SQL Project

Presented by

Janah Vijayarathnam

Index

INTRODUCTION

SQL stands for Structured Query Language designed to facilitate retrieving specific information from databases. SQL is the language of databases used to communicate with a data.

PROJECT DETAILS

This project report contains details of a small database 'Bank' divided into two phases.

- Phase 1- Table Design and Data addition

In this phase tables are created along with assigning constraints to relate the tables. Few data has been inserted to process the queries and retrieve results.

- Phase 2- Data Manipulation

Sets of queries has been executed and results presented. This has facilitated in learning and understanding different functions deeply.

Project Phase 1:

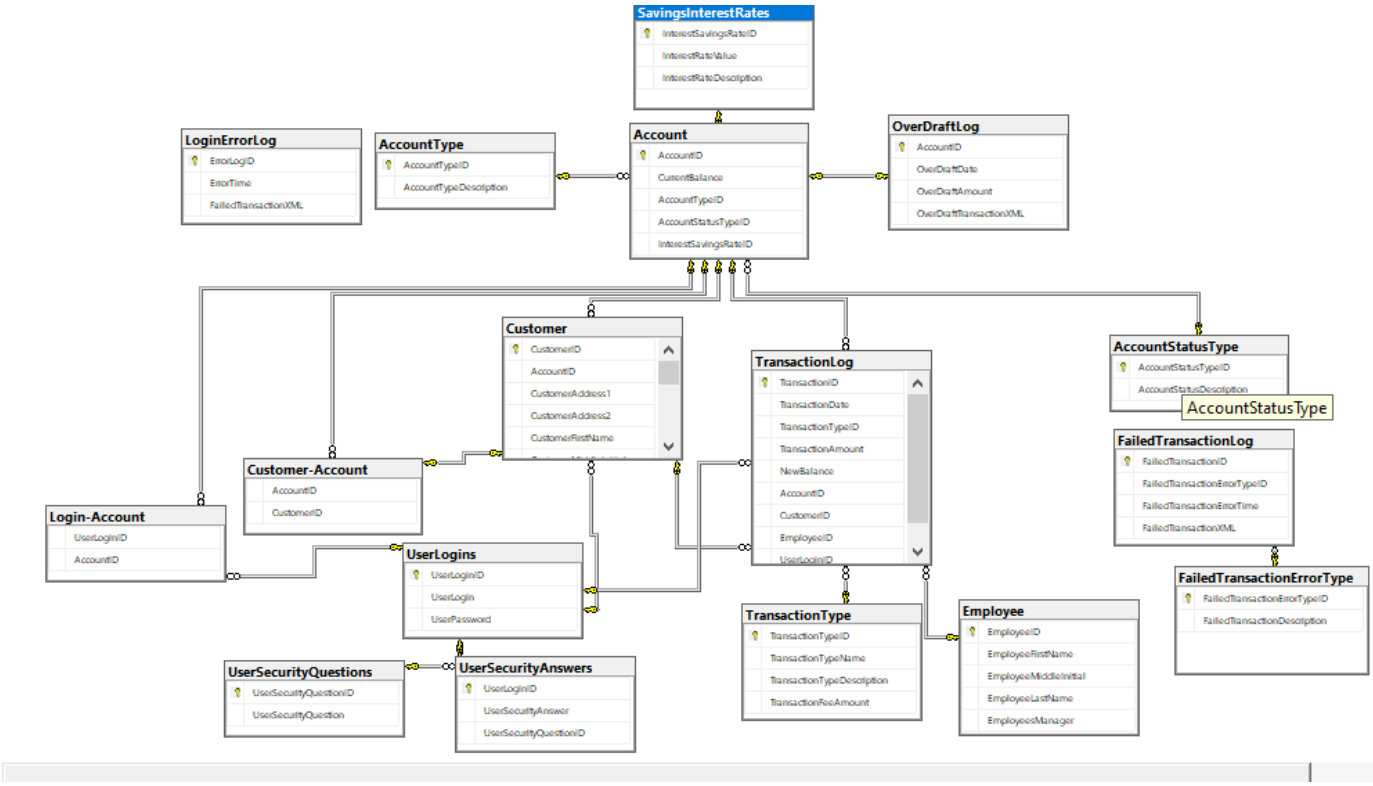
Question/Problem 1

1. **Create a database for a banking application called "Bank". [Basic]**
2. **Create all the tables mentioned in the database diagram. [Moderate]**
3. **Create all the constraints based on the database diagram. [Advanced]**
4. **Insert at least 5 rows in each table. [Basic]**

SUMMARY:

- Bank Database has been created with 17 interrelated Tables.
- Constraints used Primary Key, Foreign Key, Not Null and Unique.
- Data consists of 6 Customers and 5 Account which includes 1 Joint Account.

