Database Systems and Security

7CI022

Lecturer- Mr Julius Odede

Group Assignment

Author

Alpha Osman Bah

Scenario

Mr Osman owns a small private hospital in a rural area in West Midlands in the United Kingdom. Mr Osman, new to this industry, is encountering trouble managing patient records, appointments, staff, etc. Dr Julius suggested a database to improve efficiency. The database would save him time in processing patients, appointment scheduling, staff records, etc. Mr Osman agreed and asked for help finding an expert to design a database with six to seven entities.

Dr Julius asked Alpha to build a database with these requirements. The database must have only six entities as it is a small hospital, and they need to save money as best as possible.

- Establishing all foreign keys, primary keys and attributes in the database.
- Normalising the database to improve efficiency and avoid data redundancy.
- Ensuring all the entities' relationships are clearly examined and stated. The design must clearly be explained to the owner.
- The project must explain database security, integrity, and the ethical responsibilities of storing staff and client data.
- Finally, an ER model using Oracle Data Modeller must be created and clearly presented to the owner.

Organisation of the Study

This study will be divided into 5 chapters. Chapter 1 will present the problem and discuss the entities, attributes, primary and foreign keys. Relationships will also be established and discussed. In chapter 2, will cover normalisation, its importance, and 1NF, 2NF and 3NF. Chapter 3 details the implementation phase, including SQL statements to create, populate, and query the tables. Each table will have at least one entry, and the chapter will demonstrate queries to retrieve data and Join operations. Chapter 4 will outline database security, recovery, integrity, ethics and data governance. These measures will secure the database and ensure data recovery. Chapter 5 is our conclusion.

Methodology

ER models will be designed using the Oracle SQL Data Modeller, transformed into a relational model and scripts will be generated for Oracle Live SQL. Oracle SQL Data Modeller will be used to design relationships, entities and attributes, presented in the relevant chapters.

Significance of the Project

This project demonstrates the importance of a structured database and outlines system security and governance. This will improve the hospital's efficiency and save time and money.

Limitation of the Project

This school project limited to six entities which is unlikely in a real-world setting. A real-world hospital database might require 15 to 20 entities for optimal efficiency. However, the model will provide the fundamental principles of an efficient database.

Chapter 1

1.0 Overview

This chapter details the logical model built using Oracle SQL Data Modeller. The entities are - Patients, Staff, Appointment, Department, Medication and Room.

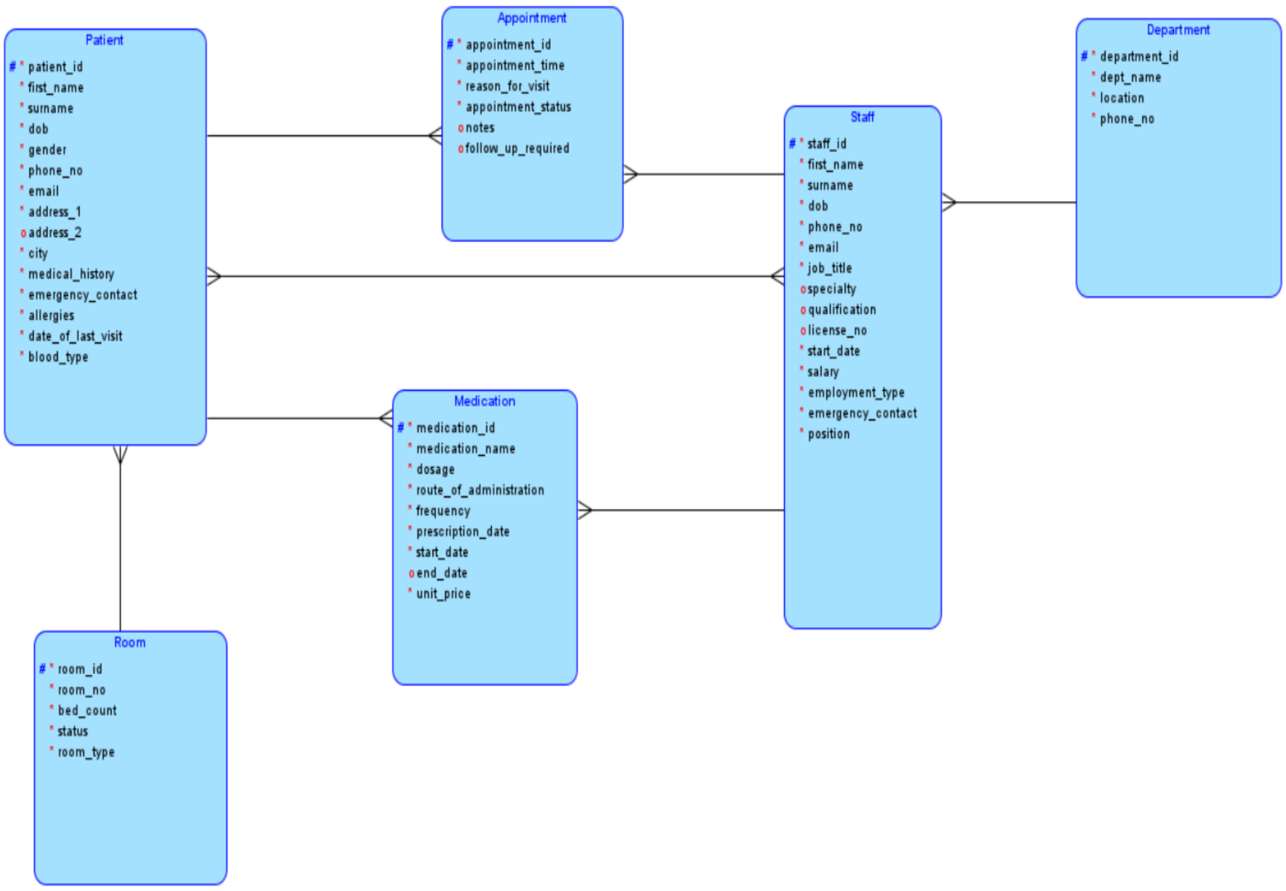


Fig 1: Logical Model

Patient (attributes)- patient_id (primary key), first_name, surname, dob (date of birth), gender, phone_no, email, address_1, address_2, city, medical_history, emergency_contact, allergies, date_of_last_visit, blood_type.

Appointment (attributes)- appointment_id (primary key), appointment_time, reason_for_visit, appointment_status (scheduled, cancelled, completed, no show), notes, follow_required (yes or no).

Staff (attributes)- staff_id (primary key), first_name, surname, dob, phone_no, email, job_title, specialty, qualification, license_no, start_date, salary, employment_type (permanent, parttime, temporary), emergency_contact, position.

Department (attributes)- department_id, dept_name, location, phone_no.

Room (attributes)- room_id (primary key), room_no, bed_count, status (available, occupied, full, not in use, maintenance), room_type (single, shared, ICU).

Medication (attributes)- medication_id (primary key), medication_name, dosage, route_of_administration, frequency, prescription_date, start_date, end_date, unit_price.

