FUNDAMENTALS OF SQL PROGRAMING

PROJECT

PHASE - 2

Shirmila Hewapathirana (*Student*)

Ms Maryam Kazemi (*Professor*)

METRO COLLEGE OF TECHNOLOGY

2024/25

PROJECT SUMMARY

Data sample:

Bank data (Checking, Savings, Business. Students and Visitor accounts)

Total number of Tables

17 tables

Development Method

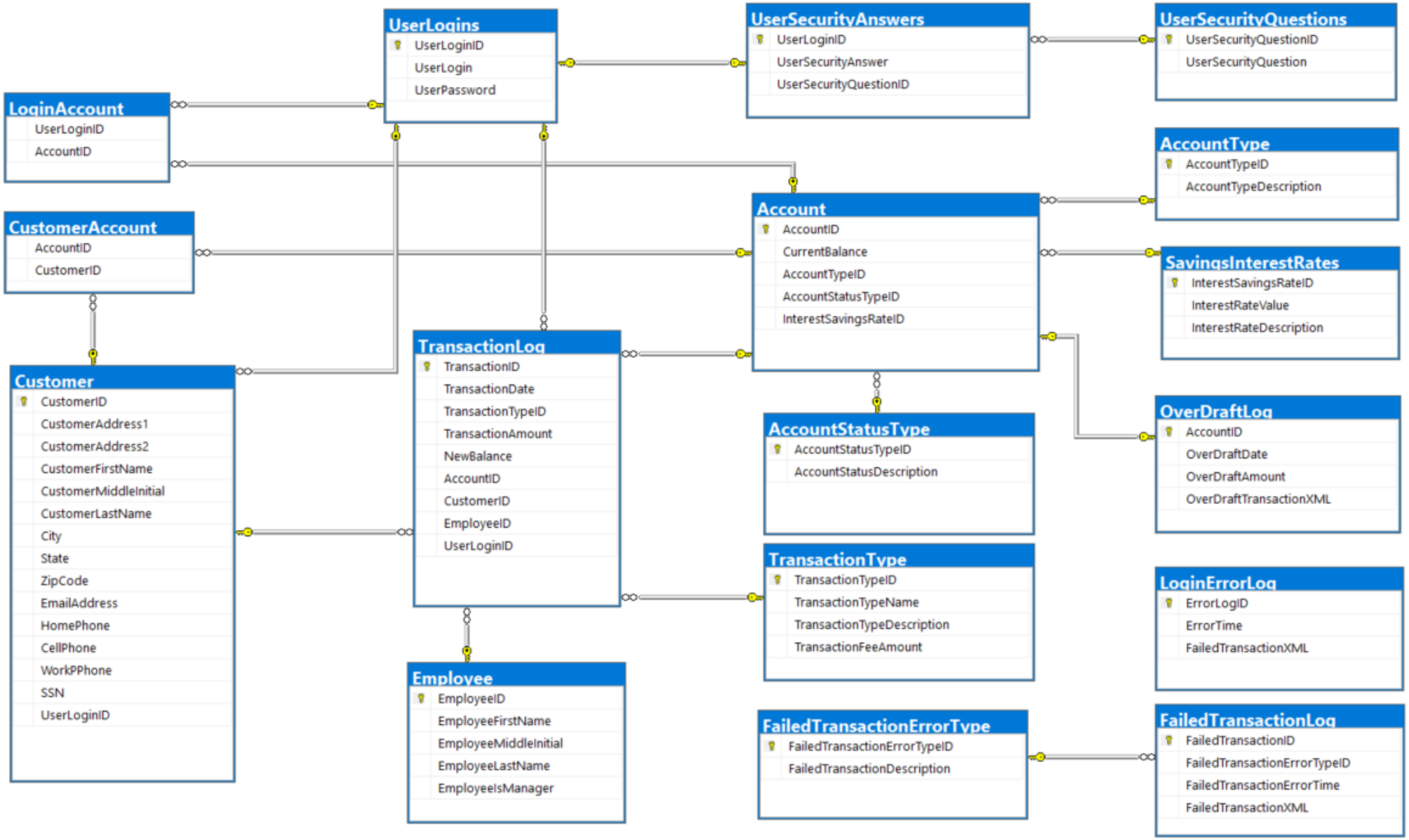
Tables : T-SQL method

Database Diagram : GUI

Process Followed

1. Creating tables with primary key
2. Creating tables with primary keys and Foreign keys
3. Creating Database Diagram
4. Coding work

