SQL

METRO COLLEGE SCHOOL OF TECHNOLOGY

BY: SARA KHOSRAVI
INSTRUCT: HAMID RAJAEE

DATA SCIENCE METRO COLLEGE OF TECHNOLOGY

SQL PROGRAMMING PROJECT PHASE 1

1. QUESTION/PROBLEM 1CREATE 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]

SQL PROGRAMMINGPROJECT PHASE 2

1. CREATE A VIEW TO GET ALL CUSTOMERS WITH CHECKING ACCOUNT FROM ON PROVINCE. [MODERATE]
2. CREATE A VIEW TO GET ALL CUSTOMERS WITH TOTAL ACCOUNT BALANCE (INCLUDING INTEREST RATE) GREATER THAN 5000. [ADVANCED]
3. CREATE A VIEW TO GET COUNTS OF CHECKING AND SAVINGS ACCOUNTS BY CUSTOMER. [MODERATE]
4. CREATE A VIEW TO GET ANY PARTICULAR USER'S LOGIN AND PASSWORD USING ACCOUNTID. [MODERATE]
5. CREATE A VIEW TO GET ALL CUSTOMERS' OVERDRAFT AMOUNT. [MODERATE]
6. CREATE A STORED PROCEDURE TO ADD "USER_" AS A PREFIX TO EVERYONE'S LOGIN (USERNAME). [MODERATE]
7. CREATE A STORED PROCEDURE THAT ACCEPTS ACCOUNTID AS A PARAMETER AND RETURNS CUSTOMER'S FULL NAME. [ADVANCED]
8. CREATE A STORED PROCEDURE THAT RETURNS ERROR LOGS INSERTED IN THE LAST 24 HOURS. [ADVANCED]
9. CREATE A STORED PROCEDURE THAT TAKES A DEPOSIT AS A PARAMETER AND UPDATES CURRENTBALANCE VALUE FOR THAT PARTICULAR ACCOUNT. [ADVANCED]
10. CREATE A STORED PROCEDURE THAT TAKES A WITHDRAWAL AMOUNT AS A PARAMETER AND UPDATES PREPARE A REPORT TO DESCRIBE THE PROJECT. [MODERATE]PREPARE A PRESENTATION FOR THE PROJECT. [MODERATE]

SQL MEANS STRUCTURED QUERY LANGUAGE

WHAT IS A DATABASE?

