­­­­

2024 Year 12 Comp Sci

Database and Programming Project

Great southern grammar

Task 6

Alec McDonald

# Table Of Contents

[Table Of Contents 1](#_Toc172034353)

[Part 1 – Planning 2](#_Toc172034354)

[Tasks to be done part 1 Investigate. 2](#_Toc172034355)

[Tasks to be done part 1 Design. 2](#_Toc172034356)

[Tasks to be done part 2 Develop. 2](#_Toc172034357)

[Tasks to be done part 2 Evaluate. 2](#_Toc172034358)

[Time frame 3](#_Toc172034359)

[Problem Outline: 3](#_Toc172034360)

[Problem Description: 3](#_Toc172034361)

[Sample Data: 4](#_Toc172034362)

[Part 1 Design 5](#_Toc172034363)

[Normalised Data: 5](#_Toc172034364)

[Data Dictionary: 6](#_Toc172034365)

[ER Diagram: 7](#_Toc172034366)

[Part 2 Develop 8](#_Toc172034367)

[SQL Queries Used to Create Database: 8](#_Toc172034368)

[SQL Queries Used to Insert Data into the Database: 11](#_Toc172034369)

[Part 2 Manipulate 15](#_Toc172034370)

[SQL Queries Used to Manipulate Data: 15](#_Toc172034371)

# Part 1 – Planning

## Tasks to be done part 1 Investigate.

* Break down tasks to do.
* Outline problem.
* Problem Description.

## Tasks to be done part 1 Design.

* ER Diagram
* Relational Notation
* Data Dictionary
* Describe several queries.

## Tasks to be done part 2 Develop.

* Create an empty database.
* Create a script to insert data.
* Data validation
* Create several different queries.
* Create front end.

## Tasks to be done part 2 Evaluate.

* Reflects on success of your solution.
* Compare ER Diagram to database.
* Extra features implemented.
* Documentation of any known bugs or limitations
* Perform a developer retrospect.
* Document sources used to get information.

## Time frame

I have 5 weeks to complete this project.

Starting week 1 term 3 and to be completed by week 5 term 3.

|  |  |  |  |  |  |
| --- | --- | --- | --- | --- | --- |
|  | = Not Started |  | = Doing |  | = Finished |

|  |  |  |  |  |
| --- | --- | --- | --- | --- |
| Part | Key Point | Item | Due Date | Status |
| 1 | Investigate | Project Breakdown | Week 1 |  |
| Problem Outline | Week 1 |  |
| Problem Description | Week 1 |  |
| Design | ER Diagram | Week 2 |  |
| Relational Notation | Week 2 |  |
| Data Dictionary | Week 2 |  |
| Describe several queries | Week 2 |  |
| 2 | Develop | Create an empty database. | Week 3 |  |
| Create a script to insert data. | Week 3 |  |
| Data validation | Week 3 |  |
| Create several different queries. | Week 4 |  |
| Create front end. | Week 4 |  |
| Evaluate | Reflects on success of your solution | Week 5 |  |
| Compare ER Diagram to database. | Week 5 |  |
| Extra features implemented. | Week 5 |  |
| Documentation of any known bugs or limitations | Week 5 |  |
| Perform a developer retrospect. | Week 5 |  |
| Document sources used to get information. | Week 5 |  |

## Problem Outline:

The purpose of this project is to create a database and interactive terminal, or GUI made from python for the Great Southern Grammar catering team. Their current method of dealing with catering requests is through an online form which contains a series of checkboxes and user input boxes. This form does not reference any data sources to provide accurate or timely information to the users of the form. I aim to create a well organised and functional database to store information like requesters first and last name, email address, function name, function date, catering charges, type of catering, location of function, number of people catering for, time catering is requested for, what meal is being catered for, the costs of the meal that is being requested, a menu, specific catering requests, special dietary requirements, and an urgent request notice.

## Problem Description:

The database will need to consist of multiple tables such as requester information, location information, meal item (breakfast, morning tea, lunch, afternoon tea, dinner, other), event table, a catering table, a catering charges table and a menu table. To address the requirements of the client to have a space to store important information on the requester and event in a safe and easily retrievable way the database also includes a table that’s have meal items and their prices so that users can easily retrieve the cost of the meals being catered for. The database will also include the location table as it is important that the caterers have easy access to find out the location that they need to bring their services to. The Database would need bank details so the catering team can bill the expenses to the right people. This would need to be stored in a safe place as it is personal information like the requesters name and contact details.

