## Remote Palliative Care – Iteration 3

## General Data Protection Regulation (GDPR)

### Overview

GDPR is based in the UK and is a regime that applies to businesses that acquire and hold personal information about European residents.

The UK GDPR sets out seven key principles:

* Lawfulness, fairness, and transparency
* Purpose limitation
* Data minimisation
* Accuracy
* Storage limitation
* Integrity and confidentiality (security)
* Accountability (ico, 2022)

Here in New Zealand, we don’t need to comply with GDPR because it is European law not New Zealand law but because the palliative care system may be holding delicate information about a European citizen then the system must comply with the GDPR.

By complying with the New Zealand privacy act, it gets you extremely close to complying with GDPR, the one thing that the NZ privacy act lacks is it does not express data portability. (privacy, 2022)

### Implications

All organizations and companies that work with personal data should appoint a data protection officer or data controller who oversees GDPR compliance.

There are tough penalties for those companies and organizations who don’t comply with GDPR fines of up to **4% of annual global revenue**or**20 million Euros**, whichever is greater.

### How it Complies

For this iteration the Remote palliative care database does not comply with New Zealand privacy act let alone the GDPR regime.

**What I have done:**

* There are different users and roles with different privileges.
* Also views that are granted to certain user.
* All sensitive data is encrypted.

**What will need to be done when system starts acquiring real data:**

* All users will need to give consent before handing over their personal information and will be given clear advice regarding how they can access for viewing or updating it at any time.
* Passwords must be 16 characters long where is the default for MySQL is 8 when using the community edition.
* I think for a production environment for a healthcare system SQL Server would be a better option that MySQL, so migration would need to happen.

## Design/Analysis

### Noun analysis

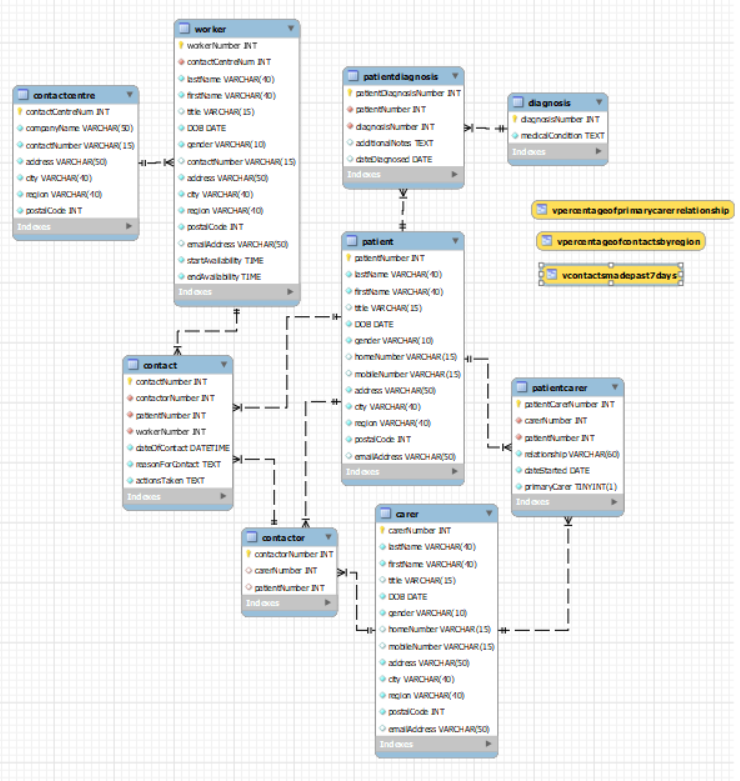
Red is what I have added in Iteration 2

|  |  |  |  |  |  |
| --- | --- | --- | --- | --- | --- |
| Patient | Carer | Worker | Contact | Contact Centre | Diagnosis |
| Name | Name | Name | Date of contact | Phone Number | Medical Condition |
| Title | Title | Title | Reason for contact | Address | Date of Diagnosis |
| Address | Address | Address | Actions taken |  | Additional diagnosis notes |
| Phone number | Phone number | Phone number | Contactor |  |  |
| Gender | Gender | Gender |  |  |  |
| DOB | DOB | DOB |  |  |  |
|  |  | Availability |  |  |  |

### Normalisation And Entity Relationship Diagram

File Path: " \ERD and Normalisation.vsdx"

### Reverse Engineered ERD



### Assumptions

**From Iteration 1**

**Patient –** The patient could have many carers.

**Carer –** The Carer could have many Patients.

**PatientCarer –** Because of the many to many relationship a bridging table is needed. Some additional attributes that would be good to record would be the relationship between the carer and patient, the time the carer started care and if the carer is the patient’s primary carer.

