

DBQL BANK

MSIS – 2603 : Database Management Systems

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

SANTA CLARA UNIVERSITY

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 PENDING_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

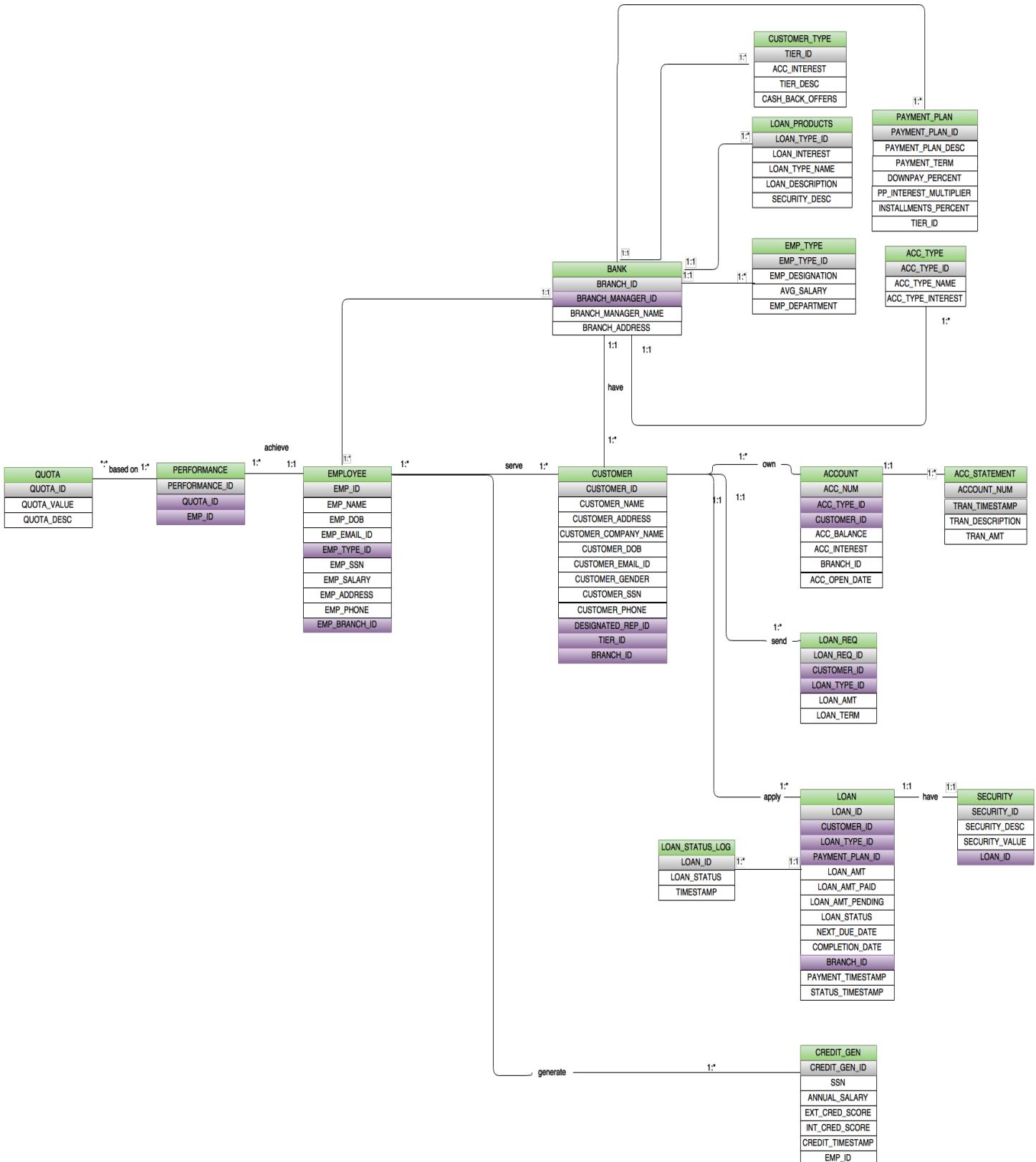
4. Collection Agent

- 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

- 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'
```

The screenshot shows the Microsoft SQL Server Management Studio interface. The query window contains the SQL code for inserting data into the CUSTOMER and ACCOUNT tables. The results pane displays the inserted data:

CUSTOMER_ID	CUSTOMER_NAME	ACC_NO	ACC_TYPE_ID	CUSTOMER_ADDRESS	CUSTOMER_COMPANY_NA...	CUSTOMER_D...
C031	renu sharma	371366140170058	AT001	9406 Shady Promenade,Ah Fong Village,CA	Santa Clara University	1965-12-02
C031	renu sharma	371366140170059	AT002	9406 Shady Promenade,Ah Fong Village,CA	Santa Clara University	1965-12-02

The status bar at the bottom indicates "Query executed successfully." and shows the session details: JESSEZ\SQLEXPRESS (12.0 RTM) | JESSEZ\Jesse (56) | MSIS 2603 Project<DBQL> | 00:00:00 | 2 rows.

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'

```

The screenshot shows the Microsoft SQL Server Management Studio interface. The query window displays the following SQL code:

```

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

```

The results pane shows the output of the query:

LOAN_ID	CUSTOMER_ID	LOAN_TYPE_ID	PAYMENT_PLAN_ID	LOAN_AMT	LOAN_AMT_PAID	LOAN_AMT_PENDING	LOAN_STATUS	NEXT_DUE_DATE
L021	C011	LT004	PP006	260000	100000	160000	INSTLM DUE	2015-11-25
L036	C011	LT002	PP005	16232	0	16232	APPLICATION COMPLETED	NULL

The status bar at the bottom indicates "Query executed successfully." The properties pane on the right shows connection details for the session.

c. Change customer C001's L001 status to PENDING_DOCUMENT

```

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

```

The screenshot shows the Microsoft SQL Server Management Studio interface. The title bar reads "SQLQuery2.sql - JESSEZ\SQLEXPRESS.MSIS-2603-Project<DBQL> (JESSEZ\Jesse (56)) - Microsoft SQL Server Management Studio". The main window contains a query editor with the following code:

```

1 UPDATE L SET L.LOAN_STATUS='PENDING_DOCUMENT'
2 FROM LOAN L
3 WHERE L.LOAN_ID='L001'
4 SELECT *
5 FROM LOAN L
6 WHERE L.LOAN_ID='L001'

```

The results pane shows a single row of data from the LOAN table:

LOAN_ID	CUSTOMER_ID	LOAN_TYPE_ID	PAYMENT_PLAN_ID	LOAN_AMT_PAID	LOAN_AMT_PENDING	LOAN_STATUS	NEXT_DUE_DATE	COMMENTS
L001	C001	LT001	PP013	200000	0	PENDING_DOCUMENT	NULL	NULL

The status bar at the bottom indicates "Query executed successfully." and "JESSEZ\SQLEXPRESS (12.0 RTM) | JESSEZ\Jesse (56) MSIS-2603-Project<DBQL> 00:00:00 1 rows".

- 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')

```

The screenshot shows the Microsoft SQL Server Management Studio interface. The query window displays the following code:

```

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')

```

The results grid shows 24 rows of payment plan data, with columns including PAYMENT_PLAN_ID, PAYMENT_PLAN_DESC, PAYMENT_TERM, PP_INTEREST_MULTIPLI..., DOWNPAY_PERCE..., INSTALLMENTS_PERCE..., and TIER_ID. The data includes various payment types like MONTHLY BASIC, QUARTERLY BASIC, SEMI YEARLY BASIC, etc., across different tiers (T001, T002).

PAYMENT_PLAN_ID	PAYMENT_PLAN_DESC	PAYMENT_TERM	PP_INTEREST_MULTIPLI...	DOWNPAY_PERCE...	INSTALLMENTS_PERCE...	TIER_ID
1	MONTHLY BASIC	5	1	0.1	0.275	T001
2	QUARTERLY BASIC	5	1	0.1	0.285	T001
3	SEMI YEARLY BASIC	5	1	0.1	0.25	T001
4	YEARLY BASIC	5	1	0.1	0.21	T001
5	MONTHLY BASIC	5	1	0.1	0.175	T001
6	QUARTERLY BASIC	10	1	0.1	0.185	T001
7	SEMI YEARLY BASIC	10	1	0.1	0.15	T001
8	YEARLY BASIC	10	1	0.1	0.19	T001
9	MONTHLY BASIC	15	1	0.1	0.075	T001
10	QUARTERLY BASIC	15	1	0.1	0.085	T001
11	SEMI YEARLY BASIC	15	1	0.1	0.05	T001
12	YEARLY BASIC	15	1	0.1	0.095	T001
13	MONTHLY ADVANCED	5	0.5	0.3	0.275	T002
14	QUARTERLY ADVANCED	5	0.5	0.1	0.285	T002
15	SEMI YEARLY ADVANCED	5	0.5	0.1	0.25	T002
16	YEARLY ADVANCED	5	0.5	0.1	0.21	T002
17	MONTHLY ADVANCED	5	0.5	0.1	0.175	T002
18	QUARTERLY ADVANCED	10	0.5	0.1	0.185	T002
19	SEMI YEARLY ADVANCED	10	0.5	0.1	0.15	T002
20	YEARLY ADVANCED	10	0.5	0.1	0.19	T002
21	MONTHLY ADVANCED	15	0.5	0.1	0.075	T002
22	QUARTERLY ADVANCED	15	0.5	0.1	0.085	T002
23	SEMI YEARLY ADVANCED	15	0.5	0.1	0.05	T002
24	YEARLY ADVANCED	15	0.5	0.1	0.095	T002

The status bar at the bottom indicates "Query executed successfully." and shows connection details: JESSEZ\SQLEXPRESS (12.0 RTM) | JESSEZ\Jesse (54) | MSIS-2603 Project<DBQL> | 00:00:00 | 24 rows.

2. Branch Manager

- 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'
```

The screenshot shows the Microsoft SQL Server Management Studio interface. The query window displays the following results:

	CUSTOMER_ID	CUSTOMER_NAME	TIER_ID	TIER_DESC	BRANCH_ID
1	C001	Shailesh Agarwal	T002	PREMIUM_CUSTOMER	CA001
2	C004	Warren Buffett	T002	PREMIUM_CUSTOMER	CA001

The Properties pane on the right shows connection details:

- Connection name: JESSEZ\SQLEXPRESS (JESSEZ\Jesse)
- Connection elapsed time: 00:00:00.029
- Connection finish time: 11/30/2015 14:14:04
- Rows returned: 2
- Start time: 11/30/2015 14:14:04
- State: Open
- Server version: 12.0.2269
- Session Tracing ID: 55