1.1 Relational Model and Relationships among the Entities

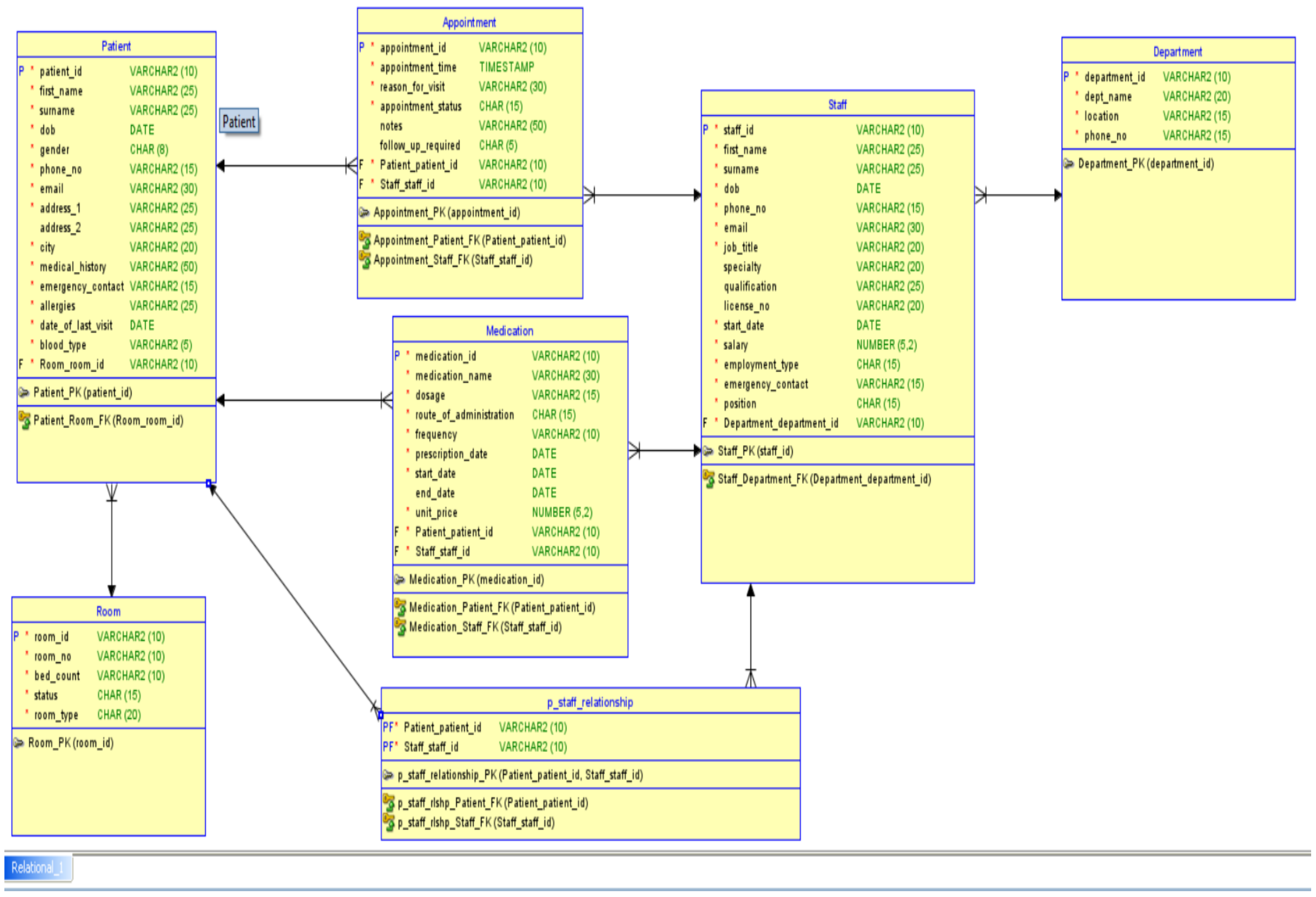

Patient
P * patient_id VARCHAR2 (10)
* first_name VARCHAR2 (25)
* surname VARCHAR2 (25)
* dob DATE
* gender CHAR (8)
* phone_no VARCHAR2 (15)
* email VARCHAR2 (30)
* address_1 VARCHAR2 (25)
address_2 VARCHAR2 (25)
* city VARCHAR2 (20)
* medical_history VARCHAR2 (50)
* emergency_contact VARCHAR2 (15)
* allergies VARCHAR2 (25)
* date_of_last_visit DATE
* blood_type VARCHAR2 (5)
F * Room_room_id VARCHAR2 (10)
Patient_PK (patient_id)
Patient_Room_FK (Room_room_id)
Patient
Appointment
P * appointment_id VARCHAR2 (10)
* appointment_time TIMESTAMP
* reason_for_visit VARCHAR2 (30)
* appointment_status CHAR (15)
notes VARCHAR2 (50)
follow_up_required CHAR (5)
F * Patient_patient_id VARCHAR2 (10)
F * Staff_staff_id VARCHAR2 (10)
Appointment_PK (appointment_id)
Appointment_Patient_FK (Patient_patient_id)
Appointment_Staff_FK (Staff_staff_id)
Staff
P * staff_id VARCHAR2 (10)
* first_name VARCHAR2 (25)
* surname VARCHAR2 (25)
* dob DATE
* phone_no VARCHAR2 (15)
* email VARCHAR2 (30)
* job_title VARCHAR2 (20)
specialty VARCHAR2 (20)
qualification VARCHAR2 (25)
license_no VARCHAR2 (20)
* start_date DATE
* salary NUMBER (6,2)
* employment_type CHAR (15)
* emergency_contact VARCHAR2 (15)
* position CHAR (15)
F * Department_department_id VARCHAR2 (10)
Staff_PK (staff_id)
Staff_Department_FK (Department_department_id)
Department
P * department_id VARCHAR2 (10)
* dept_name VARCHAR2 (20)
* location VARCHAR2 (15)
* phone_no VARCHAR2 (15)
Department_PK (department_id)
Medication
P * medication_id VARCHAR2 (10)
* medication_name VARCHAR2 (30)
* dosage VARCHAR2 (15)
* route_of_administration CHAR (15)
* frequency VARCHAR2 (10)
* prescription_date DATE
* start_date DATE
end_date DATE
* unit_price NUMBER (6,2)
F * Patient_patient_id VARCHAR2 (10)
F * Staff_staff_id VARCHAR2 (10)
Medication_PK (medication_id)
Medication_Patient_FK (Patient_patient_id)
Medication_Staff_FK (Staff_staff_id)
Room
P * room_id VARCHAR2 (10)
* room_no VARCHAR2 (10)
* bed_count VARCHAR2 (10)
* status CHAR (15)
* room_type CHAR (20)
Room_PK (room_id)
p_staff_relationship
PF* Patient_patient_id VARCHAR2 (10)
PF* Staff_staff_id VARCHAR2 (10)
p_staff_relationship_PK (Patient_patient_id, Staff_staff_id)
p_staff_rlshp_Patient_FK (Patient_patient_id)
p_staff_rlshp_Staff_FK (Staff_staff_id)
Relational_1

Fig 2: Relational Model

Primary Key: this uniquely identifies each record in a table. Examples include appointment_id, staff_id, patient_id, etc. It must be a non-null value.

Foreign Key: is a column in one table that refers to the primary key of another table in the database. It is used to establish a relationship between the tables. Examples include staff_staff_id, room_room_id, etc.

Patient – Staff- this is a many to many relationships as shown in Fig 1 above. Patients and staff members have a many-to-many relationship where each can be assigned to multiple instances of the other.

Patient – Appointment- a patient can have many appointments, and an appointment belongs to only one patient at any time.

Patient – Room- each patient can occupy one room at a time, a one-to-one relationship between patient and room entities. A room can host one or more patient at any point in time, one-to-many relationship between room and patient. **It is important to note that in our context, room is referred to as admission room**. This is to ensure that we do not have a contradictory relationship- in a real-world setting, one patient can occupy multiple rooms per visit. Hence, the need for this assumption.

Patient – Medication- one patient can be assigned one or medication at any point in time, a one-to-many relationship between patient and medication. A medication can only be assigned to one patient at a time, one-to-one relationship between medication and patient.

Staff – Department- each staff belongs to one department in the hospital, a one-to-one relationship between staff and department. A department can have one or more staff members, one-to-many relationship between department and staff.

Staff – Appointment- a staff can be assigned to one or more appointments at any given time, but one appointment can only be assigned to a staff at any given time.

Staff – Medication- a staff can prescribe one or more medication at any given time, one-to-many relationship between staff and medication. Each medication can only be prescribed by one staff at a time, one-to-one relationship between medication and staff.

1.2 Data Types

Data types of the various columns or attributes in our model defines the values that each column can hold or the nature of data that can be stored in the column. We have used Varchar2, Date, Timestamp (used in appointment_time to keep track of date and time simultaneously), Char, Number (which stores numeric data with a maximum of 5 digits including 2 decimal places). This can be clearly seen in Fig 2. Benefit of choosing the right data types are listed below.

- Makes our database consistent, accurate. That is, this enforces data integrity.
- Storage optimization - different data types require varying amounts of storage space. Selecting appropriate data types minimizes space usage
- It enables more efficient query processing through data type-based optimizations.
- It facilitates manipulating data more effectively. This allows us to perform various mathematical operations such as arithmetic calculations; sums, average, etc, on numeric data types.
- It serves as a data validation measure: the system ensures only the required data type can be stored within each column.
- Hence, data types are quite important for maintaining a well-structured, reliable and efficient database.

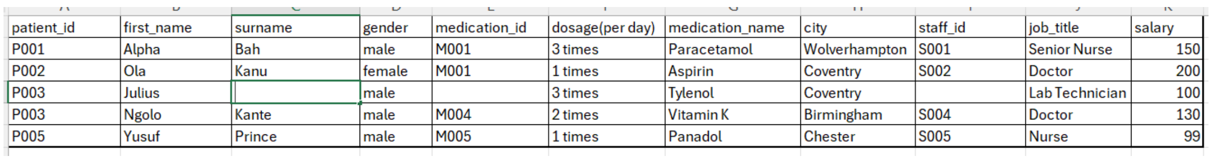
Chapter 2

2.1 Normalisation

Normalisation is an important database design process which removes data redundancy, improves data integrity, removes data anomaly and ensuring an efficient database. This involves dividing large tables into a smaller and related one. Removing or minimizing data duplication and ensuring data dependencies make sense is the fundamental objective.

For our database, we will limit our normalisation process to 1NF, 2NF, and 3NF. Let us now consider these 3 processes separately.

