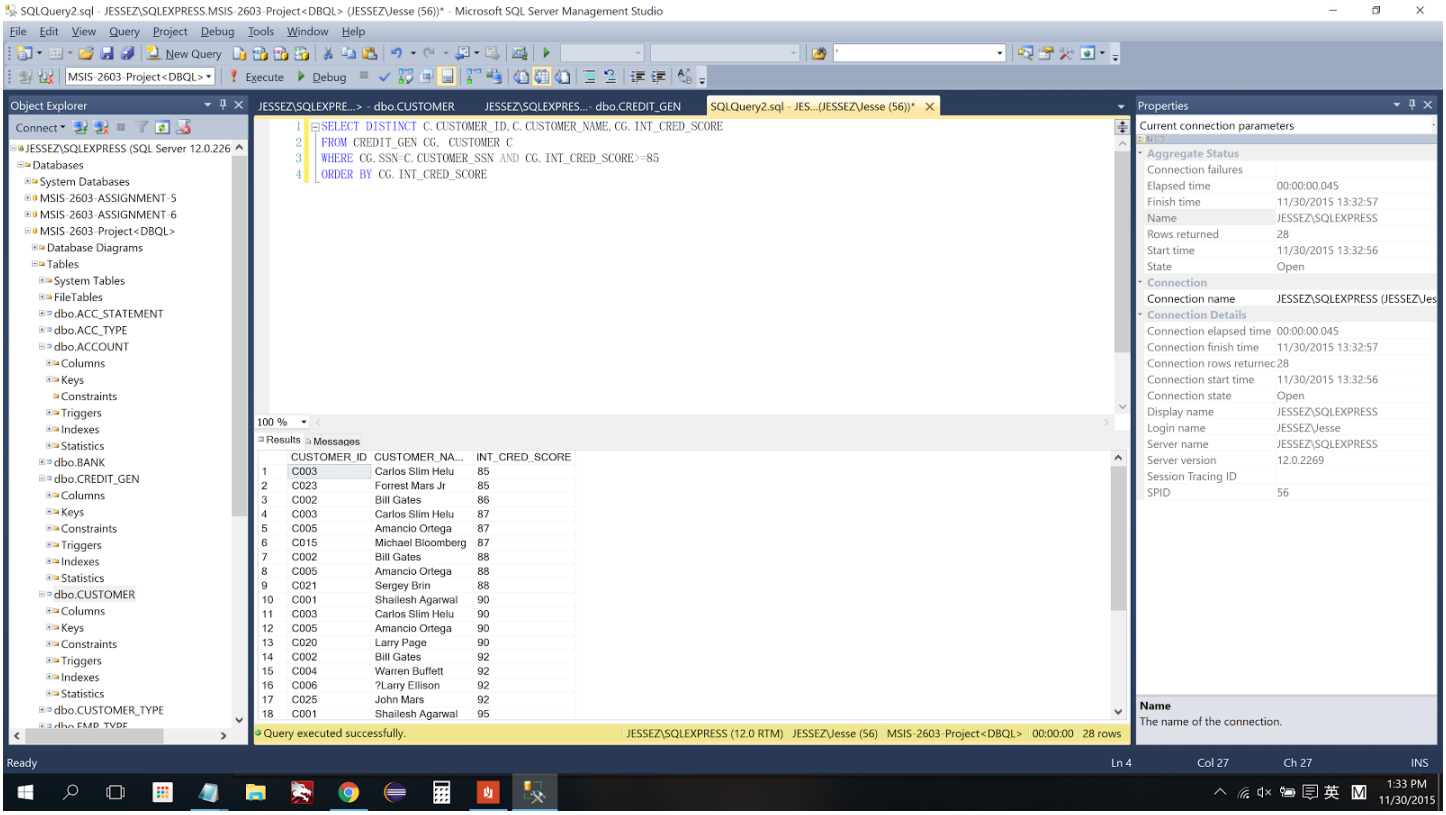
3. **Credit Analyst**
    a. Update client C025's current internal credit score to 92.

```
INSERT INTO CREDIT_GEN (CREDIT_GEN_ID,SSN, ANNUAL_SALARY,
EXT_CRED_SCORE, INT_CRED_SCORE, CREDIT_TIMESTAMP,EMP_ID)
VALUES('CR036','403-21-5433', '1100000', '763', '92',
CURRENT_TIMESTAMP,'E006')
SELECT C.CUSTOMER_ID,CG.INT_CRED_SCORE,CG.CREDIT_TIMESTAMP,CG.SSN
FROM CREDIT_GEN CG, CUSTOMER C
WHERE CG.SSN= C.CUSTOMER_SSN AND C.CUSTOMER_ID='C025'
ORDER BY CG.CREDIT_TIMESTAMP
```



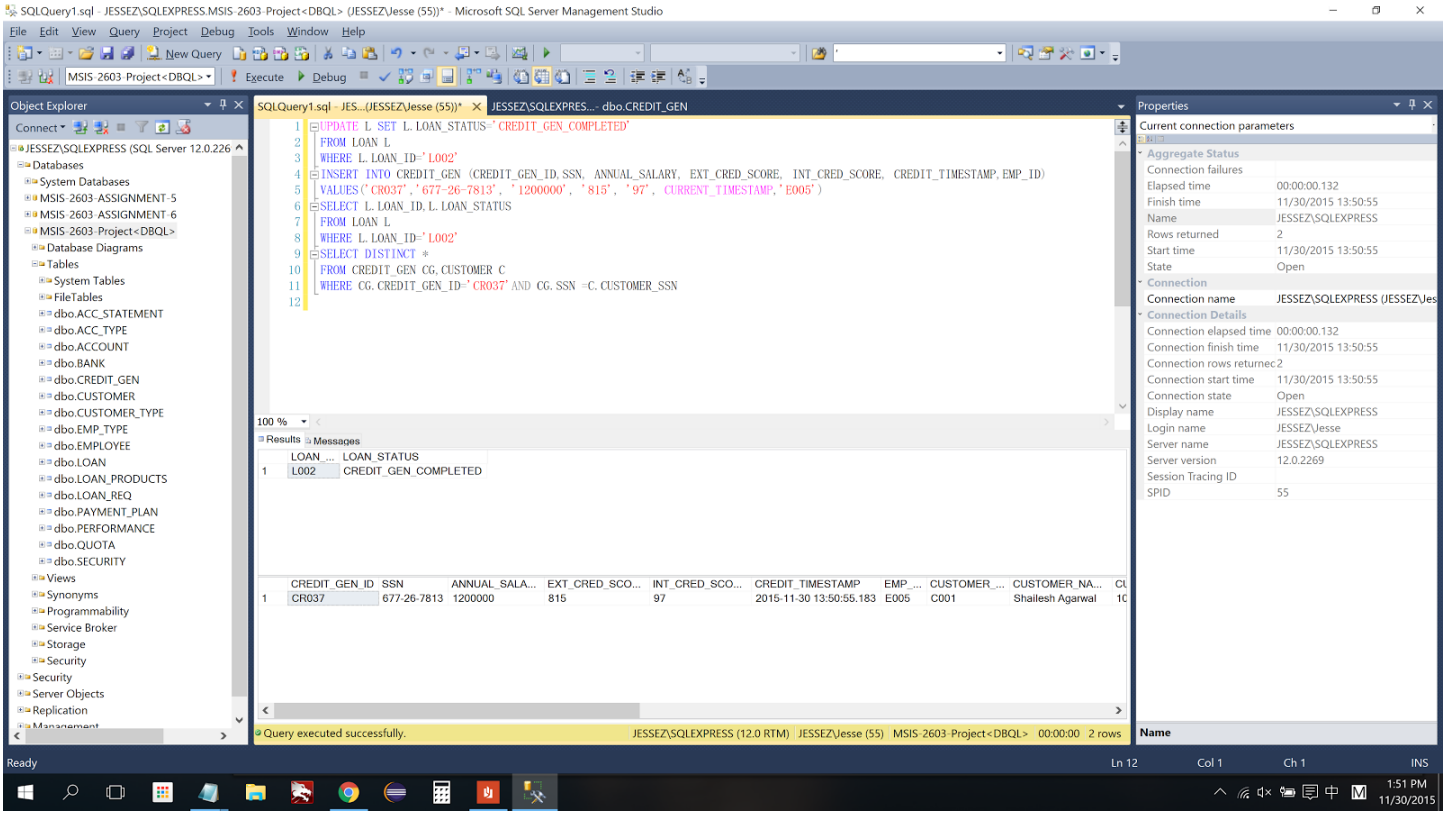
b. Search customers' internal credit score is equal or greater of 85

```
SELECT DISTINCT C.CUSTOMER_ID,C.CUSTOMER_NAME,CG.INT_CRED_SCORE
FROM CREDIT_GEN CG, CUSTOMER C
WHERE CG.SSN=C.CUSTOMER_SSN AND CG.INT_CRED_SCORE>=85
ORDER BY CG.INT_CRED_SCORE
```



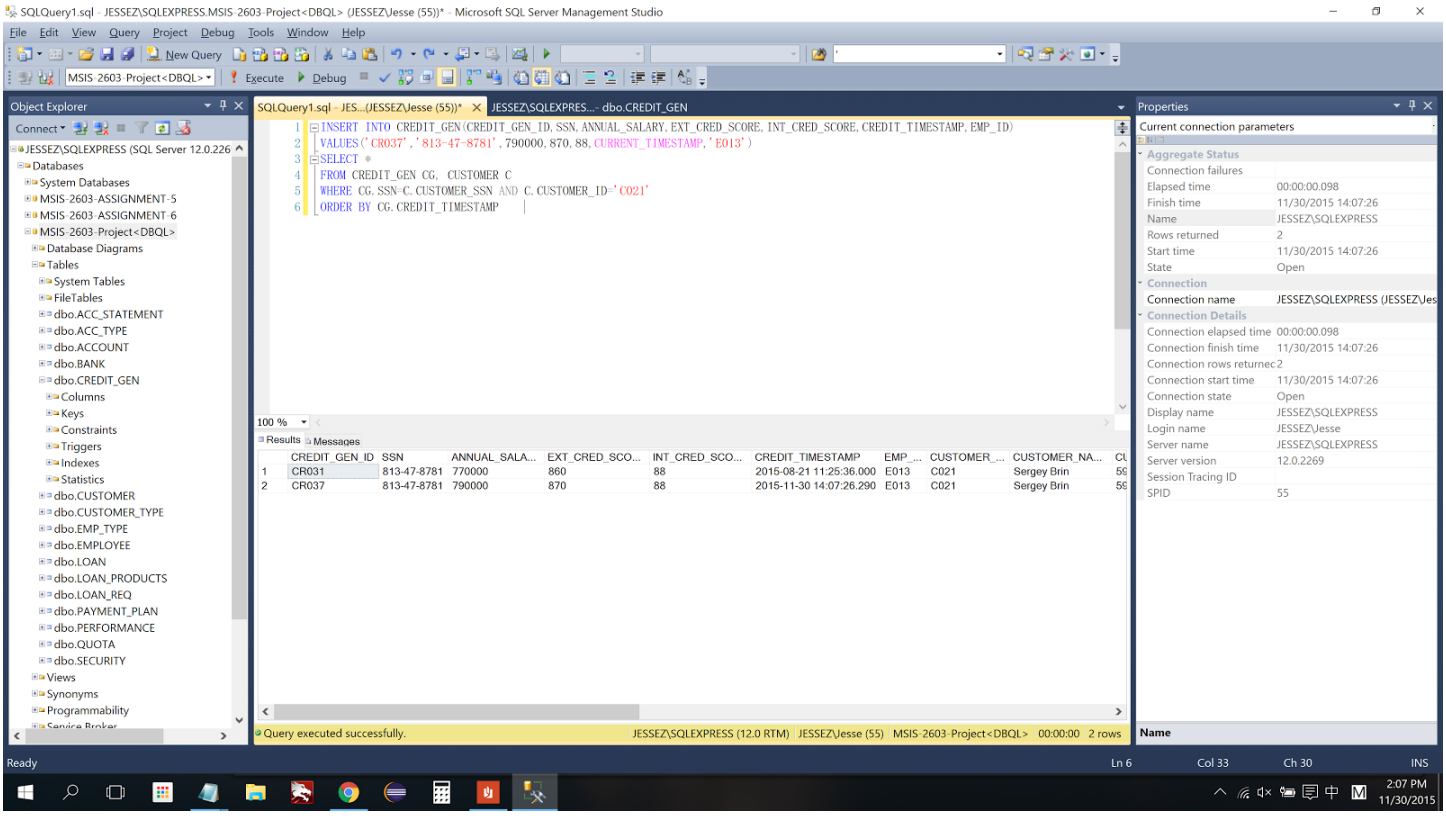
c. Change customer C001 L002 status to credit GEN completed. also insert his credit score into credit Gen table.

```
UPDATE L SET L.LOAN_STATUS='CREDIT_GEN_COMPLETED'
FROM LOAN L
WHERE L.LOAN_ID='L002'
INSERT INTO CREDIT_GEN (CREDIT_GEN_ID,SSN, ANNUAL_SALARY,
EXT_CRED_SCORE, INT_CRED_SCORE, CREDIT_TIMESTAMP,EMP_ID)
VALUES('CR037','677-26-7813', '1200000', '815', '97',
CURRENT_TIMESTAMP,'E005')
SELECT L.LOAN_ID,L.LOAN_STATUS
FROM LOAN L
WHERE L.LOAN_ID='L002'
SELECT DISTINCT *
FROM CREDIT_GEN CG,CUSTOMER C
WHERE CG.CREDIT_GEN_ID='CR037'AND CG.SSN =C.CUSTOMER_SSN
```



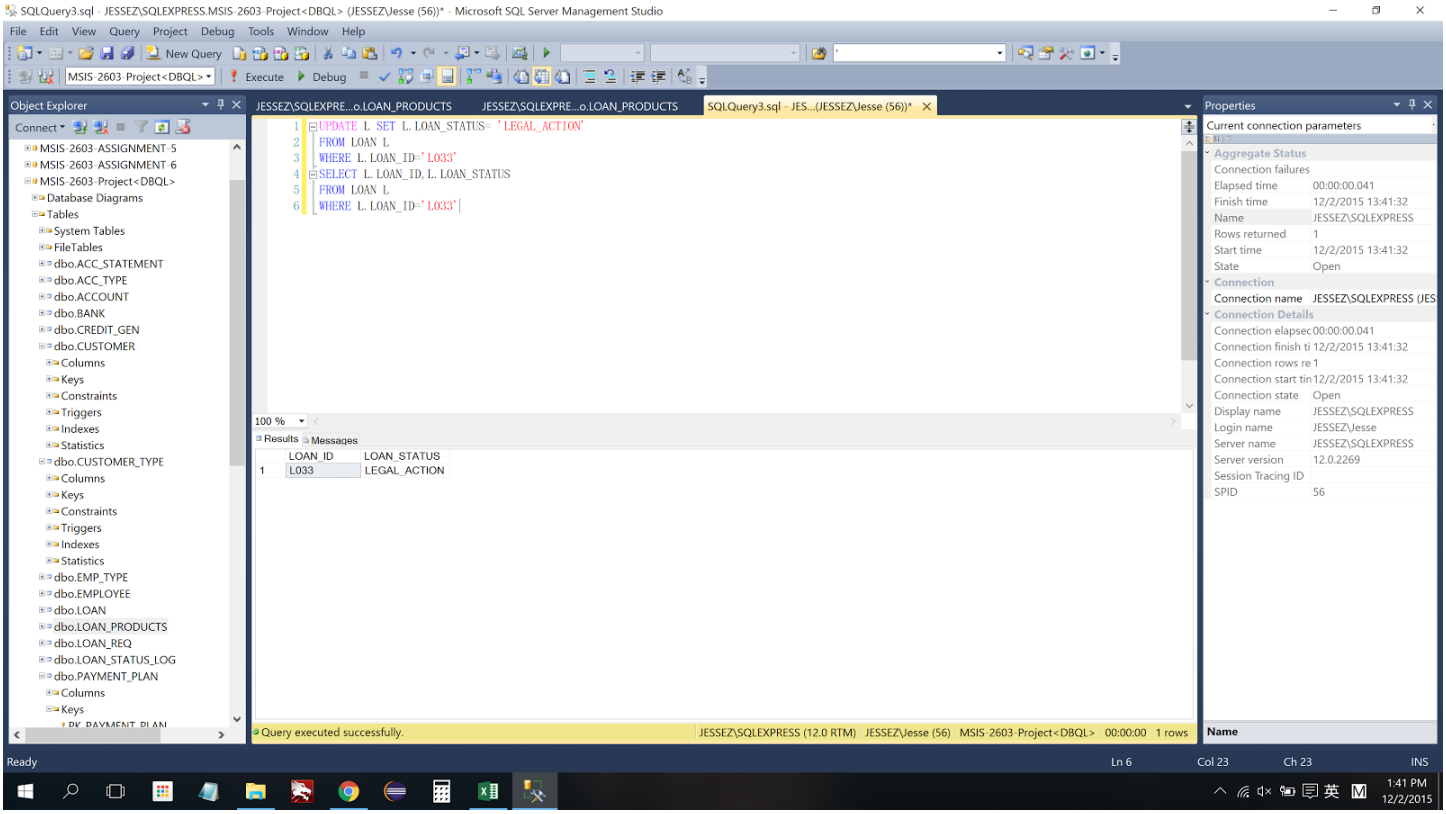
d. Updated client C021's external score to 870.

```
INSERT INTO
CREDIT_GEN(CREDIT_GEN_ID,SSN,ANNUAL_SALARY,EXT_CRED_SCORE,INT_CRED_SCO
RE,CREDIT_TIMESTAMP,EMP_ID)
VALUES('CR037','813-47-8781',790000,870,88,CURRENT_TIMESTAMP,'E013')
SELECT *
FROM CREDIT_GEN CG, CUSTOMER C
WHERE CG.SSN=C.CUSTOMER_SSN AND C.CUSTOMER_ID='C021'
ORDER BY CG.CREDIT_TIMESTAMP
```



4. **Collection Agent**
    a. Change loan L033 status to LEGAL ACTION

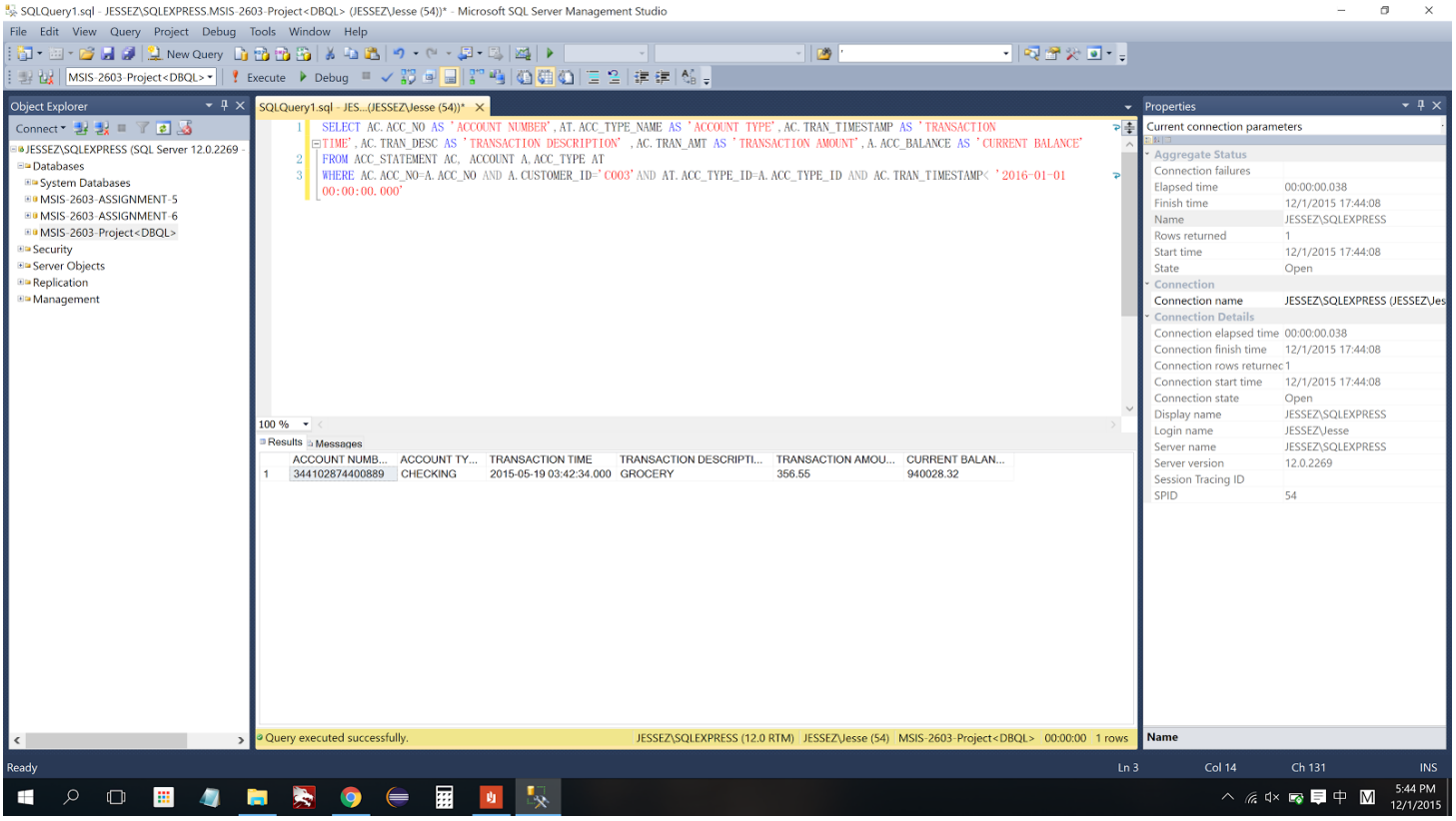
```
UPDATE L SET L.LOAN_STATUS= 'LEGAL_ACTION'
FROM LOAN L
WHERE L.LOAN_ID='L033'
SELECT L.LOAN_ID,L.LOAN_STATUS
FROM LOAN L
WHERE L.LOAN_ID='L033'
```



5. **Customer**

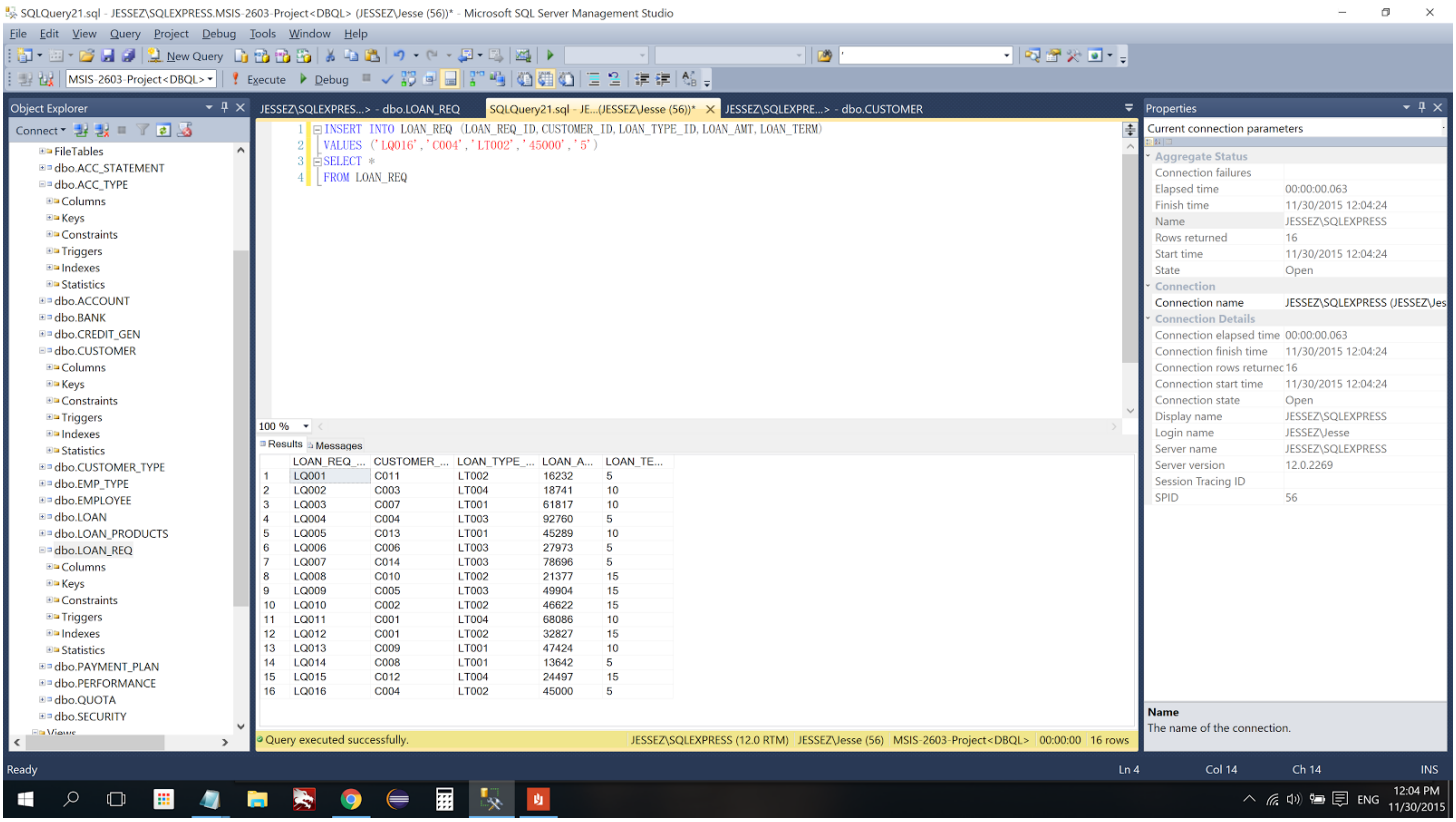
   a. Customer C003 checks his all transaction history in 2015.

```
SELECT AC.ACC_NO AS 'ACCOUNT NUMBER',AT.ACC_TYPE_NAME AS 'ACCOUNT
TYPE',AC.TRAN_TIMESTAMP AS 'TRANSACTION TIME',AC.TRAN_DESC AS
'TRANSACTION DESCRIPTION' ,AC.TRAN_AMT AS 'TRANSACTION
AMOUNT',A.ACC_BALANCE AS 'CURRENT BALANCE'
FROM ACC_STATEMENT AC, ACCOUNT A,ACC_TYPE AT
WHERE AC.ACC_NO=A.ACC_NO AND A.CUSTOMER_ID='C003'AND
AT.ACC_TYPE_ID=A.ACC_TYPE_ID AND AC.TRAN_TIMESTAMP< '2016-01-01
00:00:00.000'
```



