FIT9132 Introduction Databases - Assignment 2

Creating, Populating and Manipulating Databases - Pets First (PF)

Purpose	Students will be asked to implement, via SQL, a small database in the Oracle RDBMS from a provided logical model case study, followed by the insert of appropriate data to the created tables. Once populated the database will be used to carry out specified DML commands and make specified changes to the database structure via SQL. Students will then use SQL and NoSQL to write queries to produce specified output. This task covers learning outcomes: 1. Apply the theories of the relational database model. 3. Implement a relational database based on a sound database design. 4. Manage data that meets user requirements, including queries and transactions. 5. Contrast the differences between non-relational database models and the relational database model.
Your task	This is an open book, **individual task**. The final output for this task will be a set of tables and data implemented in the Oracle RDBMS. In addition students will create a set of relational (Oracle) and non relational (MongoDB) queries which meet the user requirements.
Value	**40%** of your total marks for the unit
Due Date	Wednesday, 5th June 2024, 11:55 PM (note: staff support is unavailable after business hours)
Submission	● Via Moodle Assignment Submission ● FIT GitLab check ins will be used to assess history of development
Assessment Criteria	● Application of relational database principles. ● Handling of transactions and the setting of appropriate transaction boundaries. ● Application of SQL statements and constructs to create and alter tables including the required constraints and column comments, populate tables, modify existing data in tables, and modify the "live" database structure to meet the expressed requirements (including appropriate use of constraints). ● Application of SQL select statements to produce outputs that meet user requirements. ● Mapping of relational database data into non relational database data structure. ● Application of MongoDB operations to produce outputs that meet user requirements.

Late Penalties	• 10% deduction per calendar day or part thereof for up to one week • Submissions more than 7 calendar days after the due date will receive a mark of zero (0) and no assessment feedback will be provided.
Support Resources	See Moodle Assessment page
Feedback	Feedback will be provided on student work via: • general cohort performance • specific student feedback ten working days post submission • a sample solution

INSTRUCTIONS

Pets First has several clinics distributed across the suburbs. For each clinic, they record the clinic's ID (which is used to identify the clinic), the clinic's name, its address, and the clinic's contact phone number. The practice has several veterinary surgeons (vets) who work in these clinics. A vet is assigned to one clinic as their home (or base) clinic. A clinic must have at least one vet assigned to it as the vet's base clinic to function.

The details Pets First records about a vet are their vet ID (used to identify a vet), the vet's given name and family name, the vet's contact phone number, and the date they were first employed by Pets First.

Some Pets First vets are specialists in areas such as oncology, cardiology, dermatology, etc. If the vet is a specialist, their area of specialisation is also recorded as part of their vet details (specialist vets only specialise in one particular area). These specialist vets, as well as having a home clinic (their base clinic) where they accept general visits, also rove around all clinics when their specialist skills are required to treat an animal.

Each clinic appoints one of the vets based in the clinic (i.e. those vets assigned to the clinic as their home clinic) as the head vet for that clinic.

Pet owners, who are each assigned a unique owner ID, have their given name and family name recorded by the practice. The practice also records the owner's contact phone number. Each owner may have one or more pets; for each pet, the practice records a unique animal ID, the animal's name, the year the animal was born and the type of animal they are, for example, a cat. Each animal is only recorded as being owned by one owner. PF also records if an animal is currently alive or not.

When an animal needs veterinary attention (such as annual injections) or is ill, the owner books a visit with a vet. During the booking, the date and time of the visit, the visit length, the required service(s), the clinic, and the attending vet are recorded (this attending vet may be one based at this particular clinic, i.e., this is their home clinic, or the visit may need the services of a roving specialist vet).

On the scheduled visit date, the owner brings their pet to the clinic to be attended to by the vet. During the visit, the practice records the pet's weight for this visit and any notes that the vet needs to make about the pet's condition. If required, the vet also updates the length of the visit and the required service(s) based on the examination.

The owner may also visit one of the practice's clinics without making any booking, e.g., for emergency treatment.

Every visit must involve at least one service charge; however, there may be no drug charges. Drugs and services have a standard charge, which must be recorded in the database. A particular service will only be charged once for a visit. If necessary, the vet will adjust the cost charged to cover any "extra" work under this service or to give a discount. For example, using the invoice below - the standard service cost for a skin allergy treatment (S009) is $85, but here on the sample invoice, the vet has given a $5 discount and charged only $80.

