

CMPG 311

Physical design

Phase 3

Group Members

➤ **Group leader**

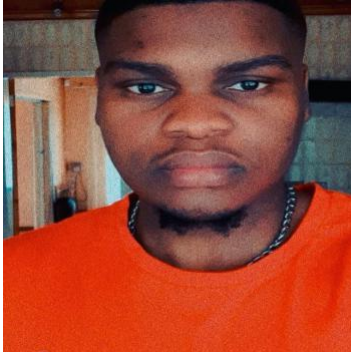
Names:Kabelo Thato

Surname:Ratshefola

Student Number:28704835

Contact Information:063 738 9918

Email : ratshefola.kt@outlook.com



➤ **Group Member 2**

Names:Mpeo Lawrence

Surname:Majeke

Student Number:34531106

Contact Information:062 960 9905

Email :Mpeomajake6@gmail.com



➤ **Group Member 3**

Names: Tlou

Surname: Seanego

Student Number: 32867824

Contact Information: 0763675765



➤ Group Member 4

Names: Akani

Surname: Mashaba

Student Number: 31163300

Contact Information: 071 132 2055

Email: Akanimmashaba@gmail.com



➤ Group Member 5

Names: Tendani Archibold

Surname: Madzivhandila

Student Number: 30332338

Contact Information: 0791601349/

Email: madzivhandila2@gmail.com



Company scenario to analyse

Northwest University receives a lot of applicants every year and also a lot of enrolling students coming back the following year. They all have something in common which is in need of accommodation. The company is a database system that provides accommodation to its students. The University has a large number of students who need housing, and the current system for managing student accommodation is inefficient and time-consuming. The University administration is finding it difficult to manage and allocate rooms to students based on their preferences, requirements, and availability. The college is also facing challenges in maintaining records of students, room assignments, billing, and other administrative tasks related to student accommodation. Northwest University gave us a hint start to create a database system that will make the life of students easier and not stressed about a place of stay for the year they are applying for, so that they can be close to campus since most of the courses are contact. The registration will be done online, when a student applies to enrol at NWU, there is a part where the student will have to choose whether they will need accommodation or not. For the past couple of years there were students who used to sleep at the outside premises of Northwest University because of the long queues so our database wants to reduce those risk of students coming to University without knowing where they will be staying at and the safety of students once they have been approved an accommodation. The process of student applying physically at campus includes storing information of students in a file cabinet and it will cause data integration, data anomalies and data inconsistency that's why it will be done online.

Company Objectives

The main objectives of the university's student accommodation management system are to provide the safety of students at our interest and with a user friendly database system with the following objectives:

1. To automate and streamline the room allocation process, reducing errors and inaccuracies.
2. To match student preferences and requirements with available rooms more efficiently.
3. To provide accurate records of students, room assignments, billing, and other administrative tasks related to student accommodation.

4. To improve the overall student experience by providing a user-friendly platform for managing accommodation.
5. To improve the university's operational efficiency and reduce administrative burden.
6. After students has applied it provide students with feedback of whether they have been accepted or not.
7. Providing the details of all the stuff related to the accommodation students has been approved at.

Company Operations

The database system is going to operate in two phases, in which one of the phases we going to have students applying for the accommodation he/she is interested in staying at. The second phase will be after a student has been approved at the accommodation where he/she will be staying at and providing the operation of the accommodation database system.

●Phase 1

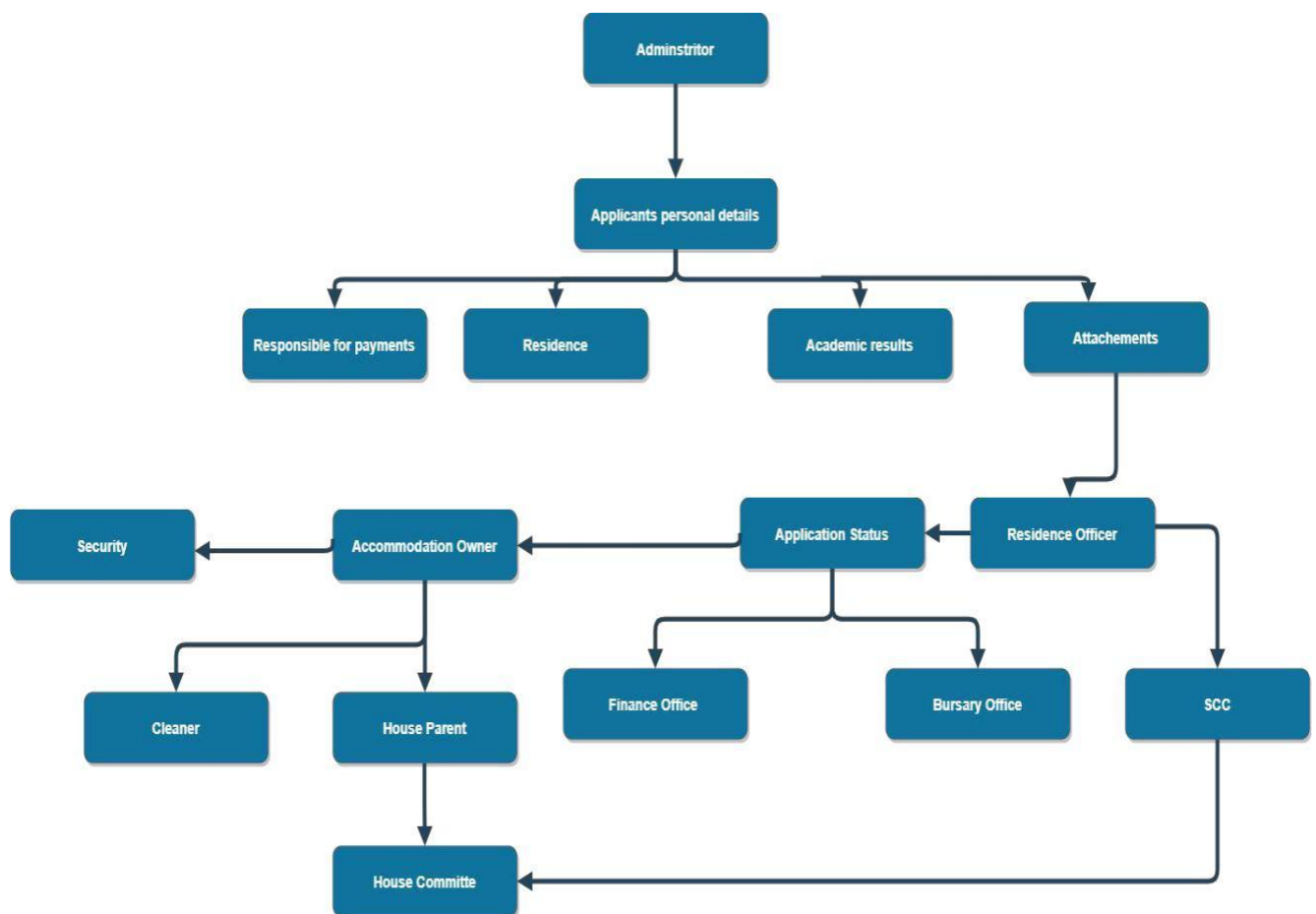
The operation of the company is that there is going to be an administrator who holds all the details of the applicant's details meaning the login details and student number in order for students to enter through the database and proceed with the application. Students will then after provide their personal details and their home address even also their contacts details, the students will have choose after who is going to be responsible for the payment of accommodation and provide the details of the person who is going to be responsible for the payments. The students will be given three choices of choosing the accommodation he/she is interested in staying at by being provided the names of the accommodations available. The last part of phase one database system will be based on students attaching the documents needed in order to secure the space, students will not be allowed to submit the application before attaching the documents.

●Phase 2

Students will receive the email of confirmation from the residence officer about the accommodation application and he/she will provide students with the status of being approved or not approved. Once students have been approved the database of management

system will show up. We going to have a bursary office that is The bursary office is responsible for checking whether the student has a bursary for current year and making sure the student has signed the bursary agreement. The residence officer will be dealing with the accommodation owner in terms of finance and security. Accommodation owner will be having stuff that works for them. SCC will be responsible in providing safety for students at a close look by deploying the HK's.

Organizational structure



The company is structured in such a way that the centralized database system will only focus mainly on the Vaal triangle campus, even though it is not the main campus. The database is a multi-user database which many students can apply at same time and it will support multiple students at the same time. In terms of structure, it will be based on a two structure form which is a semi structured database, containing structured data.

