Final Database Code For Group2

Codes for Aaron Li

Business Rules:

1. Only employees over 21 years of age can sell games rated 'RP'

```
CREATE FUNCTION fn_EmployeeSellRatingPendingGames()
RETURNS INT
AS
BEGIN
	DECLARE @RET INT=0
	IF EXISTS ( SELECT E.EmployeeID FROM tblEMPLOYEE E
				JOIN tblSHIFT S ON E.EmployeeID = S.EmployeeID
				JOIN tblSTORE ST ON ST.StoreID = S.StoreID
				JOIN tblORDER ODR ON ODR.StoreID = ST.StoreID
				JOIN tblLINE_ITEM LI ON LI.OrderID = ODR.OrderID
				JOIN tblGAMES G ON G.GameID = LI.GameID
				JOIN tblAGE_RATING AR ON AR.AgeRatingID =
G.AgeRatingID
			WHERE E.EmployeeBirthDate > DATEADD(YEAR, -21,
GETDATE())
				AND AR.AgeRatingName = 'RP'
			)
	SET @RET = 1
RETURN @RET
END
GO

ALTER TABLE tblORDER
```

```
ADD CONSTRAINT ck_EmpAgeRPGames
CHECK(dbo.fn_EmployeeSellRatingPendingGames()=0)
```

2. No Customers under 10 years old is allowed to place and order for a game that is rated above "10+"

```
ALTER FUNCTION fn_CustomerESRB10()
RETURNS INT
AS
BEGIN
	DECLARE @RET INT=0
	IF EXISTS ( SELECT C.CustomerID FROM tblCUSTOMER C
				JOIN tblORDER ODR ON ODR.CustomerID =
C.CustomerID
				JOIN tblLINE_ITEM LI ON LI.OrderID = ODR.OrderID
				JOIN tblGAMES G ON G.GameID = LI.GameID
				JOIN tblAGE_RATING AGR ON AGR.AgeRatingID =
G.AgeRatingID
			WHERE C.CustomerBirthDate > DATEADD(YEAR, -10,
GETDATE())
				AND (AGR.AgeRatingName = '10+'
					OR AGR.AgeRatingName = 'TEEN'
					OR AGR.AgeRatingName = '17+'
					OR AGR.AgeRatingName = 'ADULT'
					OR AGR.AgeRatingName = 'RP')
			)
	SET @RET = 1
RETURN @RET
END
```

```
GO

ALTER TABLE tblORDER
ADD CONSTRAINT ck_CustomerESRB10
CHECK(dbo.fn_CustomerESRB10()=0)
```

Stored Procedures:

1.

```
CREATE PROCEDURE uspCreateNewDisc
@GameName varchar(35),
@PlatformName varchar(35)
AS
	DECLARE @G_ID INT, @P_ID INT
	SET @G_ID = (SELECT GameID
				FROM tblGAMES
				WHERE @GameName = GameName)
	SET @P_ID = (SELECT PlatformID
				FROM tblPLATFORM
				WHERE @PlatformName = PlatformName)
BEGIN TRAN A2
	INSERT INTO tblDISC(GameID, PlatformID)
	VALUES(@G_ID, @P_ID)
COMMIT TRAN A2
```

2.

```
CREATE PROCEDURE uspCreateNewStore
@CountryName varchar(30),
@StoreName varchar(50),
@StoreAddress varchar(50),
```

```
@StoreCity varchar(30),
@StoreState varchar(10)
As
        DECLARE @C_ID INT
        SET @C_ID = (SELECT CountryID
                                        FROM tblCOUNTRY
                                        WHERE @CountryName = CountryName)

BEGIN TRAN A1
        INSERT INTO tblSTORE(CountryID, StoreName, StoreAddress, StoreCity, StoreState)
        VALUES(@C_ID, @StoreName, @StoreAddress, @StoreCity, @StoreState)
COMMIT TRAN A1
```

Computed Columns:

1.

```
CREATE FUNCTION fn_GamesPerEachAgeRating(@PK INT)
RETURNS INT
AS
BEGIN
        DECLARE @RET INT =
                (SELECT COUNT(G.GameID)
                 FROM tblAGE_RATING ART
                 JOIN tblGAMES G ON G.AgeRatingID = ART.AgeRatingID
                 WHERE ART.AgeRatingID = @PK
                )
RETURN @RET
END
GO
```

```
ALTER TABLE tblAGE_RATING
ADD GamesPerAgeRating AS (dbo.fn_GamesForEachAgeRating(AgeRatingID))
```

2.

```
ALTER FUNCTION fn_GamesPerCustomer(@PK INT)
RETURNS INT
AS
BEGIN
	DECLARE @RET INT =
		(SELECT SUM(LI.Quantity)
		 FROM tblGAMES G
		 JOIN tblLINE_ITEM LI ON LI.GameID = G.GameID
		 JOIN tblORDER ODR ON ODR.OrderID = LI.OrderID
		 JOIN tblCUSTOMER C ON C.CustomerID = ODR.CustomerID
		 WHERE C.CustomerID = @PK
		)
RETURN @RET
END
GO

ALTER TABLE tblCUSTOMER
ADD GamesPerCustomer AS (dbo.fn_GamesPerCustomer(CustomerID))
```

Complex query:

1. Customers older than 10 with at least 2 game item purchases who have also given either a 4 star or 5 star reviews

```
CREATE VIEW [10+4or5StarReviewTotalOrder]
AS
SELECT C.CustomerFName, C.CustomerLName, SUM(LI.Quantity) AS TotalOrders
```

```
FROM tblLINE_ITEM LI
        JOIN tblORDER O ON O.OrderID = LI.OrderID
        JOIN tblCUSTOMER C ON C.CustomerID = O.CustomerID
        JOIN tblREVIEW R ON R.LineItemID = LI.LineItemID
        JOIN tblRATING RT ON RT.RatingID = R.RatingID
WHERE C.CustomerBirthDate < DATEADD(YEAR, -10, GETDATE())
        AND (RT.RatingName LIKE '%4%' OR RT.RatingName LIKE '%5%')
GROUP BY C.CustomerFName, C.CustomerLName
HAVING SUM(LI.Quantity) >= 2
```

2. Developers with games already released priced under $100 that is available on at least 2 game platforms

```
CREATE VIEW DevelopersUnder100Bucks2Platforms AS
SELECT D.DeveloperName, COUNT(P.PlatformID) AS NumPlatforms
FROM tblDEVELOPER D
        JOIN tblDEVELOPER_GAME DG ON DG.DeveloperID = D.DeveloperID
        JOIN tblGAMES G ON G.GameID = DG.GameID
        JOIN tblDISC DC ON DC.GameID = G.GameID
        JOIN tblPLATFORM P ON P.PlatformID = DC.PlatformID
WHERE G.Price < 100
        AND ReleaseDate < GETDATE()
GROUP BY D.DeveloperName
HAVING COUNT(P.PlatformID) >= 2
```

Codes for Yunrui Shao

Business Rules:

1. Purchase limit (each customer can only order at most 3 copies of the same game)

```
CREATE FUNCTION fn_EachAtMost3Copies()

RETURNS INT

AS

BEGIN

  DECLARE @RET INT = 0

  IF EXISTS (SELECT C.CustomerID, G.GameID

        FROM tblCUSTOMER C

        JOIN tblORDER O ON C.CustomerID = O.CustomerID

        JOIN tblLINE_ITEM LI ON O.OrderID = LI.OrderID

        JOIN tblGAMES G ON LI.GameID = G.GameID

        WHERE LI.Quantity > 3

        GROUP BY C.CustomerID, G.GameID)

  SET @RET = 1

RETURN @RET

END

GO

ALTER TABLE tblORDER

ADD CONSTRAINT CK_EachAtMost3Copies

CHECK (dbo.fn_EachAtMost3Copies() = 0)
```