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'
```

The screenshot shows the Microsoft SQL Server Management Studio interface. On the left is the Object Explorer pane, which lists the database 'MSIS-2603 Project<DBQL>' under 'JESSEZ\SQLEXPRESS (SQL Server 12.0.226)'. In the center is the SQL Query window titled 'SQLQuery1.sql - JES... (JESSEZ\Jesse (55))' containing the provided SQL code. To the right is the Properties pane, which displays connection parameters like 'Connection name: JESSEZ\SQLEXPRESS (JESSEZ\Jesse)' and 'Connection state: Open'. Below the query window is the Results pane, which shows the output of the UPDATE statement. The results table has columns: LOAN_ID, CUSTOMER_ID, LOAN_TYPE_ID, PAYMENT_PLAN_ID, LOAN_AMOUNT_PAID, LOAN_AMT_PENDING, LOAN_STATUS, NEXT_DUE_DATE, and COMPLETION_DATE. One row is shown with values: L006, C002, LT002, PP017, 1250000, 0, APPROVED, NULL, NULL. At the bottom of the interface, a status bar indicates 'Query executed successfully.' and the session details 'JESSEZ\SQLEXPRESS (12.0 RTM) | JESSEZ\Jesse (55) MSIS 2603 Project<DBQL> 00:00:00 1 rows'.

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'
```

The screenshot shows the Microsoft SQL Server Management Studio interface. On the left is the Object Explorer pane, which lists the database structure including tables like ACC_ASSIGNMENT, ACC_TYPE, ACCOUNT, BANK, CREDIT_GEN, CUSTOMER, EMPLOYEE, LOAN, LOAN_PRODUCTS, LOAN_REQ, PAYMENT_PLAN, PERFORMANCE, QUOTA, and SECURITY. In the center is the SQL Query window titled 'SQLQuery1.sql - JESSEZ\SQLEXPRESS MSIS_2603 Project<DBQL> - Microsoft SQL Server Management Studio'. The query is:

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

The results pane shows the output of the query:

QUOTA_ID	QUOTA_VALUE	QUOTA_DESC	PERFORMANCE_ID	EMP_ID
Q003	35	NUMBER OF NEW ACCOUNTS OPENS PER MONTH	P001	E001
Q006	40000	TOTAL AMOUNT OF LOAN DISBURSED PER MONTH	P002	E001

The status bar at the bottom indicates 'Query executed successfully.' and shows the session details: JESSEZ\SQLEXPRESS (12.0 RTM) | JESSEZ\Jesse (55) | MSIS_2603 Project<DBQL> | 00:00:00 | 2 rows. The properties pane on the right displays connection parameters such as Connection name (JESSEZ\SQLEXPRESS), Start time (11/30/2015 14:24:10), and State (Open).

3. Credit Analyst

- 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
```

The screenshot shows the Microsoft SQL Server Management Studio interface. The query window contains the following T-SQL code:

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

The results pane shows the output of the query:

	CUSTOMER_ID	INT_CRED_SCORE	CREDIT_TIMESTAMP	SSN
1	C025	82	2015-11-30 13:20:07.637	403-21-5433
2	C025	92	2015-11-30 13:29:27.780	403-21-5433

The status bar at the bottom indicates "Query executed successfully." and shows the connection details: JESSEZ\SQLEXPRESS (12.0 RTM) | JESSEZ\Jesse (56) | MSIS 2603 Project<DBQL> | 00:00:00 | 2 rows.

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
```

The screenshot shows the Microsoft SQL Server Management Studio interface. The query window displays the following SQL code:

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

The results pane shows the output of the query, listing 28 rows of customer information. The columns are CUSTOMER_ID, CUSTOMER_NAME, and INT_CRED_SCORE. The data includes entries like Carlos Slim Helu (ID C003, Score 85), Forrest Mars Jr (ID C023, Score 85), Bill Gates (ID C002, Score 86), and others.

CUSTOMER_ID	CUSTOMER_NAME	INT_CRED_SCORE
C003	Carlos Slim Helu	85
C023	Forrest Mars Jr	85
C002	Bill Gates	86
C003	Carlos Slim Helu	87
C005	Amando Ortega	87
C015	Michael Bloomberg	87
C002	Bill Gates	88
C005	Amando Ortega	88
C021	Sergey Brin	88
C001	Shailesh Agarwal	90
C003	Carlos Slim Helu	90
C005	Amando Ortega	90
C020	Larry Page	90
C002	Bill Gates	92
C004	Warren Buffett	92
C006	Larry Ellison	92
C025	John Mars	92
C001	Shailesh Agarwal	95

The status bar at the bottom indicates "Query executed successfully." The properties pane on the right shows connection details for the current session.

- 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

```

The screenshot shows the Microsoft SQL Server Management Studio interface. The left pane displays the Object Explorer with the database structure. The center pane contains the SQL query being executed:

```

1 UPDATE L SET L.LOAN_STATUS='CREDIT_GEN_COMPLETED'
2 FROM LOAN L
3 WHERE L.LOAN_ID='L002'
4 INSERT INTO CREDIT_GEN (CREDIT_GEN_ID,SSN, ANNUAL_SALARY,
5 EXT_CRED_SCORE, INT_CRED_SCORE, CREDIT_TIMESTAMP,EMP_ID)
6 VALUES ('CR037', '677-26-7813', '1200000', '815', '97',
7 CURRENT_TIMESTAMP, 'E005')
8 SELECT L.LOAN_ID,L.LOAN_STATUS
9 FROM LOAN L
10 WHERE L.LOAN_ID='L002'
11 SELECT DISTINCT *
12 FROM CREDIT_GEN CG,CUSTOMER C
13 WHERE CG.CREDIT_GEN_ID='CR037'AND CG.SSN =C.CUSTOMER_SSN
14

```

The right pane shows the Properties window and the Results pane. The Results pane displays two tables: a temporary table with the updated loan status and a permanent table with the inserted credit generation record.

LOAN_ID	LOAN_STATUS
L002	CREDIT_GEN_COMPLETED

CREDIT_GEN_ID	SSN	ANNUAL_SALARY	EXT_CRED_SCORE	INT_CRED_SCORE	CREDIT_TIMESTAMP	EMP_ID	CUSTOMER_ID	CUSTOMER_NAME	CU
CR037	677-26-7813	1200000	815	97	2015-11-30 13:55:183	E005	C001	Shalitsh Agarwal	1C

Message bar: Query executed successfully.

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

```

INSERT INTO
CREDIT_GEN(CREDIT_GEN_ID,SSN,ANNUAL_SALARY,EXT_CRED_SCORE,INT_CRED_SCORE,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

```

The screenshot shows the Microsoft SQL Server Management Studio interface. The left pane displays the Object Explorer with the database structure. The center pane contains the SQL query being executed:

```

1 | INSERT INTO CREDIT_GEN(CREDIT_GEN_ID,SSN,ANNUAL_SALARY,EXT_CRED_SCORE,INT_CRED_SCORE,CREDIT_TIMESTAMP,EMP_ID)
2 | VALUES('CR037','813-47-8781',790000,870,88,CURRENT_TIMESTAMP,'E013')
3 | SELECT *
4 | FROM CREDIT_GEN CG, CUSTOMER C
5 | WHERE CG.SSN=C.CUSTOMER_SSN AND C.CUSTOMER_ID='C021'
6 | ORDER BY CG.CREDIT_TIMESTAMP

```

The right pane shows the Properties window and the Results pane. The Results pane displays the output of the query, showing two rows inserted into the CREDIT_GEN table:

	CREDIT_GEN_ID	SSN	ANNUAL_SALARY	EXT_CRED_SCORE	INT_CRED_SCORE	CREDIT_TIMESTAMP	EMP_ID	CUSTOMER_ID	CUSTOMER_NAME
1	CR031	813-47-8781	770000	860	88	2015-08-21 11:25:36.000	E013	C021	Sergey Brin
2	CR037	813-47-8781	790000	870	88	2015-11-30 14:07:26.290	E013	C021	Sergey Brin

The status bar at the bottom indicates "Query executed successfully." and shows the session details: JESSEZ\SQLEXPRESS (12.0 RTM) / JESSEZ\Jesse (55) MSIS 2603 Project<DBQL> 00:00:00 2 rows.

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'
```

The screenshot shows the Microsoft SQL Server Management Studio interface. The Object Explorer on the left displays the database structure, including tables like MSIS_ASSIGNMENT_5, MSIS_ASSIGNMENT_6, MSIS_PROJECT_DBQL, and several CUSTOMER and EMPLOYEE tables. The central pane shows a query window with the following content:

```
1 UPDATE L SET L.LOAN_STATUS= 'LEGAL_ACTION'  
2 FROM LOAN L  
3 WHERE L.LOAN_ID='L033'  
4 SELECT L.LOAN_ID,L.LOAN_STATUS  
5 FROM LOAN L  
6 WHERE L.LOAN_ID='L033'
```

The results pane shows a single row returned by the query:

LOAN_ID	LOAN_STATUS
L033	LEGAL_ACTION

The Properties pane on the right provides connection details:

- Current connection parameters:
 - Aggregate Status: Connection failures
 - Elapsed time: 00:00:00.041
 - Finish time: 12/2/2015 13:41:32
 - Name: JESSEZ\SQLEXPRESS
 - Rows returned: 1
 - Start time: 12/2/2015 13:41:32
 - State: Open
- Connection:
 - Connection name: JESSEZ\SQLEXPRESS (JESSEZ\Jesse)
- Connection Details:
 - Connection elapsed: 00:00:00.041
 - Connection finish: 12/2/2015 13:41:32
 - Connection rows: 1
 - Connection start: 12/2/2015 13:41:32
 - Connection state: Open
 - Display name: JESSEZ\SQL EXPRESS
 - Login name: JESSEZ\Jesse
 - Server name: JESSEZ\SQLEXPRESS
 - Server version: 12.0.2269
 - Session Tracing ID: 56
 - SPID: 56

The status bar at the bottom indicates "Query executed successfully." and shows the session details: JESSEZ\SQLEXPRESS (12.0 RTM) | JESSEZ\Jesse (56) | MSIS-2603 Project<DBQL> | 00:00:00 | 1 rows.

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'
```

The screenshot shows the Microsoft SQL Server Management Studio interface. The query window displays the following SQL code:

```
1 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'  
2 FROM ACC_STATEMENT AC, ACCOUNT A, ACC_TYPE AT  
3 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'
```

The results pane shows one row of data:

ACCOUNT NUMBER	ACCOUNT TYPE	TRANSACTION TIME	TRANSACTION DESCRIPTION	TRANSACTION AMOUNT	CURRENT BALANCE
34410287440089	CHECKING	2015-05-19 03:42:34.000	GROCERY	356.55	940028.32

The status bar at the bottom indicates "Query executed successfully." and shows the connection details: JESSEZ\SQLEXPRESS (12.0 RTM) JESSEZ\Jesse (54) MSIS-2603 Project<DBQL> 00:00:00 1 rows. The system tray shows various icons including the Start button, taskbar buttons, and system icons.

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
```

The screenshot shows the Microsoft SQL Server Management Studio interface. The query window contains the following SQL code:

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

The results pane displays the following table data:

	LOAN_REQ_ID	CUSTOMER_ID	LOAN_TYPE_ID	LOAN_AMT	LOAN_TERM
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
15	LQ015	C012	LT004	24497	15
16	LQ016	C004	LT002	45000	5

The status bar at the bottom indicates "Query executed successfully." and "JESSEZ\SQLEXPRESS (12.0 RTM) JESSEZ\Jesse (56) MSIS-2603-Project<DBQL> 00:00:00 16 rows".

