Task 1 - Francis Denton

Task 1A). Normalising the Data.

Split of data on the data set we were provided.

Patients Table Data example:

Patient Name	Date of Birth	Sex	Address	City	Postcode	Telephone
ABD111	22/12/1949	M	444 New Avenue	Bristol	BS99 7TR	0117 3333444
ABC234	11/12/2002	M	134 OldStreet Road	Bristol	BS78 9YT	0117 7687999

Date of Admission	Height (meters)	Weight (Kg)	BMI	Body Type	Smoker	Asthmatic	NJT / NGR	Hypertension	Renal RT	Ileostomy / Colostomy	Parenteral Nutrition	Referral Decision
21/11/2020	1.75	80	26.12	Slim	Y	N	Y	N	N	Y	N	N
22/11/2020	1.92	85	23.06	Athletic	N	Y	N	Y	N	Y	N	Y

The dataset we are provided with is not yet in 1st normal form as there is no unique identifier per row of the data, a patient ID is needed to solve this problem as it is possible multiple patients could have the same name.

Looking at the dataset we have been provided with, the dataset is too large to be put into a single table and needs to be broken down into a more appropriate format. As there is the possibility for a single patient to have multiple admissions, it is required that we reduce the amount of redundant and reoccurring data that is not likely to change whilst still making the tables easy to query and read to the human eye. Thus, a split is required, per admission to the clinic, it is unlikely that the patients address details will change, and their date of birth will never change. I have decided to split the data into two tables, one for admissions and one for patients.

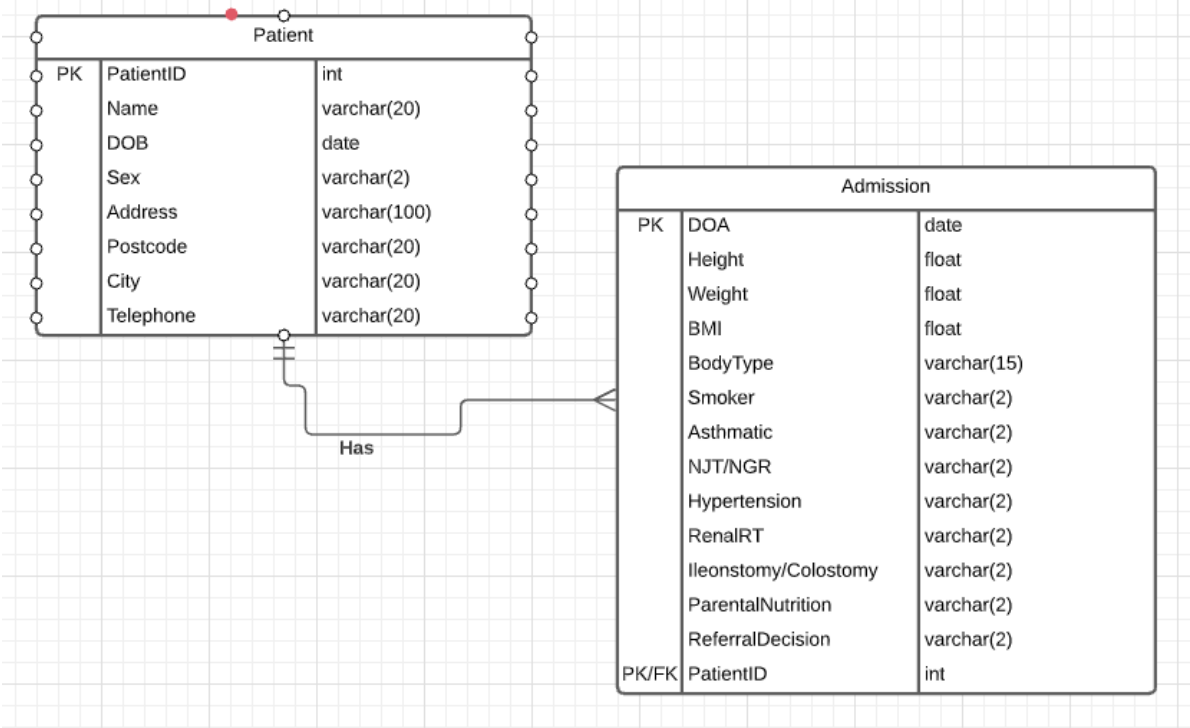
The patient table will hold the non-medical information about the patient e.g. name, sex, address, date of birth, whilst the admissions table will hold the admission and medical information which can change per admission to the clinic. Though height is unlikely to change, it wouldn't make sense to separate it from the weight and BMI as they share a relationship together. As the data we are working with is medical and highly sensitive, it is wise to use a patient ID in the admissions table instead of a patient name in order to combat against the chance that there is unauthorized access to the system, making it harder for an intruder to match a patient and their medical information, whilst not inhibiting someone who knows how to work the system.