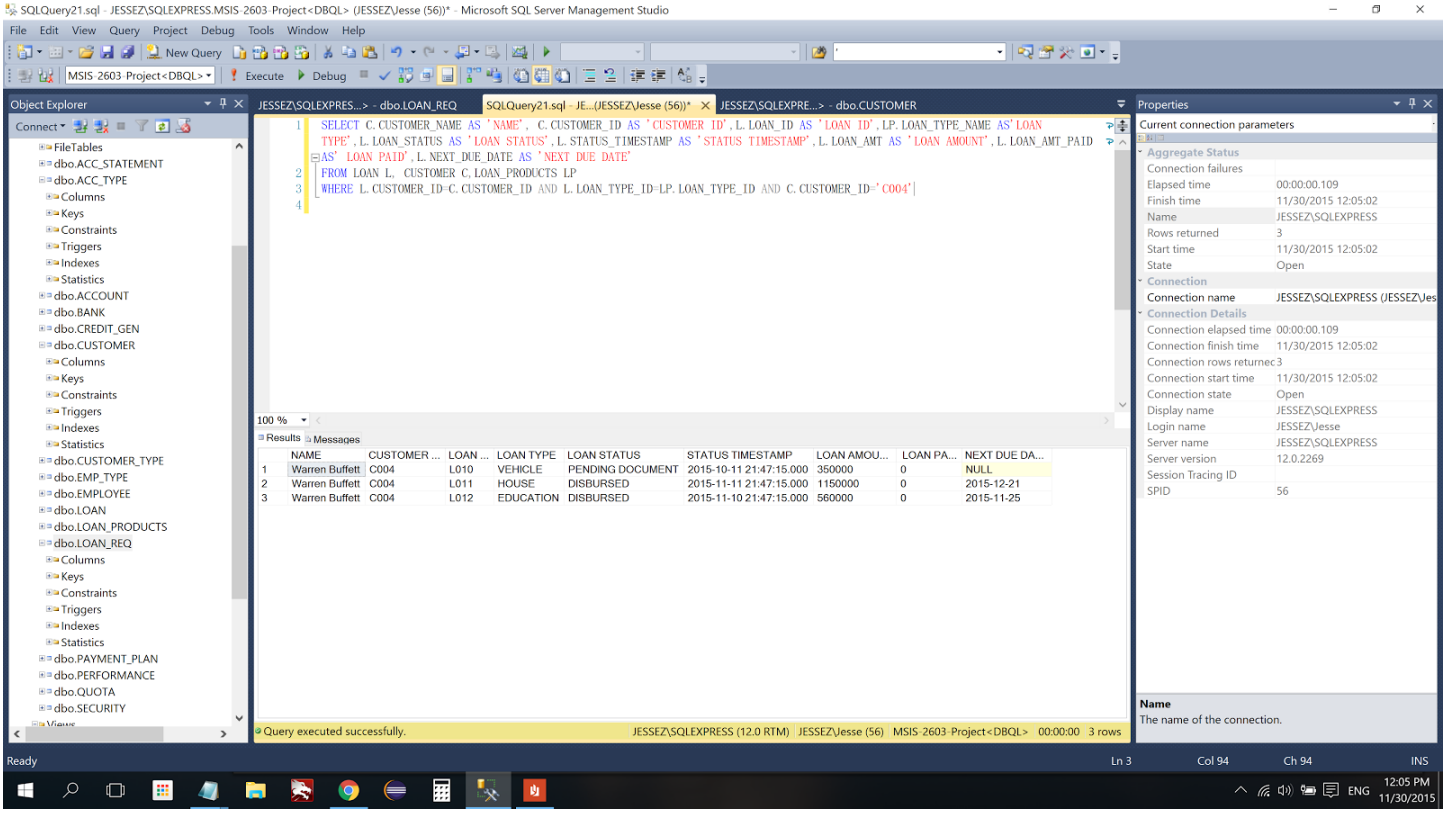
b. Customer C004 request a loan quote online.

```
INSERT INTO LOAN_REQ
(LOAN_REQ_ID,CUSTOMER_ID,LOAN_TYPE_ID,LOAN_AMT,LOAN_TERM)
VALUES ('LQ016','C004','LT002','45000','5')
SELECT *
FROM LOAN_REQ
```



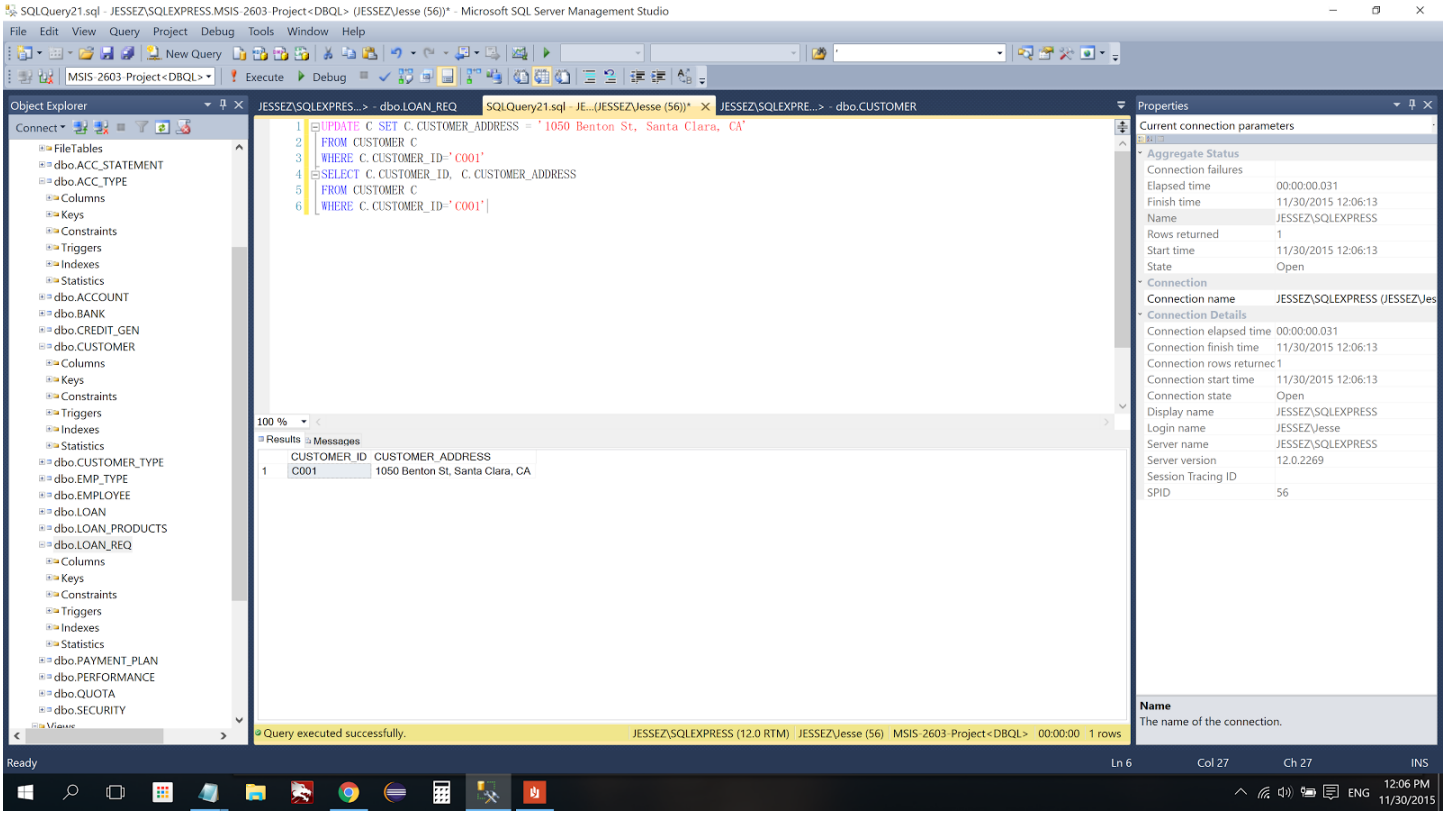
c. Customer C004 check his loans and status.

```
SELECT C.CUSTOMER_NAME AS 'NAME', C.CUSTOMER_ID AS 'CUSTOMER
ID',L.LOAN_ID AS 'LOAN ID',LP.LOAN_TYPE_NAME AS'LOAN
TYPE',L.LOAN_STATUS AS 'LOAN STATUS',L.STATUS_TIMESTAMP AS 'STATUS
TIMESTAMP',L.LOAN_AMT AS 'LOAN AMOUNT',L.LOAN_AMT_PAID AS' LOAN
PAID',L.NEXT_DUE_DATE AS 'NEXT DUE DATE'
FROM LOAN L, CUSTOMER C,LOAN_PRODUCTS LP
WHERE L.CUSTOMER_ID=C.CUSTOMER_ID AND L.LOAN_TYPE_ID=LP.LOAN_TYPE_ID
AND C.CUSTOMER_ID='C004'
```



d. Customer C001 update his home address to ‘1050 Benton St, Santa Clara, CA.

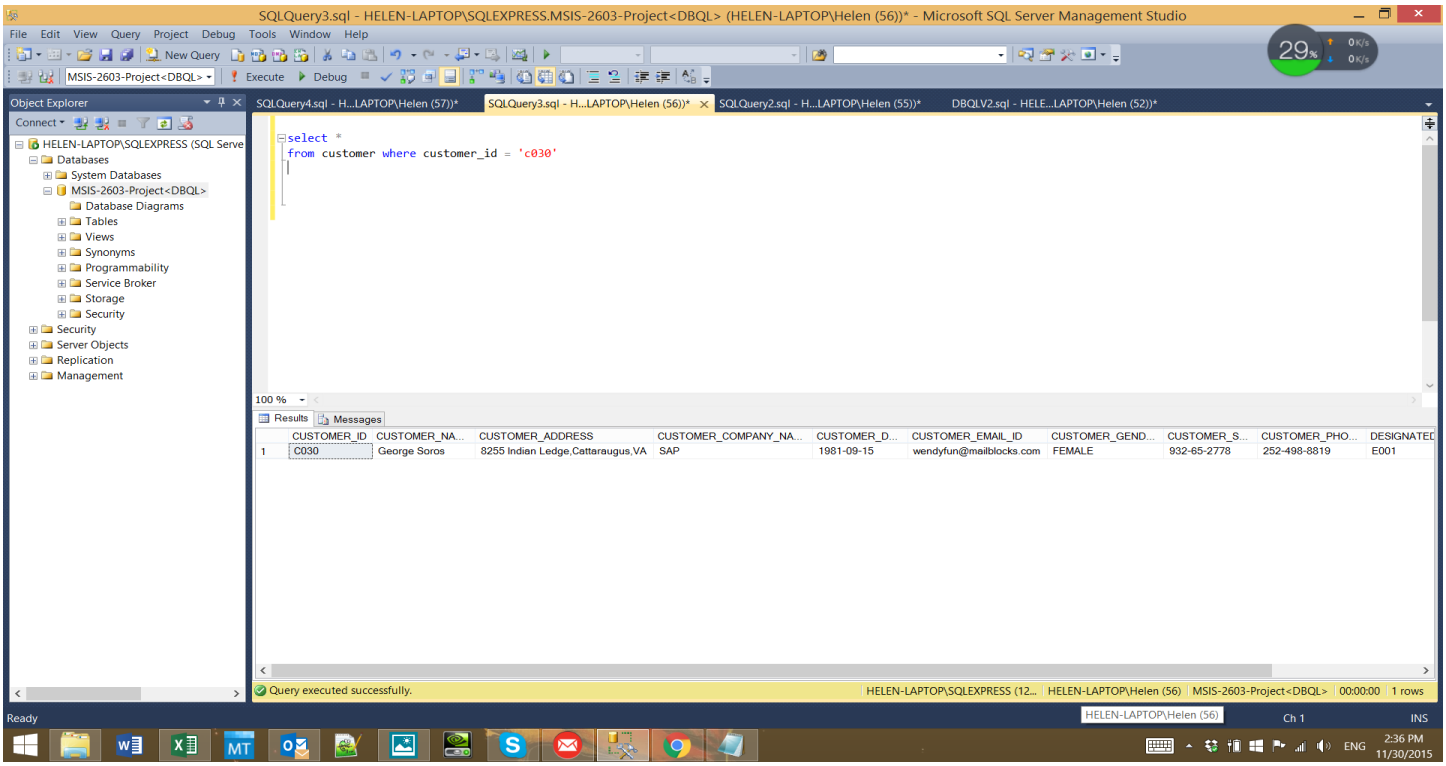
```
UPDATE C SET C.CUSTOMER_ADDRESS = '1050 Benton St, Santa Clara, CA'
FROM CUSTOMER C
WHERE C.CUSTOMER_ID='C001'
SELECT C.CUSTOMER_ID, C.CUSTOMER_ADDRESS
FROM CUSTOMER C
WHERE C.CUSTOMER_ID='C001'
```

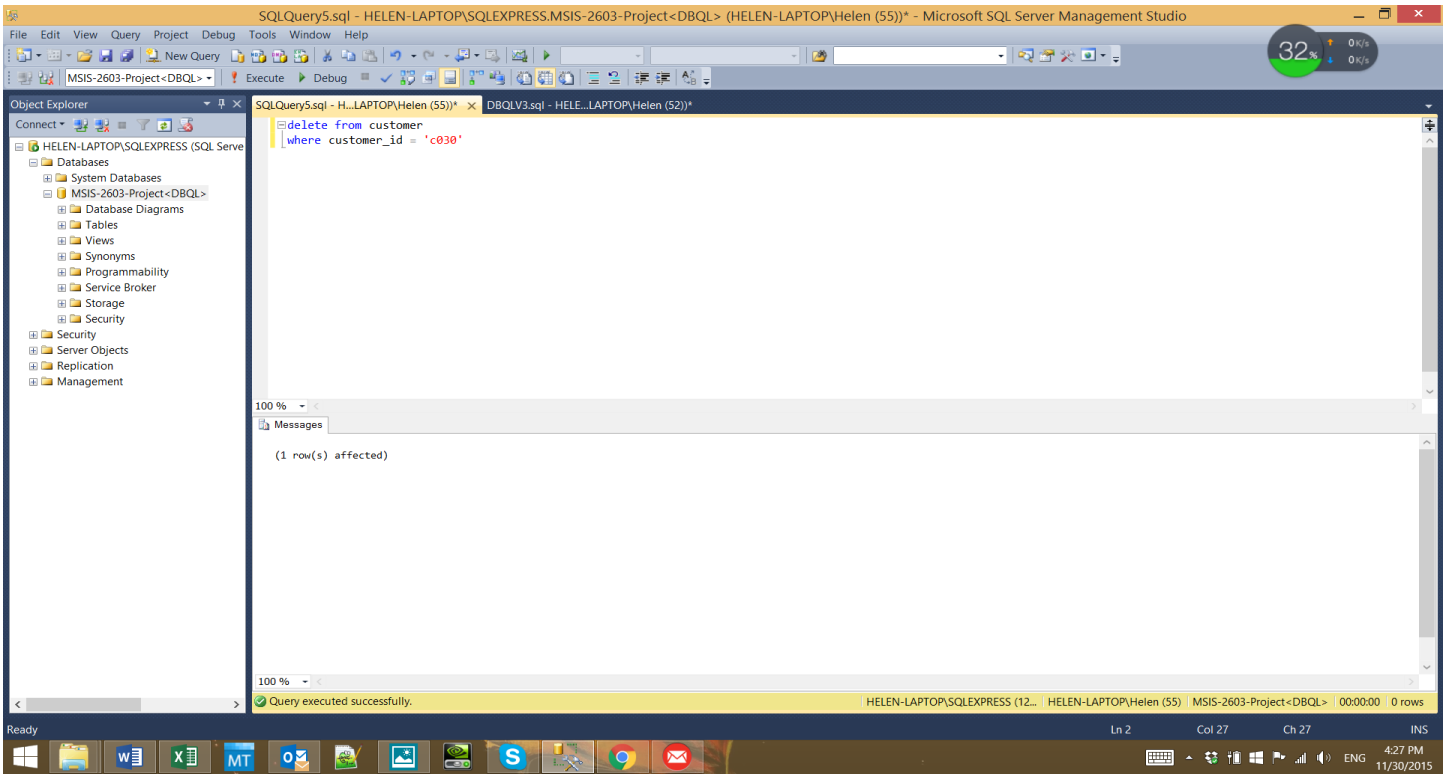


Delete Queries (Before and After)

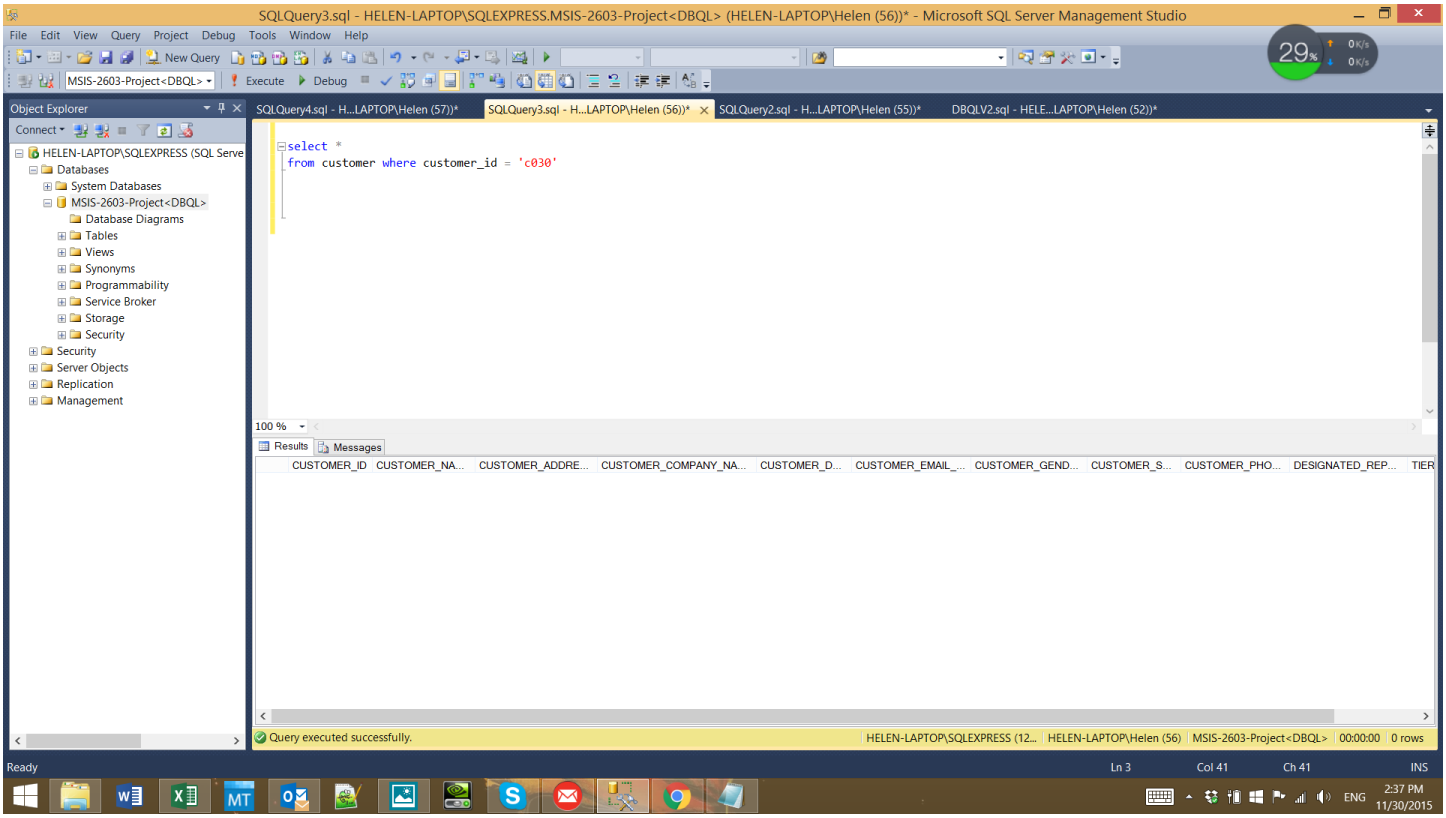
The Update queries can be found in the Use Case section

