29633108—SQL Programming & Creative Writing

Task 1

Considering the attached Diagram; provide SQL statements for the following cases:

- Create relations for the attached Diagram.
- After creating each relation, insert at least 6 rows (tuples) for each relation (table) with data that you invented. (`SALE_ITEM` relation should include at least two same `SaleID`).
- Update a "Price" value of a row in the `ITEM` relation. (Updated `ITEM` (`ItemID`) should be in `SALE_ITEM`).
- Your response should include the screen-shots of your relations with data.
    - Discuss issues while solving above problems and any assumptions that you made.

> *Cite any source in APA format.*

While using MySQL:

Create relations for attached diagram

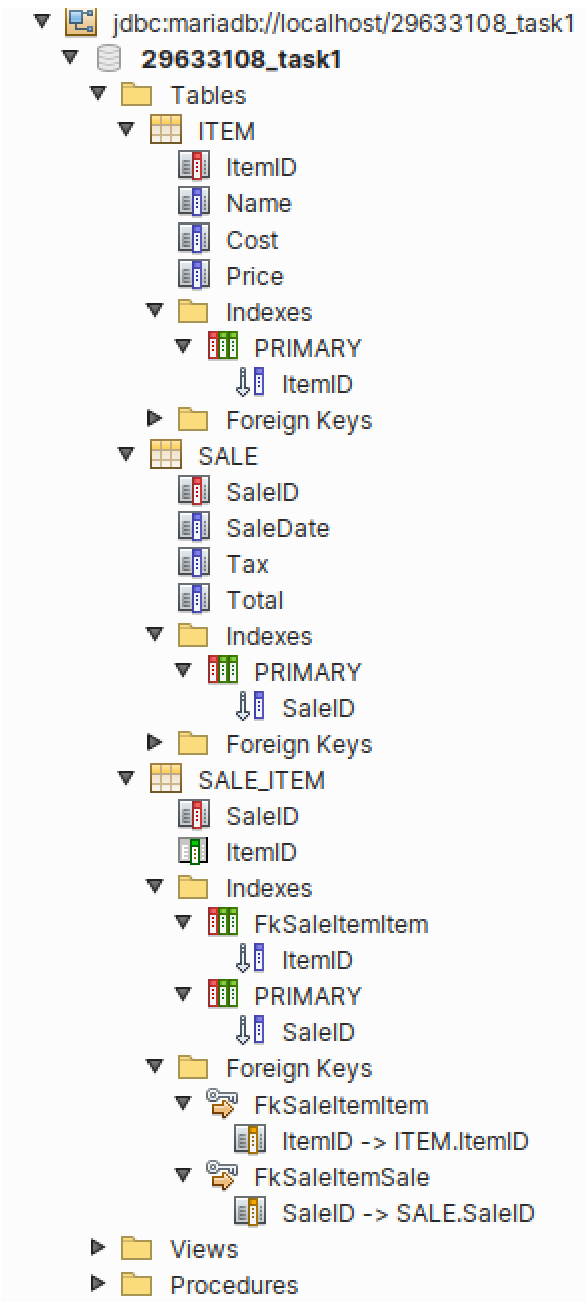
SQL:

```
CREATE TABLE SALE (
    SaleID INT NOT NULL,
    SaleDate DATE NOT NULL,
    Tax DECIMAL(5,2) NOT NULL,
    Total DECIMAL(5,2) NOT NULL,
    PRIMARY KEY (SaleID)
);

CREATE TABLE ITEM (
    ItemID INT NOT NULL,
    Name VARCHAR(255) NOT NULL,
    Cost DECIMAL(5,2) NOT NULL,
    Price DECIMAL(5,2) NOT NULL,
    PRIMARY KEY (ItemID)
);

CREATE TABLE SALE_ITEM (
    SaleID INT NOT NULL,
    ItemID INT NOT NULL,
    PRIMARY KEY (SaleID),
    CONSTRAINT FkSaleItemSale FOREIGN KEY (SaleID) REFERENCES SALE (SaleID),
    CONSTRAINT FkSaleItemItem FOREIGN KEY (ItemID) REFERENCES ITEM (ItemID)
);
```

