# INF10025 Data Management and Analytics

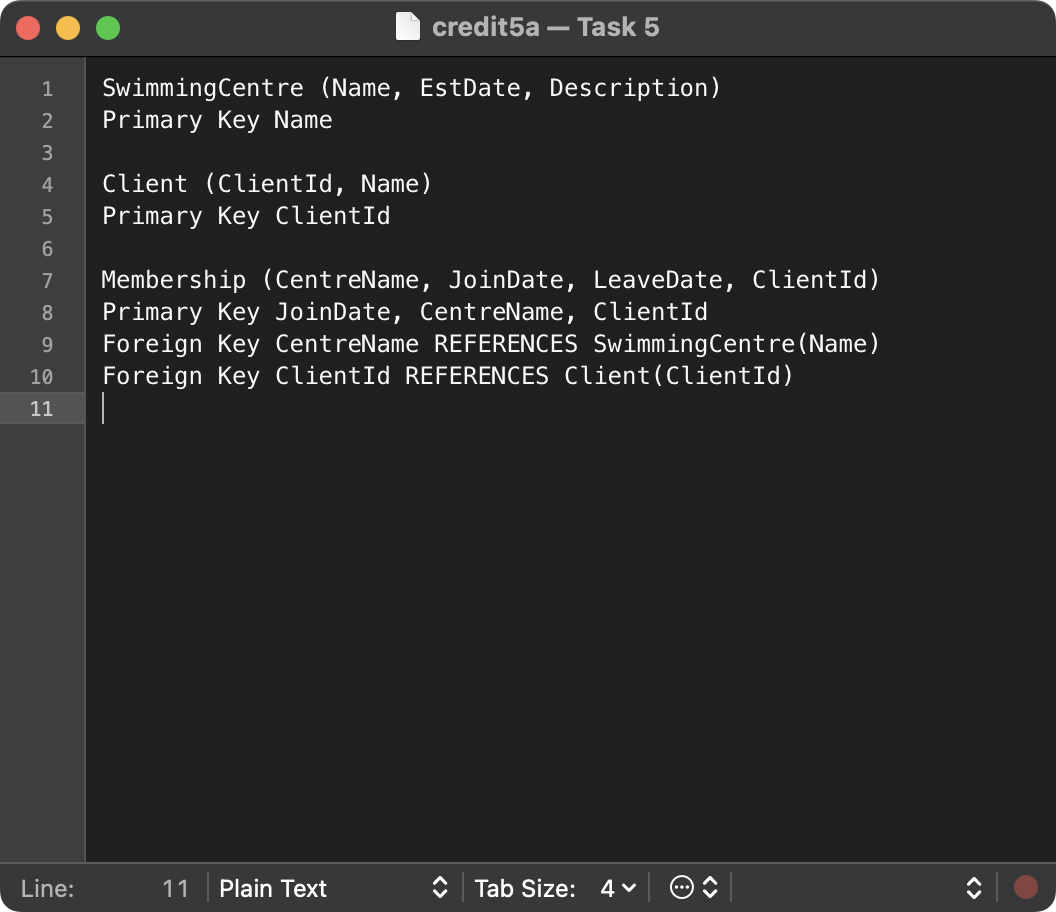
# Task 5 – Credit Submission

Student Number: 104071453  
Student Name: Marco Giacoppo

### Credit 5a

A diagram of a membership

Description automatically generated



|  |  |  |
| --- | --- | --- |
| **Centre Name** | **EstablishmentDate** | **Description** |
| Blue Lagoon | 2015-01-15 | Olympic size pool |
| Aqua Fun | 2018-05-20 | Fun activities pool |
| Deep Dive | 2020-11-05 | Diving training pool |

|  |  |
| --- | --- |
| **ClientId** | **Name** |
| 1 | John Doe |
| 2 | Marco Giacoppo |
| 3 | Corey Santarossa |