1. **CUSTOMER**
    a. Before



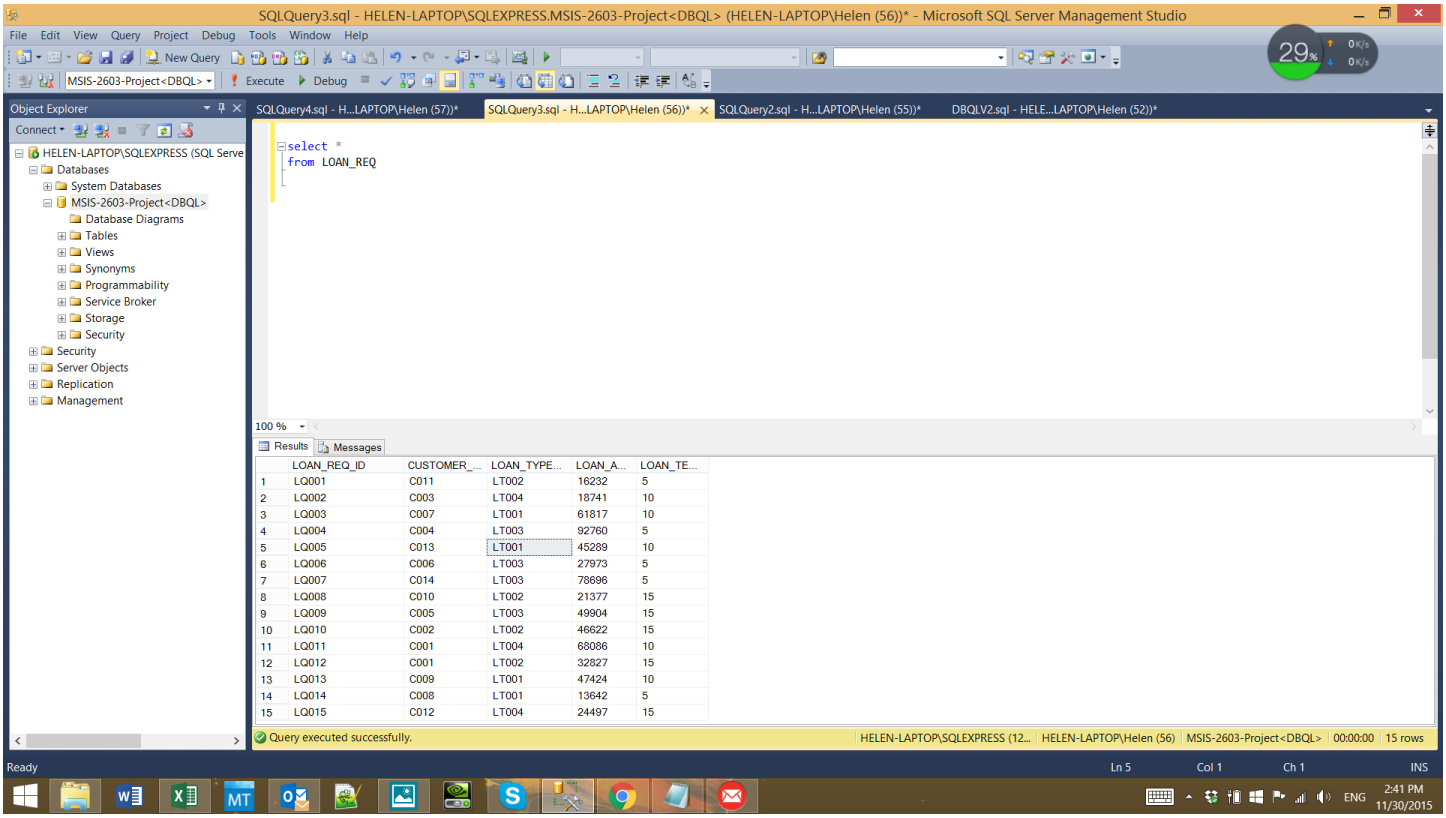


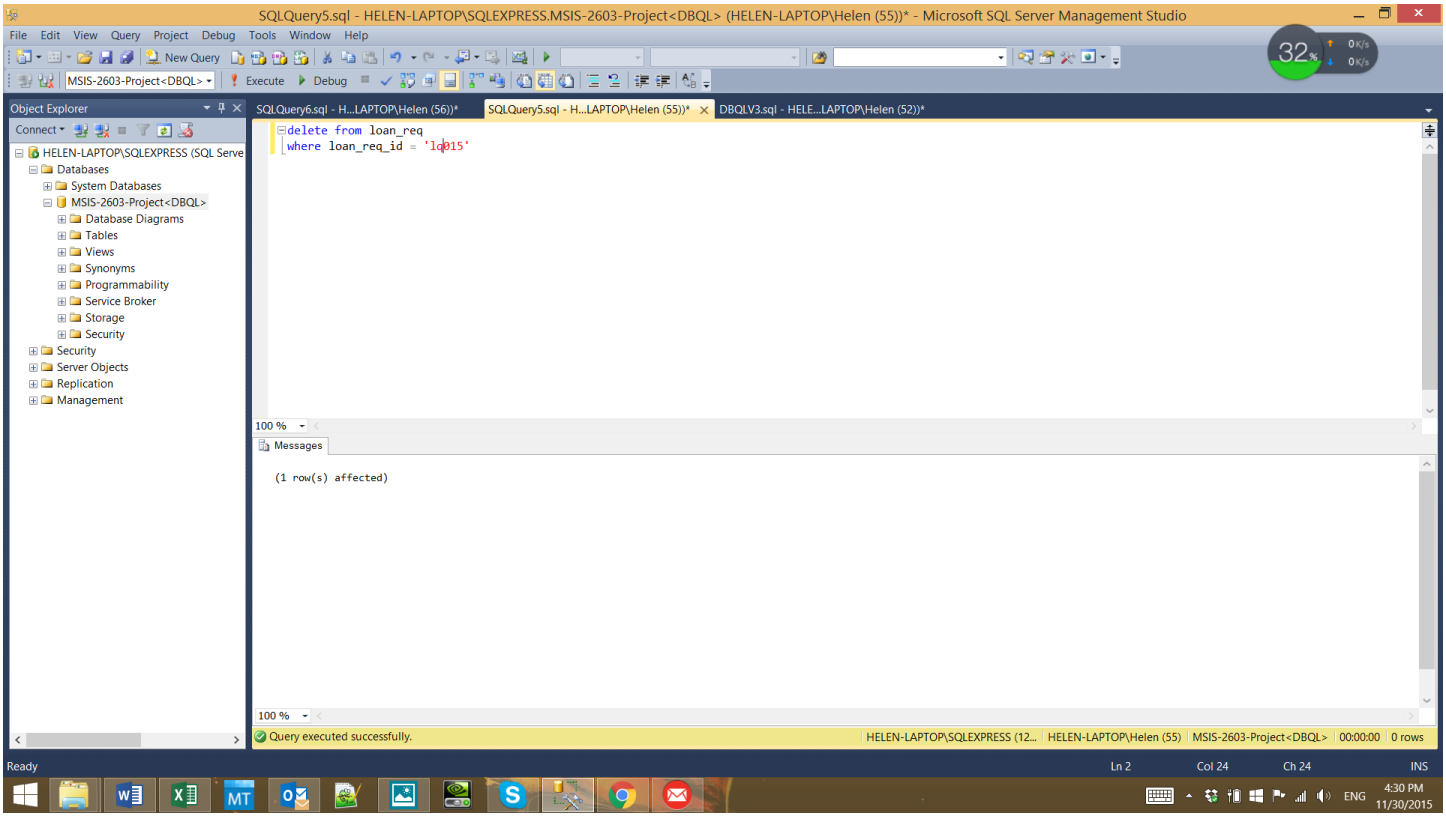
b. After



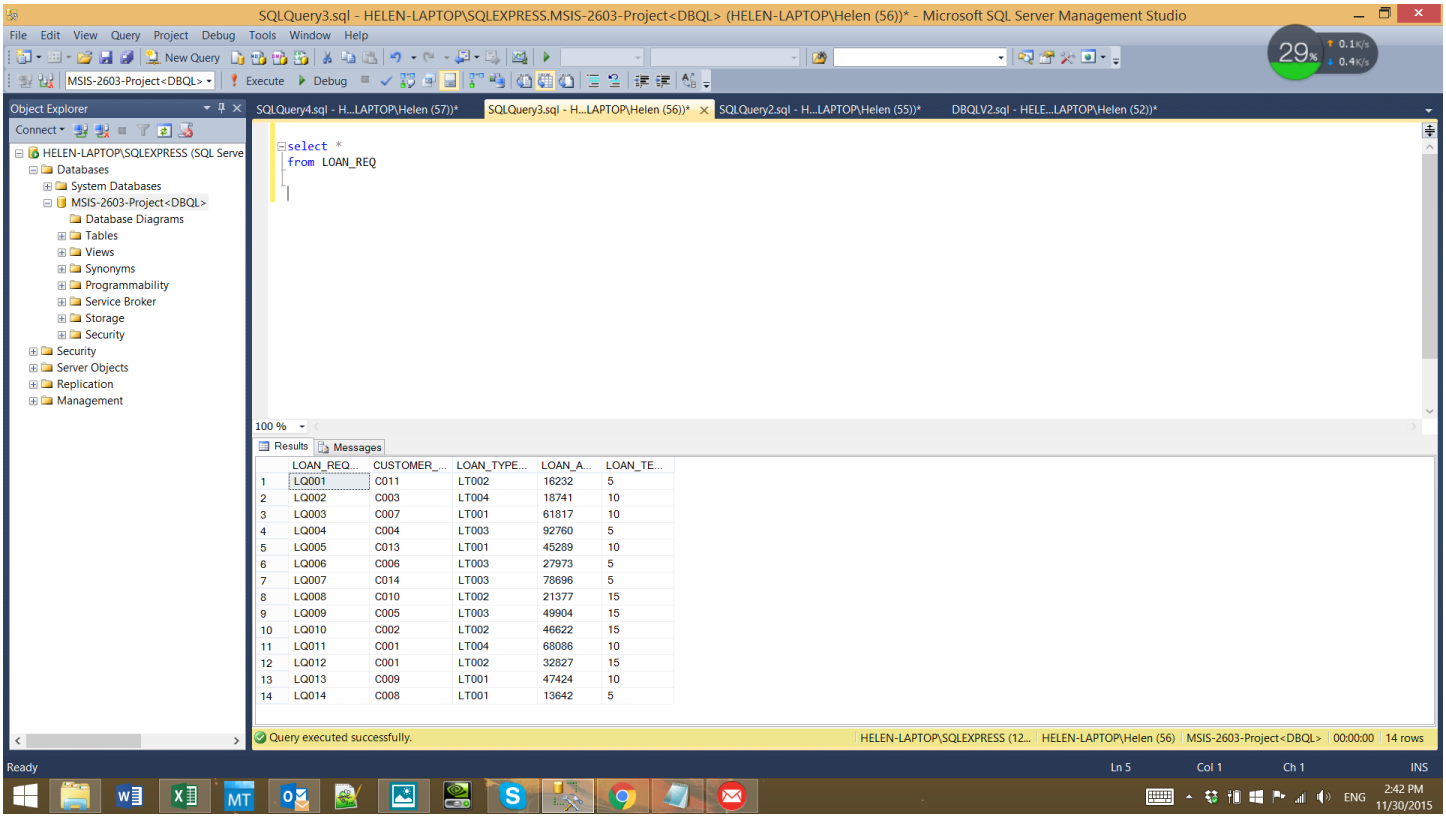
2. LOAN_REQ

a. Before

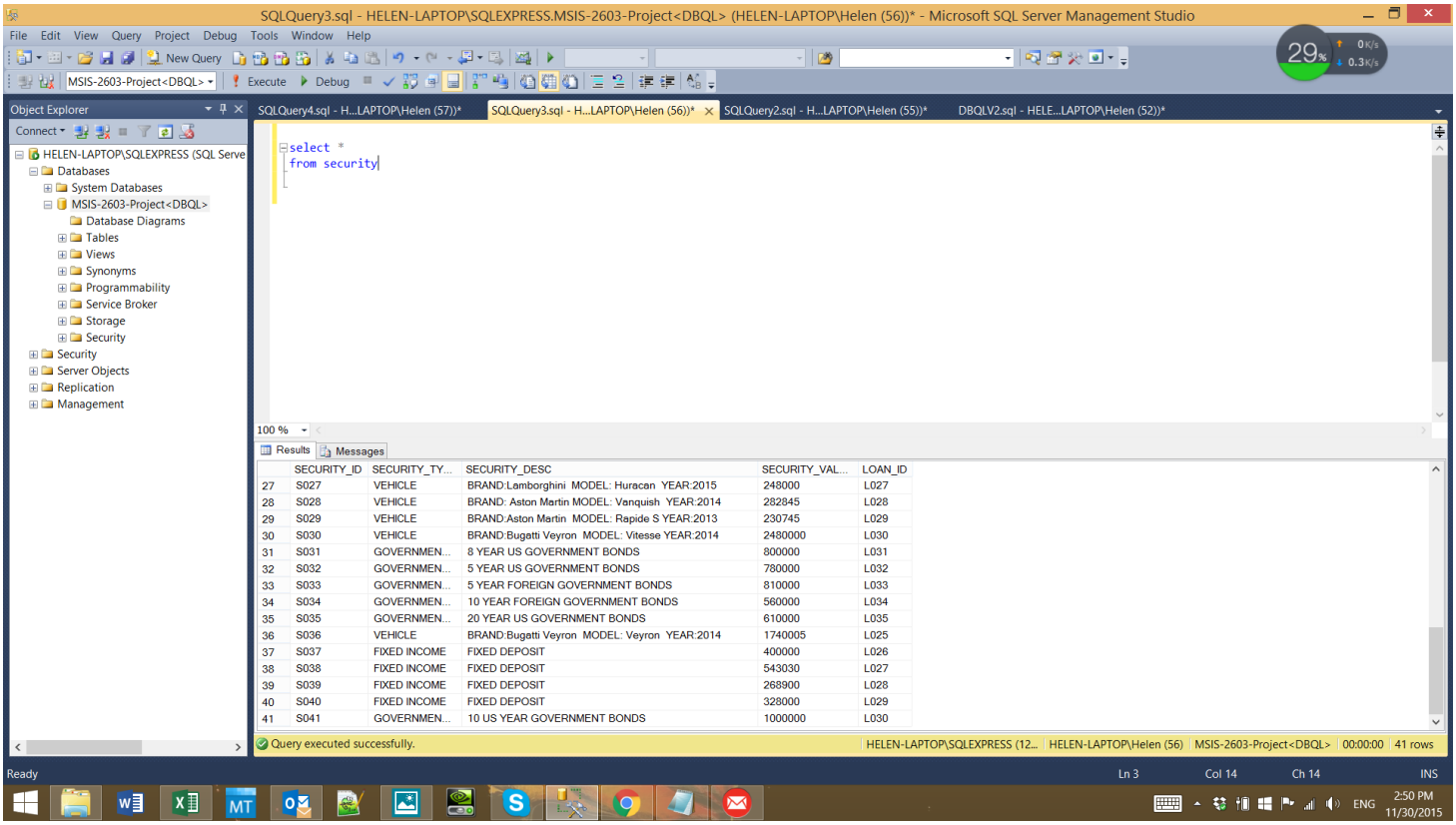


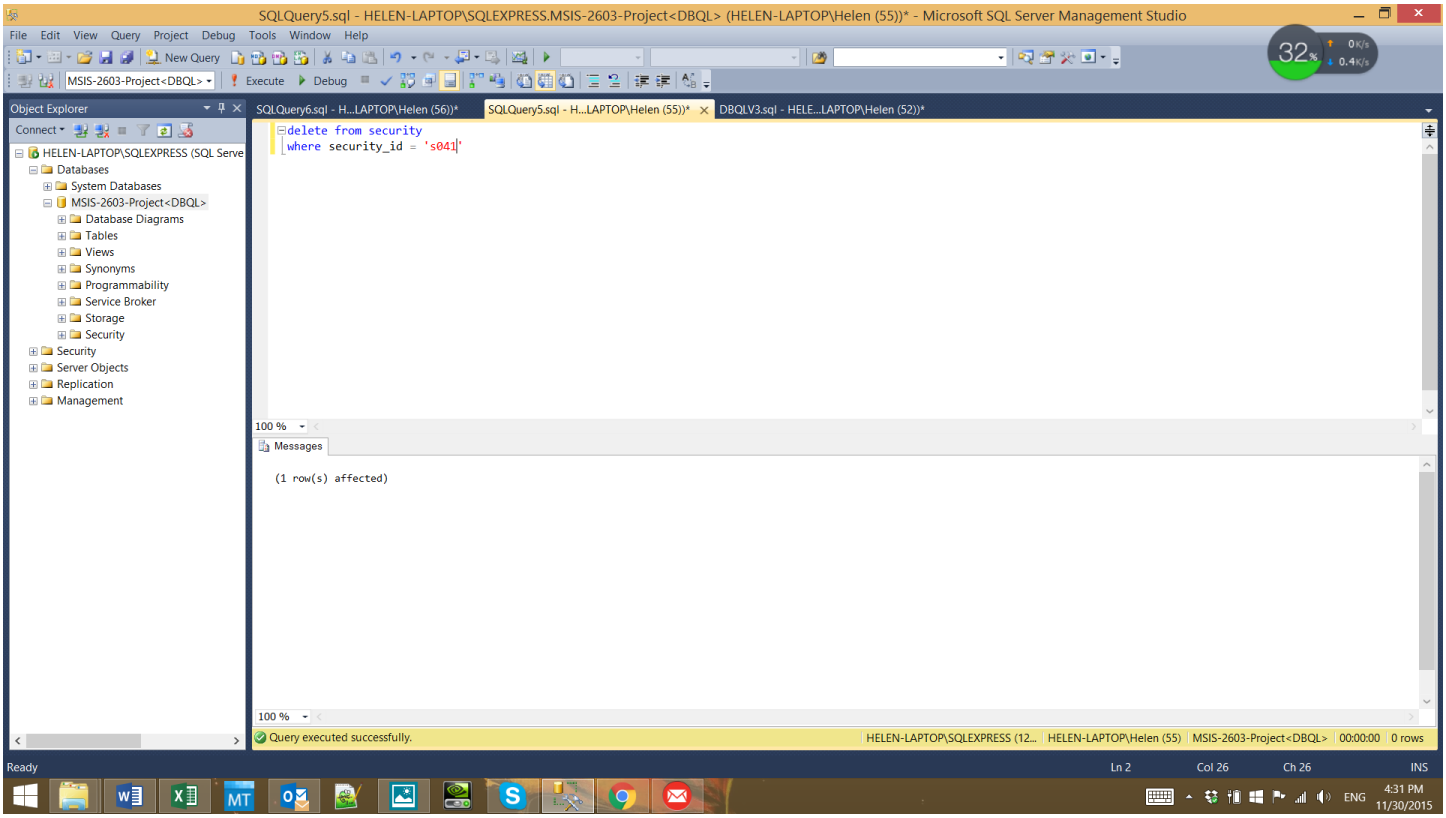

SQLQuery5.sql - HELEN-LAPTOP\SQLEXPRESS.MSIS-2603-Project<DBQL> (HELEN-LAPTOP\Helen (55))* - Microsoft SQL Server Management Studio
File Edit View Query Project Debug Tools Window Help
New Query
Execute Debug
MSIS-2603-Project<DBQL>
Object Explorer
Connect
HELEN-LAPTOP\SQLEXPRESS (SQL Serve
Databases
System Databases
MSIS-2603-Project<DBQL>
Database Diagrams
Tables
Views
Synonyms
Programmability
Service Broker
Storage
Security
Security
Server Objects
Replication
Management
SQLQuery6.sql - H...LAPTOP\Helen (56))*
SQLQuery5.sql - H...LAPTOP\Helen (55))*
DBQLV3.sql - HELE...LAPTOP\Helen (52))*
delete from loan_req
where loan_req_id = 'lq015'
100 %
Messages
(1 row(s) affected)
100 %
Query executed successfully.
HELEN-LAPTOP\SQLEXPRESS (12...
HELEN-LAPTOP\Helen (55)
MSIS-2603-Project<DBQL>
00:00:00
0 rows
Ready
Ln 2
Col 24
Ch 24
INS
32%
0 K/s
0 K/s
ENG
4:30 PM
11/30/2015

b. After

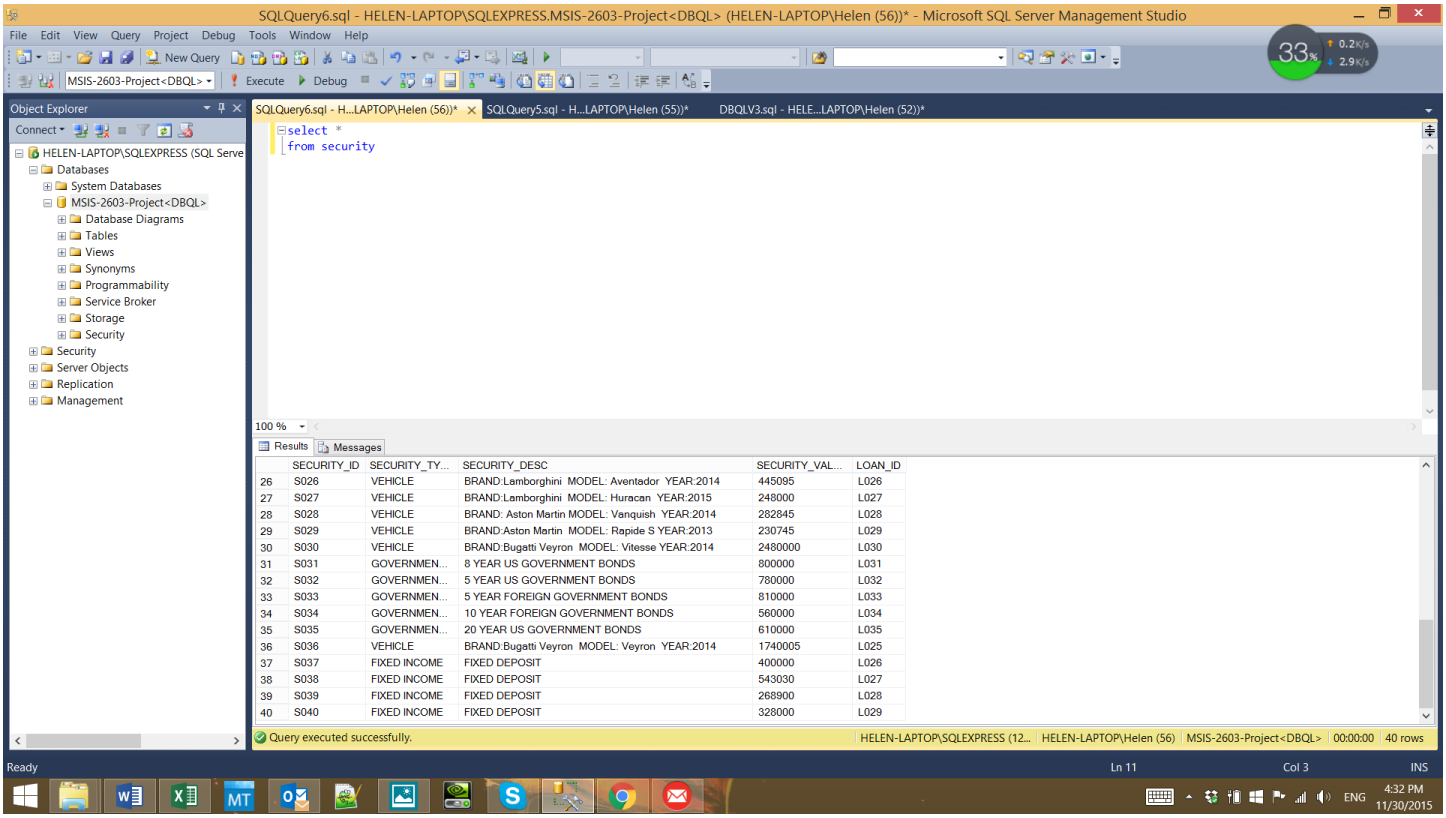

SQLQuery3.sql - HELEN-LAPTOP\SQLEXPRESS.MSIS-2603-Project<DBQL> (HELEN-LAPTOP\Helen (56))* - Microsoft SQL Server Management Studio
File Edit View Query Project Debug Tools Window Help
New Query
Execute Debug
MSIS-2603-Project<DBQL>
Object Explorer
Connect
HELEN-LAPTOP\SQLEXPRESS (SQL Serve
Databases
System Databases
MSIS-2603-Project<DBQL>
Database Diagrams
Tables
Views
Synonyms
Programmability
Service Broker
Storage
Security
Security
Server Objects
Replication
Management
SQLQuery4.sql - H...LAPTOP\Helen (57))*
SQLQuery3.sql - H...LAPTOP\Helen (56))*
SQLQuery2.sql - H...LAPTOP\Helen (55))*
DBQLV2.sql - HELE...LAPTOP\Helen (52))*
select *
from LOAN_REQ
100 %
Results
Messages
LOAN_REQ... CUSTOMER_... LOAN_TYPE... LOAN_A... LOAN_TE...
1 LQ001 C011 LT002 16232 5
2 LQ002 C003 LT004 18741 10
3 LQ003 C007 LT001 61817 10
4 LQ004 C004 LT003 92760 5
5 LQ005 C013 LT001 45289 10
6 LQ006 C006 LT003 27973 5
7 LQ007 C014 LT003 78696 5
8 LQ008 C010 LT002 21377 15
9 LQ009 C005 LT003 49904 15
10 LQ010 C002 LT002 46622 15
11 LQ011 C001 LT004 68086 10
12 LQ012 C001 LT002 32827 15
13 LQ013 C009 LT001 47424 10
14 LQ014 C008 LT001 13642 5
Query executed successfully.
HELEN-LAPTOP\SQLEXPRESS (12...
HELEN-LAPTOP\Helen (56)
MSIS-2603-Project<DBQL>
00:00:00
14 rows
Ready
Ln 5
Col 1
Ch 1
INS
29%
0.1 K/s
0.4 K/s
ENG
2:42 PM
11/30/2015