Result:



- jdbc:mariadb://localhost/29633108_task1
  - **29633108_task1**
    - Tables
      - ITEM
        - ItemID
        - Name
        - Cost
        - Price
        - Indexes
          - PRIMARY
            - ItemID
        - Foreign Keys
      - SALE
        - SaleID
        - SaleDate
        - Tax
        - Total
        - Indexes
          - PRIMARY
            - SaleID
        - Foreign Keys
      - SALE_ITEM
        - SaleID
        - ItemID
        - Indexes
          - FkSaleItemItem
            - ItemID
          - PRIMARY
            - SaleID
        - Foreign Keys
          - FkSaleItemItem
            - ItemID -> ITEM.ItemID
          - FkSaleItemSale
            - SaleID -> SALE.SaleID
    - Views
    - Procedures

Insert rows

SQL:

```sql
SET AUTOCOMMIT=0;
INSERT INTO SALE VALUES
(1, '2021-12-17', 7.33, 61.46),
(2, '2021-03-04', 6.33, 50.93),
(3, '2021-11-30', 8.97, 37.95),
(4, '2021-07-12', 7.31, 31.69),
(5, '2021-05-21', 5.64, 82.86),
(6, '2021-08-25', 8.75, 78.04),
(7, '2021-08-08', 9.43, 87.95),
(8, '2021-10-06', 8.20, 81.81),
(9, '2022-01-16', 8.90, 42.97),
(10, '2021-04-10', 6.46, 40.07);
COMMIT;

SET AUTOCOMMIT=0;
INSERT INTO ITEM VALUES
(1, 'Aerodynamic Leather Clock', 34.22, 81.48),
(2, 'Lightweight Cotton Shirt', 47.07, 91.48),
(3, 'Awesome Concrete Table', 28.13, 64.16),
(4, 'Rustic Iron Chair', 42.41, 85.37),
(5, 'Small Linen Chair', 23.44, 79.91),
(6, 'Fantastic Paper Plate', 34.78, 83.43),
(7, 'Ergonomic Aluminum Clock', 25.95, 61.16),
(8, 'Synergistic Wool Computer', 41.82, 79.90),
(9, 'Small Plastic Coat', 46.92, 97.44),
(10, 'Intelligent Iron Plate', 43.97, 62.28);
COMMIT;

SET AUTOCOMMIT=0;
INSERT INTO SALE_ITEM VALUES
(5, 5),
(5, 2),
(10, 9),
(6, 8),
(1, 7),
(8, 3),
(2, 1),
(4, 6),
(7, 4),
(9, 10);
COMMIT;
```

Result:

- Failed to insert the tuple `(5, 2)` in the `SALE_ITEM` relation immediately after the insertion of the tuple `(5, 5)`. The duplicate entry of `ItemID` with the value `5` violated the key integrity constraint (Elmasri & Navathe, 2016).

```
[54:1] Failed in 0 s.
[Exception, Error code 1,062, SQLState 23000] (conn=3) Duplicate entry '5' for key 'PRIMARY'
 Line 54, column 1
```