**Worker –** To keep it simple I have just made the worker be a person with similar attributes to carer and patient with an extra attribute (availability) and only one contact phone number.

**Contactor –** can either be a Carer or a Patient so this table has a contactor number as primary key and both patient and carer as a nullable foreign key as it can only be one of them.

**Contact –** needs date of contact, reason for contact, the actions taken the patients number the workers number and the contactor.

**ContactCentre –** The initial contact that is made would be a bigger facility and then the call would be passed to the worker based on when they are available or not so the worker would belong to a contact centre.

**Diagnosis –** This table holds a bunch of medical conditions that the patient could have. But because patient can have many medical conditions and diagnosis is for many patients there must be a bridging table.

**PatientDiagnosis –** Acts as a bridging table between patient and Diagnosis and has a date of diagnosis field. To keep a record of when they were diagnosed with that medical condition along with the date there is additional notes regarding the diagnosis.

## Code

I chose to create the database in MYSQL and generated dummy data from generatedata.com

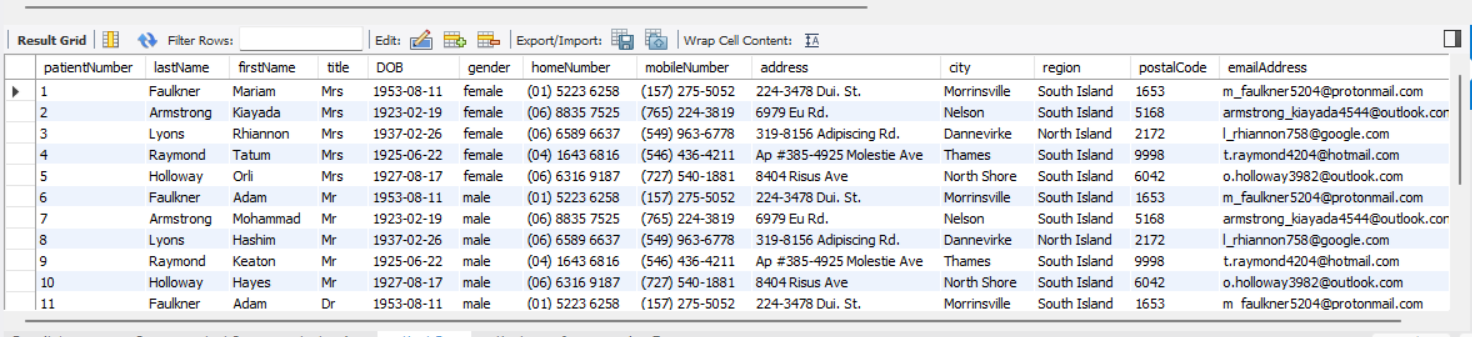
### Show Tables

Graphical user interface, text, application

Description automatically generated

**Select \* queries to display table data**

### Patient



### Carer

Graphical user interface, text, application

Description automatically generated

### PatientCarer

Table

Description automatically generated

### Worker



### Contactor

Table

Description automatically generated

### Contact

Text

Description automatically generated

### ContactCentre

Graphical user interface, text, application, email

Description automatically generated

### Diagnosis

Graphical user interface, application, table

Description automatically generated

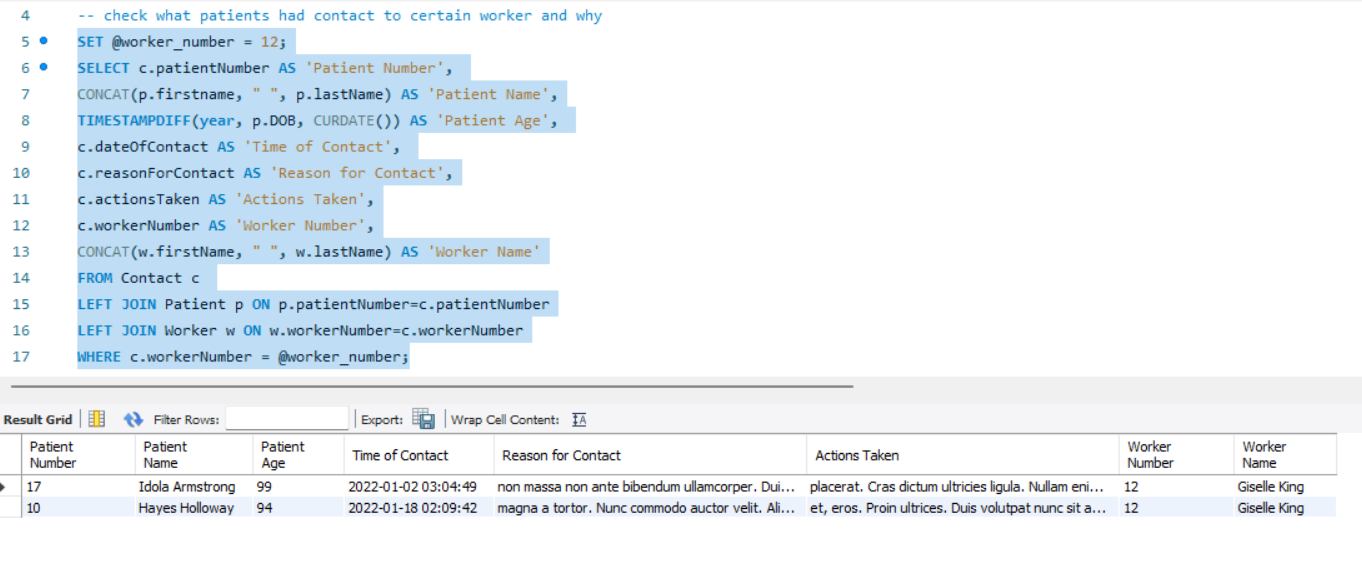
### PatientDiagnosis

Table

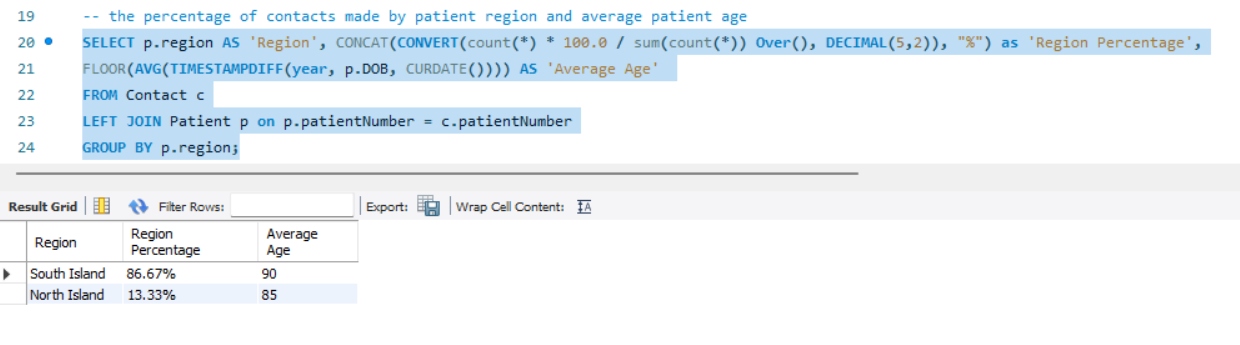
Description automatically generated

### Queries

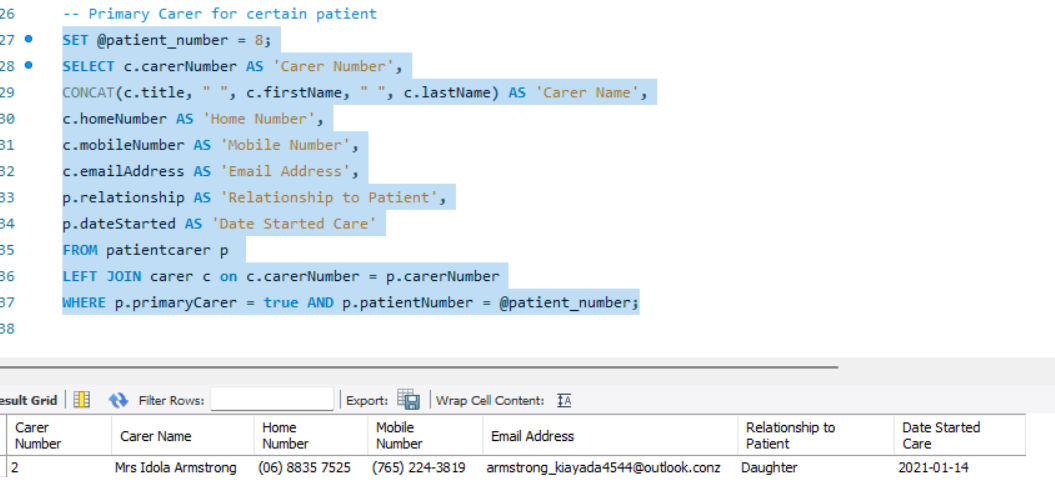
**Q#1 Check what patients had contact to certain worker and why**



**Q#2 The percentage of contacts made by patient region and average patient age (for some reason when I generated the data the only regions for New Zealand were North and South Island)**