3. SECURITY
    a. Before



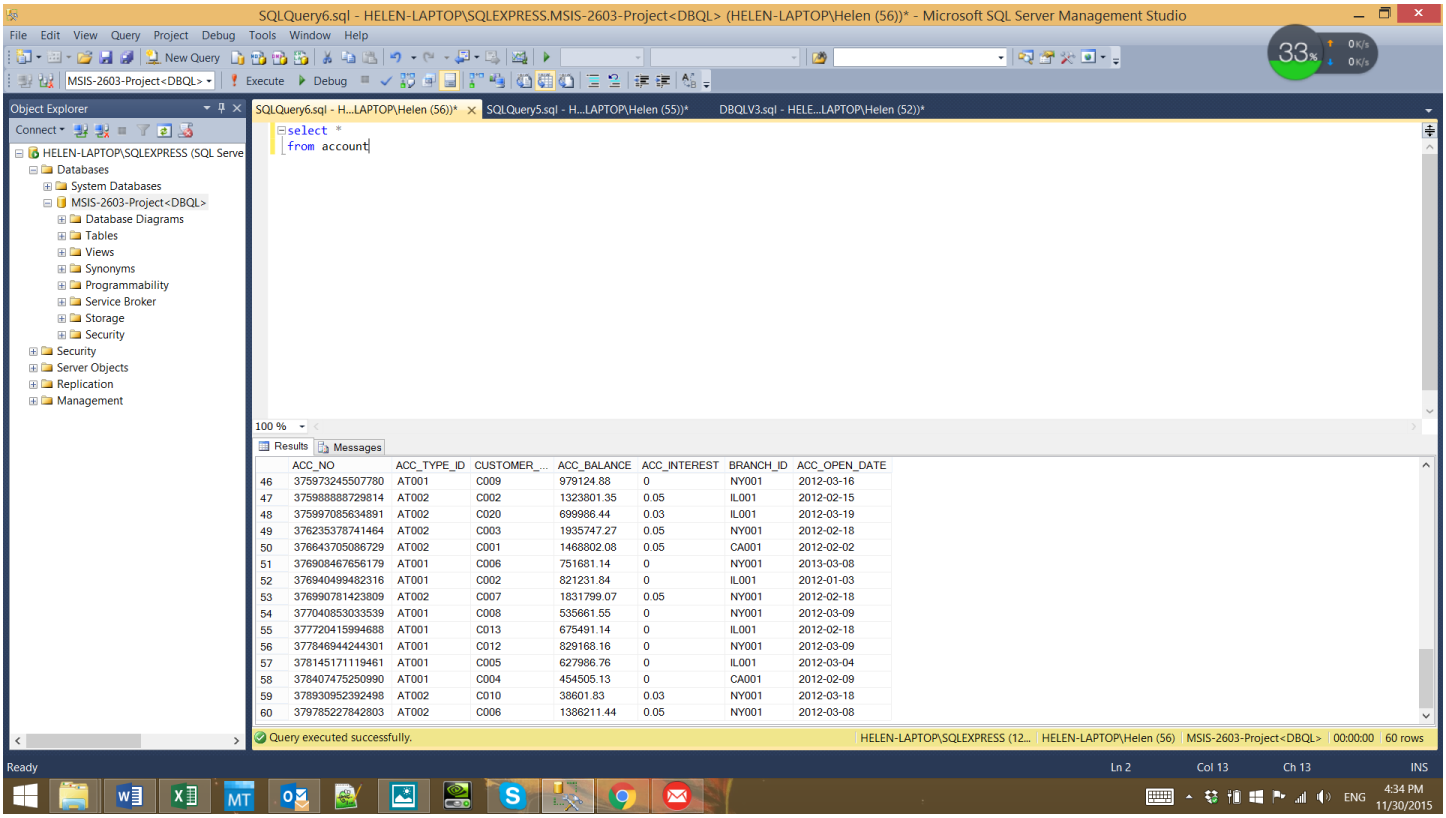


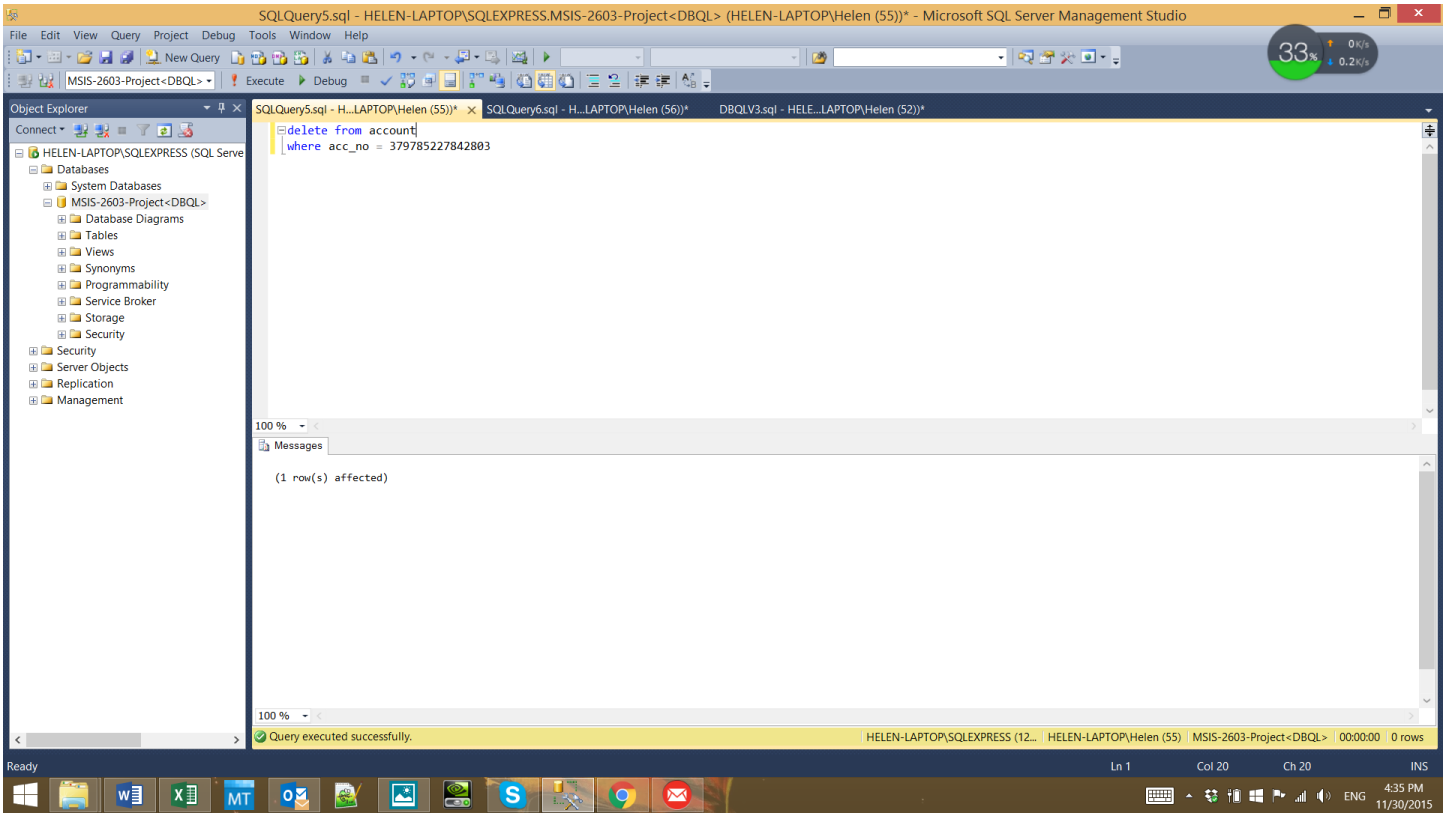
b. After



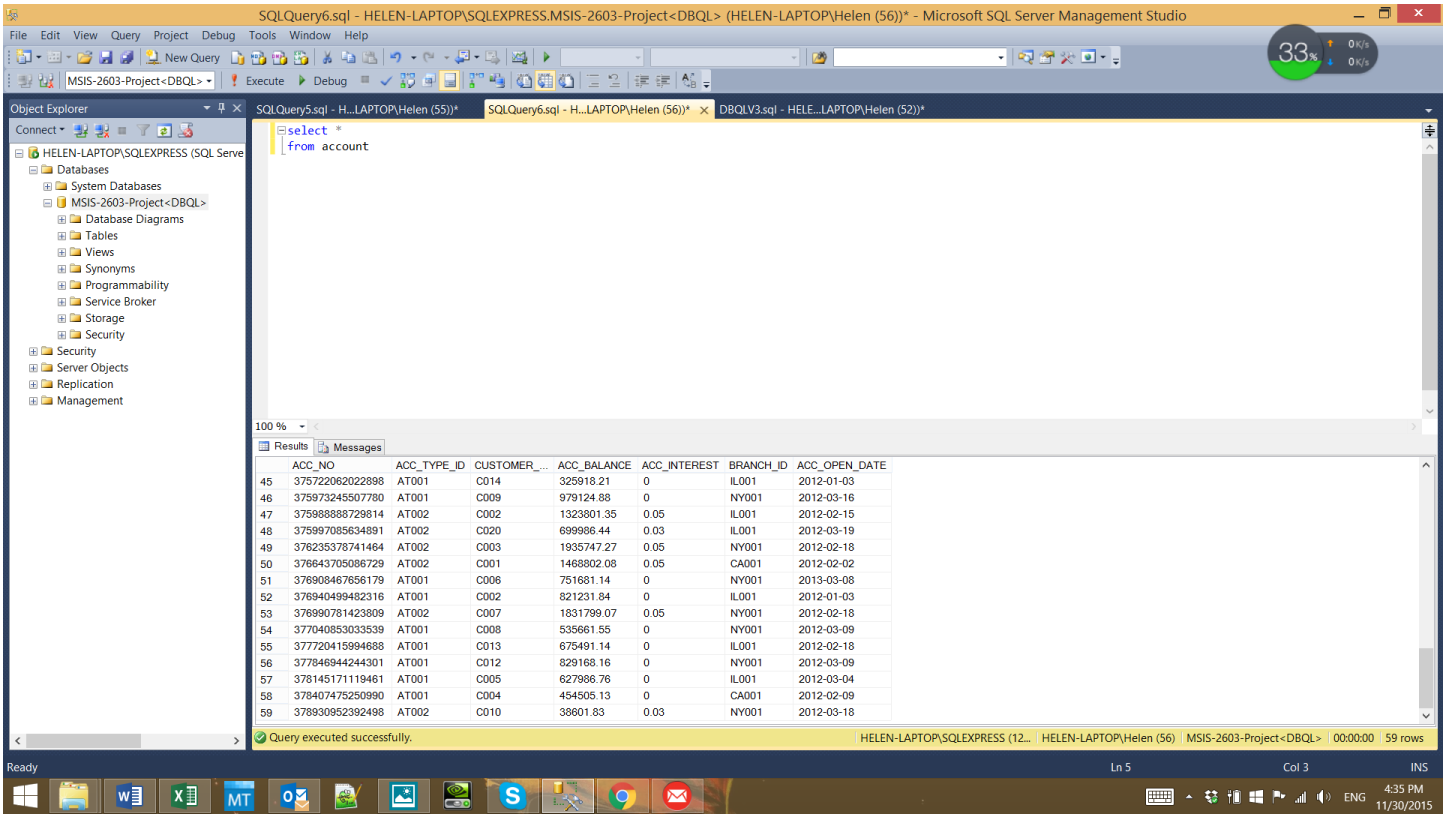
4. ACCOUNT

a. Before

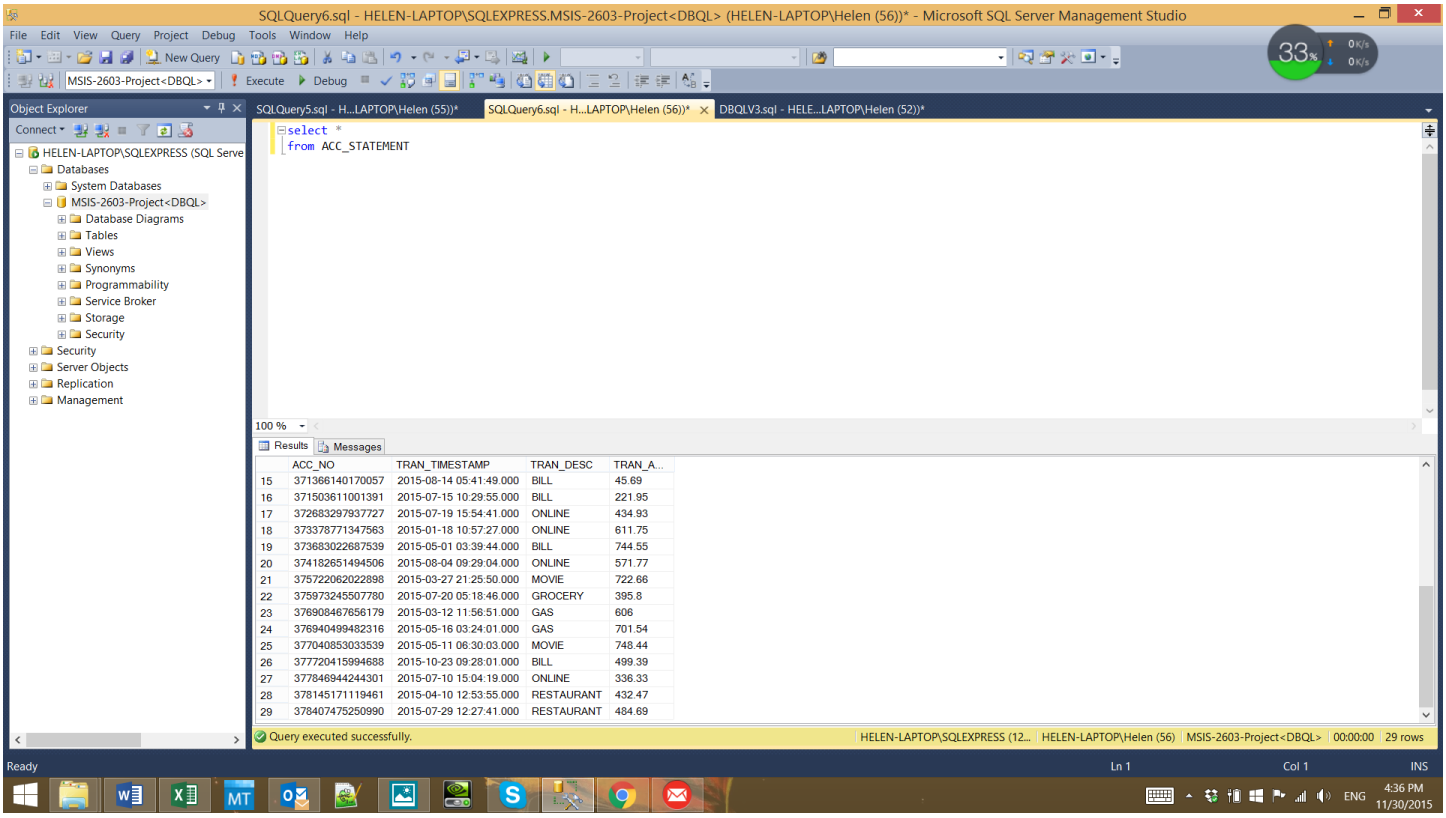


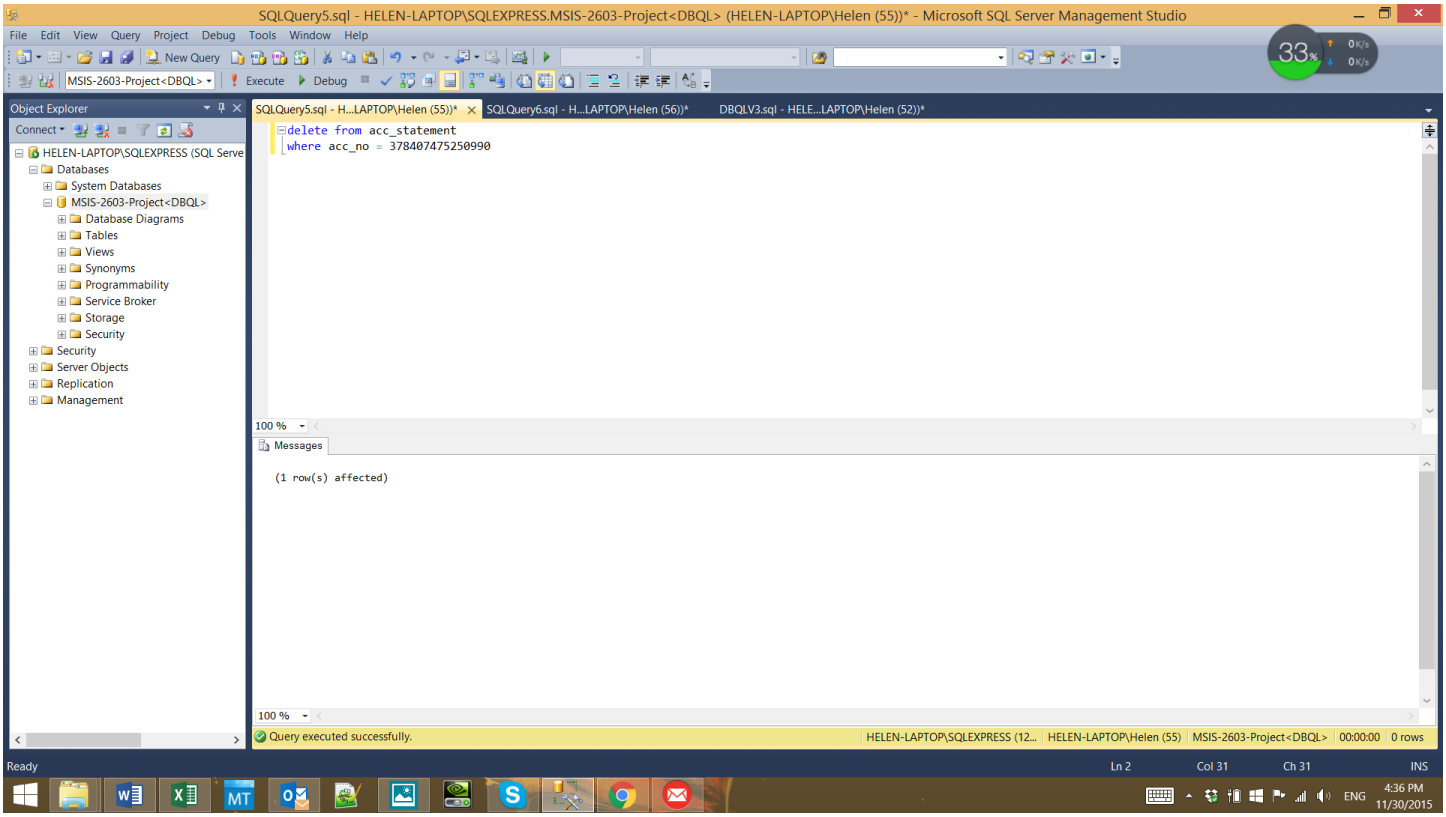

SQLQuery5.sql - HELEN-LAPTOP\SQLEXPRESS.MSIS-2603-Project<DBQL> (HELEN-LAPTOP\Helen (55))* - Microsoft SQL Server Management Studio
File Edit View Query Project Debug Tools Window Help
New Query
MSIS-2603-Project<DBQL>
Execute
Debug
Object Explorer
Connect
HELEN-LAPTOP\SQLEXPRESS (SQL Serve
Databases
System Databases
MSIS-2603-Project<DBQL>
Database Diagrams
Tables
Views
Synonyms
Programmability
Service Broker
Storage
Security
Security
Server Objects
Replication
Management
SQLQuery5.sql - H...LAPTOP\Helen (55))*
SQLQuery6.sql - H...LAPTOP\Helen (56))*
DBQLV3.sql - HELE...LAPTOP\Helen (52))*
delete from account
where acc_no = 379785227842803
100 %
Messages
(1 row(s) affected)
100 %
Query executed successfully.
HELEN-LAPTOP\SQLEXPRESS (12...
HELEN-LAPTOP\Helen (55)
MSIS-2603-Project<DBQL>
00:00:00
0 rows
Ready
Ln 1
Col 20
Ch 20
INS
ENG
4:35 PM
11/30/2015

b. After

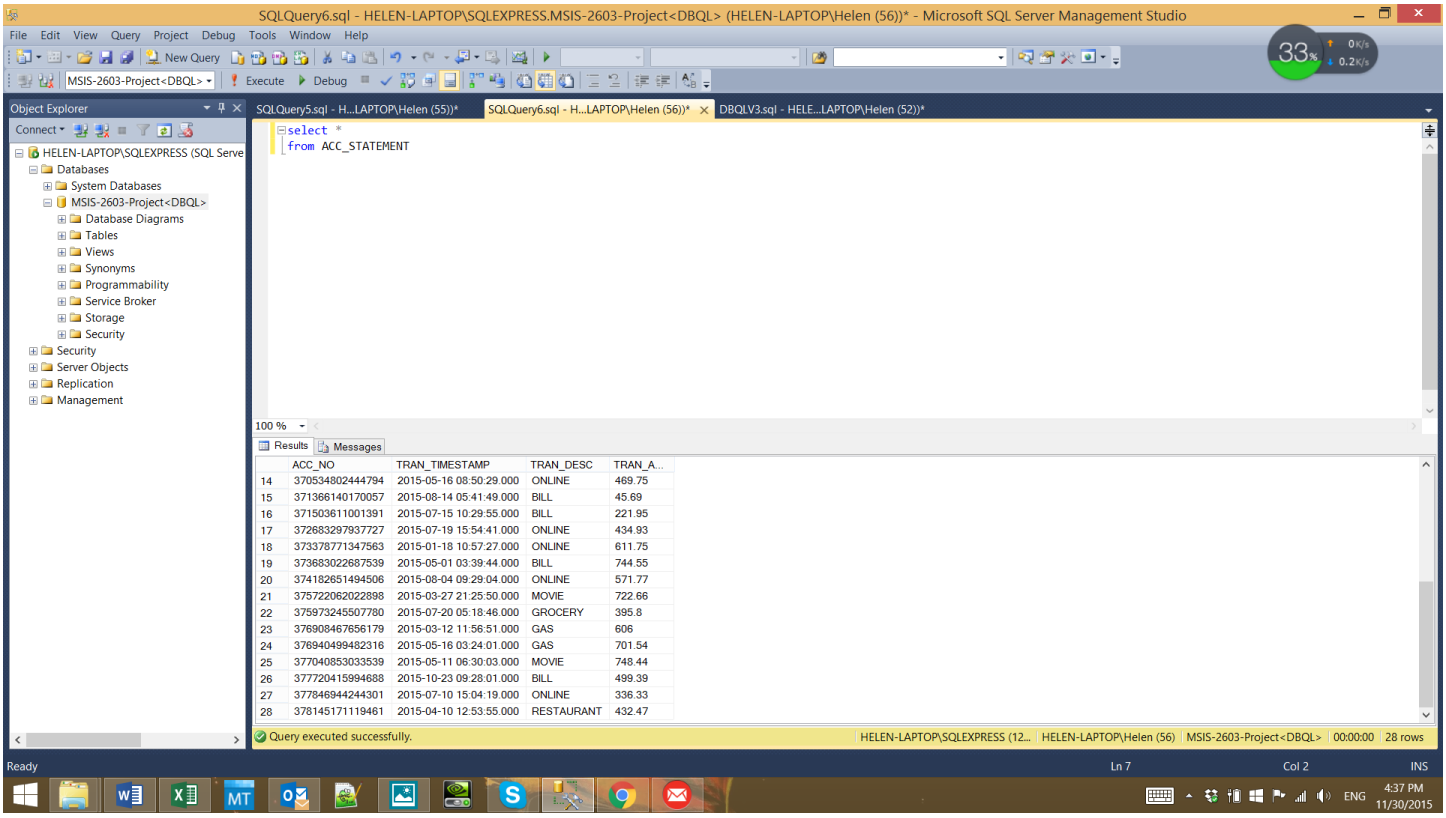

SQLQuery6.sql - HELEN-LAPTOP\SQLEXPRESS.MSIS-2603-Project<DBQL> (HELEN-LAPTOP\Helen (56))* - Microsoft SQL Server Management Studio
File Edit View Query Project Debug Tools Window Help
New Query
MSIS-2603-Project<DBQL>
Execute
Debug
Object Explorer
Connect
HELEN-LAPTOP\SQLEXPRESS (SQL Serve
Databases
System Databases
MSIS-2603-Project<DBQL>
Database Diagrams
Tables
Views
Synonyms
Programmability
Service Broker
Storage
Security
Security
Server Objects
Replication
Management
SQLQuery5.sql - H...LAPTOP\Helen (55))*
SQLQuery6.sql - H...LAPTOP\Helen (56))*
DBQLV3.sql - HELE...LAPTOP\Helen (52))*
select *
from account
100 %
Results
Messages
ACC_NO ACC_TYPE_ID CUSTOMER_... ACC_BALANCE ACC_INTEREST BRANCH_ID ACC_OPEN_DATE
45 375722062022898 AT001 C014 325918.21 0 IL001 2012-01-03
46 375973245507780 AT001 C009 979124.88 0 NY001 2012-03-16
47 375988888729814 AT002 C002 1323801.35 0.05 IL001 2012-02-15
48 375997085634891 AT002 C020 699986.44 0.03 IL001 2012-03-19
49 376235378741464 AT002 C003 1935747.27 0.05 NY001 2012-02-18
50 376643705086729 AT002 C001 1468802.08 0.05 CA001 2012-02-02
51 376908467656179 AT001 C006 751681.14 0 NY001 2013-03-08
52 376940499482316 AT001 C002 821231.84 0 IL001 2012-01-03
53 376990781423809 AT002 C007 1831799.07 0.05 NY001 2012-02-18
54 377040853033539 AT001 C008 535661.55 0 NY001 2012-03-09
55 377720415994688 AT001 C013 675491.14 0 IL001 2012-02-18
56 377846944244301 AT001 C012 829168.16 0 NY001 2012-03-09
57 378145171119461 AT001 C005 627986.76 0 IL001 2012-03-04
58 378407475250990 AT001 C004 454505.13 0 CA001 2012-02-09
59 378930952392498 AT002 C010 38601.83 0.03 NY001 2012-03-18
Query executed successfully.
HELEN-LAPTOP\SQLEXPRESS (12...
HELEN-LAPTOP\Helen (56)
MSIS-2603-Project<DBQL>
00:00:00
59 rows
Ready
Ln 5
Col 3
INS
ENG
4:35 PM
11/30/2015

