Table of contents:

[Task: Part1 – Systems Analysis (& Design) [30%] 2](#_Toc87029608)

[1.1. Executive Summary (1/2 page) [5%] 2](#_Toc87029609)

[1.2. System aims and requirements 2](#_Toc87029610)

[1.3. Use cases 6](#_Toc87029611)

[Task: Part 2 – Database Design and Implementation [45%] 15](#_Toc87029612)

[2.1. Top Down ERD [10%]: 15](#_Toc87029615)

[2.1.b. Entity derivation 15](#_Toc87029616)

[2.2. Bottom Up ERD [5%]: 17](#_Toc87029620)

[2.3. Merged Top Down and Bottom up ERD [3%] 19](#_Toc87029621)

[2.4. CREATE TABLE statements in Oracle including constraints: 19](#_Toc87029622)

[2.5. INSERT all data including FIVE(5) extra tuples into each table 24](#_Toc87029623)

[2.6. Retrieve all data from tables using a select query 27](#_Toc87029624)

[2.7. Critical design analysis and improvements 36](#_Toc87029625)

[Task: Part 3 – Query Implementation [25%] 37](#_Toc87029626)

[3.1. Write an SQL statement that lists the pet id, pet name, pet age showing the columns as “ID”, “Name” and “Age”. Their names should be those starting with letters A to M. Sort the results by the pet id in descending order. [4%] 37](#_Toc87029630)

[3.2. Write an SQL statement that shows the most overworked part-time vet doctor, i.e., any vet with 3 or more appointments for pet clinics, listing details such as id, name and office number. [4%] 38](#_Toc87029634)

[3.3. Write an SQL query that lists pet details with 2 or more appointments between 2nd Jan 2021 and 26th Sep 2021. The columns should include pet id, name and count of the number of appointments. [4%] 39](#_Toc87029635)

[3.4. Write an SQL query that finds pet id, name and cost of appointments, such that the cost of the appointment is less than the average appointment cost of all pet appointments. [4%] 40](#_Toc87029636)

[3.5. Write an SQL statement that lists a doctor id, status, the total number of appointments he/she has handled, the pet involved (id, name) and the dates these took place. The query should only show cases whose diagnoses involved “socialisation” or “dental” work and where the appointment date was at least 4 weeks ago. [5%] 41](#_Toc87029637)

[3.6. Write an SQL query that finds pet ids with no appointments or diagnoses so far. [4%] 43](#_Toc87029638)

[APPENDICIES 44](#_Toc87029639)

[Appendix 1. Assumptions Made 44](#_Toc87029640)

[Appendix 2. User Stories 46](#_Toc87029641)

[Appendix 3. Comprehensive UCD 48](#_Toc87029642)

[Appendix 4. Initial top down, merged and signed off ERD 49](#_Toc87029643)

[Appendix 5. CREATE and DROP TABLE SQL statements: 50](#_Toc87029644)

[Appendix 6. ERD including suggested improvements 56](#_Toc87029645)

Preface & context note

The general approach to Task 1 was to consider the deliverable system in its entirety. As the project brief states “You are members of a Software Design company, and your role is to help a dog clinic called Noah’s Pet Clinic (NPC) improve their paper-based pet booking system” it was considered as the project as a whole i.e., the entire MVC model. This includes the front end that the people at the vets would interact with, that we don’t actually design, and the back-end database which would store the data, that we design and create later in Task 2.

# Task: Part 1 – Systems Analysis (& Design) [30%]

All assumptions made for the entire project have been listed in the appendices at the end of this task in Appendix 1 – [click here](#_Assumptions_Made)

## Executive Summary (1/2 page) [5%]

#### **Introduction:**

Noah’s Pet Clinic (NPC) is a small-scale veterinarian surgery based in rural Manchester with a particular focus in canine care. The current organisational model of the surgery involves the usage of a pen and paper model, where appointment records are documented by hand in the “appointment diary” and “pet registration” forms are completed by pet owners, which are then filed alphabetically by the sole receptionist of the company. Furthermore, “consultation cards” are completed by doctors following an appointment and are filed alongside the owner records in the filing cabinet. Since everything is written by hand and stored in files there is always a looming worry that some of these files could go missing jeopardizing the customer clinic relationship.

NPC staff have expressed dissatisfaction with this model. They report that accessing the appointment diary is difficult and inefficient with respect to retrieving appointment information. Additionally, the receptionist has expressed a desire for a more simplistic system with respect to filing pet registration forms, as they often go a miss when they are taken by medical staff.

This leads to the problem statement of the present report: The current model of appointment, pet registration and consultation documentation at NPC is inefficient, causes confusion amongst the staff and reduces the productivity of the business.

#### **Summary of Purpose:**

Within the present report, an electronic based system is purported to improve the efficiency of the business as well as streamline the filing process for the staff at NPC. This was accomplished through the Design and Analysis of a relational database model. The Design of the Database was implemented through rigorous usage of the Oracle and APEX platforms to test the efficacy of our relational design and finally, once the relational database was complete, it was tested once more using SQL Query Implementation.

#### **Key points – how the problem was solved:**

* Before implementing any data to our electronic model; our goals, functional requirements and non-functional requirements were outlined and adhered to throughout.
  + Use Case Diagrams (UCD) were produced to illuminate the typical daily uses of the model.
  + Use Case Specifications were produced to provide further insight into the details of a UCD.
* Entity Relationship Diagrams were produced through a combined top-down and bottom-up fashion to highlight how the various elements of the system will relate to one another.
  + This led to the production of database tables using SQL to relate one entity to another, including appropriate primary keys, foreign keys, and their respective attributes.
* Finally, once the database and its’ respective tables were produced, SQL queries were implemented to judge whether the system was functional success.
  + Our database successfully returned tables pertaining to every SQL query.

## System aims and requirements

### Main goals: [5%]

Noah’s Pet Clinic is current using a paper-based system for appointments and management of client data. This has led to several complaints from the doctors, nurses, and the vets that the appointment diary gets lost and misplaced, pet records and consultation notes are incorrectly filed or passed on late. The clinic now wishes to move away from this to an electronic system to mitigate these issues.

The main goals of this system will be to:

* Upgrade the paper-based appointment and registration system to a more efficient electronic based system.
  + Real time data input and storage will provide a better picture of what is happening within the surgery on a given day.
* Streamline the functionality and accessibility of the system for vet doctors, nurses, and reception staff
  + Allowing for multiple members of staff across different functions and uses of the system to access the same data at the same time.
* Manage the data NPC collects and stores regarding:
  + Staff
  + Customers and their pets
  + Appointments and diagnosis
  + Medication options including costs and available pharmacies in the area
  + Fees

#### **User Definition:**

Users within the system are identified and fall into the following user categories:

|  |  |  |
| --- | --- | --- |
| Primary | Secondary | Tertiary |
| * Nurse * Doctor * Receptionist | * Managing Director * System Admin | * Customers * Cleaners * Maintenance staff |

The system admin isn’t explicitly mentioned within the brief, however, as mentioned in the assumptions in the appendix, there is no indication as to who will be administering the system or the level of IT proficiency of the current staff members. Therefore, it’s necessary to express system admin as a separate user. This does not mean a specific doctor, or the managing director can’t also be the system admin, but the role is split out in order to provide clarity in the event that they are different people.

#### **User stories**

User stories have been collected from select staff members at Noah’s Pet Clinic in order to understand what they want from the new system and their reasonings behind them – these are found in Appendix 2 – [click here](#_User_Stories)

Taking into consideration these Users’ Stories, and the information from the brief provided the following system requirements have been derived:

### Function and non-function requirements of the system[5%]

### Functional requirements

|  |  |
| --- | --- |
| **System level requirements** | |
| **The system MUST** be able to: | * Authenticate a user when they log in with their email and password * Have remote log-in function * Store and query the following information:   + User     - Doctor     - Nurse     - MD     - System Admin     - Receptionist   + Customer   + Pet   + Appointment, including fees and diagnosis   + Medicine and Pharmacy * Export pet records |
| **User specific requirements** | |
| **ALL users MUST** be able to: | * Log into the system with a user account |
| **The Receptionist must** be able to: | * Input and edit pet and customer information * Search for pet and customer records * Input, edit, cancel appointments * Search and view all appointments past, present and future * View outstanding cancellation fees associated with each appointment and mark received payments as paid |
| **Doctors MUST** be able to: | * View all appointments assigned to them * Input & edit consultation notes * Advise cost and availability of prescribed medication * Make referrals and deferrals |
| **Nurses MUST** be able to: | * View scheduled and previous appointments * Search pet records and appointments |
| **The System Admin MUST** be able to: | * Create and edit user accounts * Create user roles and assign them to user accounts * Input and edit into all fields except passwords |

### Non-functional requirements

|  |  |
| --- | --- |
| **The system MUST** be able to accommodate the following: | |
| **Data constraints** | * ID structure used already:   + Pets:     - Sequential between 1000 – 3000     - Maximum of 2000 pet records.   + Doctors:     - Sequential starting from 2200   + Consultation:     - Sequential starting from 105000 * Any other user IDs not specified should must be sequential from their first entry which can be determined by the system admin. * Unique email addresses must be used for each user. * Pets   + Sex must be entered as “M” or “F” only     - Upper case characters only, no lower case. No male, or female or variations of   + Age can only be entered as 1 – 12     - They do not deal with older or younger pets. |
| **All users MUST**: | |
| **Training and equipment** | * Have adequate training on how to use the system prior to using it   + Via the software development team * Have access to adequate hardware in which the system can be accessed   + This means all offices and consultation rooms |
| **The system SHOULD** be also meet the following: | |
| **Performance** | * Run on the following operating systems   + Windows 10   + MacOS from versions 10.15 onwards * Respond to search queries within a 2 second period. * Produce custom reports within a 30 second period. * Take no longer than 30 seconds to download custom reports.   + Custom reports should be downloadable in .csv format * Take no longer than 2 seconds to confirm entry of data for day-to-day functions. * Take no longer than 2 minutes to confirm entry of data for bulk upload functions. * Take no longer than 2 seconds to confirm or deny log in authentication. * Take no longer than 10 seconds to download pet records. * Cloud data servers should be located as close to Manchester, UK as possible to reduce latency periods * Allow for a current maximum of 18 concurrent users without performance deterioration. * Display an inactivity warning to users who have been inactive for more than 30 minutes and log them out after 2 minutes if left inactive. |
| **Security** | * Passwords:   + Generate a new, randomised password on user creation and email it to that user.   + Request a user set a new password on first login.   + Require confirmation of new password with a second entry.   + Send a recovery email to the user when they select “forgot my password” on login.   + Send a confirmation email to the user when they reset their password.   + Request the user reset their password at 3-month intervals.   + Never allow passwords to be viewable by anyone at the point of entry or at any other time. * Be protected against outside threats e.g., viruses and “hacks”   + shut down in the case that these protections are breached. |
| **Capacity and Scalability** | * Have adequate storage space for the system launch:   + Storage space should be determined by inputting all existing data, using business sales growth data over the last 3 years to estimate how fast the rate of data space requirement will increase, and then provide a minimum space for 10 years of projected growth. * Store data efficiently and be able to estimate a timeframe in which all the storage space will be filled based on rate of data input. * Initial business requirement is to accommodate for a concurrent capacity of 18 users logged in at the same time with a yet to be determined minimum number of system transactions per week.   + Should be scalable in case of business growth |
| **Accessibility** | * Be accessible to users with disabilities in accordance with the Equalities Act 2010. * Allow for remote login depending on the users role as defined by the system admin.   + Allow only for terminal access for anyone else * Access to data and features within the system will depend on the users role assignment as defined by the System Admin and limit the user to only be able to access the data needed for their role.   + Initially these roles will be Doctor, Nurse, Receptionist, System Admin, Managing Director   + More to add if/when required * System access to data must be allowed within the terms and service of the third-party cloud data storage provide. |
| **Availability** | * Be available at the very minimum between the hours of 8am and 7pm GMT/BST seven days a week   + Maintenance and planned outages to happen outside of these opening times   + Cloud data servers to be located in GMT time zone to facilitate support |
| **Recoverability** | * Back up all data on a weekly basis   + Store for a minimum of twice the frequency of backup * Back up appointment data on a daily basis.   + Store for a minimum of twice the generate data back-up frequency |
| **Maintenance** | * The system must be maintained via a service level agreement agreed with system developers. * The servers must be maintained by third party cloud storage provider and must be specified in the agreement terms and conditions. |
| **Regulatory and Legal** | * Comply with all regulatory requirements of a data collection and storage system storing sensitive personal data within the UK, including GDPR. |
| **The system COULD** also provide the following | |
| **User interface features** | * Calendar view:   + Display a calendar view for day/week/month with all appointments associated for each user. Receptionist and Managing Director should have full oversight.   + Allow the receptionist to enter appointments via the calendar     - See availability of nurses, doctors, and rooms at that time     - Produce an error message if a doctor, nurse, or room that the receptionist is entering a new appointment for, is unavailable at the that time. * Error messages:   + Where incorrect data is entered into a constrained field, the system should inform the users as appropriate * Reporting function:   + Provide the ability for the Managing Director to produce reports from the data stored in the system in whichever way they would like. |

## Use cases

### Use Case Diagrams (UCD) and commentary [10%]

#### **General commentary**

When putting the UCD together we found it quite difficult to know what level of detail was required. On the one hand we were told to keep it simple, but on the other hand we had examples from the lecture materials that looked like quite complex, such as this:

Diagram

Description automatically generated

We also struggled with understanding if the UCD was written PURELY from the aspect of the database, or as the system as a whole; deciding to settle somewhere in the middle on the premise that the use cases should be weighted much more heavily towards what the users **want** from the system they will interact with, and not how the designers want to build it or how it should work.