Let us consider the table below.

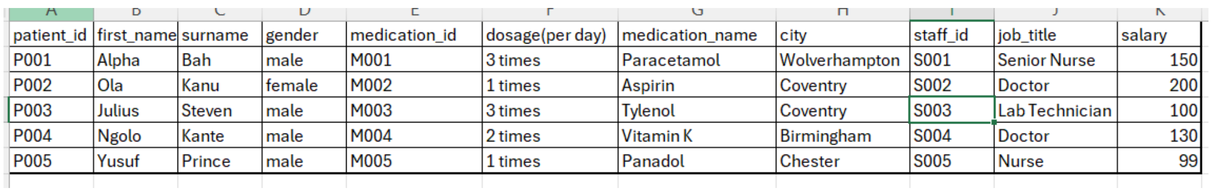


patient_id	first_name	surname	gender	medication_id	dosage(per day)	medication_name	city	staff_id	job_title	salary
P001	Alpha	Bah	male	M001	3 times	Paracetamol	Wolverhampton	S001	Senior Nurse	150
P002	Ola	Kanu	female	M001	1 times	Aspirin	Coventry	S002	Doctor	200
P003	Julius		male		3 times	Tylenol	Coventry		Lab Technician	100
P003	Ngolo	Kante	male	M004	2 times	Vitamin K	Birmingham	S004	Doctor	130
P005	Yusuf	Prince	male	M005	1 times	Panadol	Chester	S005	Nurse	99

Table 1: Unnormalized table

In a table 1, we have a table with missing values, duplicates. To ensure our table is normalised, we will start by normalising at the 1NF as shown below.

1NF- First Normal Form

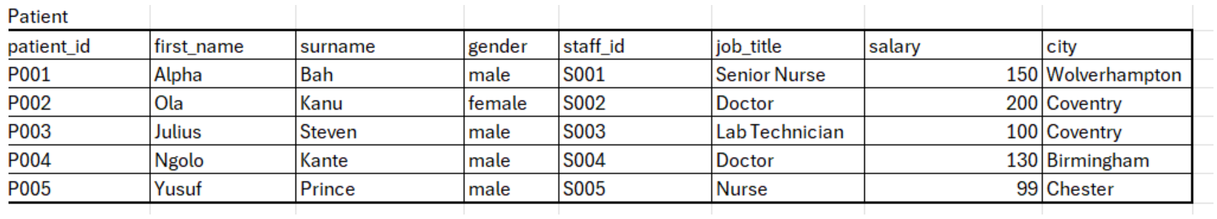


patient_id	first_name	surname	gender	medication_id	dosage(per day)	medication_name	city	staff_id	job_title	salary
P001	Alpha	Bah	male	M001	3 times	Paracetamol	Wolverhampton	S001	Senior Nurse	150
P002	Ola	Kanu	female	M002	1 times	Aspirin	Coventry	S002	Doctor	200
P003	Julius	Steven	male	M003	3 times	Tylenol	Coventry	S003	Lab Technician	100
P004	Ngolo	Kante	male	M004	2 times	Vitamin K	Birmingham	S004	Doctor	130
P005	Yusuf	Prince	male	M005	1 times	Panadol	Chester	S005	Nurse	99

Table 2: First Normal Form

Based on other information in table 1, repeating groups and duplicates have been removed from the table.

2NF- Second Normal Form



Patient

patient_id	first_name	surname	gender	staff_id	job_title	salary	city
P001	Alpha	Bah	male	S001	Senior Nurse	150	Wolverhampton
P002	Ola	Kanu	female	S002	Doctor	200	Coventry
P003	Julius	Steven	male	S003	Lab Technician	100	Coventry
P004	Ngolo	Kante	male	S004	Doctor	130	Birmingham
P005	Yusuf	Prince	male	S005	Nurse	99	Chester

Table 3: Second Normal Form

Medication

medication_id	dosage(per day)	medication_name
M001	3 times	Paracetamol
M002	1 times	Aspirin
M003	3 times	Tylenol
M004	2 times	Vitamin K
M005	1 times	Panadol

Table 4: Second Normal Form

We have removed all partial dependencies in table 2 and form two more tables. In table two, we have established a foreign key, staff_id. Hence, we can see that by breaking our table further, we have established a second normal form type of normalisation.

3NF- Third Normal Form

In table 3, we have transitive dependency in the patient table. To ensure we normalise to the 3NF, we can create a new table, staff and modify the table patient. For the 3 NF, we have established three tables, 4, 5, and 6.

Patient

patient_id	first_name	surname	gender	city	staff_id
P001	Alpha	Bah	male	Wolverhampton	S001
P002	Ola	Kanu	female	Coventry	S002
P003	Julius	Steven	male	Coventry	S003
P004	Ngolo	Kante	male	Birmingham	S004
P005	Yusuf	Prince	male	Chester	S005

Table 5: Third Normal Form

Staff

staff_id	job_title	salary	medication_id
S001	Senior Nur	150	M001
S002	Doctor	200	M002
S003	Lab Techni	100	M003
S004	Doctor	130	M004
S005	Nurse	99	M005

Table 6: Third Normal Form

Summary

We have seen how we can use normalisation to remove redundancies and anomalies in our database. The use of foreign keys to access information in other tables and ensuring primary keys are efficiently utilised. To ensure the efficiency of our database, normalisation is a crucial step in the database design process.

Chapter 3

3.1 Implementation and Query Execution

This chapter uses SQL statements to create, populate, select and join tables.

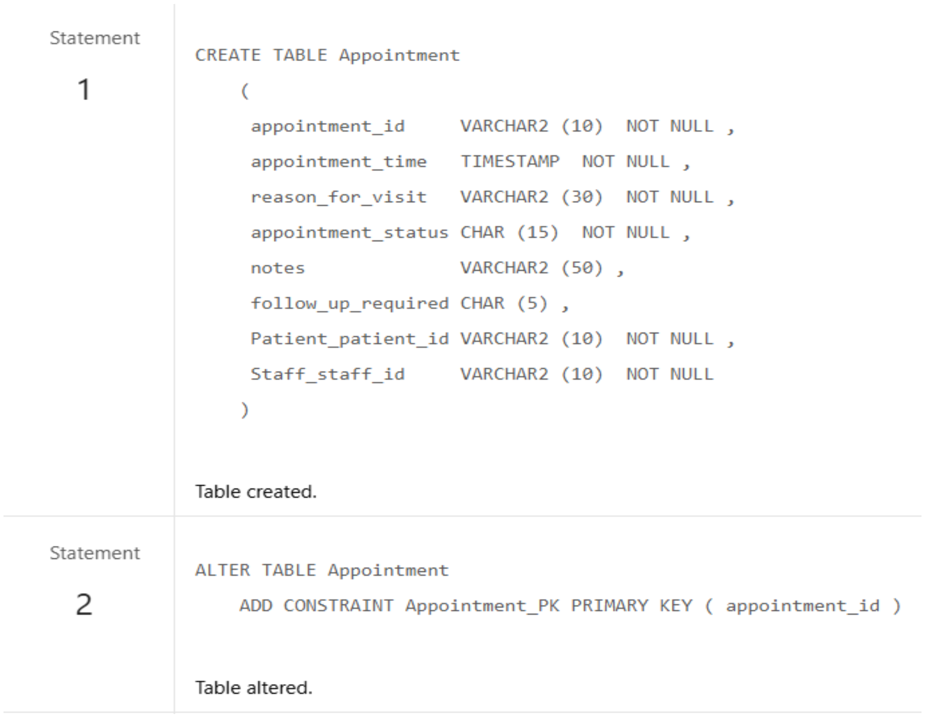
3.2 Creating and Populating Tables

For each entity below, we have created the tables by listing each attribute in the table, specifying the data types, primary key- which is added by using the ALTER statement, and the foreign keys.

CREATE Statement- this is used to define the structure of a new table, including column names, data types and constraints- primary and foreign key.

ALTER Statement- this is a powerful tool used for making changes to a table, view, indexes, and other database elements after they have been created. In this project, we have used the ALTER statement to add the primary keys to the various entities in the database.

Fig A to Fig P illustrates the CREATE, ALTER (to add primary keys), INSERT INTO (to populate), and SELECT statements used in this project.