5. ACC_STATEMENT
    a. Before

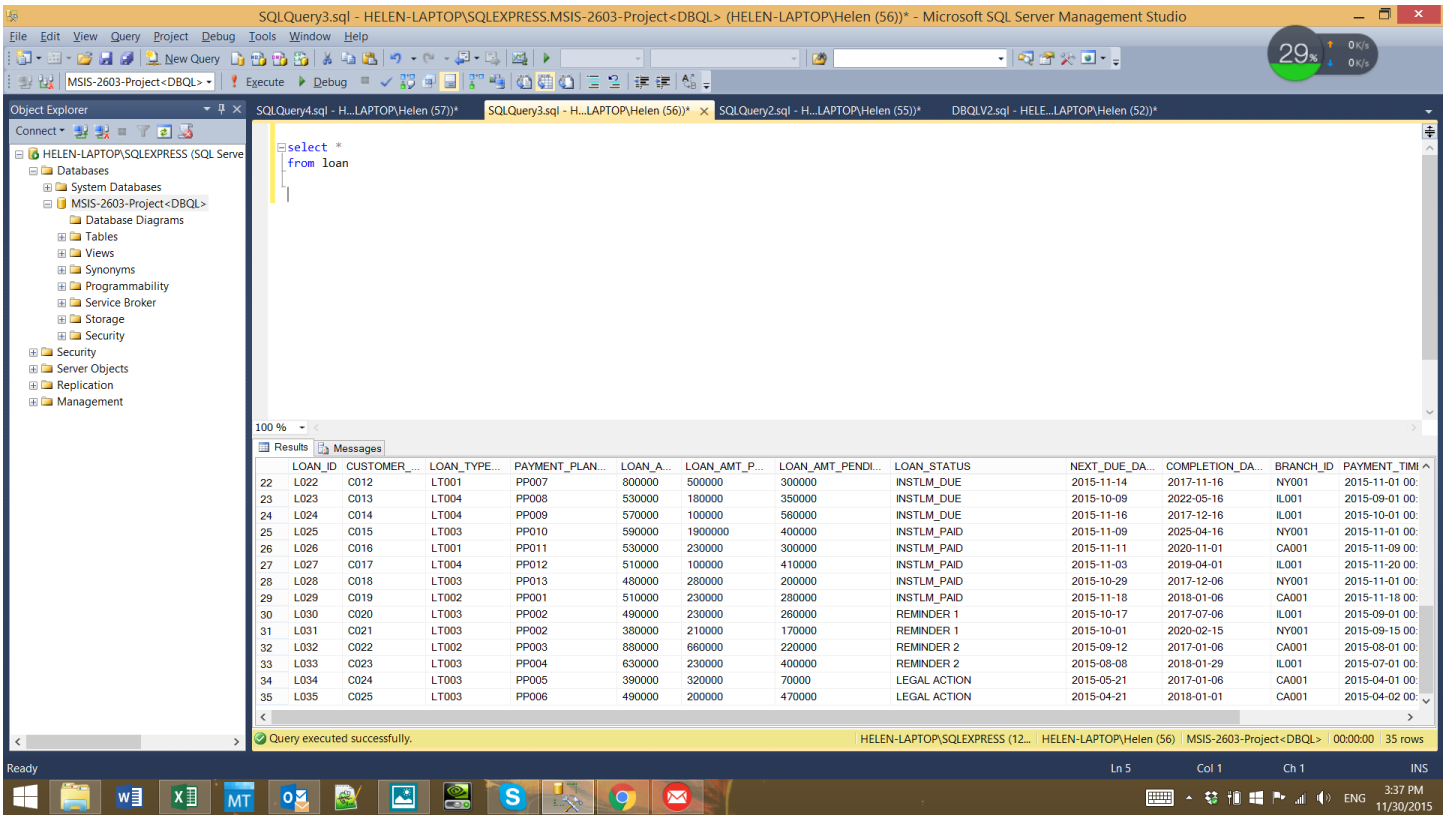


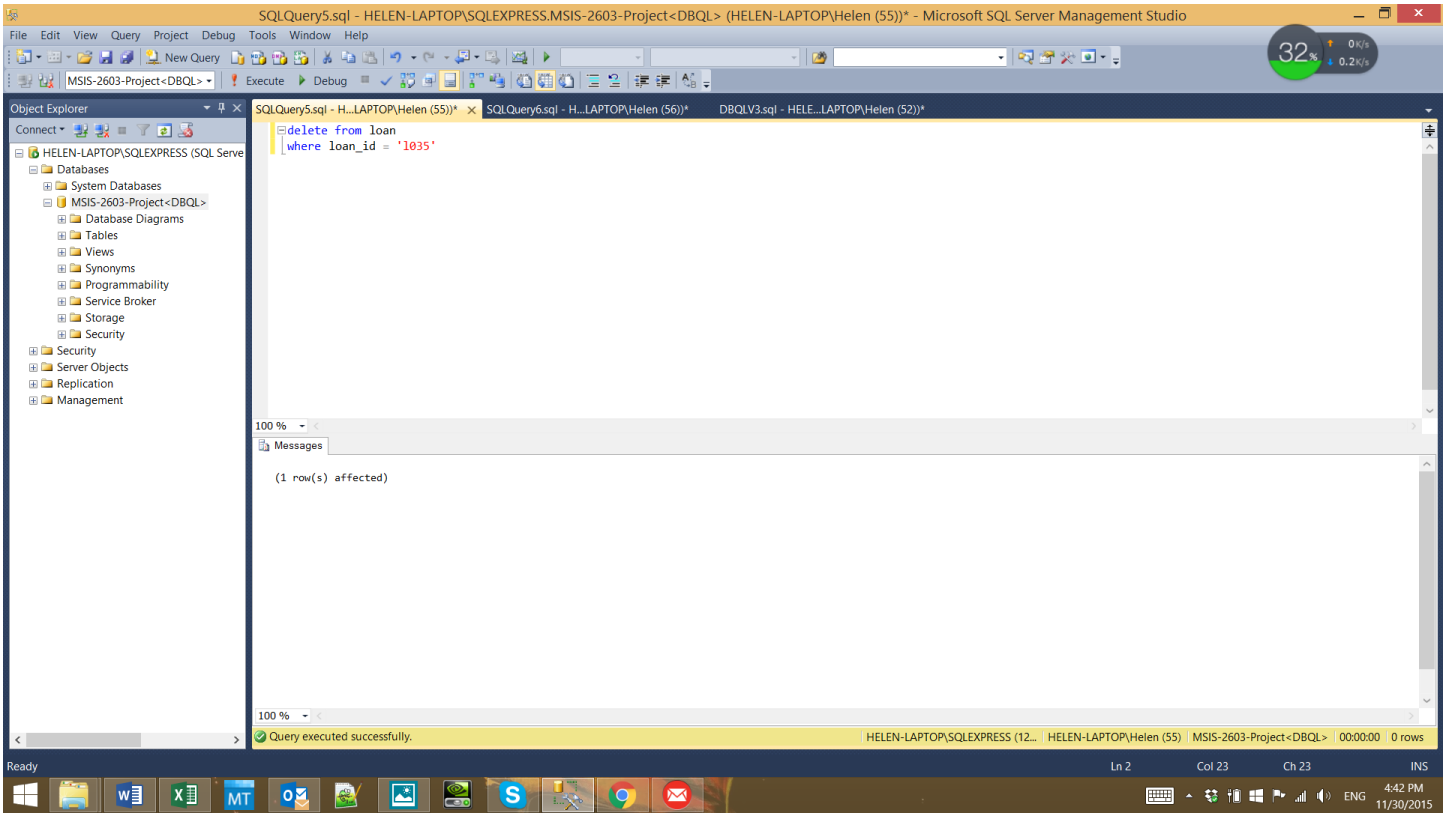


6. ACC_STATEMENT
    a. After

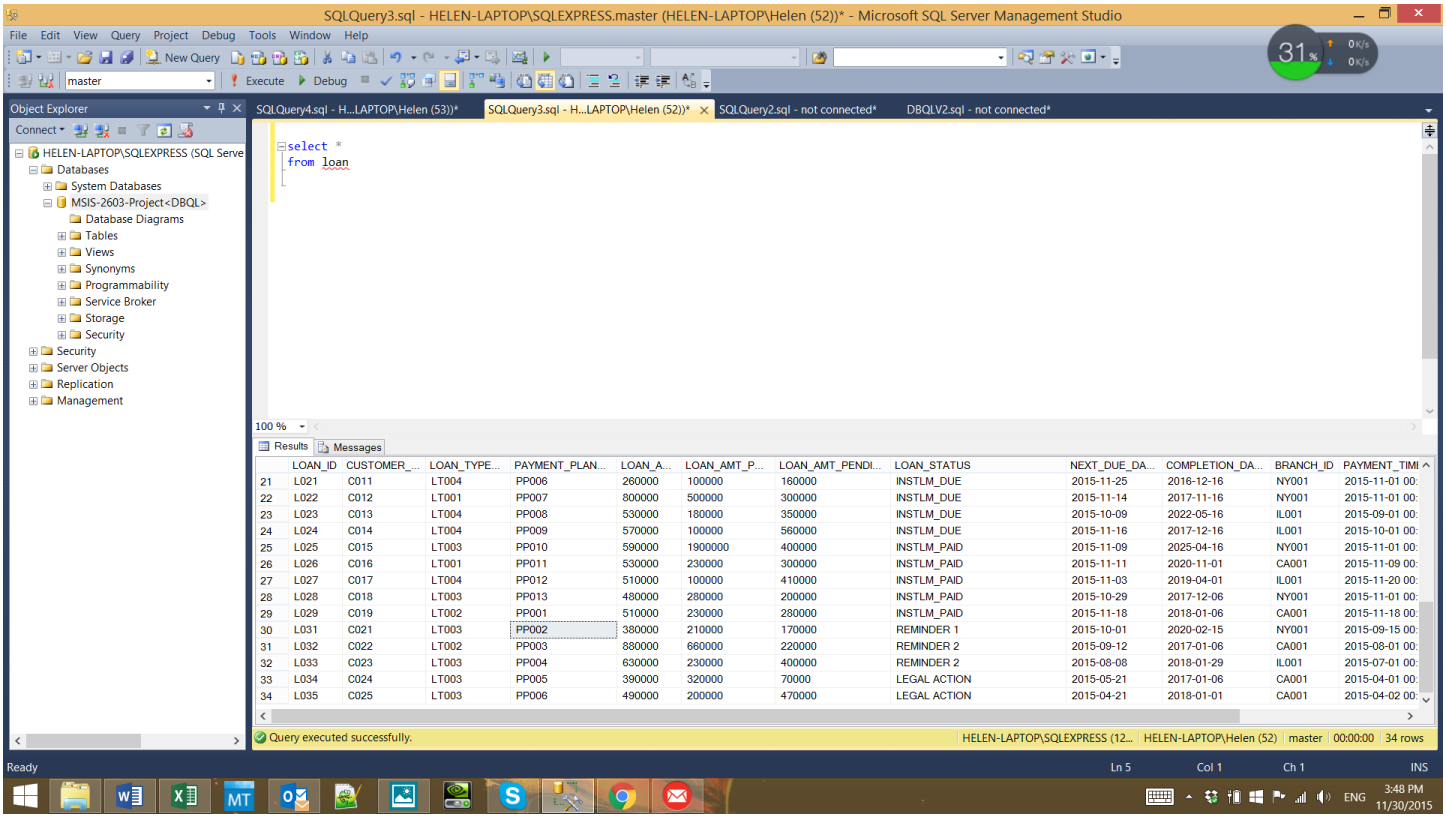


7. LOAN
    a. Before



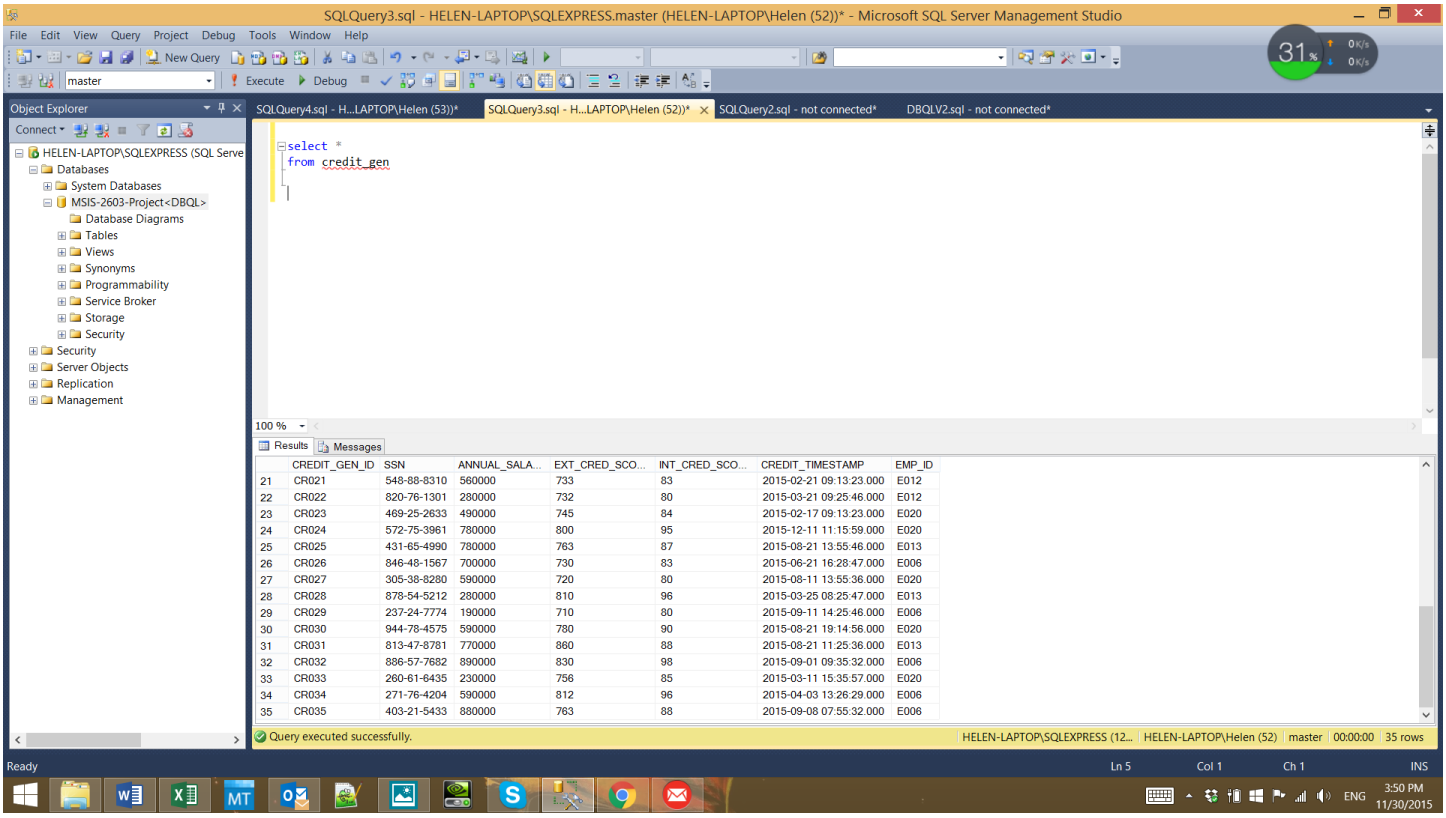

SQLQuery5.sql - HELEN-LAPTOP\SQLEXPRESS.MSIS-2603-Project<DBQL> (HELEN-LAPTOP\Helen (55))* - Microsoft SQL Server Management Studio
delete from loan
where loan_id = '1035'
(1 row(s) affected)
Query executed successfully.

b. After

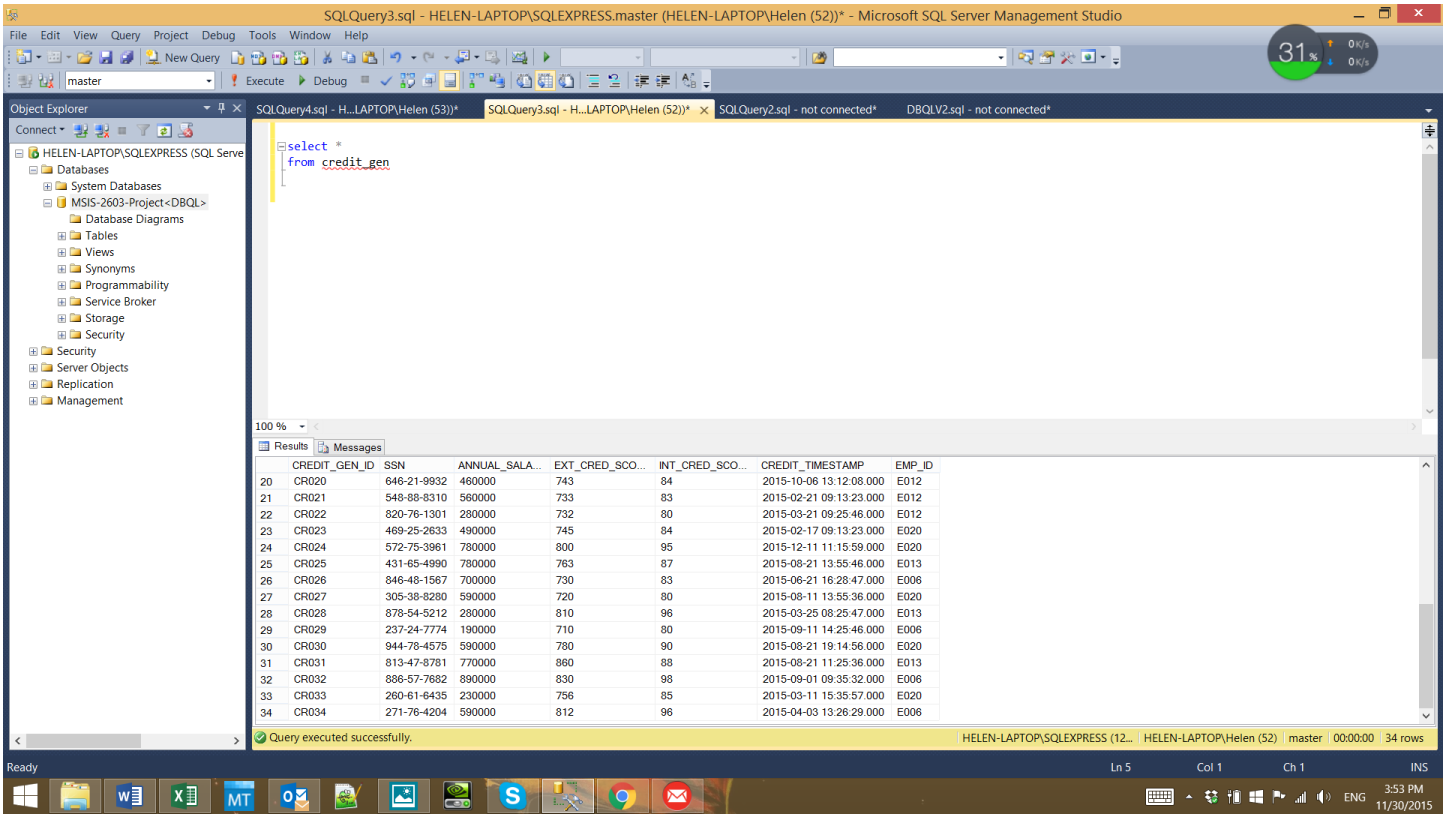

SQLQuery3.sql - HELEN-LAPTOP\SQLEXPRESS.master (HELEN-LAPTOP\Helen (52))* - Microsoft SQL Server Management Studio
select *
from loan
Query executed successfully.

8. CREDIT_GEN

a. Before

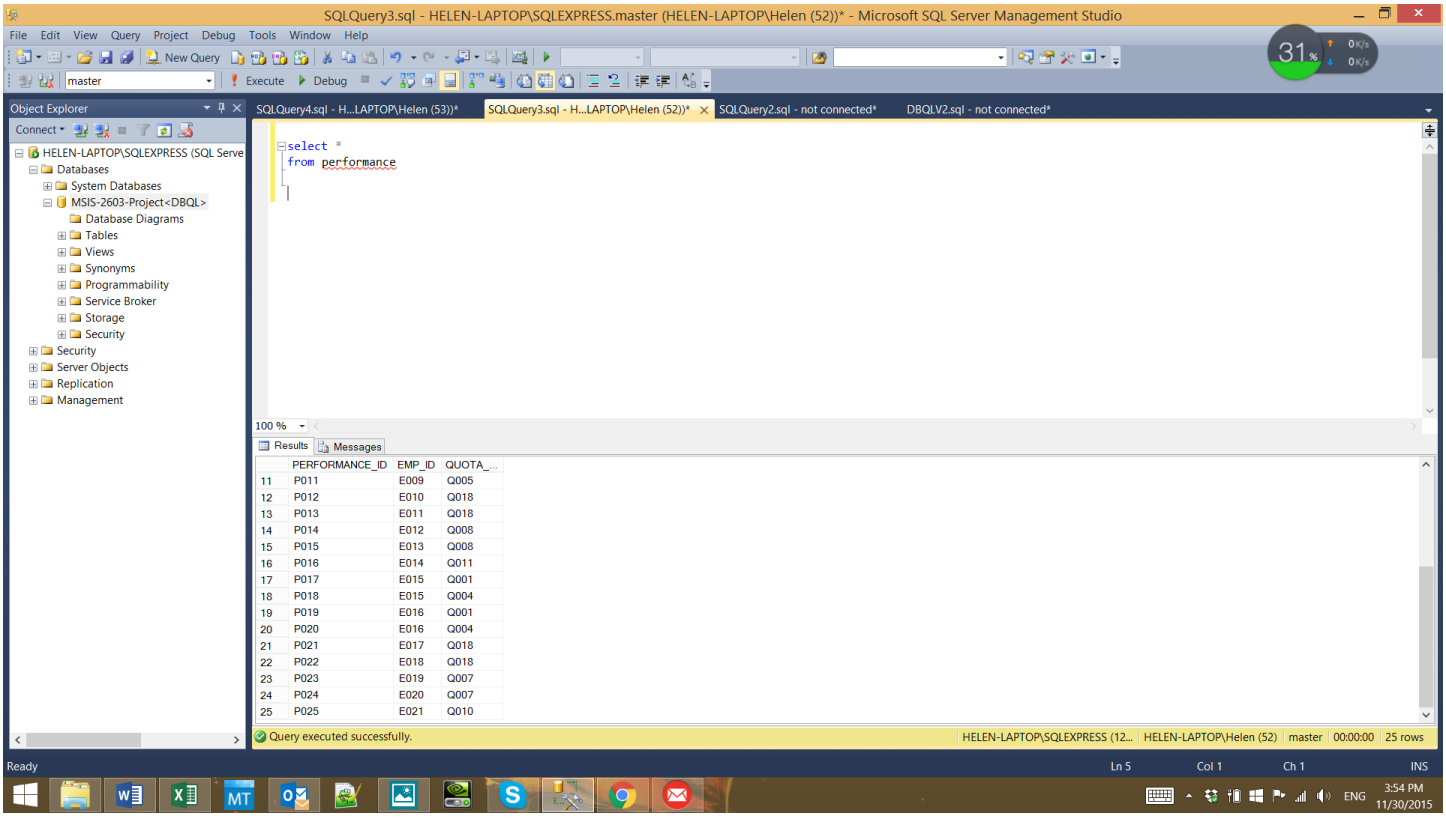


b. After



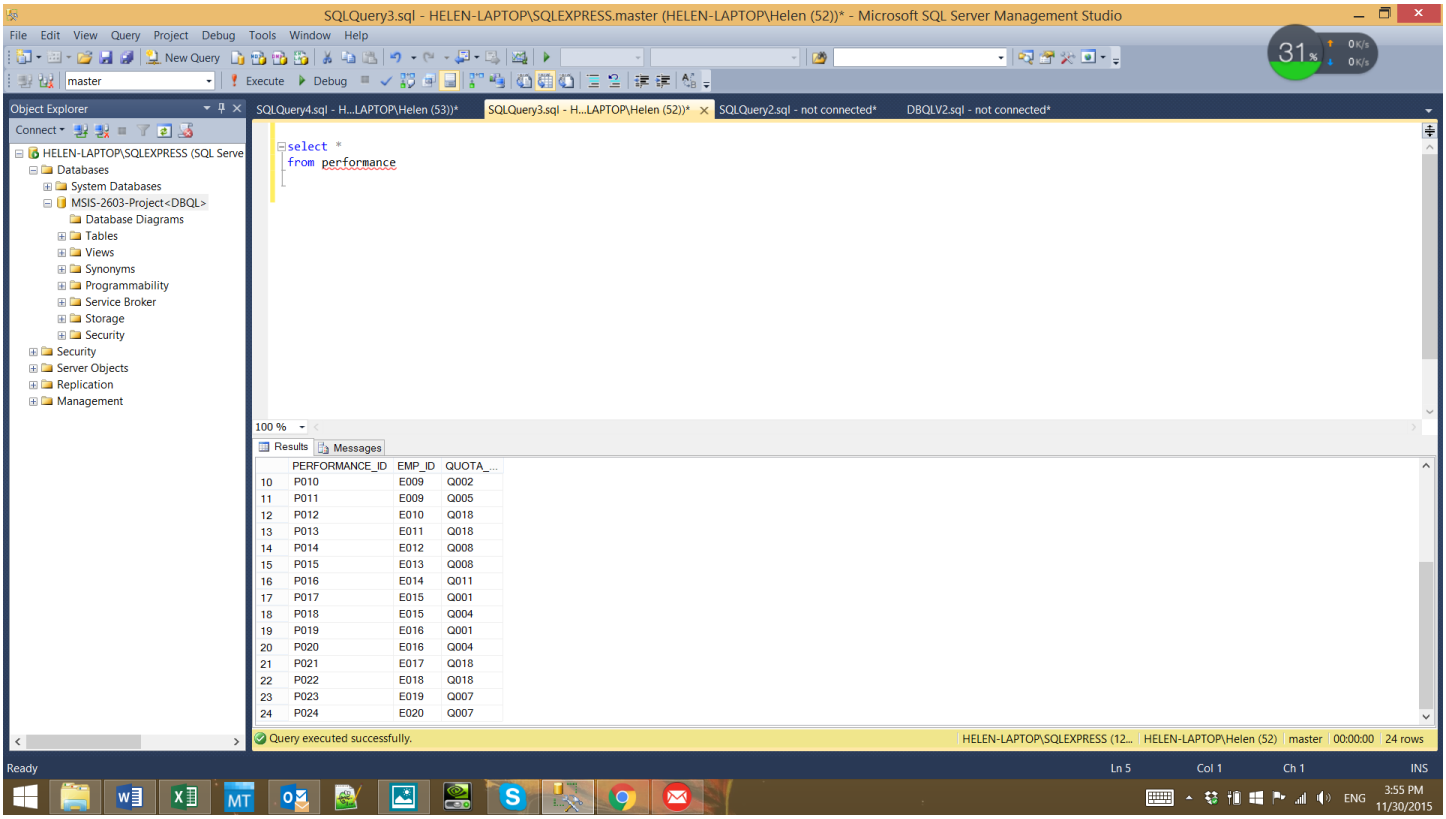
9. PERFORMANCE

a. Before



	PERFORMANCE_ID	EMP_ID	QUOTA_...
11	P011	E009	Q005
12	P012	E010	Q018
13	P013	E011	Q018
14	P014	E012	Q008
15	P015	E013	Q008
16	P016	E014	Q011
17	P017	E015	Q001
18	P018	E015	Q004
19	P019	E016	Q001
20	P020	E016	Q004
21	P021	E017	Q018
22	P022	E018	Q018
23	P023	E019	Q007
24	P024	E020	Q007
25	P025	E021	Q010

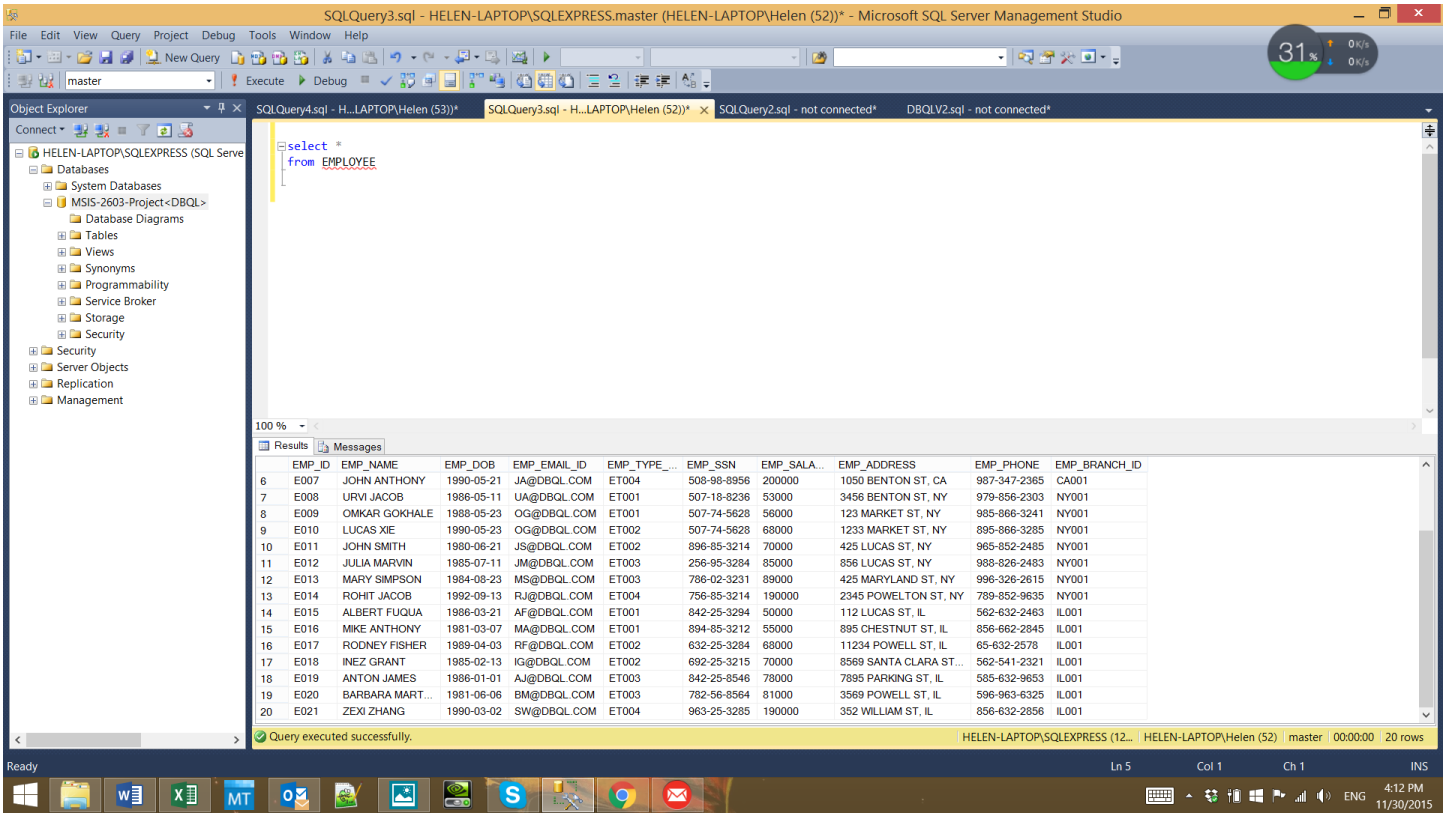
b. After



	PERFORMANCE_ID	EMP_ID	QUOTA_...
10	P010	E009	Q002
11	P011	E009	Q005
12	P012	E010	Q018
13	P013	E011	Q018
14	P014	E012	Q008
15	P015	E013	Q008
16	P016	E014	Q011
17	P017	E015	Q001
18	P018	E015	Q004
19	P019	E016	Q001
20	P020	E016	Q004
21	P021	E017	Q018
22	P022	E018	Q018
23	P023	E019	Q007
24	P024	E020	Q007

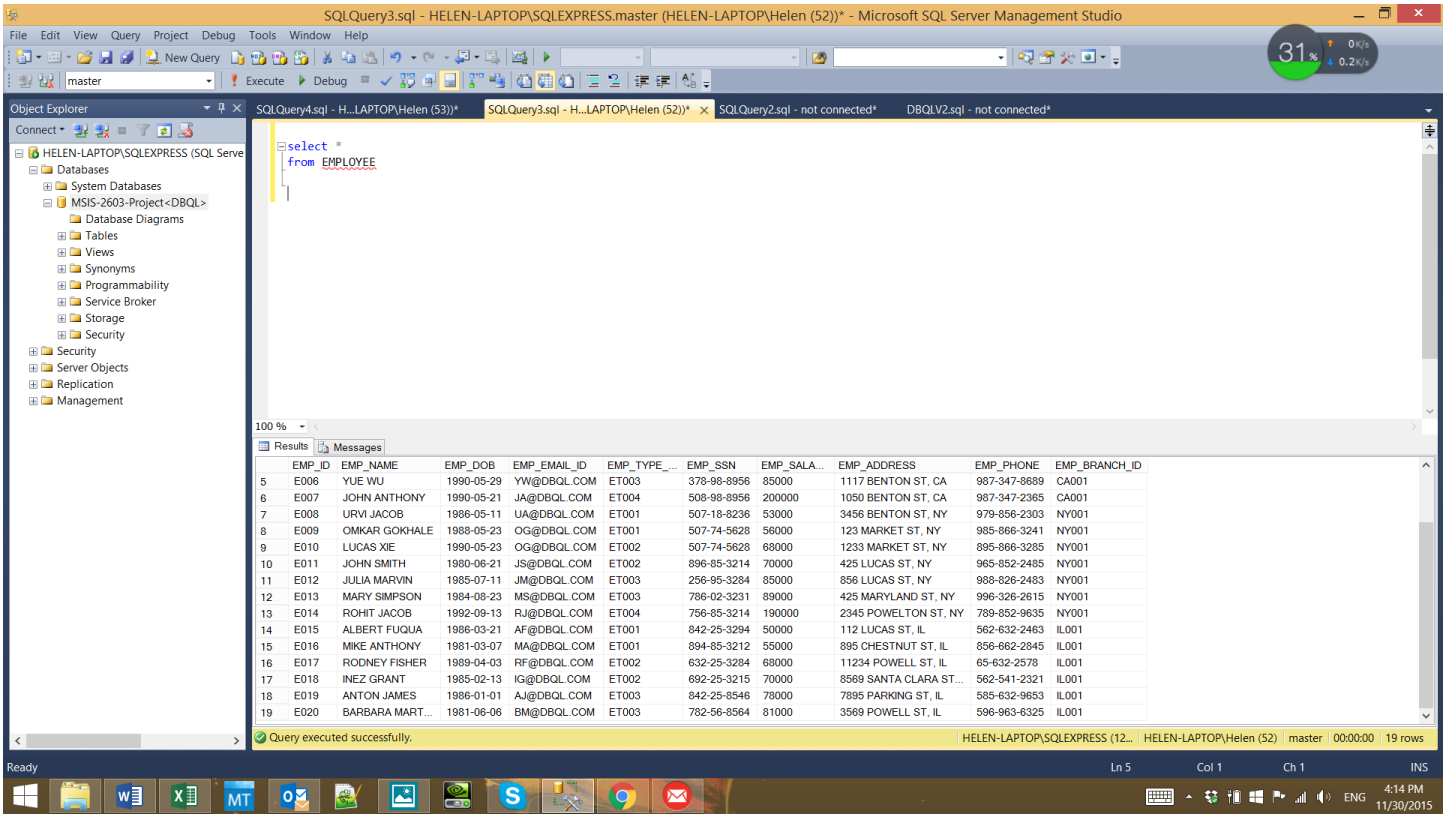
10. EMPLOYEE

a. Before



	EMP_ID	EMP_NAME	EMP_DOB	EMP_EMAIL_ID	EMP_TYPE_...	EMP_SSN	EMP_SALA...	EMP_ADDRESS	EMP_PHONE	EMP_BRANCH_ID
6	E007	JOHN ANTHONY	1990-05-21	JA@DBQL.COM	ET004	508-98-8956	200000	1050 BENTON ST, CA	987-347-2365	CA001
7	E008	URVI JACOB	1986-05-11	UA@DBQL.COM	ET001	507-18-8236	53000	3456 BENTON ST, NY	979-856-2303	NY001
8	E009	OMKAR GOKHALE	1988-05-23	OG@DBQL.COM	ET001	507-74-5628	56000	123 MARKET ST, NY	985-866-3241	NY001
9	E010	LUCAS XIE	1990-05-23	OG@DBQL.COM	ET002	507-74-5628	68000	1233 MARKET ST, NY	895-866-3285	NY001
10	E011	JOHN SMITH	1980-06-21	JS@DBQL.COM	ET002	896-85-3214	70000	425 LUCAS ST, NY	965-852-2485	NY001
11	E012	JULIA MARVIN	1985-07-11	JM@DBQL.COM	ET003	256-95-3284	85000	856 LUCAS ST, NY	988-826-2483	NY001
12	E013	MARY SIMPSON	1984-08-23	MS@DBQL.COM	ET003	786-02-3231	89000	425 MARYLAND ST, NY	996-326-2615	NY001
13	E014	ROHIT JACOB	1992-09-13	RJ@DBQL.COM	ET004	756-85-3214	190000	2345 POWELTON ST, NY	789-852-9635	NY001
14	E015	ALBERT FUQUA	1986-03-21	AF@DBQL.COM	ET001	842-25-3294	50000	112 LUCAS ST, IL	562-632-2463	IL001
15	E016	MIKE ANTHONY	1981-03-07	MA@DBQL.COM	ET001	894-85-3212	55000	895 CHESTNUT ST, IL	856-662-2845	IL001
16	E017	RODNEY FISHER	1989-04-03	RF@DBQL.COM	ET002	632-25-3284	68000	11234 POWELL ST, IL	65-632-2578	IL001
17	E018	INEZ GRANT	1985-02-13	IG@DBQL.COM	ET002	692-25-3215	70000	8569 SANTA CLARA ST...	562-541-2321	IL001
18	E019	ANTON JAMES	1986-01-01	AJ@DBQL.COM	ET003	842-25-8546	78000	7895 PARKING ST, IL	585-632-9653	IL001
19	E020	BARBARA MART...	1981-06-06	BM@DBQL.COM	ET003	782-56-8564	81000	3569 POWELL ST, IL	596-963-6325	IL001
20	E021	ZEXI ZHANG	1990-03-02	SW@DBQL.COM	ET004	963-25-3285	190000	352 WILLIAM ST, IL	856-632-2856	IL001