2. Amount of reviews a customer can leave for one game (customer can only leave up to 3 reviews for one game)

```
CREATE FUNCTION fn_EachAtMost3Reviews()

RETURNS INT
```

```
AS

BEGIN

  DECLARE @RET INT = 0

  IF EXISTS (SELECT C.CustomerID, G.GameID, COUNT(R.ReviewID) AS NumReviews

        FROM tblCUSTOMER C

        JOIN tblORDER O ON C.CustomerID = O.CustomerID

        JOIN tblLINE_ITEM LI ON O.OrderID = LI.OrderID

        JOIN tblGAMES G ON LI.GameID = G.GameID

        JOIN tblREVIEW R ON G.GameID = R.ReviewID

        GROUP BY C.CustomerID, G.GameID

HAVING COUNT(R.ReviewID) >= 3)

  SET @RET = 1

RETURN @RET

END

GO

ALTER TABLE tblREVIEW

ADD CONSTRAINT CK_EachAtMost3Reviews

CHECK (dbo.fn_EachAtMost3Reviews() = 0)
```

Stored Procedures:

1. Insert a new row in Employee

```
CREATE PROCEDURE usp_NewRowEmp
@EmpFname varchar(20),
@EmpLname varchar(20),
@EmpBirth DATE,
@EmpTypeN varchar(50),
@BeginD DATE,
@EndD DATE
AS
   DECLARE @E_ID INT, @ET_ID INT
   SET @E_ID = (SELECT E.EmployeeID
```

```
            FROM tblEMPLOYEE E
            WHERE E.EmployeeFName = @EmpFname
            AND E.EmployeeLName = @EmpLname
            AND E.EmployeeBirthDate = @EmpBirth)
   SET @ET_ID = (SELECT EmployeeTypeID FROM tblEMPLOYEE_TYPE WHERE EmployeeTypeName =
@EmpTypeN)
BEGIN TRAN YS1
INSERT INTO tblEMPLOYEE(EmployeeID, EmployeeTypeID, BeginDate, EndDate)
VALUES(@E_ID, @ET_ID, @BeginD, @EndD)
COMMIT TRAN YS1
GO
```

2. Insert a new row in Store

```
CREATE PROCEDURE usp_NewRowStore
@StoreN varchar(50),
@Address varchar(60),
@City varchar(25),
@State varchar(20),
@CountryN varchar(30)
AS
   DECLARE @S_ID INT, @C_ID INT
   SET @S_ID = (SELECT StoreID
            FROM tblSTORE
            WHERE StoreName = @StoreN
            AND StoreAddress = @Address
            AND StoreCity = @City
            AND StoreState = @State)
   SET @C_ID = (SELECT CountryID FROM tblCOUNTRY WHERE CountryName = @CountryN)
BEGIN TRAN YS2
INSERT INTO tblSTORE(StoreID, CountryID)
VALUES(@S_ID, @C_ID)
COMMIT TRAN YS2
GO
```

Computed Columns:

1. Calculate the number of employees under each shift type

```
USE INFO330_PROJ_A2

CREATE FUNCTION fn_CalcNumEmpEachShiftType(@PK INT)
RETURNS INT
AS
   BEGIN
      DECLARE @RET INT = (SELECT COUNT(E.EmployeeID)
                  FROM tblEMPLOYEE E
                     JOIN tblEMPLOYEE_TYPE ET ON E.EmployeeTypeID = ET.EmployeeTypeID
                  WHERE ET.EmployeeTypeID = @PK)
      RETURN @RET
```

```
  end
GO

ALTER TABLE tblEMPLOYEE_TYPE
ADD CalNumEmpEachShiftType AS (dbo.fn_CalcNumEmpEachShiftType(EmployeeTypeID))
GO
```

2. Number of games developed by each country

```
CREATE FUNCTION fn_CalcNumGamesByCountry(@PK INT)
RETURNS INT
AS
  BEGIN
    DECLARE @RET INT = (SELECT COUNT(G.GameID)
                FROM tblGAMES G
                  JOIN tblDEVELOPER_GAME DG ON G.GameID = DG.GameID
                  JOIN tblDEVELOPER D ON DG.DeveloperID = D.DeveloperID
                  JOIN tblCOUNTRY C ON D.CountryID = C.CountryID
                WHERE C.CountryID = @PK)
    RETURN @RET
  end
GO

ALTER TABLE tblCOUNTRY
ADD CalcNumGamesByCountry AS (dbo.fn_CalcNumGamesByCountry(CountryID))
GO
```