During a visit, the attending vet may need to prescribe drugs for the animal. The practice identifies a drug by a unique drug ID and records the drug's name and usage instructions (for example, "Analgesic for post-surgery pain relief; 0.01 mg per kg"). When a drug is prescribed during a visit, the actual drug dose and frequency of administration are recorded. The line cost listed on an invoice is the total cost for items listed on that line; for example, in the invoice below, the $13.5 charge for corticosteroid is the total charge for the 3 items.

Pets First
Invoice for Professional Services

Patient

Animal ID:	1
Name:	Buddy

Attending Vet

Vet ID:	1002
Name:	Dr. Lucas Bennet

Owner

Owner ID:	1
Given name:	Olivia
Family Name:	Smith

Account for services provided on 16/01/2024 at 02:00 PM

Service Code	Service Description	Line Cost
S009	Skin Allergy Treatment	$80.00

Drug ID	Drug Name	Qty Supplied	Line Cost
105	Corticosteroids	3	$13.50

Total Amount Due	$93.50

Some visits require further follow-up visits, the system needs to record, for these subsequent visits, which visit generated the follow-up. A given visit may require multiple follow-up visits to address an issue identified in the initial diagnosis. For example, the pet may have an infected wound from a fight. The first visit (e.g. visit id 21) for this issue results in antibiotics being given and the wound being stitched. Follow-up visits may be required for example, to remove the stitches (e.g. visit id 23), check the wound healing (e.g. visit id 25), provide further antibiotic injections (e.g. visit id 27), etc. All of these follow-up visits are related to the original visit (visit id 21).

Based on these requirements, a data model has been created for PF:
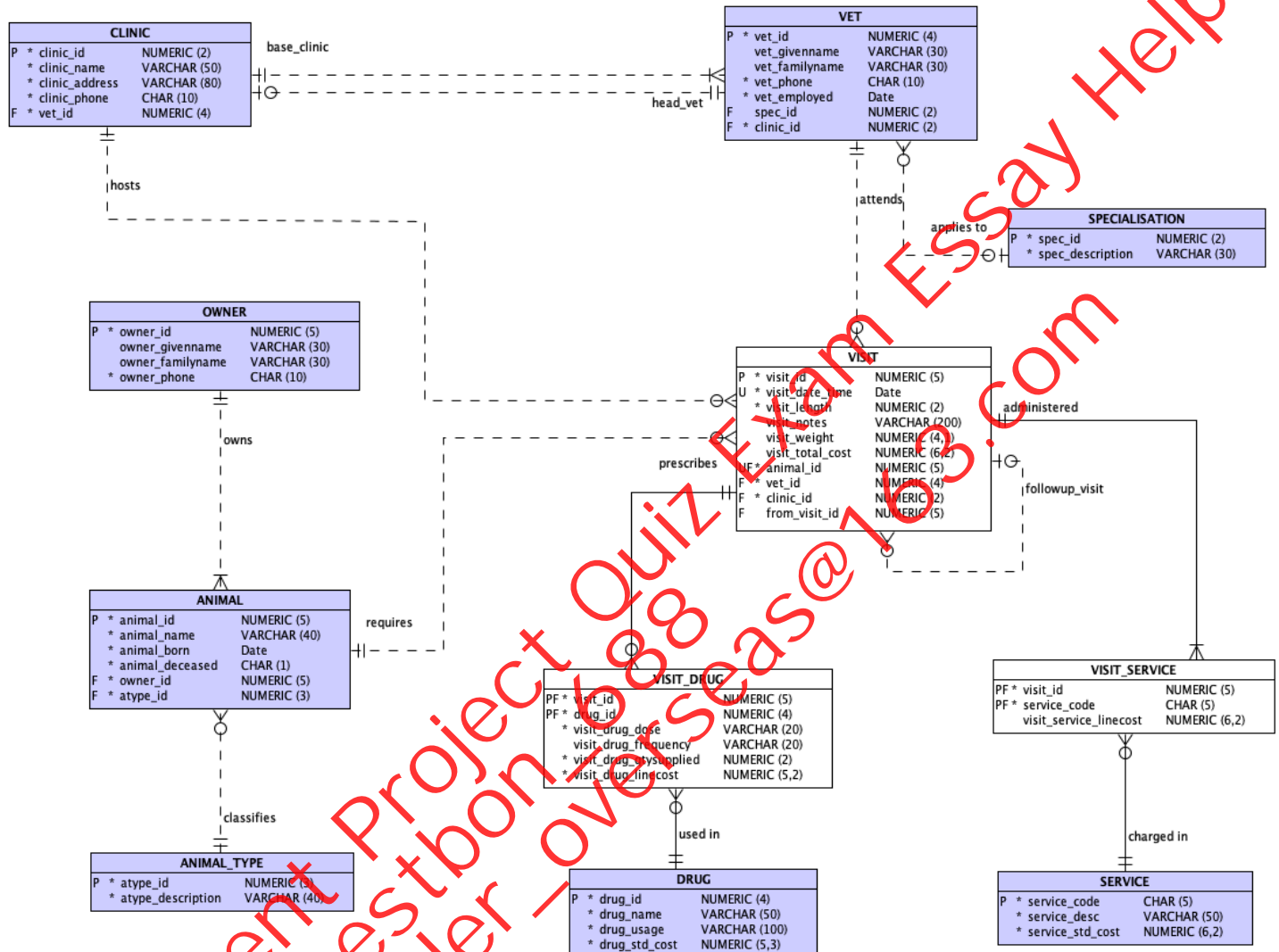
The schema/insert file for creating this model (pf_initialSchemaInsert.sql) is available in the archive ass2_student.zip. This file partially creates the Pets First tables and populates several of them (those shown in purple on the supplied model). You should read this schema carefully and be sure you understand the various data requirements.

