Trinity College Dublin
Coláiste na Tríonóide, Baile Átha Cliath
The University of Dublin

CS4D2A
Information Management II Project Report

Aditya Vishnu Garg
17317530

INDEX

DECRIPTION

For Information Management project, I have chosen to represent the various accommodation hostels of Trinity College. This model has various entities like Hostel Details, Student Details, Staff Details, Room Details, Complaint Details and Feedback Details.

For this database, there are numerous hostels, and for each hostel there are several students and rooms, and subsequently each hostel also employs several staff. For each room there can be multiple complaints and feedback on those complaints lodged.

The six-relation table that I have decided to model are listed as follows:

1. <u>Hostel Details</u>:

- This table has all the essentials details for each individual Trinity accommodation.

- It has the following attributes:

  Hostel ID : Primary Key
  Phone Number – Unique phone number of receptions for each hostel
  Capacity – Total number of rooms for each hostel
  Name – Unique name for each Trinity accommodation.
  Address. – Street Name, City and State

- Each hostel will have multiple students staying, employing multiple staff members.

2. <u>Student Details</u>

- This table contains the personal information for each student staying in the hostel accommodation.

- It has the following attributes:

  ID number: Primary Key
  Name, Major, Year, Semester, Date of birth, Mobile, Email, Father's name, Mother's name, Father's mobile number, Mother's mobile number, Address

- Each student has only 1 room and can have multiple complaints.

3. <u>Staff Details</u>

- This table contains the personal details for each staff member employed in a hostel.

- It has the following attributes:

  Staff ID number: Primary Key
  Name, Role, Hostel ID, Mobile Number, Email, Address

- Staff members will manage multiple complaints and will allot the rooms to the students.

4. <u>Room Details</u>

- This table contains the details for each room in a particular hostel.
- It has the following attributes:

  Hostel ID, Room Number, Block Number: Primary Key
  Occupancy

- Each room will have only one student allotted to it

5. <u>Complaint Details</u>

- This table contains the complaints for each room in a particular hostel.
- It has the following attributes:

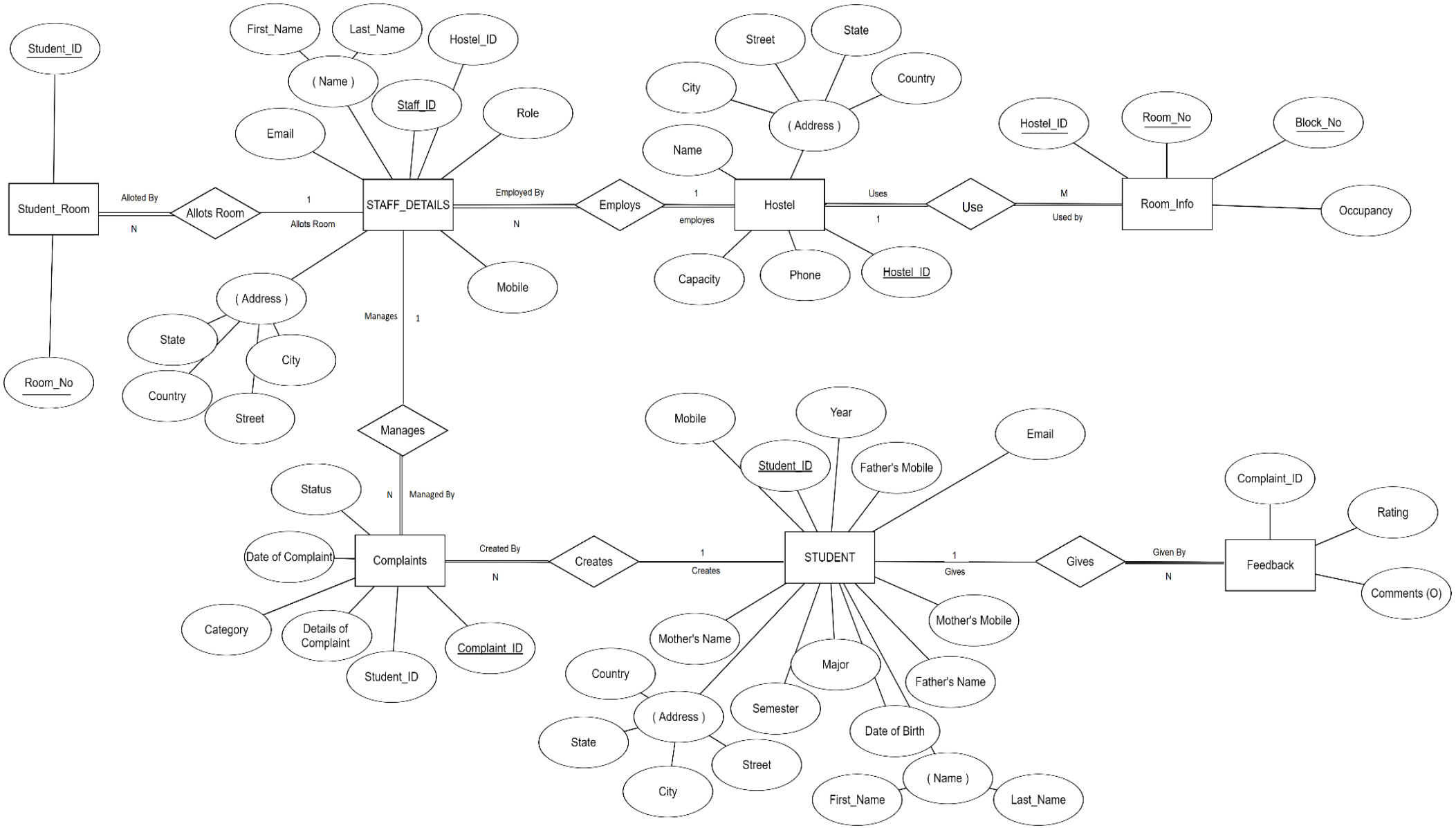
Complaint ID: Primary Key
Category of Complaint, Student ID, Details of Complaint, Status of Complaint, Date of complaint