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'

```

The screenshot shows the Microsoft SQL Server Management Studio interface. The query window contains the following SQL code:

```

1 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'
2 FROM LOAN L, CUSTOMER C, LOAN_PRODUCTS LP
3 WHERE L.CUSTOMER_ID=C.CUSTOMER_ID AND L.LOAN_TYPE_ID=LP.LOAN_TYPE_ID
4 AND C.CUSTOMER_ID='C004'

```

The results pane displays the following data:

	NAME	CUSTOMER ...	LOAN ...	LOAN TYPE	LOAN STATUS	STATUS TIMESTAMP	LOAN AMOU...	LOAN PA...	NEXT DUE DA...
1	Warren Buffett	C004	L010	VEHICLE	PENDING DOCUMENT	2015-10-11 21:47:15.000	350000	0	NULL
2	Warren Buffett	C004	L011	HOUSE	DISBURSED	2015-11-11 21:47:15.000	1150000	0	2015-12-21
3	Warren Buffett	C004	L012	EDUCATION	DISBURSED	2015-11-10 21:47:15.000	560000	0	2015-11-25

The status bar at the bottom indicates "Query executed successfully." and "3 rows". The properties pane on the right shows connection details for "JESSEZ\SQLEXPRESS (JESSEZ\Jesse)".

- 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'
```

The screenshot shows the Microsoft SQL Server Management Studio interface. The Object Explorer on the left lists various database objects like tables, stored procedures, and triggers. The central pane displays the T-SQL code for updating a customer's address. The Results pane below shows the output of the query, which successfully updated the address for customer C001. The Properties pane on the right provides connection details, and the status bar at the bottom indicates the query was executed successfully with 1 row affected.

```
1 UPDATE C SET C.CUSTOMER_ADDRESS = '1050 Benton St, Santa Clara, CA'
2 FROM CUSTOMER C
3 WHERE C.CUSTOMER_ID='C001'
4 SELECT C.CUSTOMER_ID, C.CUSTOMER_ADDRESS
5 FROM CUSTOMER C
6 WHERE C.CUSTOMER_ID='C001' |
```

CUSTOMER_ID	CUSTOMER_ADDRESS
C001	1050 Benton St, Santa Clara, CA

Query executed successfully.

JESSEZ\SQLEXPRESS (12.0 RTM) JESSEZ\Jesse (56) MSIS 2603 Project<DBQL> 00:00:00 1 rows

Name
The name of the connection.

Ready

Ln 6 Col 27 Ch 27 INS

12:06 PM 11/30/2015

Delete Queries (Before and After)

The Update queries can be found in the Use Case section

1. CUSTOMER

a. Before

SQLQuery3.sql - HELEN-LAPTOP\SQLEXPRESS.MSIS-2603-Project<DBQL> (HELEN-LAPTOP\Helen (56)) - Microsoft SQL Server Management Studio

```
select *  
from customer where customer_id = 'C030'
```

Object Explorer

Results

CUSTOMER_ID	CUSTOMER_NAME	CUSTOMER_ADDRESS	CUSTOMER_COMPANY_NAME	CUSTOMER_DATE_OF_BIRTH	CUSTOMER_EMAIL_ID	CUSTOMER_GENDER	CUSTOMER_SSN	CUSTOMER_PHONE_NUMBER	DESIGNATED_CONTACT
C030	George Soros	8255 Indian Ledge,Cattaraugus,VA	SAP	1981-09-15	wendyfun@mailblocks.com	FEMALE	932-65-2778	252-498-8819	E001

Query executed successfully.

SQLQuery5.sql - HELEN-LAPTOP\SQLEXPRESS.MSIS-2603-Project<DBQL> (HELEN-LAPTOP\Helen (55)) - Microsoft SQL Server Management Studio

```
delete from customer  
where customer_id = 'C030'
```

Object Explorer

Messages

(1 row(s) affected)

Query executed successfully.

b. After

The screenshot shows the Microsoft SQL Server Management Studio (SSMS) interface. The title bar indicates the connection is to HELEN-LAPTOP\SQLEXPRESS.MSIS-2603-Project<DBQL> (HELEN-LAPTOP\Helen (56)). The main window has four tabs open: SQLQuery4.sql, SQLQuery3.sql, SQLQuery2.sql, and DBQLV2.sql. The SQLQuery3.sql tab is active, displaying a simple SELECT query:

```
select *  
from customer where customer_id = 'c030'
```

The results pane shows the following columns: CUSTOMER_ID, CUSTOMER_NAME, CUSTOMER_ADDRESS, CUSTOMER_COMPANY_NAME, CUSTOMER_DOB, CUSTOMER_EMAIL, CUSTOMER_GENDER, CUSTOMER_SSN, CUSTOMER_PHONE, and DESIGNATED_REP. A message at the bottom of the results pane states "Query executed successfully." The status bar at the bottom right shows the session ID (HELEN-LAPTOP\SQLEXPRESS (12...)), the user (HELEN-LAPTOP\Helen (56)), the database (MSIS-2603-Project<DBQL>), the duration (00:00:00), and the number of rows (0 rows). The system tray at the bottom shows various icons for file explorer, windows, and network.

2. LOAN_REQ

a. Before

The screenshot shows the Microsoft SQL Server Management Studio interface. The title bar indicates the current database is MSIS-2603-Project<DBQL>. The Object Explorer on the left shows the database structure, including the LOAN_REQ table. The central pane displays the results of a SELECT query:

```
select *  
from LOAN_REQ
```

	LOAN_REQ_ID	CUSTOMER_ID	LOAN_TYPE_ID	LOAN_AMOUNT	LOAN_TERM
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
15	LQ015	C012	LT004	24497	15

The status bar at the bottom shows "Query executed successfully." and the session details: HELEN-LAPTOP\SQLEXPRESS (12...), HELEN-LAPTOP\Helen (56), MSIS-2603-Project<DBQL>, 00:00:00, 15 rows.

SQLQuery5.sql - HELEN-LAPTOP\SQLEXPRESS.MSIS-2603-Project<DBQL> (HELEN-LAPTOP\Helen (55)) - Microsoft SQL Server Management Studio

```
delete from loan_req
where loan_req_id = 'lq015'
```

Object Explorer

- HELEN-LAPTOP\SQLEXPRESS (SQL Server)
 - 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)*

Messages

(1 row(s) affected)

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

100 %

11/30/2015 4:30 PM

b. After

SQLQuery3.sql - HELEN-LAPTOP\SQLEXPRESS.MSIS-2603-Project<DBQL> (HELEN-LAPTOP\Helen (56)) - Microsoft SQL Server Management Studio

```
select *
from LOAN_REQ
```

Object Explorer

- HELEN-LAPTOP\SQLEXPRESS (SQL Server)
 - 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)*

Results

LOAN_REQ_ID	CUSTOMER_ID	LOAN_TYPE_ID	LOAN_AMOUNT	LOAN_TERM
1	LQ001	C011	LT002	16232
2	LQ002	C003	LT004	18741
3	LQ003	C007	LT001	61817
4	LQ004	C004	LT003	92760
5	LQ005	C013	LT001	45289
6	LQ006	C006	LT003	27973
7	LQ007	C014	LT003	78696
8	LQ008	C010	LT002	21377
9	LQ009	C005	LT003	49904
10	LQ010	C002	LT002	46622
11	LQ011	C001	LT004	68086
12	LQ012	C001	LT002	32827
13	LQ013	C009	LT001	47424
14	LQ014	C008	LT001	13642

Messages

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

100 %

11/30/2015 2:42 PM

3. SECURITY

a. Before

The screenshot shows the Microsoft SQL Server Management Studio interface. The title bar indicates the connection is to HELEN-LAPTOP\SQLEXPRESS.MSIS-2603-Project<DBQL> (HELEN-LAPTOP\Helen (56)). The Object Explorer on the left shows the database structure, including the MSIS-2603-Project database. The central pane displays a query result grid for a SELECT statement from the security table. The grid has columns: SECURITY_ID, SECURITY_TYPE, SECURITY_DESC, SECURITY_VALID_DATE, and LOAN_ID. The results show various financial instruments like vehicles, bonds, and fixed income products. A status bar at the bottom shows the query was executed successfully and provides system information.

SECURITY_ID	SECURITY_TYPE	SECURITY_DESC	SECURITY_VALID_DATE	LOAN_ID
S027	VEHICLE	BRAND Lamborghini MODEL: Huracan YEAR:2015	24/3/2015	L027
S028	VEHICLE	BRAND Aston Martin MODEL: Vanquish YEAR:2014	28/2/2014	L028
S029	VEHICLE	BRAND Aston Martin MODEL: Rapide S YEAR:2013	23/7/2015	L029
S030	VEHICLE	BRAND Bugatti Veyron MODEL: Vitesse YEAR:2014	24/8/2014	L030
S031	GOVERNMENT BOND	8 YEAR US GOVERNMENT BONDS	8/1/2015	L031
S032	GOVERNMENT BOND	5 YEAR US GOVERNMENT BONDS	7/8/2015	L032
S033	GOVERNMENT BOND	5 YEAR FOREIGN GOVERNMENT BONDS	8/1/2015	L033
S034	GOVERNMENT BOND	10 YEAR FOREIGN GOVERNMENT BONDS	5/6/2015	L034
S035	GOVERNMENT BOND	20 YEAR US GOVERNMENT BONDS	6/1/2015	L035
S036	VEHICLE	BRAND Bugatti Veyron MODEL: Veyron YEAR:2014	17/4/2005	L025
S037	FIXED INCOME	FIXED DEPOSIT	4/0/2015	L026
S038	FIXED INCOME	FIXED DEPOSIT	5/4/2015	L027
S039	FIXED INCOME	FIXED DEPOSIT	26/8/2014	L028
S040	FIXED INCOME	FIXED DEPOSIT	3/2/2014	L029
S041	GOVERNMENT BOND	10 US YEAR GOVERNMENT BONDS	10/1/2015	L030

The screenshot shows the Microsoft SQL Server Management Studio (SSMS) interface. The title bar reads "SQLQuery5.sql - HELEN-LAPTOP\SQLEXPRESS.MSIS-2603-Project<DBQL> (HELEN-LAPTOP\Helen (55)) - Microsoft SQL Server Management Studio". The Object Explorer on the left shows the database structure for "HELEN-LAPTOP\SQLEXPRESS (SQL Server)". The main query window contains the following SQL code:

```
delete from security  
where security_id = 's041'
```

The status bar at the bottom indicates "Query executed successfully." and shows the system tray with various icons.