IMPORTANT points for you to observe when completing this assignment are:

1. The ass2-student.zip archive also contains seven script files to code your answers in. **You MUST ensure these files are regularly pushed to the GitLab server <u>so that a clear development history is available</u> for the marker to verify your work (a minimum of fourteen pushes are required - 2 pushes per file).** In each file, you *must* fill in the header details with your name and student ID before beginning work. Your SQL script files **must not include any SPOOL or ECHO commands**. Although you might include such commands when testing your work, **they must be removed before submission** (a -10 mark grade penalty will be applied if your documents contain spool or echo commands).

2. You are free to make assumptions if needed. However, *your assumptions must align with the details here and in the Ed Assignment 2 forum* and must be clearly documented (see the required submission files).

   REMEMBER, you must keep up to date with the Ed Assignment 2 forum, where further clarifications may be posted (this forum is to be treated as your client). **Please ensure you do not post anything that includes your reasoning, logic or any part of your work to this assignment forum, as doing so violates Monash plagiarism/collusion rules.**

3. Views *must not* be used in arriving at any solutions for the tasks you must complete as part of this assessment.

4. When handling dates with SQL, the default date format must not be assumed; you must use *the TO_DATE and TO_CHAR functions where appropriate*.

5. **ANSI joins must be used** where the joining of tables is required.

6. In completing the following tasks, you must *design your test data so that you always get output for the queries specified below* - this may require you to add further data as you move through completing the required tasks. Such extra data MUST be added as part of Task 2 (i.e. as part of your test data load). *Queries that are correct but do not produce any output ("no rows selected" message) using your test data will lose 50% of the marks allocated*. So, you should carefully check your test data and ensure it thoroughly validates your SQL queries.

Steps for working on Assignment 2

1. Download the Assignment 2 Required Files zip archive (ass2-student.zip) from Moodle

2. Extract the zip archive and place the contained files in your local repository in the folder /Assignments/Ass2
   Do not add the zip archive to your local repo.

3. Examine the extracted files, i.e. read carefully through them and ensure you understand their content.

4. In each supplied script, fill in the header details with your name and student ID. Then, add, commit and push them to the FITGitLab server.

5. Run pf_initialSchemaInsert.sql from the supplied zip archive to set up the initial state of the database

6. Write your answer for each task in its respective file (e.g. write your answer for task 1 in T1-pf-schema.sql and so on).

7. Save, add, commit and push the file/s **regularly** while you are working on the assignment

8. Finally, when you have completed all tasks, separately run each SQL or Mongo *as a script (not as individual statements) and ensure there are no errors*. Upload all required files from your local repository to Moodle. Check that the files you have uploaded are the correct files (download them from Moodle into a temporary folder and check they are correct). After you are sure they are correct, submit your assignment.

****
For all assignment tasks, **your final script must run as a script without errors except for SQL errors generated by the DROP TABLE/DROP SEQUENCE statements**. **Any task's script that runs with an error will receive a maximum grade of half of the task's marks -1**. For example if your task 1 script runs with an error, regardless of the contained code, your maximum grade will be 15/2 => 7.5 - 1 = 6.5 marks. This will be applied even if the error is simply a forgotten semicolon. Thus, **please carefully check that your final scripts for all tasks run without error**.