6. <u>Feedback Details</u>

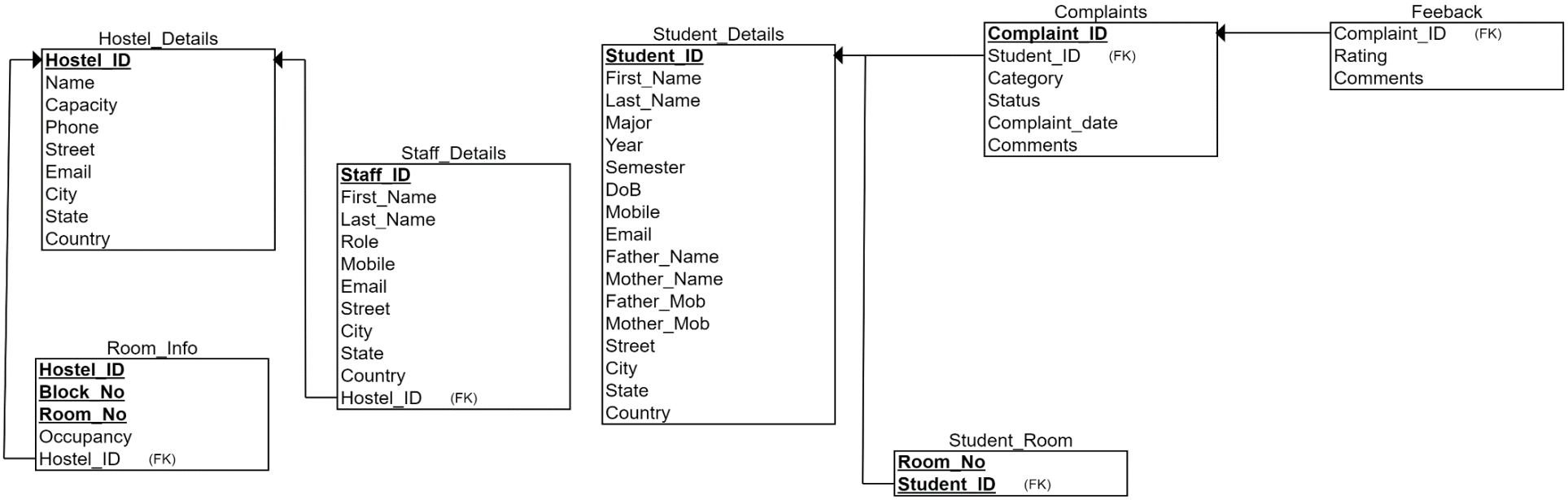
- This table contains the feedback for each complaint resolved.
- It has the following attributes:

