COMPX223-2023 Project Deliverable 1 report

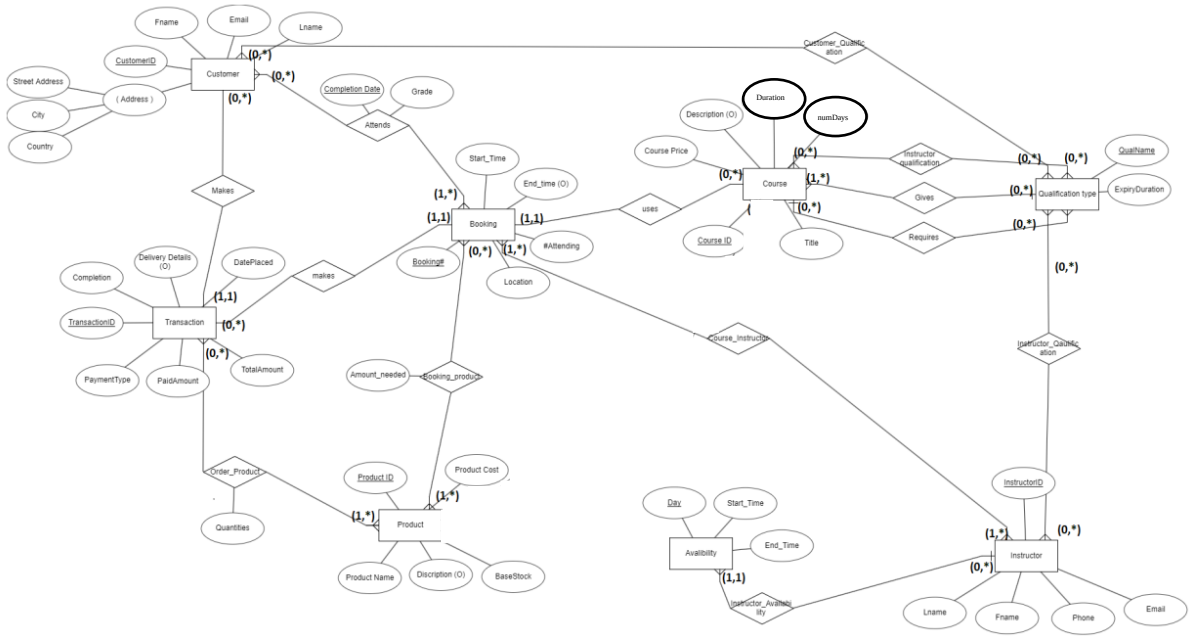
Group Name: Devo

Contribution table
Please specify the contribution (in percentage) each group member has made.

Name	Part 1	Part 2	Part 3	Part 4
Ben Bull	34%	34%	34%	34%
Morse McClennan	33%	33%	33%	33%
Samuel Lee	33%	33%	33%	33%

Note: Each group member is supposed to contribute to all four parts of the deliverable 1. You would not get any marks of a part on which you didn't do any work.

Part 1: ER Diagram (40 points)
Task 1: Draw the ER Diagram of the database that consists of entities, attributes, relationships and cardinalities, and underline the attribute(s) that will be used as primary key



Task 2. Explain the attributes that are not self-explanatory, for example, attributes with unclear names, that will be normalised, or have constraints on the values.

For customer there is a check for email that just checks that an email was entered same check for instructor email.

For weekdays in availability check that a name of a day is entered.

For course duration is how long it is each day and numDays is how many days it runs for.

For product_cost and course_cost check if more than zero same thing for total amount and paid amount in transactions and totalAmount is greater than or equal to paidAmount.

Also, in transactions check if payment type is EFPS, CASH or CRDT and paid amount stores the amount paid by the person totalAmount tracks the total cost without any discounts. Completion is for when courses has been completed or when products has been delivered.

For attend relationship grade is the grade someone gets in a course and completion date says whether some one completes the course by storing the date when completed.

For booking #attending is just amount of people the customer books the course for and start time and end time are when the course starts and ends.