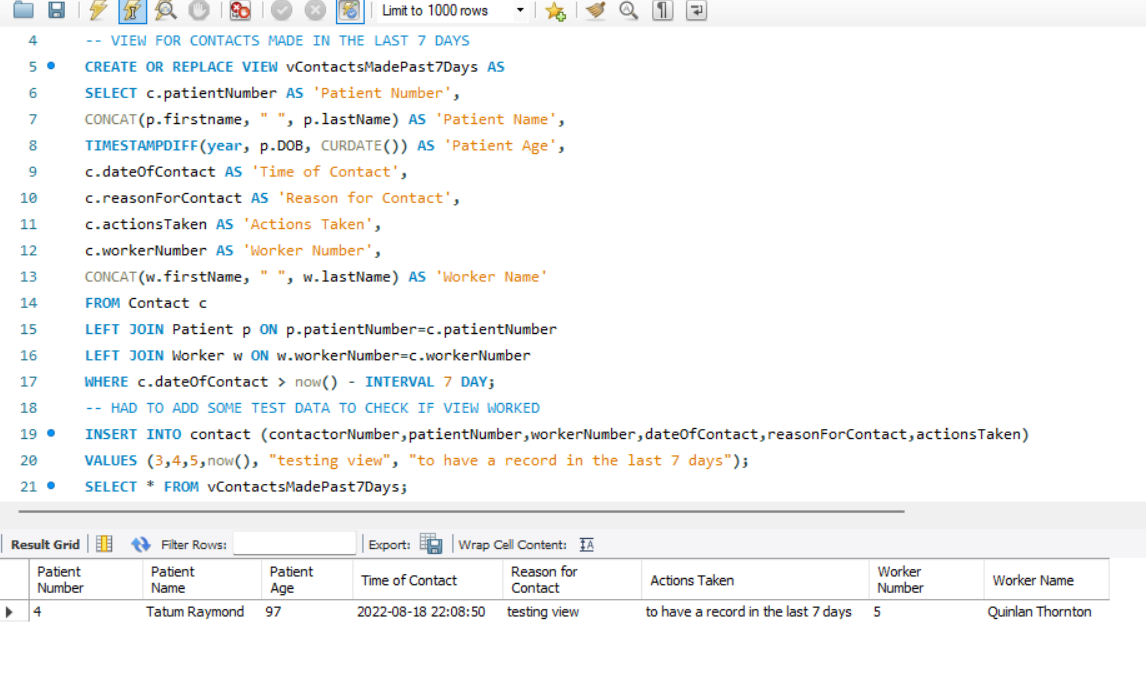


**Q#3 Primary Carer for certain patient**

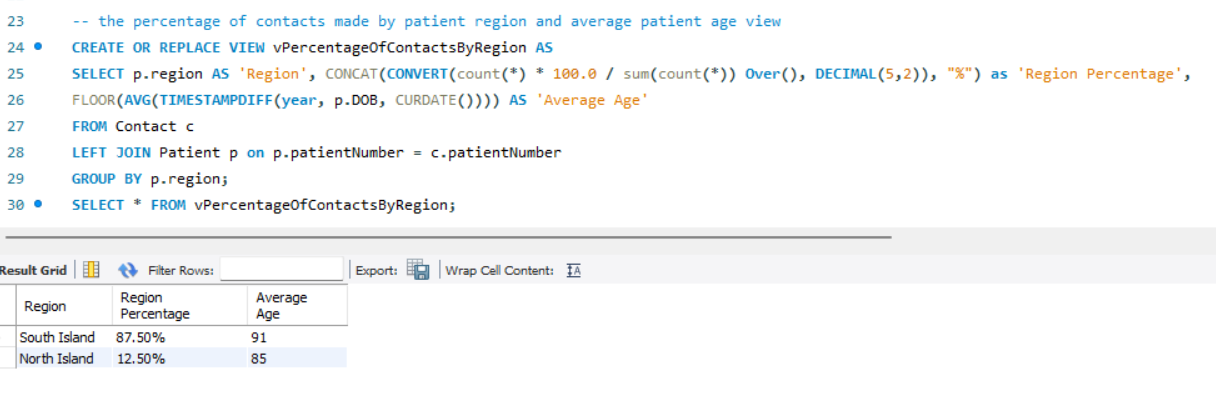


### Views

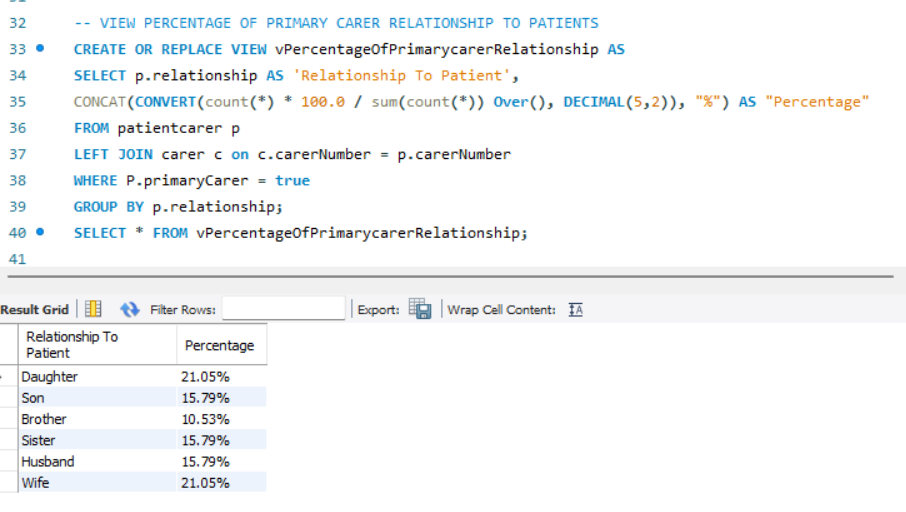
**V#1 Contacts made in the last 7 days (had to add some extra data to bring up a record)**



**V#2 The percentage of contacts made by patient region with average patient age**



**V#3 Percentage of primary carers of patients by their relationship to patient**



### Indexes

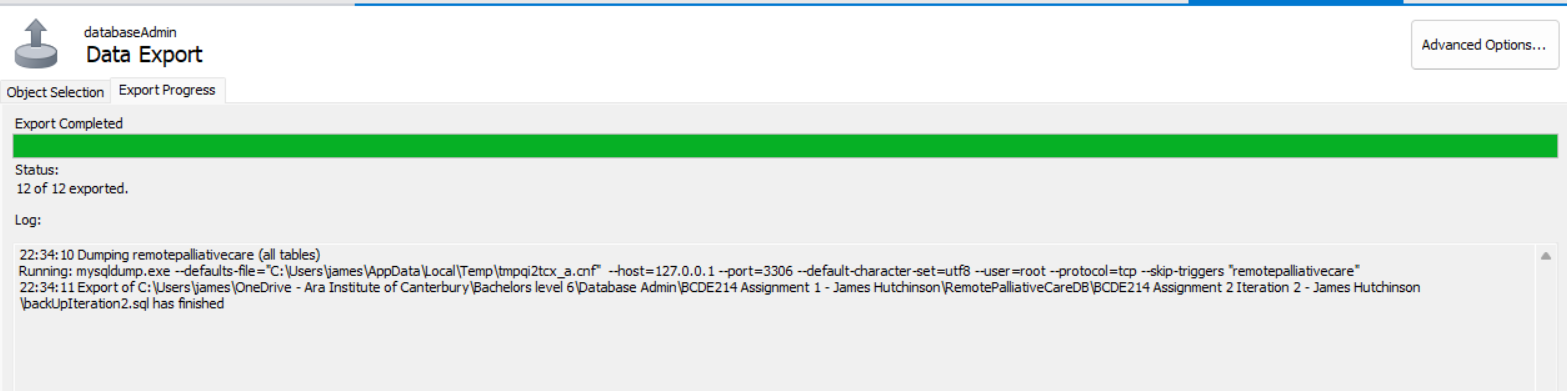
With performance of basic query before and after index created.

|  |  |
| --- | --- |
| **Before** | **After** |
| Graphical user interface, text  Description automatically generated | |
|  |  |
|  | |
|  |  |
|  | |
|  |  |
|  | |
|  |  |
|  | |
|  |  |
|  | |
|  |  |
|  | |
|  |  |
|  | |
|  |  |
|  | |
|  |  |
|  | |
|  |  |
|  | |
|  |  |
|  | |
|  |  |
|  | |
|  |  |

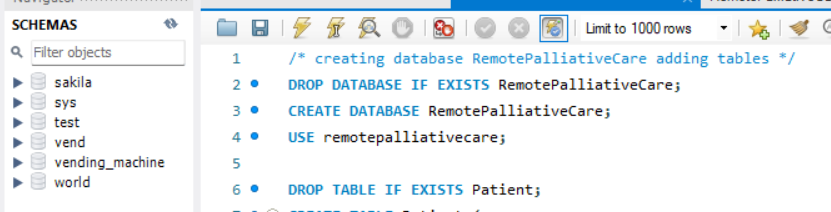
## Back up, Restore, Security and Role Management