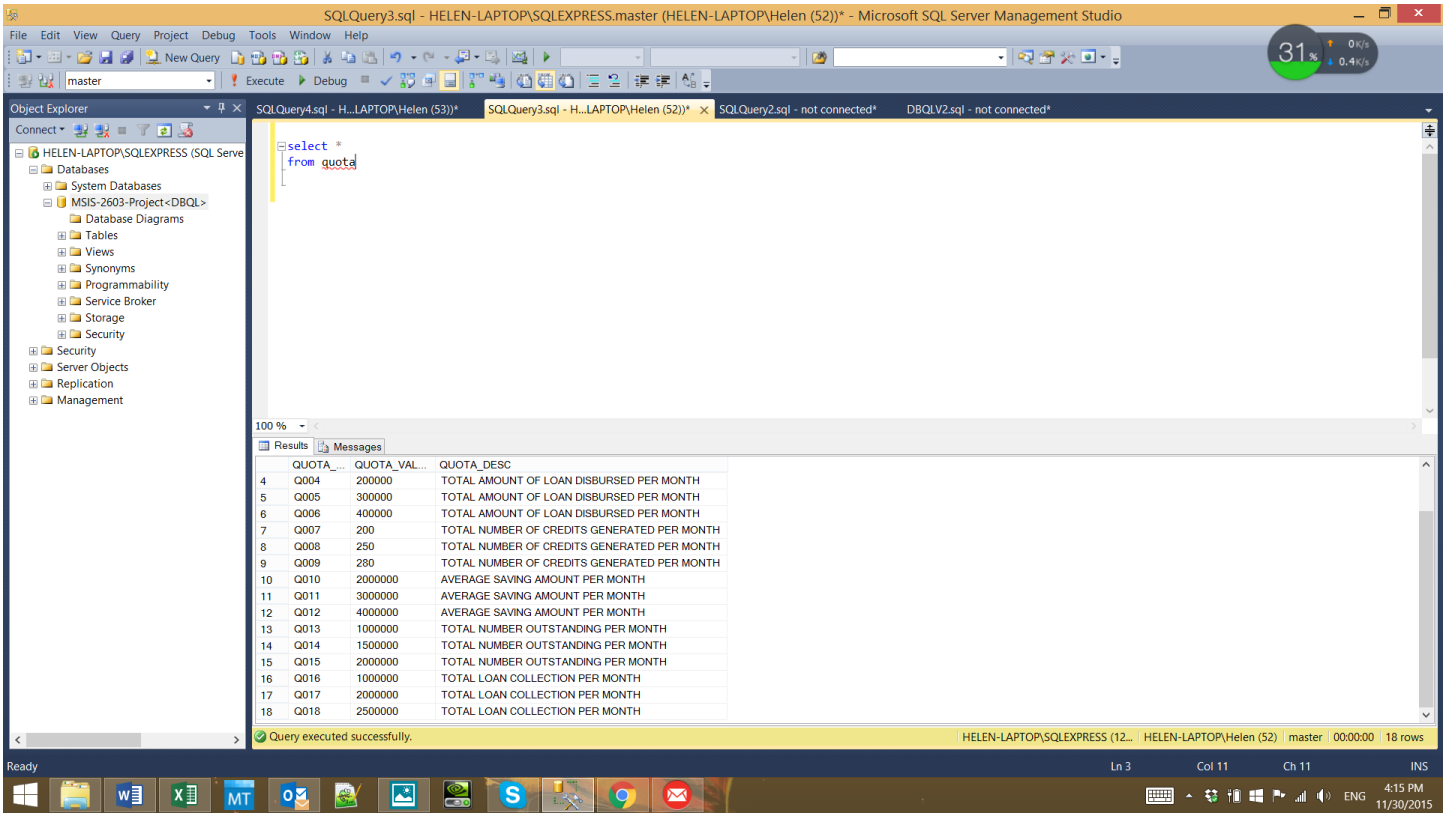
b. After

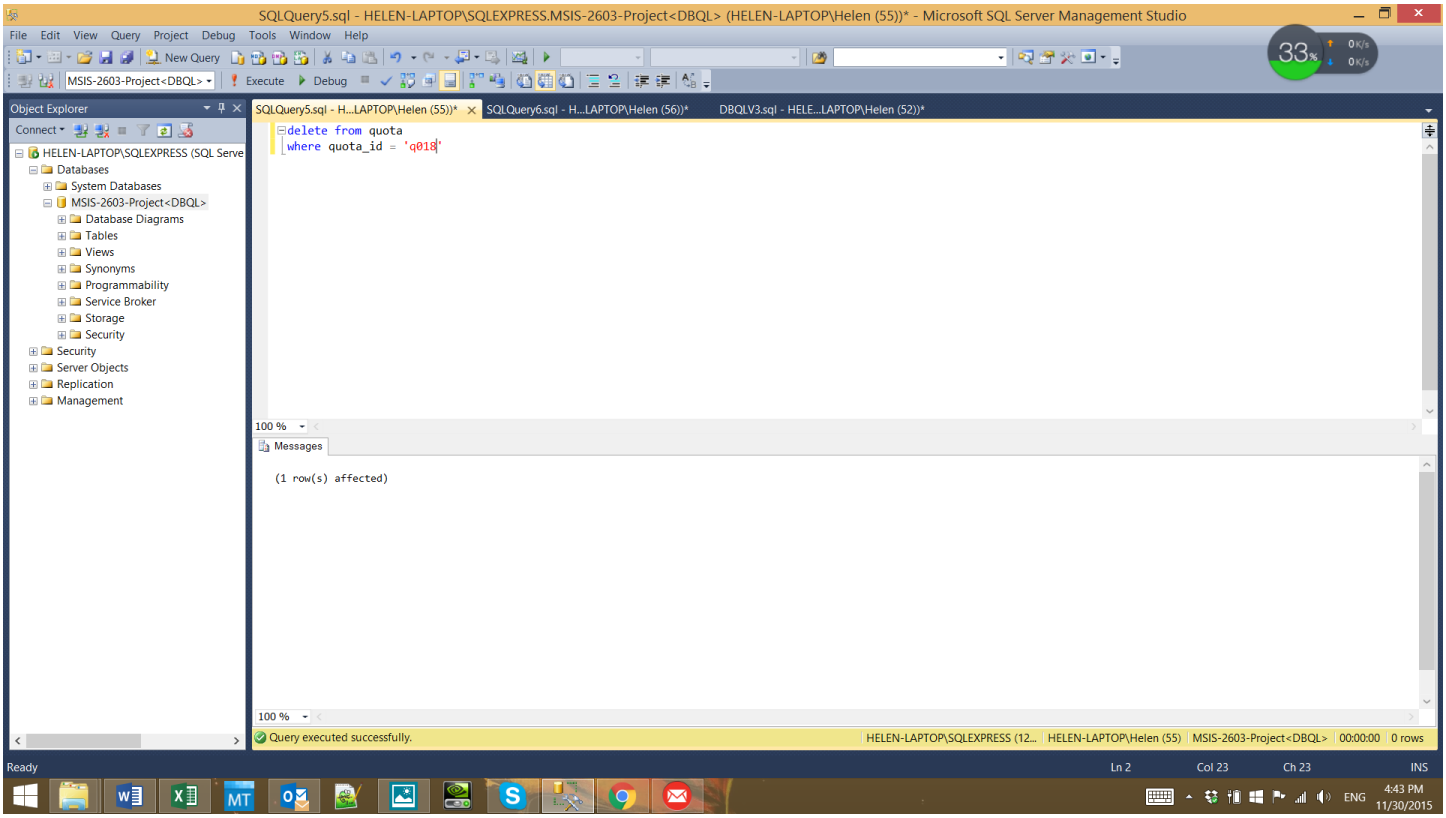


	EMP_ID	EMP_NAME	EMP_DOB	EMP_EMAIL_ID	EMP_TYPE_...	EMP_SSN	EMP_SALA...	EMP_ADDRESS	EMP_PHONE	EMP_BRANCH_ID
5	E006	YUE WU	1990-05-29	YW@DBQL.COM	ET003	378-98-8956	85000	1117 BENTON ST, CA	987-347-8689	CA001
6	E007	JOHN ANTHONY	1990-05-21	JA@DBQL.COM	ET004	508-98-8956	200000	1050 BENTON ST, CA	987-347-2365	CA001
7	E008	URVI JACOB	1986-05-11	UA@DBQL.COM	ET001	507-18-8236	53000	3456 BENTON ST, NY	979-856-2303	NY001
8	E009	OMKAR GOKHALE	1988-05-23	OG@DBQL.COM	ET001	507-74-5628	56000	123 MARKET ST, NY	985-866-3241	NY001
9	E010	LUCAS XIE	1990-05-23	OG@DBQL.COM	ET002	507-74-5628	68000	1233 MARKET ST, NY	895-866-3285	NY001
10	E011	JOHN SMITH	1980-06-21	JS@DBQL.COM	ET002	896-85-3214	70000	425 LUCAS ST, NY	965-852-2485	NY001
11	E012	JULIA MARVIN	1985-07-11	JM@DBQL.COM	ET003	256-95-3284	85000	856 LUCAS ST, NY	988-826-2483	NY001
12	E013	MARY SIMPSON	1984-08-23	MS@DBQL.COM	ET003	786-02-3231	89000	425 MARYLAND ST, NY	996-326-2615	NY001
13	E014	ROHIT JACOB	1992-09-13	RJ@DBQL.COM	ET004	756-85-3214	190000	2345 POWELTON ST, NY	789-852-9635	NY001
14	E015	ALBERT FUQUA	1986-03-21	AF@DBQL.COM	ET001	842-25-3294	50000	112 LUCAS ST, IL	562-632-2463	IL001
15	E016	MIKE ANTHONY	1981-03-07	MA@DBQL.COM	ET001	894-85-3212	55000	895 CHESTNUT ST, IL	856-662-2845	IL001
16	E017	RODNEY FISHER	1989-04-03	RF@DBQL.COM	ET002	632-25-3284	68000	11234 POWELL ST, IL	65-632-2578	IL001
17	E018	INEZ GRANT	1985-02-13	IG@DBQL.COM	ET002	692-25-3215	70000	8569 SANTA CLARA ST...	562-541-2321	IL001
18	E019	ANTON JAMES	1986-01-01	AJ@DBQL.COM	ET003	842-25-8546	78000	7895 PARKING ST, IL	585-632-9653	IL001
19	E020	BARBARA MART...	1981-06-06	BM@DBQL.COM	ET003	782-56-8564	81000	3569 POWELL ST, IL	596-963-6325	IL001

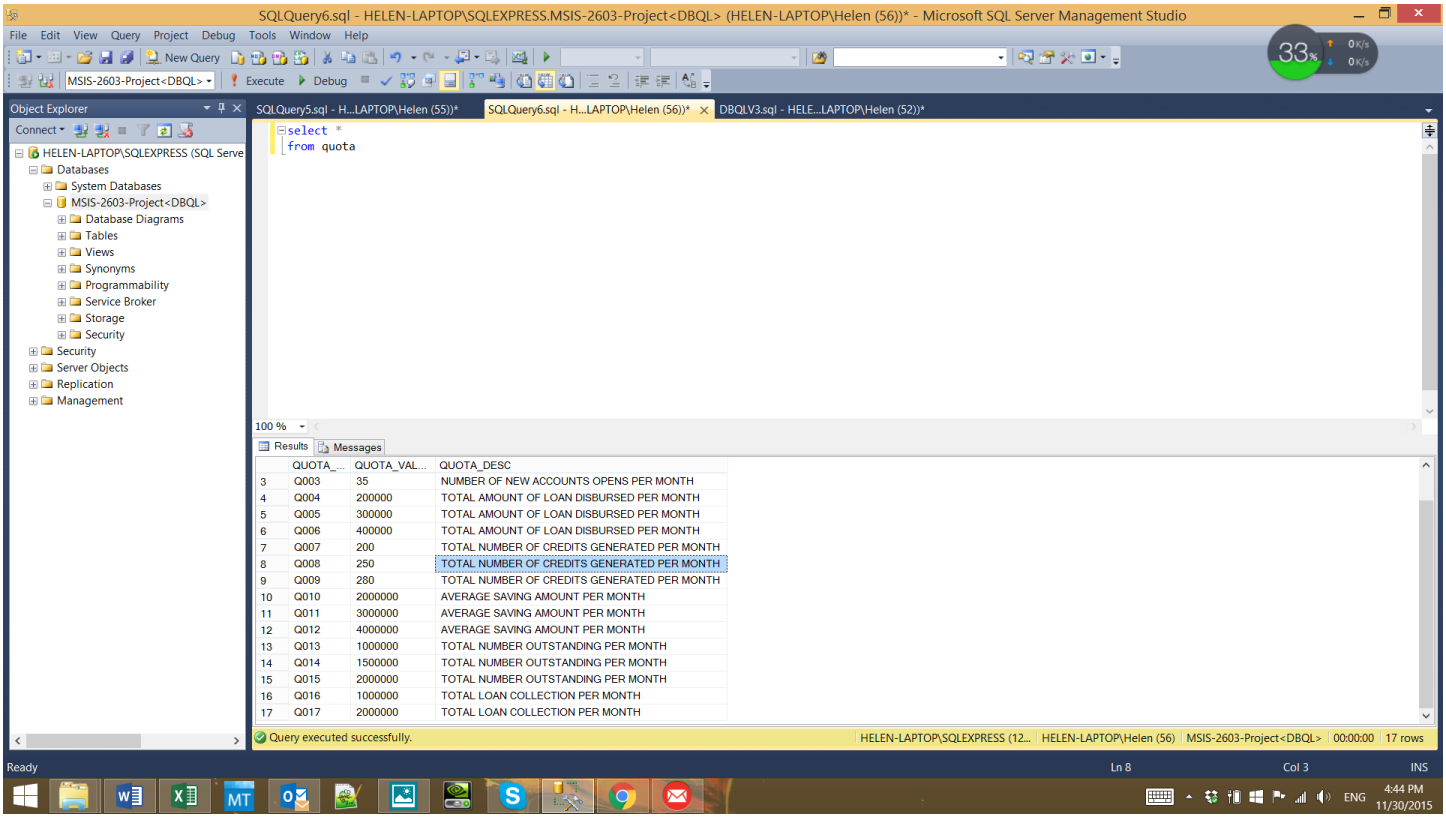
11. QUOTA

a. Before





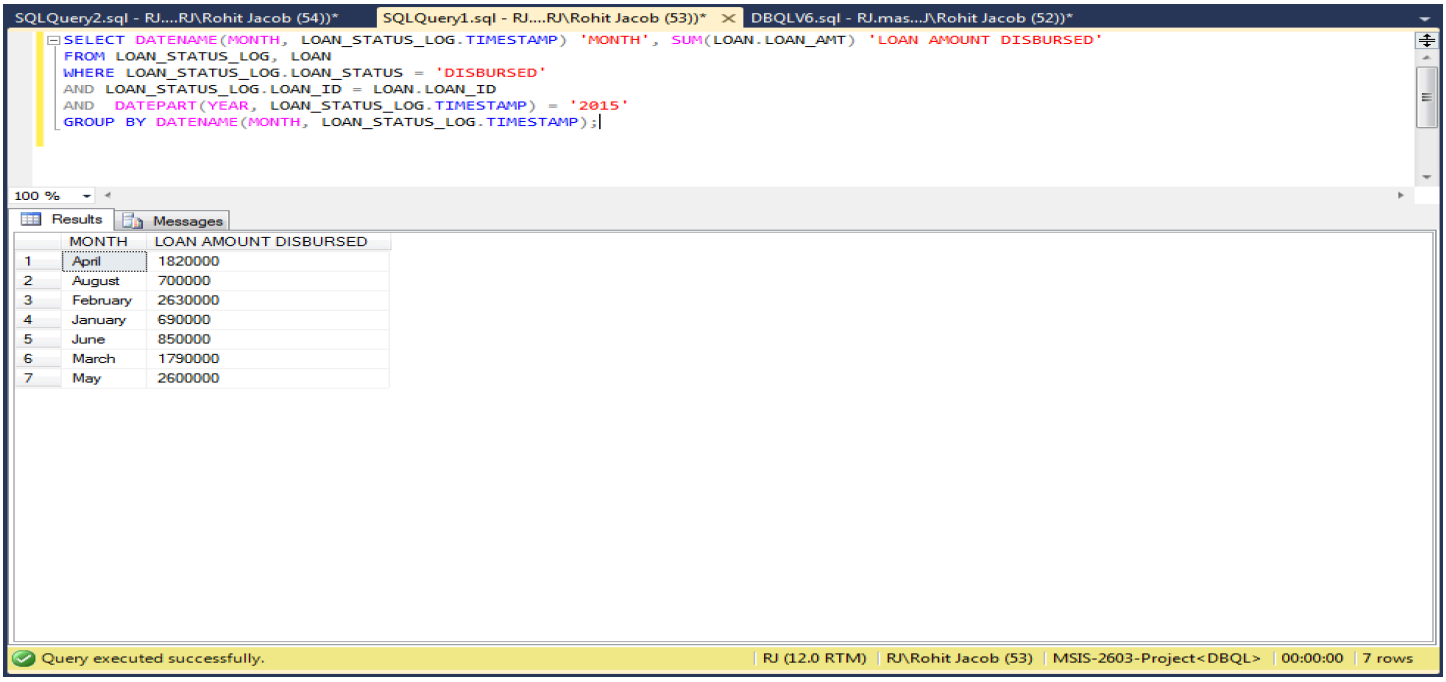
b. After

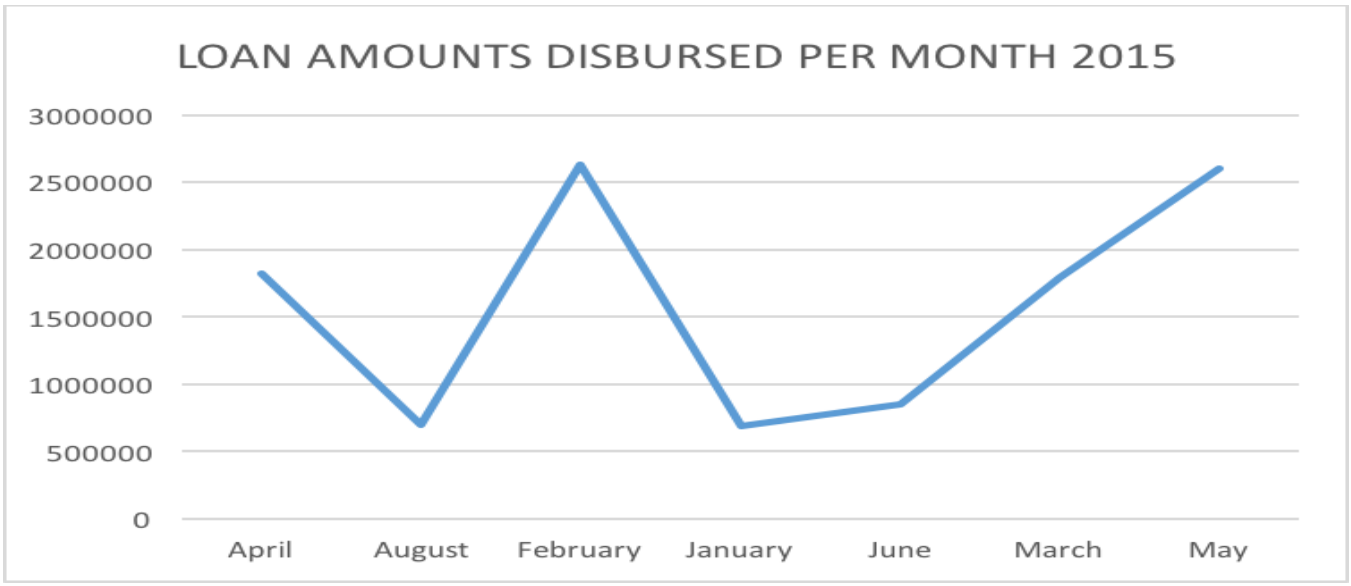

SQLQuery6.sql - HELEN-LAPTOP\SQLEXPRESS.MSIS-2603-Project<DBQL> (HELEN-LAPTOP\Helen (56))* - Microsoft SQL Server Management Studio
File Edit View Query Project Debug Tools Window Help
New Query
MSIS-2603-Project<DBQL>
Execute
Debug
Object Explorer
Connect
HELEN-LAPTOP\SQLEXPRESS (SQL Serve
Databases
System Databases
MSIS-2603-Project<DBQL>
Database Diagrams
Tables
Views
Synonyms
Programmability
Service Broker
Storage
Security
Security
Server Objects
Replication
Management
SQLQuery5.sql - H...LAPTOP\Helen (55))*
SQLQuery6.sql - H...LAPTOP\Helen (56))*
DBQLV3.sql - HELE...LAPTOP\Helen (52))*
select *
from quota
100 %
Results
Messages
QUOTA_... QUOTA_VAL... QUOTA_DESC
3 Q003 35 NUMBER OF NEW ACCOUNTS OPENS PER MONTH
4 Q004 200000 TOTAL AMOUNT OF LOAN DISBURSED PER MONTH
5 Q005 300000 TOTAL AMOUNT OF LOAN DISBURSED PER MONTH
6 Q006 400000 TOTAL AMOUNT OF LOAN DISBURSED PER MONTH
7 Q007 200 TOTAL NUMBER OF CREDITS GENERATED PER MONTH
8 Q008 250 TOTAL NUMBER OF CREDITS GENERATED PER MONTH
9 Q009 280 TOTAL NUMBER OF CREDITS GENERATED PER MONTH
10 Q010 2000000 AVERAGE SAVING AMOUNT PER MONTH
11 Q011 3000000 AVERAGE SAVING AMOUNT PER MONTH
12 Q012 4000000 AVERAGE SAVING AMOUNT PER MONTH
13 Q013 1000000 TOTAL NUMBER OUTSTANDING PER MONTH
14 Q014 1500000 TOTAL NUMBER OUTSTANDING PER MONTH
15 Q015 2000000 TOTAL NUMBER OUTSTANDING PER MONTH
16 Q016 1000000 TOTAL LOAN COLLECTION PER MONTH
17 Q017 2000000 TOTAL LOAN COLLECTION PER MONTH
Query executed successfully.
HELEN-LAPTOP\SQLEXPRESS (12...
HELEN-LAPTOP\Helen (56)
MSIS-2603-Project<DBQL>
00:00:00
17 rows
Ready
Ln 8
Col 3
INS
ENG
4:44 PM
11/30/2015

Business Metrics (SQL Queries and Reports)

1. Loan Amount Disbursed Per Month

```
SELECT DATENAME(MONTH, LOAN_STATUS_LOG.TIMESTAMP) 'MONTH',
SUM(LOAN.LOAN_AMT) 'LOAN AMOUNT DISBURSED'
FROM LOAN_STATUS_LOG, LOAN
WHERE LOAN_STATUS_LOG.LOAN_STATUS = 'DISBURSED'
AND LOAN_STATUS_LOG.LOAN_ID = LOAN.LOAN_ID
AND  DATEPART(YEAR, LOAN_STATUS_LOG.TIMESTAMP) = '2015'
GROUP BY DATENAME(MONTH, LOAN_STATUS_LOG.TIMESTAMP);
```





2. Accounts Opened Per Month

```
SELECT DATENAME(MONTH, ACCOUNT.ACC_OPEN_DATE) AS 'MONTH',
EMPLOYEE.EMP_ID, COUNT(*) 'ACCOUNTS OPENED'
FROM CUSTOMER, ACCOUNT, EMPLOYEE
WHERE CUSTOMER.DESIGNATED_REP_ID = EMPLOYEE.EMP_ID
AND CUSTOMER.CUSTOMER_ID = ACCOUNT.CUSTOMER_ID
AND DATEPART(YEAR, ACCOUNT.ACC_OPEN_DATE) = '2012'
GROUP BY DATENAME(MONTH, ACCOUNT.ACC_OPEN_DATE), EMPLOYEE.EMP_ID;
```

SQLQuery2.sql - RJ....RJ\Rohit Jacob (54))* | SQLQuery1.sql - RJ....RJ\Rohit Jacob (53))* | DBQLV6.sql - RJ.mas...J\Rohit Jacob (52))*

