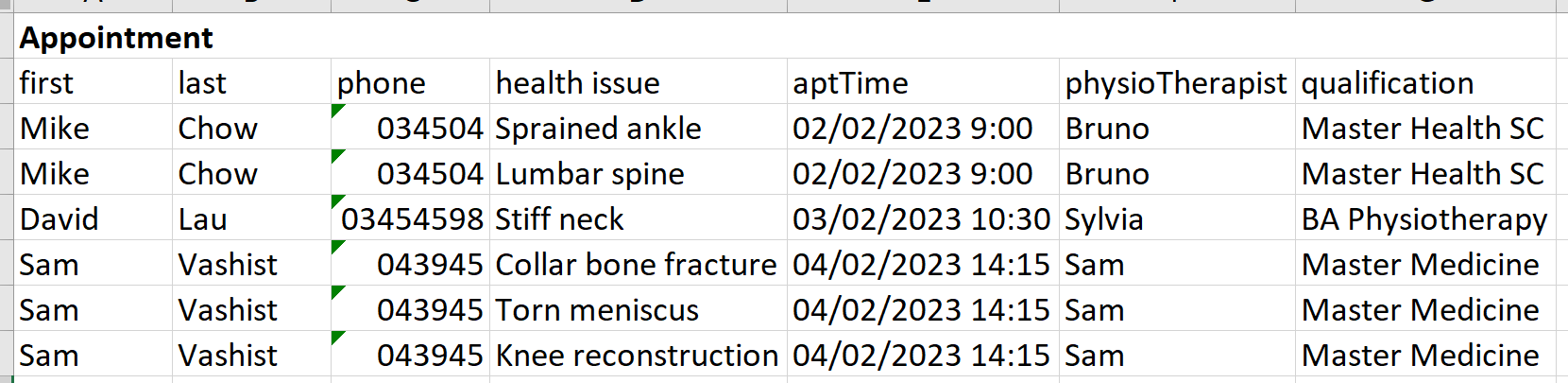
**Tutorial 5 Assessment**

**Question 1**

While we were discussing database design options with EasyLife, we found out that they have been keeping appointments in an Excel sheet until now. They would like us to record those appointments in the database. This prompts us to adjust our database to suit their existing records. You have to normalise the data sheet they have into 3NF.

**first** and **last** are the first and last names of the client, and **phone** is the client's phone number. **health issues** are the health issues a person has, not necessarily the issues they need the appointment for. **aptTime** is the date and time of the appointment. **physioTherapist** is the PT who is attending to them. qualification is the highest qualifications of the PT. While there can be multiple health issues, the **qualifications** attribute only records the highest qualification of the PT.

In this first question, upload a solution that shows the table in **1NF** without losing any information.



To convert the given table into the First Normal Form (1NF), we need to ensure that each column contains atomic (indivisible) values, and there are no repeating groups. In your current table, the "health issues" column seems to have multiple values separated by numbers. We will need to normalize it by creating separate rows for each health issue associated with a client.

In this normalized table, each row represents a single health issue associated with a client's appointment. The client's information is duplicated for each issue they have. This format adheres to the First Normal Form principles without losing any information.

**Question 2**

Continuing from your answer to Question 1, bring the solution into 2NF. You may want to add surrogate keys to account for the fact that people's names aren't truly unique, which will make the modelling easier.

What you need is a primary key for the Appointment relation, and to resolve a partial dependency by creating the appropriate tables. Draw a UML diagram with attributes, primary and foreign keys as well as relationships with cardinalities.

A graph with a line and text

Description automatically generated

In this revised structure, the Appointments table now includes a primary key ("apt\_id"), which resolves the partial dependency issue. The client and health issue information remains normalized in separate tables. The physiotherapist data is also separated into a distinct table, avoiding data redundancy. This design adheres to Second Normal Form while properly addressing the partial dependency.

Relationships:

* Clients to Appointments: One-to-Many (A client can have multiple appointments.)
* Health Issues to Appointments: One-to-Many (An issue can be associated with multiple appointments.)
* Physiotherapists to Appointments: One-to-Many (A physiotherapist can have multiple appointments.)
* Clients to Appointments and Health Issues: Many-to-Many (A client can have multiple health issues across multiple appointments.)
* Physiotherapists to Appointments: Many-to-Many (A physiotherapist can attend to multiple appointments.)

Cardinalities:

* Clients to Appointments: 1 to Many
* Health Issues to Appointments: 1 to Many
* Physiotherapists to Appointments: 1 to Many

**Question 3**

Continuing from your answer to Question 2, bring the solution into 3NF. Draw a UML diagram of the all tables you have created with attributes, primary and foreign keys as well as relationships with cardinalities.

