**Task 3b Creative Artefact**

/\*I acknowledge the use of generative AI tools to generate or enhance materials that are

included within this assessment task in modified form. \*/

/\* Situation: Deletes an existing table from the database schema, and ensures the query can run and re-run.

Rationale: These DROP TABLE IF EXISTS statements ensure primary constraints arn't conflicted when re-running the query. \*/

-- Drop Tables

DROP TABLE IF EXISTS equipmentInventory;

DROP TABLE IF EXISTS equipmentService;

DROP TABLE IF EXISTS equipment;

DROP TABLE IF EXISTS staffSpecialisation;

DROP TABLE IF EXISTS appointmentMessage;

DROP TABLE IF EXISTS message;

DROP TABLE IF EXISTS appointment;

DROP TABLE IF EXISTS staff;

DROP TABLE IF EXISTS client;

DROP TABLE IF EXISTS therapyType;

DROP TABLE IF EXISTS therapyCentre;

/\*Situation: Creates a new table in the database schema for the required table names below.

Rationale: The CREATE TABLE statements ensure tables are created in the database where inside

the primary key and foreign key constraints are added, column names, data types and lengths. \*/

--Therapy Centre Table

CREATE TABLE therapyCentre (

centreId INT PRIMARY KEY,

name VARCHAR (30),

street VARCHAR (50),

city VARCHAR (25),

state CHAR (50),

postCode VARCHAR (20),

contactNumber VARCHAR (20),

);

--Therapy Type Table

CREATE TABLE therapyType (

typeId INT PRIMARY KEY,

name VARCHAR (25),

description VARCHAR (250)

);

--Client Table

CREATE TABLE client (

clientId INT PRIMARY KEY,

fName VARCHAR (20),

lName VARCHAR (20),

city VARCHAR (25),

state CHAR (50),

postCode VARCHAR (20),

contactNumber VARCHAR (20),

email VARCHAR (25),

);

--Staff Table

CREATE TABLE staff (

staffId INT PRIMARY KEY,

fName VARCHAR (20),

lName VARCHAR (20),

city VARCHAR (25),

state CHAR (50),

contactNumber VARCHAR (20),

email VARCHAR (25),

);

--Appointment Table

CREATE TABLE appointment (

appointmentId INT PRIMARY KEY,

appointmentDate DATETIME,

therapyTypeId INT,

staffId INT,

clientId INT,

cost DECIMAL (10, 2),

centreId INT,

FOREIGN KEY (therapyTypeId) REFERENCES therapyType (typeId),

FOREIGN KEY (staffId) REFERENCES staff (staffId),

FOREIGN KEY (centreId) REFERENCES therapyCentre (centreId)

);

--Message Table

CREATE TABLE message (

messageId INT PRIMARY KEY,

appointmentMessage VARCHAR (200),

dateStamp DATETIME

);

--Appointment Message Table

CREATE TABLE appointmentMessage (

appointmentId INT,

messageId INT,

PRIMARY KEY (appointmentId, messageId),

FOREIGN KEY (appointmentId) REFERENCES appointment (appointmentId),

FOREIGN KEY (messageId) REFERENCES message (messageId)

);

--Staff Specialisation Table

CREATE TABLE staffSpecialisation (

staffId INT,

typeId INT,

PRIMARY KEY (staffId, typeId),

FOREIGN KEY (staffId) REFERENCES staff (staffId),

FOREIGN KEY (typeId) REFERENCES therapyType (typeId)

);

--Equipment Table

CREATE TABLE equipment (

equipmentId INT PRIMARY KEY,

name VARCHAR (50),

typeId INT,

centreId INT,

FOREIGN KEY (typeId) REFERENCES therapyType (typeId),

FOREIGN KEY (centreId) REFERENCES therapyCentre (centreId),

);

--Equipment Service Table

CREATE TABLE equipmentService (

serviceId INT PRIMARY KEY,

equipmentId INT,

serviceDate DATE,

FOREIGN KEY (equipmentId) REFERENCES equipment (equipmentId)

);

--Equipment Inventory Table

CREATE TABLE equipmentInventory (

inventoryId INT PRIMARY KEY,

equipmentId INT,

quantity INT,

FOREIGN KEY (equipmentId) REFERENCES equipment (equipmentId)

);

/\* Situation: Inserts data into tables in the database schema for the required table names below.