b. After

The screenshot shows the Microsoft SQL Server Management Studio interface. The title bar reads "SQLQuery6.sql - HELEN-LAPTOP\SQLEXPRESS.MSIS-2603-Project<DBQL> (HELEN-LAPTOP\Helen (56)) - Microsoft SQL Server Management Studio". The Object Explorer on the left shows the database structure for "HELEN-LAPTOP\SQLEXPRESS (SQL Server)". The main pane displays the results of a query on the "security" table:

```
select *  
from security
```

SECURITY_ID	SECURITY_TYPE	SECURITY_DESC	SECURITY_VALUE	LOAN_ID
26	VEHICLE	BRAND Lamborghini MODEL: Aventador YEAR:2014	445095	L026
27	VEHICLE	BRAND Lamborghini MODEL: Huracan YEAR:2015	248000	L027
28	VEHICLE	BRAND Aston Martin MODEL: Vanquish YEAR:2014	262845	L028
29	VEHICLE	BRAND Aston Martin MODEL: Rapide S YEAR:2013	230745	L029
30	VEHICLE	BRAND Bugatti Veyron MODEL: Vitesse YEAR:2014	2480000	L030
31	GOVERNMENT	8 YEAR US GOVERNMENT BONDS	800000	L031
32	GOVERNMENT	5 YEAR US GOVERNMENT BONDS	780000	L032
33	GOVERNMENT	5 YEAR FOREIGN GOVERNMENT BONDS	810000	L033
34	GOVERNMENT	10 YEAR FOREIGN GOVERNMENT BONDS	560000	L034
35	GOVERNMENT	20 YEAR US GOVERNMENT BONDS	610000	L035
36	VEHICLE	BRAND Bugatti Veyron MODEL: Veyron YEAR:2014	1740005	L025
37	FIXED INCOME	FIXED DEPOSIT	400000	L026
38	FIXED INCOME	FIXED DEPOSIT	543030	L027
39	FIXED INCOME	FIXED DEPOSIT	268900	L028
40	FIXED INCOME	FIXED DEPOSIT	328000	L029

Below the results, a message states "Query executed successfully." The status bar at the bottom right shows "HELEN-LAPTOP\SQLEXPRESS (12... | HELEN-LAPTOP\Helen (56) | MSIS-2603-Project<DBQL> | 000000 | 40 rows | 432 PM | 11/30/2015".

4. ACCOUNT

a. Before

The screenshot shows the Microsoft SQL Server Management Studio interface. The title bar reads "SQLQuery6.sql - HELEN-LAPTOP\SQLEXPRESS.MSIS-2603-Project<DBQL> (HELEN-LAPTOP\Helen (56)) - Microsoft SQL Server Management Studio". The Object Explorer on the left shows the database structure for "HELEN-LAPTOP\SQLEXPRESS (SQL Server)". The main pane displays the results of a query on the "account" table:

```
select *  
from account
```

ACC_NO	ACC_TYPE_ID	CUSTOMER_ID	ACC_BALANCE	ACC_INTEREST	BRANCH_ID	ACC_OPEN_DATE
46	375973245507780	AT001	979124.88	0	NY001	2012-03-16
47	37598888729814	AT002	132380.15	0.05	IL001	2012-02-15
48	375997085634991	AT002	699986.44	0.03	IL001	2012-03-19
49	376235378741464	AT002	1935747.27	0.05	NY001	2012-02-18
50	376843705086729	AT002	1468802.08	0.05	CA001	2012-02-02
51	376908467656179	AT001	751681.14	0	NY001	2013-03-08
52	376940499482316	AT001	821231.84	0	IL001	2012-01-03
53	376990781423809	AT002	1831799.07	0.05	NY001	2012-02-18
54	377040853035359	AT001	535661.55	0	NY001	2012-03-09
55	377720415994688	AT001	675491.14	0	IL001	2012-02-18
56	377846944244301	AT001	829168.16	0	NY001	2012-03-09
57	378145171119461	AT001	627986.76	0	IL001	2012-03-04
58	378407475250990	AT001	454505.13	0	CA001	2012-02-09
59	378930952392498	AT002	38601.83	0.03	NY001	2012-03-18
60	379785227842803	AT002	1386211.44	0.05	NY001	2012-03-08

Below the results, a message states "Query executed successfully." The status bar at the bottom right shows "HELEN-LAPTOP\SQLEXPRESS (12... | HELEN-LAPTOP\Helen (56) | MSIS-2603-Project<DBQL> | 00:00:00 | 60 rows | 4:34 PM | 11/30/2015".

SQLQuery5.sql - HELEN-LAPTOP\SQLEXPRESS.MSIS-2603-Project<DBQL> (HELEN-LAPTOP\Helen (55)) - Microsoft SQL Server Management Studio

```
delete from account
where acc_no = 379785227842803
```

(1 row(s) affected)

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

4:35 PM 11/30/2015

b. After

SQLQuery5.sql - HELEN-LAPTOP\SQLEXPRESS.MSIS-2603-Project<DBQL> (HELEN-LAPTOP\Helen (56)) - Microsoft SQL Server Management Studio

```
select *
from account
```

ACC_NO	ACC_TYPE_ID	CUSTOMER...	ACC_BALANCE	ACC_INTEREST	BRANCH_ID	ACC_OPEN_DATE	
45	37572206222898	AT001	C014	325918.21	0	IL001	2012-01-03
46	375973245507780	AT001	C009	979124.88	0	NY001	2012-03-16
47	37598888729814	AT002	C002	1323801.35	0.05	IL001	2012-02-15
48	375997085634891	AT002	C020	699986.44	0.03	IL001	2012-03-19
49	376235376741464	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	377040853035359	AT001	C008	535661.55	0	NY001	2012-03-09
55	377720415984688	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

(59 row(s) affected)

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

4:35 PM 11/30/2015

5. ACC_STATEMENT

a. Before

The screenshot shows the Microsoft SQL Server Management Studio interface. The Object Explorer on the left shows a connection to 'HELEN-LAPTOP\SQLEXPRESS (SQL Server)'. The 'Databases' node is expanded, showing 'System Databases', 'MSIS-2603-Project<DBQL>', and its contents: 'Database Diagrams', 'Tables', 'Views', 'Synonyms', 'Programmability', 'Service Broker', 'Storage', 'Security'. The 'Tables' node is selected. The 'Results' tab in the center displays the output of the following query:

```
select * from ACC_STATEMENT
```

The results show 29 rows of data from the ACC_STATEMENT table, with columns: ACC_NO, TRAN_TIMESTAMP, TRAN_DESC, TRAN_A..., and several other columns partially visible. The data includes various transaction details like BILL, GROCERY, GAS, MOVIE, RESTAURANT, etc., with dates ranging from 2015-04-14 to 2015-07-29.

At the bottom of the results pane, a message states: 'Query executed successfully.'

The status bar at the bottom right indicates: HELEN-LAPTOP\SQLEXPRESS (12...) | HELEN-LAPTOP\Helen (56) | MSIS-2603-Project<DBQL> | 00:00:00 | 29 rows.

The screenshot shows the Microsoft SQL Server Management Studio interface. The Object Explorer on the left shows a connection to 'HELEN-LAPTOP\SQLEXPRESS (SQL Server)'. The 'Databases' node is expanded, showing 'System Databases', 'MSIS-2603-Project<DBQL>', and its contents: 'Database Diagrams', 'Tables', 'Views', 'Synonyms', 'Programmability', 'Service Broker', 'Storage', 'Security'. The 'Tables' node is selected. The 'Messages' tab in the center displays the output of the following query:

```
delete from acc_statement  
where acc_no = 378407475250990
```

The message pane shows '(1 row(s) affected)'.

At the bottom of the messages pane, a message states: 'Query executed successfully.'

The status bar at the bottom right indicates: HELEN-LAPTOP\SQLEXPRESS (12...) | HELEN-LAPTOP\Helen (55) | MSIS-2603-Project<DBQL> | 00:00:00 | 0 rows.

6. ACC_STATEMENT

a. After

The screenshot shows the Microsoft SQL Server Management Studio interface. The title bar reads "SQLQuery6.sql - HELEN-LAPTOP\SQLEXPRESS.MSIS-2603-Project<DBQL> (HELEN-LAPTOP\Helen (56)) - Microsoft SQL Server Management Studio". The Object Explorer on the left shows the database structure for "HELEN-LAPTOP\SQLEXPRESS (SQL Server)". The Results tab in the center displays the output of the following query:

```
select * from ACC_STATEMENT
```

The results table contains 28 rows of transaction data:

ACC_NO	TRAN_TIMESTAMP	TRAN_DESC	TRAN_A...	
14	370534802444794	2015-05-16 08:50:29.000	ONLINE	469.75
15	371366140170057	2015-08-14 05:41:49.000	BILL	45.69
16	371503611001391	2015-07-15 10:29:55.000	BILL	221.95
17	372683297937727	2015-07-19 15:54:41.000	ONLINE	434.93
18	373378771347563	2015-01-18 10:57:27.000	ONLINE	611.75
19	373683022697539	2015-05-01 03:39:44.000	BILL	744.55
20	374182851494506	2015-08-04 09:29:04.000	ONLINE	571.77
21	375722062022898	2015-03-27 21:25:50.000	MOVIE	722.66
22	375973245507780	2015-07-20 05:18:46.000	GROCERY	395.8
23	376908467656179	2015-03-12 11:56:51.000	GAS	606
24	376940499482316	2015-05-16 03:24:01.000	GAS	701.54
25	377040853033539	2015-05-11 06:30:03.000	MOVIE	748.44
26	377720415994688	2015-10-23 09:28:01.000	BILL	499.39
27	377846942424301	2015-07-10 15:04:19.000	ONLINE	336.33
28	378145171119461	2015-04-10 12:53:55.000	RESTAURANT	432.47

At the bottom, a message indicates "Query executed successfully." and the status bar shows "HELEN-LAPTOP\SQLEXPRESS (12... | HELEN-LAPTOP\Helen (56) | MSIS-2603-Project<DBQL> | 00:00:00 | 28 rows".