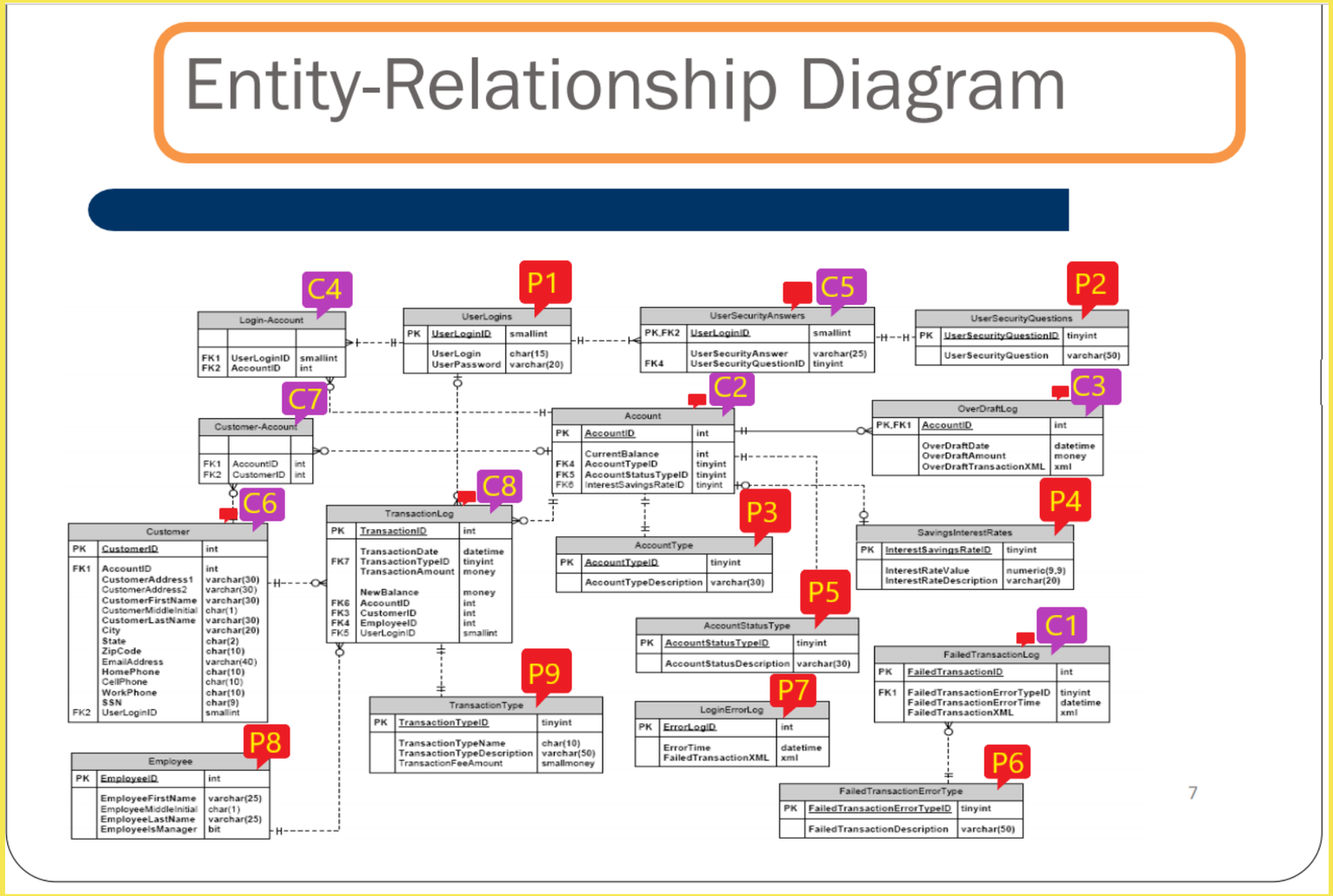
A DATABASE IS AN ORGANIZED COLLECTION OF STRUCTURED INFORMATION, OR DATA, TYPICALLY STORED ELECTRONICALLY IN A COMPUTER SYSTEM. A DATABASE IS USUALLY CONTROLLED BY A DATABASE MANAGEMENT SYSTEM (DBMS).

THE DATABASE IS CREATED BY THE COMMEND IN BELOW:

CREATE DATABASE BANK

BANK DATABASE DESIGN

THE CODE WROTE ON BASED ON THE ARRANGE TABLE BELOW:FIRST, THE TABLE HAS A PRIMARY KEYSECOND, THE TABLE HAS LESS THAN FOREIGN KEY



HOW TO CREATE TABLE WITH PRIMARY KEY AND FOREGIN KEY?

WHAT IS PRIMARY KEY AND FOREIGN KEY?

SQL PRIMARY KEY CONSTRAINTTHE PRIMARY KEY CONSTRAINT UNIQUELY IDENTIFIES EACH RECORD IN A TABLE. PRIMARY KEYS MUST CONTAIN UNIQUE VALUES, AND CANNOT CONTAIN NULL VALUES. A TABLE CAN HAVE ONLY ONE PRIMARY KEY; AND IN THE TABLE, THIS PRIMARY KEY CAN CONSIST OF SINGLE OR MULTIPLE COLUMNS (FIELDS).