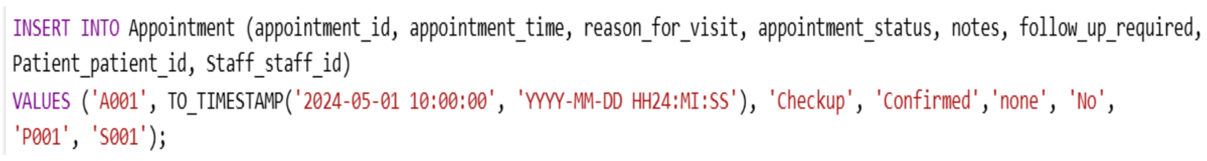

Statement 1

```
CREATE TABLE Appointment
    (
     appointment_id     VARCHAR2 (10)  NOT NULL ,
     appointment_time   TIMESTAMP  NOT NULL ,
     reason_for_visit   VARCHAR2 (30)  NOT NULL ,
     appointment_status CHAR (15)  NOT NULL ,
     notes              VARCHAR2 (50) ,
     follow_up_required CHAR (5) ,
     Patient_patient_id VARCHAR2 (10)  NOT NULL ,
     Staff_staff_id     VARCHAR2 (10)  NOT NULL
    )

Table created.
```

Statement 2

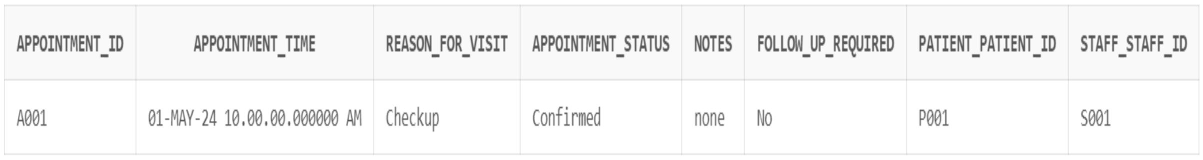
```
ALTER TABLE Appointment
    ADD CONSTRAINT Appointment_PK PRIMARY KEY ( appointment_id )

Table altered.
```

Fig A: creating the Appointment table.



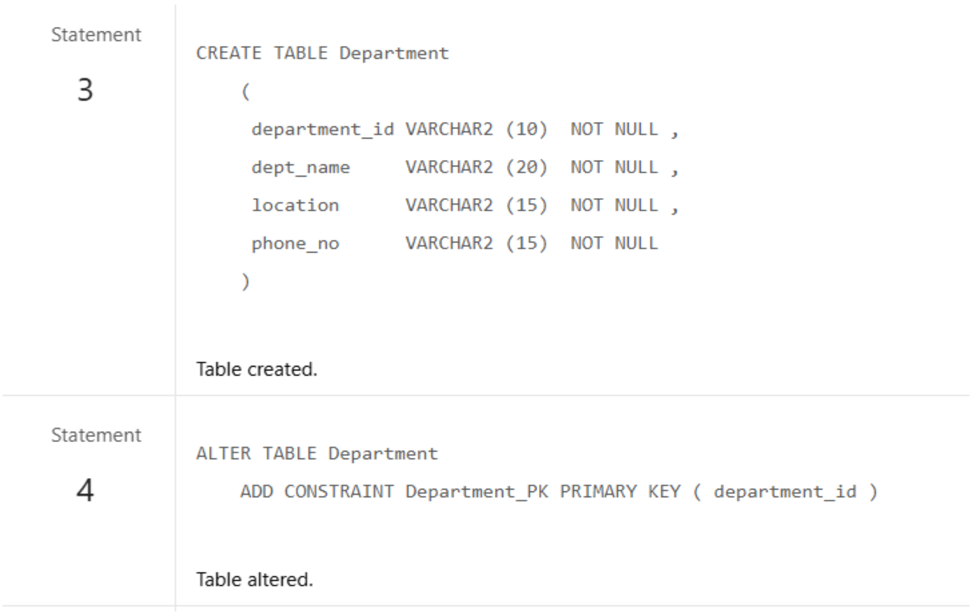
```
INSERT INTO Appointment (appointment_id, appointment_time, reason_for_visit, appointment_status, notes, follow_up_required,
Patient_patient_id, Staff_staff_id)
VALUES ('A001', TO_TIMESTAMP('2024-05-01 10:00:00', 'YYYY-MM-DD HH24:MI:SS'), 'Checkup', 'Confirmed','none', 'No',
'P001', 'S001');
```

Fig B: populating the appointment table



APPOINTMENT_ID	APPOINTMENT_TIME	REASON_FOR_VISIT	APPOINTMENT_STATUS	NOTES	FOLLOW_UP_REQUIRED	PATIENT_PATIENT_ID	STAFF_STAFF_ID
A001	01-MAY-24 10.00.00.000000 AM	Checkup	Confirmed	none	No	P001	S001

Fig C: appointment table output



Statement
3

```
CREATE TABLE Department
    (
     department_id VARCHAR2 (10)  NOT NULL ,
     dept_name     VARCHAR2 (20)  NOT NULL ,
     location      VARCHAR2 (15)  NOT NULL ,
     phone_no      VARCHAR2 (15)  NOT NULL
    )
```

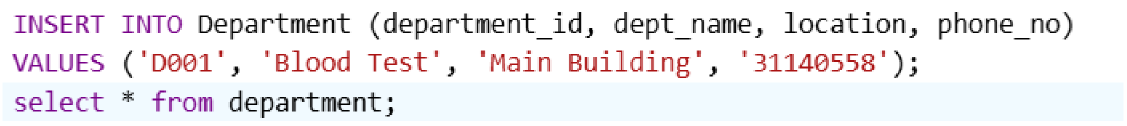
Table created.

Statement
4

```
ALTER TABLE Department
    ADD CONSTRAINT Department_PK PRIMARY KEY ( department_id )
```

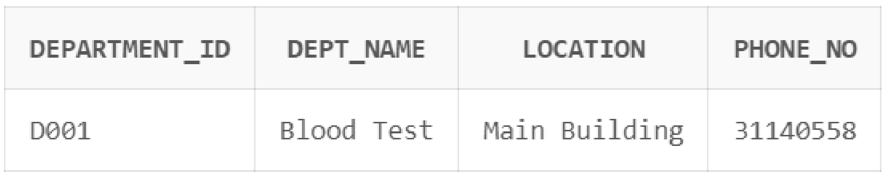
Table altered.

Fig D: creating the Department table



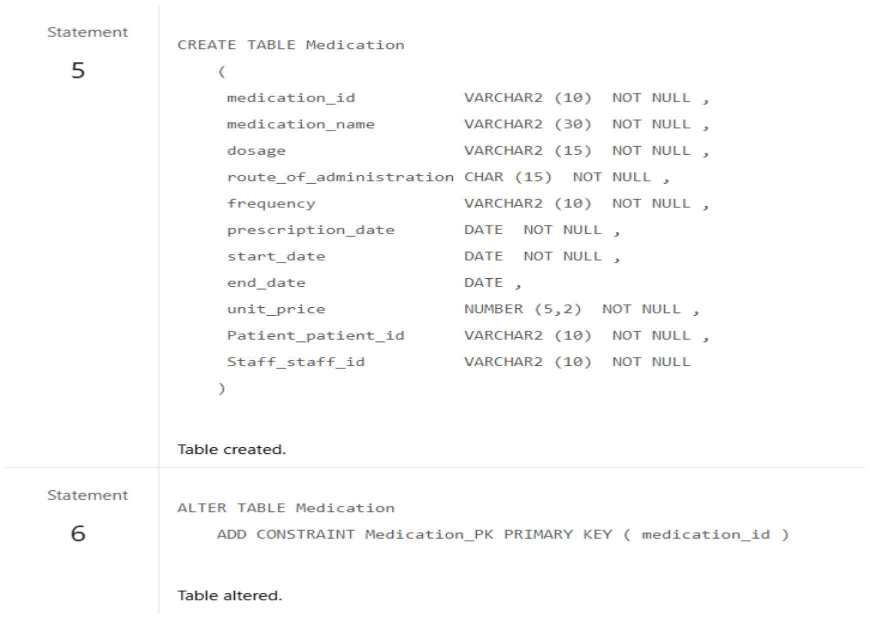
```
INSERT INTO Department (department_id, dept_name, location, phone_no)
VALUES ('D001', 'Blood Test', 'Main Building', '31140558');
select * from department;
```

Fig E: populating the department table



DEPARTMENT_ID	DEPT_NAME	LOCATION	PHONE_NO
D001	Blood Test	Main Building	31140558

Fig F: department table output



Statement 5

```
CREATE TABLE Medication
    (
     medication_id              VARCHAR2 (10)  NOT NULL ,
     medication_name            VARCHAR2 (30)  NOT NULL ,
     dosage                     VARCHAR2 (15)  NOT NULL ,
     route_of_administration CHAR (15)  NOT NULL ,
     frequency                  VARCHAR2 (10)  NOT NULL ,
     prescription_date          DATE  NOT NULL ,
     start_date                 DATE  NOT NULL ,
     end_date                   DATE ,
     unit_price                 NUMBER (5,2)  NOT NULL ,
     Patient_patient_id         VARCHAR2 (10)  NOT NULL ,
     Staff_staff_id             VARCHAR2 (10)  NOT NULL
    )
```

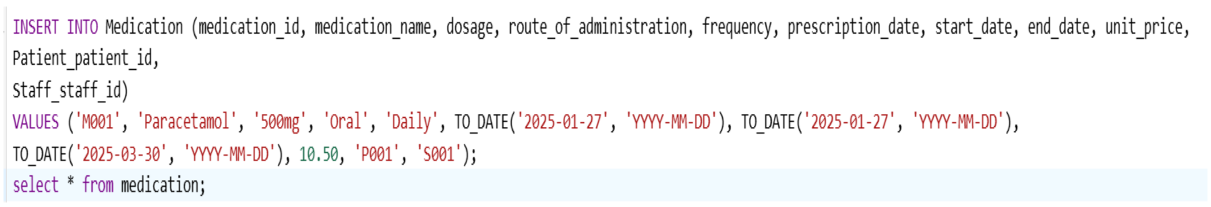
Table created.