Researching how to display use case diagrams with multiple actors, and from examples found online, as well as some from the additional case material on Moodle, we decided to represent the Use Case Diagram as follows:

* Primary actors on the left
* The “system” in the middle
* Secondary actors on the right
* Prioritisation using MoSCoW

The “system” in the case is the entirety of the MVC model (including any hypothetical front end) and not just the database we will be tackling in Tasks 2 and 3.

#### **Actors**

To start with, we analysed who the actors in the system would be and decided that there were 5 primary actors to represent in the UCDs –

1. Receptionist
2. Doctor
3. Nurse
4. Managing Director
5. System Admin

In a larger, more technical version put together, there was a secondary actor to represent, with there being some time specific elements within the system:

1. Time

In that version, some actions were also shared by all users in the system and in order to represent them “ALL users” was added as its own actor in order to avoid repetition or lots of crossed lines.

In the final submitted version, this has been omitted in favour of simplicity, however, the more complex version has been included in Appendix 3 to the project for the reader’s interest – [click here](#_Comprehensive_UCD)

#### **Receptionist**

On reading the brief and looking at the available templates, such as the appointment record and the new pet registration form, it becomes clear that the receptionist has the most points of contact with the system, so they were listed as the first actor. The Use Cases they have are as follows:

1. Create and edit owners
2. Create and edit pets
3. Create, edit, and cancel appointments
4. View outstanding appointment fees

For 1. 2. & 3. an include is added so that any creation, editing or cancellation should write these to the relevant record. This is to ensure that this is done immediately and not held locally anywhere to do later – reducing the risk of data loss.

For 4. it states in the text that any same day cancellations must be paid on the next visit. The Nurse would want to be able to see that to charge the customer on their next visit. Marking payment for them would fall under editing appointments and therefore doesn’t necessarily warrant its own use case.

Something that’s not explicitly said in the brief, but should be useful to the Receptionist, would be the ability to download and/or send pet records e.g., when a referral to an external doctor is made or when the pet becomes too old to be seen at the surgery anymore. This was added as a “could”.

#### **Doctors**

Doctors have the second most points of contact with the system, however, their interactions are much lower than the receptionist. After reading the brief it was determined that the use cases for Doctors would be as follows:

1. View all appointments assigned to me
2. Create and edit consultation card after an appointment
3. Prescribe medication
4. Advise cost and availability of prescribed medication

For 1. This is pretty much the crux of the system – they no longer want a paper-based appointment diary and want to make it electronic and non-centralised so everyone can see their appointments from their own terminals.

For 2. An include was added to write this to the diagnosis record. The consultation card contains appointment information which will already be captured in the appointment record, but also includes diagnosis, so should be written to the appropriate record. As with any data the receptionist is inputting, this is done as an include so it is instantaneous.

For 3 and 4. It’s not explicitly stated in the brief that Doctors want to prescribe medication, however some medicines aren’t available OTC so it was assumed this would be the case. It does state that the doctor would like to be able to advise on the pharmacy/pharmacies that have the drug and how much it should cost.

It also says that the doctor would like to be able to make referrals or deferrals, but as these are written into the diagnosis it was considered to be part of creating the consultation card. It was also assumed that an internal referral would require the receptionist to make a new appointment under another doctor and external referrals were outside of the system. Deferrals are - for all intents and purposes - just new appointments with the same doctor, so again, the receptionist would just schedule a new appointment. Neither of these cases would require any further use case creation.

#### **Nurse**

Nurses only need to do one thing and one thing only – view appointments assigned to them. From the brief, they don’t input or create consultation cards, they don’t create or input appointments, pets, or owners, but they will want to know what appointments are assigned to them so they can attend.

#### **Managing Director**

This isn’t explicitly asked for, but with any piece of information gathering software, the ability to run reports and analyse data sets to improve business is an extremely powerful tool. The use case of “run and download custom management reports” was added to cover this, but only included as a “could” in the MoSCoW prioritisation.

#### **System Admin**

In the hypothetical system we are creating, and in looking at the structure of NPC as described in the brief, there doesn’t appear to be anyone qualified to administer the system. So, a different type of user/third party staff has been assumed, or would be suggested, with the idea that they could create and maintain some desirable features for the system. Primarily, they would have the ability to have full admin oversight, create users for login purposes, and then further create and assign user roles to those users for access, security and audit trail purposes. With this in mind it was decided that the following use cases are “shoulds” under MoSCoW prioritisation:

1. Create and edit users
2. Create, edit, and assign user roles
3. Edit any field with full admin access (excluding passwords)

Creating and editing users has an include to writing to the user record.

Create, edit, and assign user roles has excludes to writing to the user record and the role record. This is because when you are creating and editing roles, it isn’t necessary to write to the user record, but must write to the role record. Then when assigning roles, it isn’t necessary to write anything to the role record, but it must write to the user record.

To simplify editing **any** field, an include was added to “write to the appropriate record” as a catch all for simplicity’s sake, avoiding lines going all across the system in the Use Case Diagram.

Diagram

Description automatically generated

### Use Case Specifications [5%]

It’s specified that we should produce at least one Use Case Specification per group member, so the are as follows:

#### **Ammar:**

|  |
| --- |
| **Use Case: *Consultation Card*** |
| Owner: Doctor |
| **Pre-Conditions** |
| * Appointment must be booked * Appointment must be completed |
| **Post-Conditions** |
| * Payment * Referral or deferral * Medication * Records stored |
| **Primary Path** |
| * Appointment is made * Pet is seen/treated * Consultation card is created * Medication prescribed * Payment |
| **Alternate Path** |
| * Appointment is made * Pet is seen/treated * Consultation card is created, resulting in referral or deferral (extend) * Deferral means pet must receive follow up appointment (include) |
| **Notes** |
| **Consultation Card Fields**  Date:  Consultation ID:  Pet Owner name:  Pet Name:  Doctor’s name in surgery:  Nurse’s name in surgery:  Diagnoses:  Medication required:  Cost of Medication: |

#### **Daniel:**

|  |
| --- |
| **Use Case: Create, edit and cancel appointments** |
| Owner: Receptionist |
| **Pre-Conditions** |
| * Owner is registered in the system * Pet is registered in the system * Receptionist has received request from owner for new appointment, change to an existing appointment or to cancel an appointment |
| **Post-Conditions** |
| Appointment record is created, owner and pet arrive at surgery on date and time of appointment, have appointment with doctors and nurses specified and doctor creates consultation card detailing diagnosis of pet’s problem. |
| **Primary Path** |
| **Creating appointments;**   1. Receptionist collects information from the owner, usually via telephone, specific to the request for a new appointment - .e.g requested date and time, any preferred vet etc. 2. Receptionist logs into the system 3. Receptionist enters required appointment details into system 4. Appointment record is created 5. Payment for appointment is collected on appointment date   **Editing appointments:**   1. Receptionist collects information from the owner, usually via telephone, specific to the request for editing an appointment 2. Receptionist logs into the system 3. Receptionist edits required appointment details in system 4. Appointment record is updated 5. Payment for appointment is collected on appointment date   **Cancelling appointments:**   1. Receptionist received request from owner to cancel appointment 2. Receptionist logs into the system 3. Receptionist cancels required appointment slot in system 4. Appointment record is updated 5. Payment if same day cancellation is collected on next visit 6. Payment is marked as paid on appointment record |
| **Alternate Path** |
| 1. Pet is >12 years old    1. NPC does not care for pets over 12 years old therefore appointment cant be created 2. Requested time slot or requested doctor not available    1. Suggest another available time slot to suit conditions    2. Enter required details into system    3. Appointment is created 3. Owner and pet do not turn up to appointment    1. Appointment is considered cancelled – follow from Cancelling Appointments – 2. in primary path 4. Owner calls up to change or cancel appointment on day of appointment    1. Mark appointment as cancelled and detail £5 same day cancellation charge    2. Start primary path again Creating an Appointment from 1. again.    3. At 4. change to collect appointment fee + cancellation fee at next visit.    4. Mark cancellation fee as paid on appointment record 5. Receptionist is sick / on holiday / otherwise unavailable    1. Another assigned user starts from 1 on required primary path |
| **Notes** |
| No mentioning of the extent of receptionist’s IT skills – training will be required. |

#### **Shahab:**

|  |
| --- |
| **Use Case: *PRESCRIBE MEDICATION*** |
| Owner: VET DOCTOR |
| **Pre-Conditions** |
| * A pet’s information is present within the system. * An owner’s information is present within the system. * An appointment date has been input into the system. * A doctor is assigned to the appointment. * The appointment was not cancelled. |
| **Post-Conditions** |
| * The appointment resulted in a diagnosis, not a referral or deferral. * The prescription was input to the prescriptions record within the system. * The doctor may advise on cost of the medication as well as the pharmacy that stocks it. |
| **Primary Path** |
| * Within the appointment, the doctor determines the health status of the pet. * Based off this status, a diagnosis is produced. * The diagnosis will require a specific medication prescription. * The doctor will provide the prescription, the pharmacy which stocks it, and any advise on the cost of the medication (Include). * The prescription is written to the prescription record (Include). |
| **Alternate Path** |
| * In the case of a “referral”, the pet is transferred over to the care of a specialist. * The specialist provides the diagnosis for the pet in this instance. * The specialist will provide the prescription, the pharmacy which stocks it, and any advice on the cost of the medication (Include). * The prescription is written to the prescription record (Include). |
| **Notes** |
| * Deferral not included as part of alternative path. * Deferral leads to a follow-up appointment, which would then follow the primary path, the alternate path or go through the deferral loop once more. |

#### **Wassim:**

|  |
| --- |
| Use Case: Receptionist Booking Procedure |
| Owner: Receptionist appointment booking process |
| Pre-Conditions |
| \* Receptionist books appointment  \* Pet attends appointment with owner |
| Post-Conditions |
| \*Follow up appointment booked for (inconclusive diagnosis) or (routine check-up) |
| Primary Path |
| Receptionist books appointment over phone  \* Appointment attended by owner & vet  \* Based on diagnosis referral can be made (extend) |
| Alternate Path |
| \* Appointment booked by receptionist  \* Owner has to cancel appointment  \* Receptionist logs in cancelation  \* Cancelation fee might be taken depending on notice (extend) |
| Notes |
| \* Noticed that the receptionist role within the new system to work, act like a central unit to keep  Everything intact, cancelation and fees associated with it are only an extension of the receptionist role |

# Task: Part 2 – Database Design and Implementation [45%]



## Top Down ERD [10%]:

### On attributes

#### **Pet age**

One point to note early on in this, is that we quickly decided not to use “pet age” as an attribute of pet, deciding instead that pet date of birth was much more appropriate. As the fee structure, and the ability to get pet appointments at NPC is worked around the pets age, this is a very important topic point to address.

True is the case that a lot of pet owners may not know the date of birth of their pet, but then it would also be the case that they don’t know the true age of their pets, so estimates can be made.

What pet date of birth does is remove an attribute that is static and would require manual update, with one that is also static, but can provide for dynamic calculation of age: e.g., age at the time of the appointment to calculate fees and to calculate if the appointment should even happen.

#### **Appointment attended**

We also decided that the “attended” field in the dairy record was the same as asking if the appointment was cancelled through non-attendance, so we removed this and set an assumption that if there ever is a non-attendance of an appointment, it is considered a same day cancellation.

#### **Nurse office**

The text in the brief states that nurses all have the same office number, so adding each one into the nurse record adds unnecessary redundancy – the assumption would be that anyone using the system knows the nurses have one central office with number 00.

### Entity derivation

Step 1 – Consider nouns:

* Nurse
* Doctor
* Receptionist
* Consultation
* Appointment
* Referral
* Deferral
* Diagnosis
* Medication
* Pet
* Owner
* Managing director
* Fees
* Pharmacy

Step 2 - Asked first question – “is there more than one?”:

* **Removes:**
  + Receptionist (based on assumptions)
  + Managing director

Step 3 Asked Second questions – “Can I think of a suitable key?”

* **Removes:**
  + Referral
  + Deferral

Step 4) Asked Third question – “can I think of data to be held?”

* **Removes:**
  + None

Step 5) Considered the relevance – in order to do this, we considered if any of them were just attributes of the others.

Nurse

Doctor

Appointment

Medication

Pet

Owner

Diagnosis

Pharmacy

Fees

On relevance, it seemed that appointment and consultation were the same thing. When considering their relationships to nurse and to doctor, they were identical, and the attributes within consultation would be redundant having appointment and diagnosis as attributes, so it was decided to merge appointment and consultation and use the consultation ID on the example data provided, as appointment ID instead.

This produced the Top Down ERD in Appendix 1) – [click here](#_Merged_Top_Down)



# Bottom Up ERD [5%]:

Approaching the Bottom Up ERD; we considered the following, at their given normalisation stages:

|  |  |  |  |
| --- | --- | --- | --- |
| **Dependencies** | The key | The whole key | Nothing but the key |
| **Rules to apply** | Remove repeating units | Remove partial dependencies | Remove transient dependencies |
| **Questions to ask** | For the PK, how many of "you" are there? | Do "you" depend on the whole key or only part of the key? | Do "you" depend upon anything more immediately than your current key? |
|  |  |  |  |
| **UNF** | **1NF** | **2NF** | **3NF** |

Then we applied this to the pet registration form, the appointment diary, and the pet consultation card.