Users might want to retrieve location information so that the caters know where they need to go, users might want to retrieve menu items so they know what they need to prepare for the function, they might want to be able to retrieve the function time and how many people are being catered for. These are all very important details that the catering team would be able to retrieve with relative ease. The catering team might want to retrieve the contact information of the requester as they might encounter a problem and need to contact the requester to clear up some details.

To achieve all this, I will be using SQLite and the SQL language to create a relational database. Using SQL, I will create around 7 tables to store the required data in order to meet the client’s requests. SQL queries will be used to make sure the database is functioning and in working order to be handed over to the client. The database will be integrated with a basic python interface that will easily allow the catering team to access all the information they need to provide their services to the customers functions.

Describing Queries

There are multiple queries that users will need to use such as select and inner join SQL queries. This will be used to retrieve information like retrieving the event details and requester name and contact details and showing them together. You could also retrieve the dietary requirement that need to be retrieved by selecting the dietary requirements for the event. The users can also retrieve the menu for the event so they can see what and how many meals they would specifically need to prepare.

Sample Data:  
Sample Data Can be found in the excel spreadsheet Linked bellow.

[Click me to see sample data](https://greatsoutherngrammar-my.sharepoint.com/personal/alec_mcdonald_student_gsg_wa_edu_au/Documents/.Year%2011/Computer%20Science/Task%206/Sample%20data.xlsx)

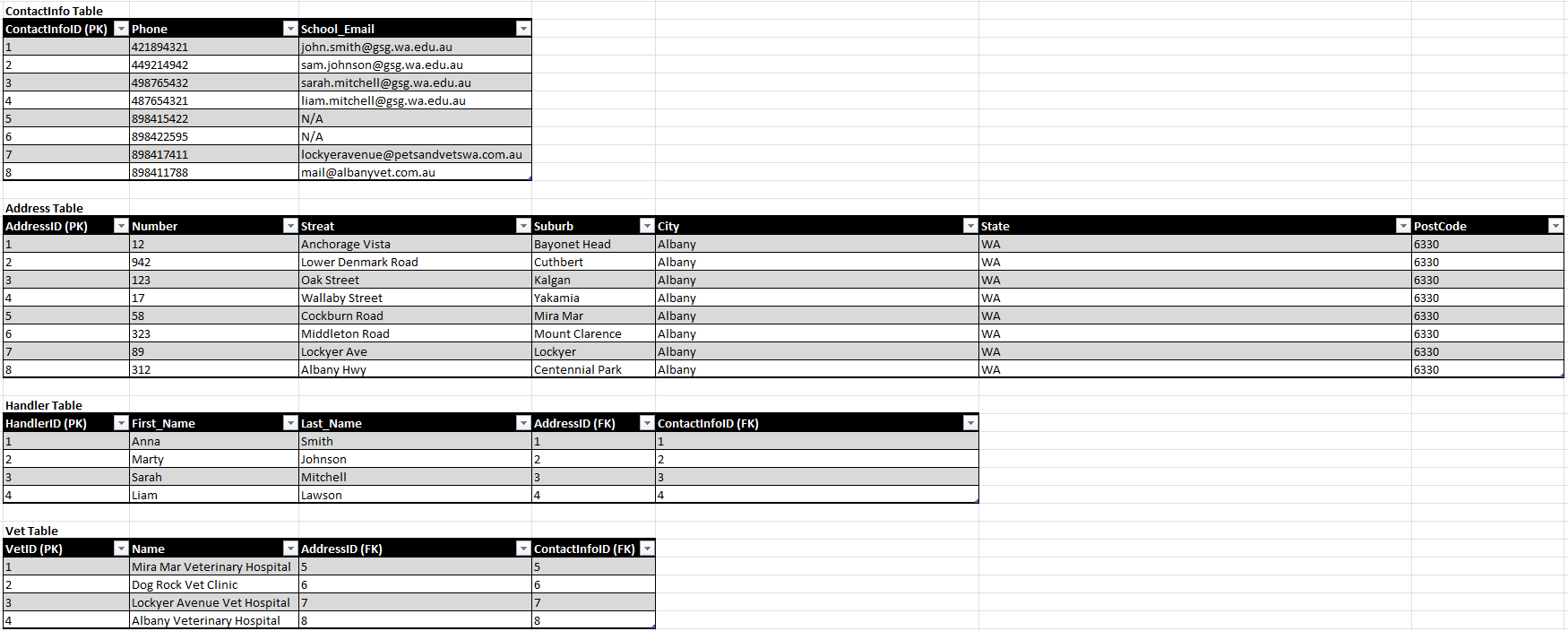
# Part 1 Design

## Normalised Data:

The normalised data can be found in the excel spreadsheet bellow.