Part 2: Relational Model (10 points)
Task: Create the relational model from the ER diagram developed in Part 1. Make sure to (1) underline primary keys, (2) point out the foreign keys and the relations they reference (look at Week 2 lecture slides on Relational Model for examples) and (3) normalise to 1NF.

Customer(CustomerID, Fname, Lname, Email, Phone, Street_Address, City, Country)

Instructor(InstuctorID, Fname, Lname, Email, Phone)

Product(Product_ID, Product_Name, Descriptions, BaseStock, Product_Cost)

Course(Course_ID, Title, Duration, numDays, Descriptions, Course_Price)

Availability(WeekDay, Start_Time, End_time, Instructor_ID)

QualificationType(QualName, ExpiryDuration)

Transactions(TransactionID, PaymentType, DatePlaced, Delivery_Details, TotalAmount, PaidAmount, Completion, CustomerID)

Booking(Booking#, Location, #Attending, Start_Time, End_Time, Course_ID, TransactionID)

Booking_Instructor(Booking#, InstructorID)

Attends(Booking#, CustomerID, Completion_Date, Grade)

BookingProduct(Booking#, Product_ID, Amount_Needed)

InstructorHasQualification(InstructorID, QualName)

CustomerHasQualification(CustomerID, qualName)

OrderProduct(TransactionID, Product_ID, Quantities)

InstructorRequiresQualification(Course_ID, QualName)

CustomerPrerequisites(Course_ID, QualName)

CourseGivesQualification(Course_ID, QualName)