3. Number of Employees in each Store

```
CREATE FUNCTION fn_CalcNumEmpPerStore(@PK INT)
RETURNS INT
AS
  BEGIN
    DECLARE @RET INT = (SELECT COUNT(E.EmployeeID)
                FROM tblEMPLOYEE E
                  JOIN tblSHIFT SF ON E.EmployeeID = SF.EmployeeID
                  JOIN tblSTORE S ON SF.StoreID = S.StoreID
                WHERE S.StoreID = @PK)
    RETURN @RET
  end
go

ALTER TABLE tblSTORE
ADD CalcNumEmpPerStore AS (dbo.fn_CalcNumEmpPerStore(StoreID))
GO
```

Complex query:

1. Determine which customers have ordered more than 5 games between January 1st, 2018 and July 12, 2019 from the store named ‘GameStop’ who have also left greater than 3 reviews in total for games having a rating name ‘5 star’.

```
SELECT C.Customer, subq1.TotalReview, SUM(LI.Quantity) AS TotalGamesBought
FROM tblCUSTOMER C
    JOIN tblORDER O ON C.CustomerID = O.CustomerID
    JOIN tblLINE_ITEM LI ON O.OrderID = LI.OrderID
    JOIN tblSTORE S ON O.StoreID = S.StoreID
    JOIN (SELECT C.Customer, COUNT(RW.ReviewID) AS TotalReview
        FROM tblCUSTOMER C
            JOIN tblORDER O ON C.CustomerID = O.CustomerID
            JOIN tblLINE_ITEM LI ON O.OrderID = LI.OrderID
            JOIN tblREVIEW RW ON LI.Line_ItemID = RW.Line_ItemID
            JOIN tblRATING RT ON RW.RatingID = RT.RatingID
        WHERE RT.RatingName = ‘5 star’
        GROUP BY C.CustomerID
        HAVING COUNT(RW.ReviewID) > 3) AS subq1 ON C.CustomerID =
subq1.CustomerID
WHERE O.OrderDate BETWEEN ‘2018-01-01’ AND ‘2019-07-12’
AND S.StoreName = ‘GameStop’
GROUP BY C.Customer, subq1.TotalReview
HAVING SUM(LI.Quantity) > 5
```

2. Determine which employees younger than 25 years old have sold more than 50 games before August 5, 2015 who have also served more than 20 customers in state ‘Washington’.

```
SELECT E.EmployeeID, subq1.NumCustServed , SUM(LI.Quantity) AS TotalGamesSold
FROM EMPLOYEE E
    JOIN SHIFT SFT ON E.EmployeeID = SFT.EmployeeID
    JOIN STORE S ON SFT.StoreID = S.StoreID
```

```
    JOIN ORDER O ON S.StoreID = O.StoreID
    JOIN LINE_ITEM LI ON O.OrderID = LI.OrderID
    JOIN (SELECT E.EmployeeID, COUNT(C.CustomerID) AS NumCustServed
        FROM EMPLOYEE E
            JOIN SHIFT SFT ON E.EmployeeID = SFT.EmployeeID
            JOIN STORE S ON SFT.StoreID = S.StoreID
            JOIN ORDER O ON S.StoreID = O.StoreID
            JOIN CUSTOMER C ON O.CustomerID = C.CustomerID
        WHERE C.CustomerState = 'Washington'
        GROUP BY E.EmployeeID
        HAVING COUNT(C.CustomerID) > 20) AS subq1 ON E.EmployeeID =
subq1.EmployeeID
WHERE E.EmployeeBirthDate > DateAdd(year, -25, GetDate())
AND O.OrderDate < '2015-08-05'
GROUP BY E.EmployeeID, subq1. NumCustServed
HAVING SUM(LI.Quantity) > 50
```

