1. Mô hình thực thể quan hệ ERD

2. Các Stored Procedure:

/\*==============================================================\*/

/\*                        Proc                                  \*/

/\*==============================================================\*/

CREATE PROCEDURE [dbo].[search\_promotion\_by\_code]

    @code NVARCHAR(50)

AS

BEGIN

    SELECT [ID], [CODE], [DISCOUNT], [STATUS], [NOTE]

    FROM [dbo].[PROMOTION]

    WHERE LOWER([CODE]) like '%' +  LOWER(@code) +  '%'

END;

go

----------------------------------------------------- Tìm kiếm sản phẩm

CREATE proc [dbo].[search\_product] @DataToFind Nvarchar(100)

AS

Select \*   From   PRODUCT

Where  LOWER(NAME) Like '%' + LOWER(@DataToFind) + '%'

go

CREATE PROCEDURE [dbo].[search\_foodcombo]

    @searchTerm NVARCHAR(100)

AS

BEGIN

    SELECT

        fc.ID AS ID,

        fc.NAME AS NAME,

        MAX(CASE WHEN p.TYPE = 'FOOD' THEN p.NAME ELSE NULL END) AS FOOD,

        STUFF((SELECT ',' + CAST(pf.QUANTITY AS VARCHAR)

               FROM product\_fcb pf

               JOIN product p2 ON pf.PRODUCT\_ID = p2.ID

               WHERE p2.TYPE = 'FOOD' AND pf.FCB\_ID = fc.ID

               FOR XML PATH('')), 1, 1, '') AS FOOD\_QUANTITY,

        MAX(CASE WHEN p.TYPE = 'DRINK' THEN p.NAME ELSE NULL END) AS DRINK,

        STUFF((SELECT ',' + CAST(pf.QUANTITY AS VARCHAR)

               FROM product\_fcb pf

               JOIN product p2 ON pf.PRODUCT\_ID = p2.ID

               WHERE p2.TYPE = 'DRINK' AND pf.FCB\_ID = fc.ID

               FOR XML PATH('')), 1, 1, '') AS DRINK\_QUANTITY,

        fc.PRICE AS PRICE,

        CAST(fc.Image AS NVARCHAR(MAX)) as IMAGE

    FROM foodcombo fc

    JOIN product\_fcb pf ON fc.ID = pf.FCB\_ID

    JOIN product p ON pf.PRODUCT\_ID = p.ID

    WHERE fc.NAME LIKE '%' + @searchTerm + '%'

    GROUP BY fc.ID, fc.NAME, fc.PRICE, CAST(fc.Image AS NVARCHAR(MAX))

END;

go

------------------------------------------------------------

CREATE PROCEDURE [dbo].[get\_movies\_in\_schedule]

AS

BEGIN

    SELECT M.[ID], M.[GENRE], M.[DURATION], M.[RATING], M.[STORY], M.[POSTER], M.[OPENING\_DAY], M.[CLOSING\_DAY], M.[TITLE]

    FROM [dbo].[MOVIE] AS M

    JOIN [dbo].[SCHEDULE] AS S ON M.[ID] = S.[MOVIE\_ID]

END;

go

-----------------------------------------------------Đổi mật khẩu

CREATE PROCEDURE change\_password

    @StaffID INT,

    @OldPassword VARCHAR(50),

    @NewPassword VARCHAR(50)

AS

BEGIN

    -- Check if the old password is correct

    IF EXISTS (SELECT 1 FROM STAFF WHERE ID = @StaffID AND PASSWORD = @OldPassword)

    BEGIN

        -- Update the password

        UPDATE STAFF

        SET PASSWORD = @NewPassword

        WHERE ID = @StaffID;

        SELECT 1 AS Result; -- Password changed successfully

    END

    ELSE

    BEGIN

        SELECT 0 AS Result; -- Incorrect old password

    END;

END;

go

--drop proc search\_product

go

----------------------------------------------------- Login

CREATE PROCEDURE Login (@email varchar(30),@password varchar(30))

As

   SELECT \* FROM Staff WHERE BINARY\_CHECKSUM(EMAIL) = BINARY\_CHECKSUM(@email)

    AND BINARY\_CHECKSUM(PASSWORD) = BINARY\_CHECKSUM(@password)

GO

--DROP PROC LOGIN

--exec Login 'admin', 'admin'

go

----------------------------------------------------- Tìm kiếm nhân viên

CREATE PROCEDURE search\_staff

    @DataToFind NVARCHAR(100)

AS

BEGIN

    SELECT \*

    FROM STAFF

    WHERE LOWER(EMAIL) LIKE '%' + LOWER(@DataToFind) + '%'

        OR LOWER(PHONE) LIKE '%' + LOWER(@DataToFind) + '%'

        OR LOWER(FIRSTNAME) LIKE '%' + LOWER(@DataToFind) + '%'

        OR LOWER(LASTNAME) LIKE '%' + LOWER(@DataToFind) + '%'

        OR LOWER(ADDRESS) LIKE '%' + LOWER(@DataToFind) + '%'