Complaint ID, Rating, Comments

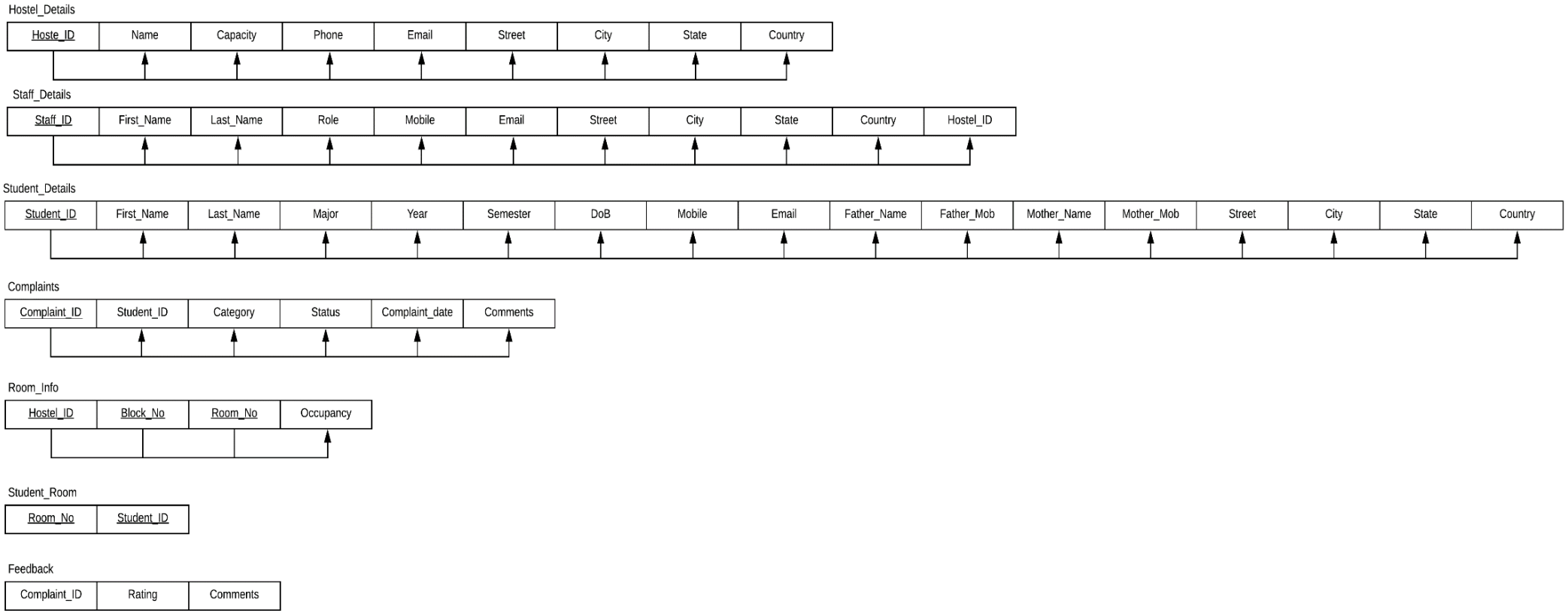
ENTITY RELATIONSHIP DIAGRAM



RELATION SCHEMA DIAGRAM



FUNCTIONAL DEPENDENCY DIAGRAM



SECURITY CONSIDERATIONS

Security in a database is an important part when designing the database, the reason being that in an unsecure database, the data can be compromised and might even lead illegal privacy violations.

For the hostel management database, we plan to create different users having limited access which enables to fulfil only their particular set of duties. This ensures that privacy of users' data is not compromised, and integrity of data is maintained by not giving extra privileges to users who don't it.

The three roles that have been implemented in this database are as follows:

1. Warden
2. Receptionist
3. Maintenance Staff

For creating the above roles, the following commands were implemented:

1. Warden:

```
CREATE ROLE Warden;;
GRANT CREATE, DELETE ON student_details TO Warden;
GRANT UPDATE, SELECT ON student_details TO Warden WITH GRANT OPTION;
GRANT CREATE, DELETE, UPDATE, READ ON Student_Room TO Warden WITH
GRANT OPTION;
```

2. Receptionist:

```
CREATE ROLE Receptionist;
GRANT CREATE, SELECT, UPDATE ON Complaints TO Receptionist;
GRANT CREATE, DELETE, UPDATE, READ ON Student_Room TO Receptionist;
GRANT UPDATE, READ ON Room_Info TO Receptionist;
```

3. Maintenance Staff:

```
CREATE ROLE MaintenanceStaff;
GRANT CREATE, SELECT, UPDATE ON Complaints TO MaintenanceStaff;
```

For assigning the above roles to the staff members we can use the following commands:

```
GRANT 'Warden' TO MollySatchz;
GRANT 'Receptionist' TO JimmySatchz
GRANT 'MaintenanceStaff' TO HannahBaker
```

For removing the privileges from staff members, the following command can be used:
:

```
REVOKE DELETE ON Room_Info FROM Receptionist;
```

Further, each user can be allotted a unique password key which automatically expires in a specific set of days:

CREATE USER 'HannahBaker' IDENTIFIED BY 'new_password' PASSWORD EXPIRE INTERVAL 180 DAYS;

SEMANTIC CONSTRAINTS

1. Integrity constraints:
    - All primary and foreign keys were declared as NOT NULL.
2. Semantic Constraints:
    - Check Constraint:
        - A check constraint is to limit the values that can be inserted in data variable.
            - For instance, CHECK (Capacity > 0) would ensure that no hostel has capacity equal to or less than 0.
        - A CHECK IN constraint is used to limit values from a set of pre-defined values.
            - For instance, CHECK `(Rating IN (1 , 2, 3, 4, 5)) would ensure that Rating can only take a value which lies in the set {1,2, 3, 4, 5}`

VIEWS

The main function of Views is collection information from various different table which are used frequently. It is useful because it saves time without having to input the same query multiple times.