[Click me to see normalised data](https://greatsoutherngrammar-my.sharepoint.com/personal/alec_mcdonald_student_gsg_wa_edu_au/Documents/.Year%2011/Computer%20Science/Task%206/Sample%20data.xlsx)

Alternatively, can be viewed in the photo bellow:



A screenshot of a computer

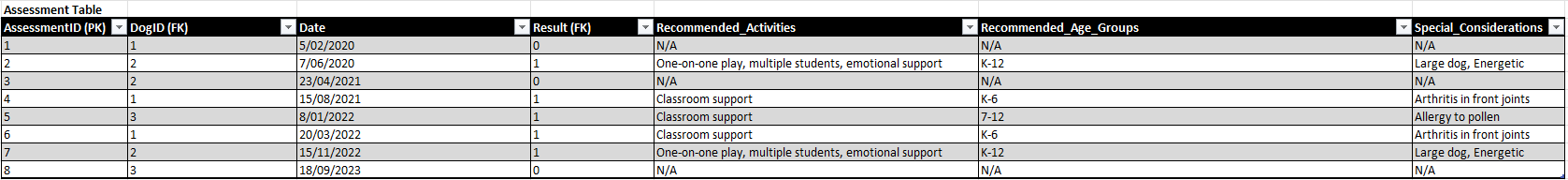
Description automatically generated

A screenshot of a computer

Description automatically generated

A screenshot of a computer

Description automatically generated

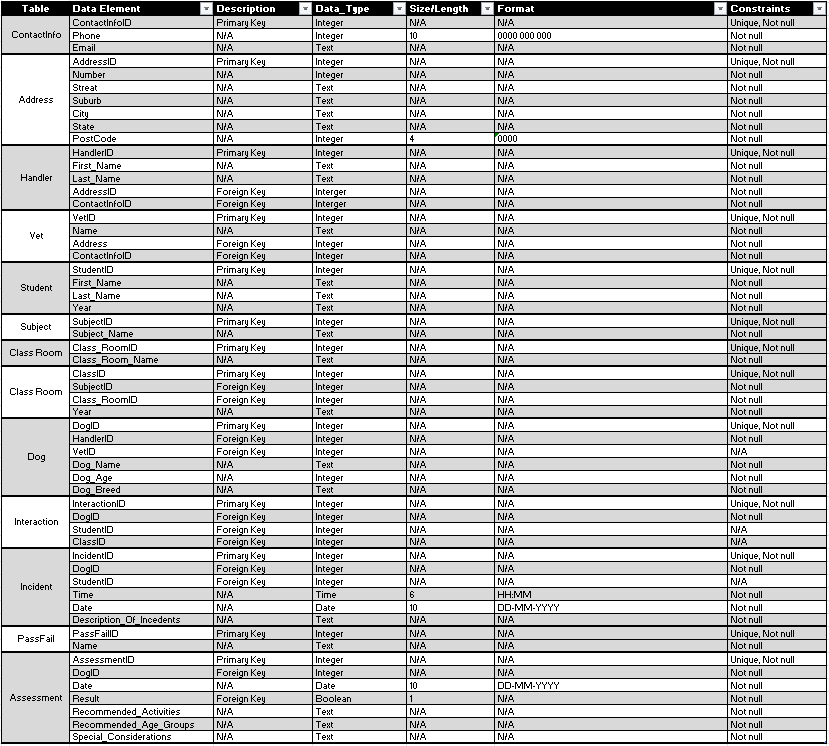


## Data Dictionary:

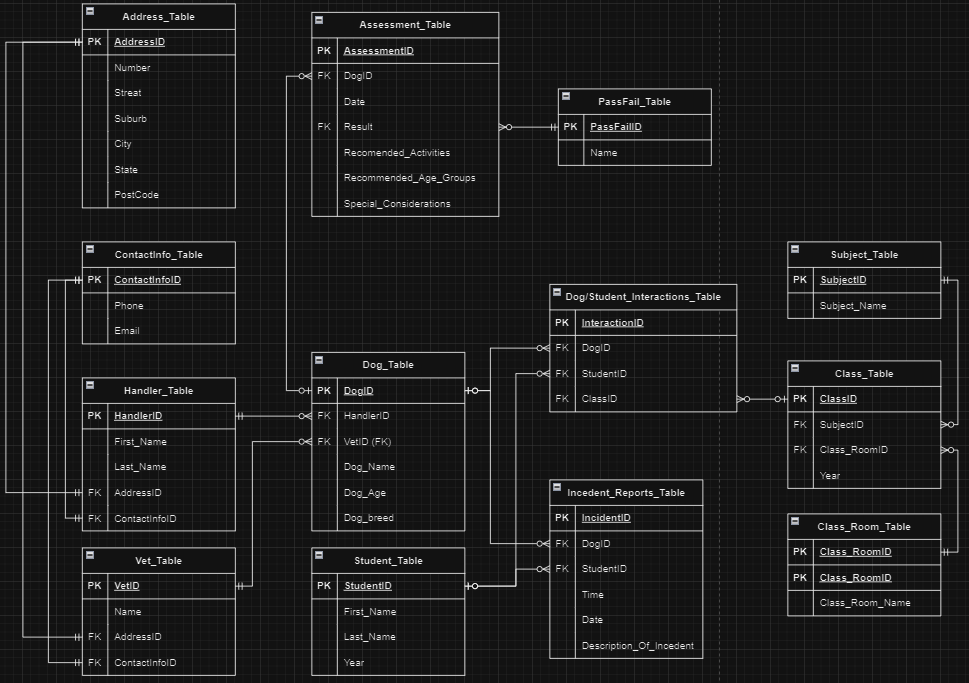
The normalised data can be found in the excel spreadsheet bellow.

[Click me to view the data dictionary](https://greatsoutherngrammar-my.sharepoint.com/personal/alec_mcdonald_student_gsg_wa_edu_au/Documents/.Year%2011/Computer%20Science/Task%206/Sample%20data.xlsx)

Alternatively, can be viewed in the photo bellow:



## ER Diagram:



# Part 2 Develop

## SQL Queries Used to Create Database:

**CREATE TABLE "Menu" (**

**MenuID integer PRIMARY KEY AUTOINCREMENT NOT NULL,**

**ItemName text NOT NULL);**

**CREATE TABLE "Event" (**

**EventID integer PRIMARY KEY AUTOINCREMENT NOT NULL,**

**EventName text NOT NULL,**

**NumberOfPeople text NOT NULL,**

**"Date" date NOT NULL CHECK(LENGTH(Date) == 10),**

**"Time" time NOT NULL CHECK(LENGTH(Time) == 5));**

**CREATE TABLE "CardDetails" (**

**CardID integer PRIMARY KEY AUTOINCREMENT NOT NULL,**

**CardName text NOT NULL,**

**CardNumber integer NOT NULL CHECK(LENGTH(CardNumber) == 16),**

**ExpiryDate integer NOT NULL CHECK(LENGTH(ExpiryDate) == 4),**

**CVV integer NOT NULL CHECK(LENGTH(CVV) == 3));**

**CREATE TABLE "TypesOfCatering" (**

**TypesOfCateringID integer PRIMARY KEY AUTOINCREMENT NOT NULL,**

**TypesOfCateringName text NOT NULL);**

**CREATE TABLE "AccountType" (**

**AccountTypeID integer PRIMARY KEY AUTOINCREMENT NOT NULL,**

**AccountTypeName text NOT NULL);**

**CREATE TABLE "Meals" (**

**MealID integer PRIMARY KEY AUTOINCREMENT NOT NULL,**

**MealName text NOT NULL,**

**MealPrice float NOT NULL);**

**CREATE TABLE "DietaryTypes" (**

**DietaryTypesID integer PRIMARY KEY AUTOINCREMENT NOT NULL,**

**DietaryTypesName text NOT NULL);**

**CREATE TABLE "User" (**

**UserID integer PRIMARY KEY AUTOINCREMENT NOT NULL,**

**AccountTypeID integer NOT NULL,**

**FirstName text NOT NULL,**

**LastName text NOT NULL,**

**PhoneNumber integer NOT NULL CHECK(LENGTH(PhoneNumber) == 10),**

**Email text NOT NULL,**

**Username text NOT NULL,**

**Password text NOT NULL,**

**CONSTRAINT User\_FK\_1 FOREIGN KEY (AccountTypeID) REFERENCES AccountType(AccountTypeID));**

**CREATE TABLE "Catering" (**

**CateringID integer PRIMARY KEY AUTOINCREMENT NOT NULL,**

**UserID integer NOT NULL,**

**MealID integer NOT NULL,**

**EventID integer NOT NULL,**

**CardID integer NOT NULL,**

**TypesOfCateringID integer NOT NULL,**

**SpecificCateringRequests text NOT NULL,**

**CONSTRAINT Catering\_FK\_1 FOREIGN KEY (UserID) REFERENCES User(UserID),**

**CONSTRAINT Catering\_FK\_2 FOREIGN KEY (MealID) REFERENCES Meals(MealID),**

**CONSTRAINT Catering\_FK\_3 FOREIGN KEY (EventID) REFERENCES Event(EventID),**

**CONSTRAINT Catering\_FK\_4 FOREIGN KEY (CardID) REFERENCES CardDetails(CardID),**

**CONSTRAINT Catering\_FK\_5 FOREIGN KEY (TypesOfCateringID) REFERENCES TypesOfCatering(TypesOfCateringID));**

**CREATE TABLE "EventMenu" (**

**EventMenuID integer PRIMARY KEY AUTOINCREMENT NOT NULL,**

**CateringID integer NOT NULL,**

**MenuID integer NOT NULL,**