Part 3: SQL Tables (20 points)
Task: Convert the relational model in Part 2 to SQL tables. Make sure to use appropriate data types, specify the primary and foreign keys, and include sensible check constraints.

```
--must have email so can be contacted by company
CREATE TABLE Customer
(
  Fname VARCHAR(50) NOT NULL,
  Lname VARCHAR(50) NOT NULL,
  Email VARCHAR(100) NOT NULL,
  Phone BIGINT,
  CustomerID INT NOT NULL IDENTITY(1,1),
  Street_Address VARCHAR(100),
  City VARCHAR(50),
  Country VARCHAR(50),
  PRIMARY KEY (CustomerID),
  CONSTRAINT Check_Customer_Email CHECK(Email LIKE '%__@__%')
);

--must have email so can be contacted by company
CREATE TABLE Instructor
(
  Fname VARCHAR(50) NOT NULL,
  Lname VARCHAR(50) NOT NULL,
  Email VARCHAR(100) NOT NULL,
  Phone BIGINT,
  InstructorID INT NOT NULL IDENTITY(1,1),
  PRIMARY KEY (InstructorID),
  CONSTRAINT Check_Instructor_Email
        CHECK(Email LIKE '%__@__%')
);

CREATE TABLE Product
(
  Product_Name VARCHAR(100) NOT NULL,
  Discriptions VARCHAR(1000),
  BaseStock INT NOT NULL,
  Product_ID INT NOT NULL IDENTITY(1,1),
  Product_Cost MONEY NOT NULL,
  PRIMARY KEY (Product_ID),
  CONSTRAINT Check_Product_Price
        CHECK(Product_cost > 0)
);
```

```
CREATE TABLE Course
(
  Title VARCHAR(100) NOT NULL,
  Duration TIME NOT NULL,
  NumDays INT NOT NULL,
  Descriptions VARCHAR(1000),
  Course_ID INT NOT NULL IDENTITY(1,1),
  Course_Price MONEY NOT NULL,
  PRIMARY KEY (Course_ID),
  CONSTRAINT Check_Course_Price
        CHECK(Course_Price > 0)
);

CREATE TABLE Avalibility
(
  WeekDays VARCHAR(10) NOT NULL,
  Start_Time TIME NOT NULL,
  End_Time TIME NOT NULL,
  InstructorID INT NOT NULL,
  PRIMARY KEY (WeekDays, InstructorID),
  FOREIGN KEY (InstructorID) REFERENCES Instructor(InstructorID),
  CONSTRAINT Check_Date CHECK(WeekDays IN
('Monday','Tuesday','Wednesday','Thursday','Friday','Saturday','Sunday'))
);

CREATE TABLE QualificationType
(
  QualName VARCHAR(150) NOT NULL,
  ExpiryDuration_YEARS INT NOT NULL,
  PRIMARY KEY (QualName)
);

--Paid amount and payment type track whether it has been paid as if null it's not paid
CREATE TABLE Transactions
(
  TransactionID INT NOT NULL IDENTITY(1,1),
  PaymentType CHAR(4),
  DatePlaced DATE NOT NULL,
  Delivery_Details VARCHAR(200),
  TotalAmount MONEY NOT NULL,
  PaidAmount MONEY,
  Completion BIT DEFAULT 0,
  CustomerID INT NOT NULL,
  PRIMARY KEY (TransactionID),
  FOREIGN KEY (CustomerID) REFERENCES Customer(CustomerID),
  CONSTRAINT check_total_paid CHECK(TotalAmount >= PaidAmount),
  CONSTRAINT check_paid CHECK(PaidAmount > 0),
  CONSTRAINT check_total CHECK(TotalAmount > 0),
  CONSTRAINT check_paymentType Check(PaymentType IN ('EFPS','CASH','CRDT'))
);
```

```
CREATE TABLE Booking
(
  Location VARCHAR(100) NOT NULL,
  #Attending INT NOT NULL,
  Booking# INT NOT NULL IDENTITY(1,1),
  Start_Time DATETIME NOT NULL,
  End_time DATETIME,
  TransactionID INT NOT NULL,
  Course_ID INT NOT NULL,
  PRIMARY KEY (Booking#),
  FOREIGN KEY (Course_ID) REFERENCES Course(Course_ID),
  FOREIGN KEY (TransactionID) REFERENCES Transactions(TransactionID)
);

CREATE TABLE BookingInstructor
(
  Booking# INT NOT NULL,
  InstructorID INT NOT NULL,
  PRIMARY KEY (Booking#, InstructorID),
  FOREIGN KEY (Booking#) REFERENCES Booking(Booking#),
  FOREIGN KEY (InstructorID) REFERENCES Instructor(InstructorID)
);

CREATE TABLE Attends
(
  Grade INT,
  Completion_Date DATE,
  CustomerID INT NOT NULL,
  Booking# INT NOT NULL,
  PRIMARY KEY (CustomerID, Booking#),
  FOREIGN KEY (CustomerID) REFERENCES Customer(CustomerID),
  FOREIGN KEY (Booking#) REFERENCES Booking(Booking#),
  UNIQUE (CustomerID, Booking#)
);

CREATE TABLE BookingProduct
(
  Amount_Needed INT NOT NULL,
  Booking# INT NOT NULL,
  Product_ID INT NOT NULL,
  PRIMARY KEY (Booking#, Product_ID),
  FOREIGN KEY (Booking#) REFERENCES Booking(Booking#),
  FOREIGN KEY (Product_ID) REFERENCES Product(Product_ID)
);

CREATE TABLE InstructorHasQaulification
(
  InstructorID INT NOT NULL,
  QualName VARCHAR(150) NOT NULL,
```

```
  PRIMARY KEY (InstructorID, QualName),
  FOREIGN KEY (InstructorID) REFERENCES Instructor(InstructorID),
  FOREIGN KEY (QualName) REFERENCES QualificationType(QualName)
);

CREATE TABLE OrderProduct
(
  Quantities INT NOT NULL,
  TransactionID INT NOT NULL,
  Product_ID INT NOT NULL,
  PRIMARY KEY (TransactionID, Product_ID),
  FOREIGN KEY (TransactionID) REFERENCES Transactions(TransactionID),
  FOREIGN KEY (Product_ID) REFERENCES Product(Product_ID)
);

CREATE TABLE InstructorRequiresQualification
(
  Course_ID INT NOT NULL,
  QualName VARCHAR(150) NOT NULL,
  PRIMARY KEY (Course_ID, QualName),
  FOREIGN KEY (Course_ID) REFERENCES Course(Course_ID),
  FOREIGN KEY (QualName) REFERENCES QualificationType(QualName)
);

CREATE TABLE CustomerPrerequisites
(
  Course_ID INT NOT NULL,
  QualName VARCHAR(150) NOT NULL,
  FOREIGN KEY (Course_ID) REFERENCES Course(Course_ID),
  FOREIGN KEY (QualName) REFERENCES QualificationType(QualName)
);

CREATE TABLE CourseGivesQualification
(
  Course_ID INT NOT NULL,
  QualName VARCHAR(150) NOT NULL,
  FOREIGN KEY (Course_ID) REFERENCES Course(Course_ID),
  FOREIGN KEY (QualName) REFERENCES QualificationType(QualName)
);
```