7. LOAN

a. Before

The screenshot shows the Microsoft SQL Server Management Studio interface. The title bar reads "SQLQuery3.sql - HELEN-LAPTOP\SQLEXPRESS.MSIS-2603-Project<DBQL> (HELEN-LAPTOP\Helen (56)) - Microsoft SQL Server Management Studio". The Object Explorer on the left shows the database structure for "HELEN-LAPTOP\SQLEXPRESS (SQL Server)". The Results tab in the center displays the output of the following query:

```
select * from loan
```

The results table contains 35 rows of loan data:

LOAN_ID	CUSTOMER_ID	LOAN_TYPE_ID	PAYMENT_PLAN_ID	LOAN_AMT_P...	LOAN_AMT_PENDI...	LOAN_STATUS	NEXT_DUE_DA...	COMPLETION_DA...	BRANCH_ID	PAYMENT_TIM...		
22	L022	C012	LT001	PP007	800000	500000	300000	INSTLM_DUE	2015-11-14	2017-11-16	NY001	2015-11-01 00:
23	L023	C013	LT004	PP008	530000	180000	350000	INSTLM_DUE	2015-10-09	2022-05-16	IL001	2015-09-01 00:
24	L024	C014	LT004	PP009	570000	100000	560000	INSTLM_DUE	2015-11-16	2017-12-16	IL001	2015-10-01 00:
25	L025	C015	LT003	PP010	590000	190000	400000	INSTLM_PAID	2015-11-09	2025-04-16	NY001	2015-11-01 00:
26	L026	C016	LT001	PP011	530000	230000	300000	INSTLM_PAID	2015-11-11	2020-11-01	CA001	2015-11-09 00:
27	L027	C017	LT004	PP012	510000	100000	410000	INSTLM_PAID	2015-11-03	2019-04-01	IL001	2015-11-20 00:
28	L028	C018	LT003	PP013	480000	280000	200000	INSTLM_PAID	2015-10-29	2017-12-06	NY001	2015-11-01 00:
29	L029	C019	LT002	PP001	510000	230000	280000	INSTLM_PAID	2015-11-18	2018-01-06	CA001	2015-11-18 00:
30	L030	C020	LT003	PP002	490000	230000	260000	REMINDER 1	2015-10-17	2017-07-06	IL001	2015-09-01 00:
31	L031	C021	LT003	PP002	380000	210000	170000	REMINDER 1	2015-10-01	2020-02-15	NY001	2015-09-15 00:
32	L032	C022	LT002	PP003	880000	660000	220000	REMINDER 2	2015-09-12	2017-01-06	CA001	2015-08-01 00:
33	L033	C023	LT003	PP004	630000	230000	400000	REMINDER 2	2015-08-08	2018-01-29	IL001	2015-07-01 00:
34	L034	C024	LT003	PP005	390000	320000	70000	LEGAL ACTION	2015-05-21	2017-01-06	CA001	2015-04-01 00:
35	L035	C025	LT003	PP006	490000	200000	470000	LEGAL ACTION	2015-04-21	2018-01-01	CA001	2015-04-02 00:

At the bottom, a message indicates "Query executed successfully." and the status bar shows "HELEN-LAPTOP\SQLEXPRESS (12... | HELEN-LAPTOP\Helen (56) | MSIS-2603-Project<DBQL> | 00:00:00 | 35 rows".

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

Object Explorer

MSIS-2603-Project<DBQL>

SQLQuery5.sql - H...LAPTOP\Helen (55)* SQLQuery6.sql - H...LAPTOP\Helen (56)* DBQLV3.sql - HELE...LAPTOP\Helen (52)*

```
delete from loan
where loan_id = '1035'
```

Messages

(1 row(s) affected)

Query executed successfully.

HELEN-LAPTOP\SQLEXPRESS (12...) HELEN-LAPTOP\Helen (55) MSIS-2603-Project<DBQL> 00:00:00 0 rows

Ready

Ln 2 Col 23 Ch 23 INS

4:42 PM 11/30/2015

b. After

SQLQuery3.sql - HELEN-LAPTOP\SQLEXPRESS.master (HELEN-LAPTOP\Helen (52)) - Microsoft SQL Server Management Studio

File Edit View Query Project Debug Tools Window Help

Object Explorer

master

SQLQuery4.sql - H...LAPTOP\Helen (53)* SQLQuery3.sql - H...LAPTOP\Helen (52)* SQLQuery2.sql - not connected* DBQLV2.sql - not connected*

```
select *
from loan
```

Results Messages

LOAN_ID	CUSTOMER_ID	LOAN_TYPE_ID	PAYMENT_PLAN_ID	LOAN_AMT_REQD	LOAN_AMT_PAID	LOAN_AMT_PEND	LOAN_STATUS	NEXT_DUE_DATE	COMPLETION_DATE	BRANCH_ID	PAYMENT_TIMING	
21	L021	C011	LT004	PP006	260000	100000	160000	INSTLM_DUE	2015-11-25	2016-12-16	NY001	2015-11-01 00:
22	L022	C012	LT001	PP007	800000	500000	300000	INSTLM_DUE	2015-11-14	2017-11-16	NY001	2015-11-01 00:
23	L023	C013	LT004	PP008	530000	180000	350000	INSTLM_DUE	2015-10-09	2022-05-16	IL001	2015-09-01 00:
24	L024	C014	LT004	PP009	570000	100000	560000	INSTLM_DUE	2015-11-16	2017-12-16	IL001	2015-10-01 00:
25	L025	C015	LT003	PP010	590000	190000	400000	INSTLM_PAID	2015-11-09	2025-04-16	NY001	2015-11-01 00:
26	L026	C016	LT001	PP011	530000	230000	300000	INSTLM_PAID	2015-11-11	2020-11-01	CA001	2015-11-09 00:
27	L027	C017	LT004	PP012	510000	100000	410000	INSTLM_PAID	2015-11-03	2019-04-01	IL001	2015-11-20 00:
28	L028	C018	LT003	PP013	480000	280000	200000	INSTLM_PAID	2015-10-29	2017-12-06	NY001	2015-11-01 00:
29	L029	C019	LT002	PP001	510000	230000	280000	INSTLM_PAID	2015-11-18	2018-01-06	CA001	2015-11-18 00:
30	L031	C021	LT003	PP002	380000	210000	170000	REMINDER 1	2015-10-01	2020-02-15	NY001	2015-09-15 00:
31	L032	C022	LT002	PP003	880000	660000	220000	REMINDER 2	2015-09-12	2017-01-06	CA001	2015-08-01 00:
32	L033	C023	LT003	PP004	630000	230000	400000	REMINDER 2	2015-08-08	2018-01-29	IL001	2015-07-01 00:
33	L034	C024	LT003	PP005	390000	320000	70000	LEGAL ACTION	2015-05-21	2017-01-06	CA001	2015-04-01 00:
34	L035	C025	LT003	PP006	490000	200000	470000	LEGAL ACTION	2015-04-21	2018-01-01	CA001	2015-04-02 00:

Query executed successfully.

HELEN-LAPTOP\SQLEXPRESS (12...) HELEN-LAPTOP\Helen (52) master 00:00:00 34 rows

Ready

Ln 5 Col 1 Ch 1 INS

3:48 PM 11/30/2015

8. CREDIT_GEN

a. Before

The screenshot shows the Microsoft SQL Server Management Studio interface. In the Object Explorer, the 'HELEN-LAPTOP\SQLEXPRESS' database is selected. In the center pane, a query window displays the following SQL code:

```
select *  
from credit_gen
```

The results pane shows the following data:

CREDIT_GEN_ID	SSN	ANNUAL_SALA...	EXT_CRED_SCO...	INT_CRED_SCO...	CREDIT_TIMESTAMP	EMP_ID	
21	CR021	548-88-8310	560000	733	83	2015-02-21 09:13:23.000	E012
22	CR022	820-76-1301	280000	732	80	2015-03-21 09:25:46.000	E012
23	CR023	469-25-2633	490000	745	84	2015-02-17 09:13:23.000	E020
24	CR024	572-75-3961	780000	800	95	2015-12-11 11:15:59.000	E020
25	CR025	431-65-4999	780000	763	87	2015-08-21 13:55:46.000	E013
26	CR026	846-48-1567	700000	730	83	2015-06-21 16:28:47.000	E006
27	CR027	305-38-8280	590000	720	80	2015-08-11 13:55:36.000	E020
28	CR028	878-54-5212	280000	810	96	2015-03-25 08:25:47.000	E013
29	CR029	237-24-7774	190000	710	80	2015-09-11 14:25:46.000	E006
30	CR030	944-78-4575	590000	780	90	2015-08-21 19:14:56.000	E020
31	CR031	813-47-8781	770000	860	88	2015-08-21 11:25:36.000	E013
32	CR032	886-57-7882	890000	830	98	2015-09-01 09:35:32.000	E006
33	CR033	260-61-6435	230000	756	85	2015-03-11 15:35:57.000	E020
34	CR034	271-76-4204	590000	812	96	2015-04-03 13:26:29.000	E006
35	CR035	403-21-5433	880000	763	88	2015-09-08 07:55:32.000	E006

At the bottom of the results pane, it says "Query executed successfully." The status bar at the bottom right shows "HELEN-LAPTOP\SQLEXPRESS (12... | HELEN-LAPTOP\Helen (52) | master | 00:00:00 | 35 rows".

b. After

The screenshot shows the Microsoft SQL Server Management Studio interface. In the Object Explorer, the 'HELEN-LAPTOP\SQLEXPRESS' database is selected. In the center pane, a query window displays the following SQL code:

```
select *  
from credit_gen
```

The results pane shows the following data:

CREDIT_GEN_ID	SSN	ANNUAL_SALA...	EXT_CRED_SCO...	INT_CRED_SCO...	CREDIT_TIMESTAMP	EMP_ID	
20	CR020	646-21-9932	460000	743	84	2015-10-06 13:12:08.000	E012
21	CR021	548-88-8310	560000	733	83	2015-02-21 09:13:23.000	E012
22	CR022	820-76-1301	280000	732	80	2015-03-21 09:25:46.000	E012
23	CR023	469-25-2633	490000	745	84	2015-02-17 09:13:23.000	E020
24	CR024	572-75-3961	780000	800	95	2015-12-11 11:15:59.000	E020
25	CR025	431-65-4999	780000	763	87	2015-08-21 13:55:46.000	E013
26	CR026	846-48-1567	700000	730	83	2015-06-21 16:28:47.000	E006
27	CR027	305-38-8280	590000	720	80	2015-08-11 13:55:36.000	E020
28	CR028	878-54-5212	280000	810	96	2015-03-25 08:25:47.000	E013
29	CR029	237-24-7774	190000	710	80	2015-09-11 14:25:46.000	E006
30	CR030	944-78-4575	590000	780	90	2015-08-21 19:14:56.000	E020
31	CR031	813-47-8781	770000	860	88	2015-08-21 11:25:36.000	E013
32	CR032	886-57-7882	890000	830	98	2015-09-01 09:35:32.000	E006
33	CR033	260-61-6435	230000	756	85	2015-03-11 15:35:57.000	E020
34	CR034	271-76-4204	590000	812	96	2015-04-03 13:26:29.000	E006

At the bottom of the results pane, it says "Query executed successfully." The status bar at the bottom right shows "HELEN-LAPTOP\SQLEXPRESS (12... | HELEN-LAPTOP\Helen (52) | master | 00:00:00 | 34 rows".

9. PERFORMANCE

a. Before

The screenshot shows the Microsoft SQL Server Management Studio interface. The Object Explorer on the left shows the database structure for 'HELEN-LAPTOP\SQLEXPRESS'. The 'Tables' node under 'MSIS-2603-Project<DBQL>' contains a single table named 'performance'. A query window is open with the following SQL code:

```
select *  
from performance
```

The results grid displays 25 rows of data from the 'performance' table:

	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

At the bottom of the results grid, a message states: "Query executed successfully." The status bar at the bottom right shows: HELEN-LAPTOP\SQLEXPRESS (12...) | HELEN-LAPTOP\Helen (52) | master | 00:00:00 | 25 rows.

b. After

The screenshot shows the Microsoft SQL Server Management Studio interface, identical to the previous one but with a different set of data in the 'performance' table. The results grid displays 24 rows of data from the 'performance' table:

	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

At the bottom of the results grid, a message states: "Query executed successfully." The status bar at the bottom right shows: HELEN-LAPTOP\SQLEXPRESS (12...) | HELEN-LAPTOP\Helen (52) | master | 00:00:00 | 24 rows.