Statement 6

```
ALTER TABLE Medication
    ADD CONSTRAINT Medication_PK PRIMARY KEY ( medication_id )
```

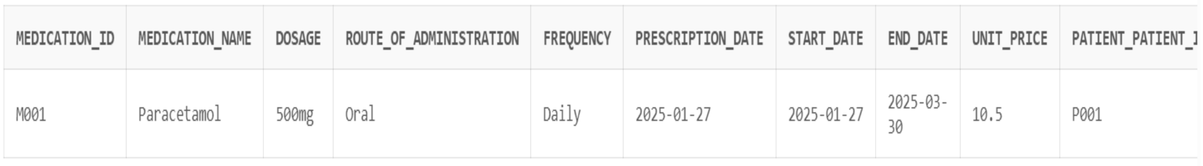
Table altered.

Fig G: creating the medication table



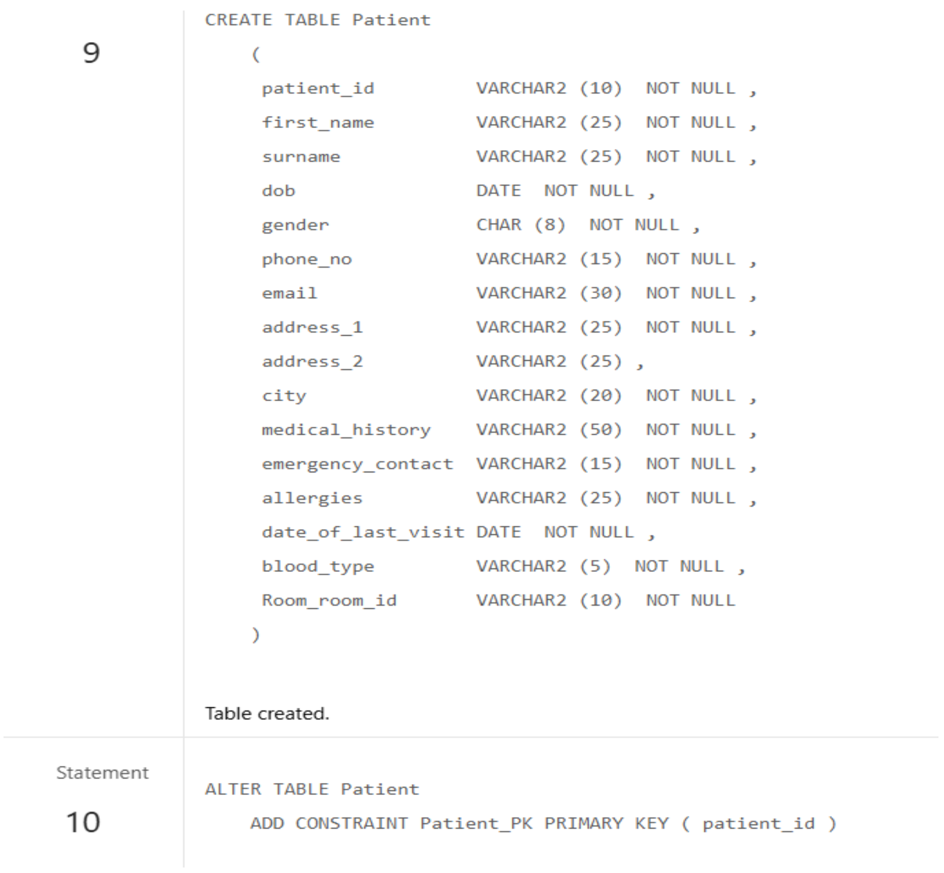
```
INSERT INTO Medication (medication_id, medication_name, dosage, route_of_administration, frequency, prescription_date, start_date, end_date, unit_price,
Patient_patient_id,
Staff_staff_id)
VALUES ('M001', 'Paracetamol', '500mg', 'Oral', 'Daily', TO_DATE('2025-01-27', 'YYYY-MM-DD'), TO_DATE('2025-01-27', 'YYYY-MM-DD'),
TO_DATE('2025-03-30', 'YYYY-MM-DD'), 10.50, 'P001', 'S001');
select * from medication;
```

Fig: populating the medication table



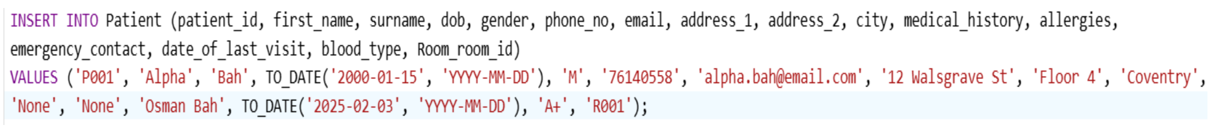
MEDICATION_ID	MEDICATION_NAME	DOSAGE	ROUTE_OF_ADMINISTRATION	FREQUENCY	PRESCRIPTION_DATE	START_DATE	END_DATE	UNIT_PRICE	PATIENT_PATIENT_I
M001	Paracetamol	500mg	Oral	Daily	2025-01-27	2025-01-27	2025-03-30	10.5	P001

Fig H: medication table output



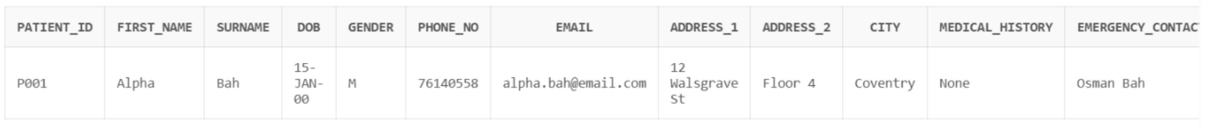
```
9       CREATE TABLE Patient
          (
           patient_id          VARCHAR2 (10)  NOT NULL ,
           first_name          VARCHAR2 (25)  NOT NULL ,
           surname             VARCHAR2 (25)  NOT NULL ,
           dob                 DATE   NOT NULL ,
           gender              CHAR (8)  NOT NULL ,
           phone_no            VARCHAR2 (15)  NOT NULL ,
           email               VARCHAR2 (30)  NOT NULL ,
           address_1           VARCHAR2 (25)  NOT NULL ,
           address_2           VARCHAR2 (25) ,
           city                VARCHAR2 (20)  NOT NULL ,
           medical_history     VARCHAR2 (50)  NOT NULL ,
           emergency_contact   VARCHAR2 (15)  NOT NULL ,
           allergies           VARCHAR2 (25)  NOT NULL ,
           date_of_last_visit  DATE   NOT NULL ,
           blood_type          VARCHAR2 (5)  NOT NULL ,
           Room_room_id        VARCHAR2 (10)  NOT NULL
          )

        Table created.

Statement
10      ALTER TABLE Patient
            ADD CONSTRAINT Patient_PK PRIMARY KEY ( patient_id )
```