Code For Ian O'Brien

Business Rules:

1. A customer cannot purchase a game that has the disc condition of 'scratched'

```
CREATE FUNCTION fn_CustCannotPurchaseScratchedDisk()
RETURNS INT
AS
BEGIN
  DECLARE @RET INT = 0
  IF EXISTS (SELECT *
    FROM tblCUSTOMER C
      JOIN tblORDER O ON C.CustomerID = O.CustomerID
      JOIN tblLINE_ITEM LI on O.OrderID = LI.OrderID
      JOIN tblGAMES G on LI.GameID = G.GameID
      JOIN tblDISC D on G.GameID = D.GameID
      JOIN tblDISC_CONDITION DC on D.DiscID = DC.DiscID
      JOIN tblCONDITION CO on DC.ConditionID = CO.ConditionID
    WHERE ConditionName = 'Scratched'
    )
  BEGIN
  SET @RET = 1
  END
RETURN @RET
END
GO
ALTER TABLE tblORDER
ADD CONSTRAINT CK_NoPurchaseScratchedDisk
CHECK (dbo.fn_CustCannotPurchaseScratchedDisk() = 0)
```

2. A FPS game cannot be sold on platform PC to a customer under 13

```
CREATE FUNCTION fn_FPSSoldUnder13OnPC()
RETURNS INT
AS
BEGIN
  DECLARE @RET INT = 0
  IF EXISTS (SELECT *
  FROM tblGAMES G
    JOIN tblGAME_GENRE tGG on G.GameID = tGG.GameID
    JOIN tblGENRE tG on tGG.GenreID = tG.GenreID
    JOIN tblDISC tD on G.GameID = tD.GameID
    JOIN tblPLATFORM tP on tD.PlatformID = tP.PlatformID
    JOIN tblLINE_ITEM tLI on G.GameID = tLI.GameID
```

```
    JOIN tblORDER t on tLI.OrderID = t.OrderID
    JOIN tblCUSTOMER tC on t.CustomerID = tC.CustomerID
  WHERE GenreName = 'FPS'
    AND CustomerBirthDate < DATEADD(YEAR, -13, GETDATE())
    AND PlatformName = 'PC')
  BEGIN
    SET @RET = 0
  END
RETURN @RET
END
GO
ALTER TABLE tblORDER
ADD CONSTRAINT CK_EmployeeCanOnlyWorkOneStoreLocation
CHECK (dbo.fn_FPSSoldUnder13OnPC() = 0)
```

Computed Columns:

1. Calculate the number of orders at each store

```
CREATE FUNCTION fn_OrdersPerStore(@PK INT)
RETURNS INT
AS
BEGIN
  DECLARE @RET INT =
  (SELECT COUNT(OrderID)
    FROM tblORDER
    JOIN tblSTORE tS on tblORDER.StoreID = tS.StoreID
    WHERE tS.StoreID = @PK)
  RETURN @RET
end
GO
ALTER TABLE tblSTORE
ADD OrdersPerStore AS (dbo.fn_OrdersPerStore(StoreID))
```

2. Calculate the number of reviews per game

```
CREATE FUNCTION fn_NumReviewsPerGame(@PK INT)
RETURNS INT
AS
BEGIN
  DECLARE @RET INT =
  (SELECT COUNT(tR.ReviewID)
  FROM tblGAMES G
    JOIN tblLINE_ITEM LI on G.GameID = LI.GameID
    JOIN tblREVIEW tR on LI.LineItemID = tR.LineItemID
  WHERE G.GameID = @PK
```

```
  )
RETURN @RET
end
GO
ALTER TABLE tblGAMES
ADD ReviewsPerGame AS (dbo.fn_NumReviewsPerGame(GameID))
```