Part 4: Testing and reflection (30 points)

Task 1: Insert at least two rows into all tables.

```
insert into Customer values('John', 'Smith', 'johnsmith@gmail.com', 5551234576, '123 main st', 'anytown', 'NZ');
insert into Customer values('Sarsh', 'Johnson', 'sarahj@gmail.com', 5559876543, '456 Elm st', 'anytown', 'NZ');
```

	Fname	Lname	Email	Phone	CustomerID	Street_Address	City	Country
1	John	Smith	johnsmith@gmail.com	5551234576	1	123 main st	anytown	NZ
2	Sarsh	Johnson	sarahj@gmail.com	5559876543	2	456 Elm st	anytown	NZ

insert into Course values('basic first aid', '4:00:00', 1, 'A short course that covers essential first aid techniques for common
injuries, such as burns, cuts, and sprains. Suitable for individuals or groups with
no prior first aid training.', 200);

insert into Course values('First Aid/CPR/AED', '6:00:00', 1, 'A one-day course that teach individuals how to respond to medical
emergencies. The course covers a variety of topics, including the proper procedures
for providing first aid, cardiopulmonary resuscitation (CPR), and using an automated
external defibrillator (AED).
Suitable for individuals or groups with no prior first aid training.', 300);

insert into Course values('Wilderness First Aid', '8:00:00', 2, 'is two-day (8 hour days) specialized course designed to teach individuals how to
respond to medical emergencies in remote or wilderness settings. The course covers
the essential skills and techniques needed to assess, stabilize, and manage a
variety of injuries and illnesses that may occur in the outdoors.
', 400);

insert into Course values('Advanced First Aid', '8:00:00', 3, 'A 3-day (8 hour days) course that builds on the skills taught in the Basic First Aid
course and covers advanced techniques for managing medical emergencies, such as
heart attacks and strokes.', 400);

	Title	Duration	NumDays	Descriptions	Course_ID	Course_Price
1	basic first aid	04:00:00.0000000	1	A short course that covers essential first aid tech...	1	200.00
2	First Aid/CPR/AED	06:00:00.0000000	1	A one-day course that teach individuals how to r...	2	300.00
3	Wilderness First Aid	08:00:00.0000000	2	is two-day (8 hour days) specialized course desig...	3	400.00
4	Advanced First Aid	08:00:00.0000000	3	A 3-day (8 hour days) course that builds on the s...	4	400.00

insert into Product values('First Aid Kit', 'A portable kit containing a range of essential first aid supplies, including bandages, gauze, antiseptic wipes, and more.', 100, 50);

insert into Product values('Automated External Defibrillator (AED)', 'An easy-to-use device that can analyse the hearts rhythm and deliver
an electric shock to restore normal heartbeat in case of cardiac arrest.', 10, 1500);

insert into Product values('CPR Manikin', ' A realistic training manikin that simulates a human body for
practicing cardiopulmonary resuscitation (CPR) techniques.', 50, 300);

	Product_Name	Discriptions	BaseStock	Product_ID	Product_Cost
1	First Aid Kit	A portable kit containing a range of essential first...	100	1	50.00
2	Automated External Defibrillator (AED)	An easy-to-use device that can analyse the hear...	10	2	1500.00
3	CPR Manikin	A realistic training manikin that simulates a hum...	50	3	300.00

insert into Instructor values('Jane', 'Doe', 'janedoe@gmail.com', 5555551212);
insert into Instructor values('Mark', 'Johnson', 'markJohnson@gmail.com', 5551234567);

