**Group Project – Skills Demonstration**

Team Name 1: Gabriel Walsh Date: 7/16/2024

Team Name 2: Nicholas Stoddard

Team Name 3: Amir Urquhart

Team Name 4: Ramon Rodriguez

# **Group Project Instructions**

1. Type the names of team members that participated and the date in the spaces provided.
2. Use the SQL Server Management Studio and create and use a new database named xxGPDatabase (where the xx are the initials of one of your team members).
3. Complete all exercises in this group project.
4. Upload and submit a completed copy of this group project worksheet before the due date. Each member of the team that participated is to upload and submit a copy of this document.

**Exercises:**

# **Group Project Exercise 1(Database Design & Creation)**

Design and create a normalized database named xxGPDatabase where the xx are the initials of one of your team members. In this database create tables that a coffee shop can use to manage their Menu, Recipes, Inventory, Employees, and Work Schedules.

The Coffee Shop is open 24 hours a day. Each employee can work a max of 40 hours a week. Each shift is to be staffed with three employees. Each menu item has a recipe that specifies the ingredients and directions. All ingredients except for water are to be tracked in an inventory table.

The database needs to be normalized (3NF / Boyce-Codd Normal Form and no Multi-Valued Dependency). Each table must have a primary key and other appropriate attributes. The attributes on the tables must have appropriate datatypes, lengths, and constraints defined. At least 3 check constraints must be present. In addition, constraints to create appropriate relationships between the tables and enforce referential integrity must be present.

**--- Paste below this line a script file that when executed creates the xxGPDatabase with all the structural components described above. --**

Paste here

USE master;

GO

DROP DATABASE IF EXISTS RRGPDatabase;

GO

CREATE DATABASE RRGPDatabase;

GO

USE RRGPDatabase;

GO

--Employees table

CREATE TABLE Employees (

EmployeeID INT IDENTITY(1,1) PRIMARY KEY NOT NULL,

FirstName NVARCHAR(50) NOT NULL,

LastName NVARCHAR(50) NOT NULL,

[State] NVARCHAR(2) NOT NULL,

PhoneNumber NVARCHAR(15) NOT NULL,

Email NVARCHAR(100) NOT NULL,

CHECK (LEN(PhoneNumber) >= 10),

CHECK (Email LIKE '%@%')

);

--Shifts table

CREATE TABLE Shifts (

ShiftID INT IDENTITY(1,1) PRIMARY KEY NOT NULL,

ShiftStartTime TIME NOT NULL,

ShiftEndTime TIME NOT NULL,

CHECK (ShiftStartTime < ShiftEndTime)

);

--EmployeeShifts table many-to-many relationship

CREATE TABLE EmployeeShifts (

EmployeeShiftID INT IDENTITY(1,1) PRIMARY KEY NOT NULL,

EmployeeID INT NOT NULL,

ShiftID INT NOT NULL,

ShiftDate DATE NOT NULL,

FOREIGN KEY (EmployeeID) REFERENCES Employees(EmployeeID),

FOREIGN KEY (ShiftID) REFERENCES Shifts(ShiftID),

UNIQUE (EmployeeID, ShiftID, ShiftDate)

);

--MenuItems table

CREATE TABLE MenuChoice(

MenuChoiceID INT IDENTITY(1,1) PRIMARY KEY,

Name NVARCHAR(100) NOT NULL,

Description NVARCHAR(255),

Price DECIMAL(5, 2) NOT NULL CHECK (Price > 0)

);

--Ingredients table

CREATE TABLE Ingredients (

IngredientID INT IDENTITY(1,1) PRIMARY KEY,

Name NVARCHAR(100) NOT NULL,

Quantity DECIMAL(10, 2) NOT NULL CHECK (Quantity >= 0),

Unit NVARCHAR(50) NOT NULL,

ReorderLevel DECIMAL(10, 2) NOT NULL CHECK (ReorderLevel >= 0)

);