Database Design



Tables with Entered values:

90 %

Results | Messages

	AccountID	CurrentBalance	AccountTypeID	AccountStatusTypeID	InterestSavingsRateID
1	1001031985	8000.00	11	200	97
2	1001042382	6120.00	22	200	95
3	1001067211	5500.00	22	200	93
4	1001074102	-250.00	11	201	97
5	1001078516	0.00	22	200	92

Account

Results | Messages

	UserLoginID	UserLogin	UserPassword
1	1010	HanaH	Hana5
2	1234	AlexA	Alex1
3	2345	RubyD	Ruby3
4	3128	SamT	Sam6
5	6789	MiaC	Mia2
6	7890	DaveH	Dave4

UserLogins

Results | Messages

	UserSecurityQuestionID	UserSecurityQuestion
1	101	Favourite Colour?
2	102	First Car?
3	103	Pets Name?
4	104	Favourite Sport?
5	105	Favourite Country?
6	106	Favourite City?

User Security Questions

PROJECT PHASE 2

Question 1 - Create a view to get all customers with checking account from ON province.

Syntax and Output:

```
Create view CheckingAccount_ON
as
select CustomerID,CustomerFirstName,CustomerLastName,AccountTypeDescription,State from Customer
join Account on Customer.AccountID=Account.AccountID
Join AccountType on Account.AccountTypeID=AccountType.AccountTypeID
where AccountTypeDescription='Checking Account' and State='ON'
go

select * from CheckingAccount_ON
```

90 %

Results | Messages

	CustomerID	CustomerFirstName	CustomerLastName	AccountTypeDescription	State
1	7410201	Sam	Thomas	Checking Account	ON

Question 2: Create a view to get all customers with total account balance (including interest rate) greater than 5000. [Advanced]

Syntax and Output:

```sql
/*2.    Create a view to get all customers with total account balance (including interest rate) greater than 5000. [Advanced]*/
select * from Account
select * from SavingsInterestRates
select * from customer

Create view View_TotalAccountBal
as
select CustomerFirstName,CustomerLastName,CurrentBalance,InterestRateValue,(CurrentBalance*InterestRateValue)
as InterestAmount,(CurrentBalance+CurrentBalance*InterestRateValue)
as TotalAccountBalance from Account
inner Join SavingsInterestRates on Account.InterestSavingsRateID=SavingsInterestRates.InterestSavingsRateID
inner Join Customer on Customer.AccountID=Account.AccountID
Where CurrentBalance+CurrentBalance*InterestRateValue>5000
go

select * from View_TotalAccountBal
```

90 %

Results | Messages

	CustomerFirstName	CustomerLastName	CurrentBalance	InterestRateValue	InterestAmount	TotalAccountBalance
1	Dave	Harvis	8500.00	0.000000000	0.0000000000000000000	8500.0000000000000000000
2	Hana	Harvis	8500.00	0.000000000	0.0000000000000000000	8500.0000000000000000000
3	Ruby	Donald	6120.00	0.700000000	21.4200000000000000000	6141.4200000000000000000
4	Alex	Antony	5500.00	0.500000000	13.7500000000000000000	5513.7500000000000000000

Question 3: Create a view to get counts of checking and savings accounts by customer. [Moderate]

Syntax and Output:

```
Create view View_TotalAccounts
as
select CustomerID,CustomerFirstName,CustomerLastName,AccountTypeDescription, Count(Account.AccountTypeID) as TotalAccounts
from Customer
Join  Account on Account.AccountID = Customer.AccountID
join AccountType on Account.AccountTypeID = AccountType.AccountTypeID
group by  CustomerID,CustomerFirstName,CustomerLastName,AccountTypeDescription
go

select * from View_TotalAccounts
```