**CONSTRAINT Catering\_FK\_1 FOREIGN KEY (CateringID) REFERENCES Catering(CateringID),**

**CONSTRAINT Catering\_FK\_2 FOREIGN KEY (MenuID) REFERENCES Menu(MenuID));**

**CREATE TABLE "EvemtDietaryRequirements" (**

**EvemtDietaryRequirementsID integer PRIMARY KEY AUTOINCREMENT NOT NULL,**

**DietaryTypesID integer NOT NULL,**

**CateringID integer NOT NULL,**

**Qty integer NOT NULL,**

**CONSTRAINT EvemtDietaryRequirements\_FK\_1 FOREIGN KEY (DietaryTypesID) REFERENCES DietaryTypes(DietaryTypesID),**

**CONSTRAINT EvemtDietaryRequirements\_FK\_2 FOREIGN KEY (CateringID) REFERENCES Catering(CateringID));**

## SQL Queries Used to Insert Data into the Database:

**INSERT** **INTO** ContactInfo (Phone, Email) /\* Insets data into the selected table and rows \*/

**VALUES** /\* Telling the program that the following lines are vaues that need to be inserted into database \*/

('0421894321','john.smith@gsg.wa.edu.au'), /\* Data that is being inserted \*/

('0449214942','sam.johnson@gsg.wa.edu.au'), /\* Data that is being inserted \*/

('0498765432','sarah.mitchell@gsg.wa.edu.au'), /\* Data that is being inserted \*/

('0487654321','liam.mitchell@gsg.wa.edu.au'), /\* Data that is being inserted \*/

('0498415422',''), /\* Data that is being inserted \*/

('0498422595',''), /\* Data that is being inserted \*/

('0498417411','lockyeravenue@petsandvetswa.com.au'), /\* Data that is being inserted \*/

('0498411788','mail@albanyvet.com.au'); /\* Data that is being inserted \*/

**INSERT** **INTO** Address ("Number", Streat, Suburb, City, State, PostCode) /\* Insets data into the selected table and rows \*/

**VALUES** /\* Telling the program that the following lines are vaues that need to be inserted into database \*/

('12','Anchorage Vista','Bayonet Head','Albany','WA','6330'), /\* Data that is being inserted \*/

('942','Lower Denmark Road','Cuthbert','Albany','WA','6330'), /\* Data that is being inserted \*/

('123','Oak Street','Kalgan','Albany','WA','6330'), /\* Data that is being inserted \*/

('17','Wallaby Street','Yakamia','Albany','WA','6330'), /\* Data that is being inserted \*/

('58','Cockburn Road','Mira Mar','Albany','WA','6330'), /\* Data that is being inserted \*/

('323','Middleton Road','Mount Clarence','Albany','WA','6330'), /\* Data that is being inserted \*/

('89','Lockyer Ave','Lockyer','Albany','WA','6330'), /\* Data that is being inserted \*/

('312','Albany Hwy','Centennial Park','Albany','WA','6330'); /\* Data that is being inserted \*/

**INSERT** **INTO** Handler (First\_Name, Last\_Name, AddressID , ContactInfoID) /\* Insets data into the selected table and rows \*/