Fig I: creating the patient table



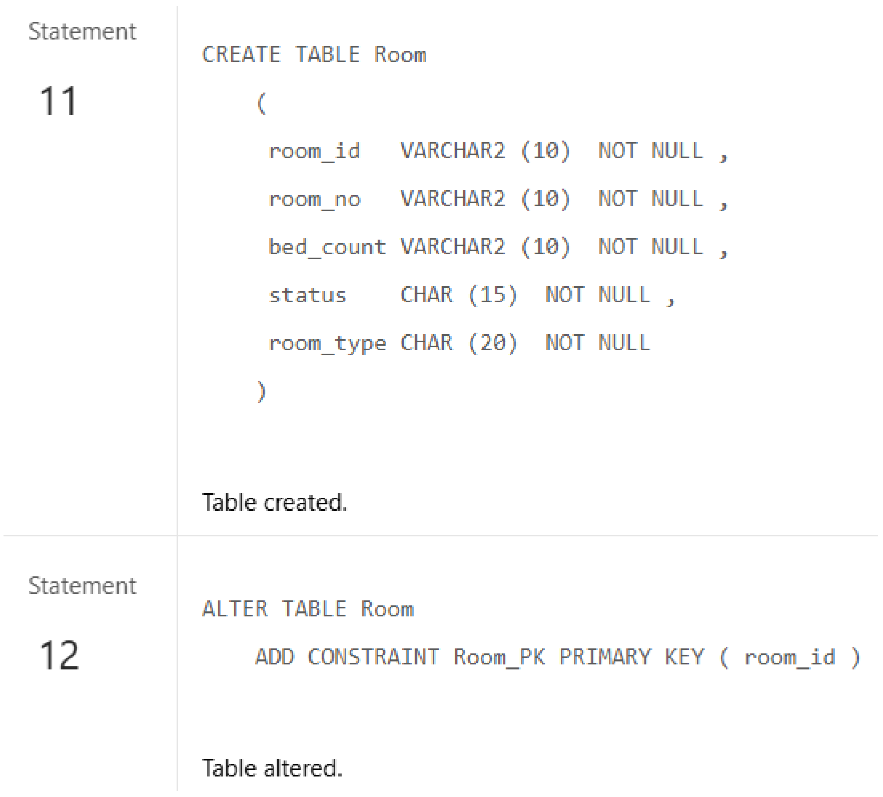
```
INSERT INTO Patient (patient_id, first_name, surname, dob, gender, phone_no, email, address_1, address_2, city, medical_history, allergies,
emergency_contact, date_of_last_visit, blood_type, Room_room_id)
VALUES ('P001', 'Alpha', 'Bah', TO_DATE('2000-01-15', 'YYYY-MM-DD'), 'M', '76140558', 'alpha.bah@email.com', '12 Walsgrave St', 'Floor 4', 'Coventry',
'None', 'None', 'Osman Bah', TO_DATE('2025-02-03', 'YYYY-MM-DD'), 'A+', 'R001');
```

Fig J: populating the patient table



PATIENT_ID	FIRST_NAME	SURNAME	DOB	GENDER	PHONE_NO	EMAIL	ADDRESS_1	ADDRESS_2	CITY	MEDICAL_HISTORY	EMERGENCY_CONTAC
P001	Alpha	Bah	15-JAN-00	M	76140558	alpha.bah@email.com	12 Walsgrave St	Floor 4	Coventry	None	Osman Bah

Fig K: patient table output



Statement 11

```
CREATE TABLE Room
    (
     room_id   VARCHAR2 (10)  NOT NULL ,
     room_no   VARCHAR2 (10)  NOT NULL ,
     bed_count VARCHAR2 (10)  NOT NULL ,
     status    CHAR (15)  NOT NULL ,
     room_type CHAR (20)  NOT NULL
    )
```

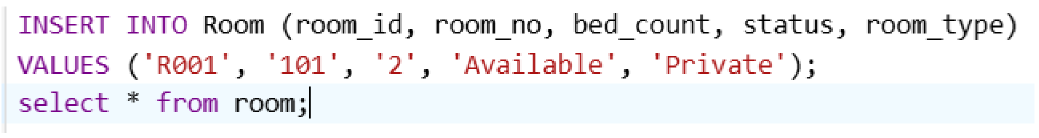
Table created.

Statement 12

```
ALTER TABLE Room
    ADD CONSTRAINT Room_PK PRIMARY KEY ( room_id )
```

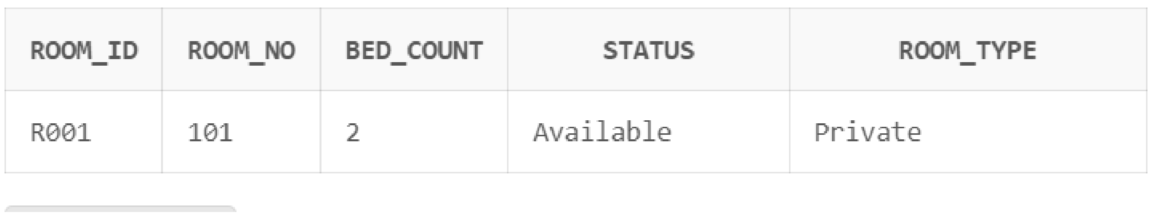
Table altered.

Fig L: creating the room table



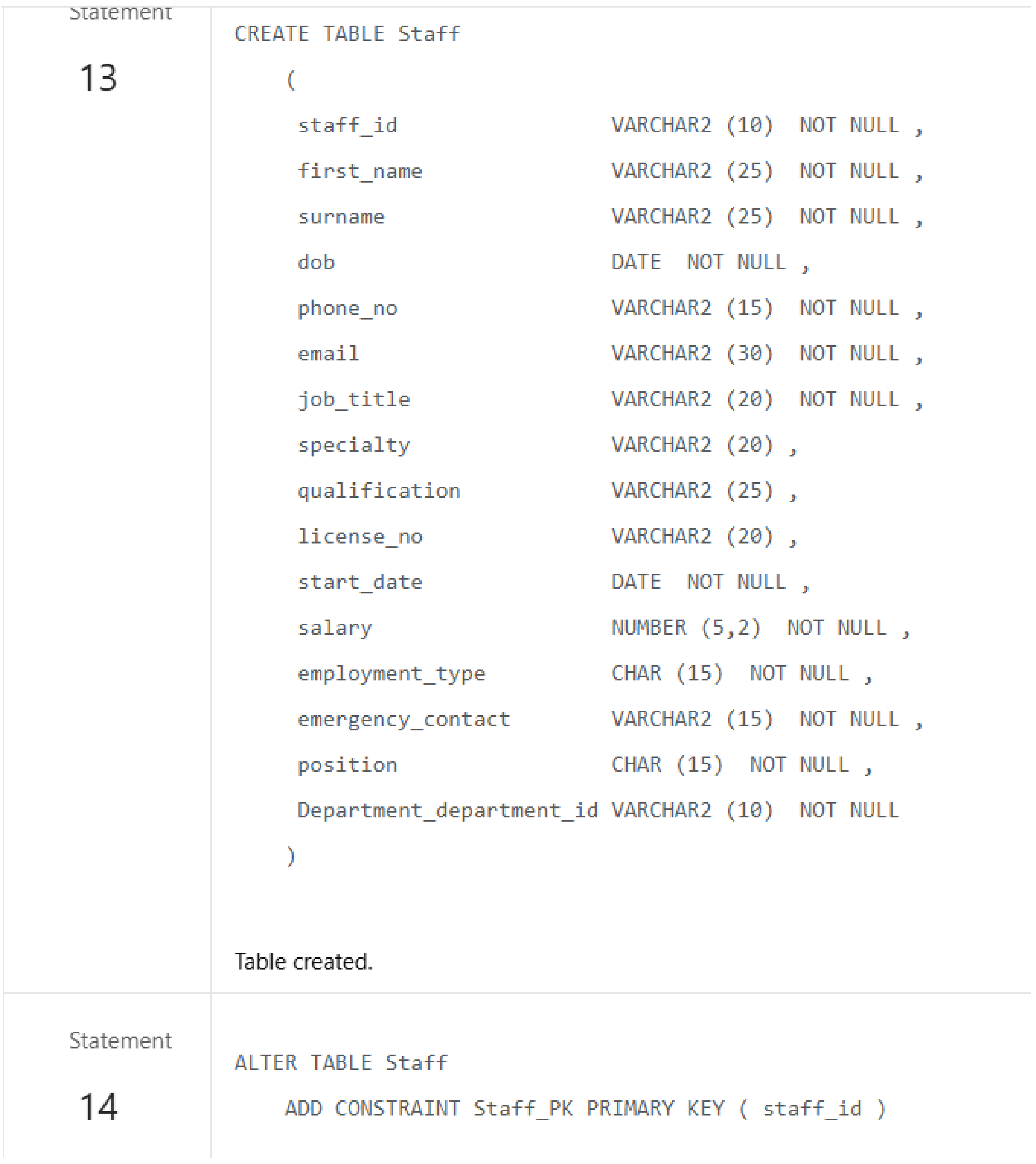
```
INSERT INTO Room (room_id, room_no, bed_count, status, room_type)
VALUES ('R001', '101', '2', 'Available', 'Private');
select * from room;
```

Fig: populating the room table



ROOM_ID	ROOM_NO	BED_COUNT	STATUS	ROOM_TYPE
R001	101	2	Available	Private

Fig M: room table output


Statement

13

```
CREATE TABLE Staff
    (
     staff_id                  VARCHAR2 (10)  NOT NULL ,
     first_name                VARCHAR2 (25)  NOT NULL ,
     surname                   VARCHAR2 (25)  NOT NULL ,
     dob                       DATE  NOT NULL ,
     phone_no                  VARCHAR2 (15)  NOT NULL ,
     email                     VARCHAR2 (30)  NOT NULL ,
     job_title                 VARCHAR2 (20)  NOT NULL ,
     specialty                 VARCHAR2 (20) ,
     qualification             VARCHAR2 (25) ,
     license_no                VARCHAR2 (20) ,
     start_date                DATE  NOT NULL ,
     salary                    NUMBER (5,2)  NOT NULL ,
     employment_type           CHAR (15)  NOT NULL ,
     emergency_contact         VARCHAR2 (15)  NOT NULL ,
     position                  CHAR (15)  NOT NULL ,
     Department_department_id VARCHAR2 (10)  NOT NULL
    )
```