	Fname	Lname	Email	Phone	InstructorID
1	Jane	Doe	janedoe@gmail.com	5555551212	1
2	Mark	Johnson	markJohnson@gmail.com	5551234567	2

insert into Transactions values(null, '2023-01-03', 'Booked for anytown Community Center', 6000, null, 0, 1);
insert into Transactions values('EFPS', '2023-01-03', 'Booked for anytown state park', 6000, 6000, 0, 2);
insert into Transactions values(null, '2023-01-03', 'pick up', 50, null, 0, 1);
insert into Transactions values('EFPS', '2023-01-03', 'deliver to 456 elm st, anytown, nz', 300, 300, 1, 2);

	TransactionID	PaymentType	DatePlaced	Delivery_Details	TotalAmount	PaidAmount	Completion	CustomerID
1	1	NULL	2023-01-03	Booked for anytown Community Center	6000.00	NULL	0	1
2	2	EFPS	2023-01-03	Booked for anytown state park	6000.00	6000.00	0	2
3	3	NULL	2023-01-03	pick up	50.00	NULL	0	1
4	4	EFPS	2023-01-03	deliver to 456 elm st, anytown, nz	300.00	300.00	1	2

insert into Booking values('Anytime community centre', 20, '2023-03-01 09:00:00', '2023-03-01 15:00:00', 1, 1);
insert into Booking values('Anytime state park', 15, '2023-04-15 09:00:00', '2023-04-16 17:00:00', 2, 2);

	Location	#Attending	Booking#	Start_Time	End_time	TransactionID	Course_ID
1	Anytime community centre	20	1	2023-03-01 09:00:00.000	2023-03-01 15:00:00.000	1	1
2	Anytime state park	15	2	2023-04-15 09:00:00.000	2023-04-16 17:00:00.000	2	2

insert into BookingInstructor values(1, 1);
insert into BookingInstructor values(2, 2);

	Booking#	InstructorID
1	1	1
2	2	2

insert into Avalibility values('Monday', '9:00:00', '17:00:00', 1);
insert into Avalibility values('Tuesday', '9:00:00', '17:00:00', 1);
insert into Avalibility values('Wednesday', '9:00:00', '17:00:00', 1);
insert into Avalibility values('Thursday', '9:00:00', '17:00:00', 1);
insert into Avalibility values('Friday', '9:00:00', '17:00:00', 1);
insert into Avalibility values('Saturday', '9:00:00', '17:00:00', 2);
insert into Avalibility values('Sunday', '9:00:00', '17:00:00', 2);

	WeekDays	Start_Time	End_Time	InstructorID
1	Friday	09:00:00.0000000	17:00:00.0000000	1
2	Monday	09:00:00.0000000	17:00:00.0000000	1
3	Saturday	09:00:00.0000000	17:00:00.0000000	2
4	Sunday	09:00:00.0000000	17:00:00.0000000	2
5	Thursday	09:00:00.0000000	17:00:00.0000000	1
6	Tuesday	09:00:00.0000000	17:00:00.0000000	1
7	Wednesd...	09:00:00.0000000	17:00:00.0000000	1

insert into QualificationType values('basic first aid', 2);
insert into QualificationType values('First Aid/CPR/AED', 2);
insert into QualificationType values('Wilderness First Aid', 2);

insert into QualificationType values('Advanced First Aid', 2);
insert into QualificationType values('American Heart Association BLS Instructor', 2);
insert into QualificationType values('Red Cross First Aid/CPR/AED Instructor', 2);
insert into QualificationType values('Wilderness First Responder Instructor', 2);
insert into QualificationType values('National Safety Council First Aid/CPR/AED', 2);
insert into QualificationType values('Advanced First Aid Instructor', 2);

	QualName	ExpiryDuration_YEARS
1	Advanced First Aid	2
2	Advanced First Aid Instructor	2
3	American Heart Associatio...	2
4	basic first aid	2
5	First Aid/CPR/AED	2
6	National Safety Council Firs...	2
7	Red Cross First Aid/CPR/A...	2
8	Wilderness First Aid	2
9	Wilderness First Responder...	2

insert into Attends values(null, null, 1, 1);
insert into Attends values(null, null, 2, 2);
insert into Attends values(null, null, 1, 2);