Rationale: The INSERTS INTO statement specifies the table the data is being added to, the VALUES

statement species the type of values to insert into the tables where the order of values meets the

order each column is situated inside the table. \*/

--Insert Data Into Therapy Centre

INSERT INTO therapyCentre

VALUES (2, 'North', '7 Holy Street', 'Brisbane', 'QLD', '5432', '0408 708 004'),

(4, 'South', '15 Fabulous Circuit', 'Sydney', 'NSW', '2000', '0447 456 443');

--Insert Data Into Therapy Type

INSERT INTO therapyType

VALUES (1, 'Mental', 'All therapies required to understand the human mind'),

(2, 'Spiritual', 'All therapies required to understand the spiritual self');

--Insert Data Into Client

INSERT INTO client

VALUES (123456, 'Torin', 'Flanagan', 'Brisbane', 'QLD', '5432', '0456 904 987', 'torin08@zyx.com'),

(123455, 'Sophie', 'Smith', 'Bondi Beach', 'NSW', '2026', '0412 345 678', 'sophie07@123.com'),

(123454, 'Liam', 'Johnson', 'South Yarra', 'VIC', '3000', '0423 456 789', 'liam19@xyz.com'),

(123453, 'Olivia', 'Brown', 'Kelvin Grove', 'QLD', '4000', '0432 567 890', 'olivia58@abc.com'),

(123452, 'William', 'Davis', 'Ashfield', 'WA', '6000', '0412 678 901', 'william37@emz.com'),

(123451, 'Lottie', 'Wilson', 'Surry Hills', 'NSW', '2000', '0432 789 012', 'charlotte25@123.com'),

(123457, 'Ben', 'Wallace', 'Adelaide', 'SA', '5000', '0456 875 844', 'ben21@hey.com'),

(123458, 'Chloe', 'Darlington', 'Coffs Harbour', 'NSW', '2450', '0433 765 392', 'chloe19@987.com');

--Insert Data Into Staff

INSERT INTO staff

VALUES (654321, 'Judy', 'Watson', 'Perth', 'SA', '0432 678 543', 'Judy.w@bky.com'),

(654322, 'Emma', 'Jones', 'Brisbane', 'QLD', '0412 123 456', 'Emma.j@bky.com'),

(654323, 'Jack', 'Smith', 'Gold Coast', 'QLD', '0423 234 567', 'Jack.s@bky.com'),

(654324, 'Oliver', 'Brown', 'Sunshine Coast', 'QLD', '0432 345 678', 'Oliver.b@bky.com'),

(654325, 'Sophia', 'Lee', 'Sydney', 'NSW', '0432 456 789', 'Sophia.l@bky.com'),

(654326, 'Ethan', 'Wilson', 'Newcastle', 'NSW', '0432 757 908', 'Ethan.w@bky.com'),

(654327, 'Jayden', 'Watts', 'Byron Bay', 'NSW', '0456 876 678', 'Jayden.w@bky.com'),

(654328, 'Huon', 'Blundstone', 'Hobart', 'TAS', '0462 295 548', 'Huon.B@bky.com');

--Insert Data Into Appointment

INSERT INTO appointment

VALUES (1065, '2024-05-26 13:45:00', 1, 654321, 123456, 150.00, 2),

(1066, '2024-05-26 10:15:00', 2, 654322, 123455, 95.00, 4);

--Insert Data Into Message

INSERT INTO message

VALUES (1234, 'Appointment available', '2024-05-26 06:30:28'),

(1233, 'Appointment unavailable', '2024-05-27 20:45:14');