10. EMPLOYEE

a. Before

The screenshot shows the Microsoft SQL Server Management Studio interface. In the Object Explorer, the database 'MSIS-2603-Project<DBQL>' is selected. In the Results pane, a query is run:

```
select * from EMPLOYEE
```

The results show 20 rows of employee data:

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	985-866-3285	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

Query executed successfully.

b. After

The screenshot shows the Microsoft SQL Server Management Studio interface. In the Object Explorer, the database 'MSIS-2603-Project<DBQL>' is selected. In the Results pane, a query is run:

```
select * from EMPLOYEE
```

The results show 19 rows of employee data, identical to the 'Before' screenshot except for the last row which has been removed:

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	1990-05-21	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	985-866-3285	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

Query executed successfully.

11. QUOTA

a. Before

The screenshot shows the Microsoft SQL Server Management Studio interface. In the Object Explorer, the database 'HELEN-LAPTOP\SQLEXPRESS' is selected. In the center pane, a query window displays the following SQL code:

```
select *  
from quota
```

The results pane shows a table with 18 rows, each containing a quota ID, its value, and a descriptive text. The columns are labeled QUOTA_ID, QUOTA_VAL, and QUOTA_DESC.

QUOTA_ID	QUOTA_VAL	QUOTA_DESC
4	Q004	TOTAL AMOUNT OF LOAN DISBURSED PER MONTH
5	Q005	TOTAL AMOUNT OF LOAN DISBURSED PER MONTH
6	Q006	TOTAL AMOUNT OF LOAN DISBURSED PER MONTH
7	Q007	TOTAL NUMBER OF CREDITS GENERATED PER MONTH
8	Q008	TOTAL NUMBER OF CREDITS GENERATED PER MONTH
9	Q009	TOTAL NUMBER OF CREDITS GENERATED PER MONTH
10	Q010	AVERAGE SAVING AMOUNT PER MONTH
11	Q011	AVERAGE SAVING AMOUNT PER MONTH
12	Q012	AVERAGE SAVING AMOUNT PER MONTH
13	Q013	TOTAL NUMBER OUTSTANDING PER MONTH
14	Q014	TOTAL NUMBER OUTSTANDING PER MONTH
15	Q015	TOTAL NUMBER OUTSTANDING PER MONTH
16	Q016	TOTAL LOAN COLLECTION PER MONTH
17	Q017	TOTAL LOAN COLLECTION PER MONTH
18	Q018	TOTAL LOAN COLLECTION PER MONTH

At the bottom of the results pane, a message indicates: "Query executed successfully." The status bar at the bottom right shows the connection details: HELEN-LAPTOP\SQLEXPRESS (12... | HELEN-LAPTOP\Helen (52) | master | 00:00:00 | 18 rows".

The screenshot shows the Microsoft SQL Server Management Studio interface. In the Object Explorer, the database 'HELEN-LAPTOP\SQLEXPRESS' is selected. In the center pane, a query window displays the following SQL code:

```
delete from quota  
where quota_id = 'q018'
```

The results pane shows a message indicating the number of rows affected: "(1 row(s) affected)".

At the bottom of the results pane, a message indicates: "Query executed successfully." The status bar at the bottom right shows the connection details: HELEN-LAPTOP\SQLEXPRESS (12... | HELEN-LAPTOP\Helen (55) | MSIS-2603-Project<DBQL> | 00:00:00 | 0 rows".

b. After

SQLQuery6.sql - HELEN-LAPTOP\SQLEXPRESS.MSIS-2603-Project<DBQL> (HELEN-LAPTOP\Helen (56)) - Microsoft SQL Server Management Studio

File Edit View Project Debug Tools Window Help

New Query Execute Debug

Object Explorer

Connect HELEN-LAPTOP\SQLEXPRESS (SQL Server)

Databases System Databases MSIS-2603-Project<DBQL>

- Tables
- Views
- Synonyms
- Programmability
- Service Broker
- Storage
- 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
```

Result Messages

QUOTA_ID	QUOTA_VAL	QUOTA_DESC
3	Q003	35
4	Q004	200000
5	Q005	300000
6	Q006	400000
7	Q007	200
8	Q008	250
9	Q009	280
10	Q010	2000000
11	Q011	3000000
12	Q012	4000000
13	Q013	1000000
14	Q014	1500000
15	Q015	2000000
16	Q016	1000000
17	Q017	2000000

Query executed successfully.

HELEN-LAPTOP\SQLEXPRESS (12...) HELEN-LAPTOP\Helen (56) MSIS-2603-Project<DBQL> 0000000 17 rows

Ready Col 3 INS

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);
```

SQlQuery2.sql - RJ...RJ\Rohit Jacob (54)* SQLQuery1.sql - RJ...RJ\Rohit Jacob (53)* DBQLV6.sql - RJ.mas...J\Rohit Jacob (52)*

```
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);
```

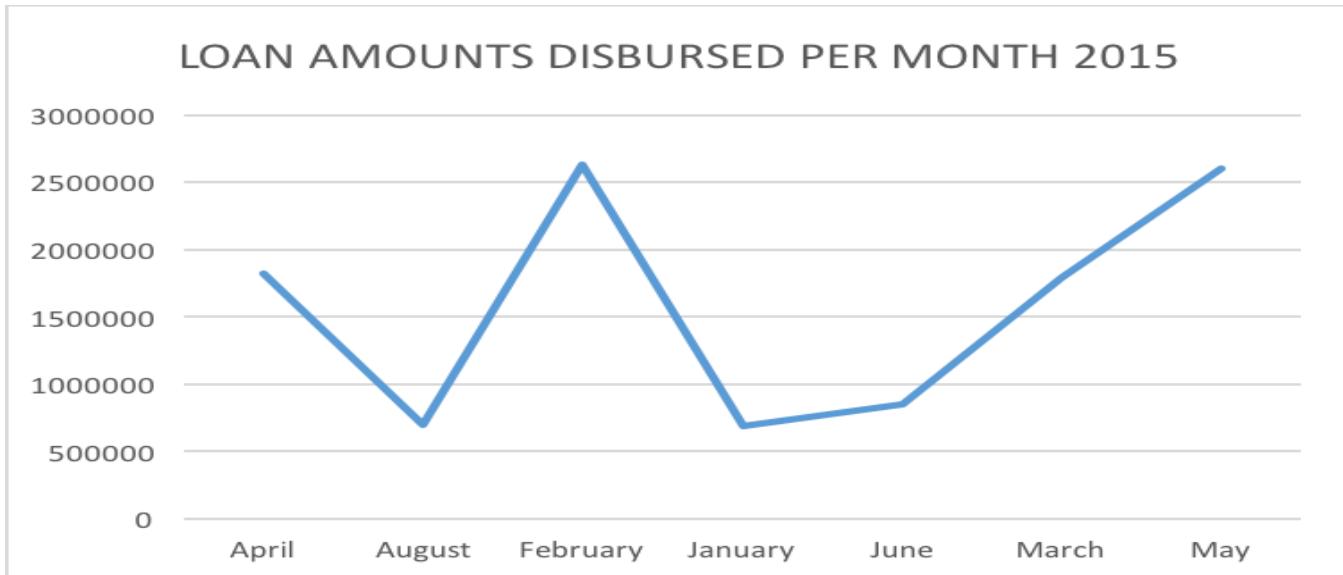
100 %

Results Messages

	MONTH	LOAN AMOUNT DISBURSED
1	April	1820000
2	August	700000
3	February	2630000
4	January	690000
5	June	850000
6	March	1790000
7	May	2600000

Query executed successfully.

RJ (12.0 RTM) | RJ\Rohit Jacob (53) | MSIS-2603-Project<DBQL> | 00:00:00 | 7 rows



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;
```

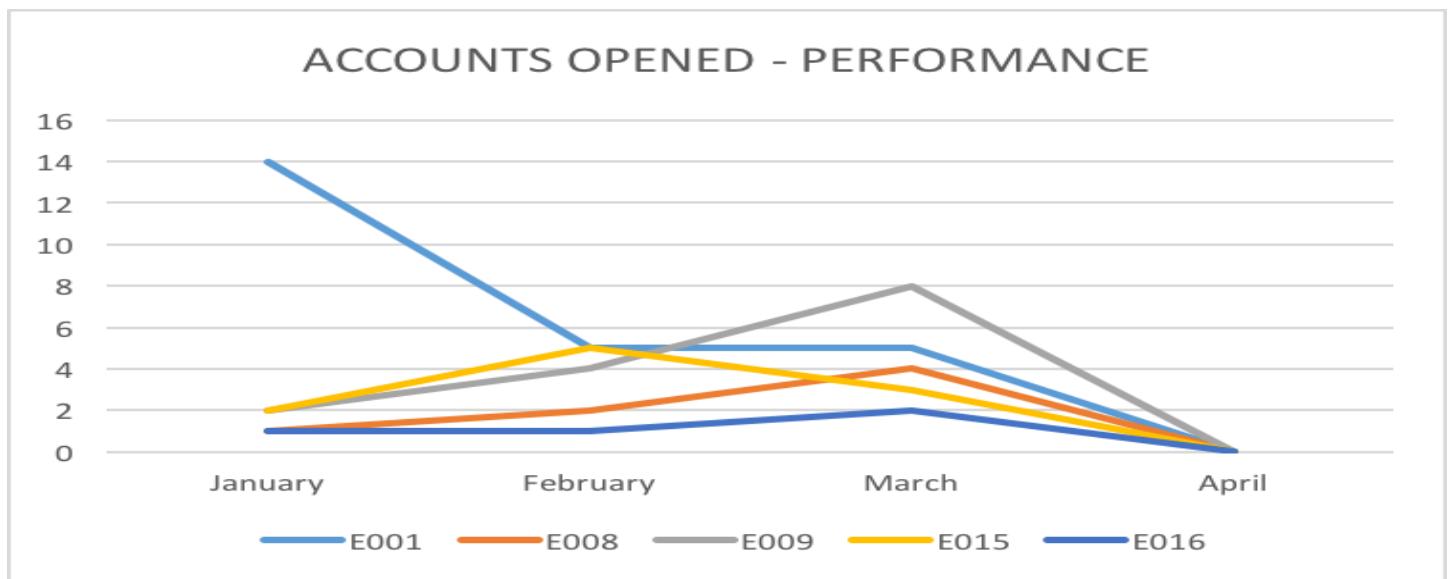
SQLQuery2.sql - RJ...RJ\Rohit Jacob (54)* SQLQuery1.sql - RJ...RJ\Rohit Jacob (53)* DBQLV6.sql - RJ.mas...J\Rohit Jacob (52)*