#### **Pet Registration form:**

Graphical user interface, application

Description automatically generated

#### **Appointment diary:**

A picture containing calendar

Description automatically generated

#### **Consultation card:**

Table

Description automatically generated

# Merged Top Down and Bottom up ERD [3%]

When approaching the merge, there were two main differences between the Top Down ERD and the Bottom Up ERDs we had created.

1. Appointment diary suggested an app fee being an attribute of appointment, rather than creating the Fee structure as its own entity and giving it a primary key of fee\_id.
2. The consultation card suggested there should be a direct entity relationship between consultation (or as we said earlier, appointment), and medication.

For 1) We decided that in order to maintain the integrity of the fee structure, it would be better that it existed in the database as its own entity, which could be queried. In hindsight we took the wrong approach. Fees didn’t need to be their own entity and could exist as a lookup table. At the time we didn’t even know lookup tables existed, let alone how they were different to entities. In the final table build this was done as a lookup table with a foreign key reference in the fee field within the appointment record to ensure referential integrity, however, this wasn’t reflected in the merged ERD as we decided to stick with how it was displayed in the top down ERD.

For 2) we considered this, but it didn’t seem to make sense that medication could be administered without a diagnosis first, and if we were considering consultation to be appointment, then this would be wrong. However, when taking a step back and thinking rationally, the bottom up might suggest this, but in the table design, the consultation card is more likely a joining of certain aspects of the appointment record, with the diagnosis record. So, when taking that into consideration, there isn’t a direct link between appointment and medication, but more likely one between diagnosis and medication.   
  
Using the rationale above, we decided that the Bottom Up ERD all but confirmed our Top Down ERD and we were ready to start creating tables.

To review again – [click here](#_Initial_top_down,)

# CREATE TABLE statements in Oracle including constraints:

#### **CREATE TABLE and DROP TABLE SQL statements**

All CREATE TABLE and DROP TABLE SQL statements can be found in Appendix 5 – [click here](#_CREATE_and_DROP)

When creating the tables, all parent tables were created first, in order that the script could run properly and maintain referential integrity. All drop statements were listed in reverse order at the start of the script for the same reason.

When inserting data into the tables, placing constraints on the tables, and querying the tables, it became apparent we needed to change a few things from the ERDs and will be discussed with their tables and insert statements. There are a few things that are necessary to talk about to understand why it was done this way.

#### **Lookup tables:**

Text

Description automatically generated

We used lookup tables for information that was generally fixed and wasn’t likely to change. We could have added a lot more in, such as the pets breed, age was a possibility (but we handled it differently), doctors office numbers etc. But as we didn’t have comprehensive lists for any of those, and there wasn’t a constraint for them that we hadn’t already handled, then we decided against it.   
We did briefly consider adding “species” as a field under pet, with a lookup table that just stated “Dog”, as Noah’s is a dog clinic, but it does mention that the surgery deals with other pet from time to time, so decided against it.

#### **yesno:**

As many fields were being populated with Yes or No (more so Y and N), it made sense to create a lookup table for Y and N to ensure referential integrity.

#### **gender:**

One of the constraints listed for pet, was that the sex had to be entered as ‘M’ or ‘F’, however, in the pet registration form it mentions the gender of the pet and not sex, we decided that gender and sex were interchangeable terms for the purpose of this project. This constraint can be achieved with a check constraint on the pet gender column; however, a more elegant solution is to add gender as a lookup table. This means that, should NPC decide to, they could collect data on the gender of their customers and staff at a later date. Gender could also be expanded then to include situations where the person does not like to disclose or if they have a different gender association that can all be added. You could then add a check onto the pet’s gender field later to ensure this stays M or F.

#### **fees:**

As mentioned earlier, Fee Structure was changed from an entity to be fees, and was a simple lookup table of the fee without any reference to the pet age or its on ID. This allowed the fee to be placed in the appointment record and be queried upon in a much simpler way. When attempting to do Part 3, question 4 without fee in the appointment table, it felt like it was maybe possible with a nested loop join, but that was something we haven’t yet learnt and something that we didn’t quite manage to work, making this method the preferred option.

#### **apptimeslots:**

In our assumptions, we stated that all appointments would be on a 15-minute basis, and the surgery would be open from 8am to 7pm. This meant that timeslots were created in a lookup table, that could be referenced in a drop-down list in a front-end system, for someone to select, rather than inputting manually.

### Specific constraints

All referential integrity in each table and the ERD structure was managed using the appropriate foreign keys. These can be viewed in the create table statements.

#### **C1. “Observe preservation of entity integrity constraints through use of primary keys. As there are many pets, their unique pet\_id will be sequential and is to be valid between 1000 and 3000 inclusive.”**

To manage constraint C1. In the pet table, the pet ID was constructed as follows:

pet\_id NUMBER GENERATED ALWAYS AS IDENTITY START WITH 1000 INCREMENT BY 1 CONSTRAINT pet\_id\_ck CHECK (pet\_id <=3000) CONSTRAINT pet\_pet\_id\_pk PRIMARY KEY

#### **C2. Doctors’ email addresses must be unique**

To manage constraint C2. In the doctor table, the email column was constructed as follows:

email VARCHAR2(40) CONSTRAINT doctor\_email\_nn NOT NULL CONSTRAINT doctor\_email\_uk UNIQUE

#### **C3. Management have decided that a pet’s appointment date can only be made on Monday or Friday and not any other day of the week. They are understaffed.**

When inserting the data, we realised there were some appointment dates that weren’t on Mondays or Fridays, which made this interested as we couldn’t do this at the create table level. The assumption we made was that decision was a new decision from management, and that as such, should come into effect from the system launch. To that end, it would need to be added as an ALTER TABLE statement AFTER all previous appointments had been entered. The alter table statement looked as follows:

ALTER TABLE appointment

ADD CONSTRAINT app\_weekday\_app\_ck CHECK ((to\_char(app\_date, 'DAY') LIKE 'MONDAY%') OR (to\_char(app\_date, 'DAY') LIKE 'FRIDAY%')) ENABLE NOVALIDATE;

Interestingly, this doesn’t work with just ‘MONDAY’ or ‘FRIDAY’ when using IN, but using LIKE ‘MONDAY%’ will work.

#### **C4. A pet’s age can only be 1 to 12 years. They do not deal with older pets.**

As mentioned previously, we decided to use pet date of birth instead of pet age as this was a more dynamic solution.

In order to address this constraint we considered that there were two points in which this mattered:

1. At the registration of the pet AND
2. At the creation of an appointment for that pet.

To address 1) we added an attribute to the pet table called “date\_of\_pet\_entry”. For all intents and purpose, this is a timestamp from the front end when the receptionist creates the record, however, as we are inserting the data ourselves, SYSDATE was used in the insert fields for this.

What we can do then to enforce this constraint at registration of the pet is use the following:

CONSTRAINT pet\_age\_at\_entry\_ck CHECK (TRUNC((date\_of\_pet\_entry - date\_of\_birth)/365) BETWEEN 1 AND 12)

For 2) we pulled the pets date of birth through from the pet record to the appointment record. This isn’t an elegant solution as it creates redundancy, and it may be possible to do this better if you were designing a front end, but for our database alone this felt like the best option. Pulling the pets date of birth through allows you to check it against the date of the appointment to give the following constraint at table level:

CONSTRAINT app\_pet\_age\_at\_app\_ck CHECK (TRUNC((app\_date - pet\_date\_of\_birth)/365) BETWEEN 1 AND 12)

#### **C5. A pet’s sex will be entered as ‘M’ or ‘F’.**

Discussed previously under lookup tables. This was done via referential integrity to a lookup table called gender.

### Non-Specific constraints:

#### **Fees**

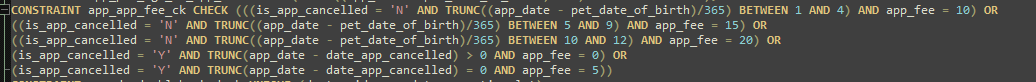
For the fees, having a simple dropdown at the front end with data obtained from a lookup table, that has referential integrity to the fee field in the appointment, still leaves room for human error. To combat this, we added a check constraint on the appointment table that checked the following:

1. That the appointment was cancelled and that the number of days between the appointment date and the cancelation date were > 0.
2. That the appointment was cancelled and that the number of days between the appointment date and the cancelation date were = 0.
3. That the appointment wasn’t cancelled and that the pets age is between 1 and 4.
4. That the appointment wasn’t cancelled and that the pets age is between 5 and 9.
5. That the appointment wasn’t cancelled and that the pets age is between 10 and 12.

For the following checks, it makes sure that the app fee field is set as follows:

1. 0
2. 5
3. 10
4. 15
5. 20

Ensuring the wrong fee cannot be input. This produces a constraint on the appointment table, at table level, which looks like this:



In order for this constraint to work, the following fields cannot be null, and are pivotal to :

Is\_app\_cancelled, date\_app\_cancelled (only required when is\_app\_cancelled = ‘Y’), app\_date, pet\_date\_of\_birth and app\_fee.

If an appointment is changed later, e.g., to update it as cancelled, the front end can prompt the user to input the date that it was cancelled – this should be designed so that a NULL date\_app\_cancelled entry would present an error message to the user in the front end, before trying to insert it into the database. However, for the purpose of this project, we set it so that any change without the required information won’t be saved to the system.

This is by no means an elegant solution, but means that the fee entered can only ever be correct.

#### **Double booking of appointments for pets, included doctors and nurses**

We decided that we wanted to ensure the system wouldn’t allow for the Double booking of Doctors and Nurses to an appointment, or to book two appointments for the same pet at the same time. One of the main goals of the system is streamlining the process and making things more efficient, we wanted to make sure this wasn’t interrupted or deteriorated in any way, by bad practices.

We achieved this using the fields pet\_id/doctor\_id/nurse\_id, app\_date and app\_timeslot – pulling app\_date and app\_timeslot into nurse\_appointment. Understandably, this creates redundancy, however, it provides as solution at the database level, which is what we are trying to achieve with this project.

Putting unique constraints these three together in appointment and in nurse\_appointment respectively, ensures that an appointment can’t be created at the same date and the same time with the same doctor or pet, and, that a nurse can’t be assigned to an appointment if they already have one entered for that date and time.

We assumed that any doctor sharing an office would have the facility to have more than one appointment occurring at one – i.e., they would have distinct spaces such as curtains etc.

#### **Drug cost:**

The idea behind the way that the drug\_pharmacy and prescription tables were created, is that you can query join prescription onto drug\_pharmacy on drug\_id where the units (e.g., mg etc) match. When you consider a prescription, its usually for something such as the following: 30\*100mg.

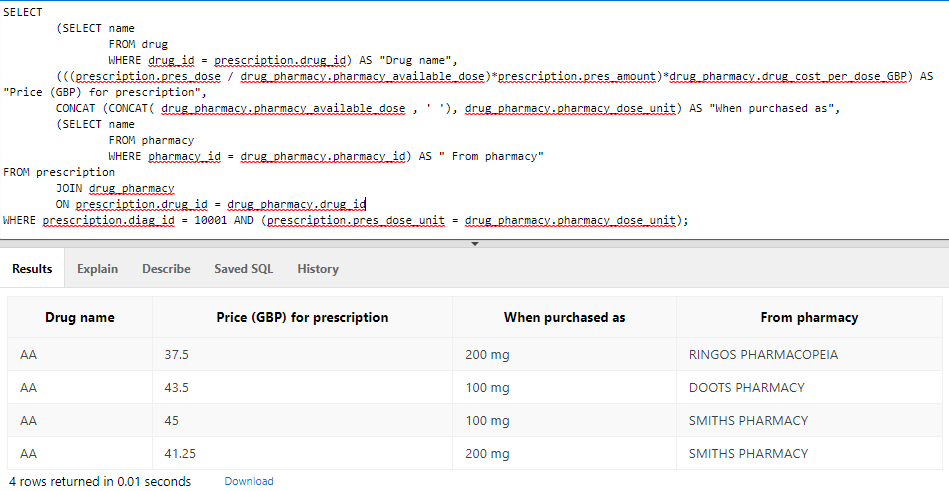
Here you can break this down into amount – 30, dose - 100, units - mg.

Using the information in the prescription about the dose, you can generate a multiplier.

For example, if the prescription says you need 100mg tablets of AA, but you joined onto the pharmacy medication at the drug ID for AA and the units of mg, and they only have 50 mg, you can divide (100/50) = 2. Using this value, you know you need twice the amount of tables as you would if they were 100mg, so you can then multiply the “amount” field specific in the prescription. If that stated you needed 10(amount) tables at 100mg, at 50mg, you would need 2(multiplier)\*10(amount).

Knowing the amount of tablets, you would need, in this case 20, you can then multiply the cost per unit by 20 to get the price of that prescribed medication.

The query was run against the prescription made for diag\_10001 and the results are shown here:



# INSERT all data including FIVE(5) extra tuples into each table

All data from the appendices in the brief was inserted into the tables. In total we have 15 tables, and an example of the INSERT statements can be found below along with any comments. A large number of their ids are generated as sequential numbers just for the sake of saving time making them up.

#### **yesno**

INSERT INTO yesno

(y\_n, yes\_no)

VALUES ('Y', 'Yes');

#### **gender**

INSERT INTO gender

(m\_f, gender\_desc)

VALUES ('F', 'Female');

#### **fees**

INSERT INTO fees

(app\_fee)

VALUES (0);

For the sake of the fee that was required for the appointment – we decided that cancellation fees should also be included as that’s what would be required to be paid (and shown on the appointment record) should the appointment be cancelled.