DATABASE DIAGRAM



Q1. Create a view to get all customers with checking accounts from ON province.

```
392  Create view AllCustomersWithCheckingAcView
393  as
394  select
395          c.CustomerFirstName + ' ' + c.CustomerLastName [Customer Full Name],
396          c.State [Province],  --added an alias to change State into Province in the result
397          at.AccountTypeDescription [Account Type]
398  from Customer c
399  left join CustomerAccount ca
400  on c.CustomerID = ca.CustomerID
401  left join Account a
402  on ca.AccountID = a.AccountID
403  left join AccountType at
404  on a.AccountID = at.AccountTypeID
405  where
406      c.State = 'ON' and
407      AccountTypeDescription= 'checking'
408
409  select * from AllCustomersWithCheckingAcView
```

Results | Messages

	Customer Full Name	Province	Account Type
1	John Doe	ON	Checking

Q2. Create a view to get all customers with a total account balance (including interest rate) greater than 5000.

```
425  Create view CustomersWithTotalBalanceAbove5k
426  as
427  Select  c.CustomerFirstName + ' ' + c.CustomerLastName [Customer Full Name],
428          a.AccountID,
429          sum(a.CurrentBalance * (1+sir.InterestRateValue)) as [TotalAccountBalance]
430  from Customer c
431  left join CustomerAccount ac
432  on c.CustomerID = ac.CustomerID
433  left join Account a
434  on ac.AccountID = a.AccountID
435  left join SavingsInterestRates sir
436  on a.InterestSavingsRateID = sir.InterestSavingsRateID
437  Group by
438          c.CustomerFirstName,
439          c.CustomerLastName,
440          a.AccountID
441  having sum(a.CurrentBalance * (1+sir.InterestRateValue)) > 5000
442
443  select * from CustomersWithTotalBalanceAbove5k
```

Results | Messages

	Customer Full Name	AccountID	TotalAccountBalance
1	George Bush	3	15150.000000000
2	John Doe	1	5151.000000000

Q3. Create a view to get counts of checking and savings accounts by customer.

```
486 create view CheckingandSavingAcbyCustomerView1 as
487 select
488     c.CustomerFirstName + ' ' + c.CustomerLastName [CustomerFullName],
489     sum(case
490             when at.AccountTypeDescription = 'Checking'
491             then 1 else 0 end) as [CheckingAccounts],
492     sum(case
493             when at.AccountTypeDescription = 'Savings'
494             then 1 else 0 end) as [SavingsAccounts]
495 from Customer c
496 left join CustomerAccount ca
497 on c.CustomerID = ca.CustomerID
498 left join Account a
499 on ca.AccountID = a.AccountID
500 left join AccountType at
501 on a.AccountTypeID = at.AccountTypeID
502 group by    c.CustomerFirstName,
503             c.CustomerLastName
504
505 select * from CheckingandSavingAcbyCustomerView1
```

Results | Messages

	CustomerFullName	CheckingAccounts	SavingsAccounts
1	Jimmy Anderson	1	0
2	George Bush	1	1
3	Sean Cooper	0	1
4	John Doe	1	3
5	Anna Jones	1	0
6	Dave Miller	0	0
7	Bob Smith	0	1

Q4. Create a view to get any particular user's login and password using AccountId.

```
520 create view UserLoginCredentialsbyAcIDView
521 as
522 select
523     ca.AccountID,
524     c.CustomerFirstName + ' ' + c.CustomerLastName [CustomerFullName],
525     ul.UserLogin,
526     ul.UserPassword
527 from Customer c
528 left join CustomerAccount ca
529 on c.CustomerID = ca.CustomerID
530 left join UserLogins ul
531 on c.UserLoginID = ul.UserLoginID
532 where AccountID = 3
533
534 select * from UserLoginCredentialsbyAcIDView
535 go
```