In arriving at your solutions for assignment 2 you are ONLY permitted to use the SQL/NoSQL structures and syntax **which have been covered within this unit**:

- Topic 6 Workshop and Applied 7 - Creating & Populating the Database
- Topic 7 Workshop and Applied 8 - Insert, Update, Delete (DML) and Transaction Management
- Topic 8 Workshop and Applied 9 - SQL Part I - Basic
- Topic 9 Workshop and Applied 10 - SQL Part II- Intermediate
- Topic 10 Workshop and Applied 11 - SQL Part III - Advanced
- Topic 11 Workshop and Applied 12 - Non-Relational Database

As detailed above, SQL/NoSQL syntax and commands outside of the covered work will NOT be accepted/marked.

Views must not be used in completing these tasks.

You must also keep up to date with the Ed Assignment 2 forum where further clarifications may be posted. Please **ensure you do not publicly post anything that includes your reasoning, logic, or any part of your work to this forum**; doing *so violates Monash plagiarism/collusion rules* and has significant academic penalties. Attend a consultation session or use a private Ed post to raise such questions.

****

GIT STORAGE
Your work for these tasks MUST be saved in your <u>individual local working directory (repo)</u> in the Assignment 2 folder and **regularly pushed to the FIT GitLab server to build a clear history of the development of your approach**. A minimum of fourteen pushes to the FIT GitLab server is required (2 pushes per file). Please note that fourteen pushes are a minimum; we expect significantly more in practice. All commits must include a **meaningful commit message** that clearly describes what the particular commit is about and **must be correctly assigned to your valid GitLab author**.

You must regularly check that your pushes have been successful by logging in to the FIT GitLab server's web interface; you must not simply assume they are working. Before submission via Moodle, you must log in to the GitLab server's web interface and ensure your submission files are present and their names are unchanged.

****

Assignment Tasks

TASK 1: DDL [15 mks]

ENSURE your **ID and name are shown at the top of any file you submit**.
For this task, you must add to T1-pf-schema.sql the CREATE TABLE and CONSTRAINT definitions, which are missing from the supplied partial schema script, in the positions indicated by the script's comments.

The table below details the attributes' meaning in the missing two tables. You **MUST** use exactly the same relation and attribute names shown in the data model above to name the tables and attributes you add. The attributes must be in the same order as shown in the model. These new DDL commands *must be hand-coded, not generated in any manner (generated code will not be marked)*.

Table name	Attribute name	Meaning
VISIT		
	visit_id	Identifier for visit (added surrogate key)
	visit_date_time	Date and time of visit
	visit_length	Visit length in minutes (must be between 30 - 90 minutes, inclusive)
	visit_notes	Vet notes from visit
	visit_weight	Weight in Kgs
	visit_total_cost	Total cost for this visit
	animal_id	Animal identifier
	vet_id	Identifier for the vet
	clinic_id	Identifier for the clinic
	from_visit_id	The previous visit's identifier (for follow-up visit only)
VISIT_DRUG		
	visit_id	Identifier for visit (added surrogate key)
	drug_id	Drug identifier
	visit_drug_dose	Dose prescribed in this visit
	visit_drug_frequency	Frequency prescribed for this drug for this visit
	visit_drug_qtysupplied	Quantity of drug supplied
	visit_drug_linecost	Cost charged for drug in this visit
VISIT_SERVICE		
	visit_id	Identifier for visit (added surrogate key)
	service_code	Service identifier
	visit_service_linecost	Cost charged for this service in this visit

To test your code, you must first run the provided script pf_initialSchemaInsert.sql to create the other required tables. pf_initialSchemaInsert.sql contains the drop commands for all tables in this model at the head of the file. If you have problems with task 1 simply rerun pf_initialSchemaInsert.sql, which will cause all tables to be dropped and correct the issues in your script. **DO NOT add drop table statements to T1-pf-schema.sql**

TASK 2: INSERT [20 mks]

Before proceeding with Task 2, you must ensure you have run the file pf_initialSchemaInsert.sql (which **must not be edited in any way**) followed by the extra definitions that you added in Task 1 above (T1-pf-schema.sql).

Load the VISIT, VISIT_DRUG and VISIT_SERVICE tables with **your own test data** using the supplied **T2-pf-insert.sql** script file. Write SQL commands that will insert *as a minimum, (i.e. you may and should insert more)* the following sample data:

- (i) 10 VISIT entries
  - Include at least 5 different animals
  - Include at least 3 different vets
  - Include at least 3 different clinics
  - Include at least 2 follow-up visits
  - Include at least 8 completed visits and 2 incomplete visits (i.e. future visit bookings)
- (ii) 15 VISIT_SERVICE entries
  - Include at least 4 visits that require more than one service in a single visit
- (iii) 10 VIST_DRUG entries
  - Include at least 2 visits with more than one drug prescribed in a single visit

In adding this data, you must ensure that the test data thoroughly tests the model as supplied, to ensure your schema is correct (you are not required to submit or code fail tests, all insert statements must execute correctly).