--Insert Data Into Appointment Message

INSERT INTO appointmentMessage

VALUES (1065, 1234),

(1066, 1233);

--Insert Data Into Staff Specialisation

INSERT INTO staffSpecialisation

VALUES (654327, 1),

(654328, 2);

--Insert Data Into Equipment

INSERT INTO equipment

VALUES (54321, 'Stress ball', 1, 2),

(54322, 'Notebook', 2, 4);

--Insert Data Into Equipment Service

INSERT INTO equipmentService

VALUES (9, 54321, '2024-05-26'),

(10, 54322, '2024-05-27');

--Insert Data Into Equipment Inventory

INSERT INTO equipmentInventory

VALUES (7, 54321, 3),

(8, 54322, 5);

/\* Situation: Selects everything inside the mentioned table.

Rationale: The SELECT \* FROM statements are selecting everything in each mentioned table

and presents it in the Results tab in the output window. \*/

--Select Everything From Each Table

SELECT \* FROM client;

SELECT \* FROM staff;

SELECT \* FROM appointment;

SELECT \* FROM appointmentMessage;

SELECT \* FROM equipment;

SELECT \* FROM equipmentInventory;

SELECT \* FROM equipmentService;

SELECT \* FROM message;

SELECT \* FROM staffSpecialisation;

SELECT \* FROM therapyCentre;

SELECT \* FROM therapyType;

/\* Situation: Creates a view, specified by whats selected and where its from.

Rationale: The CREATE VIEW statement creates a view name, it then specifies what is selected

by the SELECT and from what table by the FROM, lastly the WHERE clause filters the rows

based on the specified conditions. \*/

--Create Selected Client\_Details View

GO

CREATE VIEW client\_details AS

SELECT fName, lName, contactNumber, email

FROM client

WHERE clientId IN (123452, 123453, 123454, 123455, 123456, 123457);

--Create Selected Staff\_Details View

GO

CREATE VIEW staff\_details AS

SELECT fName, lName, contactNumber, email

FROM staff

WHERE staffId IN (654322, 654323, 654324, 654325, 654326, 654327);

/\* Situation: Displays the virtual tables created above.

Rationale: The SELECT statement is used to select everything from both virtual tables. \*/

--Display Client And Staff Virtual Tables

GO

SELECT \* FROM client\_details;

SELECT \* FROM staff\_details;

/\* Situation: Deletes an existing view from the database schema, ensuring the views can run and re-run.

Rationale: The DROP VIEW IF EXISTS statement is used to delete the existing clients\_details and staff\_details views. \*/

--Drop Client and Staff Virtual Tables

GO

DROP VIEW IF EXISTS client\_details;

DROP VIEW IF EXISTS staff\_details;

/\* Situation: Creates a reminder system for appointments in the database system.

Rationale: The CREATE TRIGGER statement creates the trigger name, ON attaches the trigger to the Appointment table,

AFTER INSERT automattically fires the trigger after a new row is successfully inserted into the table, AS BEGIN DECLARE @ReminderHours INT = 24

is used to declare and initialise the local variable @ReminderHours with the integer value 24. \*/

--Message Firing Trigger

GO

CREATE TRIGGER ReminderForAppointment

ON Appointment

AFTER INSERT

AS BEGIN DECLARE @ReminderHours INT = 24;

/\* Situation: Sets a reminder time for when a new appointment is added to the database system.