For this database, I have created a view wherein I am showing complaints which were resolved and the feedback that was obtained.

```
CREATE VIEW view_resolved_complaints AS
    SELECT
        complaints.complaint_ID,
        complaints.category AS 'Category of Complaint',
        complaints.complaint_date AS 'Complaint Date',
        complaints.student_ID AS 'Student ID',
        student_room.room_no AS 'Room Number',
        feedback.rating AS 'Feedback Rating',
        feedback.comments AS 'Feedback Comments'
    FROM
        complaints,
        student_room,
        feedback
    WHERE
        complaints.complaint_ID = feedback.complaint_ID
            AND complaints.student_ID = student_room.student_ID;
```

For the above view code we obtain the following output table:

Complaint_ID	Category of Complaint	Complaint Date	Student ID	Room Number	Feedback Rating	Feedback Comments
3	Electricity	2018-11-28	11317530	301	4	It is working perfect now
4	Kitchen	2018-11-28	10313030	201	3	Water leakage is fixed for now
5	Wi-Fi	2018-11-28	10017530	601	5	Wi-Fi speed is stable

TRIGGERS

A trigger is set of SQL queries which are automatically executes when there is some operation performed on a table. The three operations on which triggers can be executed are DELETE, INSERT, and UPDATE.

All the triggers definitions along with SQL commands have been implemented in Appendix 3.

ADVANCED & ADDITIONAL FEATUES

1. Emergency Contact Listing

Emergency Contact Listing is crucial for every organization, yet it just so happens that it is overlooked and most of the time not given enough importance. It might happen that a particular student living in a hostel is in medical emergency or might go missing.

To address this issue, I have designed using a special curated procedure in SQL, which uses cursor to parse through the entire student detail table.

It basically returns just names and contact information for a student's parents.

The SQL query for this part is attached in Appendix 5.1.

2. Auto Scheduler Back Up Data

It might happen that particular data entry has been accidentally deleted by the user or it might happen that the user needs access to some deleted data due to variety of reasons. This might include the office reception trying to contact student for payment of fine or some lost belonging which was found after the student left the hostel accommodation.

Hence, it would become easy for hostel administration to access this backup data.

To address this issue, I have created temporary table and a trigger which stores the data values deleted from student_details into delete_backup table.

And to make that those data values don't stay in the database for forever, I have created an event which checks every for data entries which are older than 30 days and deletes them automatically, thus ensuring free space.

The SQL query for this part is attached in Appendix 5.2

QUERYING THE DATABASE(EXAMPLES)

1. Select * from hostel_details

Hostel_ID	Name	Email	Phone	Capacity	Country	City	State	Street
1	Trinity Halls	thal@tcd.ie	432678948	100	Ireland	Dublin	Leinster	Rathmines
2	GoldSmith Halls	gsmit@tcd.ie	634578948	400	Ireland	Dublin	Leinster	Westland Row
3	Kavangh Court	kcourt@tcd.ie	634578948	600	Ireland	Dublin	Leinster	Gardiner Street
4	Binary Hub	bhub@tcd.ie	432673948	1200	Ireland	Dublin	Leinster	Bonham Street
5	Broad Stone	bstone@tcd.ie	122678948	500	Ireland	Dublin	Leinster	Phibsborough

2. Joins Example

```
1.  SELECT
2.      hostel_details.hostel_id,
3.      hostel_details.Name,
4.      staff_details.staff_id,
5.      staff_details.first_name,
6.      staff_details.last_name,
7.      staff_details.role
8.  FROM
9.      hostel_details,
10.     staff_details
11. WHERE
12.     hostel_details.hostel_ID = staff_details.hostel_ID
13.         AND hostel_details.hostel_ID = 1;
```

Hostel_ID	Name	Staff_ID	First_Name	Last_Name	Role
1	Trinity Halls	1	Molly	Satchz	Warden
1	Trinity Halls	2	Brian	Foro	Receptionist
1	Trinity Halls	3	Jimmy	Satchz	Receptionist
1	Trinity Halls	4	Hannah	Baker	Maintenance
1	Trinity Halls	5	James	Boruh	Maintenance

APPENDIX 1

Data definition commands used to create the tables in database.

1. Table for hostel details

```
CREATE TABLE hostel_details (
    Hostel_ID INTEGER UNIQUE NOT NULL,
    Name VARCHAR(100) NOT NULL,
    Email VARCHAR(100) NOT NULL,
    Phone INTEGER NOT NULL,
    Capacity INTEGER NOT NULL,
    Country VARCHAR(100) NOT NULL,
    State VARCHAR(100) NOT NULL,
    City VARCHAR(100) NOT NULL,
    Street VARCHAR(100) NOT NULL,
    PRIMARY KEY (Hostel_ID),
    CHECK (Capacity > 0)
);
```