#### **apptimeslots**

INSERT INTO apptimeslots

(avail\_time)

VALUES (TO\_DATE('08.00','HH24:MI'));

INSERT INTO apptimeslots

(avail\_time)

VALUES (TO\_DATE('08.15','HH24:MI'));

Incremented every 15 minutes until a top entry of 18.45.

#### **owner**

INSERT INTO owner

(first\_name, last\_name, age, tel, email, street\_address, postcode)

VALUES ( 'David', 'Guetta', 53, '01616623454', 'David.Guetta@aol.com', '50 Chester Road', 'M16 4TU');

We dummied up a lot of data for owner – including adding a surname. All we had to go on for owner was a first name.

#### **pet**

INSERT INTO pet (name, type, gender, date\_of\_birth, date\_of\_pet\_entry, colour, weight\_in\_kg, owner\_id)

VALUES ('chappyDog', 'Alsation', 'M', '08 MAR 2019', SYSDATE, 'beige', 03.0, (SELECT owner\_id

FROM owner

WHERE UPPER(first\_name) = UPPER('David') AND UPPER(last\_name) = UPPER('Guetta')));

This looks a lot more complicated than it actually is, and is done for a specific reason. Within the data we were provided, the pet records had the owners name and not their ID. When inserting the data, we looked to see if there was a way that we could insert the owners ID from their name, without having to manually reference the owner table each time, this is also more representative of a real life situation, where if you asked the customer for their name or other details, they would be able to provide them, however they wouldn’t necessarily know their id in your system, unless that was used specifically as your account number.

We found that using a subquery to query the owners name from the owner table, returning the ID, was an option. We added a surname to make things more distinct – e.g., we added two Kylie’s for the owner “Kylie” in the same data, and showed how surname can differentiate. In a larger system with more data, adding postcode in would also be required as the possibility that two owners have the same name is quite high, and the select subquery within the insert wont work with multiline return values.

If we were designing a front-end system, this wouldn’t be necessary as you could have a drop-down table referencing the owner table to achieve this instead.

#### **nurse**

INSERT INTO nurse

(first\_name, last\_name, tel, email, is\_Full\_Time)

VALUES('Davidth', 'Chapelle', '0161 3446 801', 'chapelleD@noahs.com', 'N');

For employee emails (doctors and nurses) we used the email format of surname + first initial @noahs.com. We assumed that some of the doctor’s emails were set up wrong, but the majority were using that format.

#### **doctor**

INSERT INTO doctor

(first\_name, last\_name, office\_num, tel, email, is\_Full\_Time)

VALUES('Claire', 'Cleverly', 12, '0161 3446 543', 'cleverly\_cl@noahs.com', 'N');

#### **pharmacy**

INSERT INTO pharmacy

(name, street\_address)

VALUES(UPPER('ringos pharmacopeia'),UPPER('1 dramhall lane'));

#### **drug**

INSERT INTO drug

(name)

VALUES(UPPER('metronidazole'));

#### **appointment**

INSERT INTO appointment

(app\_id, app\_date, app\_timeslot, pet\_id, pet\_date\_of\_birth, doctor\_id, is\_app\_cancelled, app\_fee)

VALUES(105078, TO\_DATE('06/09/2021' , 'DD/MM/YYYY'), TO\_DATE('10:30' , 'hh24:mi') ,1000, (SELECT date\_of\_birth FROM pet WHERE pet\_id = 1000), 2201, 'N', 10);

Pet date of birth is pulled through using a select from the pet table, with the pet\_id. Appointment IDs were assumed to be sequential, starting from 100000, however, we didn’t put that in the table as the sample data we have has very large ranges of appointment IDs.

As mentioned previously, after the appointment inserts, the table was altered to add the Monday and Friday appointment day constraint.

#### **diagnosis**

INSERT INTO diagnosis

(app\_id, diag\_desc)

VALUES((SELECT app\_id FROM appointment WHERE pet\_id = 1000 AND doctor\_id = 2201 AND TRUNC(app\_date) = TO\_DATE('05/08/2021','DD/MM/YYYY')), 'Bring him Tuesdays 10 to 12.pm');

We used a select statement to get the app\_id from the appointment table where we knew the pet\_id and the doctor\_id and date, as the examples on the consultation cards included these numbers and it saved cross referencing with the appointment table. For this we had to assume that where it says “later” in the brief, with respect to completing the consultation card, meant later, but also the same day.

#### **nurse\_appointment**

INSERT INTO nurse\_appointment

(app\_id, nurse\_id, app\_date, app\_timeslot)

VALUES(105078, 22004, (SELECT app\_date FROM appointment WHERE app\_id = 105078) , (SELECT app\_timeslot FROM appointment WHERE app\_id = 105078));

Select statements are used here to match the app\_date and app\_timeslot too to the appointment number. These are used to check the availability of the nurse in the double-booked constraint.

#### **prescription**

INSERT INTO prescription

(diag\_id, drug\_id, pres\_dose, pres\_dose\_unit, pres\_amount, pres\_guidance)

VALUES(10003, (SELECT drug\_id FROM drug WHERE name LIKE UPPER('%metro%')), 15, 'mg',15,'One 15mg table to be taken per day for 15 days. Do not give to animal on an empty stomach.');

We used select statements here to get the drug\_id from the drug table, as the examples we had from the consultation card were the drug name in the diagnosis field. In a larger model this probably would need more consideration.

#### **drug\_pharmacy**

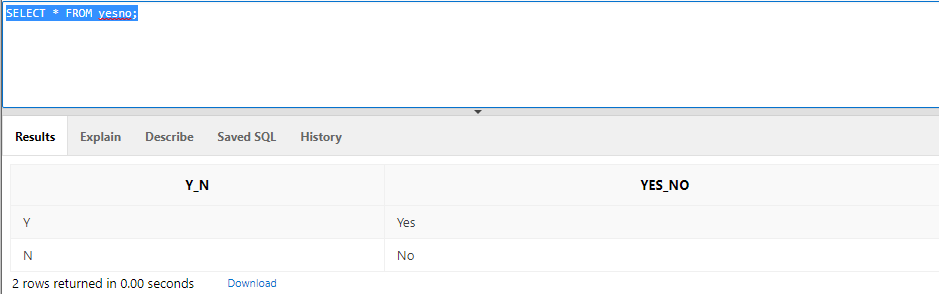
INSERT INTO drug\_pharmacy

(drug\_id, pharmacy\_id, pharmacy\_available\_dose, pharmacy\_dose\_unit, drug\_cost\_per\_dose\_gbp)

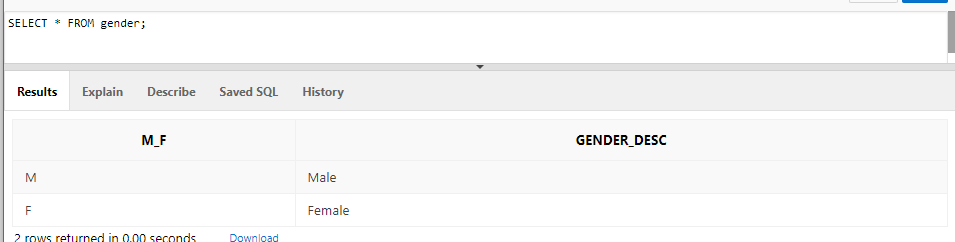
VALUES((SELECT drug\_id FROM drug WHERE name LIKE UPPER('%AA%')),(SELECT pharmacy\_id FROM pharmacy WHERE name LIKE UPPER('%Smiths%')), 100, 'mg', 1.50 );

# Retrieve all data from tables using a select query

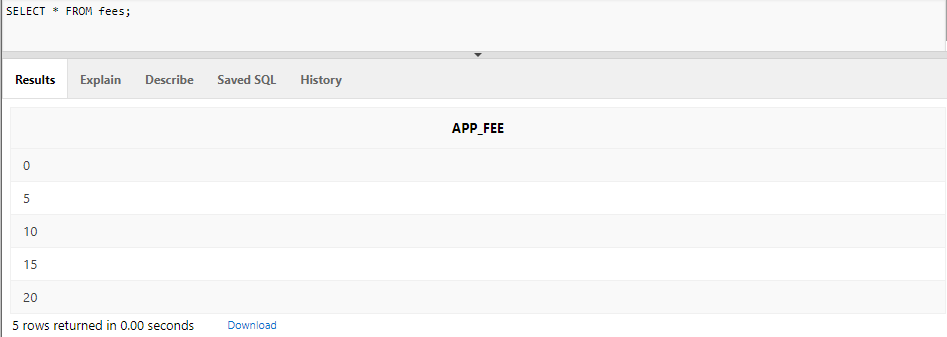
#### **yesno**



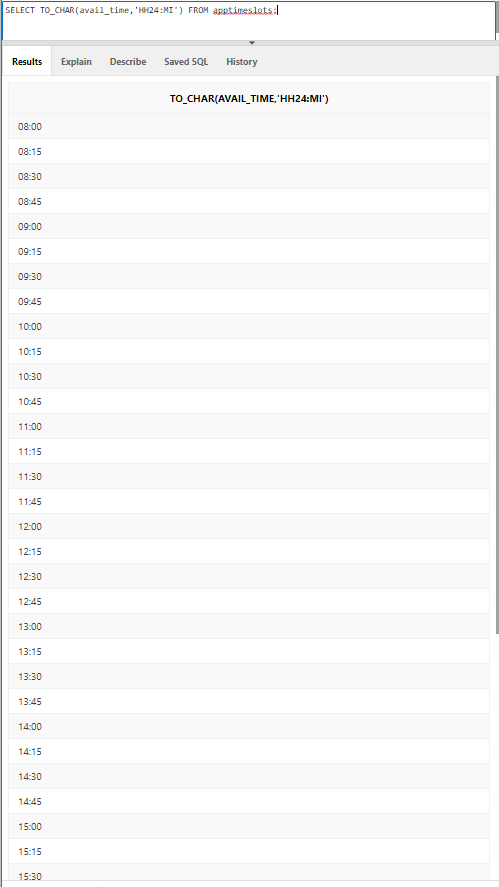
#### **gender**



#### **fees**

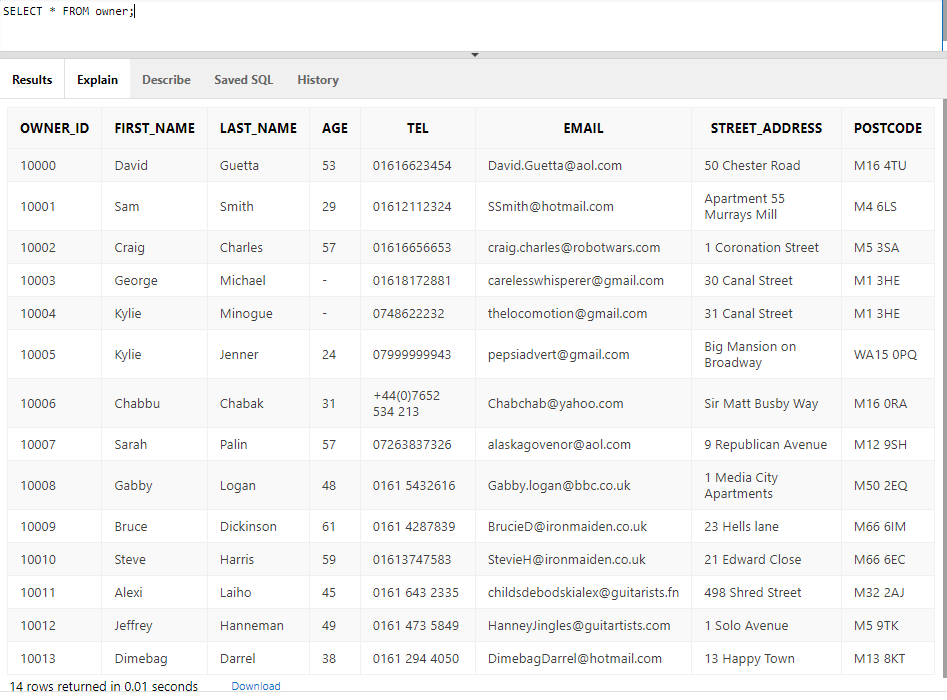


#### **apptimeslots**

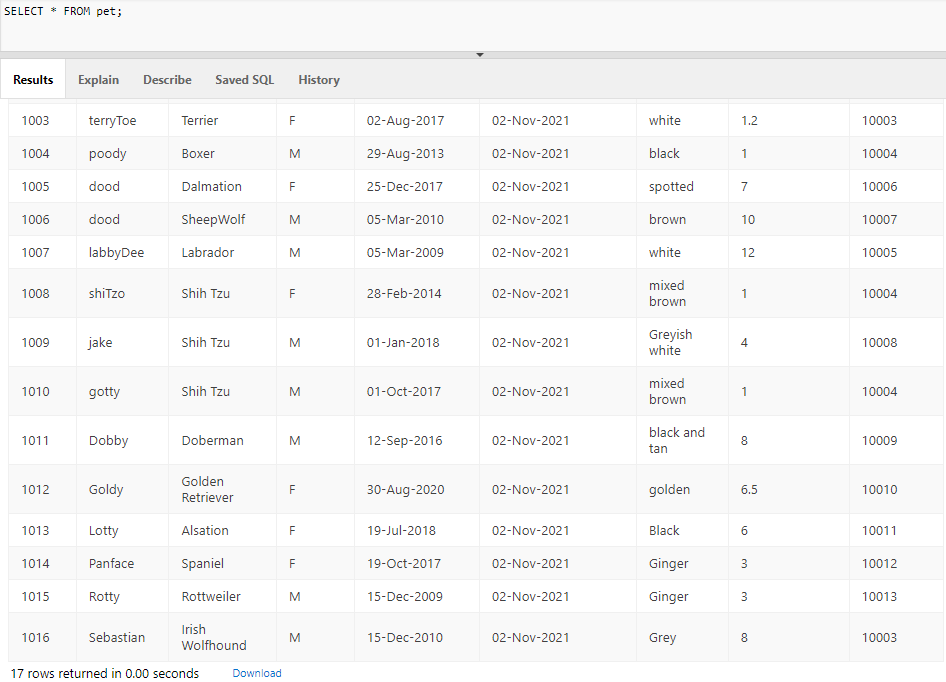
SELECT TO\_CHAR(avail\_time,'HH24:MI') FROM apptimeslots;

As there are many rows to this table – I won’t include them all for the sake of the document size.