A compromise is required in some regards in the way in which you break down the data, a rule of third normal form is that no non-key fields are dependant on another non-key field. In this example BMI is dependant on the weight and height of the patient, and the city is dependant on a postcode, neither of which are key fields. If we were to follow third normal form all the way through, it would result in having a table with postcode being a primary key and the corresponding city being the only field in that table. This is not practical and a reason why it can be common to see relational databases in 2nd normal form. Whilst limiting the amount of repeating data is important, it can also cause a database to quickly become hard to navigate and overly complex. I was contemplating adding a third table, which would be an address table on the off chance that multiple patients lived at the same address it would reduce the amount of repeated data, this came at the cost of increasing the complexity of the database without much gain and was decided against for ease of use purposes.

Task 1B). ER Diagram.

```
The Entities and attributes in our data set provided are:
Entity: Patient
    Attributes: ID
                Name
                Date of Birth
```

```
                Sex
                Address
                Telephone
                City
                Postcode
Entity: Admission
    Attributes: Date
                Height of Patient
                Weight of Patient
                Patient BMI
                Medical conditions (Smoker - Parental Neutrition)
                Referral Decision
                PatientID
```



The ER diagram has been designed using Crows Foot Notation. The cardinality of the relationship between patients and admissions in the data set we are provided is one-to-one-to-many. But a case could be made for the cardinality to be one-to-zero-to-many. This is because it is highly possible a new patient will be added to the database before any appointments will have been made for them in a future scenario. Though the data set we are provided has no examples of this so I have not incorporated it into the design. The primary key of the patients table is an auto-incrementing ID field.

The Primary key of the Admissions table is a composite primary key consisting of the date of the Appointment and the ID of the patient which is referenced as the foreign key, there is no need to create a field for something like Appointment ID In this scenario. If I were to have a third address table in the database as referenced in the normalisation section, it would have had its own entity and a cardinality of one-to-one-to-many with the patients as multiple patients live at the same address.

Implementation of the database

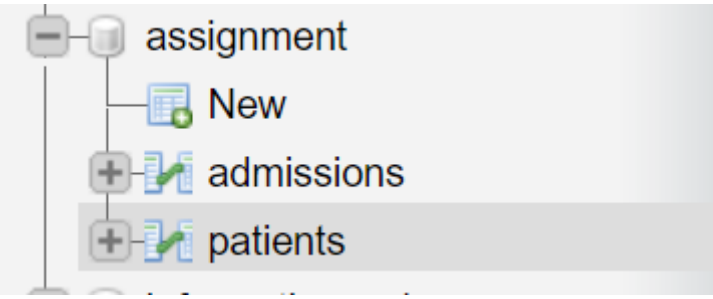
I was using the phpmyadmin platform in order to implement and query my relational database, the phpmyadmin platform was accessed through the xampp contol panel which starts a server for you. I created the structure of my tables through an SQL statement and then ran the statement on the platform to create tables.

Tables

```
CREATE TABLE admissions (
          DOA date,
          Height float,
          Weight int,
          BMI float,
          BodyType varchar(20),
          Smoker varchar(2),
          Asthmatic varchar(2),
          NJT_NGR varchar(2),
          Hypertension varchar(2),
          RenalRT varchar(2),
          Ileostomy_Colostomy varchar(2),
          ParentalNutrition varchar(2),
          ReferralDecision varchar(2),
          PatientID varchar(20) references patients(PatientID),
          PRIMARY KEY (PatientID, DOA)
     ) ;

CREATE TABLE patients (
          PatientID int not null auto_increment primary key,
          Name varchar(20),
          DOB date,
          Sex varchar (8),
          Address varchar(20),
          City varchar(20),
          Postcode varchar(20),
          Telephone varchar(20)
);
```