Rationale: The DECLARE statement sets the variable @ReminderTime as a DATETIME data type, it then

calaculates a new DATETIME data type by subtracting a specified number of hours from the appointment date. \*/

-- Calculate Reminder Time

DECLARE @ReminderTime DATETIME = DATEADD(HOUR, -@ReminderHours, (SELECT AppointmentDate FROM inserted));

/\* Situation: Checks the reminder time to be 24 hours from now

Rationale: The IF statement checks if the local variable @ReminderTime is less than or equal to the current date. \*/

-- Check If The Reminder Time Is Within 24 Hours From Now

IF @ReminderTime <= GETDATE()

/\* Situation: Obtains the client's details required for the message

Rationale: The first 2 lines assign the full name of a client to the variable @ClientName by concatenating their first and last names

from the Client table based on the ClientID in the inserted table. The rest of the code then prints out the reminder message. \*/

-- Get The Client Details For The Reminder Message And Send The Message

BEGIN

DECLARE @ClientName NVARCHAR(100);

SELECT @ClientName = CONCAT(fName, ' ', lName) FROM Client WHERE ClientID = (SELECT ClientID FROM inserted);

PRINT 'Reminder: ' + @ClientName + ', your appointment is scheduled for tomorrow. Please remember to attend :)';

END

END;

/\* Situation: Tests the trigger created.

Rationale: The INSERT INTO, VALUES and SELECT \* FROM statements are used to insert the appropriate data, testing the trigger. \*/

--Test Trigger

GO

INSERT INTO Appointment (appointmentId, appointmentDate, therapyTypeId, staffId, clientId, cost, centreId)

VALUES (1067, DATEADD(HOUR, 23, GETDATE()), 1, 654324, 123458, 150.00, 4);

SELECT \* FROM Appointment;

/\* Situation: Generates a report listing staff members' first names, last names, contact numbers, the type of therapy they provide,

and the names of their assigned tutors.

Rationale: The is done by joining the tables including staff, staffSpecialisation, therapyType, and staff again to retrieve

relevant information for the report. \*/

-- Query 1: Staff Report with Tutor's Name

GO

SELECT s.fName AS 'First Name', s.lName AS 'Last Name', s.contactNumber AS 'Contact Number', t.name AS 'Type of Therapy', tutor.fName AS 'Tutor First Name', tutor.lName AS 'Tutor Last Name'

FROM staff AS s

JOIN staffSpecialisation AS ss ON s.staffId = ss.staffId

JOIN therapyType AS t ON ss.typeId = t.typeId

JOIN staff AS tutor ON tutor.staffId = 654321;

/\* Situation: Generates a report listing appointment details from the therapy centre's database, including the client's first and last names,

appointment date, type of therapy, and the first and last names of the staff member assigned to each appointment.

Rationale: This is done by joining the tables including clients, appointments, therapy types, and staff. It creates a comprehensive report

that facilitates the management of appointments and staff assignments within the therapy center. \*/

-- Query 2: Client Report with Your Name

GO

SELECT c.fName AS 'Client First Name', c.lName AS 'Client Last Name', a.appointmentDate AS 'Appointment Date', tt.name AS 'Type of Therapy', s.fName AS 'Staff First Name', s.lName as 'Staff Last Name'

FROM client AS c

JOIN appointment AS a ON c.clientId = a.clientId

JOIN therapyType AS tt ON a.therapyTypeId = tt.typeId

JOIN staff AS s ON s.staffId = 654324;

/\* Situation: Generates a report detailing equipment information including the type and quantity of equipment,

along with the date of its last service and the names of tutors responsible for overseeing its usage.

Rationale: This done by joining the tables including equipment, inventory, service history, and staff. It provides a

comprehensive overview facilitating equipment management and accountability within the institute. \*/

-- Query 3: Management Report with Your Tutor's Name

GO

SELECT equip.name AS 'Type of Equipment', equipInv.quantity AS 'Equipment Quantity', es.equipServ AS 'Last Service Date', tutor.fName AS 'Tutor First Name', tutor.lName AS 'Tutor Last Name'

FROM equipment AS equip

JOIN equipmentInventory AS equipInv ON equip.equipmentId = equipInv.equipmentId

JOIN (SELECT equipmentId, MAX(serviceDate) AS equipServ

FROM equipmentService

GROUP BY equipmentId) AS es ON equip.equipmentId = es.equipmentId

JOIN staff AS tutor ON tutor.staffId = 654321;