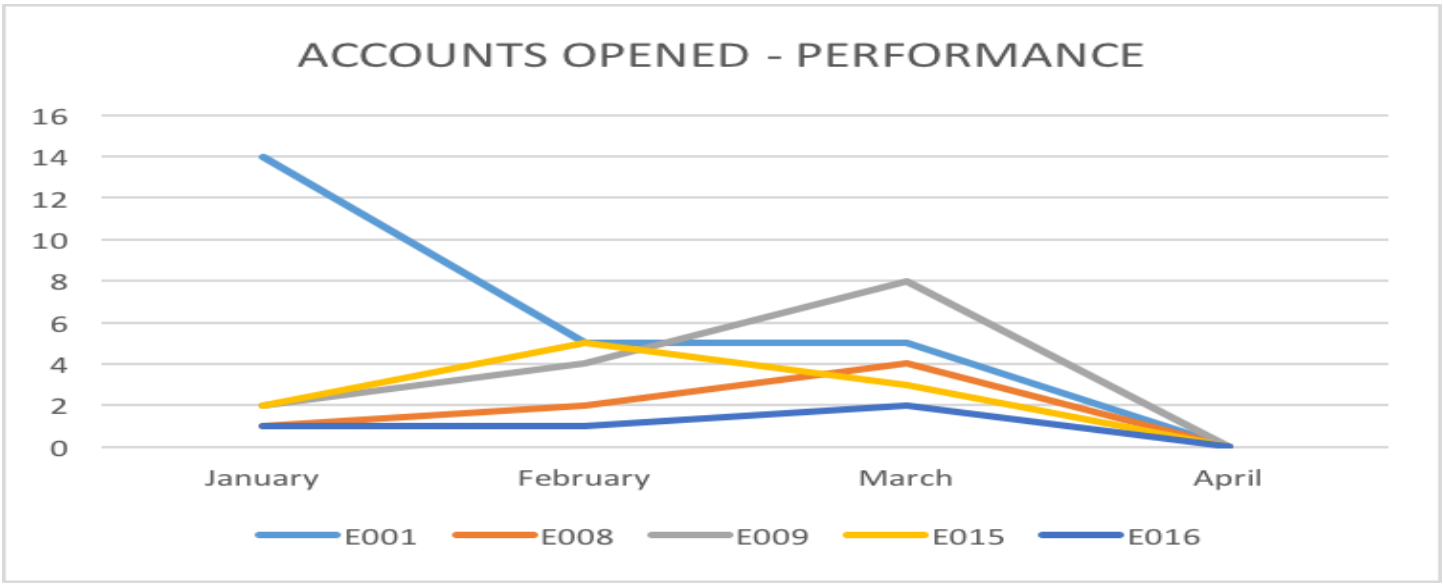
```
FROM CUSTOMER, ACCOUNT, EMPLOYEE
WHERE CUSTOMER.DESIGNATED_REP_ID = EMPLOYEE.EMP_ID
AND CUSTOMER.CUSTOMER_ID = ACCOUNT.CUSTOMER_ID
AND DATEPART(YEAR, ACCOUNT.ACC_OPEN_DATE) = '2012'
GROUP BY DATENAME(MONTH, ACCOUNT.ACC_OPEN_DATE), EMPLOYEE.EMP_ID;
```

100 %

Results | Messages

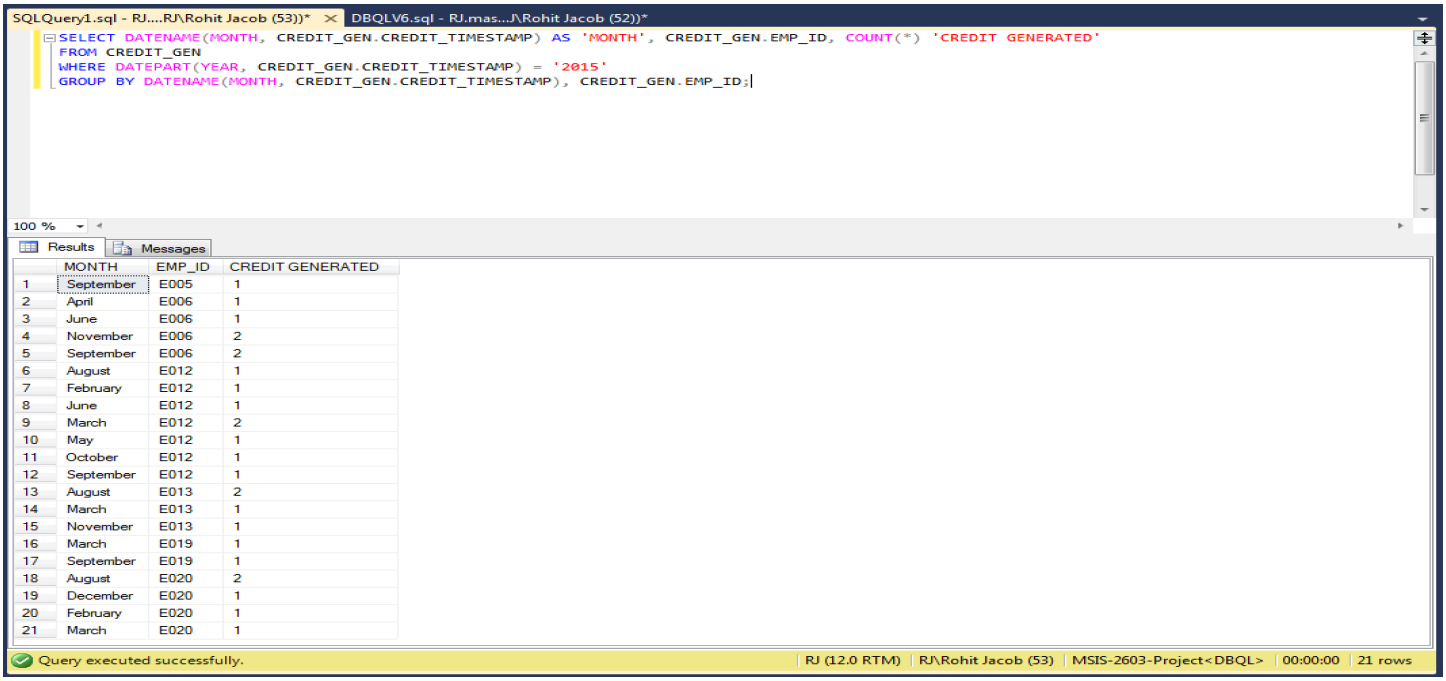
	MONTH	EMP_ID	ACCOUNTS OPENED
1	February	E001	5
2	January	E001	14
3	March	E001	5
4	February	E008	2
5	January	E008	1
6	March	E008	4
7	February	E009	4
8	January	E009	2
9	March	E009	8
10	February	E015	5
11	January	E015	2
12	March	E015	3
13	February	E016	1
14	January	E016	1
15	March	E016	2

Query executed successfully. | RJ (12.0 RTM) | RJ\Rohit Jacob (53) | MSIS-2603-Project<DBQL> | 00:00:00 | 15 rows

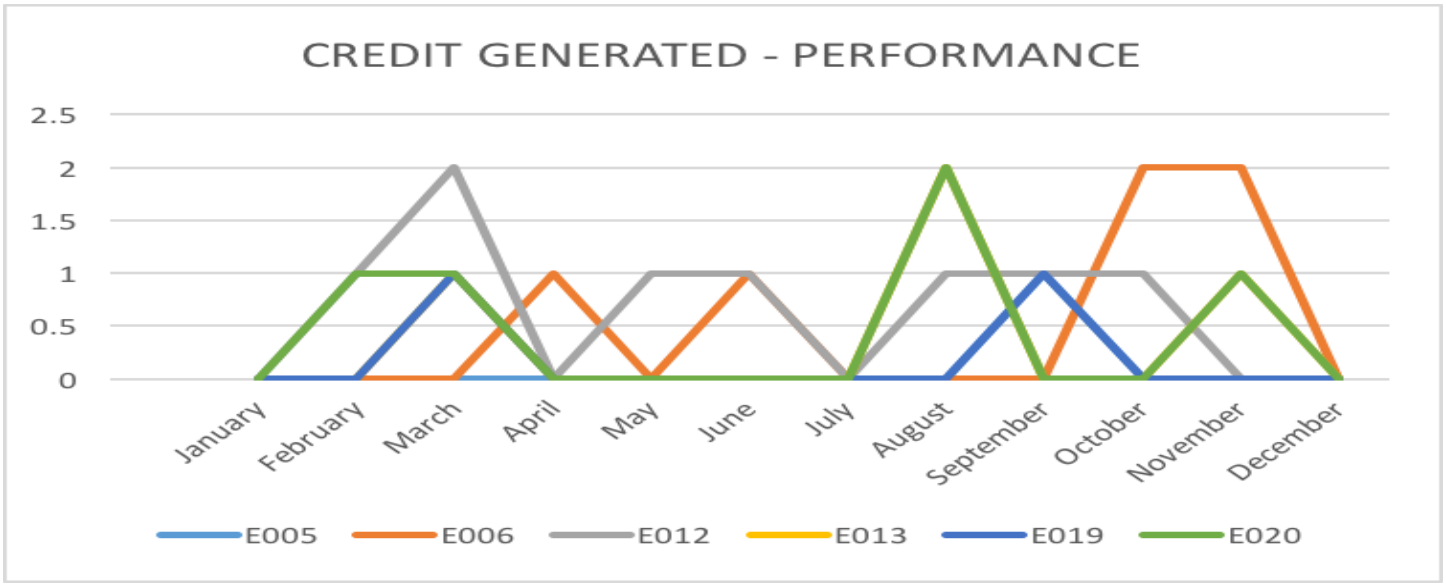


3. Credit Analyst KPI – Credit Generated/Month

```
SELECT DATENAME(MONTH, CREDIT_GEN.CREDIT_TIMESTAMP) AS 'MONTH',
CREDIT_GEN.EMP_ID, COUNT(*) 'CREDIT GENERATED'
FROM CREDIT_GEN
WHERE DATEPART(YEAR, CREDIT_GEN.CREDIT_TIMESTAMP) = '2015'
GROUP BY DATENAME(MONTH, CREDIT_GEN.CREDIT_TIMESTAMP),
CREDIT_GEN.EMP_ID;
```

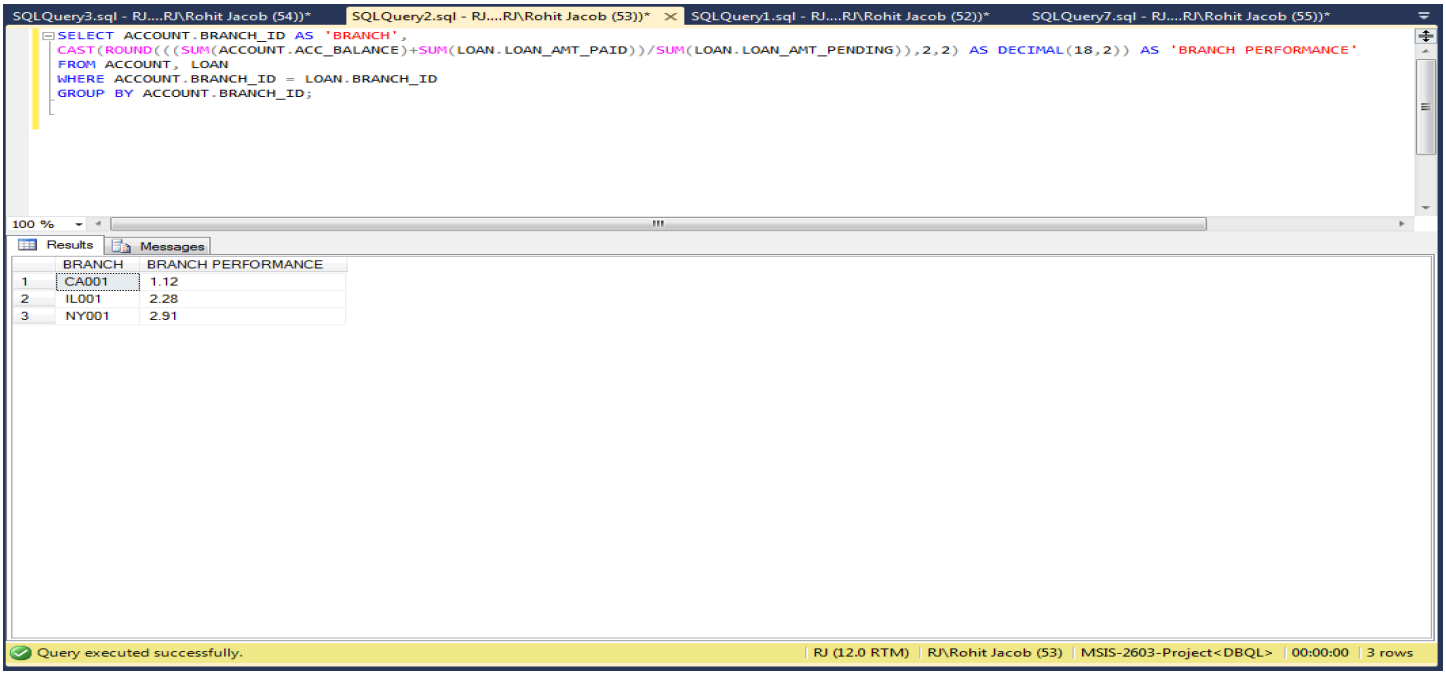


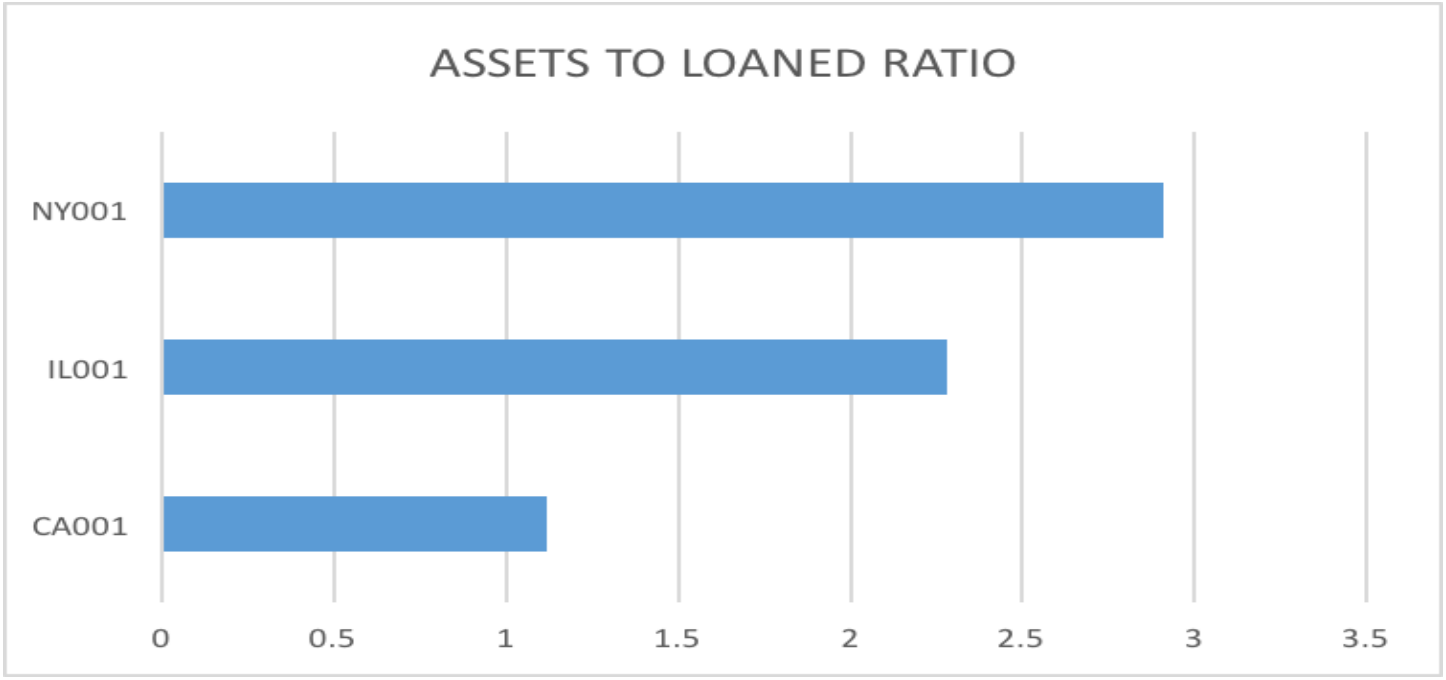
	MONTH	EMP_ID	CREDIT GENERATED
1	September	E005	1
2	April	E006	1
3	June	E006	1
4	November	E006	2
5	September	E006	2
6	August	E012	1
7	February	E012	1
8	June	E012	1
9	March	E012	2
10	May	E012	1
11	October	E012	1
12	September	E012	1
13	August	E013	2
14	March	E013	1
15	November	E013	1
16	March	E019	1
17	September	E019	1
18	August	E020	2
19	December	E020	1
20	February	E020	1
21	March	E020	1



4. Branch Performance – Assets to Loan Ratio

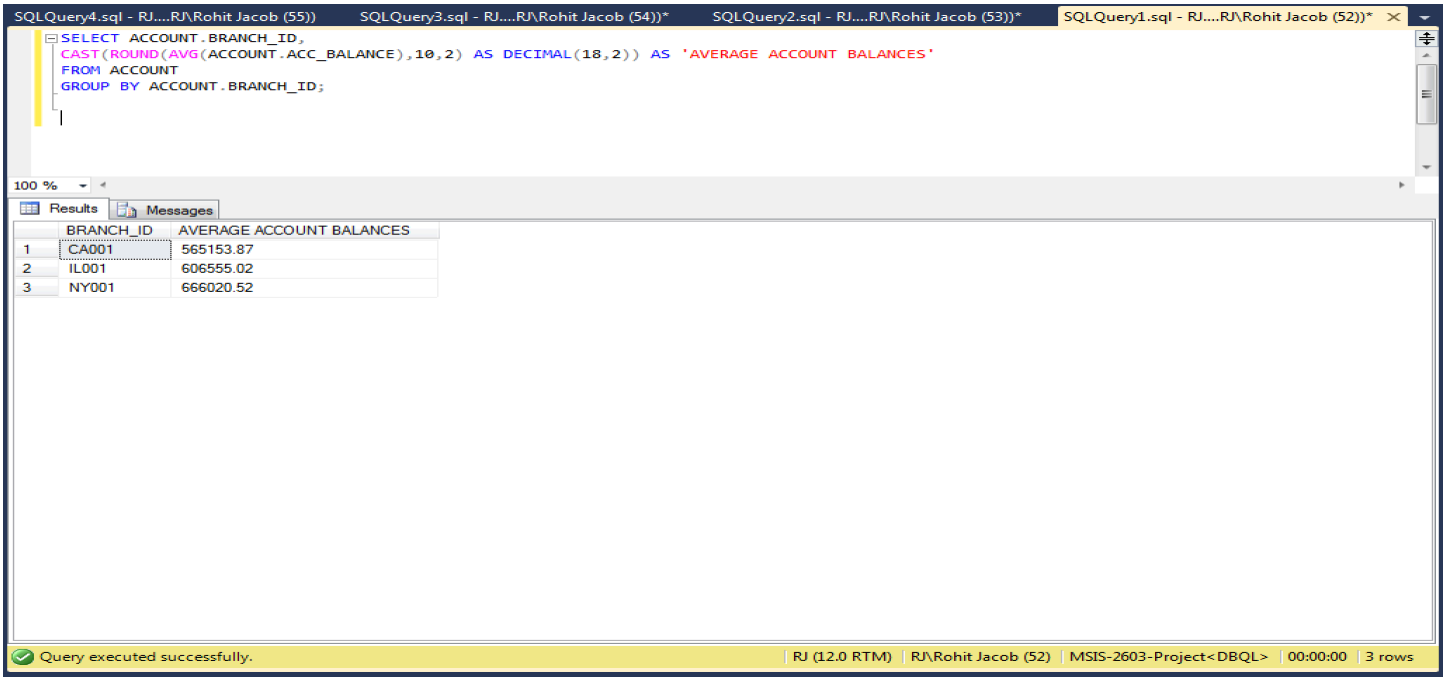
```
SELECT ACCOUNT.BRANCH_ID AS 'BRANCH'
CAST(ROUND((SUM(ACCOUNT.ACC_BALANCE) +
SUM(LOAN.LOAN_AMT_PAID))/SUM(LOAN.LOAN_AMT_PENDING)), 2, 2) AS
DECIMAL(18,2)) AS 'BRANCH PERFORMANCE'
FROM ACCOUNT LOAN
WHERE ACCOUNT.BRANCH_ID = LOAN.BRANCH_ID
GROUP BY ACCOUNT.BRANCH_ID;
```

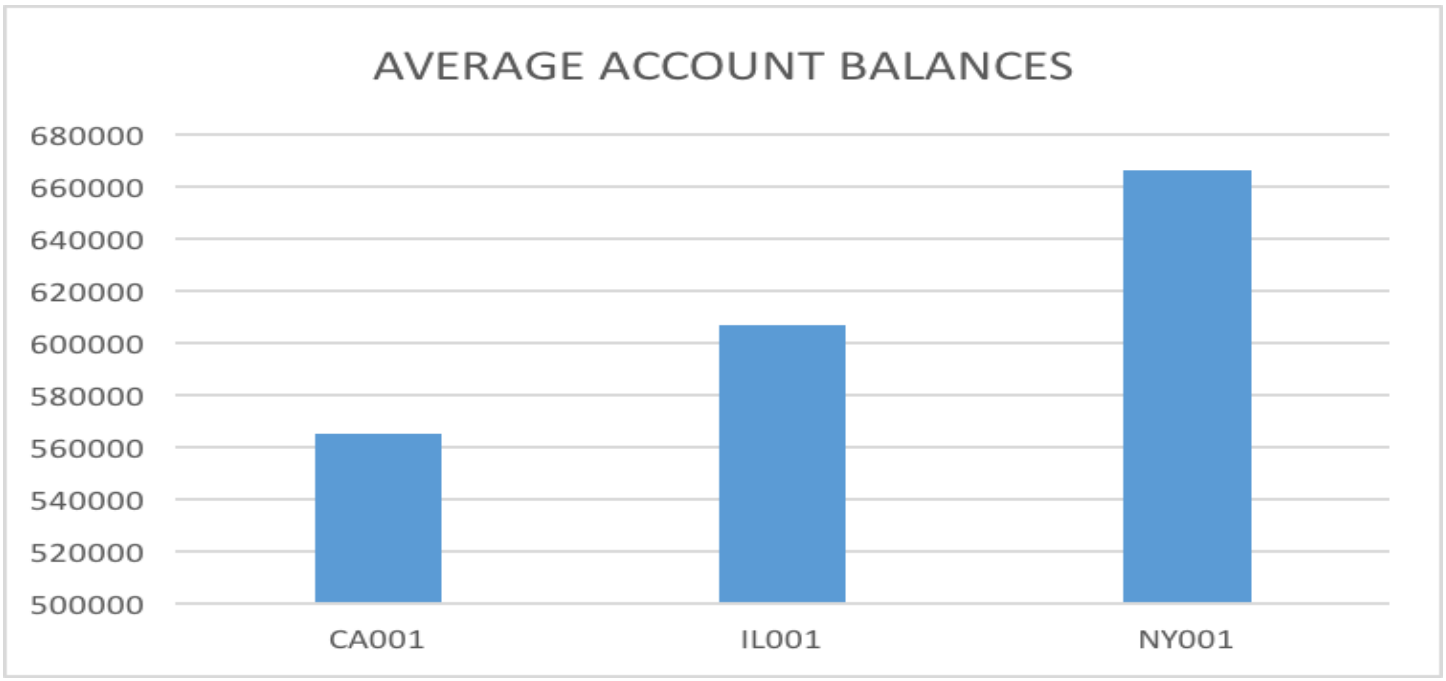




5. Average Account Balances

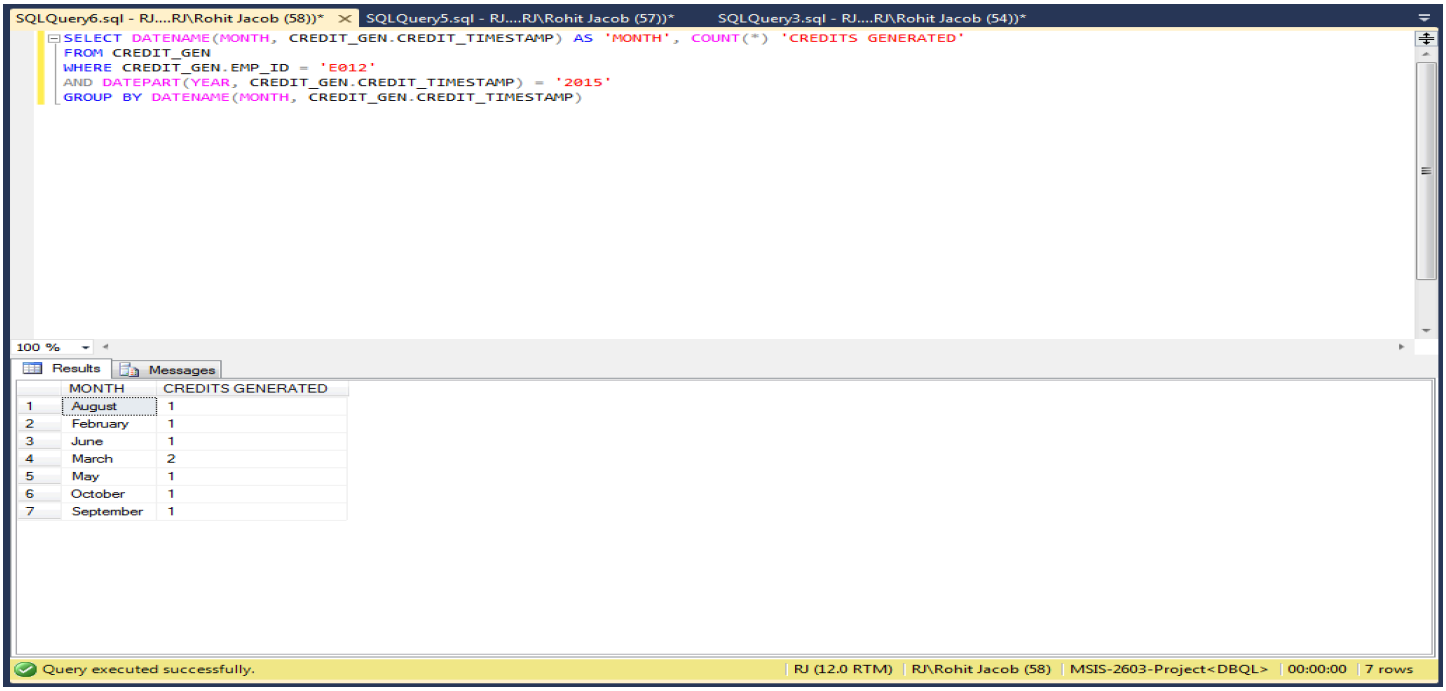
```
SELECT ACCOUNT.BRANCH_ID,
CAST(ROUND(AVG(ACCOUNT.ACC_BALANCE), 10,2) AS DECIMAL(18,2)) AS
'AVERAGE ACCOUNT BALANCES'
FROM ACCOUNT
GROUP BY ACCOUNT.BRANCH_ID;
```