**VALUES** /\* Telling the program that the following lines are vaues that need to be inserted into database \*/

('Anna','Smith','1','1'), /\* Data that is being inserted \*/

('Marty','Johnson','2','2'), /\* Data that is being inserted \*/

('Sarah','Mitchell','3','3'), /\* Data that is being inserted \*/

('Liam','Lawson','4','4'); /\* Data that is being inserted \*/

**INSERT** **INTO** Vet (Name, AddressID, ContactInfoID) /\* Insets data into the selected table and rows \*/

**VALUES** /\* Telling the program that the following lines are vaues that need to be inserted into database \*/

('Mira Mar Veterinary Hospital','5','5'), /\* Data that is being inserted \*/

('Dog Rock Vet Clinic','6','6'), /\* Data that is being inserted \*/

('Lockyer Avenue Vet Hospital','7','7'), /\* Data that is being inserted \*/

('Albany Veterinary Hospital','8','8'); /\* Data that is being inserted \*/

**INSERT** **INTO** Student (First\_Name, Last\_Name, "Year") /\* Insets data into the selected table and rows \*/

**VALUES** /\* Telling the program that the following lines are vaues that need to be inserted into database \*/

('Ella','Anderson','11'), /\* Data that is being inserted \*/

('Tom','Baker','8'), /\* Data that is being inserted \*/

('Noni','Nadella','4'), /\* Data that is being inserted \*/

('Luke','Roecker','6'); /\* Data that is being inserted \*/

**INSERT** **INTO** Subject (Subject\_Name) /\* Insets data into the selected table and rows \*/

**VALUES** /\* Telling the program that the following lines are vaues that need to be inserted into database \*/

('English'), /\* Data that is being inserted \*/

('Science'), /\* Data that is being inserted \*/

('Math'), /\* Data that is being inserted \*/

('HASS'), /\* Data that is being inserted \*/

('Health'), /\* Data that is being inserted \*/

('Sport'), /\* Data that is being inserted \*/

('Japanese'), /\* Data that is being inserted \*/

('VACS'); /\* Data that is being inserted \*/

**INSERT** **INTO** Class\_Room (Class\_Room\_Name) /\* Insets data into the selected table and rows \*/

**VALUES** /\* Telling the program that the following lines are vaues that need to be inserted into database \*/

('PC1'), /\* Data that is being inserted \*/

('PC2'), /\* Data that is being inserted \*/

('PC3'), /\* Data that is being inserted \*/

('PC4'), /\* Data that is being inserted \*/

('TC1'), /\* Data that is being inserted \*/

('TC2'), /\* Data that is being inserted \*/

('TC3'), /\* Data that is being inserted \*/

('TC4'); /\* Data that is being inserted \*/

**INSERT** **INTO** Classes (SubjectID, Class\_RoomID, "Year") /\* Insets data into the selected table and rows \*/

**VALUES** /\* Telling the program that the following lines are vaues that need to be inserted into database \*/

('1','8','12'), /\* Data that is being inserted \*/

('2','7','11'), /\* Data that is being inserted \*/

('3','6','9'), /\* Data that is being inserted \*/

('4','5','7'), /\* Data that is being inserted \*/

('5','4','5'), /\* Data that is being inserted \*/

('6','3','3'), /\* Data that is being inserted \*/

('7','2','1'), /\* Data that is being inserted \*/

('8','1','Kindergarten'); /\* Data that is being inserted \*/

**INSERT** **INTO** Dog (HandlerID, VetID, Dog\_Name, Dog\_Age, Dog\_Breed) /\* Insets data into the selected table and rows \*/

**VALUES** /\* Telling the program that the following lines are vaues that need to be inserted into database \*/

('1','1','Buddy','4','Labrador'), /\* Data that is being inserted \*/

('2','2','Bella','3','Australian Shepherd'), /\* Data that is being inserted \*/

('3','3','Max','5','Kelpie'), /\* Data that is being inserted \*/

('4','4','Steve','15','Pug'); /\* Data that is being inserted \*/

**INSERT** **INTO** Interactions (DogID, StudentID , ClassID) /\* Insets data into the selected table and rows \*/

**VALUES** /\* Telling the program that the following lines are vaues that need to be inserted into database \*/

('1','1',''), /\* Data that is being inserted \*/

('1','2',''), /\* Data that is being inserted \*/

('1','','8'), /\* Data that is being inserted \*/

('2','2',''), /\* Data that is being inserted \*/

('3','','6'), /\* Data that is being inserted \*/

('4','4',''), /\* Data that is being inserted \*/

('4','','3'), /\* Data that is being inserted \*/

('3','1',''); /\* Data that is being inserted \*/

**INSERT** **INTO** Incedents (DogID, StudentID , **Time**, **Date**, Description\_Of\_Incedent) /\* Insets data into the selected table and rows \*/