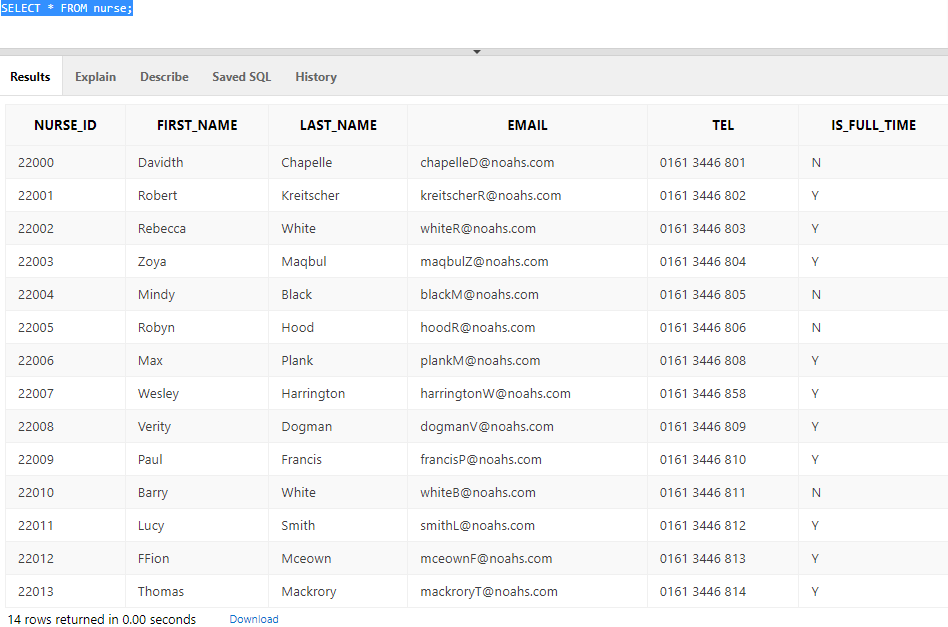
#### **owner**



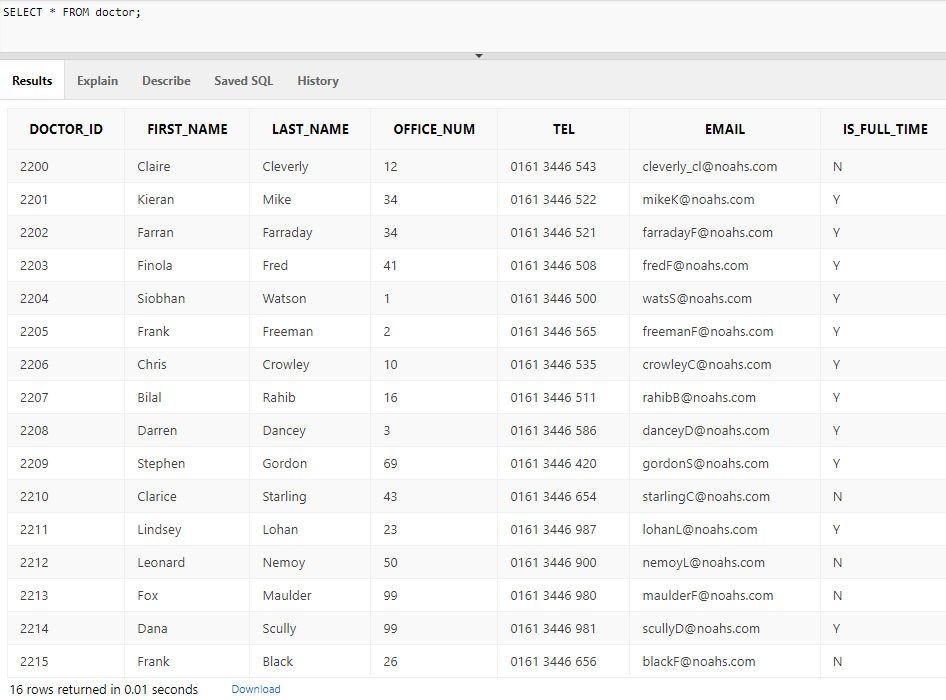
#### **pet**



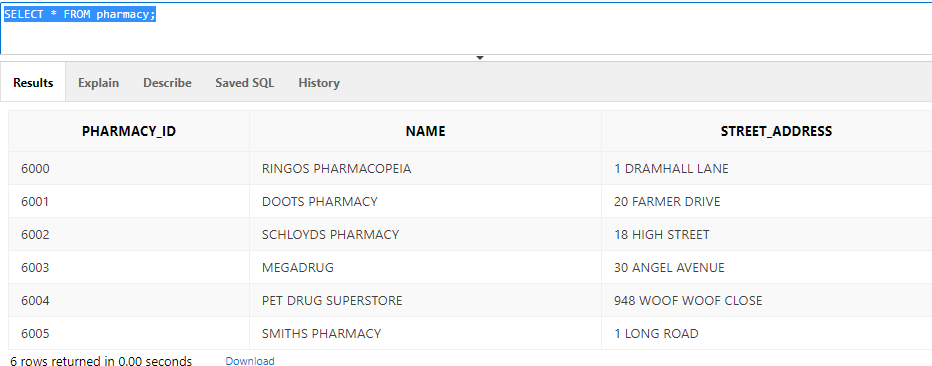
#### **nurse**



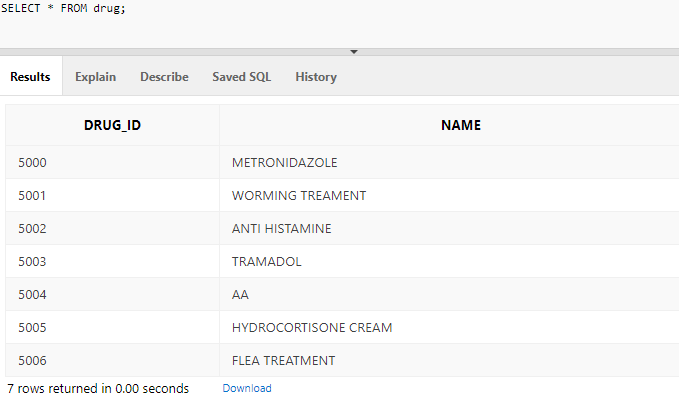
#### **doctor**



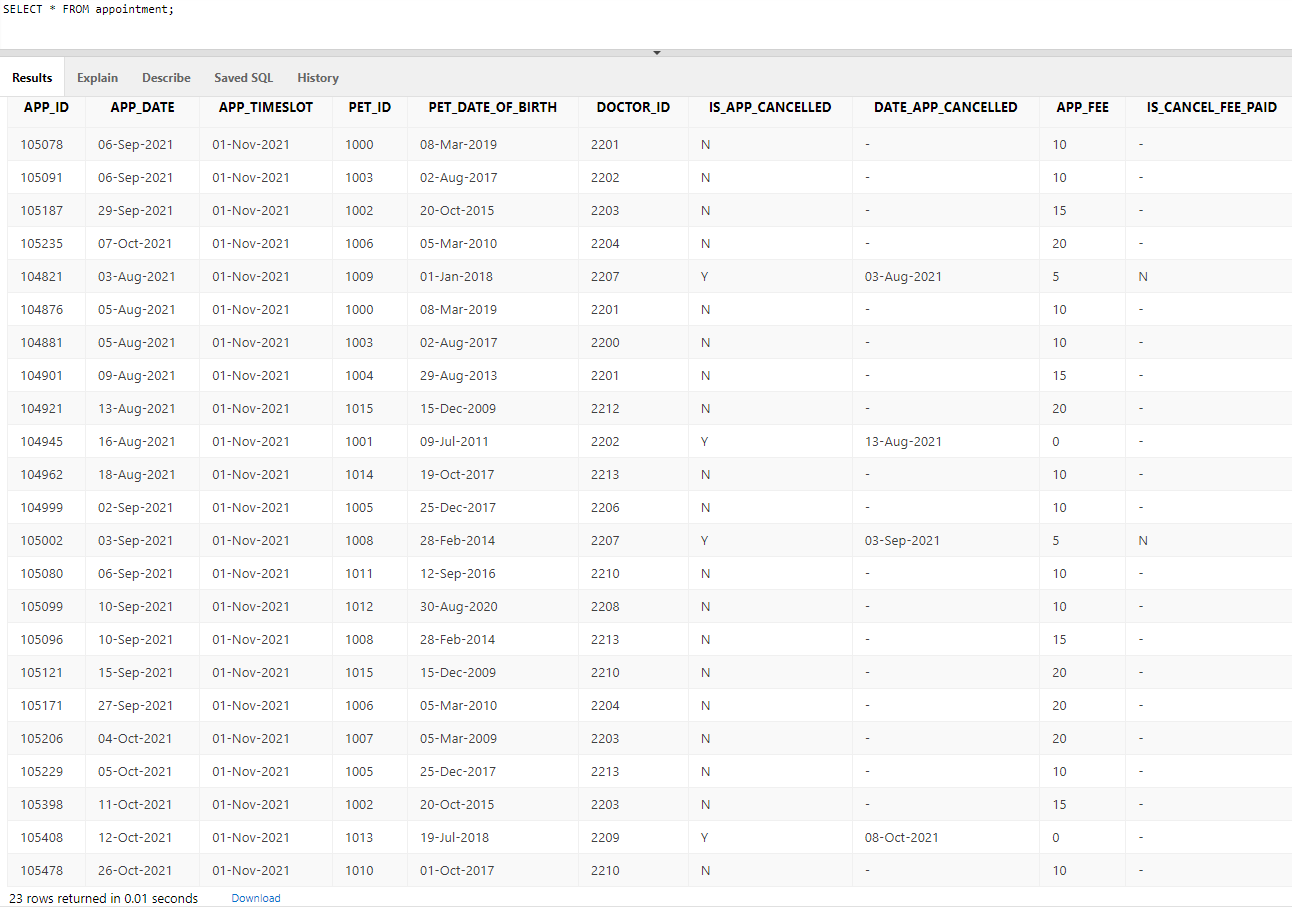
#### **pharmacy**



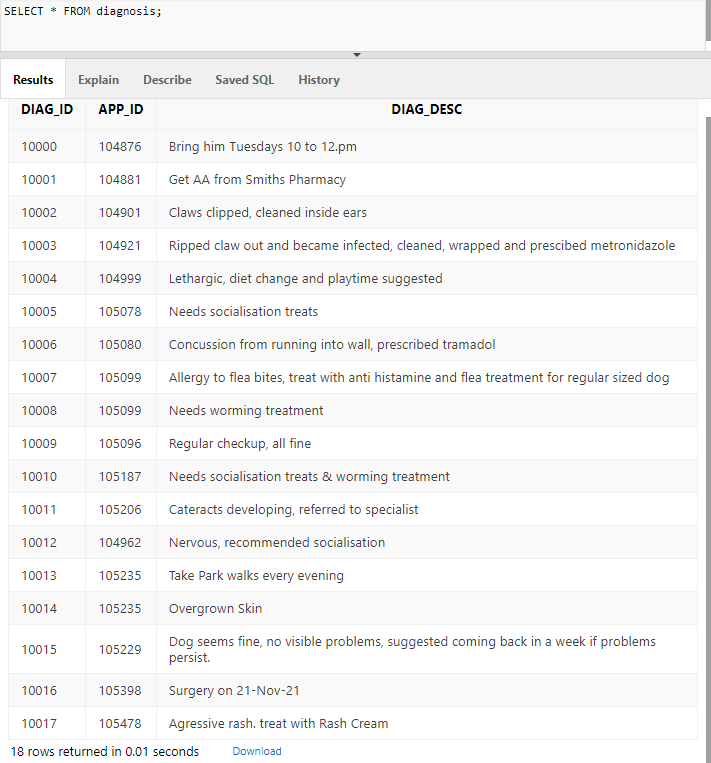
#### **drug**



#### **appointment**



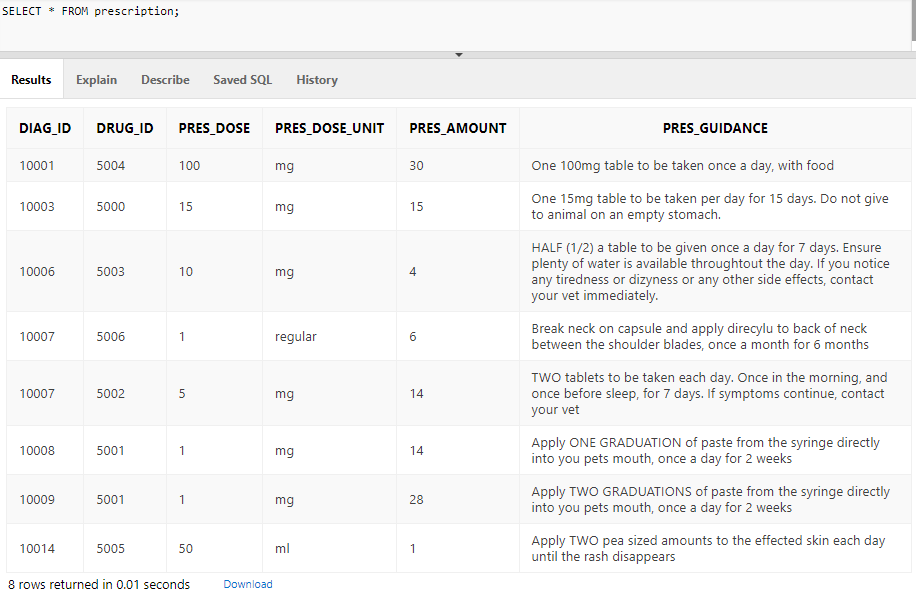
#### **diagnosis**



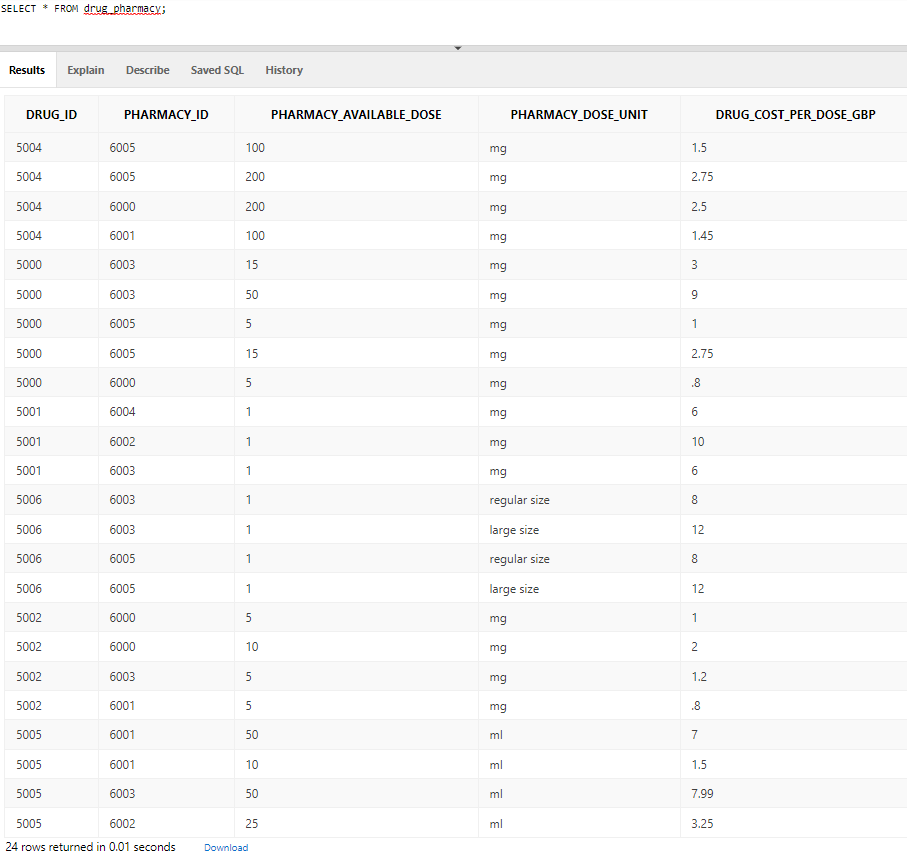
#### **nurse\_appointment**



#### **prescription**



#### **drug\_pharmacy**



# Critical design analysis and improvements

Aside from some obvious small improvements, like removing the 2000 pet constraint and the Monday to Friday appointments, there are some more considerable improvements that could be made. As the entities in the ERD and the respective tables created in oracle are structured in a specific way to cater for NPCs needs, it would make sense to consider how these may differ in another, larger scale Vet, or even be ported across multiple vets.

#### **Employee structure:**

We made the assumption that Noah’s only has one receptionist and it only has one manager (managing director), because of this, they were omitted from the entity diagram and merely exist as lookup tables in Oracle. However, another vet might have multiple receptionists or a different employee structure altogether, to that end we would suggest removing the entities that related to specific jobs and replace them with Employee, Job, department, and Location.

Considering the ERD for this, it would look something like the ERD in Appendix 6 – [click here](#_ERD_including_suggested)

This structure could then also cover things like who schedule the appointment (e.g., receptionist), who cleaned the appointment room etc. if necessary.

#### **Surgical procedures and other treatment options:**

In NPCs model and the information from the brief, the only outcome from a diagnosis is deferral, referral, or medication. This isn’t really realistic, and a lot of larger veterinarian surgeries will have the option of surgical procedures too. Along with things like physiotherapy, behavioural therapy etc.

If we focus on surgical procedures, considering where they fit into our ERD, they could be considered as appointments for the sake of storing data, but have different constraints. E.g., the appointment table could have a new attribute that says, “Surgery y/n” as a field, and where it’s selected as ‘Y’, the doctor\_id is forced NULL and a new weak entity between doctor and appointment is created called “Doctor\_surgery” to accommodate for more doctors being available, such as anaesthesiologists, or consults. The app\_id is then considered the same as the surgery ID.