--Recipes table

CREATE TABLE Recipes (

RecipeID INT IDENTITY(1,1) PRIMARY KEY,

MenuChoiceID INT NOT NULL,

Directions NVARCHAR(MAX) NOT NULL,

FOREIGN KEY (MenuChoiceID) REFERENCES MenuChoice(MenuChoiceID)

);

--RecipeIngredients table many-to-many relationship

CREATE TABLE RecipeIngredients (

RecipeIngredientID INT IDENTITY(1,1) PRIMARY KEY,

RecipeID INT NOT NULL,

IngredientID INT NOT NULL,

Quantity DECIMAL(10, 2) NOT NULL CHECK (Quantity > 0),

FOREIGN KEY (RecipeID) REFERENCES Recipes(RecipeID),

FOREIGN KEY (IngredientID) REFERENCES Ingredients(IngredientID),

UNIQUE (RecipeID, IngredientID)

);

**--- Paste below this line an image of a database diagram that shows the relationships between the tables --**

Paste here

A diagram of a company

Description automatically generated with medium confidence

# **Group Project Exercise 2 (Database Design & Creation)**

Code and execute Insert Statements to insert six or seven records into each of the tables in the xxGPDatabase Database. Make sure to use each team members name once when coding attribute values for first and last name attributes. Make up appropriate demonstration values for other attribute values.

**--- Paste below this line the insert statements your team coded. –**

Paste here

USE RRGPDatabase;

GO

-- Insert into Employees table

INSERT INTO Employees (FirstName, LastName, [State], PhoneNumber, Email)

VALUES

('Ramon', 'Rodriguez', 'SC', '1234567890', 'Ramon.Rod@example.com'),

('Gabriel', 'Walsh', 'SC', '1895678910', 'Gabriel.W@example.com'),

('Nicholas', 'Stoddard', 'SC', '4784563390', 'Nicholas.S@example.com'),

('Amir', 'Urquhart', 'SC', '5922367870', 'Amir.U@example.com'),

('Jane', 'Smith', 'AZ', '0987654321', 'jane.smith@example.com'),

('Alice', 'Johnson', 'ca', '5556667777', 'alice.johnson@example.com'),

('Bob', 'Brown', 'il', '2223334444', 'bob.brown@example.com'),

('Carol', 'Davis', 'la','3334445555', 'carol.davis@example.com'),

('Dave', 'Miller', 'me', '4445556666', 'dave.miller@example.com'),

('Eve', 'Wilson', 'fl', '5556667778', 'eve.wilson@example.com');

-- Insert into Shifts table

INSERT INTO Shifts (ShiftStartTime, ShiftEndTime)

VALUES

('06:00', '14:00'),

('14:00', '22:00');

-- Insert into EmployeeShifts table

INSERT INTO EmployeeShifts (EmployeeID, ShiftID, ShiftDate)

VALUES

(1, 1, '2024-06-10'),

(2, 2, '2024-06-10'),

(3, 2, '2024-06-10'),

(4, 1, '2024-06-11'),

(5, 2, '2024-06-11'),

(6, 2, '2024-06-11'),

(7, 1, '2024-06-12'),

(8, 2, '2024-06-15'),

(9, 1, '2024-06-18'),

(10, 1, '2024-06-18');

-- Insert into MenuChoice table

INSERT INTO MenuChoice (Name, Description, Price)

VALUES

('Espresso', 'Strong and bold espresso', 3.20),

('Latte', 'Espresso with steamed milk', 1.50),

('Cappuccino', 'Espresso with steamed and frothed milk', 3.75),

('Mocha', 'Espresso with chocolate and steamed milk', 5.20),

('Americano', 'Espresso with hot water', 3.08),

('Macchiato', 'Espresso with a small amount of foamed milk', 2.75),

('Flat White', 'Espresso with steamed milk and microfoam', 1.47);

-- Insert into Ingredients table

INSERT INTO Ingredients (Name, Quantity, Unit, ReorderLevel)