Define problems and constraints

The University is facing several problems and constraints in managing student accommodation. Some of these include:

1. Inefficient manual processes for managing room allocations, which are prone to errors and inaccuracies.
2. Difficulty in matching student preferences and requirements with available rooms.
3. Limited availability of suitable accommodation options for students.
4. The need to maintain accurate records of students, room assignments, billing, and other administrative tasks.
5. Limited resources and budget constraints for implementing a new system.
6. Students are being scammed because of being desperate for a place to stay.

Constraints

Lack of devices

Some students come from a poor background where they cannot afford to buy devices such as phones or laptops to apply for accommodation online.

Network constraints

The network connectivity for students who come from rural areas can be a problem, because some rural areas do not have cell phone towers which means their network is always slow.

Budget/Money constraints

Not all students can afford administration fee or even have money to buy data, most students do not have stable internet connections at home.

Communication issues

There is no direct communication between the school, students and the accommodation owners and students tend to get worried and lose their cool during the registration because of lack of communication.

Database system specifications

Objectives to help solve problems identified

For room allocation, this new system will respect the law of first come, first serve to allocate rooms to applicants.

Institution will make sure that they accredit more suitable accommodations to increase options for student to apply.

We will make sure that there is enough accommodation and make it easy to apply, in order to avoid student being scammed because of being desperate.

Our data security will be strong in order to keep student confidential information safe all the time.

Our system is free, in order for financially challenged students to apply.

Information that company requires from database

The information that is going to be required by the database is the information of the students and the information of the accommodation, where students will be providing the personal details and the information of who is going to be responsible for the payments. The up to date list of accommodation that are credited by the university, and all the details of the staff members of the accommodation. Students will have to provide an average results of their latest academic records. In terms of the attachments the students will have to provide ID copy of not more than 3 months, the latest academic results and criminal record. The up to date details of residence officer and SCC students who will be helping, also provide the details.

The information that is going to be needed by the database to ensure the process is being done is as follows:

<pre>CREATE TABLE STUDENT(+STD_FNAME, +STD_LNAME, +STD_ID, +STD_PHONE_NUMBER, +STD_ADDRESS, +STD_EMAIL);</pre>	<pre>CREATE TABLE SCHOOL ADMIN(+ADMIN_ID, + ADMIN_FNAME, + ADMIN_LNAME, + ADMIN_EMAIL, + ADMIN_PHONENUMBER);</pre>
<pre>CREATE TABLE PROSPECTIVE(+STD_ID, +MATRIC_RESULTS, +PROOF_OF_PAYMENT);</pre>	<pre>CREATE TABLE RESIDENCEOFFICER(+OFFICER_ID, + OFFICER_FNAME, + OFFICER_LNAME, + OFFICER_EMAIL, + OFFICER_PHONENUMBER);</pre>
<pre>CREATE TABLE CURRENT(+STD_ID, +ACADEMIC_RECORD);</pre>	<pre>CREATE TABLE FINANCE(+FINANCE_ID, + FINANCE_FNAME, + FINANCE_LNAME, + FINANCE_EMAIL, + FINANCE_PHONENUMBER);</pre>
<pre>CREATE TABLE ONRESIDENCE(+ONRESIDENT_ID, +ONRESIDENCE_NAME, +ONRESIDENCE_TYPE, +ONRESIDENCE_REQUIREMENTS, +ONRESIDENCE_OVERVIEW, +ONRESIDENCE_SECURITY, +NUM_ROOMSAVAILABLE);</pre>	<pre>CREATE TABLE BURSARY(+BURSARY_ID, + BURSARY_FNAME, + BURSARY_LNAME, + BURSARY_PHONENUMBER, + BURSARY_EMAIL);</pre>
<pre>CREATE TABLE OFFRESIDENCE(+ OFFRESIDENCE_ID, + OFFRESIDENCE_NAME, + OFFRESIDENCE_ACCREDITED, + OFFRESIDENCE_OWNERS, + OFFRESIDENCE_TYPE, + OFFRESIDENCE_OVERVIEW, + OFFRESIDENCE_SECURITY, +NUM_ROOMSAVAILABLE);</pre>	<pre>CREATE TABLE APPLICATION(+STATUS, +APPLICATION_NUMBER, +STD_NUMBER);</pre>

<pre>CREATE TABLE CLEANER(+ CLEANER_ID, + CLEANER_FNAME, + CLEANER_LNAME, + CLEANER_EMAIL, + CLEANER_PHONENUMBER);</pre>	<pre>CREATE TABLE STUDENT_ACCOMMODATIONOWNER(+OWNER_NUMBER, +OWNER_NAME, +OWNER_PHONE +OWNER_EMAIL);</pre>
<pre>CREATE TABLE HOUSEPARENT(+ HOUSEPARENT_NUMBER, + HOUSEPARENT_ID, + HOUSEPARENT_NAME + HOUSEPARENT_PHONENUMBER + HOUSEPARENT_EMAILADDRESS);</pre>	<pre>CREATE TABLE SECURITY(+ SECURITY_ID, + SECURITY_FNAME, + SECURITY_LNAME, + SECURITY_EMAIL, + SECURITY_PHONENUMBER);</pre>




Scope and Boundaries:

The scope and boundaries of the University's student accommodation management system are as follows:




1. The system will automate the room allocation process and provide a user-friendly interface for students and staff to manage accommodation.
2. The system will maintain accurate records of students, room assignments, billing, and other administrative tasks related to student accommodation.
3. The system will provide a platform for students to communicate their preferences and requirements for accommodation.
4. The system will provide real-time information on the availability of rooms and other related details.
5. The system will not handle financial transactions related to student accommodation.

Scope

The database will not be complex, which will reduce cost's because the information that is going to be needed is going to be for students and accommodation owners.

-  The academic records of students will be stored in the database for easier selection.
-  Students personal information will be kept private and also of staff members of the accommodation.
-  The database will store finances information like accommodation fee.

Boundaries

-  Money/Budget will be needed in order to create the database even though it will be free of charge for it to operate and also to pay staff members/employees for the work they will be doing.
-  The latest software in terms of a system that the database will be functioning at so that the database could be up to date and stay relevant.
-  Devices will be needed for the use of database.

In conclusion, the University's student accommodation management system will help streamline the room allocation process, reduce errors and inaccuracies, and improve the overall student experience. The system will provide a user-friendly interface for managing accommodation and maintaining accurate records. However, the implementation of the system should be done while keeping in mind the budget constraints and resource limitations of the college

PHASE TWO

Business rules:

- Before student continue with the application process, they will have to go through the administration processes where they will be providing the student number and university password as a login in the system. Only students who are registered at the university system will have access to the database system and they could apply for residences. Students will be given unlimited chances to login in the system.
- A student will be given a chance to place only one application, once a student submits there are application they will never be any submission left to place an application, unless a student has not submitted the application can still login the system and continue with the application process until they submit there are application. After student has submitted they application they will no longer be able to login the system again.
- A student can apply for a maximum of three residences, it will be based on first choice if it is full the second choice will follow but if also the second choice is full the last choice will be considered. The whole processes will be depended on the capacity of the residences can hold and regarding this it will be on first come first sever
- Each residence can have multiple number of students but depending on the number of rooms are available to secure the space for students.
- Residence officer can manage many different types of residences but each residence can be managed by one and only one residence officer.
- Residence officer will be the only one who deals with applications, as they will be many students applying for residences and the students will need to receive the feedback on their application.
- Residence officer will be the one giving out the accredited accommodations and that will be given to the accommodation owner, Accommodation owner will be given an accredit go ahead by the residence officer.
- Accommodation owner will be the only one who employ people/companies to work and make sure the accommodation is in good state.
- Under Employment there is going to be the ones that work as cleaners, security, and lastly house parent who will be taking care of the deal operations of the accommodation.
- Under the house parent there is going to be the house committees that will be supervising the life of students on the accommodation under the guidance of house parent and the SCC.
- Many SCC will be working with many house committees to ensure that productivity is maintained and many students can get help anytime. Many house committees will be guided by many SCC in order to maintain stability.

Final business rule:

One student can login in to the system only once, Student can only have one login details.

Student can apply for a maximum of three residences, but each student can be accepted at only one residence.

Student is going to choose who is going to responsible for the payment between two factors or nothing at all.

Student is going to upload one or many documents required, documents are going to be uploaded by one student.