END

go

---------------------------------------------------Tìm kiếm phim

CREATE PROCEDURE [dbo].[Search\_movie]

   @DataFind Nvarchar(50)

AS

BEGIN

    SELECT [ID], [TITLE], [GENRE], [DURATION], [RATING], [STORY], [POSTER], [OPENING\_DAY], [CLOSING\_DAY]

    FROM [IIEXCinema].[dbo].[MOVIE]

    WHERE LOWER(CAST(movie.TITLE AS NVARCHAR(MAX))) LIKE '%' + LOWER(@DataFind) + '%'

    OR LOWER([GENRE]) LIKE '%' + LOWER(@DataFind) + '%'

END

go

----------------------------------------------------------------------------- Tìm lịch chiếu phim

CREATE PROCEDURE [dbo].[search\_schedule]

    @SHOWROOM\_id INT,

    @keyword Nvarchar(60)

AS

SELECT

    schedule.ID,

    SHOWROOM.SHOWROOMNUMBER,

    CAST(MOVIE.POSTER AS varchar(MAX)) AS POSTER,

    CAST(movie.TITLE AS NVARCHAR(MAX)) AS TITLE,

    movie.DURATION,

    SCHEDULE.STARTTIME,

    SCHEDULE.ENDTIME,

    SCHEDULE.PRICE

FROM

    schedule

INNER JOIN

    SHOWROOM ON schedule.SHOWROOM\_ID = SHOWROOM.ID

INNER JOIN

    movie ON movie.ID = schedule.MOVIE\_ID

WHERE

    SHOWROOM.ID = @SHOWROOM\_id

    AND (LOWER(CAST(movie.TITLE AS NVARCHAR(MAX))) LIKE '%' + LOWER(@keyword) + '%' OR LOWER(CAST(MOVIE.POSTER AS varchar(MAX))) LIKE '%' + LOWER(@keyword) + '%')

ORDER BY

    schedule.STARTTIME DESC

go

---------------------------------------------------------- Tạo phòng chiếu

CREATE PROCEDURE create\_showroom

  @SHOWROOM\_num INT

AS

BEGIN

  -- Tạo rạp chiếu phim mới

  INSERT INTO SHOWROOM

  VALUES (@SHOWROOM\_num);

  -- Lấy ID của rạp chiếu phim mới

  DECLARE @SHOWROOM\_id INT;

  SET @SHOWROOM\_id = SCOPE\_IDENTITY();

  -- Tạo các ghế Standard trong rạp chiếu phim

  DECLARE @letters VARCHAR(8);

  DECLARE @number INT;

  DECLARE @letter\_index INT;

  DECLARE @letter CHAR(1);

  SET @letters = 'ABCDEFGH';

  SET @letter\_index = 1;

  WHILE @letter\_index <= LEN(@letters)

  BEGIN

    SET @number = 1;

    WHILE @number <= 10

    BEGIN

      SET @letter = SUBSTRING(@letters, @letter\_index, 1);

      INSERT INTO seat (SHOWROOM\_ID, SEATNUMBER, SEATTYPE)

      VALUES (@SHOWROOM\_id, CONCAT(@letter, @number), 'Standard');

      SET @number = @number + 1;

    END;

    SET @letter\_index = @letter\_index + 1;

  END;

  -- Tạo các ghế Couple trong rạp chiếu phim

  SET @letters = 'IJ';

  SET @number = 1;

  WHILE @number <= 5

  BEGIN

    SET @letter\_index = 1;

    WHILE @letter\_index <= LEN(@letters)

    BEGIN

      SET @letter = SUBSTRING(@letters, @letter\_index, 1);

      INSERT INTO seat (SHOWROOM\_ID, SEATNUMBER, SEATTYPE)

      VALUES (@SHOWROOM\_id, CONCAT(@letter, @number), 'Couple');

      SET @letter\_index = @letter\_index + 1;

    END;

    SET @number = @number + 1;

  END;

END;

go

-- select \* from SEAT

-- drop proc create\_showroom

----------------------------------------------- Tạo lịch chiếu phim

---drop proc create\_schedule

CREATE PROCEDURE create\_schedule

    @p\_SHOWROOMID INT,

    @p\_movieID INT,

    @p\_startTime DATETIME,

    @p\_price FLOAT

AS

BEGIN

    SET NOCOUNT ON;

    -- Check for schedule conflicts within the showroom

    IF EXISTS (

        SELECT 1

        FROM SCHEDULE

        WHERE SHOWROOM\_ID = @p\_SHOWROOMID

          AND (

            (@p\_startTime >= STARTTIME AND @p\_startTime < ENDTIME)

            OR

            (DATEADD(MINUTE, (SELECT DURATION FROM MOVIE WHERE ID = @p\_movieID), @p\_startTime) > STARTTIME

             AND @p\_startTime < ENDTIME)

          )

    )

    BEGIN

        -- Handle the conflict (e.g., raise an error, return a message)

        THROW 51000, 'Schedule conflict: The showroom is already booked at this time.', 1;

        RETURN;

    END;

    DECLARE @seatCount INT;