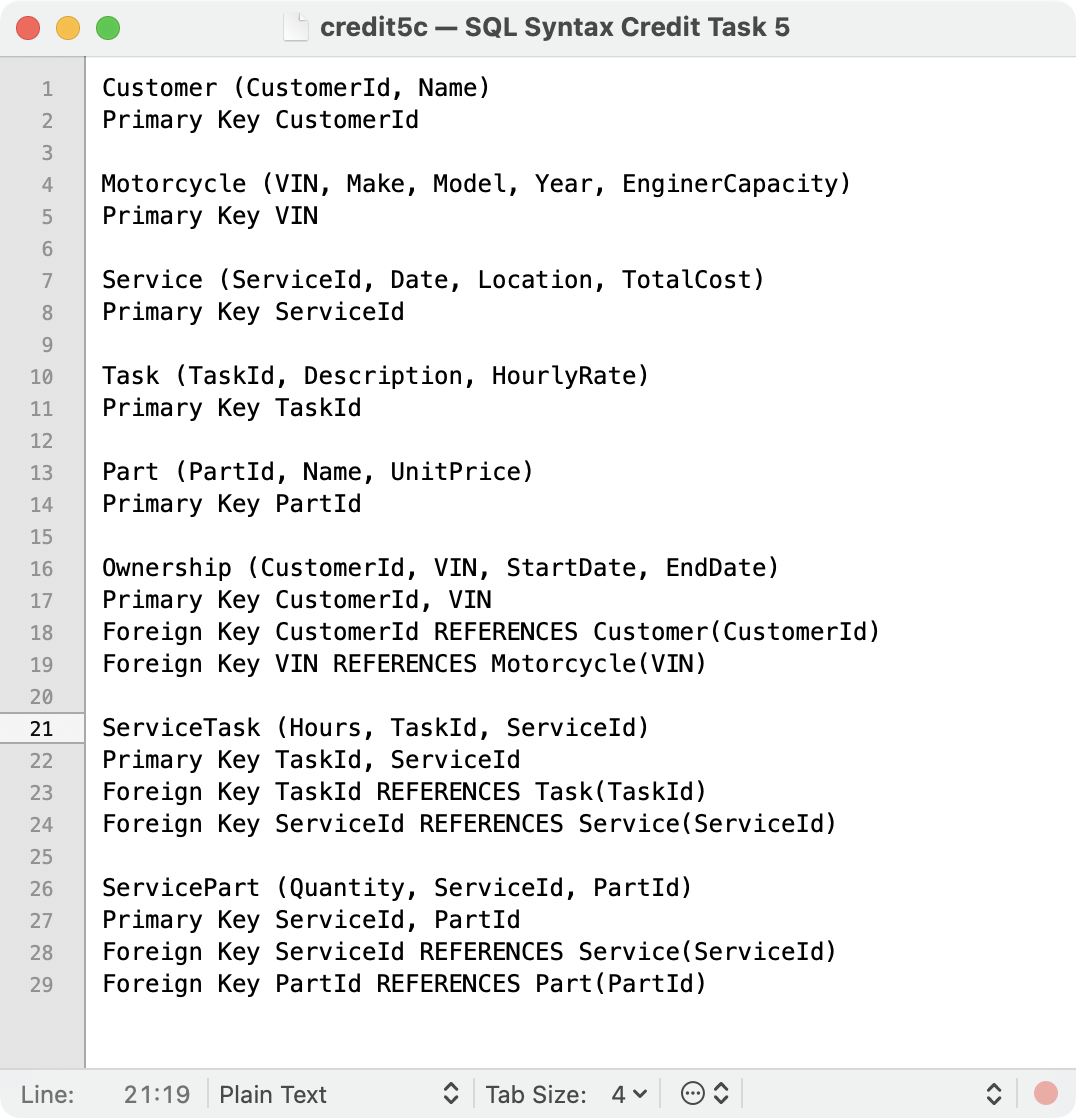
|  |  |  |  |
| --- | --- | --- | --- |
| **CentreName** | **ClientId** | **JoinDate** | **LeaveDate** |
| Blue Lagoon | 1 | 2022-04-12 | 2022-06-01 |
| Blue Lagoon | 1 | 2023-01-10 | NULL |
| Aqua Fun | 2 | 2022-07-15 | 2022-09-15 |
| Deep Dive | 3 | 2023-04-01 | NULL |

### Credit 5b

A diagram of a service

Description automatically generated

### Credit 5c



### Credit 5d

**Creating the Customer table**

CREATE TABLE Customer (

CustomerId INT PRIMARY KEY,

Name VARCHAR(255)

);

**Creating the Motorcycle table**

CREATE TABLE Motorcycle (

VIN VARCHAR(255) PRIMARY KEY,

Make VARCHAR(100),

Model VARCHAR(100),

Year INT,

EngineCapacity INT

);

**Creating the Service table**

CREATE TABLE Service (

ServiceId INT PRIMARY KEY,

Date DATE,

Location VARCHAR(255),

TotalCost DECIMAL(10, 2)