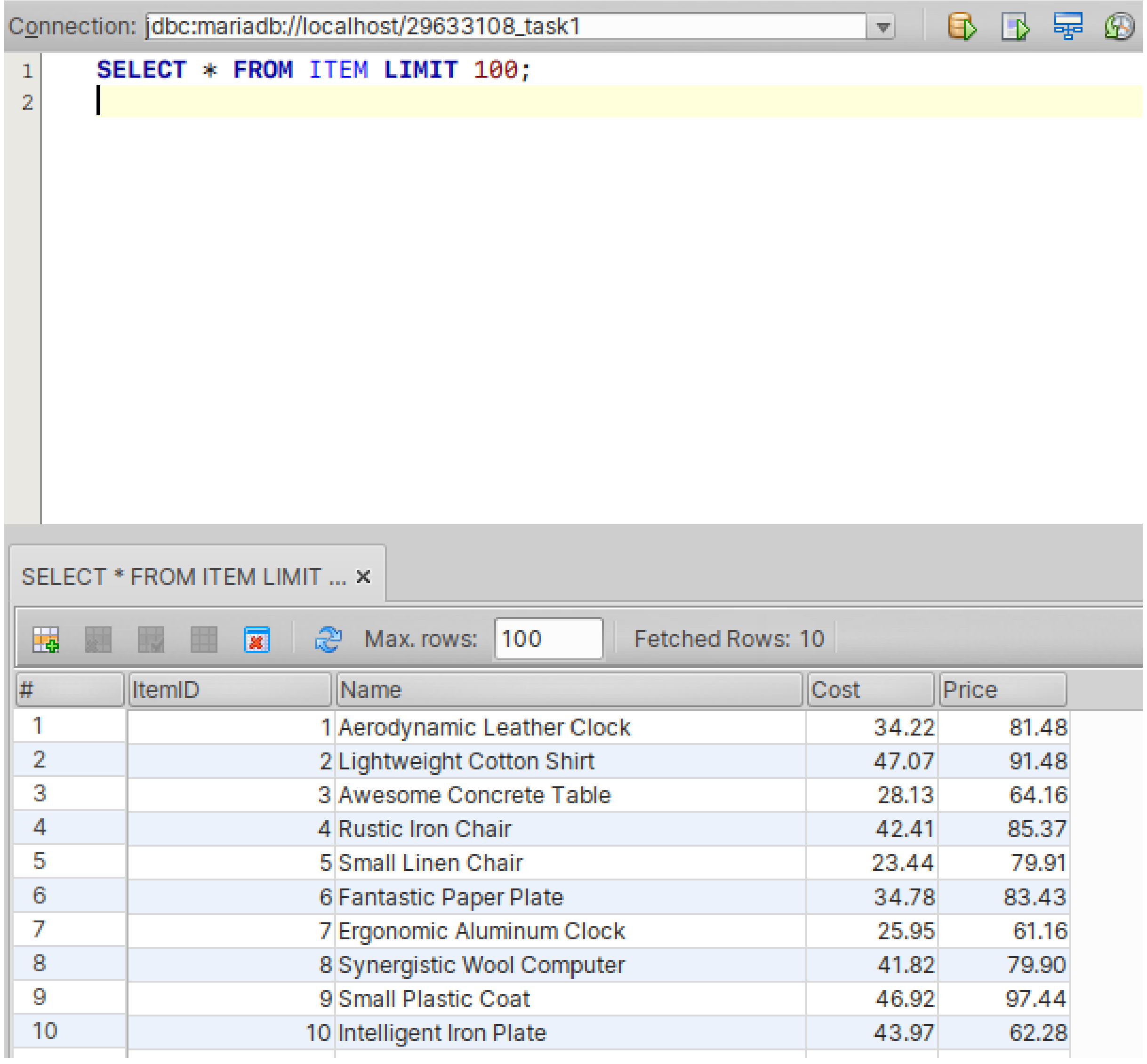
Fixing this error required the creation of a new primary key for the `SALE_ITEM` relation. Because the relation acted as a bridging table for the `SALE` and `ITEM` relations, its primary key needed to be declared as a combination of the `SaleID` and `ItemID` relations.

Thus, the SQL DDL for creating the `SALE_ITEM` relation had to be modified to read as:

```
CREATE TABLE SALE_ITEM (
    SaleID INT NOT NULL,
    ItemID INT NOT NULL,
    PRIMARY KEY (SaleID, ItemID),
    CONSTRAINT FkSaleItemSale FOREIGN KEY (SaleID) REFERENCES SALE (SaleID),
    CONSTRAINT FkSaleItemItem FOREIGN KEY (ItemID) REFERENCES ITEM (ItemID)
);
```

With this new modification, the insertion did not throw any errors and the resulting tables' data was:

`ITEM` data



Connection: jdbc:mariadb://localhost/29633108_task1

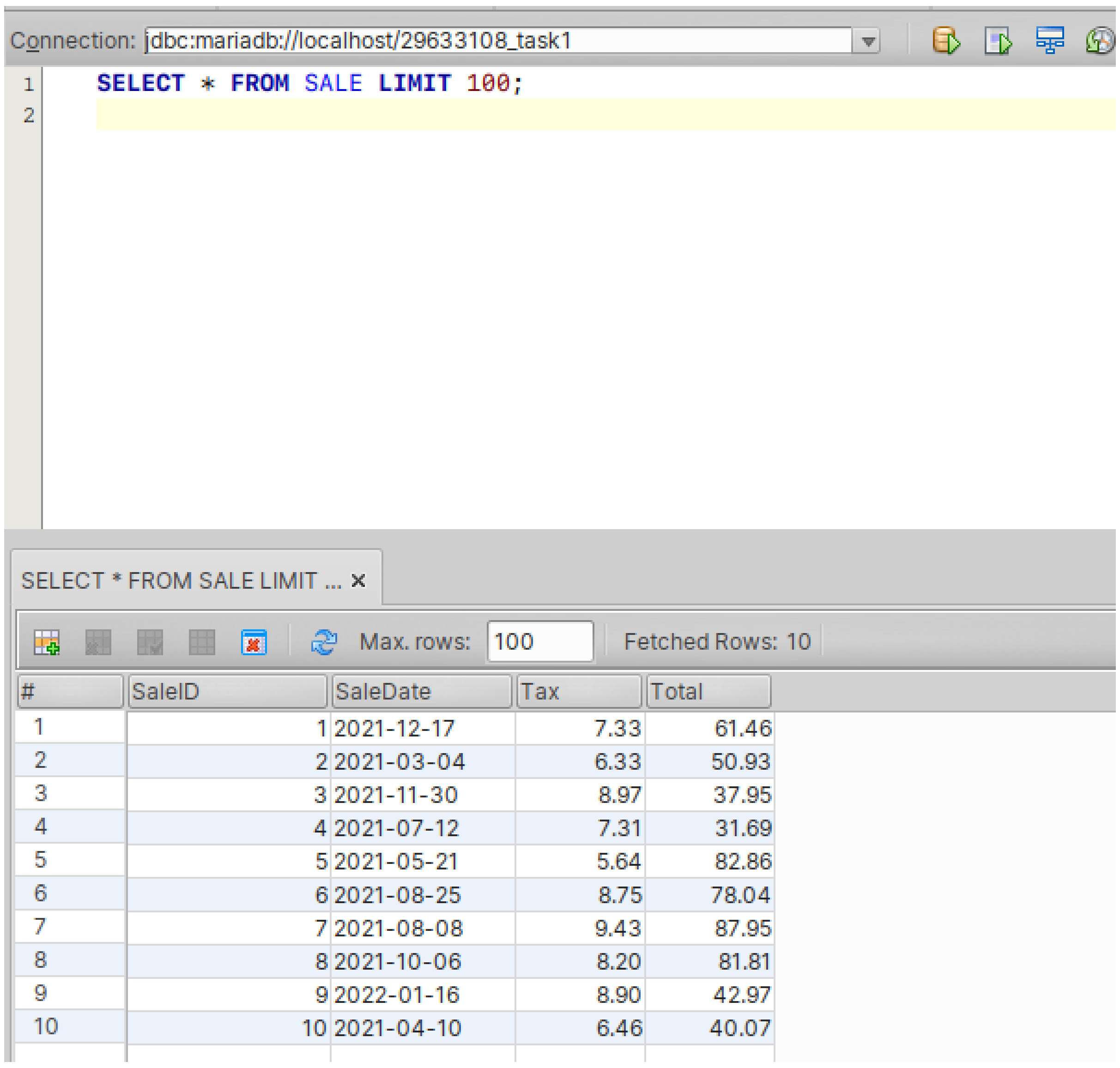
```
SELECT * FROM ITEM LIMIT 100;
```

SELECT * FROM ITEM LIMIT ... ×

Max. rows: 100 Fetched Rows: 10

#	ItemID	Name	Cost	Price
1	1	Aerodynamic Leather Clock	34.22	81.48
2	2	Lightweight Cotton Shirt	47.07	91.48
3	3	Awesome Concrete Table	28.13	64.16
4	4	Rustic Iron Chair	42.41	85.37
5	5	Small Linen Chair	23.44	79.91
6	6	Fantastic Paper Plate	34.78	83.43
7	7	Ergonomic Aluminum Clock	25.95	61.16
8	8	Synergistic Wool Computer	41.82	79.90
9	9	Small Plastic Coat	46.92	97.44
10	10	Intelligent Iron Plate	43.97	62.28