90 %

Results | Messages

	CustomerID	CustomerFirstName	CustomerLastName	AccountTypeDescription	TotalAccounts
1	3198501	Dave	Harvis	Checking Account	1
2	3198502	Hana	Harvis	Checking Account	1
3	4238201	Ruby	Donald	Savings Account	1
4	6721101	Alex	Antony	Savings Account	1
5	7410201	Sam	Thomas	Checking Account	1
6	7851601	Mia	Chan	Savings Account	1

Question 4: Create a view to get any particular user's login and password using AccountId. [Moderate]

Syntax and Output:

```sql
select * from View_TotalAccounts
/*4.    Create a view to get any particular user's login and password using AccountId. [Moderate]*/
Select * from customer
select * from TransactionLog
select * from UserLogins
Create View View_AccountID_UsrPswd
as
Select Distinct CustomerID,UserLogins.UserLoginID,UserPassword
from TransactionLog
Join UserLogins on TransactionLog.UserLoginID=UserLogins.UserLoginID
Where Transactionlog.AccountID = 1001031985
go

select * from View_AccountID_UsrPswd
```

90 %

Results | Messages

	CustomerID	UserLoginID	UserPassword
1	3198501	7890	Dave4
2	3198502	1010	Hana5

Question 5 : Create a view to get all customers' overdraft amount. [Moderate]

Syntax and Output:

```sql
select * from customer
select * from OverDraftLog

Create View View_Customer_OD
as
Select CustomerID,CustomerFirstName,CustomerLastName,OverDraftAmount
from Customer
Join OverDraftLog on Customer.AccountID=OverDraftLog.AccountID
go

select * from View_Customer_OD
```

0 %

Results | Messages

	CustomerID	CustomerFirstName	CustomerLastName	OverDraftAmount
1	7410201	Sam	Thomas	230.00

Question 6: Create a stored procedure to add "User_" as a prefix to everyone's login (username). [Moderate]

Syntax and Output:

```
/*6.    Create a stored procedure to add "User_" as a prefix to everyone's login (username). [Moderate]*/
Create proc SP_Username
as
begin
        Update UserLogins set UserLogin= 'User_'+ UserLogin

end

exec SP_Username
go

select * from UserLogins
go
```

90 %

Results | Messages

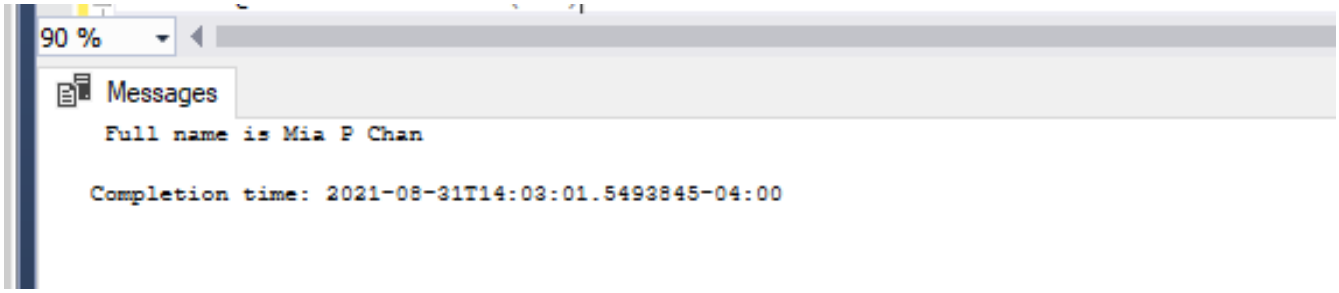
	UserLoginID	UserLogin	UserPassword
1	1010	User_HanaH	Hana5
2	1234	User_AlexA	Alex1
3	2345	User_RubyD	Ruby3
4	3128	User_SamT	Sam6
5	6789	User_MiaC	Mia2
6	7890	User_DaveH	Dave4

Question 7: Create a stored procedure that accepts AccountId as a parameter and returns customer's full name. [Advanced]

Syntax :