3. Number of games purchased with genre ‘MOBA’ by customer

```
CREATE FUNCTION fn_OrdersPerCust(@PK INT)
RETURNS INT
AS
BEGIN
 DECLARE @RET INT =
   (SELECT COUNT(t.OrderID)
   FROM tblCUSTOMER C
     JOIN tblORDER t on C.CustomerID = t.CustomerID
      JOIN tblLINE_ITEM tLI on t.OrderID = tLI.OrderID
   WHERE C.CustomerID = @PK
   )
RETURN @RET
END
GO
ALTER TABLE tblCUSTOMER
ADD OrdersPerCustomer AS (dbo.fn_OrdersPerCust(CustomerID))
```

Stored Procedures:

1. Insert a new row into the order table, requires the unique identifiers for a customer and the store name as well as an order date.

```
CREATE PROCEDURE USPianoNewOrder
@F varchar(20),
@L varchar(30),
@B Date,
@S varchar(40),
@O date
AS
DECLARE @C_ID INT, @S_ID INT

SET @C_ID = (SELECT CustomerID FROM tblCUSTOMER
  WHERE CustomerFName = @F
  AND CustomerLName = @L
  AND CustomerBirthDate = @B)
SET @S_ID = (SELECT StoreID FROM tblSTORE
  WHERE StoreName = @S)
```

```
BEGIN TRAN O2
INSERT INTO tblORDER (StoreID, CustomerID, OrderDate)
VALUES (@S_ID, @C_ID, @O)
COMMIT TRAN O2
```

2. Insert a new row into the line item table, needs the game name and the order number for confirmation also a user specified quantity

```
CREATE PROCEDURE USPianoNewLineItem
@Order INT,
@G varchar(40),
@Q INT
AS
DECLARE @O_ID INT, @G_ID INT

SET @G_ID = (SELECT GameID FROM tblGAMES
  WHERE GameName = @G)
SET @O_ID = (SELECT OrderID FROM tblORDER
  WHERE OrderID = @Order)

BEGIN TRAN O2
INSERT INTO tblLINE_ITEM(OrderID, GameID, Quantity)
VALUES (@O_ID, @G_ID, @Q)
COMMIT TRAN O2
```

Complex Queries:

1. Determine which developers have more than 3 games after October 11, 2014, who also have created 2 games of genre 'FPS' before January 01, 2016

```
SELECT D.DeveloperID, D.DeveloperName, NumFPSGames, COUNT(*) AS NumGamesAfter2014
FROM tblDEVELOPER D
  JOIN tblDEVELOPER_GAME tDG on D.DeveloperID = tDG.DeveloperID
  JOIN tblGAMES tG on tDG.GameID = tG.GameID
  JOIN (
    SELECT D.DeveloperID, COUNT(*) AS NumFPSGames
    FROM tblDEVELOPER D
      JOIN tblDEVELOPER_GAME DG on D.DeveloperID = DG.DeveloperID
      JOIN tblGAMES G on DG.GameID = G.GameID
      JOIN tblGAME_GENRE GG ON G.GameID = GG.GameID
      JOIN tblGENRE GR ON GG.GenreID = GR.GenreID
    WHERE ReleaseDate < 'January 01, 2016'
      AND GenreName = 'FPS'
    GROUP BY D.DeveloperID
```

```
    HAVING COUNT(*) > 2) AS subq1 ON D.DeveloperID = subq1.DeveloperID
WHERE tG.ReleaseDate > 'October 11, 2014'
GROUP BY D.DeveloperID, D.DeveloperName, NumFPSGames
HAVING COUNT(*) > 3
```