`SALE` data

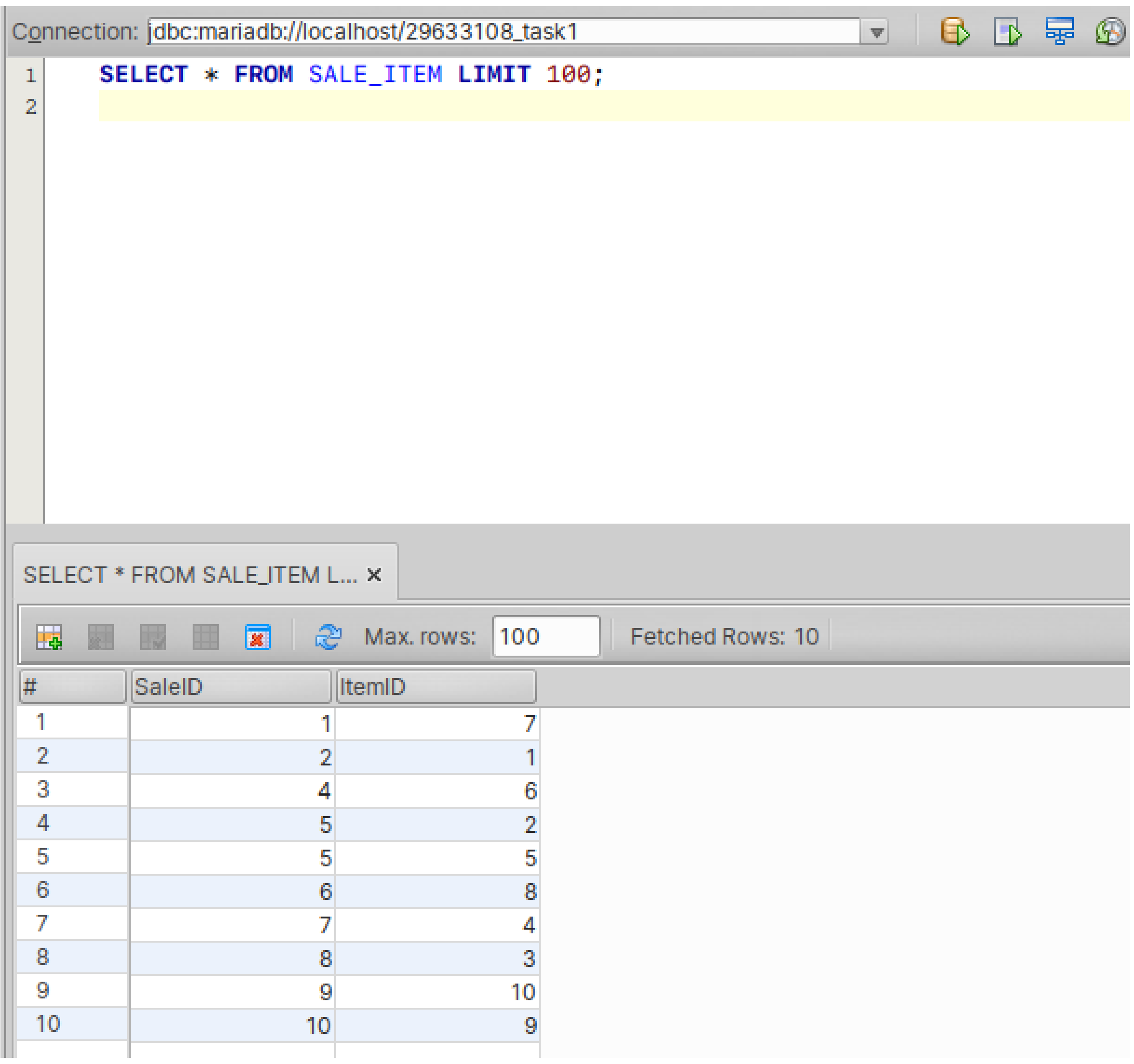


Connection: jdbc:mariadb://localhost/29633108_task1
SELECT * FROM SALE LIMIT ... ×
Max. rows: 100
Fetched Rows: 10

```sql
SELECT * FROM SALE LIMIT 100;
```

#	SaleID	SaleDate	Tax	Total
1	1	2021-12-17	7.33	61.46
2	2	2021-03-04	6.33	50.93
3	3	2021-11-30	8.97	37.95
4	4	2021-07-12	7.31	31.69
5	5	2021-05-21	5.64	82.86
6	6	2021-08-25	8.75	78.04
7	7	2021-08-08	9.43	87.95
8	8	2021-10-06	8.20	81.81
9	9	2022-01-16	8.90	42.97
10	10	2021-04-10	6.46	40.07

`SALE_ITEM` data



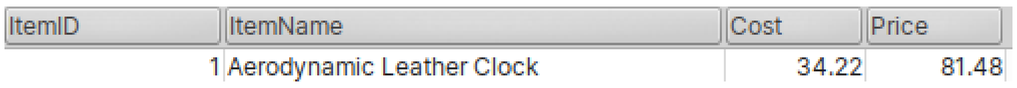
Connection: jdbc:mariadb://localhost/29633108_task1
SELECT * FROM SALE_ITEM L... ×
Max. rows: 100
Fetched Rows: 10

```sql
SELECT * FROM SALE_ITEM LIMIT 100;
```

#	SaleID	ItemID
1	1	7
2	2	1
3	4	6
4	5	2
5	5	5
6	6	8
7	7	4
8	8	3
9	9	10
10	10	9

Update `Price` in `ITEM` table

Before the update, the row with an `ItemID` value of `1` in the `ITEM` relation read:

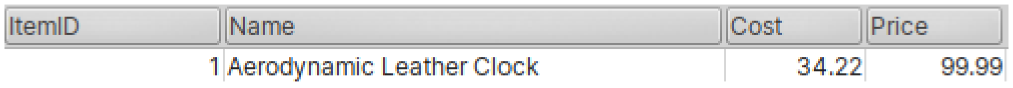


ItemID	ItemName	Cost	Price
1	Aerodynamic Leather Clock	34.22	81.48

After an update using the SQL script:

```sql
UPDATE ITEM SET Price = 99.99 WHERE ItemID = 1;
```

The same row read as:



ItemID	Name	Cost	Price
1	Aerodynamic Leather Clock	34.22	99.99

Even with the successful execution of the `Price` attribute update, the `SALE_ITEM` relation that contained the row with a value of `1` for its `ItemID` was not affected because no key nor referential integrities were violated (Elmasri & Navathe, 2016).

Reference

Elmasri, R., & Navathe, S. (2016). *Fundamentals of Database Systems*. Pearson.

Task 2

Using the attached Task 2, create the SQL DDL statements necessary to implement your database schema as an OpenOffice database. You may also implement your database in MySQL, IBM DB2 Express, Microsoft Access, SQL Server if you have access to these database systems. Your assignment must include a document that contains all of the SQL statements that you created and a screen-shot that shows the structures that you implemented in the database of your choice. Your DDL statements must accommodate the following elements:

- Create statements to create tables from the entities defined in the attached Task 2.
- Appropriate use of Null (and Not Null) parameters to ensure data validity.
- Appropriate use of constraint clauses to implement appropriate referential integrity.
- Use of data types and formats that is appropriate for the data in your database schema.
- Appropriate use of keys including automatic generation of key values if appropriate.