Other treatment options can be considered as appointments too – and the appointment table could be adjusted to a referential lookup to the services that the surgery offers, which each in turn can place different constraints on the table.

#### **Room & other resource management**

Currently, there is no means to track limited resources in NPCs table structure. For example, if an appointment required a specialist piece of equipment that a surgery had limited amounts of, it could be booked and tracked against appointments to ensure that appointments weren’t being booked when equipment wasn’t available.

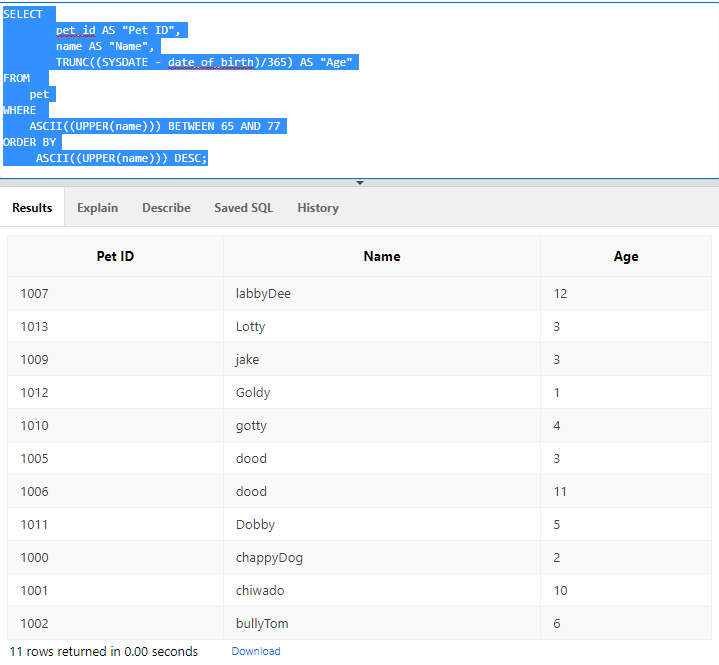
Things like consultation and surgical rooms could also be tracked, along with specialist equipment like hoists.

To do this, a lookup table would be created for things in limited amounts, the relevant columns added to the appointment record, then a unique constraint at the table level level would be added for that resource, appointment date and appointment time added.

# Task: Part 3 – Query Implementation [25%]



# Write an SQL statement that lists the pet id, pet name, pet age showing the columns as “ID”, “Name” and “Age”. Their names should be those starting with letters A to M. Sort the results by the pet id in descending order. [4%]

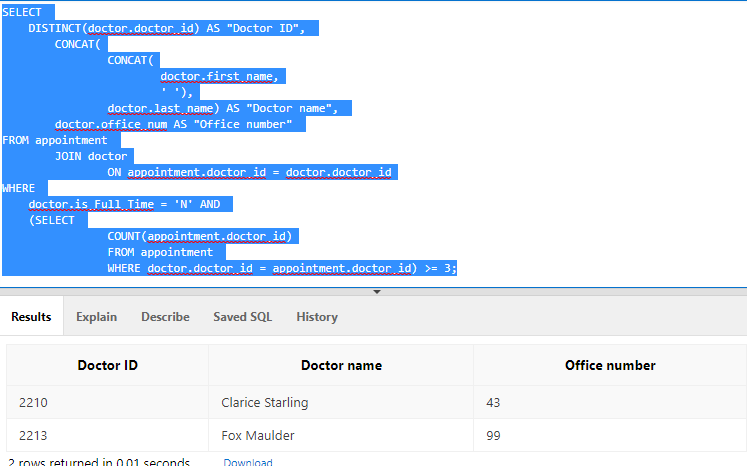


Age was calculated used SYSDATE and the date of birth of the pet.

Using the ASCII command you can convert the first letter to a number, which allows the WHERE statement to be a BETWEEN command, instead of using LIKE ‘A%’ AND LIKE ‘B%’…… etc. As some names were capitalised and some weren’t, ordered by the ASCII number of the upper case of name, in descending order.



# Write an SQL statement that shows the most overworked part-time vet doctor, i.e., any vet with 3 or more appointments for pet clinics, listing details such as id, name and office number. [4%]



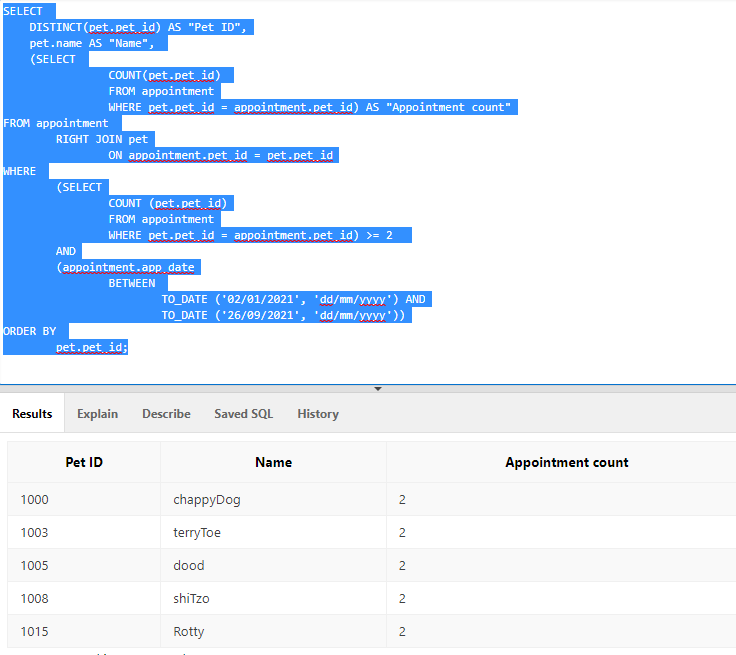
Used a join of doctor and appointment on doctor id to link doctor and appointment records to link the appointment count to the doctor specific information.

Seleced distinct part time pet doctors to avoid repetition

Concatenated the first and last doctor name.

Filtered by part time in the WHERE conditional, also included an appointment count in the conditional to filter anything >= 3.

# Write an SQL query that lists pet details with 2 or more appointments between 2nd Jan 2021 and 26th Sep 2021. The columns should include pet id, name and count of the number of appointments. [4%]



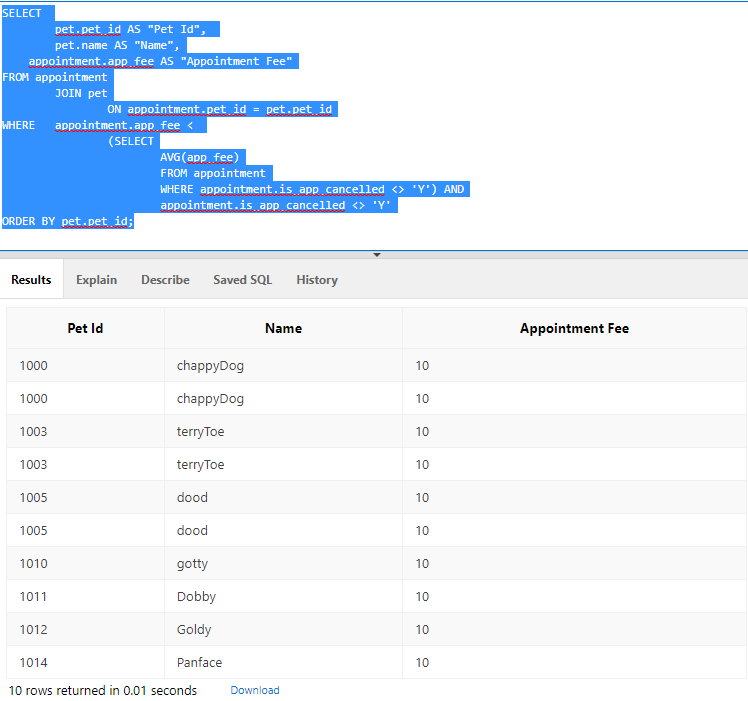
Joined pet onto appointment on pet\_id to get pet specific information alongside an appointment count for that pet.

Selected DISTINCT pet to avoid repetition, could group this way as all columns for a give pet would repeat.

Filtered by count of appointments for that pet >=2

AND by appointment date being between 2nd Jan 2021 and 26th September 20201.