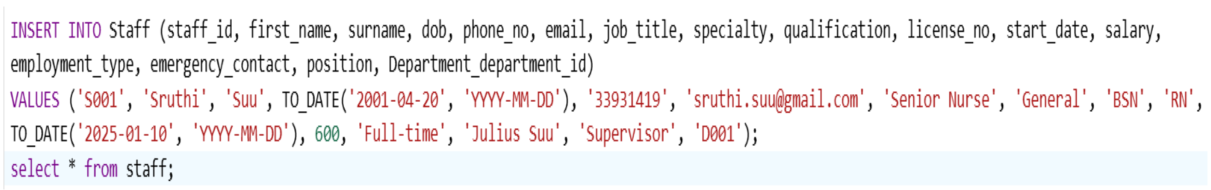
Table created.

Statement

14

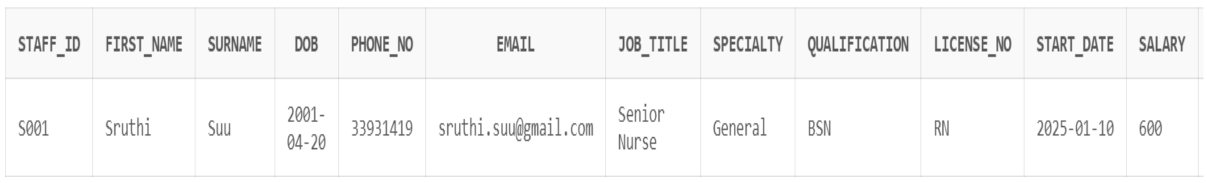
```
ALTER TABLE Staff
    ADD CONSTRAINT Staff_PK PRIMARY KEY ( staff_id )
```

Fig N: creating the staff table



```
INSERT INTO Staff (staff_id, first_name, surname, dob, phone_no, email, job_title, specialty, qualification, license_no, start_date, salary,
employment_type, emergency_contact, position, Department_department_id)
VALUES ('S001', 'Sruthi', 'Suu', TO_DATE('2001-04-20', 'YYYY-MM-DD'), '33931419', 'sruthi.suu@gmail.com', 'Senior Nurse', 'General', 'BSN', 'RN',
TO_DATE('2025-01-10', 'YYYY-MM-DD'), 600, 'Full-time', 'Julius Suu', 'Supervisor', 'D001');
select * from staff;
```

Fig O: Populating the staff table



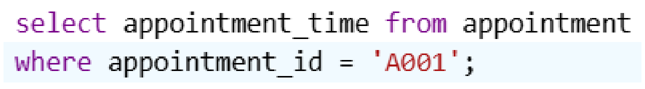
STAFF_ID	FIRST_NAME	SURNAME	DOB	PHONE_NO	EMAIL	JOB_TITLE	SPECIALTY	QUALIFICATION	LICENSE_NO	START_DATE	SALARY
S001	Sruthi	Suu	2001-04-20	33931419	sruthi.suu@gmail.com	Senior Nurse	General	BSN	RN	2025-01-10	600

Fig P: staff table output

3.3 Devising SQL Queries to Retrieve Data from our Database

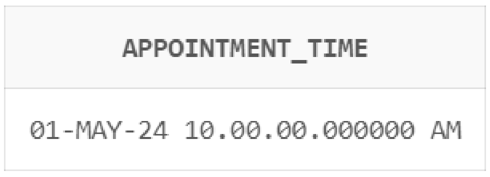
In this section, we have used four SQL queries on Oracle Live SQL to retrieve essential data from our database.

- **SELECT and WHERE Statement**- The SELECT query allows us to select a specific column or table from the database. The WHERE statement filters the rows returned by the SELECT clause based on the specified condition(s).



```sql
select appointment_time from appointment
where appointment_id = 'A001';
```

Fig I

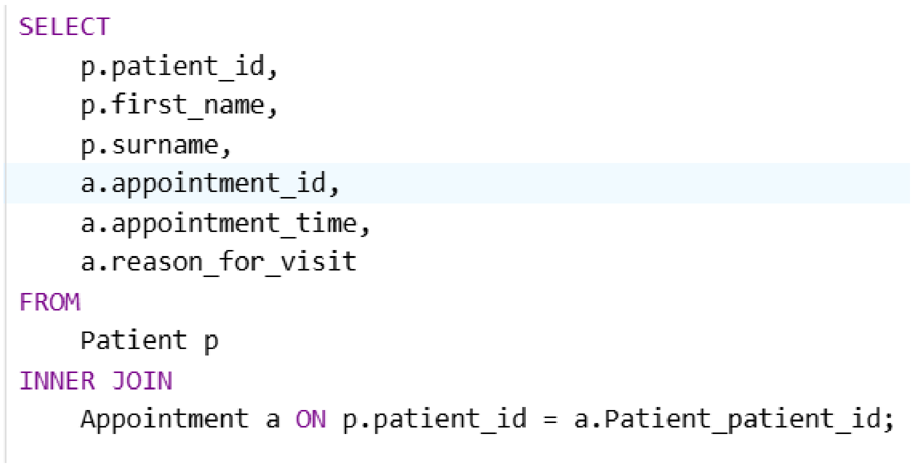


APPOINTMENT_TIME
01-MAY-24 10.00.00.000000 AM

Fig II

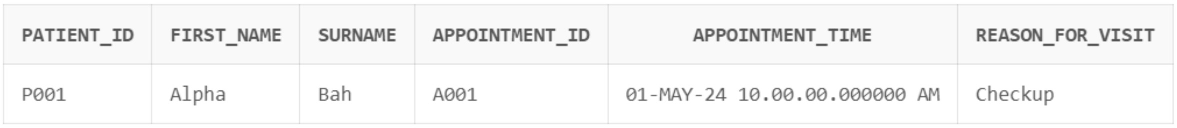
Fig I illustrates the SELECT and WHERE statements are used to retrieve data from the database. Fig II the output.

- **INNER JOIN**- The INNER JOIN operation combines rows from two or more tables based on a related column between them.



```sql
SELECT
    p.patient_id,
    p.first_name,
    p.surname,
    a.appointment_id,
    a.appointment_time,
    a.reason_for_visit
FROM
    Patient p
INNER JOIN
    Appointment a ON p.patient_id = a.Patient_patient_id;
```

Fig III



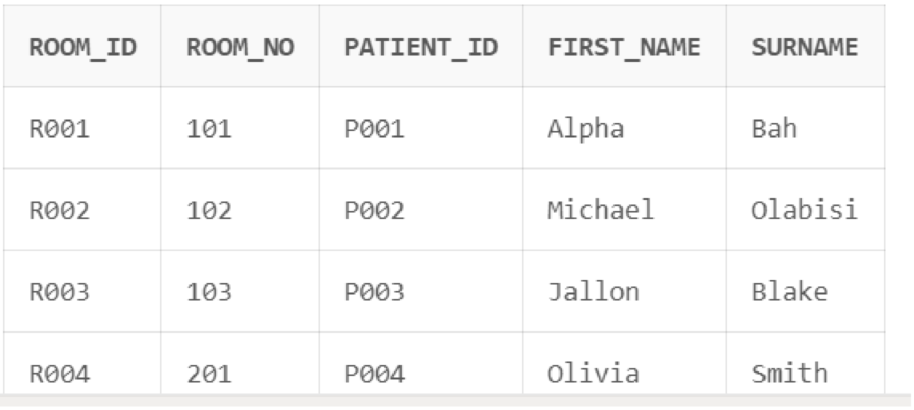
PATIENT_ID	FIRST_NAME	SURNAME	APPOINTMENT_ID	APPOINTMENT_TIME	REASON_FOR_VISIT
P001	Alpha	Bah	A001	01-MAY-24 10.00.00.000000 AM	Checkup

Fig IV

- **RIGHT JOIN**: this returns all rows in the right table and uniform columns in the left table. If there are no matching rows in the left table, the right table is returned but with null values in the left table's columns.

```
SELECT
    r.room_id,
    r.room_no,
    p.patient_id,
    p.first_name,
    p.surname
FROM
    Room r
RIGHT JOIN
    Patient p ON r.room_id = p.Room_room_id;
```