Instructions:

- The assignment must include the SQL DDL statements required to implement at least the following relations: `Doctor`, `Patient`, `Appointment`, `Specialty`, `PatientMedicine`, `Medicine`, `PatientAllergy`, `Allergy`.
- The submission must make appropriate use of Null (and Not Null) parameters to ensure data validity. The minimum standard will be based upon rules associated with primary and foreign keys and integrity constraint rules.
- The assignment must make appropriate use of constraint clauses to ensure the referential integrity of the relations in the schema. The minimum standard will be based upon rules associated with primary and foreign keys and integrity constraint rules.
- The submission must make appropriate use of keys including the automatic generation of key values where appropriate.

> *Cite any source in APA format.*

SQL:

```

CREATE TABLE Specialty (
    Specialty_Number INT NOT NULL AUTO_INCREMENT,
    Specialty VARCHAR(45) NOT NULL,
    PRIMARY KEY (Specialty_Number)
);

CREATE TABLE Doctor (
    Doctor_ID INT NOT NULL AUTO_INCREMENT,
    Specialty_Number INT NOT NULL,
    Doctor_Name VARCHAR(155) NOT NULL,
    Doctor_Phone VARCHAR(20) NOT NULL,
    PRIMARY KEY (Doctor_ID),
    CONSTRAINT fk_doctor_specialty_number FOREIGN KEY (Specialty_Number)
    REFERENCES Specialty (Specialty_Number)
);

CREATE TABLE Patient (
    Patient_ID INT NOT NULL AUTO_INCREMENT,
    Doctor_ID INT NOT NULL,
    Patient_Name VARCHAR(155) NOT NULL,
    Patient_Phone VARCHAR(20) NOT NULL,
    Email VARCHAR(50) DEFAULT NULL,
    Address VARCHAR(50) DEFAULT NULL,
    Added_Date DATE NOT NULL DEFAULT (CURRENT_DATE),
    PRIMARY KEY (Patient_ID),
    CONSTRAINT fk_patient_doctor FOREIGN KEY (Doctor_ID)
```

```sql
    REFERENCES Doctor (Doctor_ID)
);

CREATE TABLE Allergy (
    Allergy_ID INT NOT NULL AUTO_INCREMENT,
    Allergy VARCHAR(155) NOT NULL,
    PRIMARY KEY (Allergy_ID)
);

CREATE TABLE Patient_Allergy (
    Patient_ID INT NOT NULL,
    Allergy_ID INT NOT NULL,
    PRIMARY KEY (Patient_ID, Allergy_ID),
    CONSTRAINT fk_patient_allergy_patient FOREIGN KEY (Patient_ID)
    REFERENCES Patient (Patient_ID),
    CONSTRAINT fk_patient_allergy_allergy FOREIGN KEY (Allergy_ID)
    REFERENCES Allergy (Allergy_ID)
);

CREATE TABLE Appointment (
    Appointment_ID INT NOT NULL AUTO_INCREMENT,
    Appointment_Date DATE NOT NULL,
--  A BP reading should be entered as: "132/88"; where the units are mmHg
    BP VARCHAR(6) NOT NULL,
--  Pulse should be entered as: 90; where the units are BPM
--  The value should not be negative, hence is unsigned
    Pulse SMALLINT UNSIGNED NOT NULL,
    Notes VARCHAR(255) DEFAULT NULL,
    PRIMARY KEY (Appointment_ID, Appointment_Date)
);

CREATE TABLE Patient_Appointment (
    Patient_ID INT NOT NULL,
    Appointment_ID INT NOT NULL,
    PRIMARY KEY (Patient_ID, Appointment_ID),
    CONSTRAINT fk_patient_appointment_patient FOREIGN KEY (Patient_ID)
    REFERENCES Patient (Patient_ID),
    CONSTRAINT fk_patient_appointment_appointment FOREIGN KEY (Appointment_ID)
    REFERENCES Appointment (Appointment_ID)
);

CREATE TABLE Medicine (
    Medicine_ID INT NOT NULL AUTO_INCREMENT,
    Medicine VARCHAR(155) NOT NULL,
    PRIMARY KEY (Medicine_ID)
);

CREATE TABLE Appointment_Medicine (
    Appointment_ID INT NOT NULL,
    Medicine_ID INT NOT NULL,
    PRIMARY KEY (Appointment_ID, Medicine_ID),
    CONSTRAINT fk_appointment_medicine_appointment FOREIGN KEY (Appointment_ID)
    REFERENCES Appointment (Appointment_ID),
    CONSTRAINT fk_appointment_medicine_medicine FOREIGN KEY (Medicine_ID)
    REFERENCES Medicine (Medicine_ID)
);
```