Officer will receive more than one documents, many documents will be sent to the officer.

House parent will employ many students to help, many students can become house committee because of the house parent

Residence officer will employ many SCC to help students, many students can become SCC and help a residence officer.

Finance officer will receive the payment, each payment will be received by a financial officer.

Bursary officer will check if a student has one or many bursary.

Officer can either be Finance officer, Bursary officer and lastly Residence officer

They are going to be two types of residence which will be Off campus residence and On campus residence.

Residence Officer will be in contact with the accommodation owner to update.

Accommodation owner will employ a company that will be helping with the residence

A company can either be Cleaning company or security company.

Weak entities

Officer
Company
Accommodation
Application
Bursary

Composite keys

Leadership are hc_ID and scc_ID
Studentdocuments are student_ID and resofficer

Composite entities/bridge entities

Leadership
Studentdocuments

Strong relationship:

Administration & Student
Residence & Student
Residence & it's subtypes (Offresidence and On-residnce)
Student & Studentdocuments
Studentdocuments & Officer
Officer & it's subtypes (Financeofficer, Bursaryofficer, and Residenceofficer)
Payment & Financeofficer
Bursaryofficer & Bursary
Accomodationowner & Houseparent
Houseparent & Housecommittee
Housecommittee & Leadership
Leadership & SCC
Residenceofficer & Application
Payment & Student

Weak relationship

Leadership & Student
Accomodationowner & Company
Residenceofficer & Accomodationowner
Application and Accomodationowner

Multi-valued attribute

All of the entities are multi-valued attributes beside the subtypes of payment and residence.

Mandatory relationship

StudentDocuments & Students
Officer also to the subtypes
Leadership in between Housecommittee and SCC

Derived attributes

Student_ID will be a USERNAME

Supertypes entities

Officer
Payment
Residence
Company

Subtypes entities

Financeofficer
Bursaryofficer
Residenceofficer
Selfpayment
Bursary
Offresidence
Onresidence
Security
Cleaner

Logical Design (Normalised 3rd normal form):

ADMINISTRATION(student_ID(PKFK), username, password)

STUDENT(student_ID(PK), student_Fname, student_Lname, student_address, student_Email, student_Phone_Number)

STUDENTDOCUMENTS(student_ID(PKFK), officer_ID(FK), student_IDcopy, student_Academic_Record, proof_Of_Payment)

SCC(scc_ID(PK), resofficer_ID(PKFK), scc_Fname, scc_Address, scc_Email, scc_Phone_Number)

LEADERSHIP(hc_ID(PKFK), scc_ID(PKFK), student_ID(FK))

HOUSECOMMITTEE(hc_ID(PK), housep_ID(PKFK), hc_Fname, hc_Address, hc_Email, hc_Phone_Number)

HOUSEPARENT(housep_ID(PK), accowner_ID(PKFK), housep_Fname, housep_Lname, housep_Address, housep_Email, housep_Phone_Number)

APPLICATION(application_ID(PK), resofficer_ID(PKFK), residence_ID(FK), application_Number, application_Status)

Supertype/Subtype:

OFFICER(officer_ID(PK), officer_Fname, officer_Lname, officer_Address, officer_Email, officer_Phone_Number, officer_Typy)

PAYMENT(payment_ID(PK), student_ID(PKFK), finance_ID(PKFK), date_Of_Payment)

RESIDENCE(residence_ID(PK), residence_ID(PKFK), residence_Name, residence_Address, residence_Requirements, residence_Overview, residence_Type)

COMPANY(company_ID(PK), accowner_ID(FK), company_Name, company_Type)

OFFRESIDENCE(offr_ID(PK), residence_ID(PKFK), num_RoomsAvailable)

ONRESIDENCE(onr_ID(PK), residence_ID(PKFK), num_RoomsAvailable)

FINANCE(finance_ID(PK), officer_ID(PKFK), total_Amount, responsible_for_payments, finance_Email, finance_Telephone_Number)

BURSARYOFFICER(bursary_ID(PK), officer_ID(PKFK), bursary_Available,
bursary_Name, bursary_Email, bursary_Telephone_Number)
RESIDENCEOFFICER(resofficer_ID(PK), officer_ID(PKFK), resofficer_Email,
resofficer_Telephone_Number)
SELPAYMENT(payment_ID(PKFK), payment_Method)
BURSARY(payment_ID(PKFK), bursary_ID(FK))
CLEANER(clean_ID(PK), company_ID(PKFK), clean_Address, clean_Email,
clean_Telephone_Number)
SECURITY(security_ID(PK), company_ID(PKFK), security_Address, security_Email,
security_Telephone_Number)

The Logical design is in a normalized form.

PHASE THREE

Table of content

1. Creation of tables
2. Alter tables
3. Create sequence
4. Drop tables
5. Dropping sequences
6. Create indexes
7. Create view
8. Insert data
9. QUERIES

1. CREATION OF TABLES

The following SQL code will create a table for the administration to use, which consists of the student id, user name and password

```
CREATE TABLE administration (  
    student_id NUMBER NOT NULL,  
    username VARCHAR(50) NOT NULL,  
    password VARCHAR(20) NOT NULL,  
    PRIMARY KEY (student_id),  
    FOREIGN KEY (student_id) REFERENCES student(student_id)  
);
```

The following SQL statements is responsible for creating a new student table. What is needed is a student's id, first name and last name, address, email and phone number belonging to the student.

```
CREATE TABLE student (  
    student_id NUMBER NOT NULL,  
    first_name VARCHAR(255) NOT NULL,  
    last_name VARCHAR(255) NOT NULL,  
    address VARCHAR(255) NOT NULL,  
    email VARCHAR(255) NOT NULL,  
    phone_number VARCHAR(255) NOT NULL,  
    PRIMARY KEY (student_id)  
);
```

The following code will create a student document table that is going to be utilized by an officer. Only the student id is going to be needed from the student table. From an officer, we will require the officer's first and last names, officer address, email, phone number and the officer type

```
CREATE TABLE student_documents (  
    student_id NUMBER NOT NULL,  
    officer_id NUMBER NOT NULL,  
    document_name VARCHAR(50) NOT NULL,  
    document_type VARCHAR(50) NOT NULL,  
    document_size INT NOT NULL,  
    document_date DATE NOT NULL,  
    PRIMARY KEY (student_id, document_name, officer_id)  
);
```

The following statement will create a residence table where a student will choose to either reside on campus or off campus. The following attributes will be needed to create the residence table: Student id, residence name, address, type, requirements and residence overview

```
CREATE TABLE residence (  
    residence_id INT NOT NULL,  
    student_id INT NOT NULL,  
    residence_name VARCHAR(50) NOT NULL,  
    address VARCHAR(50) NOT NULL,  
    residence_type VARCHAR(255) NOT NULL,  
    residence_requirements DECIMAL(10,2) NOT NULL,  
    residence_overview VARCHAR(40) NOT NULL,  
    PRIMARY KEY (residence_id),  
    FOREIGN KEY (student_id) REFERENCES student(student_id)  
);
```

.For off residence

```
CREATE TABLE OFF_residence (  
    residence_id NUMBER NOT NULL,  
    number_of_rooms_available INT NOT NULL,  
    CONSTRAINT residence_PK  
        FOREIGN KEY (residence_id)  
        REFERENCES residence(residence_id)  
);
```

For on residence

```
CREATE TABLE ON_residence (  
    residence_id NUMBER NOT NULL,  
    number_of_rooms_available INT NOT NULL,  
    CONSTRAINT residence_OK  
        FOREIGN KEY (residence_id)  
        REFERENCES residence(residence_id)  
);
```

Payment table is created using the following code. A student id is needed to know for which payment is made by which student, and the payment id to link it to a particular student. The finance id ,date of payment and payment type is also required.

```
CREATE TABLE Payment (  
    payment_id NUMBER PRIMARY KEY,  
    student_id NUMBER NOT NULL,  
    Date_Of_Payment DATE,  
    Payment_Type VARCHAR(1),  
    FOREIGN KEY (student_id) REFERENCES student(student_id)  
);
```

The first type of payment is self-payment and the following lines of code will create the self-payment table

Table SELFPAYMENT created.