Database After Creation



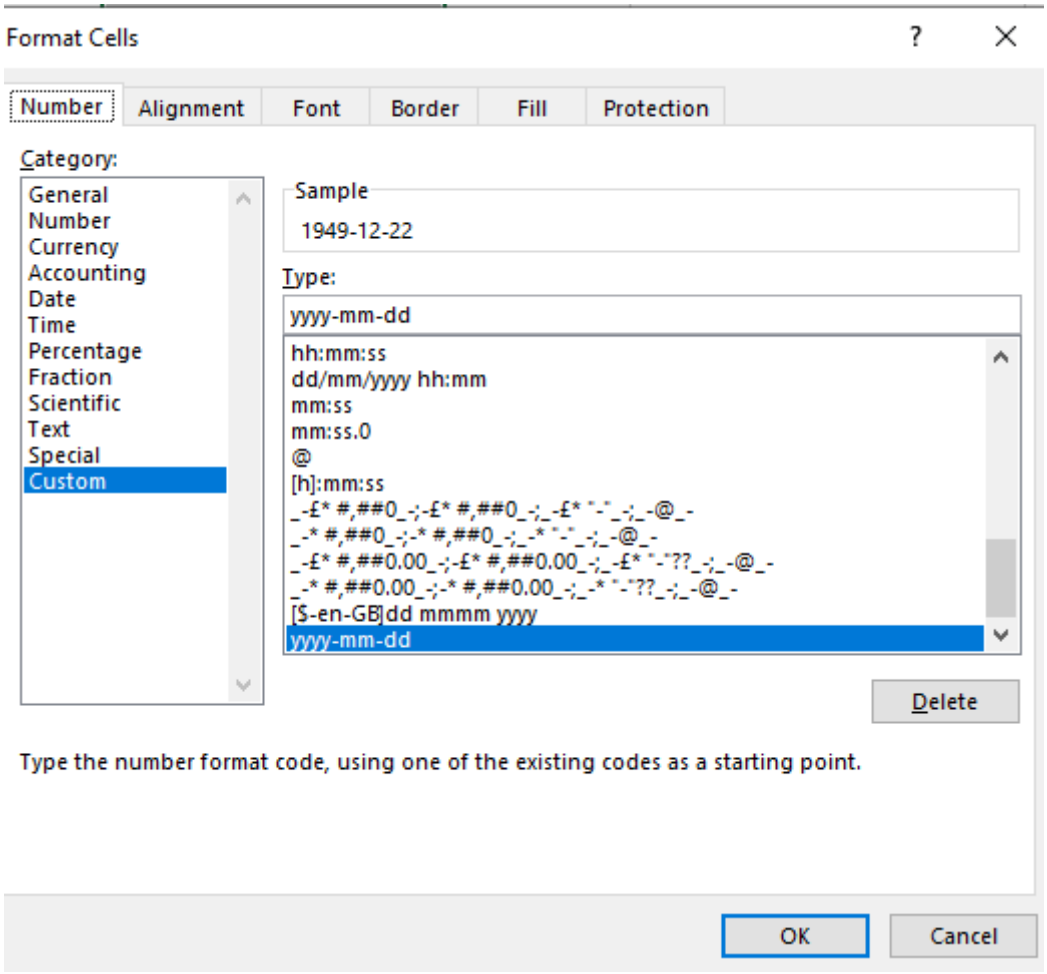
Populating the database:

A SQL statement can be used to insert data into the database, an example of such statement is as follows:

```
Patients Table:
INSERT INTO patients (Name, DOB, Sex, Address, City, Postcode, Telephone)
    VALUES ('ABD111', '1949-12-22', 'M','444 New Avenue', 'Bristol', 'BS99 7TR',
'01173333444');

Admissions Table:
INSERT INTO admission (PatientID, DOA , Height, Weight, BMI, BodyType, Smoker,
Asthmatic,NJT/NGR ,Hypertension ,RenalRT ,Ileostomy/Colostomy ,ParentalNutrition
, ReferralDecision, PatientID)
    VALUES ('2020-11-21','1.75',    '80',   '26.12',    'Slim', 'Y',    'N',
'Y',    'N',    'N',    'Y',    'N',    'N', '1')
);
```

There is also the ability to import data from a csv file into the database, this required some reformatting of the data. For example, since the date format we were provided with is not in line with SQL date format, I needed to edit the excel data using the inbuilt functionality of excel which allows for the conversion of date formats. I changed the dates form DD/MM/YYYY Format to YYYY-MM-DD.