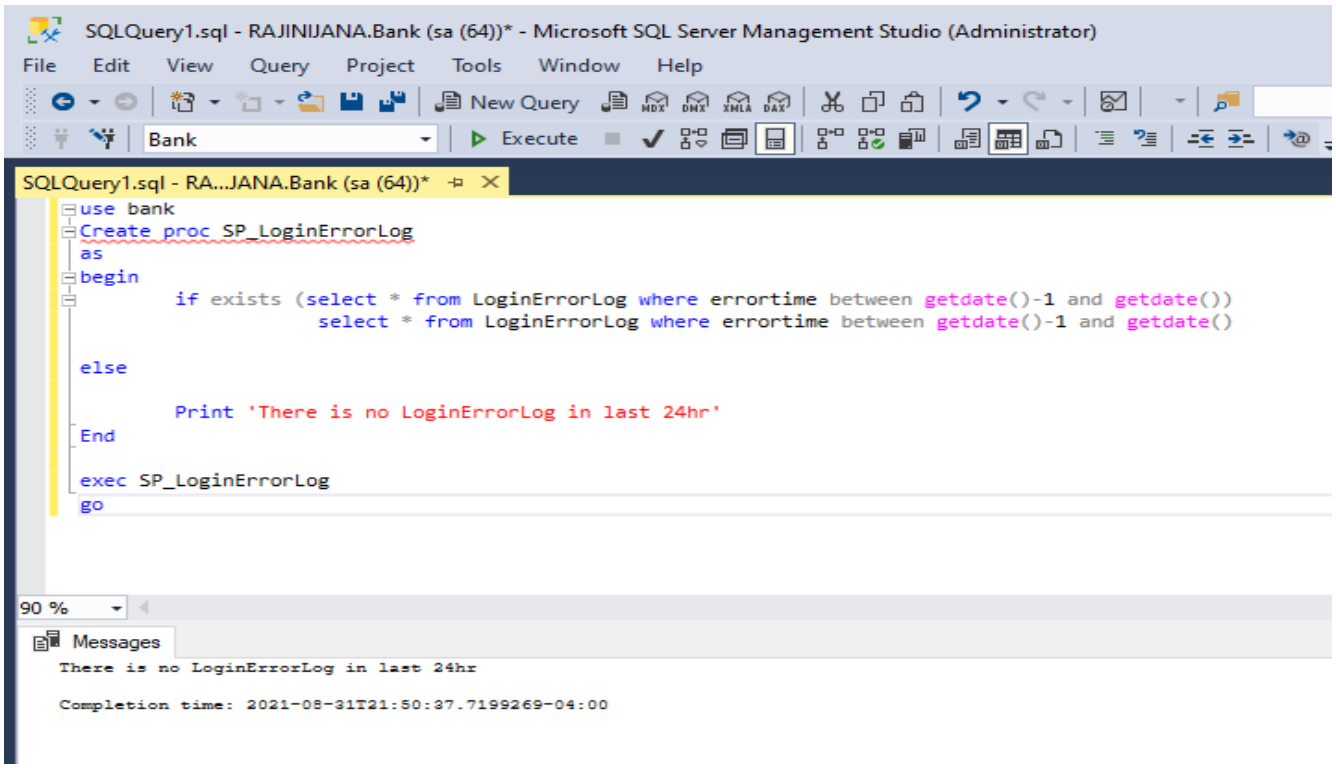
```
/*7.    Create a stored procedure that accepts AccountId as a parameter and returns customer's fu
create proc spFullNameFromAccountId
            @AccountID int,

            @Fullname nvarchar(100) output
as
begin
  if (@AccountID in (select AccountID from Customer))
    begin
       select @Fullname=c.customerFirstName+' '+c.customerMiddleInitial+' '+c.customerLastName
       from customer C

    end
   else
    begin
     print 'This Account Id does not exists!'
    end
end
go

--Executing for valid account id
Declare @FullName nvarchar(100)
exec spFullNameFromAccountId 1001067211, @FullName out
Print ' Full name is '+replace (@FullName,'   ',' ')
go

--Executing for invalid account id
Declare @FullName nvarchar(100)
exec spFullNameFromAccountId 2999, @FullName out
Print ' Full name is '+@FullName
go
```

Output:



Question 8: Create a stored procedure that returns error logs inserted in the last 24 hours. [Advanced]

Syntax and Output:



Question 9: Create a stored procedure that takes a deposit as a parameter and updates CurrentBalance value for that particular account. [Advanced]

Syntax:

SQLQuery1.sql - RAJINIJANA.Bank (sa (64))* - Microsoft SQL Server Management Studio (Administrator)