	Grade	Completion_Date	CustomerID	Booking#
1	NULL	NULL	1	1
2	NULL	NULL	1	2
3	NULL	NULL	2	2

insert into BookingProduct values(10, 1, 1);
insert into BookingProduct values(10, 2, 1);
insert into BookingProduct values(2, 2, 2);
insert into BookingProduct values(5, 2, 3);

	Amount_Needed	Booking#	Product_ID
1	10	1	1
2	10	2	1
3	2	2	2
4	5	2	3

insert into InstructorHasQaulification values(1, 'American Heart Association BLS Instructor');
insert into InstructorHasQaulification values(1, 'Red Cross First Aid/CPR/AED Instructor');
insert into InstructorHasQaulification values(2, 'Wilderness First Responder Instructor');
insert into InstructorHasQaulification values(2, 'National Safety Council First Aid/CPR/AED');

	InstructorID	QualName
1	1	American Heart Association BLS Instructor
2	1	Red Cross First Aid/CPR/AED Instructor
3	2	National Safety Council First Aid/CPR/AED
4	2	Wilderness First Responder Instructor

insert into OrderProduct values(1, 3, 1);
insert into OrderProduct values(1, 4, 3);

	Quantities	TransactionID	Product_ID
1	1	3	1
2	1	4	3

insert into InstructorRequiresQualification values(1, 'American Heart Association BLS Instructor');
insert into InstructorRequiresQualification values(2, 'Red Cross First Aid/CPR/AED Instructor');
insert into InstructorRequiresQualification values(3, 'Wilderness First Responder Instructor');
insert into InstructorRequiresQualification values(4, 'Advanced First Aid Instructor');

	Course_ID	QualName
1	1	American Heart Association BLS Instructor
2	2	Red Cross First Aid/CPR/AED Instructor
3	3	Wilderness First Responder Instructor
4	4	Advanced First Aid Instructor

insert into CustomerPrerequisites values(4, 'basic first aid');

	Course_ID	QualName
1	4	basic first aid

insert into CourseGivesQualification values(1, 'basic first aid');
insert into CourseGivesQualification values(2, 'First Aid/CPR/AED');
insert into CourseGivesQualification values(3, 'Wilderness First Aid');
insert into CourseGivesQualification values(4, 'Advanced First Aid');

	Course_ID	QualName
1	1	basic first aid
2	2	First Aid/CPR/AED
3	3	Wilderness First Aid
4	4	Advanced First Aid

Task 2: Write a query that retrieves data from all tables. For example, retrieve the details of each customer including the details of courses they have attended and equipment they have purchased. Show the results.

```
--links all entities together
select distinct cus.Fname as 'Customer Fname', B.Booking#, T.TransactionID, c.Title,
I.Fname as 'Instructor Fname', Av.Start_Time,  Q.QualName, BP.Amount_Needed,
P.Product_Name
from Customer cus, Instructor I, Course c, Avalibility Av, Transactions T, Booking B,
BookingInstructor BI, Attends A, QualificationType Q, InstructorHasQaulification IHQ,
BookingProduct BP, Product P
where cus.CustomerID = A.CustomerID and cus.CustomerID = T.CustomerID and
T.TransactionID = B.TransactionID and B.Course_ID = C.Course_ID and B.Booking# =
BI.Booking#
and BI.InstructorID = I.InstructorID and I.InstructorID = Av.InstructorID and
Q.QualName = IHQ.QualName and I.InstructorID = IHQ.InstructorID and B.Booking# =
BP.Booking# and BP.Product_ID = P.Product_ID;
```