Results | Messages

	AccountID	CustomerFullName	UserLogin	UserPassword
1	3	George Bush	User_Brad Pitt	dshjk

Q5. Create a view to get all customers' overdraft amounts.

```
565 create view AllCustomersODAmountsView1
566 as
567 select
568         c.CustomerFirstName + ' ' + c.CustomerLastName [CustomerFullName],
569         sum(odl.OverDraftAmount) as TotalOverDraftAmount
570 from Customer c
571 left join CustomerAccount ca
572 on c.CustomerID = ca.CustomerID
573 right join OverDraftLog odl
574 on ca.AccountID = odl.AccountID
575 group by
576         c.CustomerFirstName,
577         c.CustomerLastName
578
579 select * from AllCustomersODAmountsView1
```

Results | Messages

	CustomerFullName	TotalOverDraftAmount
1	Jimmy Anderson	700.00
2	George Bush	700.00
3	Sean Cooper	600.00
4	John Doe	300.00
5	Dave Miller	500.00

Q6. Create a stored procedure to add “User_” as a prefix to everyone’s login (username)

```
596 create proc sp_AddPrefixToUserLogin
597 as
598 begin
599     if(select count(*) from UserLogins where UserLogin not like 'User_%') > 0
600     begin
601         update UserLogins
602         set UserLogin = 'User_' + UserLogin
603         where UserLogin not like 'User_%'
604         print 'UserLogins successfully updated with prefix "User_"'
605     end
606     else
607     begin
608         print 'No records updated. Allrecords already have the prefix "User_" .'
609     end
610 end
```

Messages

```
(7 rows affected)
UserLogins successfully updated with prefix "User_"

Completion time: 2024-11-13T21:08:18.7071891-05:00
```

Messages

```
No records updated. Allrecords already have the prefix "User_" .

Completion time: 2024-11-13T21:07:33.5166701-05:00
```

Q7. Create a stored procedure that accepts AccountId as a parameter and returns the customer's full name.

```
642 Create proc sp_FullNamebyAccountID
643         (@AcID int,
644          @fullname nvarchar(100) out)
645 as
646 begin
647 if (@AcID in (select AccountID from CustomerAccount))
648     begin
649     select @fullname = c.customerfirstname + ' ' + c.CustomerMiddleInitial +' ' + c.customerlastname
650     from Customer c
651     left join CustomerAccount ca
652     on c.CustomerID = ca.CustomerID
653     where ca.AccountID = @AcId
654     print 'Full name for Account ID '+ convert(varchar(10), @AcID)+ ' is:'
655     end
656     else
657     begin
658     print 'There is no customer withis acount ID: ' + convert(varchar(10), @AcID)
659     end
660 end
```

```
667 declare @output nvarchar(100)
668 exec sp_FullNamebyAccountID 5, @output output
669 print @output
```

```
Messages
Full name for Account ID 5 is:
Dave E Miller

Completion time: 2024-11-13T21:13:17.1001541-05:00
```

```
672 declare @output nvarchar(100)
673 exec sp_FullNamebyAccountID 101, @output output
674 print @output
```

```
Messages
There is no customer withis acount ID: 101

Completion time: 2024-11-13T21:13:45.0913995-05:00
```

Q8. Create a stored procedure that returns error logs inserted in the last 24 hours.

```
685 create proc sp_LoginErrorLog @hrsAgo int
686 as
687 begin
688 select * from LoginErrorLog
689 where ErrorTime >= DATEADD(HOUR, -@hrsAgo, GETDATE())
690 end
691
692 exec sp_LoginErrorLog @hrsAgo=24
```

Results | Messages

	ErrorLogID	ErrorTime	FailedTransactionXML
1	9	2024-11-13 18:19:27.890	<Error><Reason>Account Locked</Reason></Error>
2	11	2024-11-13 18:22:07.503	<Error><Reason>Account Locked</Reason></Error>