The following code will create the bursary table

```
CREATE TABLE BURSARY (  
    payment_id NUMBER NOT NULL REFERENCES payment(payment_id),  
    bursary_Name VARCHAR(20) NOT NULL  
);
```

The officer table will be created by the following lines. An offer id, officer full names, address, email, phone number and officer type is needed.

```
CREATE TABLE officer (  
    officer_ID INT NOT NULL ,  
    officer_Fname VARCHAR(50) NOT NULL,  
    officer_Lname VARCHAR(50) NOT NULL,  
    officer_Address VARCHAR(50) NOT NULL,  
    officer_Email VARCHAR(50) NOT NULL,  
    officer_Phone_Number VARCHAR(50),  
    officer_Type VARCHAR(1),  
    PRIMARY KEY(officer_ID)  
);
```

First type of officer that we have is the finance officer. We will need both the officer id and finance id, the total amount, payment responsibilities, email and telephone number

```
CREATE TABLE finance_officer (  
    officer_id NUMBER NOT NULL,  
    finance_Id INTEGER,  
    CONSTRAINT fk_officer  
    FOREIGN KEY (officer_id) REFERENCES Officer (officer_id),  
    PRIMARY KEY(officer_id, finance_id),  
    total_amount NUMBER NOT NULL,  
    responsible_for_payments VARCHAR(50) NOT NULL,  
    finance_Email VARCHAR(50) NOT NULL,  
    finance_telephone_number VARCHAR(20) NOT NULL  
);
```

Second type is the bursary officer, and to create the table, we need officer id, bursary name, availability, email and bursary telephone number

```
CREATE TABLE bursary_officer (  
    officer_id NUMBER NOT NULL,  
    bursary_Id INTEGER,  
    CONSTRAINT bk_officer  
    FOREIGN KEY (officer_id) REFERENCES Officer (officer_id),  
    PRIMARY KEY(officer_id, bursary_id),  
    bursary_avilaible VARCHAR(50) NOT NULL,  
    bursary_Name VARCHAR(50) NOT NULL,  
    bursary_Email VARCHAR(50) NOT NULL,  
    bursary_telephone_number VARCHAR(20) NOT NULL  
);
```

The third officer we have is the residence officer, to create the table, we need the officer id, res officer email and telephone number

```
CREATE TABLE residence_officer (  
    officer_id NUMBER NOT NULL,  
    resofficer_id INTEGER,  
    CONSTRAINT pk_officer  
    FOREIGN KEY (officer_id) REFERENCES Officer (officer_id),  
    PRIMARY KEY(officer_id, resofficer_id),  
    resofficer_email VARCHAR(50) NOT NULL,  
    resofficer_telephone_number VARCHAR(20) NOT NULL  
);
```

The following code will create the Leadership table using the house committee id, scc id and student id because it has to be one of the students

```
CREATE TABLE leadership (  
    hc_ID INT NOT NULL ,  
    scc_ID INT NOT NULL,  
    student_ID INT NOT NULL,  
    PRIMARY KEY(hc_ID, scc_ID, student_ID)  
);
```

The leader also has to be part of the scc committee. To create the scc table, we need the res officer id, scc full names, address, email and phone number

```
CREATE TABLE SCC (  
    scc_ID INT NOT NULL ,  
    resofficer_ID INT NOT NULL,  
    scc_Fname VARCHAR(50) NOT NULL,  
    scc_Address VARCHAR(50) NOT NULL,  
    scc_Email VARCHAR(50) NOT NULL,  
    scc_Phone_Number VARCHAR(50) NOT NULL,  
    PRIMARY KEY(scc_id, resofficer_id),  
    FOREIGN KEY (resofficer_ID)  
    REFERENCES administration(student_ID)  
);
```

To create the house committee table, we need house parent id, house committee full names, address, email and phone number

```
CREATE TABLE housecommittee (  
    hc_ID INT NOT NULL,  
    housep_ID INT NOT NULL,  
    hc_Fname VARCHAR(255) NOT NULL,  
    hc_Address VARCHAR(255) NOT NULL,  
    hc_Email VARCHAR(255) NOT NULL,  
    hc_Phone_Number VARCHAR(255) NOT NULL,  
    PRIMARY KEY(hc_ID, housep_ID)  
);
```


Each accommodation has one house parent, who is also part of the house committee. To create the house parent table, we need the accommodation id, house parent full names, address, email and phone number

```
CREATE TABLE houseparent (
    housep_ID INT NOT NULL,
    accowner_ID INT NOT NULL,
    housep_Fname VARCHAR(255) NOT NULL,
    housep_Lname VARCHAR(255) NOT NULL,
    housep_Address VARCHAR(255) NOT NULL,
    housep_Email VARCHAR(255) NOT NULL,
    housep_Phone_Number VARCHAR(255) NOT NULL,
    PRIMARY KEY(housep_id, accowner_id)
);
```

The following code is responsible for creating the Accommodation owner table. What's needed to create the table is the application id, res officer id, accommodation owner first name, address, email and phone number

```
CREATE TABLE accommodationowner (
    accowner_ID INT NOT NULL ,
    resofficer_ID INT NOT NULL,
    application_id INT NOT NULL,
    accowner_Fname VARCHAR(50) NOT NULL,
    accowner_Address VARCHAR(50) NOT NULL,
    accowner_Email VARCHAR(50) NOT NULL,
    accowner_Phone_Number VARCHAR(50) NOT NULL,
    PRIMARY KEY(accowner_id, resofficer_id, application_id),
    FOREIGN KEY (resofficer_ID)
    REFERENCES administration(student_ID)
);
```

The creation of the company table is done via the following statements. The accommodation owner id, company name and type is needed

```
CREATE TABLE company (
    company_id NUMBER NOT NULL,
    accowner_id NUMBER NOT NULL,
    company_Name VARCHAR(50) NOT NULL,
    comapny_Type VARCHAR(1) NOT NULL,
    PRIMARY KEY (company_id, accowner_id)
);
```

The company employs cleaners. And to create the cleaner table, we need the company id to identify the company, cleaner name, address, email and telephone number

```
CREATE TABLE cleaner (
    cleaner_id INT NOT NULL,
    company_id INT NOT NULL,
    clean_Address VARCHAR(30) NOT NULL,
    clean_Email VARCHAR(40) NOT NULL,
    clean_telephone_Number VARCHAR(20) NOT NULL,
    PRIMARY KEY(cleaner_id, company_id)
);
```

Another type of company that we have is the security company. To create the security table, we need the company id, security address, email and telephone number

```
CREATE TABLE security (
  sec_id INT NOT NULL,
  company_id INT NOT NULL,
  sec_Address VARCHAR(30) NOT NULL,
  sec_Email VARCHAR(40) NOT NULL,
  sec_telephone_Number VARCHAR(20) NOT NULL,
  PRIMARY KEY(sec_id, company_id)
);
```

For the residence application, to create the table, we will need the res officer id and residence id, application number and application status

```
CREATE TABLE application (
  application_ID INT NOT NULL,
  resofficer_ID INT NOT NULL,
  residence_ID INT NOT NULL,
  application_Number INT NOT NULL,
  application_Status VARCHAR(255) NOT NULL,
  CONSTRAINT fk_resofficer
    FOREIGN KEY (resofficer_ID)
    REFERENCES administration(student_ID),
  CONSTRAINT fk_residence
    FOREIGN KEY (residence_ID)
    REFERENCES residence(residence_ID)
);
```