    SET @seatCount = 90;

    -- Insert the schedule

    INSERT INTO SCHEDULE (SHOWROOM\_ID, MOVIE\_ID, STARTTIME, ENDTIME, PRICE)

    SELECT @p\_SHOWROOMID, @p\_movieID, @p\_startTime, DATEADD(MINUTE, (SELECT DURATION FROM MOVIE WHERE ID = @p\_movieID), @p\_startTime), @p\_price;

END;

GO

----------------------------------------------------- Lấy lịch chiếu của 1 phòng

CREATE PROCEDURE [dbo].[get\_schedule\_by\_showroom]

    @SHOWROOM\_id INT

AS

BEGIN

    SET NOCOUNT ON;

    SELECT

        schedule.ID,

        SHOWROOM.SHOWROOMNUMBER,

        CAST(MOVIE.POSTER AS varchar(MAX)) AS POSTER,

        CAST(movie.TITLE AS NVARCHAR(MAX)) AS TITLE,

        movie.DURATION,

        SCHEDULE.STARTTIME,

        SCHEDULE.ENDTIME,

        SCHEDULE.PRICE

    FROM

        schedule

    INNER JOIN

        SHOWROOM ON schedule.SHOWROOM\_ID = SHOWROOM.ID

    INNER JOIN

        movie ON movie.ID = schedule.MOVIE\_ID

    WHERE

         SHOWROOM.ID = @SHOWROOM\_id

    GROUP BY

        schedule.ID,

        CAST(movie.TITLE AS NVARCHAR(MAX)),

        movie.DURATION,

        schedule.SHOWROOM\_ID,

        schedule.MOVIE\_ID,

        schedule.STARTTIME,

        SCHEDULE.ENDTIME,

        schedule.PRICE,

        SHOWROOM.SHOWROOMNUMBER,

        CAST(MOVIE.POSTER AS varchar(MAX))

    ORDER BY

        schedule.STARTTIME desc

END

GO

--exec [get\_schedule\_by\_showroom] 1

go

-- drop proc get\_schedule\_by\_showroom

----------------------------------------------------- Lấy lịch chiếu của 1 phim

CREATE PROCEDURE [dbo].[get\_schedule\_by\_movie]

    @MOVIE\_id INT

AS

BEGIN

    SELECT schedule.ID,SHOWROOM.SHOWROOMNUMBER,CAST(MOVIE.POSTER AS varchar(MAX)) As POSTER, CAST(movie.TITLE AS NVARCHAR(MAX)) As TITLE

    , movie.DURATION, SCHEDULE.STARTTIME,Schedule.PRICE, SCHEDULE.ENDTIME, COUNT(TICKET.ID) AS BOOKED

    FROM schedule

    INNER JOIN SHOWROOM ON schedule.SHOWROOM\_ID = SHOWROOM.ID

    INNER JOIN TICKET ON schedule.ID = TICKET.SCHEDULE\_ID

    INNER JOIN movie ON movie.ID = schedule.MOVIE\_ID

    WHERE  MOVIE.ID = @MOVIE\_id

    GROUP BY schedule.ID, CAST(movie.TITLE AS NVARCHAR(MAX)), movie.DURATION,schedule.SHOWROOM\_ID,

    schedule.MOVIE\_ID, schedule.STARTTIME, SCHEDULE.ENDTIME, schedule.PRICE, SHOWROOM.SHOWROOMNUMBER, CAST(MOVIE.POSTER AS varchar(MAX))

    ORDER BY

        schedule.STARTTIME desc

END

go

-- exec [get\_schedule\_by\_movie] 1

select \* from schedule

go

-- drop proc [get\_schedule\_by\_movie]

-------------------------------------------------- lấy danh sách phim đang chiếu

CREATE proc getIshowingMovie

AS

BEGIN

   SELECT  MOVIE.\*

   FROM MOVIE

   WHERE GETDATE() BETWEEN MOVIE.OPENING\_DAY AND MOVIE.CLOSING\_DAY;

END

GO

-- drop PROC getIshowingMovie

-- EXEC getIshowingMovie

GO

-------------------------------------------------- Lấy các ghế đã được đặt

CREATE PROCEDURE get\_booked\_seats

    @schedule\_id INT

AS

BEGIN

    SELECT seat.ID, seat.SEATNUMBER, SEAT.SEATTYPE, TICKET.ID, BOOKING\_ID, SCHEDULE\_ID

    FROM seat

    INNER JOIN TICKET ON seat.ID = TICKET.SEAT\_ID

    INNER JOIN schedule ON schedule.ID = TICKET.SCHEDULE\_ID

    WHERE  schedule.ID = @schedule\_id

END;

go

-- drop proc get\_booked\_seats

-- exec get\_booked\_seats 5

-- select \* from TICKET

-- select \* from booking

go

--------------------------------- Lấy danh sách các ghế

CREATE PROCEDURE get\_seats

    @schedule\_id INT

AS

BEGIN

    SELECT seat.ID, seat.SEATNUMBER, SEAT.SEATTYPE

    FROM seat

    INNER JOIN SHOWROOM ON SHOWROOM.ID = SEAT.SHOWROOM\_ID

    INNER JOIN schedule ON SHOWROOM.ID = SCHEDULE.SHOWROOM\_ID

    WHERE

         schedule.ID = @schedule\_id

END;