### Backup

**This time I decided to export dump contained in 1 file as last iteration I exported tables in project folder**



### Drop Database



### Restore

Graphical user interface, application

Description automatically generated

Graphical user interface, text, application

Description automatically generated

Graphical user interface, text, application

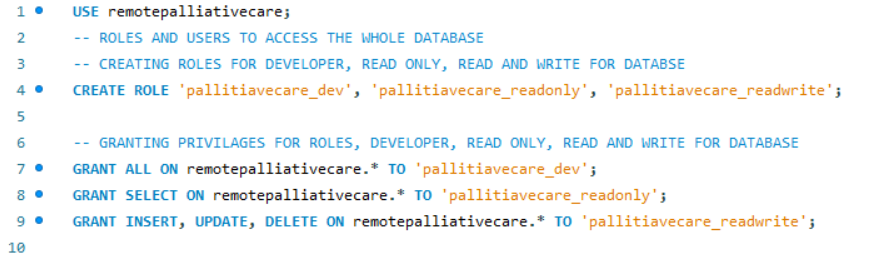
Description automatically generated

Graphical user interface, text, application, email

Description automatically generated

### Role management

**Creating Roles and Granting Privileges for them, for developer, read only, read & write.**

****

**Creating Users developer, read only, read & write and assigning them to a role.**

**A picture containing calendar

Description automatically generated**

**Graphical user interface, text, application

Description automatically generated**

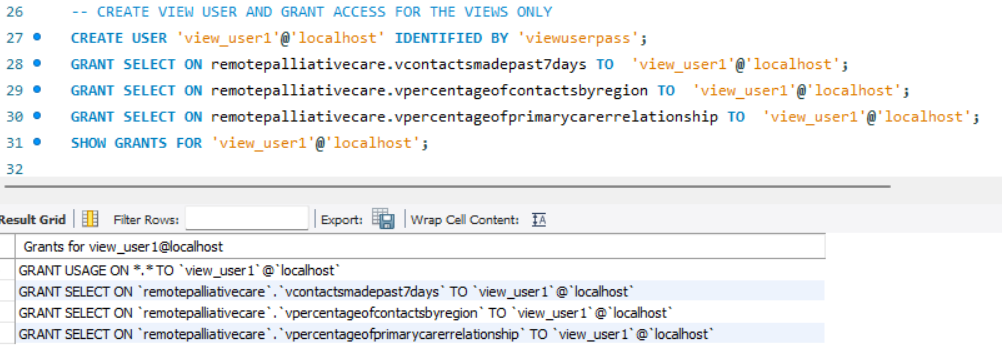
**Graphical user interface, text, application