2. Table for student details

```
CREATE TABLE student_details (
    Student_ID INTEGER UNIQUE NOT NULL,
    First_Name VARCHAR(100) NOT NULL,
    Last_Name VARCHAR(100) NOT NULL,
    Major VARCHAR(100),
    Year VARCHAR(100) NOT NULL,
    Semester INTEGER NOT NULL,
    DoB DATE,
    Email VARCHAR(100) NOT NULL,
    Mobile INTEGER NOT NULL,
    Father_Name VARCHAR(100) NOT NULL,
    Mother_Name VARCHAR(100) NOT NULL,
    Father_Mob INTEGER NOT NULL,
    Mother_Mob INTEGER NOT NULL,
    Country VARCHAR(100) NOT NULL,
    State VARCHAR(100) NOT NULL,
    City VARCHAR(100) NOT NULL,
    Street VARCHAR(100) NOT NULL,
    PRIMARY KEY (Student_ID),
    CHECK (Semester > 0)
);
```

3. Table for staff details

```
CREATE TABLE staff_details (
    Staff_ID INTEGER UNIQUE NOT NULL,
    First_Name VARCHAR(100) NOT NULL,
    Last_Name VARCHAR(100) NOT NULL,
    Role VARCHAR(100) NOT NULL,
    Hostel_ID INTEGER NOT NULL,
    Email VARCHAR(100) NOT NULL,
    Mobile INTEGER NOT NULL,
    Country VARCHAR(100) NOT NULL,
    State VARCHAR(100) NOT NULL,
    City VARCHAR(100) NOT NULL,
```

```
    Street VARCHAR(100) NOT NULL,
    FOREIGN KEY (Hostel_ID)
        REFERENCES hostel_details (Hostel_ID),
    CHECK (Role IN ('Warden' , 'Receptionist', 'Maintenance'))
);
```

4. Table for room details

```
CREATE TABLE Room_Info (
    Hostel_ID INTEGER NOT NULL,
    Room_No INTEGER NOT NULL,
    Block_No INTEGER NOT NULL,
    Occupancy VARCHAR(30) NOT NULL,
    PRIMARY KEY (Hostel_ID , Room_No , Block_No),
    FOREIGN KEY (Hostel_ID)
        REFERENCES hostel_details (Hostel_ID),
    CHECK (Occupancy IN ('Occupied' , 'Vacant'))
);
```

5. Table for room allotted to student

```
CREATE TABLE Student_Room (
    Room_No INTEGER,
    Student_ID INTEGER,
    PRIMARY KEY (Room_No , Student_ID),
    FOREIGN KEY (Student_ID)
        REFERENCES Student_details (Student_ID)
);
```

6. Table for complaints filed by student

```
CREATE TABLE Complaints (
    Complaint_ID INTEGER PRIMARY KEY,
    Category VARCHAR(100),
    Student_ID INTEGER,
    Comments VARCHAR(100),
    Complaint_date DATE,
    Status VARCHAR(50),
    FOREIGN KEY (Student_ID)
        REFERENCES Student_details (Student_ID)
);
```

7. Table for recording feedback taken from students.

```
CREATE TABLE Feedback (
    Complaint_ID INTEGER NOT NULL,
    Rating INTEGER NOT NULL,
    Comments VARCHAR(100),
    FOREIGN KEY (Complaint_ID)
        REFERENCES Complaints (Complaint_ID),
    CHECK (Rating IN (1 , 2, 3, 4, 5))
);
```

APPENDIX 2

Data definition commands used to insert data in tables.

1. Data Insertion for hostels

```
1. INSERT INTO hostel_details VALUES (0001, 'Trinity Halls', 'thall@tcd.ie','432678948
   ',1000,'Ireland', 'Dublin', 'Dublin', 'Rathmines');
2. INSERT INTO hostel_details VALUES (0002, 'GoldSmith
   Halls', 'gsmit@tcd.ie','532648948',400,'Ireland', 'Dublin', 'Leinster', 'Westland
   Row');
3. INSERT INTO hostel_details VALUES (0003, 'Kavangh
   Court', 'kcourt@tcd.ie','634578948',600,'Ireland', 'Dublin', 'Leinster', 'Gardiner
   Street');
4. INSERT INTO hostel_details VALUES (0004, 'Binary
   Hub', 'bhub@tcd.ie','432673948',1200,'Ireland', 'Dublin', 'Leinster', 'Bonham
   Street');
5. INSERT INTO hostel_details VALUES (0005, 'Broad
   Stone', 'bstone@tcd.ie','122678948',500,'Ireland', 'Dublin', 'Leinster', 'Phibsboro
   ugh');
```