```
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 %

	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;

```

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

```

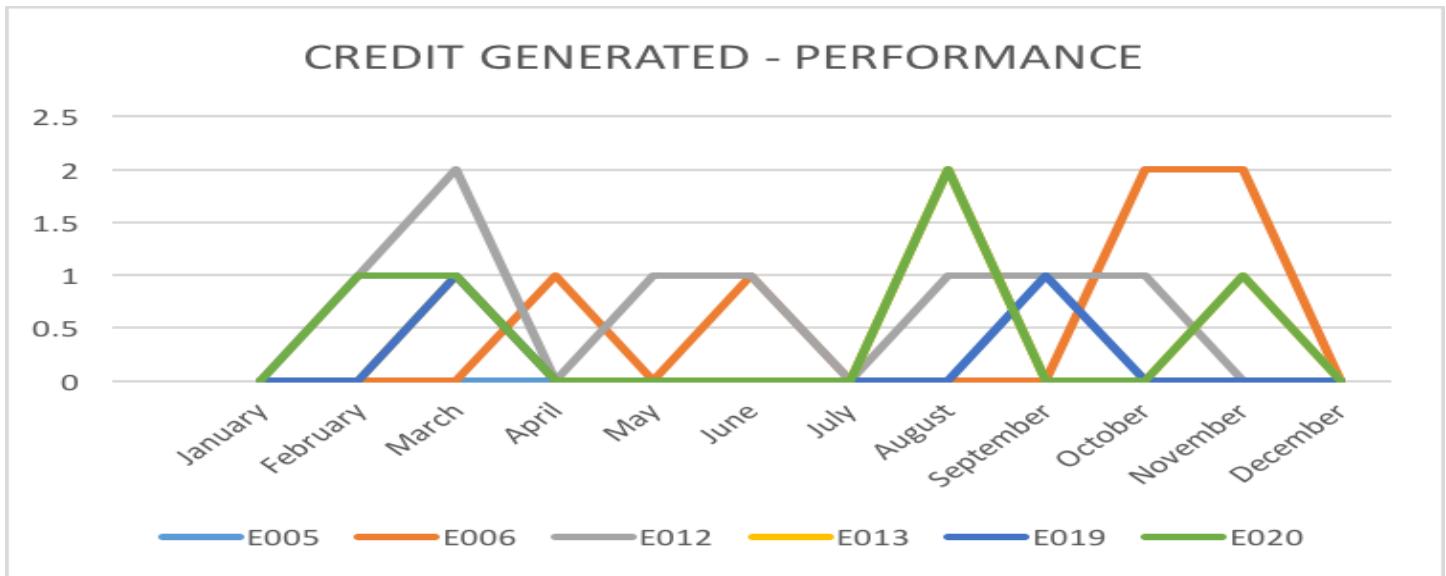
SELECT DATENAME(MONTH, CREDIT_GEN.CREDIT_TIMESTAMP) AS 'MONTH', CREDIT_GEN.EMP_ID, COUNT(*) 'CREDIT GENERATED'
FROM CREDIT_GEN
WHERE DATEPART(YEAR, CREDIT_GEN.CREDIT_TIMESTAMP) = '2015'
GROUP BY DATENAME(MONTH, CREDIT_GEN.CREDIT_TIMESTAMP), CREDIT_GEN.EMP_ID;

```

Results Messages

	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

Query executed successfully. | RJ (12.0 RTM) | RJ\Rohit Jacob (53) | MSIS-2603-Project<DBQL> | 00:00:00 | 21 rows



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;
```

SQlQuery3.sql - RJ...RJ\Rohit Jacob (54)* SQLQuery2.sql - RJ...RJ\Rohit Jacob (53)* SQLQuery1.sql - RJ...RJ\Rohit Jacob (52)* SQLQuery7.sql - RJ...RJ\Rohit Jacob (55)*

```
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;
```

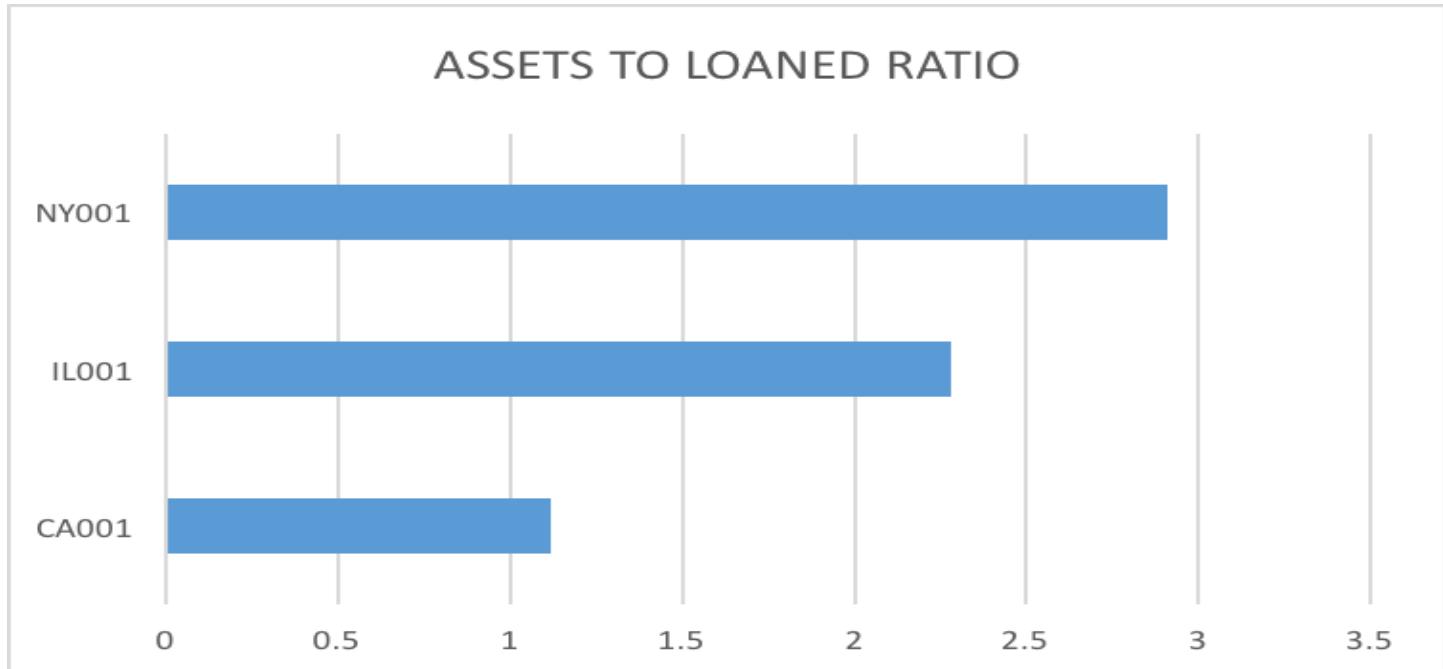
100 %

Results Messages

	BRANCH	BRANCH PERFORMANCE
1	CA001	1.12
2	IL001	2.28
3	NY001	2.91

Query executed successfully.

RJ (12.0 RTM) | RJ\Rohit Jacob (53) | MSIS-2603-Project<DBQL> | 00:00:00 | 3 rows



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;
```

SQlQuery4.sql - RJ....RJ\Rohit Jacob (55) SQlQuery3.sql - RJ....RJ\Rohit Jacob (54)* SQlQuery2.sql - RJ....RJ\Rohit Jacob (53)* SQlQuery1.sql - RJ....RJ\Rohit Jacob (52)*

```
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;
```

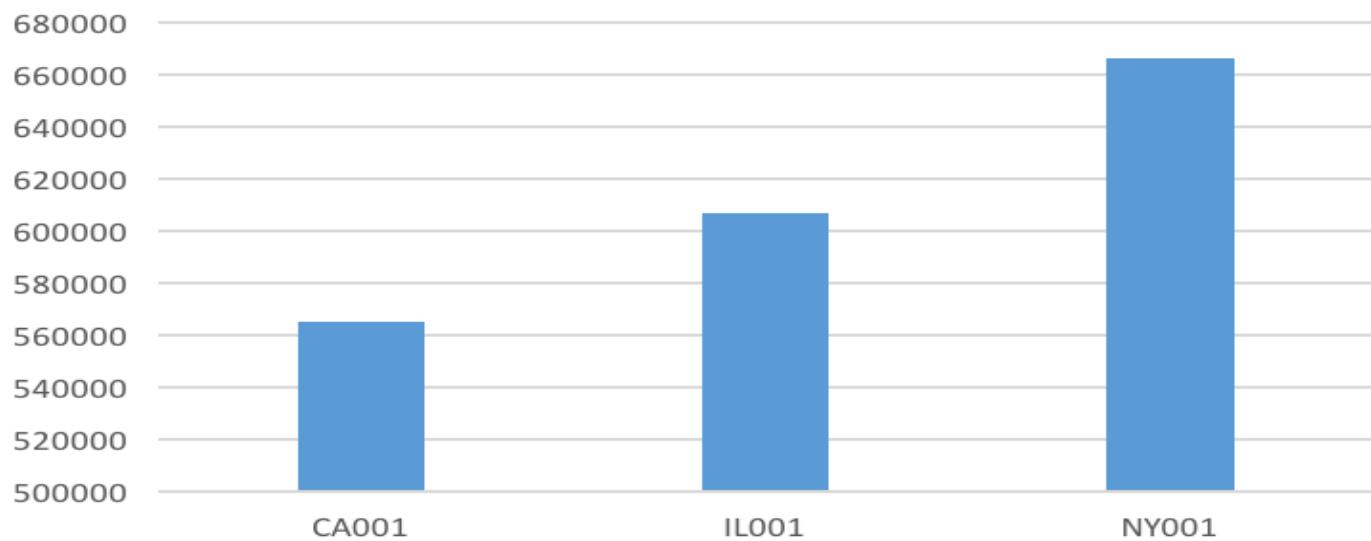
100 %

Results Messages

	BRANCH_ID	AVERAGE ACCOUNT BALANCES
1	CA001	565153.87
2	IL001	606555.02
3	NY001	666020.52

Query executed successfully. | RJ (12.0 RTM) | RJ\Rohit Jacob (52) | MSIS-2603-Project<DBQL> | 00:00:00 | 3 rows

AVERAGE ACCOUNT BALANCES



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)
```

SQLQuery6.sql - RJ...RJ\Rohit Jacob (58)* × SQLQuery5.sql - RJ...RJ\Rohit Jacob (57)* × SQLQuery3.sql - RJ...RJ\Rohit Jacob (54)*