File Edit View Query Project Tools Window Help

New Query | Bank | Execute

SQLQuery1.sql - RA...JANA.Bank (sa (64))*

```
go

Create Proc Deposit_CurrentBalance
            @AcctID int,@Deposit_Amt money
as
Begin
        If (@AcctID in (Select AccountID from Account))
        Begin
            Select CurrentBalance as [Balance Before Deposit] from Account
            where AccountID=@AcctID
            Update Account
            Set CurrentBalance = CurrentBalance+@Deposit_Amt where AccountID=@AcctID
            Select CurrentBalance [Balance After Deposit] from Account where AccountID=@Acct
            End
        Else
        Begin
            Print
            'AccountID does not exists'
        End
End

-- Executing with a valid AccountID
Exec Deposit_CurrentBalance 1001031985,500

-- Executing with invalid AccountID
Exec Deposit_CurrentBalance 1001031333,500

go
```

Output 1:

90 %

Results | Messages

	Balance Before Deposit
1	8000.00

	Balance After Deposit
1	8500.00

Query executed successfully.

Output 2:

```
go
```

90 %

Messages

```
AccountID does not exists

Completion time: 2021-08-31T22:01:54.1972792-04:00
```

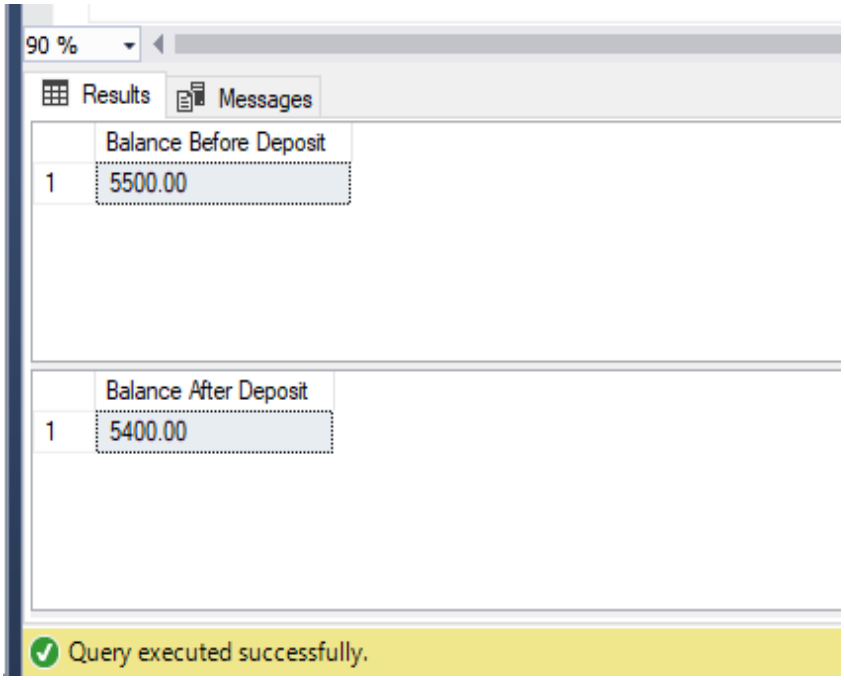
Question 10: Create a stored procedure that takes a withdrawal amount as a parameter and updates

Syntax:

SQL Project Query-...JANA.Bank (sa (52))

```
use bank

Create Proc Withdrawal_CurrentBalance
            @AcctID int,@Withdrawal_Amt money
as
Begin
        If (@AcctID in (Select AccountID from Account))
        Begin
            Select CurrentBalance as [Balance Before Deposit] from Account
            where AccountID=@AcctID
            Update Account
            Set CurrentBalance = CurrentBalance-@Withdrawal_Amt where AccountID=@AcctID
            Select CurrentBalance [Balance After Deposit] from Account where AccountID=@AcctID
            End
        Else
        Begin
            Print
            'AccountID does not exist'
        End
End

-- Executing with a valid AccountID

Exec Withdrawal_CurrentBalance 1001067211,100

-- Executing with invalid AccountID

Exec Withdrawal_CurrentBalance 100106278,500
go
```

Output1:


90 %
Results
Messages
Balance Before Deposit
1
5500.00
Balance After Deposit
1
5400.00
Query executed successfully.