2. Data Insertion for students

```
1. INSERT INTO student_details VALUES (17317530, 'Aditya', 'Vishnu', 'Computers','Four
   th',1,'1996-10-
   29', 'adll@tcd.ie',122003894,'Sanjiv Vishnu','Seema Vishnu', 232333894, 982311894,'
   Ireland','Leinster', 'Dublin', 'Rathmines');
2. INSERT INTO student_details VALUES (15314530, 'Billy', 'Smith', 'Computers','Fourth
   ',2,'1995-01-22', 'smith@tcd.ie',100103894,'John Smith','Shiny
   Smith', 972003894, 112322894,'United Kingdom','Oxfordshire', 'Oxford', 'Port
   Street');
3. INSERT INTO student_details VALUES (11317530, 'Susy', 'Jean', 'Physics','First',2,'
   1998-10-21', 'susy@tcd.ie',199003894,'Sam Jean','Alex
   Jean', 231003894, 002311894,'Ireland','Leinster', 'Dublin', 'Rathmines');
4. INSERT INTO student_details VALUES (10313030, 'Alice', 'Walsh', 'History','Fourth',
   1,'1999-10-29', 'alice@tcd.ie',122003894,'Henry Walsh','Sussy
   Walsh', 567333894, 102311894,'Ireland','Leinster', 'Naas', 'Henry Street');
5. INSERT INTO student_details VALUES (10017530, 'Jaimie', 'Clarke', 'Computers','Four
   th',1,'1998-10-
   29', 'jaime@tcd.ie',122003894,'Conor Clarke','Soso Clarke', 231033894, 102311894,'I
   reland','Munster', 'Cork', 'Glanmire');
```

3. Data Insertion for staff in Hostel : Trinity

```
1. INSERT INTO staff_details VALUES (001, 'Molly', 'Satchz', 'Warden',1,'molly@tcd.ie'
   ,122003894,'Ireland','Leinster', 'Dublin', 'Phisborough');
2. INSERT INTO staff_details VALUES (002, 'Brian', 'Foro', 'Receptionist',1,'brian@tcd
   .ie',700003894,'Ireland','Munster', 'Cork', 'Port Street');
3. INSERT INTO staff_details VALUES (003, 'Jimmy', 'Satchz', 'Receptionist',1,'jimmy@t
   cd.ie',922003894,'Ireland','Leinster', 'Dublin', 'Grand Canal');
4. insert INTO staff_details VALUES (004, 'Hannah', 'Baker', 'Maintenance',1,'hannah@t
   cd.ie',765003894,'Ireland','Leinster', 'Tallaght', 'OConor Street');
5. INSERT INTO staff_details VALUES (005, 'James', 'Boruh', 'Maintenance',1,'james@tcd
   .ie',532003894,'Ireland','Leinster', 'Dublin', 'Dundalk');
```

4. Data Insertion for staff in Hostel : Goldsmith

```
1. INSERT INTO staff_details VALUES (006, 'Brian', 'Satchz', 'Warden',2,'briany@tcd.ie
   ',122003893,'Ireland','Leinster', 'Dublin', 'Phisborough');
2. INSERT INTO staff_details VALUES (007, 'Donald', 'Blythe', 'Receptionist',2,'donald
   @tcd.ie',910003894,'Ireland','Munster', 'Cork', 'Port Street');
3. INSERT INTO staff_details VALUES (008, 'Henry', 'Satchz', 'Receptionist',2,'henry@t
   cd.ie',762003894,'Ireland','Leinster', 'Dublin', 'Grand Canal');
4. INSERT INTO staff_details VALUES (009, 'Mark', 'Baker', 'Maintenance',2,'mark@tcd.i
   e',125003894,'Ireland','Leinster', 'Tallaght', 'OConor Street');
5. INSERT INTO staff_details VALUES (010, 'Evan', 'Boruh', 'Maintenance',2,'evan@tcd.i
   e',122003894,'Ireland','Leinster', 'Dublin', 'Dundalk');
```

5. Data Insertion for staff in Hostel : Kavangh Court

```
1. INSERT INTO staff_details VALUES (011, 'Randy', 'Mars', 'Warden',3,'randy@tcd.ie',1
   220038924,'Ireland','Leinster', 'Dublin', 'Tars Street');
2. INSERT INTO staff_details VALUES (012, 'Raymond', 'Foroh', 'Receptionist',3,'raymon
   d@tcd.ie',900003894,'Ireland','Munster', 'Cork', 'Roux Street');
3. INSERT INTO staff_details VALUES (013, 'Michael', 'Satchz', 'Receptionist',3,'micha
   el@tcd.ie',922333894,'Ireland','Leinster', 'Dublin', 'Grand Canal Dock');
4. INSERT INTO staff_details VALUES (014, 'Tim', 'Bane', 'Maintenance',3,'tim@tcd.ie',
   715003894,'Ireland','Leinster', 'Tallaght', 'OConor Street');
5. INSERT INTO staff_details VALUES (015, 'Jam', 'Boruh', 'Maintenance',3,'jam@tcd.ie'
   ,532203894,'Ireland','Leinster', 'Dublin', 'Dundalk');
```