# Write an SQL query that finds pet id, name and cost of appointments, such that the cost of the appointment is less than the average appointment cost of all pet appointments. [4%]



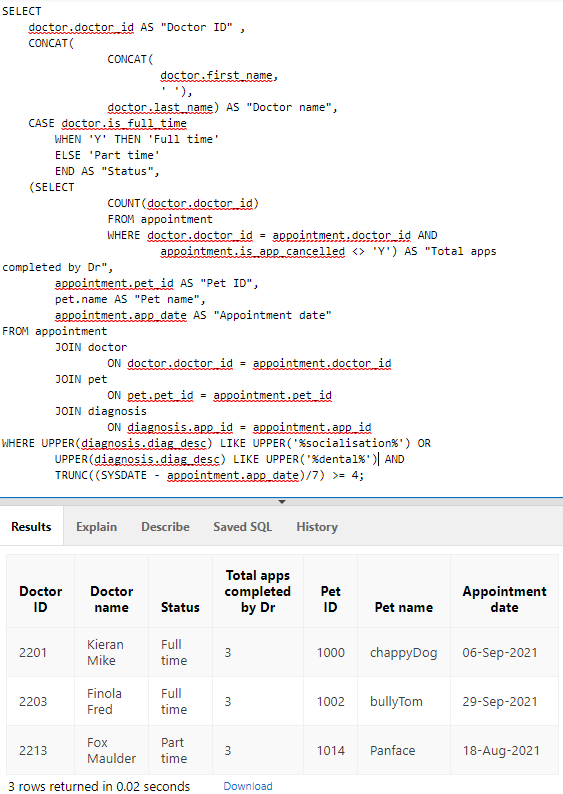
There’s no context in this table, but its showing rows that exists for a pet, in the appointment table, where the appointment cost for that row, is less than the average cost of all appointments.

Specifically, NOT included in this, are cancellation costs – e.g., £5 same day cancellation fees and the £0 fee for cancellation not on the same day as they are not considered to be appointment costs.

This was achieved by joining pet onto appointment on the pet\_id and selecting the pet Id, pet name and appointment fee.

Then in the WHERE, the appointment fee column was filted using < , against the average of all appointments that weren’t marked as cancelled. WHERE also filters out any cancelled appointments as the costs of these aren’t considered appointment costs, but cancellation costs.

# Write an SQL statement that lists a doctor id, status, the total number of appointments he/she has handled, the pet involved (id, name) and the dates these took place. The query should only show cases whose diagnoses involved “socialisation” or “dental” work and where the appointment date was at least 4 weeks ago. [5%]



Here we used a quadruple join of appointment, doctor, pet and diagnosis in order to be able to pull pet and doctor specific information through, count the number of appointments and filter by diagnosis.

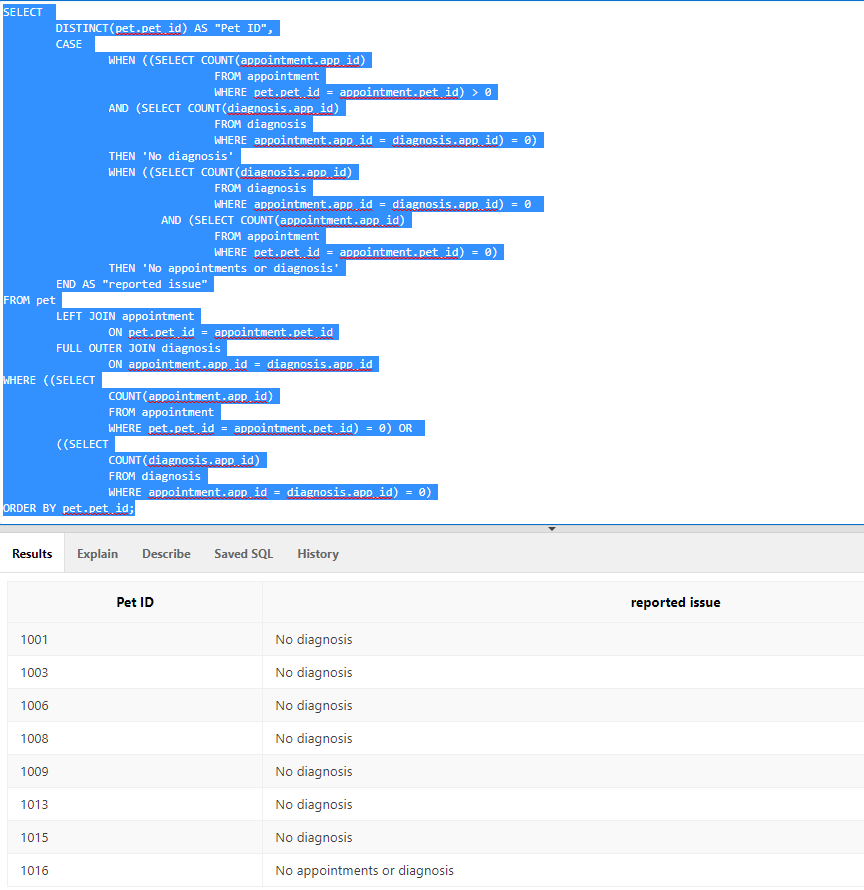
For “Status” we considered this to be whether they were full or part time, and used a CASE expression to express this from the “is\_full\_time” column in the doctor table.

The number appointments the doctor has handled was considered to be the number of appointments they have had that haven’t been cancelled, and the count was don’t to reflect that and expressed as “total apps completed by Dr”

The results were filtered in WHERE using UPPER(diagnosis.diag\_desc) LIKE UPPER(‘%socialisation) - and the same for dental – to search for those terms, or similar ones, and remove any case sensitivity.

The results also filtered any appointments less than 4 weeks old, by using SYSDATE minus the respective appointment date, dividing by 7 to get the number of weeks, truncating to removing an possible carry over, and then checking >= 4.

# Write an SQL query that finds pet ids with no appointments or diagnoses so far. [4%]



This was approached by ensuring that any join made didn’t remove NULL fields for appointment ID or for Diagnosis ID. To do this, we used pet and left joined appointment and full outer joined diagnosis.

After that, we then selected distinct pet as to not have any repeats if the pet had multiple appointment cancellations, where they had an appointment but no diagnosis.

Next, we set a case to count the number of diagnosis and appointments: if the pet had 0 of both, it would display as “No appointments or diagnosis”, if it had an appointment but no diagnosis, then it would express as “No diagnosis”. The third situation of a diagnosis without an appointment for a pet is impossible as there is no direct relationship with the pet to the diagnosis, only through appointment. So if diagnosis could exist (it cant in our tables), without an appointment, it wouldn’t reference back to any pet, so there would be a NULL value in pet ID.

Lastly WHERE was set to filter out anything with an appointment count of 0 OR a diagnosis count of zero.

# APPENDICIES

# Assumptions Made

To return to the start of the document – [click here](#_Task:_Part_1)

If not otherwise stated in the relevant section – these are the assumptions made for the projects:

* Customer and owner are used interchangeably
* NPC = PNC = Noah’s Pet Clinic
* Visit == appointment/consultation
* Assume emergency treatment, if any, is outside of the scope of the system – consultations are limited to appointments only.
* Each Appointment is 15 minutes long for scheduling
  + Max appointments per day:
    - assuming opening times of 8am – 7pm = 11 hours working – take from local vets – for use in this project
* Each user on site will have access to their own terminal concurrently making peak user number – 5 doctors, 2 nurses per doctor is 10 nurses, 1 system admin, 1 MD, 1 receptionist. Total = 18.
* No remote log in functionality for doctors, receptionists, and nurses, has to be done at on-site terminal – staff numbers limited to who is present in the surgery. MD and system admin to have remote log in functionality.
* MD will not input data, only use the system to download or view data.
* No accounting requirement from the system has been specified but could be an improvement later.
* “Doctors email addresses must be unique” is specified only for doctors, but this should be for all users and owners.
* Data not stored on site – will make use of third-party cloud storage.
* Cost by age is the same for all types of animal – no distinction made.
* Appointments are being held in the Doctor’s office
* System admin will be maintaining all data regarding fee structure, medicines and pharmacies, and consultation rooms
* No customer access to system – this is an internal management system.
* Pet records need to be able to be packaged and downloaded/sent in case of referrals and/or changing of vets.
* The brief states that “Management have decided that a pet’s appointment date can only be made on Monday or Friday and not any other day of the week. They are understaffed.”
  + This means that appointments can only happen on a Monday or Friday, not that appointments can only be entered into the system by the receptionist on Monday and Friday.
  + Assumption based on being more likely to be shortage of medical trained professionals than receptionists.
  + Assume this is temporary and that functionality could be for 7 days a week later on
* Assume a receptionist is still present 5 days a week
  + No mention of opening times – assume Monday to Friday for now
  + i.e., nurse can enter appointments on a Wednesday, for one of the available days.
* Receptionist is not a medical professional
* Never any mention of Receptionists plural, only singular, assume only 1 receptionist works at the clinic
  + For scalability in future – this may be a bad assumption as hiring a second would mean that receptionist becomes an entity for ERDs and may cause problems in data structure
* System admin will be one external IT staff at least at beginning:
  + No indication if they have any IT staff at present – no user ID for them.
    - Assume any emails set up previously have been done by external support
    - Assume any terminals and hardware set up have been done by external support
  + Using paper systems currently then it should be a safe assumption.
  + See scalability issue for receptionist
  + Assume MD doesn’t want to deal with admin of adding/removing/editing users or data
  + Assuming receptionist will not be system admin – no indication either way, but in order to restrict access for a non-medical professional working at the clinic from editing medically sensitive consultation notes, safer to assume receptionist isn’t system admin. Also, no indication into level of IT training of the receptionist.
* Paper based system of data collection and storage has been used up until this point, which suggests minimal, if any, current hardware, and software available.
  + Limited to possibly printing of typed consultation notes
  + Can derive from this that minimal IT training has been provided as well.
* Sample data fields aren’t comprehensive and “don’t exist” as they appear:
  + Designed to give us an example of the data
  + No current database of records is kept
  + Populate with necessary (pretend) data for the sake of the project
* It states: “There are a variety of pharmacies that stock different types of medicines and they all can be recommended for certain drugs once the diagnosis is done”
  + Assume this means the system can query a drug against a pharmacy
  + This is where the “cost of medication” field data is sourced from.
* Clinical staff records are not comprehensive list of doctors – says “sample data”
  + Only have doctors surname and first initial available
  + No doctor IDs on any other example data provided
  + Email addresses follow a set naming convention:
    - surname(lowercase)first initial(uppercase)
      * assume mike is a surname
    - entry one and entry 5 are errors
      * watsS instead of watsonS
      * cleverly\_cl instead of cleverlyC
* Assume more data on the owner is collected at some point other than what is specified on the Pet Registration Form
  + Owner’s age is specified on the appointment form which isn’t collected on the pet registration form
  + No contact details other than address specified, assume email and phone collected as well

# User Stories

To return – [click here](#_User_stories_1)

#### **Receptionist:**

Name: Little John

As a receptionist I want to be able to:

* Create and manage our customers and their pets in a better way than being stored in the filing cabinets in my office anymore, so people don’t forget to bring them back and they go walkies!
* Search in the system for our customer and pets by names or addresses so I can pull them up quickly when I’m on the phone.
* Schedule appointments in a computer system for our customers so we can’t misplace or lose it anymore like we have on a few occasions before.
* Assign doctors and nurse(s) to appointments so they know where they need to be on the day and can see it in their own calendars and don’t have to keep coming and checking in my office for the appointment diary.
* Cancel appointments when a customer requests over the phone, or when they don’t show up to their appointment - and then see any cancellation fees so I can make sure I’m keeping track of them. I would also like to be able to have an option to say if the customer has paid these on their next visit.
* Avoid having to file anything, if possible, please – doctors turn up with stacks of their consultation cards at the end of every week and put them in my in-tray, so I must spend hours at the start of the week filing them all, and sometimes the pet files and owner files haven’t been brought back yet so it’s a mess!

#### **Doctor:**

Dr Siobhan Watson

As a doctor using a new electronic system, I would like to be able to:

* See all appointments I have that day/week/month, and which pet it is with so I can prepare visits adequately in advance.
* Input consultation notes to the system after an appointment so we don’t have to do it via the consultation cards that we currently leave with the receptionist to file – there have been complaints about cards being filed at appropriate times, so if we can avoid that it would be great.
* Search up previous appointments the pets may have had, and their consultation notes associated with that pet in the case that the pet has an ongoing or pre-existing condition so I can make a more informed decision on treatment options.
* Make a referral to another specialist where my knowledge on the diagnosis isn’t sufficient to advise on treatment options.
* Prescribe medication and advise the customer of which chemists in the area sell the medication I’m prescribing and how much it will cost them.

#### **Nurse:**

Rebecca White

As a Nurse I’d like to be able to:

* View what appointments I’ve got coming up for that day or week so I can prepare in advance, if I could also see what doctor I’ll be with that would be great because some have different preferences on how we set things up during the appointments.

#### **Managing director:**

Big John (son of Noah)

As a managing director I’d like to be able to:

* Have an overview of what is happening in the surgery on a given day/week/month – such as what appointments are happening, and which doctors and nurses are doing them.
* Access the system remotely as I don’t always work on site.
* Run custom reports against numerous things within the business, for example:
  + Appointments per day/week/month to analyse where we are busiest
  + Referrals vs diagnosis to identify where we may have skills gaps
  + Which doctors and nurses vs number of appointments to see if work is being distributed correctly
  + Outstanding fees and collected fees.
  + Etc

#### **System administrator:**

As a system administrator I’d like to be able to:

* Access the system remotely.
* Offer an option to anyone logging into the front-end section that says “forgot my password” so they can get an automatic email recovery instead of having to manually handle password reset requests.
* Create and manage staff records for each staff member within the system so that we can ensure any new starters are added, anyone who leaves is removed and anyone who changes personal information such as names or roles can be modified.
* Define roles within the system and assign them to users which will determine what level of access each individual user has
  + This will help when people edit things they shouldn’t, or don’t know the consequence of – if they can’t edit fields they shouldn’t, I won’t have as much admin to do.
* Input any other data required by the system, including new lookup tables, editing price structures etc.