Q9. Create a stored procedure that takes a deposit as a parameter and updates the CurrentBalance value for that particular account.

```
730 create proc sp_UpdateCurrentBalanceDep
731     @AccountID int,
732     @AmtDeposited money,
733     @UpdatedBalance money out
734 as
735 begin
736     --check whether the entered account number is avalid pne
737     if exists (select * from Account where AccountID = @AccountID)
738         begin
739         --updating current balance for the acount
740         update Account
741         set CurrentBalance = CurrentBalance + @AmtDeposited
742         where AccountID = @AccountID
743         --retrieve the updated balance(or new balance)
744             begin
745             select @UpdatedBalance = CurrentBalance from Account
746             where AccountID = @AccountID
747             --message to print
748             print 'Deposit successful. Your updated current balance in account No. ' + convert(nvarchar(20), @AccountID) +
749                    ' is $.' + convert(nvarchar(20), @UpdatedBalance)
750             end
751         end
752     else
753         --if the entered account number is an invalid one, then inform the customer with the below message
754         begin
755         print 'Invalid acount number. ' + 'Please check the account number: ' + convert(nvarchar(20), @AccountID) +' and try again.'
756         end
757 end
758 go
```

Q9. -/Contid

```
760   --sample for valid account number
761   declare @output1 money
762   exec sp_UpdateCurrentBalanceDep 11,125, @output1 output
```

Messages

```
(1 row affected)
Deposit successful. Your updated current balance in account No. 11 is $.405.00

Completion time: 2024-11-13T21:22:17.2337511-05:00
```

Results | Messages

	AccountID	CurrentBalance	AccountTypeID	AccountStatusTypeID	InterestSavingsRateID
1	1	5100	1	1	1
2	2	925	2	1	2
3	3	15000	1	1	1
4	4	2000	2	1	2
5	5	3000	5	1	5
6	6	300	1	2	1
7	7	446	2	1	2
8	8	200	1	1	1
9	9	160	1	1	1
10	10	955	1	1	1
11	11	280	2	1	2

```
765   --sample for invalid account number
766   declare @output2 money
767   exec sp_UpdateCurrentBalanceDep 12,150, @output2 output
```

Messages

```
Invalid acount number. Please check the account number: 12 and try again.

Completion time: 2024-11-13T21:23:05.4498305-05:00
```

Results | Messages

	AccountID	CurrentBalance	AccountTypeID	AccountStatusTypeID	InterestSavingsRateID
1	1	5100	1	1	1
2	2	925	2	1	2
3	3	15000	1	1	1
4	4	2000	2	1	2
5	5	3000	5	1	5
6	6	300	1	2	1
7	7	446	2	1	2
8	8	200	1	1	1
9	9	160	1	1	1
10	10	955	1	1	1
11	11	405	2	1	2

Q10. Create a stored procedure that takes a withdrawal amount as a parameter and updates

```
783 create proc sp_UpdateCurrentBalanceForWithdrawal
784     @AccountID INT,                     --Input
785     @AmtWithdraw money,                 --Input
786     @UpdatedCurrentBalance money out    --Output
787 as
788 begin
789     --Check if the account exists
790     if exists (select * from Account where AccountID = @AccountID)
791         begin
792         --Check if the account has sufficient balance
793         if (select CurrentBalance from Account where AccountID = @AccountID)>=@AmtWithdraw
794             begin
795             -- Update the CurrentBalance value for the specified account
796             update Account
797             set CurrentBalance = CurrentBalance - @AmtWithdraw
798             where AccountID = @AccountID;
799             --retrieve current balance to show it in below message
800             select @UpdatedCurrentBalance = CurrentBalance from Account
801             where AccountID = @AccountID
802             print 'Withdrawal successful. Your updated current balance is: $.'+ convert(nvarchar(20), @UpdatedCurrentBalance)
803             end
804         else
805             begin
806             --retrieve current balance to show it in below message
807             select @UpdatedCurrentBalance = CurrentBalance from Account
808             where AccountID = @AccountID
809             print 'Whthdrawal failed due to insufficent funds. Your current balance in account No. '+
810                    convert(nvarchar(20), @AccountID) +' is: $.'+ convert(nvarchar(20), @UpdatedCurrentBalance)
811             end
812         end
813         else
814             begin
815                 print 'Invalid acount number. ' + 'Please check the account number: ' + convert(nvarchar(20), @AccountID) +' and try again.'
816             end
817 end
818 go
```