VALUES

('Espresso Beans', 100, 'kg', 20),

('Milk', 200, 'liters', 50),

('Sugar', 50, 'kg', 10),

('Chocolate Syrup', 30, 'liters', 5),

('Vanilla Syrup', 20, 'liters', 5),

('Caramel Syrup', 20, 'liters', 5),

('Whipped Cream', 15, 'liters', 3);

-- Insert into Recipes table

INSERT INTO Recipes (MenuChoiceID, Directions)

VALUES

(1, 'Use 18g of espresso beans. Brew for 25-30 seconds.'),

(2, 'Use 18g of espresso beans. Brew for 25-30 seconds. Add 200ml of steamed milk.'),

(3, 'Use 18g of espresso beans. Brew for 25-30 seconds. Add 100ml of steamed milk. Add frothed milk on top.'),

(4, 'Use 18g of espresso beans. Brew for 25-30 seconds. Add 150ml of steamed milk and 30ml of chocolate syrup.'),

(5, 'Use 18g of espresso beans. Brew for 25-30 seconds. Add 200ml of hot water.'),

(6, 'Use 18g of espresso beans. Brew for 25-30 seconds. Add a small amount of foamed milk.'),

(7, 'Use 18g of espresso beans. Brew for 25-30 seconds. Add 200ml of steamed milk with microfoam.');

-- Insert into RecipeIngredients table

INSERT INTO RecipeIngredients (RecipeID, IngredientID, Quantity)

VALUES

(1, 1, 0.018), -- Espresso uses 18g of espresso beans

(2, 1, 0.018), -- Latte uses 18g of espresso beans

(2, 2, 0.200), -- Latte uses 200ml of milk

(3, 1, 0.018), -- Cappuccino uses 18g of espresso beans

(3, 2, 0.100), -- Cappuccino uses 100ml of milk

(4, 1, 0.018), -- Mocha uses 18g of espresso beans

(4, 2, 0.150), -- Mocha uses 150ml of milk

(4, 4, 0.030), -- Mocha uses 30ml of chocolate syrup

(5, 1, 0.018), -- Americano uses 18g of espresso beans

(5, 2, 0.200), -- Americano uses 200ml of hot water (using milk for demonstration)

(6, 1, 0.018), -- Macchiato uses 18g of espresso beans

(6, 2, 0.050), -- Macchiato uses 50ml of foamed milk

(7, 1, 0.018), -- Flat White uses 18g of espresso beans

(7, 2, 0.200); -- Flat White uses 200ml of steamed milk with microfoam

**---** **Paste below this line the completion status messages from executing your team’s insert statements-**

Paste here

A screenshot of a computer program

Description automatically generated

# **Group Project Exercise 3 (View s)**

Create a view named GPViewXXXXXXXXXXXX where the XXXXXXXXXXXX is the last name of one of your team members. This view must use at least two of your tables and the result set needs to be meaningful.

**--- Paste below this line your team’s view creation code. –**

Paste here

DROP VIEW IF EXISTS GPViewRodriguez

GO

CREATE VIEW GPViewRodriguez

AS

SELECT TOP 10 e.FirstName + ' ' + e.LastName AS EmployeeFullName,

s.ShiftStartTime, s.ShiftEndTime, es.ShiftDate

FROM Employees e

INNER JOIN EmployeeShifts es ON e.EmployeeID = es.EmployeeID

INNER JOIN Shifts s ON es.ShiftID = s.ShiftID

ORDER BY EmployeeFullName, ShiftDate;

**---** **Paste below this line the completion status messages from executing your team’s view creation code--**

Paste here

A close-up of a number

Description automatically generated

**---** **Paste below this line the result set returned by running a SELECT \* from the view your team created. --**

Paste here

A screenshot of a computer

Description automatically generated

# **Group Project Exercise 4 (Stored Procedures)**

Create a stored procedure named GPProcXXXXXXXXXXXX where the XXXXXXXXXXXX is the last name of one of your team members. This stored procedure must contain logic that involves at least two of your tables and it must produce a meaningful report when executed.