Description automatically generated**

**Graphical user interface, application

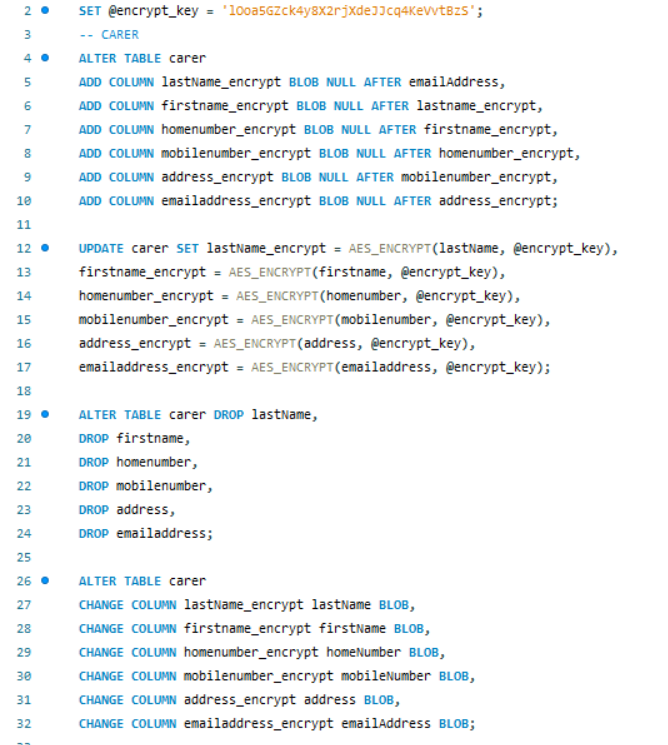
Description automatically generated**

**Creating user and assigning privileges to read only the 3 views created**

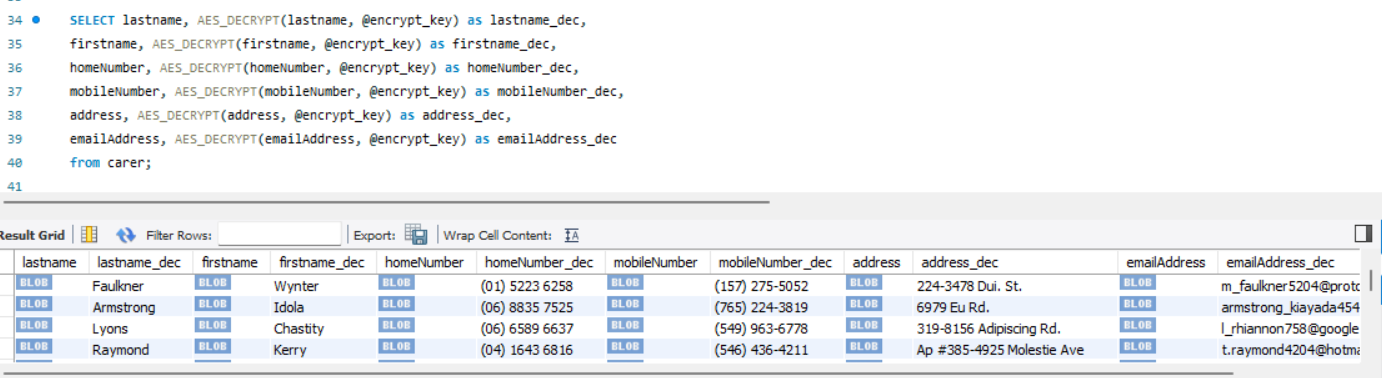
****

### Encryption

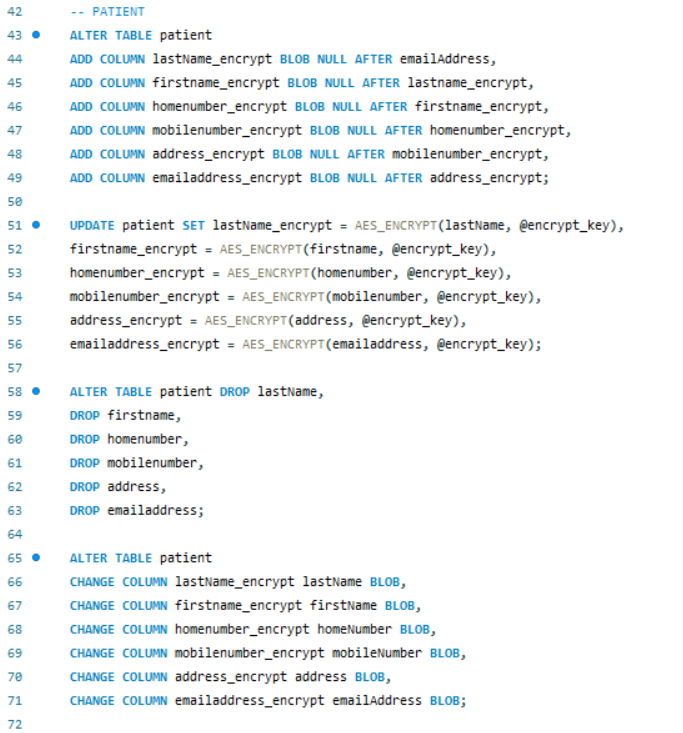
**Encrypting personal data for Carer table**



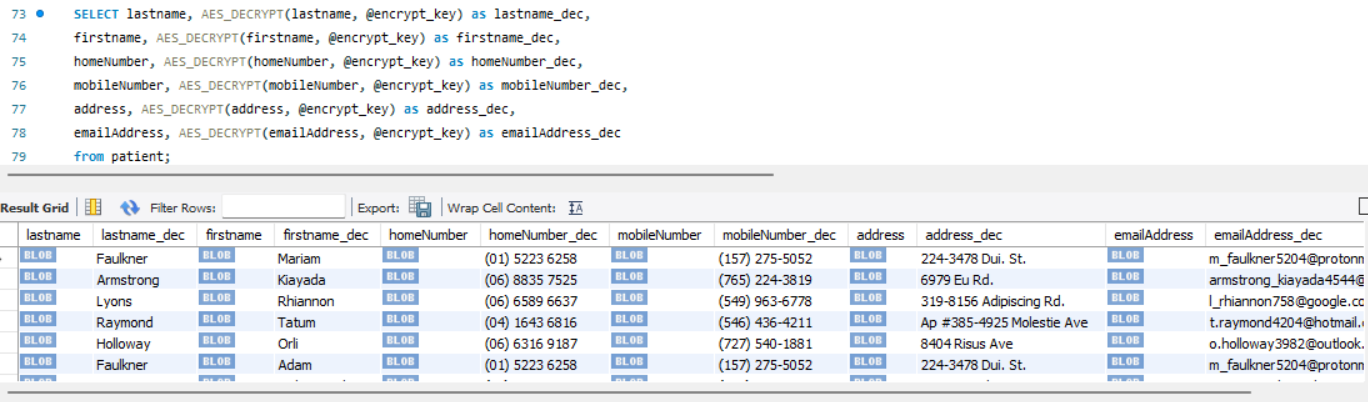
**Query with encrypt / decrypt data for Carer table**