**VALUES** /\* Telling the program that the following lines are vaues that need to be inserted into database \*/

('1','','11:43','24/02/2023','Loud barking in class'), /\* Data that is being inserted \*/

('1','','09:40','03/02/2023','Handler dropped lead and dog ran to flock of ibis on the oval to chase them'), /\* Data that is being inserted \*/

('2','4','13:34','06/07/2021','Student dropped dog, required vet visit'), /\* Data that is being inserted \*/

('3','','15:00','20/09/2023','Dog got off leash and ran away'), /\* Data that is being inserted \*/

('4','4','08:50','26/11/2022','Student pulled tail, dog then bite student'), /\* Data that is being inserted \*/

('3','2','10:45','11/05/2023','Dropped by student'), /\* Data that is being inserted \*/

('2','','12:06','16/07/2023','Stole Students Lunch'), /\* Data that is being inserted \*/

('4','','13:15','25/09/2022','Chased student around school'); /\* Data that is being inserted \*/

**INSERT** **INTO** PassFail (PassFailID, Name) /\* Insets data into the selected table and rows \*/

**VALUES** /\* Telling the program that the following lines are vaues that need to be inserted into database \*/

('0', 'Fail'), /\* Data that is being inserted \*/

('1', 'Pass'); /\* Data that is being inserted \*/

**INSERT** **INTO** Assessments (DogID, "Date", "Result", Recommended\_Activities, Recommended\_Age\_Groups, Special\_Considerations) /\* Insets data into the selected table and rows \*/

**VALUES** /\* Telling the program that the following lines are vaues that need to be inserted into database \*/

('1','5/02/2020','0','','',''), /\* Data that is being inserted \*/

('2','7/06/2020','1','One-on-one play, multiple students, emotional support','K-12','Large dog, Energetic'), /\* Data that is being inserted \*/

('2','23/04/2021','0','','',''), /\* Data that is being inserted \*/

('1','15/08/2021','1','Classroom support','K-6','Arthritis in front joints'), /\* Data that is being inserted \*/

('3','8/01/2022','1','Classroom support','7-12','Allergy to pollen'), /\* Data that is being inserted \*/

('1','20/03/2022','1','Classroom support','K-6','Arthritis in front joints'), /\* Data that is being inserted \*/

('2','15/11/2022','1','One-on-one play, multiple students, emotional support','K-12','Large dog, Energetic'), /\* Data that is being inserted \*/