SQL FOREIGN KEY CONSTRAINTA FOREIGN KEY IS A KEY USED TO LINK TWO TABLES TOGETHER. A FOREIGN KEY IS A FIELD (OR COLLECTION OF FIELDS) IN ONE TABLE THAT REFERS TO THE PRIMARY KEY IN ANOTHER TABLE. THE TABLE CONTAINING THE FOREIGN KEY IS CALLED THE CHILD TABLE, AND THE TABLE CONTAINING THE CANDIDATE KEY IS CALLED THE REFERENCED OR PARENT TABLE.

CREATE THE TABLES IN THE DATABASE DIAGRAM

BY USING T-SQL STATEMENTS. AN EXAMPLE OF A CODE WRITTEN FOR A TABLE IS GIVEN BELLOW.

IN THE SAME WAY, CODE HAS BEEN WRITTEN FOR 17 TABLE IN THE DIAGARM.

Day5-Stored Proced...ESS.Bank (sa (57)) | Phase2.sql - MKTKH...ESS.Bank (sa (53)) | Bank-Database-Des...ql - not connected

```sql
---**************....PHASE ONE...SARA....****************************************
create database dbBank_SARA


go
use dbBank_SARA
go

--create a table that includes information about UserLogins.....ParentKey 1.........

create table UserLogins
(UserLoginID smallint not null primary key,
UserLogin char(15),
UserPassword varchar(20))
go

--create a table that includes information about UserSecurityQuestions.....ParentKey 2
```

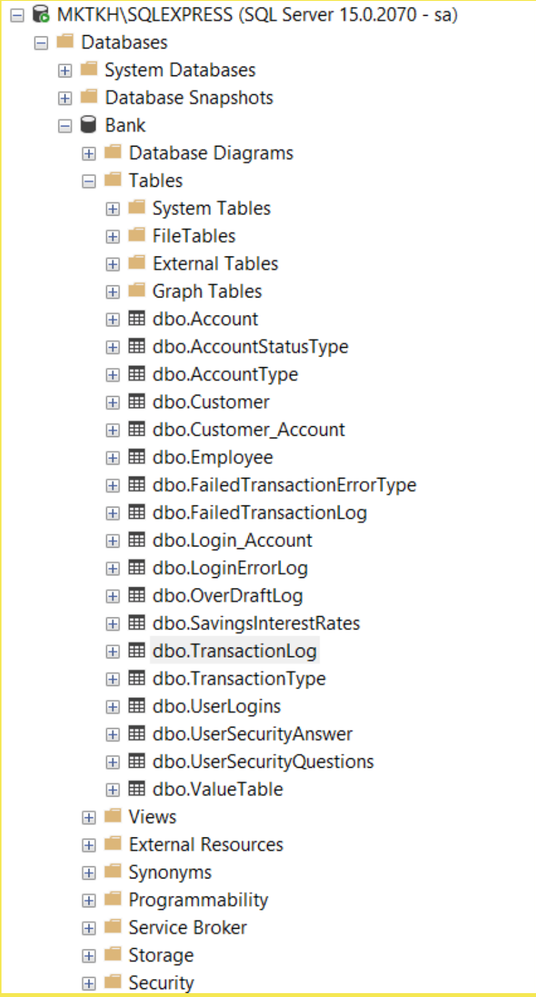
110 %

Results | Messages

	FailedTransactionID	FailedTransactionErrorTypeID	FailedTransactionErrorTime	FailedTransactionXML
1	1	1	2020-12-04 07:30:56.000	First
2	2	2	2020-12-09 12:34:57.000	Second
3	3	3	2020-12-15 02:14:00.000	Third
4	4	4	2020-12-20 05:56:59.000	Fourth
5	5	5	2020-12-21 08:34:15.000	Fifth

BANK DATABASE TABLES:

AFTER COMPLETING THE FIRST PHASE OF THE PROJECT, THE FOLLOWING DATA BANK HAS BEEN CREATED.