2. Determine which game has more than 10 reviews,which also is of genre 'MOBA' and has been sold to at least 15 customers, who are over the age of 21

```
SELECT G.GameID, G.GameName, NumOfCustomersPurchased, COUNT(*) AS NumReviews
FROM tblGAMES G
  JOIN tblLINE_ITEM tLI on G.GameID = tLI.GameID
  JOIN tblREVIEW tR on tLI.LineItemID = tR.LineItemID
  JOIN (
    SELECT G.GameID, COUNT(*) AS NumOfCustomersPurchased
    FROM tblGAMES G
      JOIN tblLINE_ITEM t on G.GameID = t.GameID
      JOIN tblORDER O on t.OrderID = O.OrderID
      JOIN tblCUSTOMER tC on O.CustomerID = tC.CustomerID
      JOIN tblGAME_GENRE tGG on G.GameID = tGG.GameID
      JOIN tblGENRE tG on tGG.GenreID = tG.GenreID
    WHERE tG.GenreName = 'MOBA'
      AND CustomerBirthDate < DATEADD(YEAR, -21, GETDATE())
    GROUP BY G.GameID
    HAVING COUNT(*) >= 15) AS subq1 ON G.GameID = subq1.GameID
GROUP BY G.GameID, G.GameName, NumOfCustomersPurchased
HAVING COUNT(*) > 10
```

Code For Soham Hinduja:

Computed Columns:

```
-- Computed Column 1
-- Calculate the number of awards per game
CREATE FUNCTION fn_CalcTotalAwards(@PK INT)
RETURNS int
AS
BEGIN
  DECLARE @RET int =
    (  SELECT COUNT(Game_AwardID)
       FROM tblGAMES G
          JOIN tblGAME_AWARD GA on G.GameID = GA.GameID
       WHERE G.GameID = @PK
    )
RETURN @RET
END
GO

ALTER TABLE tblGAMES
ADD TotalAwards AS (dbo.fn_CalcTotalAwards(GameID))

-- Computed Column 2.
-- Calculate the number of games developed by each developer
CREATE FUNCTION fn_CalcTotalDevGames(@PK INT)
RETURNS int
AS
BEGIN
  DECLARE @RET int =
    (  SELECT COUNT(G.GameID)
       FROM tblGAMES G
          JOIN tblDEVELOPER_GAME DG on G.GameID = DG.GameID
          JOIN tblDEVELOPER D on DG.DeveloperID = D.DeveloperID
       WHERE D.DeveloperID = @PK
    )
RETURN @RET
END
GO
```

```
ALTER TABLE tblDEVELOPER
ADD TotalGames AS (dbo.fn_CalcTotalDevGames(DeveloperID))
```

Stored Procedures:

```
-- Stored Procedure 1
-- Adding a Game Award
CREATE PROCEDURE uspINSERTGameAward
@GName varchar(75),
@AName varchar(50),
@YEAR char(4)
AS

DECLARE @G_ID INT, @A_ID INT,
SET @G_ID = (   SELECT GameID
            FROM tblGAMES
            WHERE GameName = @GName
        )
SET @A_ID = (   SELECT AwardID
            FROM tblAWARDS
            WHERE AwardName = @AName
        )

BEGIN TRANSACTION S1
INSERT INTO tblGAME_AWARD (GameID, AwardID, YEAR)
VALUES (@G_ID, @A_ID, @YEAR)
COMMIT TRANSACTION S1

-- Stored Procedure 2
-- Adding into Game_Genre
CREATE PROCEDURE uspINSERTGameGenre
@GName varchar(75),
@GEName varchar(50),
AS

DECLARE @GE_ID INT, @G_ID INT
SET @GE_ID = (  SELECT GenreID
            FROM tblGENRE
            WHERE GenreName = @GEName
        )
```

```
SET @G_ID = (   SELECT GameID
            FROM tblGAMES
            WHERE GameName = @GName
          )

BEGIN TRANSACTION S2
INSERT INTO tblGAME_GENRE (GenreID, GameID)
VALUES (@GE_ID, @G_ID)
COMMIT TRANSACTION S2
```