2. ALTER TABLES

Next, we need to alter the constraint of foreign keys in order to connect the tables because we having primary keys with foreign keys .

```
ALTER TABLE PAYMENT ADD CONSTRAINT fk_finance FOREIGN KEY(FINANCE_ID) REFERENCE FINANCE_OFFICER(FINANCE_ID);
ALTER TABLE ACCOMMODATIONOWNER ADD CONSTRAINT Rk_RESIDENCE_OFFICER FOREIGN KEY(RESIDENCE_ID) REFERENCE RESIDENCE_OFFICER(RESIDENCE_ID);
ALTER TABLE ACCOMMODATIONOWNER ADD CONSTRAINT Ak_APPLICATION FOREIGN KEY(APPLICATION_ID) REFERENCE APPLICATION(APPLICATION_ID);
ALTER TABLE STUDENT_DOCUMENTS ADD CONSTRAINT Ok_OFFICER FOREIGN KEY(OFFICER_ID) REFERENCE OFFICER(OFFICER_ID);
ALTER TABLE STUDENT_DOCUMENTS ADD CONSTRAINT Sk_STUDENT FOREIGN KEY(STUDENT_ID) REFERENCE STUDENT(STUDENT_ID);
ALTER TABLE OFF_RESIDENCE ADD CONSTRAINT Ok_RESIDENCE FOREIGN KEY(RESIDENCE_ID) REFERENCE RESIDENCE(RESIDENCE_ID);
ALTER TABLE ON_RESIDENCE ADD CONSTRAINT Fk_RESIDENCE FOREIGN KEY(RESIDENCE_ID) REFERENCE RESIDENCE(RESIDENCE_ID);
ALTER TABLE OFF_RESIDENCE ADD CONSTRAINT Ok_RESIDENCE FOREIGN KEY(RESIDENCE_ID) REFERENCE RESIDENCE(RESIDENCE_ID);
ALTER TABLE PAYMENT ADD CONSTRAINT Sk_STUDENT FOREIGN KEY(STUDENT_ID) REFERENCE STUDENT(STUDENT_ID);
ALTER TABLE COMPANY ADD CONSTRAINT Ak_ACCOMMODATIONOWNER FOREIGN KEY(ACCOWNER_ID) REFERENCE ACCOMMODATIONOWNER(ACCOWNER_ID);
ALTER TABLE SECURITY ADD CONSTRAINT Sk_COMPANY FOREIGN KEY(COMPANY_ID) REFERENCE COMPANY(COMPANY_ID);
ALTER TABLE CLEANER ADD CONSTRAINT Ck_CLEANER FOREIGN KEY(CLEANER_ID) REFERENCE CLEANER(CLEANER_ID);
ALTER TABLE HOUSEPARENT ADD CONSTRAINT Hk_ACCOMMODATIONOWNER FOREIGN KEY(ACCOWNER_ID) REFERENCE ACCOMMODATIONOWNER(ACCOWNER_ID);
ALTER TABLE HOUSECOMMITTEE ADD CONSTRAINT Hk_HOUSEP FOREIGN KEY(HOUSEP_ID) REFERENCE HOUSEPARENT(HOUSEP_ID);
ALTER TABLE LEADERSHIP ADD CONSTRAINT Sk_SCC FOREIGN KEY(SCC_ID) REFERENCE SCC(SCC_ID);
ALTER TABLE LEADERSHIP ADD CONSTRAINT Hk_HOUSECOMMITTEE FOREIGN KEY(HC_ID) REFERENCE HOUSECOMMITTEE(HC_ID);
ALTER TABLE LEADERSHIP ADD CONSTRAINT Dk_STUDENT FOREIGN KEY(STUDENT_ID) REFERENCE STUDENT(STUDENT_ID);
```


3. CREATE SEQUENCES

The creation of sequences ensures that each table's primary key has a unique value, and that value is incremented by 1 for each attribute created

Create student_id value

```
CREATE SEQUENCE student_id_value
start with 1
INCREMENT by 1
minvalue 1
maxvalue 10000000
CYCLE;
```

Create application_id value

```
CREATE SEQUENCE application_id_value
start with 1
INCREMENT by 1
minvalue 1
maxvalue 10000
CYCLE;
```

Create payment_id value

```
CREATE SEQUENCE payment_id_value
start with 1
INCREMENT by 1
minvalue 1
maxvalue 10000
CYCLE;
```

Create residence_id value

```
CREATE SEQUENCE residence_id_value
start with 1
INCREMENT by 1
minvalue 1
maxvalue 100
CYCLE;
```

Create resofficer_id value

```
CREATE SEQUENCE resofficer_id_value
start with 1
INCREMENT by 1
minvalue 1
maxvalue 100000
CYCLE;
```

Create officer_id value

```
CREATE SEQUENCE officer_id_value
start with 1
INCREMENT by 1
minvalue 1
maxvalue 100000
CYCLE;
```

Create finance_id value

```
CREATE SEQUENCE finance_id_value
start with 1
INCREMENT by 1
minvalue 1
maxvalue 100000
CYCLE;
```

Create bursary_id value

```
CREATE SEQUENCE bursary_id_value
start with 1
INCREMENT by 1
minvalue 1
maxvalue 100000
CYCLE;
```

Create company_id value

```
CREATE SEQUENCE company_id_value
start with 1
INCREMENT by 1
minvalue 1
maxvalue 1000
CYCLE;
```

Create accowner_id value

```
CREATE SEQUENCE accowner_id_value
start with 1
INCREMENT by 1
minvalue 1
maxvalue 100
CYCLE;
```

Create hc_id value

```
CREATE SEQUENCE hc_id_value
start with 1
INCREMENT by 1
minvalue 1
maxvalue 1000
CYCLE;
```

Create housep_id value

```
CREATE SEQUENCE housep_id_value
start with 1
INCREMENT by 1
minvalue 1
maxvalue 100
CYCLE;
```

Create scc_id value

```
CREATE SEQUENCE scc_id_value
start with 1
INCREMENT by 1
minvalue 1
maxvalue 100000000
CYCLE;
```

Create cleaner_id value

```
CREATE SEQUENCE cleaner_id_value  
start with 1  
INCREMENT by 1  
minvalue 1  
maxvalue 1000  
CYCLE;
```

Create security_id value

```
CREATE SEQUENCE security_id_value  
start with 1  
INCREMENT by 1  
minvalue 1  
maxvalue 1000  
CYCLE;
```

4. DROP TABLES

The DROP TABLE command will remove the table structure and associated indexes, triggers, constraints and permissions from all tables

```
DROP TABLE ACCOMMODATIONOWNER;  
DROP TABLE ADMISTRATION;  
DROP TABLE APPLICATION;  
DROP TABLE BURSARY;  
DROP TABLE BURSARY_OFFICER;  
DROP TABLE CLEANER;  
DROP TABLE COMPANY;  
DROP TABLE FINANCE_OFFICER;  
DROP TABLE HOUSECOMMITTEE;  
DROP TABLE HOUSEPARENT;  
DROP TABLE LEADERSHIP;  
DROP TABLE OFF_RESIDENCE;  
DROP TABLE OFFICER;  
DROP TABLE ON_RESIDENCE;  
DROP TABLE PAYMENT;  
DROP TABLE RESIDENCE;  
DROP TABLE RESIDENCE_OFFICER;  
DROP TABLE SCC;  
DROP TABLE SECURITY;  
DROP TABLE SELFPAYMENT;  
DROP TABLE STUDENT;  
DROP TABLE STUDENT_DOCUMENTS;
```

5. DROP SEQUENCE

DROP SEQUENCE statement removes all sequences created from the database

```
DROP SEQUENCE ACCOWNER_ID_VALUE;  
DROP SEQUENCE APPLICATION_ID_VALUE;  
DROP SEQUENCE BURSARY_ID_VALUE;  
DROP SEQUENCE CLEANER_ID_VALUE;  
DROP SEQUENCE COMAPANY_ID_VALUE;  
DROP SEQUENCE HC_ID_VALUE;  
DROP SEQUENCE HOUSEP_ID_VALUE;  
DROP SEQUENCE OFFICER_ID_VALUE;  
DROP SEQUENCE PAYMENT_ID_VALUE;  
DROP SEQUENCE RESIDENCE_ID_VALUE;  
DROP SEQUENCE RESOFFICER_ID_VALUE;  
DROP SEQUENCE SCC_ID_VALUE;  
DROP SEQUENCE SECURITY_ID_VALUE;  
DROP SEQUENCE STUDENT_ID_VALUE;
```