2. CREATE A VIEW TO GET ALL CUSTOMERS WITH TOTAL ACCOUNT BALANCE (INCLUDING INTEREST RATE) GREATER THAN 5000. [ADVANCED]

2. CREATE A VIEW TO GET ALL CUSTOMERS WITH TOTAL ACCOUNT BALANCE (INCLUDING INTEREST RATE) GREATER THAN 5000.

Created a view by the name VW_Customer_ACBIR. By selecting (SELECT) records from the customer table which have an account with balances>5000 and showing the sum of their account balance and interest amount. A cross reference inner join (Join) in built between (Customer, Account, and Saving InterestRate)To identify account balance and interest rates.

Tables required to create the view:

SavingInterestRate

Column Name	Data Type	Allow Nulls
InterestSavingRatesID	tinyint	☐
InterestRatesValue	numeric(9, 9)	☐
InterestRatesDescription	varchar(20)	☐
		☐

Account

Column Name	Data Type	Allow Nulls
AccountID	int	☐
CurrentBalance	int	☐
AccountTypeID	tinyint	☐
AccountStatusTypeID	tinyint	☐
InterestSavingRatesID	tinyint	☐
		☐

Customer

Column Name	Data Type	Allow Nulls
CustomerID	int	☐
AccountID	int	☐
CustomerAddress1	varchar(30)	☐
CustomerAddress2	varchar(30)	☑
CustomerFirstName	varchar(30)	☐
CustomerMiddleInitial	char(1)	☑
CustomerLastName	varchar(30)	☐
City	varchar(20)	☐
State	char(2)	☐
ZipCode	char(10)	☐
EmailAddress	char(40)	☐
HomePhone	varchar(10)	☐
CellPhone	varchar(10)	☐
WorkPhone	varchar(10)	☐
SSN	varchar(9)	☑
UserLoginID	smallint	☐
		☐

2. CREATE A VIEW TO GET ALL CUSTOMERS WITH TOTAL ACCOUNT BALANCE (INCLUDING INTEREST RATE) GREATER THAN 5000.

```sql
/*****************************************************************************************************/
/*.....................................................................................................*/
--Question2. Create a view to get all customers with total account balance (including interest rate) greater than 5000. [Advanced]

DROP VIEW VW_Customer_ACBIR;
GO

CREATE VIEW VW_Customer_ACBIR AS
SELECT c.CustomerFirstName, SUM(a.CurrentBalance) AS Ac_Balance, SUM(a.CurrentBalance + (a.CurrentBalance * s.InterestSavingRatesID)/100) AS Total_Ac_Balance
FROM Customer c
JOIN Account a
ON c.AccountID = a.AccountId
JOIN SavingsInterestRates s
ON a.InterestSavingRatesID = s.InterestSavingRatesID
GROUP BY c.CustomerFirstName
HAVING SUM(a.CurrentBalance + (a.CurrentBalance * s.InterestRatesValue)/100) > 5000;
GO
```

100 %

Messages

```
Commands completed successfully.

Completion time: 2020-12-26T15:25:53.7795572-05:00
```

2. CREATE A VIEW TO GET ALL CUSTOMERS WITH TOTAL ACCOUNT BALANCE (INCLUDING INTEREST RATE) GREATER THAN 5000.

```sql
--Question2. Create a view to get all customers with total account balance (including interest rate) greater than 5000. [Advanced]

DROP VIEW VW_Customer_ACBIR;
GO

CREATE VIEW VW_Customer_ACBIR AS
SELECT c.CustomerFirstName, SUM(a.CurrentBalance) AS Ac_Balance, SUM(a.CurrentBalance + (a.CurrentBalance * s.InterestSavingRatesID)/100) AS Total_Ac_Balance
FROM Customer c
JOIN Account a
ON c.AccountID = a.AccountId
JOIN SavingsInterestRates s
ON a.InterestSavingRatesID = s.InterestSavingRatesID
GROUP BY c.CustomerFirstName
HAVING SUM(a.CurrentBalance + (a.CurrentBalance * s.InterestRatesValue)/100) > 5000;
SELECT * FROM VW_Customer_ACBIR
GO
```