Business Rules:

```
-- BUSINESS RULE 1
  --A customer from 'Washington' cannot purchase a game from a developer  in country 'Canada'
CREATE FUNCTION fn_WACustomerNoCanada()
RETURNS INT
AS
BEGIN
  DECLARE @Ret INT = 0
  IF EXISTS ( SELECT *
            FROM tblCUSTOMER C
               JOIN tblORDER O on C.CustomerID = O.CustomerID
               JOIN tblLINE_ITEM LI on O.OrderID = LI.OrderID
               JOIN tblGAMES G on LI.GameID = G.GameID
               JOIN tblDEVELOPER_GAME DG on G.GameID = DG.GameID
               JOIN tblDEVELOPER D on DG.DeveloperID = D.DeveloperID
               JOIN tblCOUNTRY CO on D.CountryID = CO.CountryID
            WHERE C.CustomerState = 'Washington'
            AND CO.CountryName = 'Japan'
            )
  SET @Ret = 1
RETURN @RET
END
ALTER TABLE tblORDER WITH NOCHECK -- assuming for all new entries
ADD CONSTRAINT CK_WACustomerNoCanada
CHECK (dbo.fn_WACustomerNoCanada() = 0)

--BUSINESS RULE 2
  --No one from Zip 98103 can order more than 2 copies of 'Halo 3'
CREATE FUNCTION fn_98103NoMoreThan2Halo3()
```

```
RETURNS INT
AS
BEGIN
  DECLARE @Ret INT = 0
  IF EXISTS ( SELECT *
            FROM tblCUSTOMER C
               JOIN tblORDER O on C.CustomerID = O.CustomerID
               JOIN tblLINE_ITEM LI on O.OrderID = LI.OrderID

          JOIN tblGAMES G on LI.GameID = G.GameID
            WHERE C.CustomerPostal = '98103'
            AND G.GameName = 'Halo 3'
            GROUP BY C.CustomerID
            HAVING COUNT(O.OrderID) > 2
            )
  SET @Ret = 1
RETURN @RET
END

ALTER TABLE tblORDER WITH NOCHECK -- assuming for all new entries
ADD CONSTRAINT CK_98103CantOrderMoreThan2Halo3
CHECK (dbo.fn_98103NoMoreThan2Halo3() = 0)
```

Complex Queries:

Select a game that has been ordered more than 20 times which has also won an award for game of the year after 2017

```
SELECT G.GameID, G.GameName, subq1.TotalOrders, tGA.YEAR
FROM tblGAMES G
  JOIN tblGAME_AWARD tGA on G.GameID = tGA.GameID
  JOIN tblAWARDS tA on tGA.AwardID = tA.AwardID
  JOIN    (   Select G.GameID, COUNT(t.OrderID) AS 'TotalOrders'
            From tblGAMES G
               JOIN tblLINE_ITEM tLI on G.GameID = tLI.GameID
               JOIN tblORDER t on tLI.OrderID = t.OrderID
            GROUP BY G.GameID
            Having COUNT(t.OrderID) >2

          )AS subq1 ON G.GameID = subq1.GameID
WHERE tA.AwardName = 'Game of the Year'
```

```
AND tGA.YEAR > '2017'
```

Which Employees younger than 30 years have worked in more than 3 stores in the united states who have also served more than 30 customers in the postal code 98102

```
Select E.EmployeeID, Count(C.CustomerID) AS 'Customers Served', subq1.TotalStoresWorked
FROM tblEMPLOYEE E
  JOIN tblSHIFT S on E.EmployeeID = S.EmployeeID
  JOIN tblSTORE ST on S.StoreID = ST.StoreID
  JOIN tblORDER O on ST.StoreID = O.StoreID
  JOIN tblCUSTOMER C on O.CustomerID = C.CustomerID
  JOIN    (   SELECT E.EmployeeID, COUNT(S.StoreID) AS 'TotalStoresWorked'
            FROM tblEMPLOYEE E
               JOIN tblSHIFT tS on E.EmployeeID = tS.EmployeeID
               JOIN tblSTORE S on tS.StoreID = S.StoreID
               JOIN tblCOUNTRY tC on S.CountryID = tC.CountryID
            WHERE E.EmployeeBirthDate > DATEADD(year, -30, getdate())
            AND tc.CountryName = 'United States'
            GROUP BY E.EmployeeID
            HAVING COUNT(S.StoreID) > 3
         ) AS subq1 on E.EmployeeID = subq1.EmployeeID
WHERE C.CustomerPostal = '98102'

GROUP BY E.EmployeeID
HAVING COUNT(C.CustomerID) > 30
```