go

-- select \* from seat

-- exec get\_seats 1

go

-- drop proc get\_seats

------------------------------------------------- Lấy các combo

CREATE PROCEDURE dbo.get\_foodcombo

AS

BEGIN

    SELECT

        fc.ID AS ID,

        fc.NAME AS NAME,

        MAX(CASE WHEN p.TYPE = 'FOOD' THEN p.NAME ELSE NULL END) AS FOOD,

        STUFF((SELECT ',' + CAST(pf.QUANTITY AS VARCHAR)

               FROM product\_fcb pf

               JOIN product p2 ON pf.PRODUCT\_ID = p2.ID

               WHERE p2.TYPE = 'FOOD' AND pf.FCB\_ID = fc.ID

               FOR XML PATH('')), 1, 1, '') AS FOOD\_QUANTITY,

        MAX(CASE WHEN p.TYPE = 'DRINK' THEN p.NAME ELSE NULL END) AS DRINK,

        STUFF((SELECT ',' + CAST(pf.QUANTITY AS VARCHAR)

               FROM product\_fcb pf

               JOIN product p2 ON pf.PRODUCT\_ID = p2.ID

               WHERE p2.TYPE = 'DRINK' AND pf.FCB\_ID = fc.ID

               FOR XML PATH('')), 1, 1, '') AS DRINK\_QUANTITY,

        fc.PRICE AS PRICE,

         CAST(fc.Image AS NVARCHAR(MAX)) as IMAGE

    FROM foodcombo fc

    JOIN product\_fcb pf ON fc.ID = pf.FCB\_ID

    JOIN product p ON pf.PRODUCT\_ID = p.ID

    GROUP BY fc.ID, fc.NAME, fc.PRICE, CAST(fc.Image AS NVARCHAR(MAX))

END;

go

--- exec get\_foodcombo

go

------------------------------------------------ Lấy combo by ID

CREATE PROCEDURE dbo.get\_foodcombo\_by\_id @id int

AS

BEGIN

    SELECT

        fc.ID AS ID,

        fc.NAME AS NAME,

        MAX(CASE WHEN p.TYPE = 'FOOD' THEN p.NAME ELSE NULL END) AS FOOD,

        STUFF((SELECT ',' + CAST(pf.QUANTITY AS VARCHAR)

               FROM product\_fcb pf

               JOIN product p2 ON pf.PRODUCT\_ID = p2.ID

               WHERE p2.TYPE = 'FOOD' AND pf.FCB\_ID = fc.ID

               FOR XML PATH('')), 1, 1, '') AS FOOD\_QUANTITY,

        MAX(CASE WHEN p.TYPE = 'DRINK' THEN p.NAME ELSE NULL END) AS DRINK,

        STUFF((SELECT ',' + CAST(pf.QUANTITY AS VARCHAR)

               FROM product\_fcb pf

               JOIN product p2 ON pf.PRODUCT\_ID = p2.ID

               WHERE p2.TYPE = 'DRINK' AND pf.FCB\_ID = fc.ID

               FOR XML PATH('')), 1, 1, '') AS DRINK\_QUANTITY,

        fc.PRICE AS PRICE,

         CAST(fc.Image AS NVARCHAR(MAX)) as IMAGE

    FROM foodcombo fc

    JOIN product\_fcb pf ON fc.ID = pf.FCB\_ID

    JOIN product p ON pf.PRODUCT\_ID = p.ID

    WHERE FC.ID = @id

    GROUP BY fc.ID, fc.NAME, fc.PRICE, CAST(fc.Image AS NVARCHAR(MAX))

END;

go

--- exec get\_foodcombo\_by\_id 1

go

------------------------------------------------- Đặt vé

CREATE PROC create\_booking  @staffID int,@promotionID int, @bookingID int OUTPUT

AS

BEGIN

   -- TAO BOOKING

      INSERT INTO [dbo].[BOOKING]

           ([STAFF\_ID]

           ,[CREATED\_AT]

           ,[FOODPRICE]

           ,[TICKETPRICE]

           ,[PROMOTION\_ID])

      VALUES

           (@staffID

           ,GETDATE()

           ,0

           ,0

           ,@promotionID)

   -- LAY ID CUA BOOKING VUA TAO

      SET @bookingID = SCOPE\_IDENTITY();

      SELECT @bookingID as BOOKING\_id;

END

GO

CREATE PROC create\_booking\_without\_promotion   @staffID int, @bookingID int OUTPUT

AS

BEGIN

   -- TAO BOOKING

      INSERT INTO [dbo].[BOOKING]

           ([STAFF\_ID]

           ,[CREATED\_AT]

           ,[FOODPRICE]

           ,[TICKETPRICE])

      VALUES

           (@staffID

           ,GETDATE()

           ,0

           ,0)