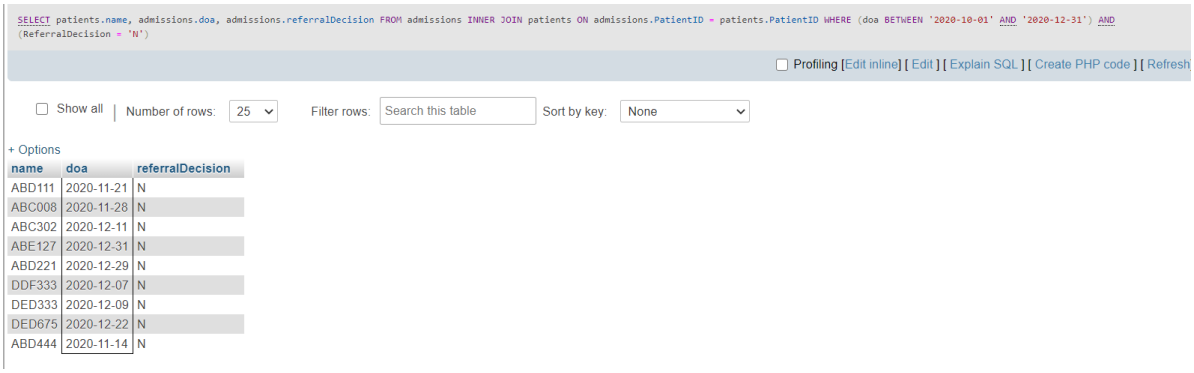
Queries

Query 1

List all the Patients that were admitted between the months of October to December 2020 (inclusive) and were not referred to a dietitian

```
SELECT patients.name, admissions.doa, admissions.referralDescision
FROM admissions
INNER JOIN patients
ON admissions.PatientID = patients.PatientID
WHERE (doa BETWEEN '2020-10-01' AND '2020-12-31')
  AND (ReferralDecision = 'N');
```

Output



SELECT patients.name, admissions.doa, admissions.referralDecision FROM admissions INNER JOIN patients ON admissions.PatientID = patients.PatientID WHERE (doa BETWEEN '2020-10-01' AND '2020-12-31') AND (ReferralDecision = 'N')

Profiling [Edit inline] [Edit] [Explain SQL] [Create PHP code] [Refresh]

Show all | Number of rows: 25 Filter rows: Search this table Sort by key: None

+ Options

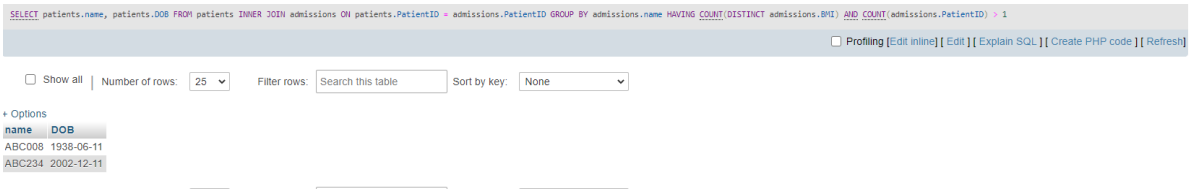
name	doa	referralDecision
ABD111	2020-11-21	N
ABC008	2020-11-28	N
ABC302	2020-12-11	N
ABE127	2020-12-31	N
ABD221	2020-12-29	N
DDF333	2020-12-07	N
DED333	2020-12-09	N
DED675	2020-12-22	N
ABD444	2020-11-14	N

Query 2

List all the patients that have had at least two admissions to the CCU and between admissions have had a change in their BMI.

```
SELECT patients.name, patientsDOB
FROM patients
INNER JOIN admissions
ON patients.PatientID = admissions.PatientID
GROUP BY admissions.PatientID
HAVING COUNT(DISTINCT admissions.BMI) AND COUNT(admissions.PatientID) > 1
```

Output



SELECT patients.name, patients.DOB FROM patients INNER JOIN admissions ON patients.PatientID = admissions.PatientID GROUP BY admissions.name HAVING COUNT(DISTINCT admissions.BMI) AND COUNT(admissions.PatientID) > 1

Profiling [Edit inline] [Edit] [Explain SQL] [Create PHP code] [Refresh]

Show all | Number of rows: 25 Filter rows: Search this table Sort by key: None

+ Options

name	DOB
ABC008	1938-06-11
ABC234	2002-12-11

Query 3

List the patients that have been admitted in the CCU on 28/02/2021 and were referred to a dietitian in the order of priority of referral

```
SELECT patients.name, admissions.doa, admissions.ReferralDecision
FROM patients
INNER JOIN admissions
ON patients.PatientID = admissions.PatientID
WHERE (doa = '2021-02-28' AND ReferralDecision = 'Y')

'Couldnt work out how to implement the custom priority!'
'Half of the Query is met, list the patients admissted on 28/01/2021 and were reffered to by a dietitan'
```

SELECT patients.Name, admissions.doa, admissions.ReferralDecision FROM admissions INNER JOIN patients ON admissions.PatientID = patients.PatientID WHERE (admissions.doa = '2021-02-28' AND admissions.ReferralDecision = 'Y')

Profiling [Edit inline] [Edit] [Explain SQL] [Create PHP code] [Refresh]