Your inserted data must conform to the following rules:
1. You may treat all the data you add as a single transaction since you are setting up the initial test state for the database.
2. The primary key values for this data should be hardcoded values (i.e. **NOT** make use of sequences) and must consist of values below 100.
3. Dates used must be chosen between the 1st April 2024 and 30th June 2024.
4. The data added must be sensible (e.g., the follow-up visit date time should be after the previous visit's date time, the drug line cost must be correctly calculated, etc.).

For this task **ONLY,** Task 2, you may manually look up and include values for the loaded tables/data directly where required. However, you can still use SQL to get any non-key values if you wish.

In carrying out Task 2, you must not modify any data or add any further data to the tables populated by the pf_initialSchemaInsert.sql script. Design your test data so that you get output for the SQL scripts/queries specified below - this may require you to add further data as you complete the required tasks.

TASK 3: DML [20 mks]

Your answers for this task must be placed in the SQL file T3-pf-dml.sql

For this and *all subsequent Tasks*, **you are NOT permitted to**:

● manually lookup a value in the database, obtain its primary key, or manually obtain the highest/lowest value in a column,
● manually calculate values external to the database, e.g. on a calculator and then use such values in your answers. *Any necessary calculations must be carried out as part of your SQL code* or
● assume any particular contents in the database - rows in a table are potentially in a constant state of change

Your answers must recognise that you are dealing with only a very small sample snapshot of a multiuser database; as such, you must operate on the basis that there will be *more data in all of the database tables than you currently have access to. Thus, data will be in a constant state of change. Your answers must work regardless of the extra quantity of this extra "real" data and the fact that multiple users will operate in the tables simultaneously. You must consider this aspect when writing SQL statements.*

For any following SQL tasks, **your SQL must correctly manage transactions and use sequences to generate new primary keys for numeric primary key values** (under no circumstances may a new primary key value be hardcoded as a number or value).

You must ONLY use the data as provided in the text of the questions.

For Task 3, you must complete the following sub-tasks in the same order they are listed. Where you have been supplied with a string in italics, such as *Bayside Veterinary Clinic*, you may search in the database using the string *as listed*. Where a particular case for a word is provided, *you must only use that case (same spacing, case, etc) in your SQL code*. When a name is supplied, you may break the name into first name and last name. For example, *Olivia SMITH* can be split into Olivia and SMITH; again, note that the case must be maintained as it was supplied. **Failure to adhere to these requirements, such as changing the case of a provided string, will result in a grade penalty**:

(a) An Oracle sequence will be implemented in the database to insert records for the VISIT table.

Provide the CREATE SEQUENCE statement to create a sequence that could be used to provide primary key values for the VISIT tables. This sequences must start at 100 and increment by 10. Immediately before the create sequence command, place an appropriate DROP SEQUENCE command so that the sequence wil be dropped before being created, if it exists. Please note that this is the ONLY sequence that can be introduced and used in Task 3.

[1 mark]

Questions 3b, 3c, 3d and 3e are related questions. You can use the information below as needed in any part of Task 3.

(b) *Jack JONES*, a current customer, called Pets First to book a visit for *Oreo*, his rabbit which was born on 01 Jun 2018. The visit is scheduled for 19 May 2024 at 2 PM for 30 minutes. He indicated that the visit is for general consultation (service code: *S001*) with Dr. *Anna KOWALSKI* at Bayside Veterinary Clinic (clinic id: 3). You may assume that there is only one customer with such a name and Jack only has one rabbit born on such a date. You may also assume that there is only one vet with such a name in the system. Take the necessary steps in the database to record the required entries for this visit booking.

[5 marks]

(c) On 19 May 2024, Jack brought Oreo to the clinic as scheduled. Dr. Anna KOWALSKI examined Oreo and found an ear infection. She decided to record an *ear infection treatment* service only and charge Jack the standard cost for this treatment. She gave (and charged) 1 bottle of *Clotrimazole* to Jack. You may assume that there is only one service named *ear infection treatment* and one drug named *Clotrimazole* in the system.

Dr. KOWALSKI also scheduled a follow-up visit for Oreo at 2 PM seven days after this visit for another *ear infection treatment* at the same clinic.

Make these required changes to the database data. You may make up (invent) any other required information when making these changes.

[10 marks]

(d) On 21 May 2024, Jack called PF and cancelled Oreo's 19 May follow-up visit since he has to go overseas for a work emergency.

Make these required changes to the data in the database.