**--- Paste below this line your team’s stored procedure creation code. –**

Paste here

DROP PROCEDURE IF EXISTS GPProcWalsh;

GO

CREATE PROCEDURE GPProcWalsh

AS

BEGIN

-- Search for employees with names starting with R, D, or E

SELECT e.FirstName + ' ' + e.LastName AS EmployeeFullName, s.ShiftStartTime, s.ShiftEndTime

FROM Employees e

INNER JOIN EmployeeShifts es ON e.EmployeeID = es.EmployeeID

INNER JOIN Shifts s ON es.ShiftID = s.ShiftID

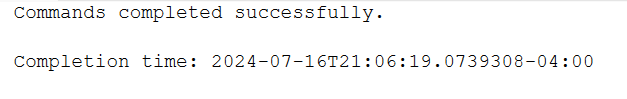
WHERE e.FirstName LIKE '[RDE]%'

ORDER BY EmployeeFullName, s.ShiftStartTime;

END;

**---** **Paste below this line the completion status messages from executing your team’s stored procedure creation code--**

Paste here



**---** **Paste below the execute statement your team used to test the stored procedure --**

Paste here

EXEC GPProcWalsh;

**---** **Paste below this line the run results from the statement used to test the stored procedure--**

Paste here

A screenshot of a computer

Description automatically generated

# **Group Project Exercise 5 (Triggers)**

Code a trigger named TR\_GP\_DML\_XXXXXXXXXXXXXX where the XXXXXXXXXXXXXX is the last name of one of your team members. This trigger must respond to inserts and/or updates to one of your tables and the trigger must perform a meaningful purpose.

**--- Paste below this line your team’s trigger creation code. –**

Paste here

DROP TRIGGER IF EXISTS TR\_GP\_DML\_Urquhart;

GO

CREATE TRIGGER TR\_GP\_DML\_Urquhart

ON Employees

AFTER INSERT

AS

BEGIN

-- Update the State column to uppercase for the newly inserted rows

UPDATE Employees

SET [State] = UPPER(i.[State])

FROM inserted i

WHERE Employees.EmployeeID = i.EmployeeID;

END;

**---** **Paste below this line the completion status messages from executing your team’s trigger creation code--**

Paste here

A close up of a number

Description automatically generated

**---** **Paste below this line the insert and/or update statements your team used to test the trigger --**

Paste here

INSERT INTO Employees (FirstName, LastName, [State], PhoneNumber, Email)

VALUES ('Monica', 'Dias', 'tx', '8642964578', 'mon.dias@example.com');

**---** **Paste below this line the run results from executing the statements used to test the trigger (include comments explaining what each test demonstrated) --**

Paste here

A number and numbers on a white background

Description automatically generated

# **Group Project Exercise 6 (Security)**

Write a script that creates a user-defined database role in the XXGPDatabase database (reminder replace the XX with the appropriate initials). Name this new role ProductManagementXX where the XX are your initials. Give UPDATE permission to the new role for the tables that contain menu, recipe, and inventory data, and SELECT permission for all other user tables. Make sure to include a comment at the top of the script with your name and the date you coded the script. Format this comment like the following sample:

/\* \*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*/

/\* ProductManagementXX Role Creation Script \*/

/\* Coded by: Gabriel Walsh \*/

/\* Coded on: 07/18/24 \*/

/\* \*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*/

**---** **Paste below this line the Script your team wrote for this exercise --**

USE RRGPDatabase;

GO

-- Clear Complications

DROP ROLE IF EXISTS ProductManagementGW;

GO

-- Create

CREATE ROLE ProductManagementGW;

GO

-- Grant UPDATE permission on tables related to menu, recipe, and inventory

GRANT UPDATE ON MenuChoice TO ProductManagementGW;

GRANT UPDATE ON Recipes TO ProductManagementGW;

GRANT UPDATE ON Ingredients TO ProductManagementGW;

GO

-- Grant SELECT permission on all other user tables

GRANT SELECT ON Employees TO ProductManagementGW;

GRANT SELECT ON Shifts TO ProductManagementGW;

GRANT SELECT ON EmployeeShifts TO ProductManagementGW;

GRANT SELECT ON RecipeIngredients TO ProductManagementGW;

GO

**---** **Paste below this line the run results from executing the script your team wrote--**

Commands completed successfully.