   -- LAY ID CUA BOOKING VUA TAO

      SET @bookingID = SCOPE\_IDENTITY();

      SELECT @bookingID as BOOKING\_id;

END

GO

--drop proc create\_booking

-- THEM THONG TIN VE COMBO DO AN

-- CAP NHAT TRANG THAI CUA CAC VE

-------------------------------------------- Tổng doanh thu trong ngày của nhân viên

CREATE PROCEDURE get\_price\_sum

    @staff\_id INT,

    @created\_at DATE

AS

BEGIN

    SELECT SUM((FOODPRICE + TICKETPRICE) \*(100 - COALESCE(Promotion.Discount, 0))/ 100) AS TotalPrice

    FROM [IIEXCinema].[dbo].[BOOKING]

    LEFT JOIN Promotion ON Promotion.ID = BOOKING.promotion\_id

    WHERE [STAFF\_ID] = @staff\_id AND CAST([CREATED\_AT] AS DATE) = @created\_at

END

GO

--drop proc  get\_price\_sum

-- select \* from statitics

-- EXEC get\_price\_sum 2, '03/16/2023'

GO

-----------------------------------------------------------------Doanh thu theo thang

CREATE PROC get\_revenue\_by\_month @month date

AS

BEGIN

    SELECT

        CAST(CREATED\_AT AS DATE) AS BookingDate,

        SUM((FOODPRICE + TICKETPRICE) \* (100 - COALESCE(Promotion.Discount, 0)) / 100) AS TotalPrice

    FROM

        [IIEXCinema].[dbo].[BOOKING]

    LEFT JOIN

        Promotion ON Promotion.ID = BOOKING.promotion\_id

    where MONTH(CREATED\_AT) = MONTH(@month) and YEAR(CREATED\_AT) = YEAR(@month)

    GROUP BY

        CAST(CREATED\_AT AS DATE)

    ORDER BY

        CAST(CREATED\_AT AS DATE)

END

GO

--EXEC get\_revenue\_by\_month '03/01/2023'

GO

3. Function:

4. Trigger:

/\*==============================================================\*/

/\*                        TRIGGERS                              \*/

/\*==============================================================\*/

--Select ID, CONCAT(Name, ' - ', Price) AS Name\_Price from product

--Select \* from food\_booking

CREATE TRIGGER CheckDuplicateSeat

ON TICKET

AFTER INSERT

AS

BEGIN

    IF  (

        (SELECT count(\*) from ticket, inserted WHERE ticket.SCHEDULE\_ID = inserted.SCHEDULE\_ID and ticket.SEAT\_ID = inserted.SEAT\_ID) >1

    )

    BEGIN

        RAISERROR('Duplicate seat found within the same schedule.', 16, 1)

        ROLLBACK TRANSACTION

        RETURN

    END

    ----

    UPDATE booking

    SET TICKETPRICE = (SELECT SUM(price)

                        FROM ticket

                        WHERE BOOKING\_ID = (SELECT BOOKING\_ID FROM inserted))

    WHERE ID = (SELECT BOOKING\_ID FROM inserted)

END

go

-- drop  TRIGGER CheckDuplicateSeat

-------------------------------------- ON TICKET UPDATE

CREATE TRIGGER update\_ticket

ON ticket

AFTER UPDATE

AS

BEGIN

    IF  (

        (SELECT count(\*) from ticket, inserted WHERE ticket.SCHEDULE\_ID = inserted.SCHEDULE\_ID and ticket.SEAT\_ID = inserted.SEAT\_ID) >1

    )

    BEGIN

        RAISERROR('Duplicate seat found within the same schedule.', 16, 1)

        ROLLBACK TRANSACTION

        RETURN

    END

    ----

    UPDATE booking

    SET TICKETPRICE = (SELECT SUM(price)

                        FROM ticket

                        WHERE BOOKING\_ID = (SELECT BOOKING\_ID FROM inserted))

    WHERE ID = (SELECT BOOKING\_ID FROM inserted)

END

GO

----------------------------------------ON FOOD\_BOOKING INSERT

CREATE TRIGGER insert\_food\_booking

ON food\_booking

AFTER INSERT

AS

BEGIN

    UPDATE booking

    SET FOODPRICE = ISNULL((SELECT SUM(p.PRICE \* pb.QUANTITY)

                           FROM product p

                           INNER JOIN product\_booking pb ON pb.PRODUCT\_ID = p.ID

                           WHERE pb.BOOKING\_ID = (SELECT BOOKING\_ID FROM inserted)), 0)

                     + ISNULL((SELECT SUM(fc.PRICE \* fb.QUANTITY)

                               FROM foodcombo fc

                               INNER JOIN food\_booking fb ON fb.FOOD\_ID = fc.ID

                               WHERE fb.BOOKING\_ID = (SELECT BOOKING\_ID FROM inserted)), 0)

    WHERE ID = (SELECT BOOKING\_ID FROM inserted);

    UPDATE p

    SET p.QUANTITY = p.QUANTITY - (fb.QUANTITY \* pf.QUANTITY)