# Comprehensive UCD

To return to the use case diagram commentary – [click here](#_General_commentary)

See here for a link to the VPD one drive file – down and enter into Visual Paradigm Online to view.   
  
<https://stummuac-my.sharepoint.com/:u:/g/personal/21435099_stu_mmu_ac_uk/EXTu-KCFHlJIun-9kR1H1J8BJIswIK0Uni78R4jfFkmVEA?e=S7Gm8x>

# Initial top down, merged and signed off ERD

To return to the top down ERD – [click here](#_Bottom_Up_ERD)

To return to the merged ERD – [click here](#_Merged_Top_Down)

Diagram

Description automatically generated

# CREATE and DROP TABLE SQL statements:

To return to table creation commentary – [click here](#_CREATE_TABLE_statements)

FOREWARNING – these are much easier to view in notepad++ :

<https://stummuac-my.sharepoint.com/:u:/g/personal/21435099_stu_mmu_ac_uk/EWcDkDAb2G9DlNa5V07tzAgBXwmm4-aMExyAkQY1IoVmXQ?e=nXiJDE>

DROP TABLE drug\_pharmacy ;

DROP TABLE prescription;

DROP TABLE nurse\_appointment;

DROP TABLE diagnosis;

DROP TABLE appointment;

DROP TABLE drug;

DROP TABLE pharmacy;

DROP TABLE doctor;

DROP TABLE nurse;

DROP TABLE pet;

DROP TABLE owner;

DROP TABLE apptimeslots;

DROP TABLE fees;

DROP TABLE gender;

DROP TABLE yesno;

CREATE TABLE yesno (

y\_n CHAR(1) CONSTRAINT yesno\_yes\_no\_pk PRIMARY KEY,

yes\_no VARCHAR2(3)

);

CREATE TABLE gender (

m\_f CHAR(1) CONSTRAINT gender\_m\_f\_pk PRIMARY KEY,

gender\_desc VARCHAR2(6)

);

CREATE TABLE fees (

app\_fee NUMBER(6,2) CONSTRAINT fees\_app\_fee\_pk PRIMARY KEY

);

CREATE TABLE apptimeslots (

avail\_time DATE CONSTRAINT apptimeslots\_avail\_time\_pk PRIMARY KEY

);

CREATE TABLE owner (

owner\_id NUMBER GENERATED ALWAYS AS IDENTITY START WITH 10000 INCREMENT BY 1 CONSTRAINT owner\_owner\_id\_pk PRIMARY KEY ,

first\_name VARCHAR2(15) ,

last\_name VARCHAR2(15) CONSTRAINT owner\_last\_name\_nn NOT NULL ,

age NUMBER(2) ,

tel VARCHAR2(20) CONSTRAINT owner\_tel\_nn NOT NULL ,

email VARCHAR2(40) ,

street\_address VARCHAR2(25) CONSTRAINT owner\_street\_address\_nn NOT NULL ,

postcode VARCHAR2(8) CONSTRAINT owner\_postcode\_nn NOT NULL

);

CREATE TABLE pet (

pet\_id NUMBER GENERATED ALWAYS AS IDENTITY START WITH 1000 INCREMENT BY 1 CONSTRAINT pet\_id\_ck CHECK (pet\_id <=3000) CONSTRAINT pet\_pet\_id\_pk PRIMARY KEY ,

name VARCHAR2(25) CONSTRAINT pet\_name\_nn NOT NULL ,

type VARCHAR2(25) ,

gender CHAR(1) CONSTRAINT pet\_gender\_ck REFERENCES gender(m\_f) ,

date\_of\_birth DATE CONSTRAINT pet\_date\_of\_birth\_nn NOT NULL ,

date\_of\_pet\_entry DATE CONSTRAINT pet\_date\_of\_pet\_entry\_nn NOT NULL ,

colour VARCHAR2(40) ,

weight\_in\_kg NUMBER(3,1) ,

owner\_id NUMBER(5) CONSTRAINT pet\_owner\_id\_nn NOT NULL CONSTRAINT pet\_owner\_id\_fk REFERENCES owner(owner\_id) ,

CONSTRAINT pet\_age\_at\_entry\_ck CHECK (TRUNC((date\_of\_pet\_entry - date\_of\_birth)/365) BETWEEN 1 AND 12)

);

CREATE TABLE nurse (

nurse\_id NUMBER GENERATED ALWAYS AS IDENTITY START WITH 22000 INCREMENT BY 1 CONSTRAINT nurse\_nurse\_id\_pk PRIMARY KEY ,

first\_name VARCHAR2(15) ,

last\_name VARCHAR2(15) CONSTRAINT nurse\_last\_name\_nn NOT NULL ,

email VARCHAR2(40) CONSTRAINT nurse\_email\_nn NOT NULL CONSTRAINT nurse\_email\_uk UNIQUE ,

tel VARCHAR2(13) ,

is\_full\_time CHAR(1) CONSTRAINT nurse\_is\_full\_time\_fk REFERENCES yesno(y\_n)

);

CREATE TABLE doctor (

doctor\_id NUMBER GENERATED ALWAYS AS IDENTITY START WITH 2200 INCREMENT BY 1 CONSTRAINT doctor\_doctor\_id\_pk PRIMARY KEY ,

first\_name VARCHAR2(15) ,

last\_name VARCHAR2(15) CONSTRAINT doctor\_last\_name\_nn NOT NULL ,

office\_num VARCHAR2(2) ,

tel VARCHAR2(13) ,

email VARCHAR2(40) CONSTRAINT doctor\_email\_nn NOT NULL CONSTRAINT doctor\_email\_uk UNIQUE ,

is\_full\_time CHAR(1) CONSTRAINT doctor\_is\_full\_time\_fk REFERENCES yesno(y\_n)

);

CREATE TABLE pharmacy (

pharmacy\_id NUMBER GENERATED ALWAYS AS IDENTITY START WITH 6000 INCREMENT BY 1 CONSTRAINT pharmacy\_pharmacy\_id\_pk PRIMARY KEY ,

name VARCHAR2(25) CONSTRAINT pharmacy\_name\_nn NOT NULL ,

street\_address VARCHAR2(40)

);

CREATE TABLE drug (

drug\_id NUMBER GENERATED ALWAYS AS IDENTITY START WITH 5000 INCREMENT BY 1 CONSTRAINT drug\_drug\_id\_pk PRIMARY KEY ,

name VARCHAR2(25) CONSTRAINT drug\_name\_nn NOT NULL

);

CREATE TABLE appointment (

app\_id NUMBER(6) CONSTRAINT appointment\_app\_id\_pk PRIMARY KEY ,

app\_date DATE CONSTRAINT appointment\_app\_date\_nn NOT NULL ,

app\_timeslot DATE CONSTRAINT appointment\_app\_timeslot\_fk REFERENCES apptimeslots(avail\_time) CONSTRAINT appointment\_app\_timeslot\_nn NOT NULL ,

pet\_id NUMBER(4) CONSTRAINT appointment\_pet\_id\_fk REFERENCES pet(pet\_id) CONSTRAINT appointment\_pet\_id\_nn NOT NULL ,

pet\_date\_of\_birth DATE CONSTRAINT appointment\_pet\_dob\_nn NOT NULL ,

doctor\_id NUMBER(4) CONSTRAINT appointment\_doctor\_id\_fk REFERENCES doctor(doctor\_id) ,

is\_app\_cancelled Char(1) CONSTRAINT app\_app\_cancelled\_fk REFERENCES yesno(y\_n) ,

date\_app\_cancelled DATE ,

app\_fee NUMBER (6,2) CONSTRAINT app\_app\_fee\_fk REFERENCES fees(app\_fee) CONSTRAINT appointment\_app\_fee\_nn NOT NULL ,

is\_cancel\_fee\_paid Char(1) CONSTRAINT app\_cancel\_fee\_paid\_fk REFERENCES yesno(y\_n) ,

CONSTRAINT app\_cancelled\_date\_ck CHECK ((date\_app\_cancelled IS NOT NULL) OR (date\_app\_cancelled IS NULL AND is\_app\_cancelled <> 'Y')) ,

CONSTRAINT app\_cancel\_fee\_paid\_ck CHECK ((is\_cancel\_fee\_paid IS NOT NULL) OR (is\_cancel\_fee\_paid IS NULL AND is\_app\_cancelled <> 'Y' ) OR

(is\_cancel\_fee\_paid IS NULL AND is\_app\_cancelled = 'Y' AND(TRUNC(app\_date - date\_app\_cancelled) > 0))) ,

CONSTRAINT app\_pet\_age\_at\_app\_ck CHECK (TRUNC((app\_date - pet\_date\_of\_birth)/365) BETWEEN 1 AND 12),

CONSTRAINT app\_app\_fee\_ck CHECK (((is\_app\_cancelled = 'N' AND TRUNC((app\_date - pet\_date\_of\_birth)/365) BETWEEN 1 AND 4) AND app\_fee = 10) OR

((is\_app\_cancelled = 'N' AND TRUNC((app\_date - pet\_date\_of\_birth)/365) BETWEEN 5 AND 9) AND app\_fee = 15) OR

((is\_app\_cancelled = 'N' AND TRUNC((app\_date - pet\_date\_of\_birth)/365) BETWEEN 10 AND 12) AND app\_fee = 20) OR

(is\_app\_cancelled = 'Y' AND TRUNC(app\_date - date\_app\_cancelled) > 0 AND app\_fee = 0) OR

(is\_app\_cancelled = 'Y' AND TRUNC(app\_date - date\_app\_cancelled) = 0 AND app\_fee = 5)) ,

CONSTRAINT app\_dr\_doublebooked\_ck UNIQUE (doctor\_id, app\_date, app\_timeslot) ,

CONSTRAINT app\_pet\_doublebooked\_ck UNIQUE (pet\_id, app\_date, app\_timeslot)

);

CREATE TABLE diagnosis (

diag\_id NUMBER GENERATED ALWAYS AS IDENTITY START WITH 10000 INCREMENT BY 1 CONSTRAINT diagnosis\_diag\_id\_pk PRIMARY KEY ,

app\_id NUMBER(6) CONSTRAINT diagnosis\_app\_id\_nn NOT NULL CONSTRAINT diagnosis\_app\_id\_fk REFERENCES appointment(app\_id) ,

diag\_desc VARCHAR2(500)

);

CREATE TABLE nurse\_appointment (

nurse\_id NUMBER(6) CONSTRAINT nurse\_appointment\_nurse\_id\_nn NOT NULL CONSTRAINT nurse\_appointment\_nurse\_id\_fk REFERENCES nurse(nurse\_id) ,

app\_id NUMBER(6) CONSTRAINT nurse\_appointment\_app\_id\_nn NOT NULL CONSTRAINT nurse\_appointment\_app\_id\_fk REFERENCES appointment(app\_id) ,

app\_date DATE CONSTRAINT nurse\_app\_app\_date\_nn NOT NULL ,

app\_timeslot DATE CONSTRAINT nurse\_app\_app\_timeslot\_nn NOT NULL ,

CONSTRAINT nurse\_appointment\_pk PRIMARY KEY (nurse\_id, app\_id) ,

CONSTRAINT nurse\_app\_doublebooked\_ck UNIQUE (nurse\_id, app\_date, app\_timeslot)

);

CREATE TABLE prescription (

diag\_id NUMBER(5) CONSTRAINT pres\_diag\_id\_nn NOT NULL CONSTRAINT pres\_diag\_id\_fk REFERENCES diagnosis(diag\_id) ,

drug\_id NUMBER(4) CONSTRAINT pres\_drug\_id\_nn NOT NULL CONSTRAINT pres\_drug\_id\_fk REFERENCES drug(drug\_id) ,

pres\_dose NUMBER (6,2) ,

pres\_dose\_unit VARCHAR2(10) ,

pres\_amount NUMBER (3) ,

pres\_guidance VARCHAR2 (500) ,

CONSTRAINT pres\_pk PRIMARY KEY (diag\_id, drug\_id)

);

CREATE TABLE drug\_pharmacy (

drug\_id NUMBER(4) CONSTRAINT pharmacy\_drugs\_drug\_id\_nn NOT NULL CONSTRAINT pharmacy\_drugs\_drug\_id\_fk REFERENCES drug(drug\_id) ,

pharmacy\_id NUMBER(4) CONSTRAINT pharmacy\_drugs\_pharmacy\_id\_nn NOT NULL CONSTRAINT pharmacy\_drugs\_pharmacy\_id\_fk REFERENCES pharmacy(pharmacy\_id) ,

pharmacy\_available\_dose NUMBER(6,2) CONSTRAINT pharmacy\_available\_dose\_nn NOT NULL ,

pharmacy\_dose\_unit VARCHAR2(20) CONSTRAINT pharmacy\_dose\_unit\_nn NOT NULL ,

drug\_cost\_per\_dose\_gbp NUMBER(6,2) CONSTRAINT drug\_cost\_per\_dose\_gbp\_nn NOT NULL ,

CONSTRAINT pharmacy\_drugs\_pk PRIMARY KEY (drug\_id, pharmacy\_id, pharmacy\_available\_dose, pharmacy\_dose\_unit)

);

# ERD including suggested improvements

To return to improvement suggestions – [click here](#_Critical_design_analysis)