Completion time: 2024-07-18T17:35:24.8253715-04:00

# **Group Project Exercise 7 (Security)**

Use the XXGPDatabase database and write a script that:

1. Creates a login ID named “GPDLOG01XX” where the XX are your initials
2. Assigns this login an initial password of “GPDr33333”
3. Sets the default database for this login to the XXGPDatabase database
4. Creates a user named “GPDLOG01XX” where the XX are your initials
5. Assigns the “GPDLOG01XX” user to the ProductManagementXX role you created in exercise 6
6. Includes a comment at the top of the script with your name and the date you coded the script. Format this comment like the following sample:

/\* \*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*/

/\* GPDLOG01XX Login and User Creation Script \*/

/\* Coded by: Gabriel Walsh \*/

/\* Coded on: 07/18/24 \*/

/\* \*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*/

**--- Paste below this line the Script your team wrote for this exercise --**

USE RRGPDatabase;

GO

-- Drop the login if it exists

IF EXISTS (SELECT 1 FROM sys.sql\_logins WHERE name = 'GPDLOG01GW')

BEGIN

DROP LOGIN GPDLOG01GW;

END

GO

-- Create the login with a password and set the default database

CREATE LOGIN GPDLOG01GW

WITH PASSWORD = 'GPDr33333',

DEFAULT\_DATABASE = RRGPDatabase;

GO

-- Drop the user if it exists

IF EXISTS (SELECT 1 FROM sys.database\_principals WHERE name = 'GPDLOG01GW')

BEGIN

DROP USER GPDLOG01GW;

END

GO

-- Create the user for the login

CREATE USER GPDLOG01GW FOR LOGIN GPDLOG01GW;

GO

-- Assign the user to the ProductManagementGW role

ALTER ROLE ProductManagementGW ADD MEMBER GPDLOG01GW;

GO

**---** **Paste below this line the run results from executing the script your team wrote--**

Commands completed successfully.

Completion time: 2024-07-18T17:41:47.8921353-04:00

# **Group Project Exercise 8 (Security)**

Write a script that uses a table, dynamic SQL, and a cursor to create login IDs in the XXGPDatabase database based on the contents of the table.

1. Get started by first creating the table. Below is a create statement that you can modify and execute to create this table. Make sure to change the XX in the table name to your initials:

CREATE TABLE NewLoginsXX (LoginName varchar(128));

1. After the table is created. Insert four rows in it. Below is an insert statement that you can modify and execute to put the rows in your NewLoginsXX table. In addition to modifying the table name, make sure to change the XX at the start of each login name to your initials before running this insert.

INSERT NewLoginsXX VALUES ('XXFFlake'), ('XXGGlute'), ('XXMMrit'), ('XXPPert');

1. Start coding the script with a comment block. Format this comment like the following sample (make sure to revise the sample to reflect your name and the date you coded it):

/\* \*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*/

/\* Login Generator Script \*/

/\* This Script uses dynamic SQL and a cursor to process the \*/

/\* NEWLoginsXX table one row at a time. For each row on this \*/

/\* table the script will perform four actions. \*/

/\* \*/

/\* Coded by: Your Name \*/

/\* Coded on: MM/DD/YY \*/

/\* \*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*/

1. Code the rest of your script using dynamic SQL and a cursor to perform four actions for each row in this table:

(1) create a login with a temporary password that's based on the first four letters of the login name followed by “33333”

(2) set the default database to the XXGPDatabase database

(3) create a user for the login with the same name as the login

(4) assign the user to the ProductManagementXX role you created in exercise 6.