    FROM product p

    INNER JOIN product\_fcb pf ON p.ID = pf.PRODUCT\_ID

    INNER JOIN food\_booking fb ON pf.FCB\_ID = fb.FOOD\_ID

    WHERE fb.BOOKING\_ID = (SELECT BOOKING\_ID FROM inserted);

END

go

---------------------------------------- ON product\_booking

CREATE TRIGGER insert\_product\_booking

ON product\_booking

AFTER INSERT

AS

BEGIN

   UPDATE booking

    SET FOODPRICE = ISNULL((SELECT SUM(p.PRICE \* pb.QUANTITY)

                           FROM product p

                           INNER JOIN product\_booking pb ON pb.PRODUCT\_ID = p.ID

                           WHERE pb.BOOKING\_ID = (SELECT BOOKING\_ID FROM inserted)), 0)

                     + ISNULL((SELECT SUM(fc.PRICE \* fb.QUANTITY)

                               FROM foodcombo fc

                               INNER JOIN food\_booking fb ON fb.FOOD\_ID = fc.ID

                               WHERE fb.BOOKING\_ID = (SELECT BOOKING\_ID FROM inserted)), 0)

    WHERE ID = (SELECT BOOKING\_ID FROM inserted);

    UPDATE p

    SET p.QUANTITY = p.QUANTITY - pb.QUANTITY

    FROM product p

    INNER JOIN product\_booking pb ON pb.PRODUCT\_ID = p.ID

    WHERE pb.BOOKING\_ID = (SELECT BOOKING\_ID FROM inserted);

END

GO

5. Các câu lệnh tạo csdl:

/\*==============================================================\*/

/\* Table: BOOKING                                               \*/

/\*==============================================================\*/

create table BOOKING (

   ID                   int                  identity,

   STAFF\_ID             int                  null,

   CLIENT\_ID            int                  null,

   CREATED\_AT           datetime             null,

   FOODPRICE            float                null,

   TICKETPRICE          float                null,

   PROMOTION\_ID         int                  null,

   constraint PK\_BOOKING primary key (ID)

)

go

/\*==============================================================\*/

/\* Index: CLIENT\_BOOKING\_FK                                     \*/

/\*==============================================================\*/

create nonclustered index CLIENT\_BOOKING\_FK on BOOKING (CLIENT\_ID ASC)

go

/\*==============================================================\*/

/\* Index: STAFF\_BOOKING\_FK                                      \*/

/\*==============================================================\*/

create nonclustered index STAFF\_BOOKING\_FK on BOOKING (STAFF\_ID ASC)

go

/\*==============================================================\*/

/\* Table: FOODCOMBO                                             \*/

/\*==============================================================\*/

create table FOODCOMBO (

   ID                   int                  identity,

   NAME                 nvarchar(255)        null,

   PRICE                float                null,

   IMAGE                text                 null,

   constraint PK\_FOODCOMBO primary key (ID)

)

go

/\*==============================================================\*/

/\* Table: FOOD\_BOOKING                                          \*/

/\*==============================================================\*/

create table FOOD\_BOOKING (

   FOOD\_ID              int                  not null,

   BOOKING\_ID           int                  not null,

   QUANTITY             int                  null,

   constraint PK\_FOOD\_BOOKING primary key (FOOD\_ID, BOOKING\_ID)

)

go

/\*==============================================================\*/

/\* Index: FOOD\_BOOKING\_FK                                       \*/

/\*==============================================================\*/

create nonclustered index FOOD\_BOOKING\_FK on FOOD\_BOOKING (FOOD\_ID ASC)

go

/\*==============================================================\*/

/\* Index: FOOD\_BOOKING\_FK2                                      \*/

/\*==============================================================\*/

create nonclustered index FOOD\_BOOKING\_FK2 on FOOD\_BOOKING (BOOKING\_ID ASC)

go

/\*==============================================================\*/

/\* Table: MOVIE                                                 \*/

/\*==============================================================\*/

create table MOVIE (

   ID                   int                  identity,

   TITLE                ntext                null,

   GENRE                nvarchar(50)         null,

   DURATION             int                  null,

   RATING               float                null,

   STORY                ntext                null,

   POSTER               text                 null,

   OPENING\_DAY          datetime             null,

   CLOSING\_DAY          datetime             null,

   constraint PK\_MOVIE primary key (ID)

)

go

/\*==============================================================\*/

/\* Table: PRODUCT                                               \*/

/\*==============================================================\*/

create table PRODUCT (

   ID                   int                  identity,

   NAME                 nvarchar(50)         null,

   TYPE                 nvarchar(20)         null,

   PRICE                float                null,

   QUANTITY             int                  null,

   EXPIRY\_DATE          datetime             null,

   LAST\_ORDER             int                  null,

   constraint PK\_PRODUCT primary key (ID)

)

go

/\*==============================================================\*/

/\* Table: PRODUCT\_BOOKING                                       \*/

/\*==============================================================\*/

create table PRODUCT\_BOOKING (

   PRODUCT\_ID           int                  not null,

   BOOKING\_ID           int                  not null,

   QUANTITY             int                  null,

   constraint PK\_PRODUCT\_BOOKING primary key (PRODUCT\_ID, BOOKING\_ID)