);

**Creating the Task table**

CREATE TABLE Task (

TaskId INT PRIMARY KEY,

Description VARCHAR(255),

HourlyRate DECIMAL(10, 2)

);

**Creating the Part table**

CREATE TABLE Part (

PartId INT PRIMARY KEY,

Name VARCHAR(255),

UnitPrice DECIMAL(10, 2)

);

**Creating the Ownership table**

CREATE TABLE Ownership (

CustomerId INT,

VIN VARCHAR(255),

StartDate DATE,

EndDate DATE,

PRIMARY KEY (CustomerId, VIN),

FOREIGN KEY (CustomerId) REFERENCES Customer(CustomerId),

FOREIGN KEY (VIN) REFERENCES Motorcycle(VIN)

);

**Creating the ServiceTask table**

CREATE TABLE ServiceTask (

TaskId INT,

ServiceId INT,

Hours DECIMAL(10, 2),

PRIMARY KEY (TaskId, ServiceId),

FOREIGN KEY (TaskId) REFERENCES Task(TaskId),

FOREIGN KEY (ServiceId) REFERENCES Service(ServiceId)

);

**Creating the ServicePart table**

CREATE TABLE ServicePart (

Quantity INT,

ServiceId INT,

PartId INT,

PRIMARY KEY (ServiceId, PartId),

FOREIGN KEY (ServiceId) REFERENCES Service(ServiceId),

FOREIGN KEY (PartId) REFERENCES Part(PartId)

);

### Credit 5e

**Customer Table**

|  |  |  |
| --- | --- | --- |
| **CustId** | **Name** | **Phone** |
| 125 | John Coles | 0401112233 |
| 278 | Erin Trump | 0466121455 |
| 721 | Emma Knox | 0423544117 |

**Car Table**

|  |  |
| --- | --- |
| **CarRego** | **MakeModel** |
| 1AU8HK | Mazda 3 |
| 1LM3AB | Hyundai i30 |
| 1KA2CA | Toyota Camry |
| 1CZ8JK | Mazda 3 |

**Rental Table**

|  |  |  |  |
| --- | --- | --- | --- |
| **CustId** | **CarRego** | **StartDate** | **ReturnDate** |
| 125 | 1AU8HK | 2020-08-31 | 2020-09-07 |
| 125 | 1LM3AB | 2020-11-14 | 2020-11-21 |
| 278 | 1AU8HK | 2020-09-12 | 2020-09-19 |
| 278 | 1KA2CA | 2020-10-01 | 2020-10-08 |
| 278 | 1CZ8JK | 2020-11-10 | 2020-11-12 |
| 278 | 1AU8HK | 2020-11-26 | 2020-12-01 |
| 721 | 1LM3AB | 2020-09-10 | 2020-09-13 |

### Credit 5f

\*start transaction for the purchase

### START TRANSACTION;

**UPDATE** Product

**SET** QtyInStock = QtyInStock – 2

**WHERE** ProdID = ‘G43546’;

\*check if the update was successful

**SELECT** QtyInStock **FROM** Product **WHERE** ProdId = ‘G43546’;

**COMMIT;**

**\***start transaction for the return

**START** TRANSACTION;

**UPDATE** Product

**SET** QtyInStock – QtyInStock + 1

**WHERE** ProdID = ‘G43546’;

\*check if the update was successful

**SELECT** QtyInStock **FROM** Product **WHERE** ProdID = ‘G43546’;

**COMMIT;**

Explanation of Transaction Commit and Rollback:

1. **Transaction Committed:**

A transaction is committed when all the operations within the transaction are successfully completed without any errors. Committing the transaction writes all changes made during the transaction to the database. This means that these changes become permanent and visible to other users.

1. **Transaction Rollback:**

A transaction is rolled back if any operation within the transaction fails or if a condition is met that invalidates the transaction (like insufficient stock). Rolling back a transaction undoes all changes made during the transaction, returning the database to its previous state before the transaction began.

**Smooth Operation (Commit):** If the update operations for both purchasing and returning the product adjust the ‘**QtyInStock’** correctly without violating any constraints (like inventory going below 0), the transactions are committed, making these changes permanent.

**Problematic Operation (Rollback):** If there’s an issue during the transaction, such as attempting to sell more units than are available in stock, the transaction should be rolled back to avoid corrupting the database state.