**--- Paste below this line the Script your team wrote for this exercise --**

/\* \*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*/

/\* Create Logins and Users from NewLoginsGW Table \*/

/\* Coded by: Gabriel Walsh \*/

/\* Coded on: 07/18/2024 \*/

/\* \*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*/

**USE RRGPDatabase;**

**GO**

**-- Create the table**

**IF OBJECT\_ID('NewLoginsGW', 'U') IS NOT NULL**

**DROP TABLE NewLoginsGW;**

**GO**

**CREATE TABLE NewLoginsGW (**

**LoginName VARCHAR(128)**

**);**

**GO**

**-- Insert rows into the table**

**INSERT INTO NewLoginsGW(LoginName)**

**VALUES**

**('GWFFlake'),**

**('GWGGlute'),**

**('GWMMrit'),**

**('GWPPert');**

**GO**

**-- Declare a cursor to iterate over rows**

**DECLARE @LoginName VARCHAR(128);**

**DECLARE @SQL NVARCHAR(MAX);**

**DECLARE LoginCursor CURSOR FOR**

**SELECT LoginName**

**FROM NewLoginsGW;**

**OPEN LoginCursor;**

**FETCH NEXT FROM LoginCursor INTO @LoginName;**

**WHILE @@FETCH\_STATUS = 0**

**BEGIN**

**-- Drop the login if it exists**

**SET @SQL = N'IF EXISTS (SELECT 1 FROM sys.sql\_logins WHERE name = ''' + @LoginName + ''')**

**DROP LOGIN [' + @LoginName + '];';**

**EXEC sp\_executesql @SQL;**

**-- Create login with a password**

**SET @SQL = N'CREATE LOGIN [' + @LoginName +**

**'] WITH PASSWORD = ''' + LEFT(@LoginName, 4) + '33333'', ' +**

**'DEFAULT\_DATABASE = RRGPDatabase;';**

**EXEC sp\_executesql @SQL;**

**-- Drop the user if it exists**

**SET @SQL = N'IF EXISTS (SELECT 1 FROM sys.database\_principals WHERE name = ''' + @LoginName + ''')**

**DROP USER [' + @LoginName + '];';**

**EXEC sp\_executesql @SQL;**

**-- Create user in the database**

**SET @SQL = N'USE RRGPDatabase; CREATE USER [' + @LoginName +**

**'] FOR LOGIN [' + @LoginName + '];';**

**EXEC sp\_executesql @SQL;**

**-- Assign user to role**

**SET @SQL = N'USE RRGPDatabase; ALTER ROLE ProductManagementGW ADD MEMBER [' +**

**@LoginName + '];';**

**EXEC sp\_executesql @SQL;**

**FETCH NEXT FROM LoginCursor INTO @LoginName;**

**END**

**CLOSE LoginCursor;**

**DEALLOCATE LoginCursor;**

**GO**

**---** **Paste below this line the run results from executing the script your team wrote—**

A screenshot of a computer

Description automatically generated

# **Group Project Exercise 9 (Security)**

Using the Management Studio, create a login ID named “YYalkXX” where the XX are your initials. Assign this login ID a password “YYal33333,” and set the default database to the XXGPDatabase database. Then, grant the login ID access to the XXGPDatabase database, create a user for the login ID named “YYalkXX”, and assign the user to the PaymentEntry role you created in exercise 6.

Note: If you get an error that says, “The MUST\_CHANGE option is not supported”, you can deselect the “Enforce password policy” option for the login ID.

**---** **Paste below this line a screen shot that shows the new login creation window with the YYalkXX information showing—**

A screenshot of a computer

Description automatically generated

**---** **Paste below a screen shot that shows the login properties window with the YYalkXX’s mapping to the XXGPDatabase database and membership in the ProductManagementXX role your team created in exercise 6.--**

A screenshot of a computer

Description automatically generated

A screenshot of a computer

Description automatically generated