)

go

/\*==============================================================\*/

/\* Index: FOOD\_BOOKING\_FK                                       \*/

/\*==============================================================\*/

create nonclustered index FOOD\_BOOKING\_FK on PRODUCT\_BOOKING (PRODUCT\_ID ASC)

go

/\*==============================================================\*/

/\* Index: FOOD\_BOOKING\_FK2                                      \*/

/\*==============================================================\*/

create nonclustered index FOOD\_BOOKING\_FK2 on PRODUCT\_BOOKING (BOOKING\_ID ASC)

go

/\*==============================================================\*/

/\* Table: PRODUCT\_FCB                                           \*/

/\*==============================================================\*/

create table PRODUCT\_FCB (

   PRODUCT\_ID           int                  not null,

   FCB\_ID               int                  not null,

   QUANTITY             int                  null,

   constraint PK\_PRODUCT\_FCB primary key (PRODUCT\_ID, FCB\_ID)

)

go

/\*==============================================================\*/

/\* Index: FOOD\_FCB\_FK                                           \*/

/\*==============================================================\*/

create nonclustered index FOOD\_FCB\_FK on PRODUCT\_FCB (PRODUCT\_ID ASC)

go

/\*==============================================================\*/

/\* Index: FOOD\_FCB\_FK2                                          \*/

/\*==============================================================\*/

create nonclustered index FOOD\_FCB\_FK2 on PRODUCT\_FCB (FCB\_ID ASC)

go

/\*==============================================================\*/

/\* Table: PROMOTION                                             \*/

/\*==============================================================\*/

create table PROMOTION (

   ID                   int                  identity,

   CODE                 varchar(5)           null,

   DISCOUNT             float                null,

   STATUS               nvarchar(20)         null,

   NOTE                 nvarchar(255)        null,

   constraint PK\_PROMOTION primary key (ID)

)

go

/\*==============================================================\*/

/\* Table: ROLE                                                  \*/

/\*==============================================================\*/

create table ROLE (

   ID                   int                  identity,

   NAME                 varchar(50)          null,

   constraint PK\_ROLE primary key (ID)

)

go

/\*==============================================================\*/

/\* Table: SCHEDULE                                              \*/

/\*==============================================================\*/

create table SCHEDULE (

   ID                   int                  identity,

   SHOWROOM\_ID          int                  not null,

   MOVIE\_ID             int                  not null,

   STARTTIME            datetime             null,

   ENDTIME              datetime             null,

   PRICE                float                null,

   constraint PK\_SCHEDULE primary key (ID)

)

go

/\*==============================================================\*/

/\* Index: MOVIE\_SCHEDULE\_FK                                     \*/

/\*==============================================================\*/

create nonclustered index MOVIE\_SCHEDULE\_FK on SCHEDULE (MOVIE\_ID ASC)

go

/\*==============================================================\*/

/\* Index: CINEMA\_SCHEDULE\_FK                                    \*/

/\*==============================================================\*/

create nonclustered index CINEMA\_SCHEDULE\_FK on SCHEDULE (SHOWROOM\_ID ASC)

go

/\*==============================================================\*/

/\* Table: SEAT                                                  \*/

/\*==============================================================\*/

create table SEAT (

   ID                   int                  identity,

   SHOWROOM\_ID          int                  not null,

   SEATNUMBER           varchar(5)           null,

   SEATTYPE             varchar(20)          null,

   constraint PK\_SEAT primary key (ID)

)

go

/\*==============================================================\*/

/\* Index: THEATER\_SEAT\_FK                                       \*/

/\*==============================================================\*/

create nonclustered index THEATER\_SEAT\_FK on SEAT (SHOWROOM\_ID ASC)

go

/\*==============================================================\*/

/\* Table: SHOWROOM                                              \*/

/\*==============================================================\*/

create table SHOWROOM (

   ID                   int                  identity,

   SHOWROOMNUMBER       int                  null,

   constraint PK\_SHOWROOM primary key (ID)

)

go

/\*==============================================================\*/

/\* Table: STAFF                                                 \*/

/\*==============================================================\*/

create table STAFF (

   ID                   int                  identity,

   EMAIL                varchar(50)          null,

   PASSWORD             varchar(50)          null,

   FIRSTNAME            nvarchar(50)         null,

   LASTNAME             nvarchar(20)         null,

   SEX                  nvarchar(5)          null,

   BIRTHDAY             datetime             null,

   PHONE                char(15)             null,

   ADDRESS              nvarchar(255)        null,

   SALARY               float                null,

   ROLE\_ID              int                  null,

   constraint PK\_STAFF primary key (ID)

)

go

/\*==============================================================\*/

/\* Table: STATITICS                                             \*/

/\*==============================================================\*/

create table STATITICS (

   ID                   int                  identity,

   STAFF\_ID             int                  null,

   TOTAL\_PRICE          float                null,

   TOTAL\_CASH           float                null,

   CREATED\_AT           datetime             null,

   STATUS               nvarchar(20)         null,

   NOTE                 nvarchar(255)        null,

   constraint PK\_STATITICS primary key (ID)