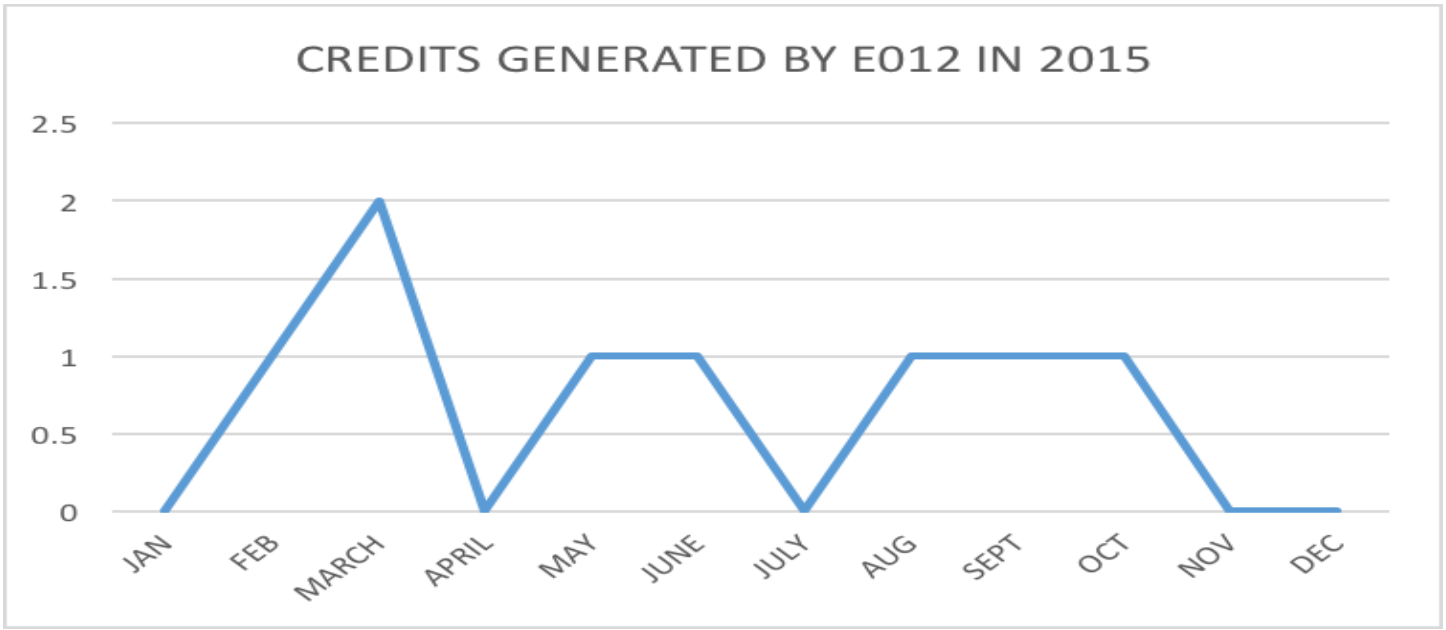




6. Credit Analyst KPI – Credit Generated

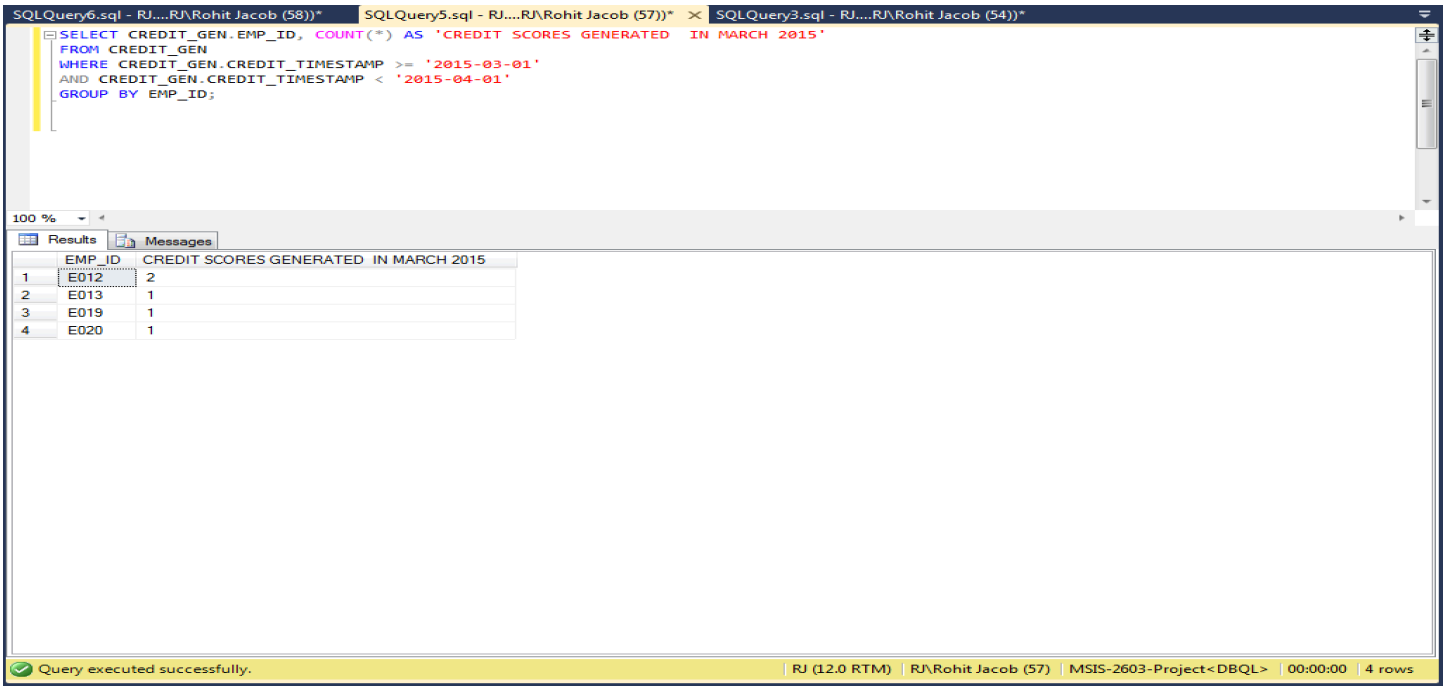
```
SELECT DATENAME(MONTH, CREDIT_GEN.CREDIT_TIMESTAMP) AS 'MONTH',
COUNT(*) 'CREDITS GENERATED'
FROM CREDIT_GEN
WHERE CREDIT_GEN.EMP_ID = 'E012'
AND DATEPART(YEAR, CREDIT_GEN.CREDIT_TIMESTAMP) = '2015'
GROUP BY DATENAME(MONTH, CREDIT_GEN.GEN_TIMESTAMP)
```

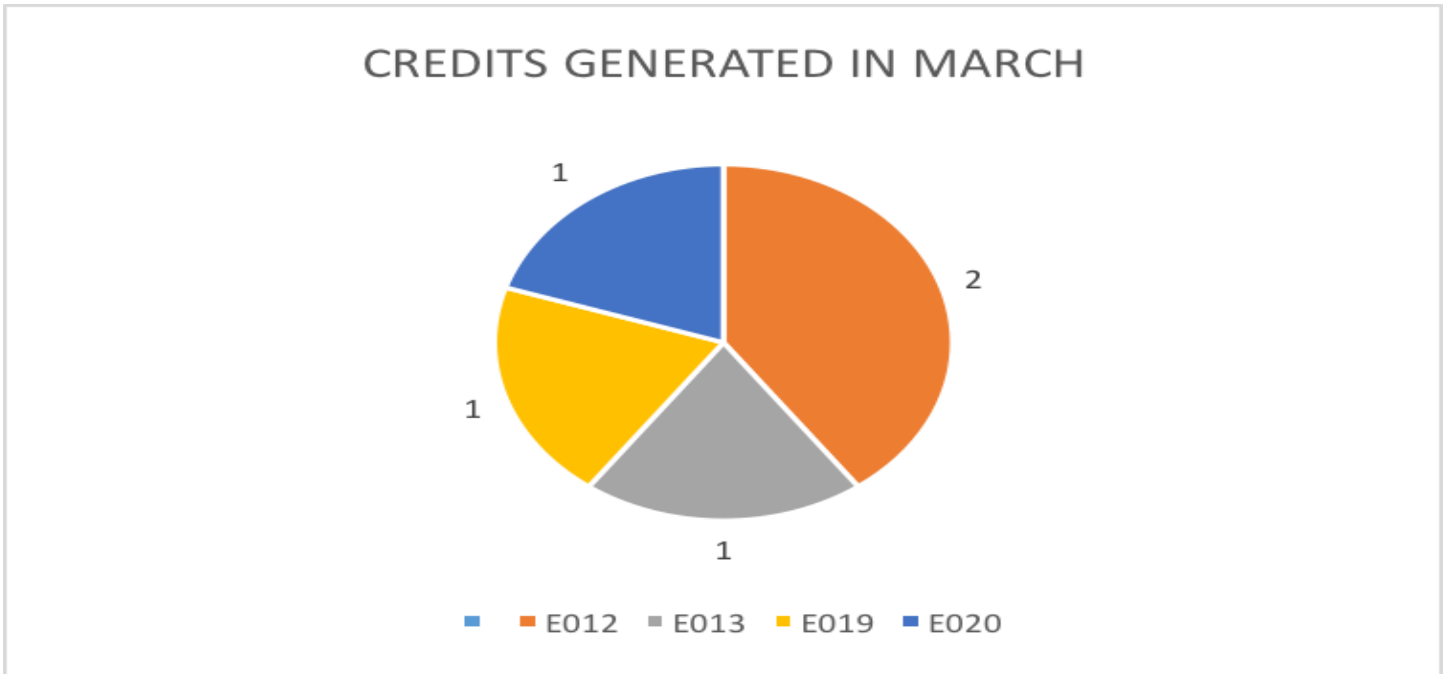




7. Credit Scores Generated

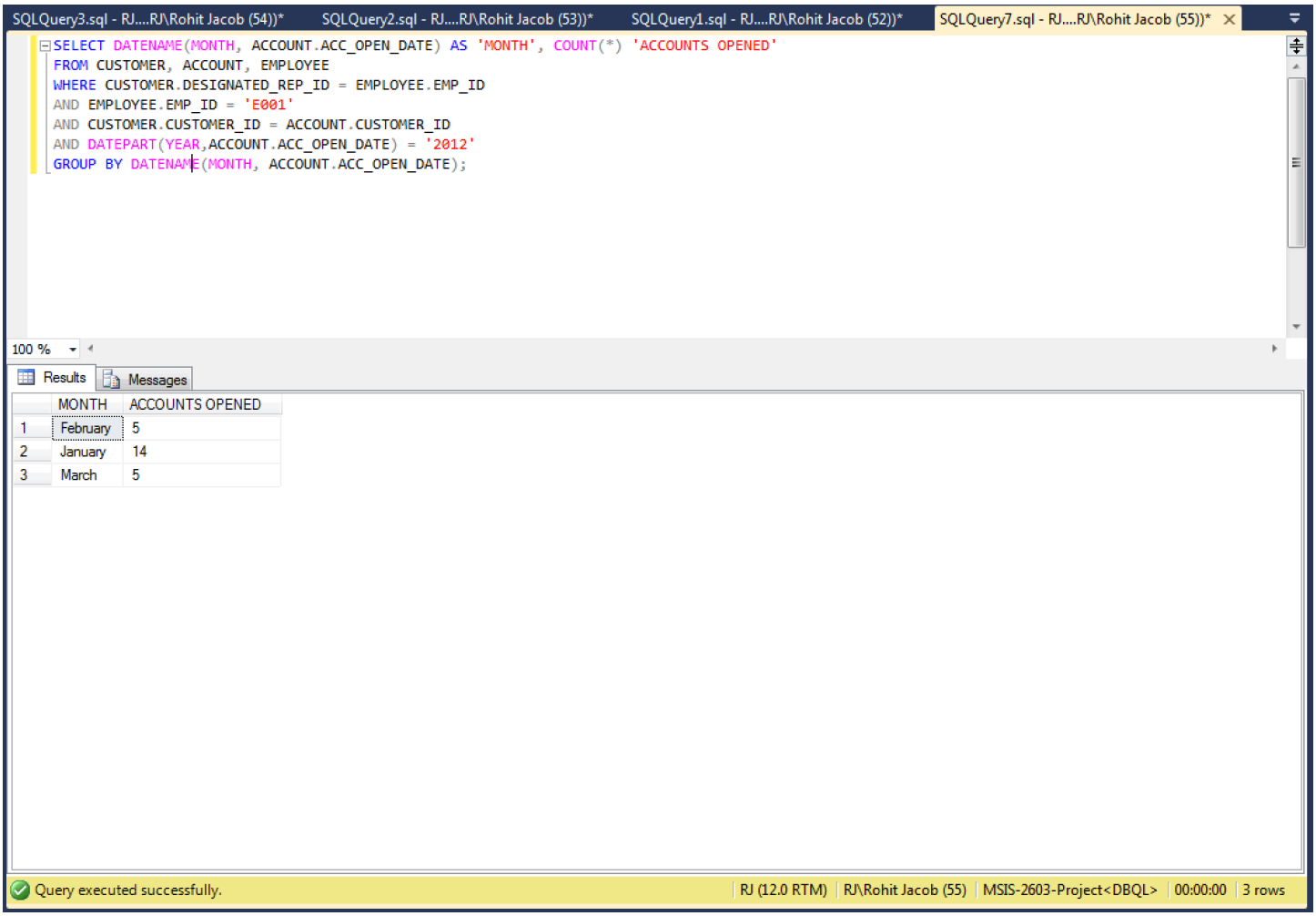
```
SELECT CREDIT_GEN.EMP_ID, COUNT(*) AS 'CREDIT SCORES GENERATED IN
MARCH 2015'
FROM CREDIT_GEN
WHERE CREDIT_GEN.CREDIT_TIMESTAMP >= '2015-03-01'
AND CREDIT_GEN.CREDIT_TIMESTAMP < '2015-04-01'
GROUP BY EMP_ID;
```





8. Accounts Opened By Employee E001

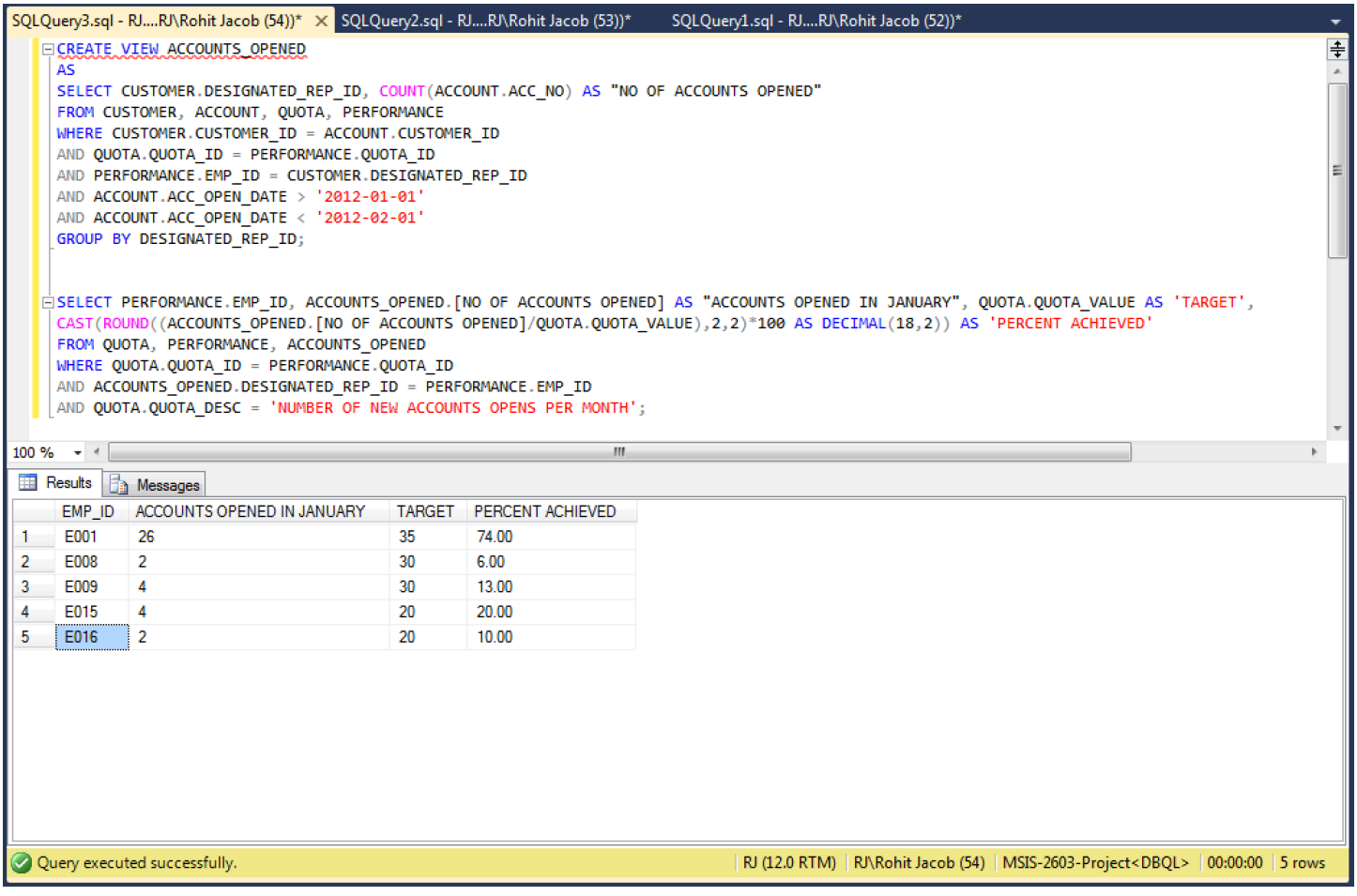
```
SELECT DATENAME(MONTH, ACCOUNT.ACC_OPEN_DATE) AS 'MONTH', COUNT(*)
'ACCOUNTS OPENED'
FROM CUSTOMER, ACCOUNT, EMPLOYEE
WHERE CUSTOMER.DESIGNATED_REP_ID = EMPLOYEE.EMP_ID
AND EMPLOYEE.EMP_ID = 'E001'
AND CUSTOMER.CUSTOMER_ID = ACCOUNT.CUSTOMER_ID
AND DATEPART(YEAR, ACCOUNT.ACC_OPEN_DATE) = '2012'
GROUP BY DATENAME(MONTH, ACCOUNT.ACC_OPEN_DATE);
```



9. Employee KPI

```
CREATE VIEW ACCOUNTS_OPENED
AS
SELECT CUSTOMER.DESIGNATED_REP_ID, COUNT(ACCOUNT.ACC_NO) AS “NO OF
ACCOUNTS OPENED”
FROM CUSTOMER, ACCOUNT, QUOTA, PERFORMANCE
WHERE CUSTOMER.CUSTOMER_ID = ACCOUNT.CUSTOMER_ID
AND QUOTA.QUOTA_ID = PERFORMANCE.QUOTA_ID
AND PERFORMANCE.EMP_ID = CUSTOMER.DESIGNATED_REP_ID
AND ACCOUNT.ACC_OPEN_DATE > ‘2012-01-01’
AND ACCOUNT.ACC_OPEN_DATE < ‘2012-02-01’
GROUP BY DESIGNATED_REP_ID;

SELECT PERFORMANCE.EMP_ID, ACCOUNTS_OPENED.[NO OF ACCOUNTS OPENED] AS
“ACCOUNTS OPENED IN JANUARY”, QUOTA.QUOTA_VALUE AS ‘TARGET’,
CAST(ROUND(ACCOUNTS_OPENED.[NO OF ACCOUNTS
OPENED]/QUOTA.QUOTA_VALUE),2,2)*100 AS DECIAML(18,2)) AS ‘PERCENT
ACHIEVED’
FROM QUOTA, PERFORMANCE, ACCOUNT_OPENED
WHERE QUOTA.QUOTA_ID = PERFORMANCE.QUOTA_ID
AS ACCOUNTS_OPENED.DESIGNATED_REP_ID = PERFORMANCE.EMP_ID
AND QUOTA.QUOTA_DESC = ‘NUMBER OF NEW ACCOUNTS OPENS PER MONTH’;
```



Project Summary

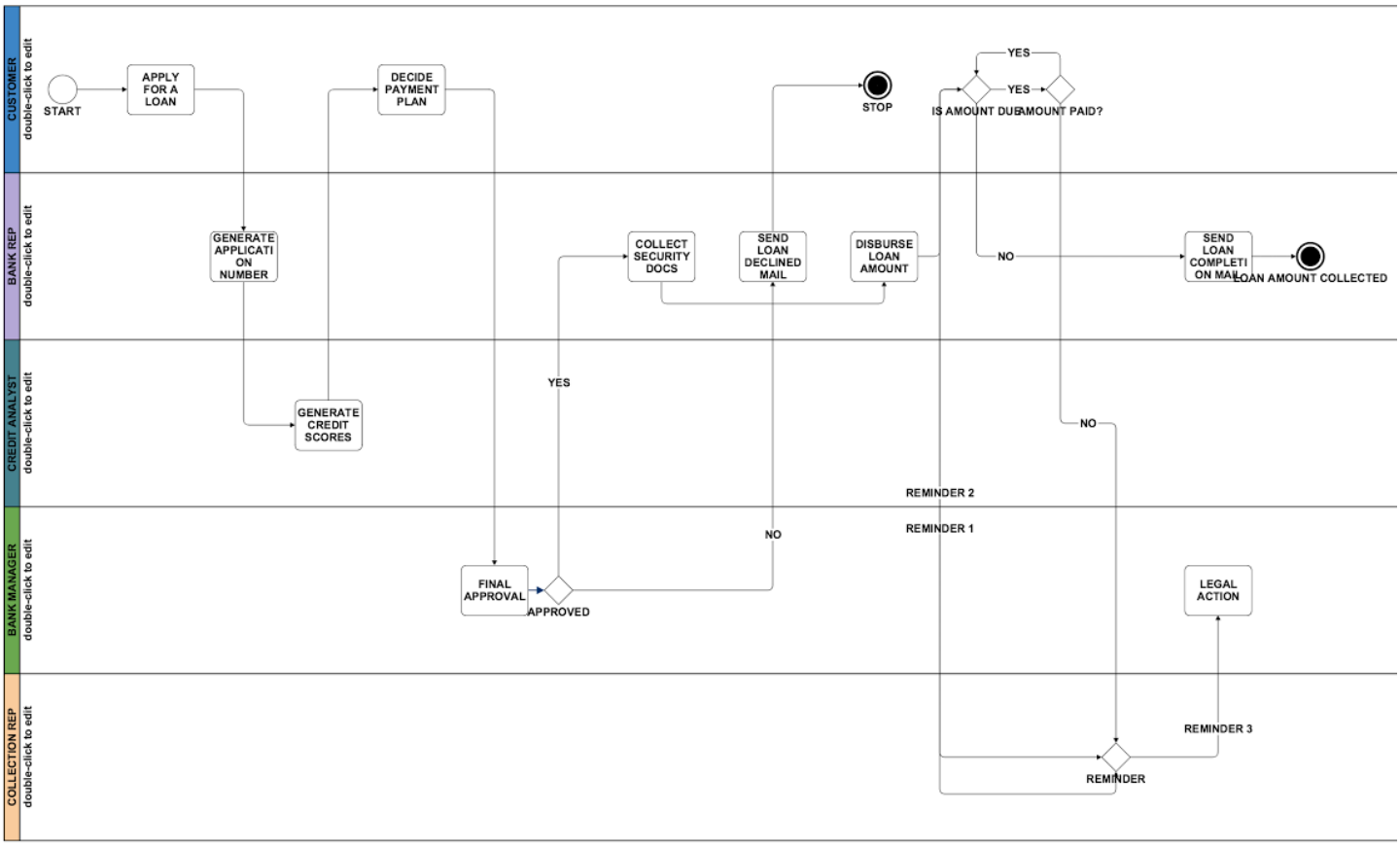
Our project started with a single branch Bank that provides only education loans to students. This project is evolved to a complex database schema of a real-life bank with multiple branches that provides more than 3 types of loans products to customer. We have created different KPIs for each employee and also created a highly scalable database design to support more branches, loans, customers, customer tier, employees, managers, designations and so on.

To provide a unique twist to our bank, we have provided the following differentiators to other competitive banks. These differentiators will help us cater to a larger set of customer's base and therefore increase revenue.

1. Customer has the option to choose between different types of payment plans
2. Customers with a lower credit score can get a loan with one of the following options
    a. Higher percentage of loan interest
    b. Higher percentage of down payment
    c. Security options like Property, Vehicle, Fixed Deposits or Government Bonds.
3. Customer has a model to generate his own quote online to see his eligibility for a particular loan.
4. Customer gets to see the real time status of the loan from the time of generating a quote to completing re-payment of his loan amount.
5. Bank Representatives have the option of proposing a counter-offer to ineligible customers before initiating a process
6. SOX Compliant!

The hardest part of our project was finalizing the process-flow of the loan-origination and loan-collection process and therefore designing the UML diagram of our database schema. We had to make sure that we adopt an optimum process that helps us design an agile, easily understandable and scalable UML Model.

By researching on the real world processes and with multiple discussions, we finalized the final process to be as shown in the below diagram.



Challenges and Solutions

We ran into multiple issues in the physical and logical design of the project. Finally, we had some issues with the pulling out the right information for our bank. We have mentioned them below along with the steps taken by us to resolve it.

- ***Challenge:*** Most of our employees have only one type of target except for the Bank Rep. The Bank Rep was responsible for two targets (number of new customers and amount of loan generated). Here, we weren't able to address the "many-to-many" relationship ambiguity.

  Solution: We resolved it by dividing the TARGET table into PERFORMANCE and QUOTA. Here we were able to point a particular employee type to any number of targets and monitor their performance more easily.

- ***Challenge:*** We wanted to provide loan to our customers based on their current credit scores and their credit history. While the current credit score is important, a credit history view will give the customers trends of paying his past debts or bills. Also, tracking the loan amount disbursed was a challenge.

  Solution: We included a timestamp and started tracking each of them as a separate transaction all pointing to one single customer, therefore we were able to track the credit history allocated in the past to a particular customer. The timestamp solution was used for storing the account transaction history and also helped solve our challenge with tracking loan amount disbursed.

- ***Challenge:*** We wanted to have a scalable solution for adding new loan products and new payment options for customer. However, our limited knowledge in the banking, loans and financial analysis restricted us from creating complex products or payment plans.

  Solution: We researched by applying for different loans on real-world banks and created 4 primary types of loan products. We were able to scale our payment plans to 24 different types bifurcating it to advanced and basic types for a specific period. E.g.: Quarterly Basic, Annually Advanced and so on.

- ***Challenge:*** As we wanted to be a self-sustaining bank (branches), we need to know the ratio between the amount of savings and the amount disbursed as loan. As our customers were also paying back the existing loans, we were unable to calculate the total current assets of the branch.

  Solution: We separated the "AMOUNT PAID" and "AMOUNT PENDING" as separate tables. With this, we were not only able to overcome our original challenge but we were now also able to track the "Loan Collection Agents" KPI.

- ***Challenge:*** We want provide our customer with real time update about their loan status, which require automatic generated log.

  Solution: We use trigger to generate loan status log together with timestamp and incremental ID. For each time new loan inserted to our system, automatically a new line will insert to our loan status log with their loan identification, status, timestamp and log ID. For each time any updates happened to our loan status, automatically another new line will insert to our loan status log with their loan identification, status, timestamp and log ID.

Recommendations and if We were to do this Again

If we were to do this project again we would first start by researching on the industry and finalize on our solution. Our project steadily oscillated between the physical design and the conceptual design. This caused multiple delays and restructuring of dependent tables with foreign keys and primary keys. If we were to do this project again, we will strictly restrict ourselves to the Conceptual – Logical – Physical design. While we understand that this may be an ongoing circle in the real-world, we will be able minimize database development time.

Finally, we would like to recommend to the next class that they look for the "unique twist" in their projects as opposed to the standard processes. We compelled ourselves to introduce a uniqueness to our solution. This forced us to research and develop deeper into the solution thus giving us a strong understanding of the loan process and DBMS.