****

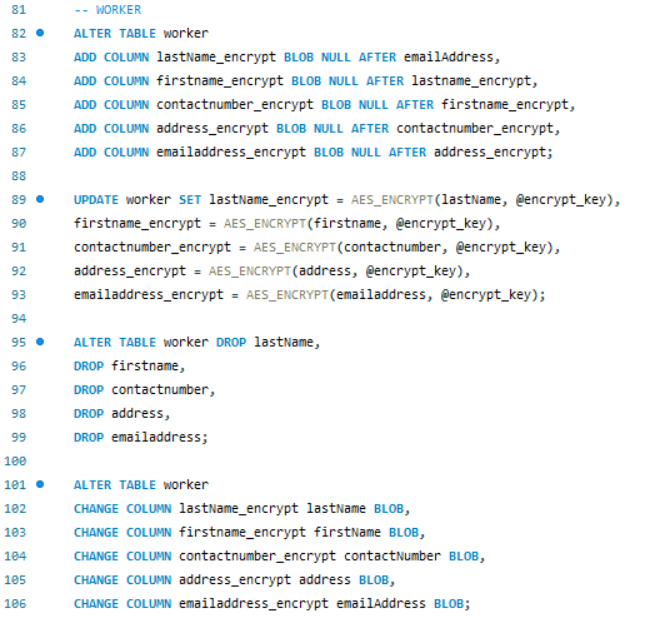
**Encrypting personal data for Patient table**

****

**Query with encrypt / decrypt data for Patient table**

****

**Encrypting personal data for Worker table**

****

**Query with encrypt / decrypt data for Worker table**

**Graphical user interface, application

Description automatically generated**

## Performance Review Iteration 1

### What worked well

**Design –** Noun analysis was good I didn’t want to get too out of hand for the first iteration regarding the number of attributes that would be required. The table set up and ERD was relatively quick to implement.

**Code –** Creating the database was good once I started it everything, I learnt from database design class came flooding back so was quick to do this stage, I think the queries I done are insightful and would be used in different areas of the healthcare sector.

**Backup –** was easy and I done this quick.

### What didn’t work

**Design –** I had trouble with generating the data when it came to linking the patient to the carer for the contactor table it took quite a while because after generating, I had to change so it would match the patient Number to the right carer.

**Code –** I took awhile to think of the queries so they would be good enough and retrieve insightful information.

**Security –** I didn’t implement any security but that was part of my plan, anyway, waiting for next iteration.

### What I would do next time

Regarding this iteration not much because I wanted to leave some things to do for the next iteration, but I would bulk insert data from file instead of inserting into table. I will rectify this on the upcoming iterations.

## Performance Review Iteration 2

### What worked well

**Design –** Adding the 3 new tables and extra attributes really made the database feel like it makes more sense. I’m happy with the design now.

**Code –** Coding the new tables and connecting them went well and all the queries still work fine. I thought the queries I had done in iteration 1 were very informative so I didn’t bother in changing them, but I did write some views that I feel show desired information within the database. Also, I changed the method in which I inserted data to ‘infile’ rather than in iteration 1 was straight insert into table.

**Backup –** Iteration 1, I dumped the back up file as table data files within a project folder, so I decided to dump in 1 contained file this time and restore worked fine.

**Security –** I created the roles and users that would be using the database and a user that can only read the views I wrote.

### What didn’t work

**Design –** After adding the extra tables I didn’t change the normalisation diagram that I originally created as it was already huge.

**Code –** I didn’t create any indexes for query performance as I ran out of time.

**Security –** I could have made more users, but I thought the main 3 roles are there, I read into auto generating but might be a bit out of scope I thought.

### What I would do next time

This iteration went well and I’m on target with my planning for the final system. The only thing left for me to do is encryption of the sensitive data which I will complete in the final iteration and then the database will be GDPR compliant.

# References

ico. (2022, 08 10). *guide to GDPR*. Retrieved from ico.org.uk: https://ico.org.uk/for-organisations/guide-to-data-protection/guide-to-the-general-data-protection-regulation-gdpr/principles/

privacy. (2022, 08 10). *knowledge-base*. Retrieved from privacy.or.nz: https://www.privacy.org.nz/tools/knowledge-base/view/482