100 %

Results | Messages

	CustomerFirstName	Ac_Balance	Total_Ac_Balance
1	Customer1	22000	22220
2	Customer2	25000	25500
3	Customer3	17000	17170
4	Customer4	45000	45900
5	Customer5	35000	35700

3. CREATE A VIEW TO GET COUNTS OF CHECKING AND SAVINGS ACCOUNTS BY CUSTOMER.

3. CREATE A VIEW TO GET COUNTS OF CHECKING AND SAVINGS ACCOUNTS BY CUSTOMER. [MODERATE]

Since we use Aggregate to find the account we need to use group by in SQL Query Joined Tables Customer, Account, and AccountType.

Tables required to create the view:

Account

Column Name	Data Type	Allow Nulls
AccountID	int	☐
CurrentBalance	int	☐
AccountTypeID	tinyint	☐
AccountStatusTypeID	tinyint	☐
InterestSavingRatesID	tinyint	☐
		☐

AccountType

Column Name	Data Type	Allow Nulls
AccountTypeID	tinyint	☐
AccountTypeDescription	varchar(30)	☐
		☐

Customer

Column Name	Data Type	Allow Nulls
CustomerID	int	☐
AccountID	int	☐
CustomerAddress1	varchar(30)	☐
CustomerAddress2	varchar(30)	☑
CustomerFirstName	varchar(30)	☐
CustomerMiddleInitial	char(1)	☑
CustomerLastName	varchar(30)	☐
City	varchar(20)	☐
State	char(2)	☐
ZipCode	char(10)	☐
EmailAddress	char(40)	☐
HomePhone	varchar(10)	☐
CellPhone	varchar(10)	☐
WorkPhone	varchar(10)	☐
SSN	varchar(9)	☑
UserLoginID	smallint	☐
		☐

3. CREATE A VIEW TO GET COUNTS OF CHECKING AND SAVINGS ACCOUNTS BY CUSTOMER. [MODERATE]

```sql
--Question3. Create a view to get counts of checking and savings accounts by customer.
USE BANK
GO

DROP VIEW VW_Customer_ACC;
GO

CREATE VIEW VW_Customer_ACC
AS
SELECT c.CustomerFirstName, at.AccountTypeDescription, COUNT(*) AS Total_AC_Types
FROM Customer c
JOIN Account a
ON c.AccountID = a.AccountId
JOIN AccountType at
ON a.AccountTypeID = at.AccountTypeID
GROUP BY c.CustomerFirstName, at.AccountTypeDescription;
Select * from VW_Customer_ACC;
GO
```

100 %

Messages

```
Commands completed successfully.

Completion time: 2020-12-26T14:42:51.1849594-05:00
```

3. CREATE A VIEW TO GET COUNTS OF CHECKING AND SAVINGS ACCOUNTS BY CUSTOMER. [MODERATE]

```sql
--Question3. Create a view to get counts of checking and savings accounts by customer.
USE BANK
GO

DROP VIEW VW_Customer_ACC;
GO

CREATE VIEW VW_Customer_ACC
AS
SELECT c.CustomerFirstName, at.AccountTypeDescription, COUNT(*) AS Total_AC_Types
FROM Customer c
JOIN Account a
ON c.AccountID = a.AccountId
JOIN AccountType at
ON a.AccountTypeID = at.AccountTypeID
GROUP BY c.CustomerFirstName, at.AccountTypeDescription;
Select * from VW_Customer_ACC;
GO
```

100 %

Results | Messages

	CustomerFirstName	AccountTypeDescription	Total_AC_Types
1	Customer2	CheckingAccount1	1
2	Customer4	CheckingAccount1	1
3	Customer5	CheckingAccount1	1
4	Customer1	SavingAccount1	1
5	Customer3	SavingAccount1	1