	Customer Fname	Booking#	TransactionID	Title	Instructor Fname	Start_Time	QualName	Amount_Needed	Product_Name
1	John	1	1	basic first aid	Jane	09:00:00.0000000	American Heart Association BLS Instructor	10	First Aid Kit
2	John	1	1	basic first aid	Jane	09:00:00.0000000	Red Cross First Aid/CPR/AED Instructor	10	First Aid Kit
3	Sarsh	2	2	First Aid/CPR/AED	Mark	09:00:00.0000000	National Safety Council First Aid/CPR/AED	2	Automated External Defibrillator (AED)
4	Sarsh	2	2	First Aid/CPR/AED	Mark	09:00:00.0000000	National Safety Council First Aid/CPR/AED	5	CPR Manikin
5	Sarsh	2	2	First Aid/CPR/AED	Mark	09:00:00.0000000	National Safety Council First Aid/CPR/AED	10	First Aid Kit
6	Sarsh	2	2	First Aid/CPR/AED	Mark	09:00:00.0000000	Wilderness First Responder Instructor	2	Automated External Defibrillator (AED)
7	Sarsh	2	2	First Aid/CPR/AED	Mark	09:00:00.0000000	Wilderness First Responder Instructor	5	CPR Manikin
8	Sarsh	2	2	First Aid/CPR/AED	Mark	09:00:00.0000000	Wilderness First Responder Instructor	10	First Aid Kit

Task 3: Write a query that selects each class, showing details about the course, attending students and instructors, the first aid supplies used, and if the class has been paid for yet. Show the results.

```
--get course info and how is attending plus instructors and product used
select distinct C.Title, C.Descriptions, B.Location, B.Start_Time, B.End_time,
C.Course_Price, T.PaidAmount, Cus.Fname as 'Customer fname', I.Fname as 'Instuctor
fname', P.Product_Name
from Course C, Attends A, Customer Cus, Booking B, Transactions T, Instructor I,
BookingInstructor BI, BookingProduct BP, Product P
where A.CustomerID = Cus.CustomerID and C.Course_ID = B.Course_ID and A.Booking# =
B.Booking# and T.TransactionID = B.TransactionID
and B.Booking# = BI.Booking# and BI.InstructorID = I.InstructorID and B.Booking# =
BP.Booking# and P.Product_ID = BP.Booking#;
```

	Title	Descriptions	Location	Start_Time	End_time	Course_Price	PaidAmount	Customer fname	Instuctor fname	Product_Name
1	basic first aid	A short course that covers essential first aid tech...	Anytime community centre	2023-03-01 09:00:00.000	2023-03-01 15:00:00.000	200.00	NULL	John	Jane	First Aid Kit
2	First Aid/CPR/AED	A one-day course that teach individuals how to r...	Anytime state park	2023-04-15 09:00:00.000	2023-04-16 17:00:00.000	300.00	6000.00	John	Mark	Automated External Defibrillator (AED)
3	First Aid/CPR/AED	A one-day course that teach individuals how to r...	Anytime state park	2023-04-15 09:00:00.000	2023-04-16 17:00:00.000	300.00	6000.00	Sarsh	Mark	Automated External Defibrillator (AED)

Task 4: Discuss the problems or challenges encountered when designing the database (between 200-300 words). This can include the following:

- Any assumptions you have made based on insufficient information from the client.
- Any limitations of your design that affect what data can be stored or could cause anomalies that should be addressed when designing the application.
- Any changes made to the design (ER, Relational model, or table definition) and why these changes were made.

Assumptions:

- People can join the company with other qualifications and be added to the database.
- Qualification checks completed at application level.
- Instructors have one availability time per day.
- Start and End time for courses is selected at application level based on availability.
- Discounts completed at application level; totalAmount and paidAmount track the difference.
- Client makes booking on behalf of a company.
- Products required are organised by staff on the day.
- 1 day = 8-hour session.

Limitations:

Adding people that have other qualifications to the database could lead to incomplete or inconsistent data. Qualification checks completed at application level may also be inefficient.

The process of selecting start and end times could be inefficient and could lead to Instructors being scheduled inefficiently due to limited availability times.

The database may be limited in its ability to offer complex discounts since they are completed at application level.

If products are organised by staff on the day, problems may arise regarding inventory, which could further lead to problems during the booking process.

Changes:

We thought our ER diagram was large. We tried to compress it but found less tables and more attributes in other tables led to lots of null values. Avoiding this, we created separate tables for Availability and Qualifications instead of storing these as attributes in other tables.

After a few iterations we had two tables covering payment - one for bookings, and one for product. We encountered difficulties later because of identical values in these tables. To fix this we made one table that covered transactions for both bookings and product.