)

go

/\*==============================================================\*/

/\* Table: TICKET                                                \*/

/\*==============================================================\*/

create table TICKET (

   ID                   int                  identity,

   BOOKING\_ID           int                  null,

   PRICE                float                null,

   SCHEDULE\_ID          int                  null,

   SEAT\_ID              int                  null,

   constraint PK\_TICKET primary key (ID)

)

go

/\*==============================================================\*/

/\* Index: TICKET\_BOOKING\_FK                                     \*/

/\*==============================================================\*/

create nonclustered index TICKET\_BOOKING\_FK on TICKET (BOOKING\_ID ASC)

go

/\*==============================================================\*/

/\* Table: "USER"                                                \*/

/\*==============================================================\*/

create table "USER" (

   ID                   int                  identity,

   USERNAME             varchar(50)          null,

   PASSWORD             varchar(50)          null,

   NAME                 nvarchar(50)         null,

   PHONE                char(15)             null,

   ADDRESS              nvarchar(255)        null,

   constraint PK\_USER primary key (ID)

)

go

/\*==============================================================\*/

/\* Table: "City"                                                \*/

/\*==============================================================\*/

CREATE TABLE City (

    ID INT IDENTITY(1,1) PRIMARY KEY,

    Name NVARCHAR(255)

);

go

/\*==============================================================\*/

/\* Table: "GENRE"                                                \*/

/\*==============================================================\*/

CREATE TABLE GENRE (

    ID INT IDENTITY(1,1) PRIMARY KEY,

    Name NVARCHAR(255)

);

go

/\*==============================================================\*/

/\* Table: "ProductType"                                         \*/

/\*==============================================================\*/

CREATE TABLE ProductType (

    ID INT IDENTITY(1,1) PRIMARY KEY,

    Name VARCHAR(255)

);

go

alter table BOOKING

   add constraint FK\_BOOKING\_REF\_PROMOTIO foreign key (PROMOTION\_ID)

      references PROMOTION (ID)

         on update cascade

go

alter table STATITICS

   add constraint FK\_STATITIC\_REF\_STAFF foreign key (STAFF\_ID)

      references STAFF (ID)

         on update cascade

go

alter table BOOKING

   add constraint FK\_BOOKING\_CLIENT foreign key (CLIENT\_ID)

      references "USER" (ID)

         on update cascade

go

alter table BOOKING

   add constraint FK\_BOOKING\_STAFF foreign key (STAFF\_ID)

      references STAFF (ID)

      on update cascade

go

alter table FOOD\_BOOKING

   add constraint FK\_FCB\_BOOKING foreign key (BOOKING\_ID)

      references BOOKING (ID)

      on update cascade

go

alter table FOOD\_BOOKING

   add constraint FK\_Booking\_FCB foreign key (FOOD\_ID)

      references FOODCOMBO (ID)

      on update cascade

go

alter table PRODUCT\_BOOKING

   add constraint FK\_PRODUCT\_BOOKING\_\_REF\_BOOKING foreign key (BOOKING\_ID)

      references BOOKING (ID)

      on update cascade

go

alter table PRODUCT\_BOOKING

   add constraint FK\_PRODUCT\_BOOKING\_\_REF\_PRODUCT foreign key (PRODUCT\_ID)

      references PRODUCT (ID)

      on update cascade

go

alter table PRODUCT\_FCB

   add constraint FK\_FCB\_PRODUCT foreign key (FCB\_ID)

      references FOODCOMBO (ID)

         on update cascade

go

alter table PRODUCT\_FCB

   add constraint FK\_PRODUCT\_FCB foreign key (PRODUCT\_ID)

      references PRODUCT (ID)

         on update cascade

go

alter table SCHEDULE

   add constraint FK\_SCHEDULE\_MOVIE foreign key (MOVIE\_ID)

      references MOVIE (ID)

         on update cascade on delete cascade

go

alter table SCHEDULE

   add constraint FK\_SCHEDULE\_SHOWROOM foreign key (SHOWROOM\_ID)

      references SHOWROOM (ID)

         on update cascade on delete cascade

go

alter table SEAT

   add constraint FK\_SEAT\_SHOWROOM foreign key (SHOWROOM\_ID)

      references SHOWROOM (ID)

         on update cascade on delete cascade

go

alter table STAFF

   add constraint FK\_STAFF\_REF\_ROLE foreign key (ROLE\_ID)

      references ROLE (ID)

         on update cascade on delete set null

go

alter table TICKET

   add constraint FK\_TICKET\_REF\_SCHEDULE foreign key (SCHEDULE\_ID)

      references SCHEDULE (ID)

      on update cascade

go

alter table TICKET

   add constraint FK\_TICKET\_REF\_SEAT foreign key (SEAT\_ID)

      references SEAT (ID)

go

alter table TICKET

   add constraint FK\_TICKET\_BOOKING foreign key (BOOKING\_ID)

      references BOOKING (ID)

      on update cascade

go