Fig V



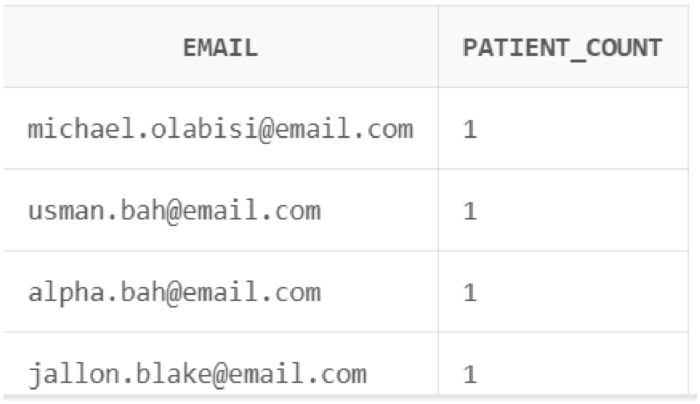
ROOM_ID	ROOM_NO	PATIENT_ID	FIRST_NAME	SURNAME
R001	101	P001	Alpha	Bah
R002	102	P002	Michael	Olabisi
R003	103	P003	Jallon	Blake
R004	201	P004	Olivia	Smith

Fig VI

- **SELECT, COUNT and GROUP BY Statements**: The COUNT statement is used to determine the number of rows in a table. The GROUP BY statement is used to group rows in a table that share the same features in the column. Fig VII and Fig VIII illustrate the query and output. The GROUP BY statement enables aggregate functions to be applied to the output.

```
SELECT
    email,
    COUNT(*) AS patient_count
FROM
    Patient
GROUP BY
    email;
```

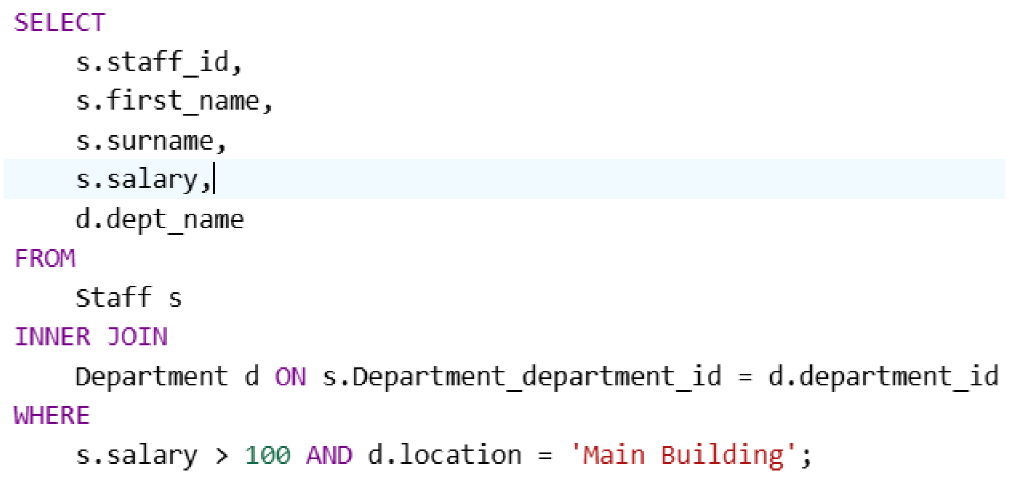
Fig VII



EMAIL	PATIENT_COUNT
michael.olabisi@email.com	1
usman.bah@email.com	1
alpha.bah@email.com	1
jallon.blake@email.com	1

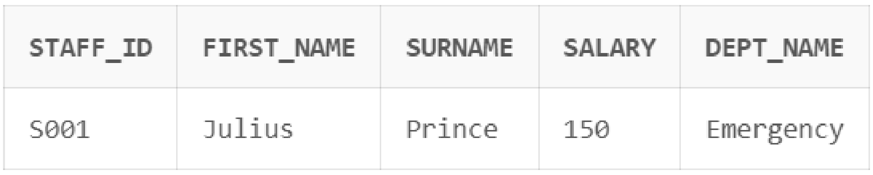
Fig VIII

- **INNER JOIN, WHERE, and AND Statements**: we have seen the use of the INNER JOIN earlier. The WHERE statement is used as a filter and the AND statement is used with WHERE statement to filter with multiple conditions that must be true to be included in the output. This is illustrated in Fig IX and Fig X.



```
SELECT
    s.staff_id,
    s.first_name,
    s.surname,
    s.salary,
    d.dept_name
FROM
    Staff s
INNER JOIN
    Department d ON s.Department_department_id = d.department_id
WHERE
    s.salary > 100 AND d.location = 'Main Building';
```

Fig IX



STAFF_ID	FIRST_NAME	SURNAME	SALARY	DEPT_NAME
S001	Julius	Prince	150	Emergency

Fig X

Chapter 4

4.0 Security, Integrity and Ethical Aspects of a Database

Data security, integrity and ethical handling are crucial for any database system, especially those storing personal information as unauthorized access could lead to detrimental impact on both the individual and company.

4.1 Security Considerations

- Implementation of strong encryption protocols,this will ensure the data is prevented from being viewed by unauthorised parties.
- Access Controls: ensuring a role-based access control (RBAC), staff can only have access to the information they need for the job.
- Authentication must be enforced for all users to keep track of those accessing the database and restricting unauthorised access. This must be complemented by keeping an auditing log to keep track of all those accessing the system.
- Data masking and redaction when processing data to protect sensitive data. Especially when sharing to third parties.
- Regular backup and a comprehensive disaster recovery plan must be in place in the case of an emergency. Cloud systems can be used for cost-effective backup. Alternatively, store backups on separate protected (fireproof and waterproof) hard drives. Regular simulation drills and vulnerability tests are conducted to assess staff preparedness and system resilience, addressing any identified shortcoming.

4.2 Integrity Considerations

Data integrity must be maintained across all departments.

- Implementing validation rules, constraints, and data entry checks to ensure data accuracy.
- Use referential integrity constraints (e.g. foreign keys) to establish consistent relationships between data tables.
- Normalisation of the database can eliminate redundancies and duplicates.
- Transaction integrity must also be maintained across the system. Incomplete or incorrect transactions can lead to data inconsistencies. The implementation of Atomicity, Consistency, Isolation, Durability (ACID) properties to prevent incomplete or incorrect transactions.
- Concurrency control- proper locking mechanisms and isolation levels to manage concurrent user access and prevent data conflicts.

4.3 Ethical Considerations

- Ensuring personal data is collected, stored and processed in accordance with privacy regulations (e.g. GDPR).
- Staff must ensure only required data is collected for the intended purposes, avoiding the storage of sensitive information.
- Transparency must be observed across all levels. Users must be informed about data collection practices. Explicit consent must be obtained before collecting personal data.
- Data Accuracy and User Control- Providing mechanisms for users to validate, update, and delete their data.

- Ensuring data collection processes are inclusive and representative could eliminate data biases. The data collected must be audited regularly for biases.
- Establishing clear guidelines on how data can be shared with third parties, ensuring security and user notification.
- Implementation of data retention policies to ensure data are deleted when no longer needed.

Regular risk assessments and security evaluations are crucial. Staff must be trained on data privacy laws to ensure compliance and prevent breaches.

4.3 Security, Integrity and Ethical Governance in Relation to the Hospital Database

To ensure the security, integrity, and ethical standards, the hospital management must enforce the action plans mentioned above. This includes encrypting patient data, regulating access by instituting RBAC, using authentication and audit logs, and utilising data masking. Regular backups and disaster recovery plans complimented with vulnerability testing are crucial to prepare staff and address weaknesses.

Data integrity must be maintained through validation rules, ACID properties, and concurrency control. Ethical practices include compliance of legal instrument, like the GDPR, emphasizing consent, transparency, user control, bias detection, controlled data sharing, and data retention policies. Management must ensure that staff receive thorough training on data privacy laws to ensure compliance and minimize legal risks. By ensuring these action plans, the company will be in a sound footing to prevent illegal access.

Chapter 5
Conclusion

The project has successfully addressed Mr Osman’s need for an effective and efficient database system to manage hospital’s record. We have designed a fully normalised database design, represented by an ER model, that comprises of six key entities: Patient, Staff, Appointment, Department, Medication, and Room. The implementation of this database, as shown through SQL queries will streamline hospital operations, reduce administrative workload and ultimately save time and money. Moreover, the project incorporates essential security measures, integrity constraints, and ethical considerations to protect sensitive data and ensure compliance with relevant regulations.

In conclusion, this project provides Mr Osman with a robust and well-designed database solution, poised to significantly improve the management and efficiency of his hospital.

References

Odede, J., et al., [2025], Normalisation, [Presentation Slides]
Odede, J., et al., [2025], Database Governance and Ethics [Presentation Slides]
Google. (2024). Gemini (Language Model). [Personal Communication]
Accessed: March 2025.