Show all | Number of rows: 25 Filter rows: Search this table Sort by key: None

+ Options

Name	doa	ReferralDecision
ABC101	2021-02-28	Y
ABC201	2021-02-28	Y
AAA322	2021-02-28	Y

Query 4

List all the patients that should have been referred to a dietitian, but were missed out by the system

```
    SELECT *
    FROM admissions
    WHERE (BodyType = 'Slim' AND BMI >= 28
           OR BodyType = 'Regular' AND BMI >= 29
           OR BodyType = 'Athletic' AND BMI >= 30
           OR BMI < 18.5
           OR ((Smoker = 'Y'
               OR Asthmatic = 'Y')
               OR NJT_NGR = 'Y'
               OR Hypertension = 'Y'
               OR RenalRT = 'Y'
               OR Ileostomy_Colostomy = 'Y'
               OR ParentalNutrition = 'Y'))
      AND ReferralDecision ='N'
GROUP BY admissions.PatientID
```

Output

DOA	Name	Height	Weight	BMI	BodyType	Smoker	Asthmatic	NJTorNGR	Hypertension	RenalRT	IleostomyorColostomy	ParentalNutrition	ReferralDecision	Patient_ID
2020-11-21	ABD111	1.75	80	26.12	Slim	Y	N	Y	N	N	Y	N	N	1
2020-11-28	ABC008	1.59	50	19.78	Regular	N	N	Y	N	Y	N	N	N	10
2021-02-28	ABC008	1.59	41	16.22	Regular	N	N	Y	N	Y	N	N	N	10
2020-12-11	ABC302	1.76	65	20.98	Regular	N	N	N	Y	N	N	Y	N	13
2021-01-13	ABC349	1.93	81	21.75	Athletic	Y	N	N	N	Y	N	N	N	15
2021-02-14	AAA102	1.72	67	22.64	Slim	Y	Y	N	N	N	N	Y	N	17
2020-09-09	ABE329	1.88	90	25.46	Athletic	Y	N	Y	N	N	Y	N	N	24
2020-12-31	ABE127	1.71	101	34.54	Regular	Y	Y	N	N	Y	N	N	N	25
2021-01-01	BDF777	1.69	42	14.7	Slim	N	N	N	N	N	N	N	N	28
2020-12-29	ABD221	1.67	61	21.88	Regular	Y	N	Y	N	N	N	Y	N	3
2020-12-07	DDF333	1.67	55	19.72	Regular	Y	N	N	N	Y	N	N	N	33
2021-02-28	DDD222	1.74	55	18.06	Regular	N	N	N	N	N	N	N	N	37
2020-12-22	DED675	1.74	87	28.74	Regular	Y	N	Y	N	Y	N	N	N	45
2021-02-06	DED879	1.75	88	28.73	Athletic	N	Y	N	N	N	N	N	N	46
2021-01-29	DEF555	1.61	67	25.85	Regular	Y	N	N	N	Y	N	N	N	48
2021-02-18	DEF666	1.63	47	17.69	Slim	N	N	N	N	N	N	N	N	49
2020-11-14	ABD444	1.64	56	20.82	Slim	N	Y	Y	N	N	N	N	N	7

Query 5

List all the patients that have slim build, normal weight condition, and were referred to a dietitian, in Descending order according to their age.

SQL Statement

```
SELECT patients.name,
       patients.dob,
       admissions.BodyType,
       admissions.BMI,
       admissions.ReferralDecision
FROM admissions
INNER JOIN patients ON admissions.PatientID = patients.PatientID
WHERE (admissions.BodyType = 'Slim'
       AND admissions.BMI BETWEEN 18.5 AND 25
       AND admissions.ReferralDecision = 'Y')
ORDER BY patients.dob ASC
```

Output

```
SELECT patients.name, patients.dob, admissions.BodyType, admissions.BMI, admissions.ReferralDecision FROM admissions INNER JOIN patients ON admissions.PatientID = patients.PatientID WHERE (admissions.BodyType =
'Slim' AND admissions.BMI BETWEEN 18.5 AND 25 AND admissions.ReferralDecision = 'Y') ORDER BY patients.dob ASC
```

Profiling [Edit inline] [Edit] [Explain SQL] [Create PHP code] [Refre

Show all | Number of rows: 25 Filter rows: Search this table Sort by key: None

+ Options

name	dob ▲ 1	BodyType	BMI	ReferralDecision
ABC101	1935-05-05	Slim	18.55	Y
ABD321	1963-03-31	Slim	20.43	Y
DDD554	1979-03-22	Slim	18.59	Y
DEF777	2003-03-14	Slim	18.51	Y