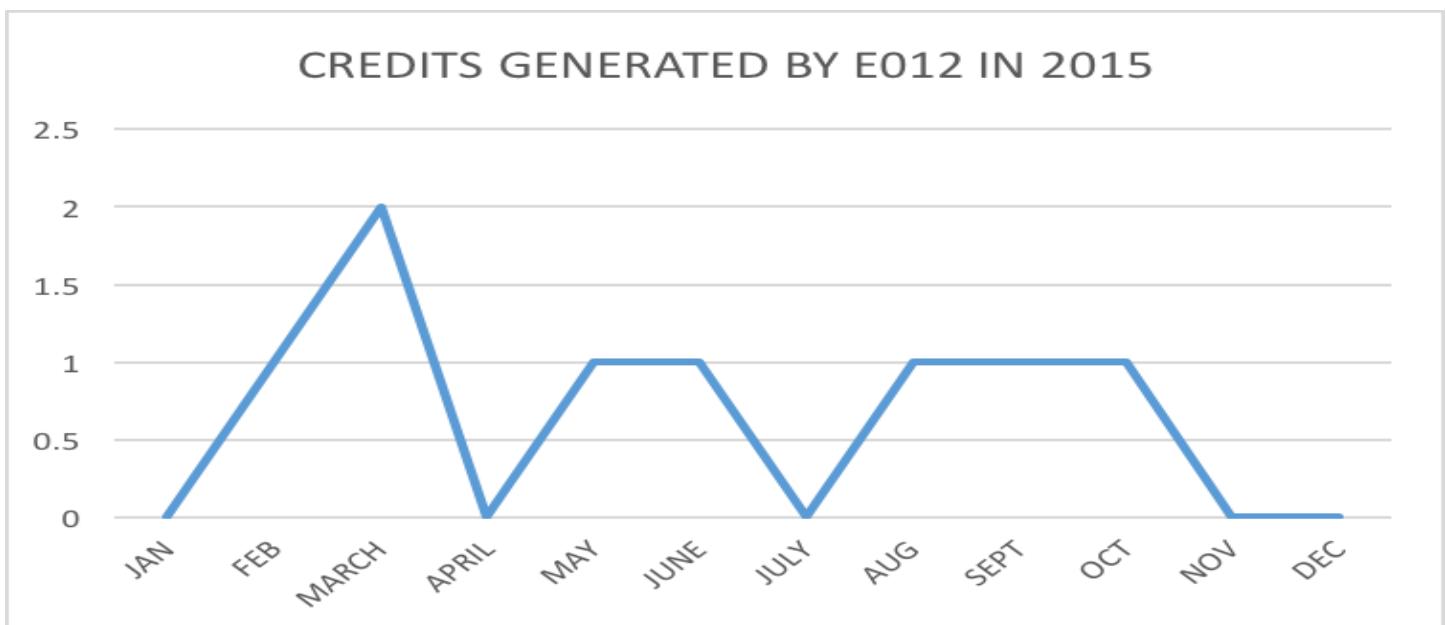
```
SELECT DATENAME(MONTH, CREDIT_GEN.CREDIT_TIMESTAMP) AS 'MONTH', COUNT(*) 'CREDITS GENERATED'
FROM CREDIT_GEN
WHERE CREDIT_GEN.EMP_ID = 'E012'
AND DATEPART(YEAR, CREDIT_GEN.CREDIT_TIMESTAMP) = '2015'
GROUP BY DATENAME(MONTH, CREDIT_GEN.CREDIT_TIMESTAMP)
```

100 %

Results Messages

	MONTH	CREDITS GENERATED
1	August	1
2	February	1
3	June	1
4	March	2
5	May	1
6	October	1
7	September	1

Query executed successfully. | RJ (12.0 RTM) | RJ\Rohit Jacob (58) | MSIS-2603-Project<DBQL> | 00:00:00 | 7 rows



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;
```

SQLQuery6.sql - RJ...RJ\Rohit Jacob (58))" [SQLQuery5.sql - RJ...RJ\Rohit Jacob (57))" [SQLQuery3.sql - RJ...RJ\Rohit Jacob (54))"

```
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;
```

100 %

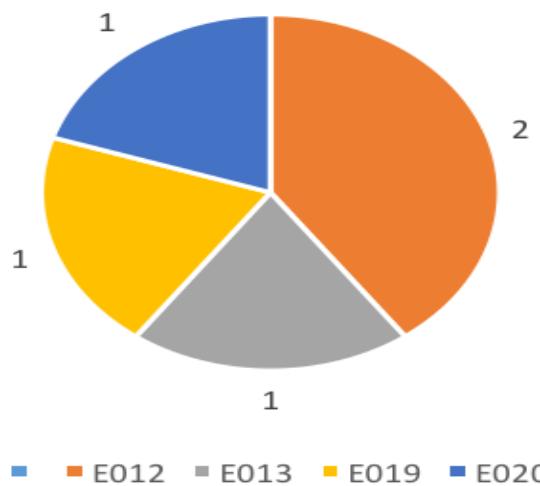
Results Messages

	EMP_ID	CREDIT SCORES GENERATED IN MARCH 2015
1	E012	2
2	E013	1
3	E019	1
4	E020	1

Query executed successfully.

RJ (12.0 RTM) | RJ\Rohit Jacob (57) | MSIS-2603-Project<DBQL> | 00:00:00 | 4 rows

CREDITS GENERATED IN MARCH



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);
```

The screenshot shows the SQL Server Management Studio interface with a query window and a results grid.

Query Window:

```
SQLQuery3.sql - RJ...RJ\Rohit Jacob (54)*      SQLQuery2.sql - RJ...RJ\Rohit Jacob (53)*      SQLQuery1.sql - RJ...RJ\Rohit Jacob (52)*      SQLQuery7.sql - RJ...RJ\Rohit Jacob (55)* X  
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);
```

Results Grid:

	MONTH	ACCOUNTS OPENED
1	February	5
2	January	14
3	March	5

Status Bar:

Query executed successfully. | RJ (12.0 RTM) | RJ\Rohit Jacob (55) | MSIS-2603-Project<DBQL> | 00:00:00 | 3 rows

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 DECIMAL(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';
```

The screenshot shows the SQL Server Management Studio interface with two queries in the query editor and their results in the results pane.

Query Editor 1 (Top):

```
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 DECIMAL(18,2)) AS 'PERCENT ACHIEVED'
FROM QUOTA, PERFORMANCE, ACCOUNTS_OPENED
WHERE QUOTA.QUOTA_ID = PERFORMANCE.QUOTA_ID
AND ACCOUNTS_OPENED.DESIGNATED REP_ID = PERFORMANCE.EMP_ID
AND QUOTA.QUOTA_DESC = 'NUMBER OF NEW ACCOUNTS OPENS PER MONTH';
```

Results Pane:

EMP_ID	ACCOUNTS OPENED IN JANUARY	TARGET	PERCENT ACHIEVED	
1	E001	26	35	74.00
2	E008	2	30	6.00
3	E009	4	30	13.00
4	E015	4	20	20.00
5	E016	2	20	10.00

Status Bar:

Query executed successfully. | RJ (12.0 RTM) | RJ\Rohit Jacob (54) | MSIS-2603-Project<DBQL> | 00:00:00 | 5 rows

Project Summary

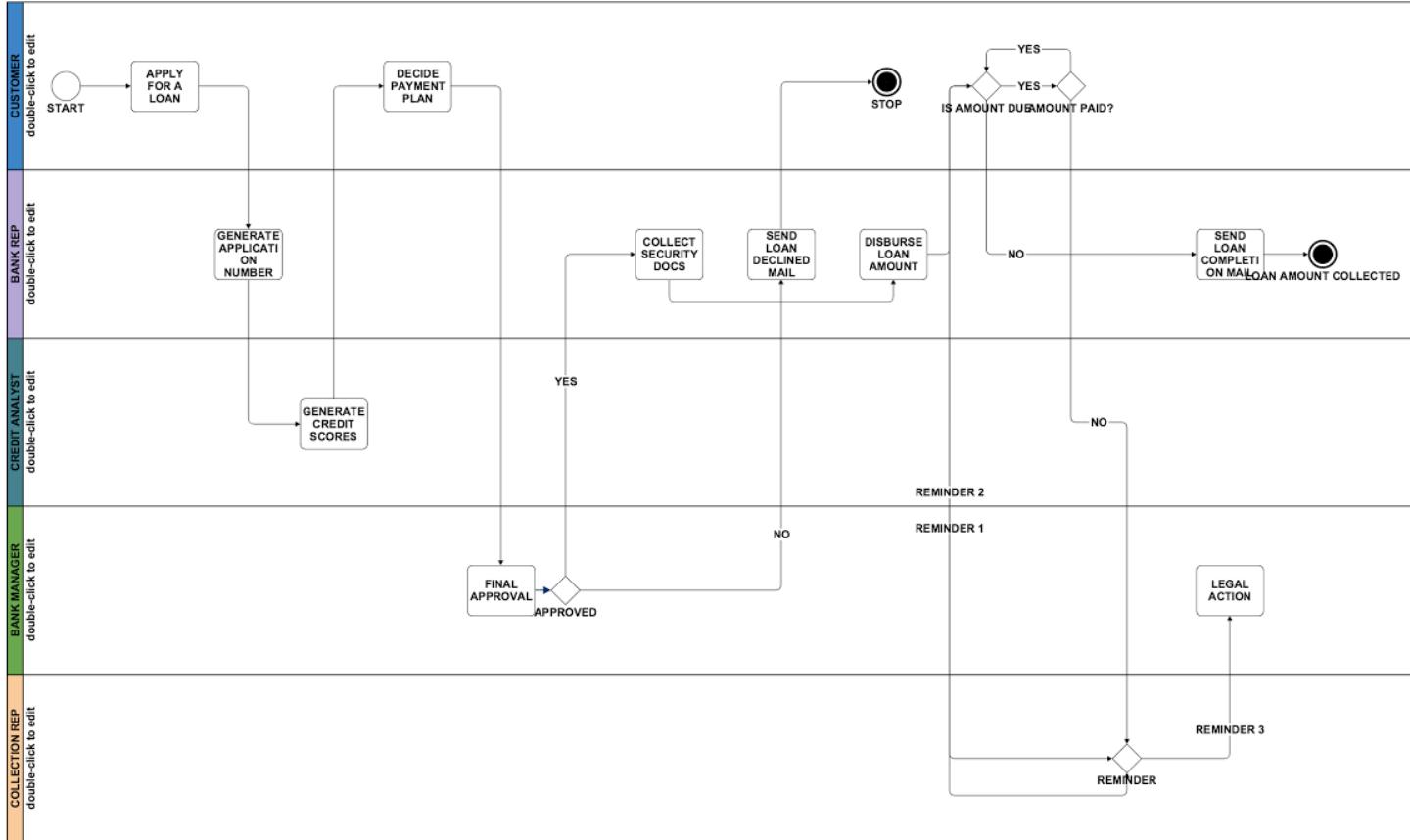
Our project started with a single branch Bank that provides only education loans to students. This project is evolved to a complex database schema of a real-life bank with multiple branches that provides more than 3 types of loans products to customer. We have created different KPIs for each employee 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.