6. CREATE INDEXES

Indexes are created with the purpose of quicker retrieval of data from the tables

```
CREATE UNIQUE INDEX STUDENT_PK ON STUDENT(STUDENT_ID);
CREATE UNIQUE INDEX RESIDENCE_PK ON RESIDENCE(RESIDENCE_ID);
CREATE UNIQUE INDEX APPLICATION_PK ON APPLICATION(APPLICATION_ID);
CREATE UNIQUE INDEX BURSARY_PK ON BURSARY(BURSARY_ID);
CREATE UNIQUE INDEX OFF_RESIDENCE_PK ON RESIDENCE(RESIDENCE_ID);
CREATE UNIQUE INDEX ON_RESIDENCE_PK ON RESIDENCE(RESIDENCE_ID);
CREATE UNIQUE INDEX SELFPAYMENT_PK ON PAYMENT(PAYMENT_ID);
CREATE UNIQUE INDEX ADMINISTRATION_PK ON STUDENT(STUDENT_ID);
CREATE UNIQUE INDEX PAYMENT_PK ON PAYMENT(PAYMENT_ID);
CREATE UNIQUE INDEX OFFICER_PK ON OFFICER(OFFICER_ID);
CREATE UNIQUE INDEX STUDENT_DOCUMENTS_PK ON STUDENT(STUDENT_ID);
CREATE UNIQUE INDEX FINANCE_OFFICER_PK ON FINANCE_OFFICER(FINANCE_ID);
CREATE UNIQUE INDEX BURSARY_OFFICER_PK ON BURSARY_OFFICER(BURSARY_ID);
CREATE UNIQUE INDEX RESIDENCE_OFFICER_PK ON RESOFFICER(RESOFFICER_ID);
CREATE UNIQUE INDEX SCC_PK ON SCC(SCC_ID);
CREATE UNIQUE INDEX ACCOMMODATIONOWNER_PK ON ACCOMMODATIONOWNER(ACCONNER_ID);
CREATE UNIQUE INDEX COMPANY_PK ON COMPANY(COMPANY_ID);
CREATE UNIQUE INDEX HOUSEPARENT_PK ON HOUSEPARENT(HOUSEP_ID);
CREATE UNIQUE INDEX CLEANER_PK ON CLEANER(CLEANER_ID);
CREATE UNIQUE INDEX SECURITY_PK ON SECURITY(SECURITY_ID);
CREATE UNIQUE INDEX HOUSECOMMITTEE_PK ON HOUSECOMMITTEE(HC_ID);
CREATE UNIQUE INDEX LEADERSHIP_PK ON STUDENT(STUDENT_ID);
```

7. CREATE VIEW

This creates a virtual table for the two table which is student and residence

```
CREATE VIEW "STUDENT APPLICATION" AS
SELECT student.student_id, residence.residence_name, application.application_number, application.application_status, application.date_applied
FROM student, residence, application
WHERE student.student_id=residence.student_id ;
```

8. INSERT

Inserting data into their respective tables

INSERT INTO ADMINISTRATION

Data is inserted into the administration table

```
INSERT INTO administration
VALUES (STUDENT_ID_VALUE.nextval, '2345246','Tabo@23j');
INSERT INTO administration
VALUES (STUDENT_ID_VALUE.nextval, '2317869','lEbo@12');
INSERT INTO administration
VALUES (STUDENT_ID_VALUE.nextval, '3452186','Mp@12majko');
INSERT INTO administration
VALUES (STUDENT_ID_VALUE.nextval, '3547264','K@12yamba');
```


INSERT INTO RESIDENCE

Inserting data to the residence table

```
INSERT INTO residence
VALUES (RESIDENCE_ID_VALUE.nextval,STUDENT_ID_VALUE.nextval, 'BOHLALE','2 VAAL', 'B',0.65,'TOP Q');
INSERT INTO residence
VALUES (RESIDENCE_ID_VALUE.nextval,STUDENT_ID_VALUE.nextval, 'BUHLE','2 VAAL', 'B',0.65,'LOW Q');
INSERT INTO residence
VALUES (RESIDENCE_ID_VALUE.nextval,STUDENT_ID_VALUE.nextval, 'FARANANE','2 VAAL', 'B',0.65,'MIDDLE Q');
INSERT INTO residence
VALUES (RESIDENCE_ID_VALUE.nextval,STUDENT_ID_VALUE.nextval, 'KHUMBA','2 VAAL', 'B',0.65,'TOPEST Q');
```

INSERT PAYMENT

Inserting data to the payment table

```
INSERT INTO payment
VALUES (payment_id_value.nextval,1,TO_DATE('2023/09/12','YYYY,MM,DD'),'P',1);
INSERT INTO payment
VALUES (payment_id_value.nextval,1,TO_DATE('2023/09/23','YYYY,MM,DD'),'P',1);
INSERT INTO payment
VALUES (payment_id_value.nextval,1,TO_DATE('2023/09/17','YYYY,MM,DD'),'P',1);
INSERT INTO payment
VALUES (payment_id_value.nextval,1,TO_DATE('2023/09/20','YYYY,MM,DD'),'P',1);
```

INSERT ADMINISTRATION

Insert data to the administration

```
INSERT INTO administration
VALUES (STUDENT_ID_VALUE.nextval, '2345246','Tabo@23j');
INSERT INTO administration
VALUES (STUDENT_ID_VALUE.nextval, '2317869','lEbo@12');
INSERT INTO administration
VALUES (STUDENT_ID_VALUE.nextval, '3452186','Mp@12majko');
INSERT INTO administration
VALUES (STUDENT_ID_VALUE.nextval, '3547264','K@12yamba');
```

INSERT STUDENT

Inserting data to the student table

```
INSERT INTO STUDENT
VALUES (STUDENT_ID_VALUE.nextval, 'Thabo','Bobe','house34 kapa','thabobobe@gmail.com','0765893421');
INSERT INTO STUDENT
VALUES (STUDENT_ID_VALUE.nextval, 'Lebo','Rama','house89 evaton','leborama@gmail.com','0658937365');
INSERT INTO STUDENT
VALUES (STUDENT_ID_VALUE.nextval, 'Mpeo','Majake','house98 vaal','mpeomajake@gmail.com','0629687905');
INSERT INTO STUDENT
VALUES (STUDENT_ID_VALUE.nextval, 'Katleho','Yamba','house12 vaal','katlehoyamba@gmail.com','0895893421');
```

INSERT SELF PAYMENT

Inserting data to the self payment table

```
INSERT INTO selfpayment
VALUES (payment_id_value.nextval,'EFT');
INSERT INTO selfpayment
VALUES (payment_id_value.nextval,'EFT');
INSERT INTO selfpayment
VALUES (payment_id_value.nextval,'CASH');
INSERT INTO selfpayment
VALUES (payment_id_value.nextval,'BANK');
```

