Rule 1: The Information Rule

A value i.e. the appointment date can be accessed using a unique identifier i.e. the primary key appointmentID.

```
select AppointmentDate from appointment where AppointmentID=1;
```

Rule 2: Guaranteed Access

A piece of data below is accessed using the table name (appointment), primary key (AppointmentID) and column name (AppointmentDate).

```
select AppointmentDate from appointment where AppointmentID = 1;
```

Rule 3: Systematic Treatment of Null Values

Null is treated as a distinct value see below query. When sum is performed on the LateCancellationFee attribute null is returned as it is not treated as a numerical value. LateCancellationFee in this case is null because it is not applicable for this appointment as the cancellation date was not considered late.

```
select sum(LateCancellationFee) from appointmentcancellations where PatientID =6;
```

To perform a calculation using a null value the below can be performed

Any null values for the LateCancellationFee from the appointmentcancellation table and TreatmentFee from the appointmentdetails table are treated as zero in the below query using ifnull(attribute, 0) where 0 is how to treat the null value. This allows a calculation of the total bill. Otherwise null would be returned if a null value was present.

```
select (ifnull((select sum(treatment.TreatmentFee) from appointmentdetails, treatment where appointmentdetails.TreatmentID=treatment.TreatmentID and appointmentdetails.AppointmentID=2),0) + ifnull((select appointmentcancellations.LateCancellationFee from appointmentcancellations where appointmentcancellations.AppointmentID=2),0)) as BillTotal;
```

Rule 4 Dynamic Online Catalog Based on the relational model

The metadata from the patient table can be obtained using the below queries. This can give information about the tables, columns and constraints.

select * from INFORMATION_SCHEMA.TABLES WHERE TABLE_NAME LIKE "PATIENT%";

select * from INFORMATION_SCHEMA.COLUMNS WHERE TABLE_NAME LIKE "PATIENT%";

select * from INFORMATION_SCHEMA.TABLE_CONSTRAINTS WHERE TABLE_NAME LIKE "PATIENT%";

The below query returns all the tables from the database.

select * from INFORMATION_SCHEMA.TABLES;

Rule 5 The Comprehensive Data Sub Language Rule

The following queries are considered DDL as they are used to create tables within the database.

```
CREATE TABLE `appointment` (
  `AppointmentID` int(11) NOT NULL,
  `AppointmentDate` date NOT NULL,
  `AppointmentTime` time NOT NULL,
  `Status` text NOT NULL,
  `PatientID` int(11) NOT NULL
);
CREATE TABLE `appointmentcancellations` (
  `AppointmentID` int(11) NOT NULL,
  `CancellationDate` date NOT NULL,
  `LateCancellationFee` float DEFAULT NULL
);
```

The following are examples of DML used to manage the database. Insert, update, and delete are examples of DML queries

The below query is used to add appointment cancellations to the appointmentcencellation table.

INSERT INTO `appointmentcancellations` (`AppointmentID`, `CancellationDate`, `LateCancellationFee`) VALUES

(2, '2022-01-17', 10),

(6, '2022-01-10', NULL);

This query is used to change an appointment date.

UPDATE *Appointments*
SET *appointmentdate=”2022-02-18”*
WHERE *appointmentID=1*;

This query is used to delete a patient.

```
Delete from Patient where patientID=1;
```

Rule 6 The View Updating Rule

A view must be updatable using insert, update or delete on the base table rows.

The following view dental report is created using the below query. This pulls the dental report and patient id from the patient table if patient id is not equal to null.

Create view Dental_Report AS select PatientID, DentalReport from patient where PatientID is not null;

The below query updates the dental report column in both the view dental_report and base patient table. Delete and insert queries can be run in a similar fashion.

update dental_report set DentalReport="Patient History redacted" where PatientID=1;

Rule 7: High Level Insert Update and Delete Rule

This rule means you should be able to modify more than one row in a table using a insert, update or delete query. For instance if the dentist decided to double all of their treatment fees the below query could be run. The update state is applied to all rows where the less than or equal statement is true for the TreatmentID primary key.

Update treatment set TreatmentFee=TreatmentFee*2 where TreatmentID<=7;

Rule 8 Physical Data Independence

Physical data independence allows the internal schema to be altered without changing the conceptual schema.

In other words any changes that occur at the physical level i.e. storage device or changing location of the database does not affect the logical or conceptual levels and the data remains accessible using sql queries.

To export a database from xampp you select export under phpmyadmin and pick the database to export. An sql file of the database is then generated. The database can then be imported in a different operating system e.g. Ubuntu. A databse is then created using Xampp and then selecting the import option. Then import the sql file already created. The database should then operate without any issues demonstrating rule 8.

Rule 9 Logical Data Independence

For the dental practice database any changes to the lower level schema i.e. internal schema or conceptual schema should not affect the higher level schema. This could be logical independence which is changes made to the conceptual schema and external view. So if a new table was added to the database we wouldn't have to change the external view of the database.

The database schema can also be changed without impacting already created queries for a database.

For instance the below query creates another table in the dental database. Any previous query should still function even with the new table created in the database. The external view of the table would also not need to be changed.

create table dummy (dm1 int, dm2 text);

Rule 10: Integrity Independence

For a database to abide by rule 10 then primary keys cannot be set to null. For instance the below insertion statement into the patient table results in the patientID being set to 7 as auto increment has been selected for the patientID column.

insert into patient values (null, "Tom Jones", "Parchment Square, Cork City", "1960-02-10", "Summary of patient history", 0857456620);

If auto increment is not selected the below error is generated as the primary key needs to be a unique value and cannot be set to null.

#1048 - Column 'PatientID' cannot be null

Rule 11 Distributed Independence

If the dental practice database was stored across several servers or on a cloud based platform it would appear to the user that it was all stored in the one location. For instance if you ran a select query it would return the same results regardless of where the underlying data was being stored.

Rule 12 Non-subversion Rule

A malicious application could access the data and change values for instance If constraints are not set up appropriately using a low level language. Requires users to use SQL and predefined tools