During a meeting finalizing the ER diagram we couldn't decide whether we needed a booking/product relationship, since we already had a course/product relationship. We learned from Colin that both relationships could exist and would help specify product required so we added them.

Appendix: marking rubrics (100 points)

Part 1: ER Diagram (40 points)

Entities	Excellent use of entities, including using entities to capture temporal data correctly, using entities to replace 'plural' attributes or attributes with predefined elements. **10 points**	One major entity is missing. Or one or two unnecessary entities are included **8 points**	Two major entities missing. Many entities that shouldn't be included. **5 points**	Majority of entities are used incorrectly. **2 points**	No ER diagram **0 points**
Attributes	All attributes attached to correct entities/ relationships. All key attributes underlined. Attributes that not self-explanatory are explained. **10 points**	Majority of attributes attached to correct entities/ relationships. Not all key attributes underlined. Some attributes that are not self-explanatory are not explained **8 points**	Some major attributes are missing or attached to incorrect entities/ relationship. Some key attributes underlined incorrectly or not at all. Attributes that	No key attributes. Not enough attributes for basic functionality of application. **2 points**	No attribute. Attributes attached to other attributes. **0 points**

			are not self-explanatory are not explained. **5 points**		
Relationships	Excellent use of relationships. **10 points**	One or two required relationships are missing. There are minor mistakes like circular references or three-way relationships not used correctly. **8 points**	Some incorrect relationships. e.g. user interactions with the application modelled as relationships **5 points**	Entities not connected correctly using relationship. Relationships connected to other relationships. **2 points**	No relationships **0 points**
Cardinalities	Excellent use of cardinalities. For example, design shows consideration of the difference between (0,*) and (1,*) or using correct cardinalities for three-way relationships. **10 points**	Minor errors in cardinalities. For example, one or two missing cardinalities or cardinalities in incorrect places **8 points**	A number of missing or incorrect cardinalities. All cardinalities completely backwards. **5 points**	Many missing cardinalities. Many incorrect cardinalities. **2 points**	Few or no cardinalities. **0 points**

Part 2: ER Diagram to Relational Model (10 points)

ER diagram fully converted into relations. Primary keys underlined. Foreign keys included correctly. Attributes normalised appropriately. **10 points**	Minor errors in relations. Minor errors in foreign keys or attribute normalisation. **8 points**	One or two major relations are missing. Some attributes missing. Or major errors in foreign keys **5 points**	Major relations missing. **2 points**	No relational model **0 points**

Part 3: Relational Model to database tables (20 points)

Complete table definitions. Appropriate data types used. Correct primary keys and references. Not null used correctly. Includes sensible check constraints. **20 points**	Minor errors in table definitions. For example, in appropriate data types used, or errors in primary keys and references, or errors in check constraints **16 points**	Some errors in table definitions, including errors in data types, missing primary keys or references, no check constraints **10 points**	Many errors in Table definitions. Table definitions do not match relational schema. **4 points**	No table definitions provided. **0 points**

Part 4: Testing and reflection (30 points)

Insert Statements	All tables contain meaningful data. Insert statements provided. **10 points**	Missing data in one or two tables contain data. Data doesn't match client requirements. Insert statements provided. **8 points**	Missing data in a number of tables. Insert statements provided. **5 points**	Query results showing data in table but no insert statements provided. **2 points**	No data inserted into table. **0 points**

Query	Tables joined correctly. SQL query and result provided. **10 points**	Tables joined with minor errors. SQL query and result provided. **8 points**	Tables joined with minor errors. Only SQL query or result provided. **5 points**	Simple query that does not join tables. **2 points**	Missing query **0 points**
Reflection	Sufficient descriptions about changes made and the reasons of changes. Or Sufficient descriptions of problem and challenges encountered and how they were addressed. **10 points**	Only descriptions about changes made are given, reasons are not provided. Or only descriptions of problem and challenges encountered and how they were addressed are not clear. **8 points**	Reflection is brief. **5 points**	Reflection is minimal. **2 points**	No reflectio n **0 points**