('3','18/09/2023','0','','',''); /\* Data that is being inserted \*/

# Part 2 Manipulate

## SQL Queries Used to Manipulate Data:

**SELECT** /\*This query returns the handlerse contact information and which dog is theirs\*/

d.Dog\_Name **AS** Dog\_Names,

h.First\_Name ||' '|| h.Last\_Name **AS** Handler\_Name,

c.Phone **AS** Phone,

c.Email **AS** Email

**FROM** Handler h

**INNER** **JOIN** Dog d

**ON** d.HandlerID = h.HandlerID

**INNER** **JOIN** ContactInfo c

**ON** h.ContactInfoID = c.ContactInfoID;

**SELECT** /\*This query returns the handlers name, dogs name and how many incidents the dog has\*/

h.First\_Name ||' '|| h.Last\_Name **AS** Handler\_Name,

d.Dog\_Name **AS** Dog\_Name,

**COUNT**(i.IncedentID) **AS** IncidentCount

**FROM** Handler h

**INNER** **JOIN** Dog d

**ON** h.HandlerID = d.HandlerID

**INNER** **JOIN** Incedents i

**ON** d.DogID = i.DogID

**GROUP** **BY** h.First\_Name, h.Last\_Name, d.Dog\_Name

**HAVING** **COUNT**(i.IncedentID) > 0;

**SELECT** /\*This query returns all incedent reports that have been made\*/

d.Dog\_Name **AS** Dog\_Name,

h.First\_Name ||' '|| h.Last\_Name **AS** Handler\_Name,

s.First\_Name ||' '|| s.Last\_Name **AS** Student\_Name,

i.**Time**,

i.**Date**,

i.Description\_Of\_Incedent

**FROM** Dog d

**INNER** **JOIN** Handler h

**ON** d.HandlerID = h.HandlerID

**INNER** **JOIN** Incedents i

**ON** d.DogID = i.DogID

**LEFT** **JOIN** Student s

**ON** i.StudentID = s.StudentID;

**SELECT** /\*This query returns all interactions each dog has with students and classes\*/

d.Dog\_Name,

s.First\_Name ||' '|| s.Last\_Name **AS** Student\_Name,

i.ClassID

**FROM** Interactions i

**LEFT** **JOIN** Student s

**ON** s.StudentID = i.StudentID

**LEFT** **JOIN** Dog d

**ON** d.DogID = i.DogID;

**SELECT** /\*This query returns how many interactions each dog has and their handler\*/

h.First\_Name ||' '|| h.Last\_Name **AS** Handler\_Name,

d.Dog\_Name,

**COUNT**(i.InteractionID) **AS** Interaction\_Count

**FROM** Handler h

**INNER** **JOIN** Dog d

**ON** h.HandlerID = d.HandlerID

**INNER** **JOIN** Interactions i

**ON** d.DogID = i.DogID

**GROUP** **BY** h.First\_Name, h.Last\_Name, d.Dog\_Name

**HAVING** **COUNT**(i.InteractionID) > 0;

**SELECT** /\*This query returns the ammount of assessment a dog has, the last assessment they had and their next assessment date\*/

d.Dog\_Name,

**COUNT**(a.AssessmentID) **AS** Assessment\_Count,

**MAX**(**STRFTIME**('%Y-%m-%d', **SUBSTR**(a.**Date**, 7, 4) || '-' || **SUBSTR**(a.**Date**, 4, 2) || '-' || **SUBSTR**(a.**Date**, 1, 2))) **AS** Last\_Assessment\_Date,

**STRFTIME**('%Y-%m-%d', **DATE**(**SUBSTR**(a.**Date**, 7, 4) || '-' || **SUBSTR**(a.**Date**, 4, 2) || '-' || **SUBSTR**(a.**Date**, 1, 2), '+1 year')) **AS** Next\_Assessment\_Date

**FROM** Assessments a

**INNER** **JOIN** Dog d

**ON** d.DogID = a.DogID

**GROUP** **BY** d.Dog\_Name

**HAVING** **COUNT**(a.AssessmentID) > 0;

**SELECT** /\*This query returns each dogs last assessment information\*/

**MAX**(**STRFTIME**('%Y-%m-%d', **SUBSTR**(a.**Date**, 7, 4) || '-' || **SUBSTR**(a.**Date**, 4, 2) || '-' || **SUBSTR**(a.**Date**, 1, 2))) **AS** Assessment\_Date,

d.Dog\_Name,

p.Name **AS** **Result**,

a.Recommended\_Activities **AS** Recommended\_Activities,

a.Recommended\_Age\_Groups **AS** Recommended\_Age\_Groups,

a.Special\_Considerations **AS** Special\_Considerations

**FROM** Assessments a

**LEFT** **JOIN** Dog d

**ON** d.DogID = a.DogID

**LEFT** **JOIN** PassFail p

**ON** a."Result" = p.PassFailID

**WHERE** **STRFTIME**('%Y-%m-%d', **SUBSTR**(a.**Date**, 7, 4) || '-' || **SUBSTR**(a.**Date**, 4, 2) || '-' || **SUBSTR**(a.**Date**, 1, 2)) = (

**SELECT** **MAX**(**STRFTIME**('%Y-%m-%d', **SUBSTR**(a2.**Date**, 7, 4) || '-' || **SUBSTR**(a2.**Date**, 4, 2) || '-' || **SUBSTR**(a2.**Date**, 1, 2)))

**FROM** Assessments a2

**WHERE** a.DogID = a2.DogID)

**GROUP** **BY** d.DogID;

**SELECT** /\*This query returns all vet information\*/

v.Name **AS** Name,

c.Phone **AS** Phone,

c.Email **AS** Email,

a.**Number** ||' '|| a.Streat ||' '|| a.Suburb ||' '|| a.City ||' '|| a.State ||' '|| a.postcode **AS** Address

**FROM** Vet v

**LEFT** **JOIN** ContactInfo c

**ON** v.ContactInfoID = c.ContactInfoID

**INNER** **JOIN** Address a

**ON** v.AddressID = a.AddressID;

**SELECT** /\*This query returns how many dogs are in the database\*/

**COUNT**(d.DogID) **AS** Dog\_Count

**FROM** Dog d;

**SELECT** /\*This query returns the hour that most incedents happen\*/

**CAST**(**STRFTIME**('%H', **SUBSTR**(i.**Time**, 1, 5) || ':00') **AS** **INTEGER**) **as** **Hour**, **COUNT**(\*) **as** Incident\_Count

**FROM** Incedents i

**GROUP** **BY** **Hour**

**ORDER** **BY** Incident\_Count **DESC**

**LIMIT** 1;