[4 marks]

TASK 4: DATABASE MODIFICATIONS (13 marks):

Your answers for these tasks (Task 4) must be placed in the supplied SQL script
T4-pf-mods.sql

**The required changes must be made to the "live" database (the database *after* you have
completed tasks 1, 2 and 3). You MUST not** edit and execute your schema file again. Before
carrying out the work below, please ensure that you have completed tasks 1, 2 and 3 above.

If, in answering these questions, you need to create a table, please place a drop table
statement immediately before your create table statement.

(a) PF would like to determine the total number of times each service was provided in a visit
where the charge made was lower or higher than the standard cost (i.e., not charged at
the standard cost). Add a new attribute that will record this requirement. The default
value for this new attribute is 0.

Based on the data *currently* stored in the system, this attribute must be initialised to the
number of times each service was provided in a visit and not charged at its standard
cost.

As part of your solution, provide appropriate select and desc statements to show the
changes you have made. Select to show any data changes that have occurred, and desc
tablename, e.g., desc customer, to show any table structural changes.

[5 marks]

(b) PF would like to allow the pet's owner to pay for the visit's cost in instalments, i.e. the
pet's owner can pay each visit's cost via multiple payments across several dates and
times. PF also wants to record the payment method for each payment. The owner can
pay the cost/part of the cost by *Card* or *Cash*. PF plans to include other payment
methods in the future.

Change the database structure to support these new business rules. All currently
completed visits in the system must be recorded as being fully paid on their visit date.
The payment method for these visits must be recorded as *Historical*.

As part of your solution provide appropriate select and desc statements to show the
changes you have made. Select to show any data changes that have occurred and desc
tablename e.g. `desc customer` to show any table structural changes.

[8 marks]

TASK 5: SQL Queries [15 mks]

Your answers for this task (Task 5) must be placed in the supplied SQL script **T5-pf-select.sql**.

You are only permitted to code **a single select statement for each question** below. Note that an SQL select statement begins with the SELECT keyword and ends with a semicolon (;) - within this statement, the SELECT keyword can be used multiple times.

Where you have been supplied with a string in *italics*, you must search the database using the provided string exactly as supplied. Where you need to show a full name, you must not have any extra spaces (e.g. leading space, extra space in the middle of the name).

Please remember VIEWS and/or PL/SQL (including anonymous blocks BEGIN…END) **must not be used**.

Sample output showing the form of what you must produce is provided. Note that this is the form of the output ONLY, i.e., the appearance; the data you return will clearly be different.

(a) Pets First would like to create a report showing the number of completed appointments involving a *dog* or a *cat* for each clinic by the servicing vet. The output must show, for a single line of output, the clinic ID, the vet ID, and the vet name. If the vet is the head of the clinic, "HEAD VET" must be shown. The number of dog and/or cat completed appointments must be shown for this vet and the percentage of this clinic's total dog and cat appointments.

The report must be in clinic order. Within a clinic, the head vet must appear first if they have treated any dogs or cats. The remaining vets who have worked in the clinic must be in vet ID. For a given vet, their output must be in animal type order.

Sample output is shown below (your output must have the same format):

```
CLINICID VETID VETNAME         ISHEAD    ANIMALTYPE NUMBERAPPTS PERCENTAPPTS
-------- ----- --------------- --------- ---------- ----------- ------------
       1  1001 Emily Tanner    HEAD VET  Dog                  2        50.0%
       1  1002 Lucas Bennet              Dog                  1        25.0%
       1  1003 John Marston              Dog                  1        25.0%
       2  1005 Owen Murphy               Dog                  1       100.0%
       3  1004 Anna Kowalski   HEAD VET  Cat                  1        33.3%
       3  1008 Clara Watson              Cat                  2        66.7%
       4  1006 Sophie Grant              Dog                  1        50.0%
       4  1011 Liam Foster               Dog                  1        50.0%
       5  1007 Jessica Lee     HEAD VET  Dog                  2        40.0%
       5  1003 John Marston              Dog                  1        20.0%
       5  1009 Sarah Morris              Cat                  1        20.0%
       5  1010 Michael Clarkson          Dog                  1        20.0%
```

[7 marks]

(b) Pets First wishes to analyse visit service charges made by a vet during a visit. They would like to determine, for a particular service code, which vets charged more than the average charged for that service code across all visits.

For services where the vet charged more than the average charge, the output should show the service code, service description, the vet ID, the vet given and family names, the clinic ID and the clinic name, the visit ID, the visit date and time and the actual charge the vet made.