6. Data Insertion for staff in Hostel : Binary Hub

```
1. INSERT INTO staff_details VALUES (016, 'Tony', 'Wane', 'Warden',4,'tony@tcd.ie',132
   0038924,'Ireland','Leinster', 'Dublin', 'Teresh Street');
2. INSERT INTO staff_details VALUES (017, 'Ram', 'Jane', 'Receptionist',4,'ram@tcd.ie'
   ,900003894,'Ireland','Munster', 'Cork', 'Roux Street');
3. INSERT INTO staff_details VALUES (018, 'Eric', 'Mers', 'Receptionist',4,'eric@tcd.i
   e',192333894,'Ireland','Leinster', 'Dublin', 'Grand Canal Dock');
4. INSERT INTO staff_details VALUES (019, 'Tony', 'Boon', 'Maintenance',4,'tonyb@tcd.i
   e',545003894,'Ireland','Leinster', 'Tallaght', 'OConor Street');
5. INSERT INTO staff_details VALUES (020, 'Hina', 'Boer', 'Maintenance',4,'hina@tcd.ie
   ',232203894,'Ireland','Leinster', 'Dublin', 'Dundalk');
```

7. Data Insertion for staff in Hostel : Broadstone

```
1. INSERT INTO staff_details VALUES (021, 'Peter', 'Parker', 'Warden',5,'peter@tcd.ie'
   ,1440038924,'Ireland','Leinster', 'Dublin', 'Tars Street');
2. INSERT INTO staff_details VALUES (022, 'Sansa', 'Foroh', 'Receptionist',5,'sansa@tc
   d.ie',900003834,'Ireland','Munster', 'Cork', 'Winster Street');
3. INSERT INTO staff_details VALUES (023, 'John', 'Satchz', 'Receptionist',5,'john@tcd
   .ie',922333894,'Ireland','Leinster', 'Dublin', 'Grand Canal Dock');
4. INSERT INTO staff_details VALUES (024, 'Tintin', 'Bane', 'Maintenance',5,'tintin@tc
   d.ie',715203894,'Ireland','Leinster', 'Tallaght', 'OConor Street');
5. INSERT INTO staff_details VALUES (025, 'Mary', 'Boruh', 'Maintenance',5,'mary@tcd.i
   e',532203894,'Ireland','Leinster', 'Dublin', 'Dundalk');
```

8. Data Insertion for room_info

```
1.  INSERT INTO room_info VALUES (001, 101, 1, 'Vacant');
2.  INSERT INTO room_info VALUES (002, 102, 1, 'Occupied');
3.  INSERT INTO room_info VALUES (003, 201, 2, 'Occupied');
4.  INSERT INTO room_info VALUES (004, 301, 3, 'Occupied');
5.  INSERT INTO room_info VALUES (001, 401, 4, 'Occupied');
6.  INSERT INTO room_info VALUES (001, 601, 6, 'Occupied');
```

9. Data Insertion for room allotted to students

```
1.  INSERT INTO student_room VALUES (102, 17317530);
2.  INSERT INTO student_room VALUES (401, 15314530);
3.  INSERT INTO student_room VALUES (301, 11317530);
4.  INSERT INTO student_room VALUES (201, 10313030);
5.  INSERT INTO student_room VALUES (601, 10017530);
```

10. Data Insertion for complaints

```
1.  INSERT INTO complaints VALUES (001, 'Wi-Fi', 17317530, 'Wi-
    Fi not working properly', NULL, NULL);
2.  INSERT INTO complaints VALUES (002, 'Washroom', 15314530, 'Hot Water Leakage',
    NULL, NULL);
3.  INSERT INTO complaints VALUES (003, 'Electricity', 11317530, 'No electricity in
    room',NULL, 'Resolved');
4.  INSERT INTO complaints VALUES (004, 'Kitchen Stove', 10313030, 'Stove not
    working',NULL, 'Resolved');
5.  INSERT INTO complaints VALUES (005, 'Wi-Fi', 10017530, 'Low wifi speed',NULL,
    'Resolved');
```

8. Data Insertion for feedback

```
1.  INSERT INTO student_room VALUES (003, 4, 'It is working perfect now');
2.  INSERT INTO student_room VALUES (004, 3, 'Water leakage is fixed for now');
3.  INSERT INTO student_room VALUES (005, 5, 'Wi-Fi speed is stable');
```

APPENDIX 3

Data definition commands used to create TRIGGERS.

1. Trigger for setting the date when a complaint is raised by the student.

```
CREATE TRIGGER Set_Date
BEFORE INSERT
ON Complaints
FOR EACH ROW
SET New.Complaint_Date = SYSDATE();
```

2. Trigger for setting the status when a complaint is raised by the student

```
CREATE TRIGGER Set_Status
BEFORE INSERT
ON Complaints
FOR EACH ROW
SET New.Status = 'Unresolved';
```

3. Trigger for changing the occupancy of the room to Occupied if it is allotted to a student

```
CREATE TRIGGER Set_Occupany
    AFTER INSERT
    ON Student_Room
    FOR EACH ROW
    UPDATE Room_Info r
    SET r.Occupancy = 'Occupied'
    where r.Room_No = NEW.Room_No;
```

4. Trigger for changing the occupancy of the room to Vacant if the student leaves the room.

```
CREATE TRIGGER Set_Occupany_Vacant
    AFTER DELETE
    ON Student_Room
    FOR EACH ROW
    UPDATE Room_Info r
    SET r.Occupancy = 'Vacant'
    where r.Room_No = OLD.Room_No;
```

APPENDIX 4

Data definition commands used to create PROCEDURES.