6. CREATE A VIEW TO GET ANY PARTICULAR USER'S LOGIN AND PASSWORD USING ACCOUNTID. [MODERATE]

CREATE A VIEW TO GET ANY PARTICULAR USER'S LOGIN AND PASSWORD USING ACCOUNTID. [MODERATE]

- **Created a stored procedure by the name StP_Update_Login**
- **Update table UserLogin and User_ to all UserLogins**
- **Execute the procedure to update the records**

Column Name	Data Type	Allow Nulls
UserLoginID	smallint	☐
UserLogin	varchar(50)	☐
UserPassword	varchar(20)	☐
		☐

```sql
/*****************************************************************************************************/
/*****************.........................PHASE2 .......SARA.........................***************/

--Question6. Create a stored procedure to add "User_" as a prefix to everyone's login (username).[Moderate]
USE BANK;
GO

DROP PROCEDURE StP_Update_Login;
GO

CREATE PROCEDURE StP_Update_Login
AS
UPDATE UserLogins
SET UserLogin = Concat('User_', UserLogin);
GO
EXECUTE StP_Update_Login;
GO

select * from UserLogins;

/*****************************************************************************************************/
```

CREATE A VIEW TO GET ANY PARTICULAR USER'S LOGIN AND PASSWORD USING ACCOUNTID. [MODERATE]

Data in the original Table

	UserLoginID	UserLogin	UserPassword
1	1	UserLogin1	Password1
2	2	UserLogin2	Password2
3	3	UserLogin3	Password3
4	4	UserLogin4	Password4
5	5	UserLogin5	Password5

Data in the original table after add User_

Results | Messages

	UserLoginID	UserLogin	UserPassword
1	1	User_UserLogin1	Password1
2	2	User_UserLogin2	Password2
3	3	User_UserLogin3	Password3
4	4	User_UserLogin4	Password4
5	5	User_UserLogin5	Password5

10. CREATE A STORED PROCEDURE THAT TAKES A WITHDRAWAL AMOUNT AS A PARAMETER AND UPDATES

CREATE A VIEW TO GET ANY PARTICULAR USER'S LOGIN AND PASSWORD USING ACCOUNTID. [MODERATE]

- assigning a name for procedure
- Defining input parameter and it's data type
- When you are using return or output: In Execution time You must assign the result of the procedure in the Temporary variable and then print or select it, so you need declaring that temporary variable and its data type before execution i.e. :1) you have to declare output variables (local variables)2) for seeing the result you have to use print or selectNote: Use return just when output is an integer

```sql
/*****************************************************************************************/
/*.........................................................................................*/
--Question10.Create a stored procedure that takes a withdrawal amount as a parameter and updates

DROP PROCEDURE UpdateCBalanceWithdraw;
GO

CREATE PROCEDURE UpdateCBalanceWithdraw @AccountID INT, @Withdraw INT
AS
UPDATE Account
SET CurrentBalance = CurrentBalance - @Withdraw
WHERE AccountID = @AccountID;
GO

EXEC UpdateCBalanceWithdraw 4, 1000;
GO

SELECT * FROM Account;

/........................................................................./
```

CREATE A VIEW TO GET ANY PARTICULAR USER'S LOGIN AND PASSWORD USING ACCOUNTID. [MODERATE]

Results | Messages

	AccountID	CurrentBalance	AccountTypeID	AccountStatusTypeID	InterestSavingRatesID
1	1	22000	1	1	1
2	2	25000	2	2	2
3	3	17000	1	1	1
4	4	50000	2	2	2
5	5	35000	2	2	2

Results | Messages

	AccountID	CurrentBalance	AccountTypeID	AccountStatusTypeID	InterestSavingRatesID
1	1	22000	1	1	1
2	2	25000	2	2	2
3	3	17000	1	1	1
4	4	49000	2	2	2
5	5	35000	2	2	2

For AccountID 4, Withdraw 1000

THANK YOU FOR YOUR ATTENTION