SERVICE_CODE	SERVICE_DESC	VET_ID	VET_GIVENNAME	VET_FAMILYNAME	CLINIC_ID	CLINIC_NAME	VISIT_ID	VISITDATETIME	VISIT_SERVICE_LINECOST
S001	General Consultation	1005	Owen	Murphy	2	Northern Suburbs Animal Hospital	7	23-Apr-2024 09:00 AM	70
S002	Vaccination - Routine	1001	Emily	Tanner	1	East Melbourne Veterinary Center	1	10-Apr-2024 10:15 AM	45
S002	Vaccination - Routine	1005	Owen	Murphy	2	Northern Suburbs Animal Hospital	6	22-Apr-2024 02:00 PM	45
S004	Emergency Care	1003	John	Marston	1	East Melbourne Veterinary Center	5	22-Apr-2024 09:30 AM	175
S007	Heartworm Prevention	1005	Owen	Murphy	2	Northern Suburbs Animal Hospital	12	29-Apr-2024 11:15 AM	50
S010	Ear Infection Treatment	1004	Anna	Kowalski	3	Bayside Veterinary Clinic	100	19-May-2024 02:00 PM	75

[8 marks]

TASK 6: MongoDB [12 mks]

Your answers for this task (Task 6) must be placed in the supplied sql file **T6-pf-json.sql** and the supplied MongoDB script file **T6-pf-mongo.mongodb.js**.

You must not add any further comments to the supplied MongoDB script file, nor remove/rename any comments indicated by //

(a) Write an SQL statement in **T6-pf-json.sql** to generate a collection of JSON documents using the following structure/format. Each document in the collection represents a clinic, their head vet details and a list of all vets who are based on this clinic. Note that the _id in this structure is the clinic_id, the no_of_vets is the number of vets who are based on this clinic. If a vet does not have a specialisation, then it is listed as "N/A".

```
{
    "_id": 5,
    "name": "Brighton East Pet Care",
    "address": "123 Thomas St, Brighton East VIC 3187",
    "phone": "0398765412",
    "head_vet": {
        "id": 1007,
        "name": "Jessica Lee"
    },
    "no_of_vets": 3,
    "vets": [
        {
            "id": 1007,
            "name": "Jessica Lee",
            "specialisation": "Behavioral Medicine"
        },
        {
            "id": 1010,
            "name": "Michael Clarkson",
            "specialisation": "Cardiology"
        },
        {
            "id": 1009,
            "name": "Sarah Morris",
            "specialisation": "N/A"
        }
    ]
}
```

[7 marks]

Write the MongoDB commands for the following questions, 6(b) - 6(d), in the supplied MongoDB script file named **T6-pf-mongo.mongodb.js**.

(b) Create a new collection and insert all documents generated in 6(a) above into MongoDB. Provide a drop collection statement right above the create collection statement. You may pick any collection name. After the documents have been inserted, use an appropriate db.find command to list all the documents you added.

[1 mark]

(c) List the name and address of all clinics which have at least one vet who is specialising in *Dermatology* or *Cardiology*.

[2 marks]

(d) A new vet who is specialised in *Dentistry* named *Sarah Wilkinson* (vet id 1020) joins Pets First. Dr. Wilkinson's home base clinic is the Brighton East Pet Care ("_id":5). Due to her extensive experience, she is appointed as the current head vet of her home base clinic.

Use an appropriate db.find command before making the change so that you illustrate which document/s will be changed.

Write the necessary MongoDB commands to add this new vet into the collection.

Use an appropriate db.find command after making the change so that you illustrate/confirm the change which was made.

[2 marks]

Submission Requirements

Due Date: **Wednesday, 5th June 2024 at 11:55 PM**

*Please note, if you need to resubmit, you **cannot** depend on your staffs' availability, for this reason, please be VERY CAREFUL with your submission*. **It is strongly recommended that you submit several hours before this time to avoid such issues.**

For this assignment, there are seven files you are **required** to submit to Moodle:

- T1-pf-schema.sql
- T2-pf-insert.sql
- T3-pf-dml.sql
- T4-pf-mods.sql
- T5-pf-select.sql
- T6-pf-json.sql
- T6-pf-mongo.mongodb.js

If you need to make any comments to your marker/tutor please place them at the head of each of your solution scripts/answers in the "Comments for your marker" section.

Do not zip these files into one zip archive, submit seven independent SQL scripts. The individual files must also have been pushed to the FIT GitLab server with an appropriate history as you developed your solutions.

Late submission will incur penalties at the rate of -10 marks for every 24 hours the submission is late.

The seven files must also exist in your FIT GitLab server repo and *show a clear history of development* (a **minimum** of two pushes per file).

Please note we **cannot mark any work on the GitLab Server**, you need to ensure that you submit correctly via Moodle since it is only in this process that you complete the required student declaration without which work **cannot be assessed**.