Q10. -/Contid

```
820   --sample for valid account number
821   declare @outputW1 money
822   exec sp_UpdateCurrentBalanceForWithdrawal 2,100, @outputW1 output
```

Messages

```
(1 row affected)
Withdrawal successful. Your updated current balance is: $.825.00

Completion time: 2024-11-13T21:27:03.6062953-05:00
```

Results | Messages

	AccountID	CurrentBalance	AccountTypeID	AccountStatusTypeID	InterestSavingsRateID
1	1	5100	1	1	1
2	2	925	2	1	2
3	3	15000	1	1	1
4	4	2000	2	1	2
5	5	3000	5	1	5
6	6	300	1	2	1
7	7	446	2	1	2
8	8	200	1	1	1
9	9	160	1	1	1
10	10	955	1	1	1
11	11	405	2	1	2

```
825   --sample for invalid account number
826   declare @outputW2 money
827   exec sp_UpdateCurrentBalanceForWithdrawal 12,150, @outputW2 output
```

Messages

```
Invalid acount number. Please check the account number: 12 and try again.

Completion time: 2024-11-13T21:27:50.1831678-05:00
```

```
831   declare @outputW3 money
832   exec sp_UpdateCurrentBalanceForWithdrawal 11,15000, @outputW3 output
```

Messages

```
Whthdrawal failed due to insufficent funds. Your current balance in account No. 1 is: $.5100.00

Completion time: 2024-11-13T21:28:23.7637593-05:00
```

Results | Messages

	AccountID	CurrentBalance	AccountTypeID	AccountStatusTypeID	InterestSavingsRateID
1	1	5100	1	1	1
2	2	825	2	1	2
3	3	15000	1	1	1
4	4	2000	2	1	2
5	5	3000	5	1	5
6	6	300	1	2	1
7	7	446	2	1	2
8	8	200	1	1	1
9	9	160	1	1	1
10	10	955	1	1	1
11	11	405	2	1	2

THANK YOU !

BACKUP

Q3. Create a view to get counts of checking and savings accounts by customer.

BACKUP

```
460 create view CheckingandSavingAcbyCustomerView
461 as
462 select
463         c.CustomerFirstName + ' ' + c.CustomerLastName [CustomerFullName],
464         at.AccountTypeDescription,
465         COUNT(ca.AccountID) as [#Accounts]
466 from Customer c
467 left join CustomerAccount ca
468 on c.CustomerID = ca.CustomerID
469 left join Account a
470 on ca.AccountID = a.AccountID
471 left join AccountType at
472 on a.AccountTypeID = at.AccountTypeID
473 where at.AccountTypeDescription in ('Savings','Checking')
474 Group by
475         c.CustomerFirstName,
476         c.CustomerLastName,
477         at.AccountTypeDescription
478
479 select * from CheckingandSavingAcbyCustomerView
```

Results | Messages

	CustomerFullName	AccountTypeDescription	#Accounts
1	Anna Jones	Checking	1
2	Bob Smith	Savings	1
3	George Bush	Checking	1
4	George Bush	Savings	1
5	Jimmy Anderson	Checking	1
6	John Doe	Checking	1
7	John Doe	Savings	3
8	Sean Cooper	Savings	1

Q5. Create a view to get all customers' overdraft amounts.

BACKUP

```
548  create view AllCustomersODAmountsView
549  as
550  select
551          c.CustomerFirstName + ' ' + c.CustomerLastName [CustomerFullName],
552          odl.AccountID,
553          odl.OverDraftAmount
554  from Customer c
555  left join CustomerAccount ca
556  on c.CustomerID = ca.CustomerID
557  right join OverDraftLog odl
558  on ca.AccountID = odl.AccountID
559
560  select * from AllCustomersODAmountsView
```

Results | Messages

	CustomerFullName	AccountID	OverDraftAmount
1	John Doe	1	100.00
2	John Doe	2	200.00
3	George Bush	3	300.00
4	George Bush	4	400.00
5	Dave Miller	5	500.00
6	Sean Cooper	6	600.00
7	Jimmy Anderson	7	700.00