1. Procedure which takes Complaint ID as the input and returns all the essential details about that complaint

```
1.  DELIMITER //
2.  CREATE PROCEDURE CheckComplaintID(in TEMPid INTEGER)
3.  BEGIN
4.      SELECT
5.          Complaints.Complaint_ID,
6.          Complaints.Category,
7.          Complaints.Comments,
8.          Complaints.Complaint_date
9.      FROM
10.         Complaints
11.     WHERE
12.         Complaints.Complaint_ID = TEMPid;
13. end//
```

2. Procedure which takes Complaint ID as the input and returns the number of days since that complaint was created.

```
1.  DELIMITER //
2.  CREATE PROCEDURE CheckComplaintDays(in TEMPid INTEGER)
3.  BEGIN
4.
5.      SELECT
6.          (curdate()-Complaints.complaint_date)
7.      FROM
8.          Complaints
9.      WHERE
10.         Complaints.Complaint_ID = TEMPid;
11. end//
```

APPENDIX 5.1

Data definition commands used to create ADVANCED FUNCTION Emergency Contact Listing using Cursor.

```
1.  DELIMITER //
2.  CREATE PROCEDURE emergency (IN r_no INTEGER)
3.  BEGIN
4.  DECLARE done INT DEFAULT FALSE;
5.
6.  DECLARE id_temp integer;
7.  DECLARE fname_temp varchar(100);
8.  DECLARE lname_temp varchar(100);
9.  DECLARE faname_temp varchar(100);
10. DECLARE mname_temp varchar(100);
11. DECLARE famob_temp integer;
12. DECLARE mmob_temp integer;
13.
14. DECLARE  c1 cursor FOR select Student_ID, First_Name, Last_Name, Father_Name, Mothe
    r_Name, Father_Mob, Mother_Mob from student_details where STUDENT_ID=r_no;
15. DECLARE CONTINUE HANDLER FOR NOT FOUND SET done = TRUE;
16.     open c1;
17.   read_loop: LOOP
18.         fetch c1 into id_temp,fname_temp,lname_temp,faname_temp,mname_temp,famob_te
    mp,mmob_temp;
19. SELECT
20.     id_temp AS 'Student ID',
21.     fname_temp AS 'First Name',
22.     lname_temp AS 'Last Name',
23.     faname_temp AS 'Fathers Name',
24.     mname_temp AS 'Mothers Name',
25.     famob_temp AS 'Father Mobile Number',
26.     mmob_temp AS 'Mother Mobile Number';
27.           IF done THEN
28.       LEAVE read_loop;
29.     END IF;
30.
31. END LOOP;
32.
33.   CLOSE c1;
34. end//
```

APPENDIX 5.2

Data definition commands used to create ADVANCED FUNCTION
Auto Scheduler Backup using Events and Triggers

```
1.  CREATE TABLE delete_backup (
2.      Student_ID INTEGER UNIQUE,
3.      First_Name VARCHAR(100),
4.      Last_Name VARCHAR(100),
5.      Major VARCHAR(100),
6.      Year VARCHAR(100),
7.      Semester INTEGER,
8.      DoB DATE,
9.      Email VARCHAR(100),
10.     Mobile INTEGER,
11.     Father_Name VARCHAR(100),
12.     Mother_Name VARCHAR(100),
13.     Father_Mob INTEGER,
14.     Mother_Mob INTEGER,
15.     Country VARCHAR(100),
16.     State VARCHAR(100),
17.     City VARCHAR(100),
18.     Street VARCHAR(100),
19.     Creation_Date date
20. );
```

```
21. DELIMITER //
22. CREATE TRIGGER BACKUP_DATA
23. BEFORE  DELETE ON STUDENT_DETAILS
24. FOR EACH ROW
25. BEGIN
26. INSERT INTO DELETE_Backup VALUES(OLD.Student_ID, OLD.FIRST_NAME, OLD.LAST_NAME, OLD
    .Major, OLD.Year, OLD.Semester, OLD.DoB, OLD.EMAIL, OLD.Mobile, OLD.Father_Name, OL
    D.Mother_Name, OLD.Father_Mob, OLD.Mother_Mob, OLD.Country, OLD.State, OLD.City, OL
    D.Street, sysdate());
27. END//
```

```
28. delimiter |
29. CREATE EVENT deletion
30. ON SCHEDULE
31.       EVERY 1 day
32.     COMMENT 'Clearing backup data'
33.     DO
34.       BEGIN
35.         DELETE FROM delete_backup WHERE (CUR_DATE()-D_DATE)>31;
36.       END |
37. delimiter ;
```