It is your responsibility to ENSURE that the files you submit are the correct files - we strongly recommend after uploading a submission, and prior to actually submitting, that you download the submission and double-check its contents.

Your assignment **MUST** show a status of "Submitted for grading" before it will be marked.

Submission status

Attempt number	This is attempt 1.
Submission status	Submitted for grading
Grading status	Not graded

If your submission shows a status of "Draft (not submitted)" it will not be assessed and **will incur late penalties after the due date/time**.

Please *carefully* read the documentation under the "Assignment Submission" on the Moodle Assessments page which covers things such as extensions and resubmission.

Marking Guide

Submitted code will be assessed against an optimal solution for these tasks. In some tasks where SQL is involved there are often several alternative approaches possible, such alternatives will be graded based on the code successfully meeting the briefs requirements. If it does, the answer will be accepted and graded appropriately.

Marking Criteria	Items Assessed
TASK 1 DDL 15 marks	
DDL Creation of tables	**Maximum 7 marks - Create table:** ● Marks awarded for correct table DDL ● Marks awarded for correct attributes/data types ● Marks awarded for correct PK definition ● Mark penalty applied if different table/attribute names used than expressed in the supplied data model ● Mark penalty applied if different order of attributes used than expressed in the supplied data model ● No marks awarded if generated schema used
DDL implementation of non-PK database constraints	**Maximum 8 marks - Non-PK Constraints:** ● Marks awarded for correct implementation of non-PK constraints ● Marks awarded for correct use of column comments
TASK 2 Data Insert 20 marks	
Insert of required items test data	**Maximum 10 marks- Insert of data:** ● Marks awarded for correct insert of required data ● Marks awarded for correct management of transactions
Insert of valid test data	**Maximum 10 marks - Valid data inserted:** ● Marks awarded for validity of data inserted ○ meets the requirements expressed in the assignment brief ● Marks awarded for correct management of dates when inserting
Task 3 DML 20 marks	
	Maximum 20 marks - Satisfy brief requirements: ● Marks awarded (a) - (d) for SQL code which meets the expressed requirement ● Mark penalty applied if commit not used appropriately ● Mark penalty applied if date handling and string database lookups not managed correctly

Task 4 Database Modifications 13 marks

	Maximum 13 marks - Satisfy brief requirements: ● Marks awarded (a) - (b) for SQL code which meets the expressed requirement (including appropriate use of constraints). In making these modifications there must be no loss of existing data or data integrity within the database. ● Mark penalty applied if commit not used appropriately ● Mark penalty applied if column comments not used where required

Task 5 SQL Queries 15 marks

	Maximum 15 marks - Satisfy brief requirements: ● Marks awarded, as listed, (a) - (b) for SQL code which meets the expressed requirement ● Mark penalty applied if output does not match the form supplied for each question ● Mark penalty applied if date handling and string database lookups are not managed correctly ● Mark penalty applied if column aliases are not used when arithmetic calculation, concatenation, functions, or other output manipulation is used in a query ● Mark penalty applied if manual lookup/calculation is used for values from the database ● Statements which do not execute correctly in Oracle will be awarded a maximum of 50% of the available marks less 1 mark. For example, if a question is worth 6 marks and runs with an error in SQL the *maximum* mark awarded will be 2 marks

Task 6 Non Relational Database Queries - MongoDB 12 marks

	Maximum 12 marks - Satisfy brief requirements: ● Maximum of 7 marks awarded for creation of a JSON document which matches the supplied document format ● Marks awarded, as listed, (b) - (d) for MongoDB code which meets the expressed requirement ● Mark penalty applied if field names and predicates (such as "&eq") are not enclosed in double quotes ● Statements which do not execute correctly in MongoDB will be awarded a maximum of 50% of the available marks less 1 mark. For example, if a question is worth 6 marks and runs with an error in MongoDB the *maximum* mark awarded will be 2 marks

Correct use of Git 5 marks	
	• Marks awarded for a minimum of fourteen pushes (two per file) showing a clear development history of the work for Assignment 2 • Marks awarded for correct Git author details used in pushes • Marks awarded for the use of meaningful commit messages (i.e. not blank or of the form "Push1")
Penalties	
Use of • VIEWs • SET ECHO or SPOOL commands, and/or • PL/SQL	Use of VIEWS, inclusion of SET ECHO/SPOOL, and/or PL/SQL commands will result in a ***grade deduction of 10 marks being applied.***
Late submission	**-10 marks for each 24 hours late or part thereof**

Final Assignment Mark Calculation

Total: 100 marks, recorded as a grade out of 40