This project consists of the following items:

SQL queries to create tables and insert the test data into these tables.

Exported file to be imported into XAMPP – this create tables and inserts test data (data is at the stage where all transactions after Codd's rules demonstration and after showing main transactions).

ERD – Entity relation diagram with all the entities, attributes, relationships, primary and foreign keys, as well as brief explanation of the structure.

Codd's Rules – demonstration of the rules using SQL and brief explanation of each.

To create the database, the following steps were required:

1. Prepare EDR diagram based on *Narrative* document provided. Preparation took into consideration primary and foreign keys and few alterations were needed to achieve the final result.
2. Create tables. It was decided to create 6 tables: specialist, patients, appointments, treatment, payment, and bills.
3. Creating test data. Written and executed insert statements to all 6 tables.

In order to use this database, it is required to import the file into XAMPP. After the import insert, update, delete and other SQL queries can be executed. Examples of these transactions, based on *Narrative* document provided:

Dr. Mulcahy has the list of patients, but if she cannot provide some treatment, she is referring the patient to another specialist. The data in the database is changed using following command: *update patients set pat_spec_id = 2 where pat_id = 5;* This patient now appears as referred to Dr. Michael Murphy in Cork.

When patients ring to book an appointment i.e. for the 5th of January at midday. Helen checks if there any slots for the requested time by running *select * from appointments where app_start_time = '2021-01-05 12:00:00';* Unfortunately, this slot is not available, and Helen checks if the following slot is available by running *select * from appointments where app_start_time = '2021-01-05 13:00:00';* A patient confirms that this time is convenient and asks to have it confirmed. Helen then checks if the patient exists in the patients' list by running i.e. *select * from patients where pat_firstname = 'Kayla' and pat_lastname = 'Patrick';* In this case, a patient exists, therefore Helen only records appointment details by running *insert into appointments values (25, '2021-01-05 13:00', '2021-01-05 14:00', NULL, NULL, 9);*

In case the patient does not exist, Helen needs to add new patient information and then record the appointment. New patient is added by *insert into patients values (25, 'Linda', 'Harrisson', 'Liberty Avenue, Harrlow town, Co.Cork', 1);* New appointment for new patient is added by *insert into appointments values (25, '2021-01-06 13:00', '2021-01-06 14:00', NULL, NULL, 25);*

If the patient informs about appointment cancelation on the day of the appointment, €10 cancelation fee applies. For example, Linda Harrison cancels her appointment the same day it was supposed to take place. Helen adds this information by running *update appointments set app_cancel_time = '2021-01-06 13:00:00', app_cancel_fee = 10 where app_pat_id = (select pat_id from patients where pat_firstname = 'Linda'and pat_lastname = 'Harrisson');*

When patient wants to change the time of the appointment Helen checks for available time and runs the following: *update appointments set app_start_time = '2021-01-25 10:00:00', app_end_time = '2021-01-25 11:00:00' where app_pat_id = (select pat_id from patients where pat_firstname = 'Kayla' and pat_lastname = 'Patrick');*

On Tuesday 5th of January, Helen checks next week's appointments and sends the reminders by post. To get the addresses with patient names and appointment details making sure not to include cancelled appointments, Helen runs the following *select pat_firstname, pat_lastname, pat_address, app_start_time, app_end_time from patients, appointments where pat_id = app_pat_id and app_start_time between '2021-01-18 01:00:00' and '2021-01-24 23:00:00' and app_cancel_fee is null;*

Then Helen sorts all the unpaid bills and sends them by post. To get the name and address of the patient as well as the amount due and for which treatment, Helen runs the following: *select pat_firstname, pat_lastname, pat_address,bill_id, bill_due_amount, treat_desc from patients inner join bills on pat_id = bill_pat_id inner join payments on bill_pay_id = pay_id inner join treatment on pay_treat_id = treat_id where bill_due_amount !=0;*

One of the patients who received €80.00 bill is Valentin Contreras for Scaling sandblasting. He rings the clinic and makes the payment by card. Helen takes the payment and records it by running:

insert into payments values (24, 80, 'Credit Card', 5, (select pat_id from patients where pat_firstname = 'Valentin' and pat_lastname = 'Contreras'));

update bills set bill_due_amount = 0 where bill_pat_id = (select pat_id from patients where pat_firstname = 'Valentin' and pat_lastname = 'Contreras');

Next time when Helen checks for patients with unpaid bills, Valentin would not be in the list.

If there are any changes to the price list, it can be updated by changing the price of the treatment. For example, the price for Composite filling has increased for €10. Helen can update it running *update treatment set treat_price = treat_price +10 where treat_desc = 'Composite filling'; The* treatment price has increased from € 200 by €10 and is now €210.