Of interest to note is that the `Doctor_ID` attribute has been left out of the `APPOINTMENT` relation because this relation has a connection to the bridging relation named `PATIENT_APPOINTMENT`. Since this bridging relation contains a `Patient_ID` attribute, it can be used to find the `DOCTOR` assigned to the `PATIENT` using the `Doctor_ID` attribute in the `PATIENT` relation. In essence, the SQL DDL statements above have re-applied 2NF normalization to remove the `Doctor_ID` attribute from the `APPOINTMENT` relation because it was only partially dependent on the primary key consisting of `Appointment_ID` and `Appointment_Date` (Elmasri & Navathe, 2016).

Reference

Elmasri, R., & Navathe, S. (2016). *Fundamentals of Database Systems*. Pearson.

Task 3

Using the attached Task 3, answer the following:

- Describe cardinalities between each relation.
- Describe connections between each relation.
- Explain why the `PatientMedicine` and `PatientAllergy` relations were created.

Use the attached Format, as an example on how to describe cardinalities and connections between relations.

> *Cite any source in APA format.*

Cardinalities and connections

- There is a cardinality ratio of `1:N` from `SPECIALTY` to `DOCTOR` because each `DOCTOR` has at most one `SPECIALTY` she is highly experienced at, whereas a `SPECIALTY` can be practiced by one or more doctors.
- There is a cardinality of `1:N` from supervising `DOCTOR` to `DOCTOR` because each `DOCTOR` in the role of supervisee has at most one direct supervisor, whereas a `DOCTOR` in the role of supervisor can supervise zero or more doctors.
- There is a cardinality ratio of `1:N` from `DOCTOR` to `PATIENT` because each `PATIENT` has at most one attending `DOCTOR`, whereas a `DOCTOR` can attend to zero or more patients.
- There is a cardinality ratio of `1:N` from `DOCTOR` to `APPOINTMENT` because each `APPOINTMENT` has at most one assigned `DOCTOR`, whereas a `DOCTOR` can be scheduled for zero or more appointments.
- There is a cardinality ratio of `M:N` from `PATIENT` to `ALLERGY` because a `PATIENT` can have many allergies or none, whereas an `ALLEGRY` can affect many patients or none.
- There is a cardinality ratio of `M:N` from `APPOINTMENT` to `MEDICINE` because during an `APPOINTMENT` a `DOCTOR` can prescribe many medicines or none, whereas a particular `MEDICINE` can be prescribed to patients over the course of many appointments or it may not be prescribed in any `APPOINTMENT`.

Bridge/composite/linking entities

The `PatientMedicine` and `PatientAllergy` relations were created to turn the `M:N` cardinalities into `1:N` . This is important because it would enable the enforcement of the `NON NULL` constraint on the keys of the composite relations (Rob & Coronel, 2009). Thus, if a `PATIENT` has no medicines prescribed to her, a corresponding row in the `PatientMedicine` relation will not be created. Yet, if the `PatientMedicine` relation did not exist, the `PATIENT` relation would need to have a `Medicine_ID` attribute where a `NULL` entry is inserted. The same applies to the `PatientAllergy` bridge relation.

Reference

Rob, P., & Coronel, C. (2009). *Database systems: Design, implementation, and management*. Thomson/Course Technology.