INSERTING ALL DATA

Inserting data for the bursary, Student documents, Finance officer, Officer, bursary officer and residence officer tables

```

INSERT INTO "BURSARY" (PAYMENT_ID, BURSARY_NAME, BURSARY_ID) VALUES (PAYMENT_ID_VALUE.nextval, 'NSFAS', '1')
INSERT INTO "BURSARY" (PAYMENT_ID, BURSARY_NAME, BURSARY_ID) VALUES (PAYMENT_ID_VALUE.nextval, 'NSFAS', '2')
INSERT INTO "BURSARY" (PAYMENT_ID, BURSARY_NAME, BURSARY_ID) VALUES (PAYMENT_ID_VALUE.nextval, 'BBD', '3')
INSERT INTO "BURSARY" (PAYMENT_ID, BURSARY_NAME, BURSARY_ID) VALUES (PAYMENT_ID_VALUE.nextval, 'FNB', '4')

INSERT INTO "STUDENT_DOCUMENTS" (STUDENT_ID, OFFICER_ID, DOCUMENT_NAME, DOCUMENT_TYPE, DOCUMENT_SIZE, DOCUMENT_DATE) VALUES (STUDENT_ID_VALUE.nextval, OFFICER_ID_VALUE.nextval, 'STUDENT_DOCUMENT', 'DOCUMENT', 1000, SYSDATE)
INSERT INTO "STUDENT_DOCUMENTS" (STUDENT_ID, OFFICER_ID, DOCUMENT_NAME, DOCUMENT_TYPE, DOCUMENT_SIZE, DOCUMENT_DATE) VALUES (STUDENT_ID_VALUE.nextval, OFFICER_ID_VALUE.nextval, 'STUDENT_DOCUMENT', 'DOCUMENT', 1000, SYSDATE)
INSERT INTO "STUDENT_DOCUMENTS" (STUDENT_ID, OFFICER_ID, DOCUMENT_NAME, DOCUMENT_TYPE, DOCUMENT_SIZE, DOCUMENT_DATE) VALUES (STUDENT_ID_VALUE.nextval, OFFICER_ID_VALUE.nextval, 'STUDENT_DOCUMENT', 'DOCUMENT', 1000, SYSDATE)
INSERT INTO "STUDENT_DOCUMENTS" (STUDENT_ID, OFFICER_ID, DOCUMENT_NAME, DOCUMENT_TYPE, DOCUMENT_SIZE, DOCUMENT_DATE) VALUES (STUDENT_ID_VALUE.nextval, OFFICER_ID_VALUE.nextval, 'STUDENT_DOCUMENT', 'DOCUMENT', 1000, SYSDATE)

INSERT INTO "OFFICER" (OFFICER_ID, OFFICER_FNAME, OFFICER_LNAME, OFFICER_ADDRESS, OFFICER_EMAIL, OFFICER_PHONE_NUMBER, OFFICER_TYPE) VALUES (OFFICER_ID_VALUE.nextval, OFFICER_FNAME_VALUE.nextval, OFFICER_LNAME_VALUE.nextval, OFFICER_ADDRESS_VALUE.nextval, OFFICER_EMAIL_VALUE.nextval, OFFICER_PHONE_NUMBER_VALUE.nextval, OFFICER_TYPE_VALUE.nextval)
INSERT INTO "OFFICER" (OFFICER_ID, OFFICER_FNAME, OFFICER_LNAME, OFFICER_ADDRESS, OFFICER_EMAIL, OFFICER_PHONE_NUMBER, OFFICER_TYPE) VALUES (OFFICER_ID_VALUE.nextval, OFFICER_FNAME_VALUE.nextval, OFFICER_LNAME_VALUE.nextval, OFFICER_ADDRESS_VALUE.nextval, OFFICER_EMAIL_VALUE.nextval, OFFICER_PHONE_NUMBER_VALUE.nextval, OFFICER_TYPE_VALUE.nextval)
INSERT INTO "OFFICER" (OFFICER_ID, OFFICER_FNAME, OFFICER_LNAME, OFFICER_ADDRESS, OFFICER_EMAIL, OFFICER_PHONE_NUMBER, OFFICER_TYPE) VALUES (OFFICER_ID_VALUE.nextval, OFFICER_FNAME_VALUE.nextval, OFFICER_LNAME_VALUE.nextval, OFFICER_ADDRESS_VALUE.nextval, OFFICER_EMAIL_VALUE.nextval, OFFICER_PHONE_NUMBER_VALUE.nextval, OFFICER_TYPE_VALUE.nextval)
INSERT INTO "OFFICER" (OFFICER_ID, OFFICER_FNAME, OFFICER_LNAME, OFFICER_ADDRESS, OFFICER_EMAIL, OFFICER_PHONE_NUMBER, OFFICER_TYPE) VALUES (OFFICER_ID_VALUE.nextval, OFFICER_FNAME_VALUE.nextval, OFFICER_LNAME_VALUE.nextval, OFFICER_ADDRESS_VALUE.nextval, OFFICER_EMAIL_VALUE.nextval, OFFICER_PHONE_NUMBER_VALUE.nextval, OFFICER_TYPE_VALUE.nextval)

INSERT INTO "FINANCE_OFFICER" (OFFICER_ID, FINANCE_ID, TOTAL_AMOUNT, RESPONSIBLE_FOR_PAYMENTS, FINANCE_EMAIL, FINANCE_PHONE_NUMBER, FINANCE_TYPE) VALUES (OFFICER_ID_VALUE.nextval, FINANCE_ID_VALUE.nextval, TOTAL_AMOUNT_VALUE.nextval, RESPONSIBLE_FOR_PAYMENTS_VALUE.nextval, FINANCE_EMAIL_VALUE.nextval, FINANCE_PHONE_NUMBER_VALUE.nextval, FINANCE_TYPE_VALUE.nextval)
INSERT INTO "FINANCE_OFFICER" (OFFICER_ID, FINANCE_ID, TOTAL_AMOUNT, RESPONSIBLE_FOR_PAYMENTS, FINANCE_EMAIL, FINANCE_PHONE_NUMBER, FINANCE_TYPE) VALUES (OFFICER_ID_VALUE.nextval, FINANCE_ID_VALUE.nextval, TOTAL_AMOUNT_VALUE.nextval, RESPONSIBLE_FOR_PAYMENTS_VALUE.nextval, FINANCE_EMAIL_VALUE.nextval, FINANCE_PHONE_NUMBER_VALUE.nextval, FINANCE_TYPE_VALUE.nextval)
INSERT INTO "FINANCE_OFFICER" (OFFICER_ID, FINANCE_ID, TOTAL_AMOUNT, RESPONSIBLE_FOR_PAYMENTS, FINANCE_EMAIL, FINANCE_PHONE_NUMBER, FINANCE_TYPE) VALUES (OFFICER_ID_VALUE.nextval, FINANCE_ID_VALUE.nextval, TOTAL_AMOUNT_VALUE.nextval, RESPONSIBLE_FOR_PAYMENTS_VALUE.nextval, FINANCE_EMAIL_VALUE.nextval, FINANCE_PHONE_NUMBER_VALUE.nextval, FINANCE_TYPE_VALUE.nextval)
INSERT INTO "FINANCE_OFFICER" (OFFICER_ID, FINANCE_ID, TOTAL_AMOUNT, RESPONSIBLE_FOR_PAYMENTS, FINANCE_EMAIL, FINANCE_PHONE_NUMBER, FINANCE_TYPE) VALUES (OFFICER_ID_VALUE.nextval, FINANCE_ID_VALUE.nextval, TOTAL_AMOUNT_VALUE.nextval, RESPONSIBLE_FOR_PAYMENTS_VALUE.nextval, FINANCE_EMAIL_VALUE.nextval, FINANCE_PHONE_NUMBER_VALUE.nextval, FINANCE_TYPE_VALUE.nextval)

INSERT INTO "BURSARY_OFFICER" (OFFICER_ID, BURSARY_ID, BURSARY_AVAILABLE, BURSARY_NAME, BURSARY_EMAIL, BURSARY_PHONE_NUMBER, BURSARY_TYPE) VALUES (OFFICER_ID_VALUE.nextval, BURSARY_ID_VALUE.nextval, BURSARY_AVAILABLE_VALUE.nextval, BURSARY_NAME_VALUE.nextval, BURSARY_EMAIL_VALUE.nextval, BURSARY_PHONE_NUMBER_VALUE.nextval, BURSARY_TYPE_VALUE.nextval)
INSERT INTO "BURSARY_OFFICER" (OFFICER_ID, BURSARY_ID, BURSARY_AVAILABLE, BURSARY_NAME, BURSARY_EMAIL, BURSARY_PHONE_NUMBER, BURSARY_TYPE) VALUES (OFFICER_ID_VALUE.nextval, BURSARY_ID_VALUE.nextval, BURSARY_AVAILABLE_VALUE.nextval, BURSARY_NAME_VALUE.nextval, BURSARY_EMAIL_VALUE.nextval, BURSARY_PHONE_NUMBER_VALUE.nextval, BURSARY_TYPE_VALUE.nextval)
INSERT INTO "BURSARY_OFFICER" (OFFICER_ID, BURSARY_ID, BURSARY_AVAILABLE, BURSARY_NAME, BURSARY_EMAIL, BURSARY_PHONE_NUMBER, BURSARY_TYPE) VALUES (OFFICER_ID_VALUE.nextval, BURSARY_ID_VALUE.nextval, BURSARY_AVAILABLE_VALUE.nextval, BURSARY_NAME_VALUE.nextval, BURSARY_EMAIL_VALUE.nextval, BURSARY_PHONE_NUMBER_VALUE.nextval, BURSARY_TYPE_VALUE.nextval)
INSERT INTO "BURSARY_OFFICER" (OFFICER_ID, BURSARY_ID, BURSARY_AVAILABLE, BURSARY_NAME, BURSARY_EMAIL, BURSARY_PHONE_NUMBER, BURSARY_TYPE) VALUES (OFFICER_ID_VALUE.nextval, BURSARY_ID_VALUE.nextval, BURSARY_AVAILABLE_VALUE.nextval, BURSARY_NAME_VALUE.nextval, BURSARY_EMAIL_VALUE.nextval, BURSARY_PHONE_NUMBER_VALUE.nextval, BURSARY_TYPE_VALUE.nextval)

INSERT INTO "RESIDENCE_OFFICER" (OFFICER_ID, RESOFFICER_ID, RESOFFICER_EMAIL, RESOFFICER_PHONE_NUMBER, RESOFFICER_TYPE) VALUES (OFFICER_ID_VALUE.nextval, RESOFFICER_ID_VALUE.nextval, RESOFFICER_EMAIL_VALUE.nextval, RESOFFICER_PHONE_NUMBER_VALUE.nextval, RESOFFICER_TYPE_VALUE.nextval)
INSERT INTO "RESIDENCE_OFFICER" (OFFICER_ID, RESOFFICER_ID, RESOFFICER_EMAIL, RESOFFICER_PHONE_NUMBER, RESOFFICER_TYPE) VALUES (OFFICER_ID_VALUE.nextval, RESOFFICER_ID_VALUE.nextval, RESOFFICER_EMAIL_VALUE.nextval, RESOFFICER_PHONE_NUMBER_VALUE.nextval, RESOFFICER_TYPE_VALUE.nextval)
INSERT INTO "RESIDENCE_OFFICER" (OFFICER_ID, RESOFFICER_ID, RESOFFICER_EMAIL, RESOFFICER_PHONE_NUMBER, RESOFFICER_TYPE) VALUES (OFFICER_ID_VALUE.nextval, RESOFFICER_ID_VALUE.nextval, RESOFFICER_EMAIL_VALUE.nextval, RESOFFICER_PHONE_NUMBER_VALUE.nextval, RESOFFICER_TYPE_VALUE.nextval)
INSERT INTO "RESIDENCE_OFFICER" (OFFICER_ID, RESOFFICER_ID, RESOFFICER_EMAIL, RESOFFICER_PHONE_NUMBER, RESOFFICER_TYPE) VALUES (OFFICER_ID_VALUE.nextval, RESOFFICER_ID_VALUE.nextval, RESOFFICER_EMAIL_VALUE.nextval, RESOFFICER_PHONE_NUMBER_VALUE.nextval, RESOFFICER_TYPE_VALUE.nextval)

```

9.QUERIES

LIMITATION OF ROWS AND COLUMNS

To limit the number of rows or columns, we used the CONCAT function to combine the first name and last name into one column, Full names

```

SELECT STUDENT_ID, CONCAT(FIRST_NAME, LAST_NAME) "FULL_NAME", ADDRESS, EMAIL, PHONE_NUMBER
FROM STUDENT

```

OUTPUT

STUDENT_ID	FULL_NAME	ADDRESS	EMAIL	PHONE_NUMBER
1	5 SeanClouds	house 78 wherever	seanclouds@yahoo.com	0626323212
2	1 ThaboBobe	house34 kapa	thabobobe@gmail.com	0765893421
3	2 LeboRama	house89 evaton	leborama@gmail.com	0658937365
4	3 MpeoMajake	house98 vaal	mpeomajake@gmail.com	0629687905
5	4 KatlehoYamba	house12 vaal	katlehoyamba@gmail.com	0895893421

SORTING

Sort the officer last name in alphabetical order

```

SELECT *
FROM OFFICER
ORDER BY officer_lname ASC, OFFICER_FNAME DESC;

```

OUTPUT

OFFICER_ID	OFFICER_FNAME	OFFICER_LNAME	OFFICER_ADDRESS	OFFICER_EMAIL	OFFICER_PHONE_NUMBER	OFFICER_TYPE
1	2 MPHO	KITE	HOUSE 123 SPR	MPHO@GMAIL	0634567623	0
2	4 KAT	REA	HRP 123	KATREA@GMAIL.COM	0761268369	0
3	3 SAMKELO	SIPHO	APER 64	SAMKELO@GAMIL.COM	0752398745	0
4	1 SISI	THISA	HOUSE 12 VAAL	SISI@GMAIL	0784537648	0

LIKE,AND and OR

The LIKE functionality searches the database according to the user specification, either using key words, or typing the full word to retrieve data related to that word

```
SELECT *  
FROM selfpayment  
WHERE (payment_method LIKE '%EFT%');
```

OUTPUT

PAYMENT_ID	PAYMENT_METHOD
1	15 EFT
2	16 EFT

```
SELECT *  
FROM RESIDENCE  
WHERE (ADDRESS LIKE '%VANDERS%'OR residence_type LIKE '%B%' AND residence_overview ='TOP Q');
```

OUTPUT

RESIDENCE_ID	STUDENT_ID	RESIDENCE_NAME	ADDRESS	RESIDENCE_TYPE	RESIDENCE_REQUIREMENTS	RESIDENCE_OVERVIEW
1	9	5 Ebokhuzin	VANDARS	B		0.65 TOP Q
2	5	1 BOHLALE	VANDERS	B		0.65 TOP Q
3	6	2 BUHLE	VANDERS	B		0.62 LOW Q

ROUND OR TRUNC

Round function rounds off the decimal number to 2 decimal places

```
SELECT ROUND(TOTAL_AMOUNT,2) "FINAL COST"  
FROM finance_officer
```

OUTPUT

FINAL COST
1 3500.56
2 5000.24
3 5120.46
4 4501.37

DATE FUNCTIONS

Date function uses the date format to perform operations, like to calculate the duration between two separate dates

```
SELECT PAYMENT_ID, ROUND(MONTHS_BETWEEN(SYSDATE,DATE_OF_PAYMENT),0) "DURATION SINCE PAYMENT"  
FROM PAYMENT
```

OUTPUT

PAYMENT_ID	DURATION SINCE PAYMENT
1	15 -4
2	16 -4
3	17 -4
4	18 -4

```
SELECT APPLICATION_NUMBER, Extract(MONTH FROM  
DATE_APPLIED) AS MONTHAPPLIED FROM APPLICATION;
```

OUTPUT

APPLICATION_NU...	MONTH APPLIED
1	12 9
2	4 9
3	6 9
4	2 9

AGGREGATE FUNCTION

The SUM aggregate function calculates the sum of the values in a common common with numerical values

```
SELECT SUM(NUMBER_OF_ROOMS_AVAILABLE)
FROM OFF_RESIDENCE
```

OUTPUT

SUM(NUMBER_OF_ROOMS_AVAILABLE)
1 4628

GROUP BY AND HAVING

GROUP BY and HAVING queries will group data according to the user specification such as grouping students according to the type of bursary they have. HAVING will retrieve the data containing the key word

```
SELECT bursary_name,
       COUNT(*) AS number_of_students
FROM bursary
GROUP BY bursary_name
HAVING BURSARY_NAME = 'NSFAS';
```

OUTPUT

BURSARY_NAME	NUMBER_OF_STUDENTS
1 NSFAS	2

JOINS

INNER JOIN function joins two tables together using a common attribute

```
SELECT student.student_id, student.first_name, student.last_name, residence.residence_name, residence.residence_overview
FROM student
INNER JOIN residence
ON student.student_id = residence.student_id;
```

OUTPUT

STUDENT_ID	FIRST_NAME	LAST_NAME	RESIDENCE_NAME	RESIDENCE_OVERVIEW
1	Thabo	Bobbe	BOHLALE	TOP Q
2	Lebo	Rama	BUHLE	LOW Q
3	Mpeo	Majake	FARANANE	MIDDLE Q
4	Katleho	Yamba	KHUMBA	TOPEST Q
5	Sean	Clouds	Ebokhuzin	TOP Q

SUB-QUERIES

This nested function will retrieve data from the residence table and display the residence requirements

```
SELECT *
FROM residence
WHERE residence_id IN (
    SELECT residence_id
    FROM residence
    WHERE residence_requirements = 0.65
```

OUTPUT

RESIDENCE_ID	STUDENT_ID	RESIDENCE_NAME	ADDRESS	RESIDENCE_TYPE	RESIDENCE_REQUIREMENTS	RESIDENCE_OVERVIEW
1	9	5 Ebokhuzin	VANDARS	B	0.65	TOP Q
2	5	1 BOHLALE	VANDERS	B	0.65	TOP Q
3	6	2 BUHLE	VANDERS	B	0.62	LOW Q

VARIABLE AND CHARACTER FUNCTION

This variable and character function will make the first letter of the word, a capital letter, and leave the rest as small letters, using INITCAP

Worksheet	Query Builder
<pre>SELECT INITCAP('APPROVED') FROM application;</pre>	

OUTPUT

	INITCAP('APPROVED')
1	Approved
2	Approved
3	Approved
4	Approved

EXTRA FUNCTIONALITY

This extra functionality will retrieve the maximum amount within a particular student in the database

<pre>SELECT MAX(total_amount) "MOST PAYING STUDENT" FROM finance_officer</pre>	
--	--

OUTPUT

	MOST PAYING STUDENT
1	5120.456

This extra function will retrieve the system current date and show the duration since the last payment

<pre>SELECT PAYMENT_ID, ROUND(MONTHS_BETWEEN(SYSDATE, DATE_OF_PAYMENT), 0) "DURATION SINCE PAYMENT" FROM PAYMENT</pre>	
--	--

OUTPUT

	PAYMENT_ID	DURATION SINCE PAYMENT
1	15	-4
2	16	-4
3	17	-4
4	18	-4

END OF ASSIGNMENT!!