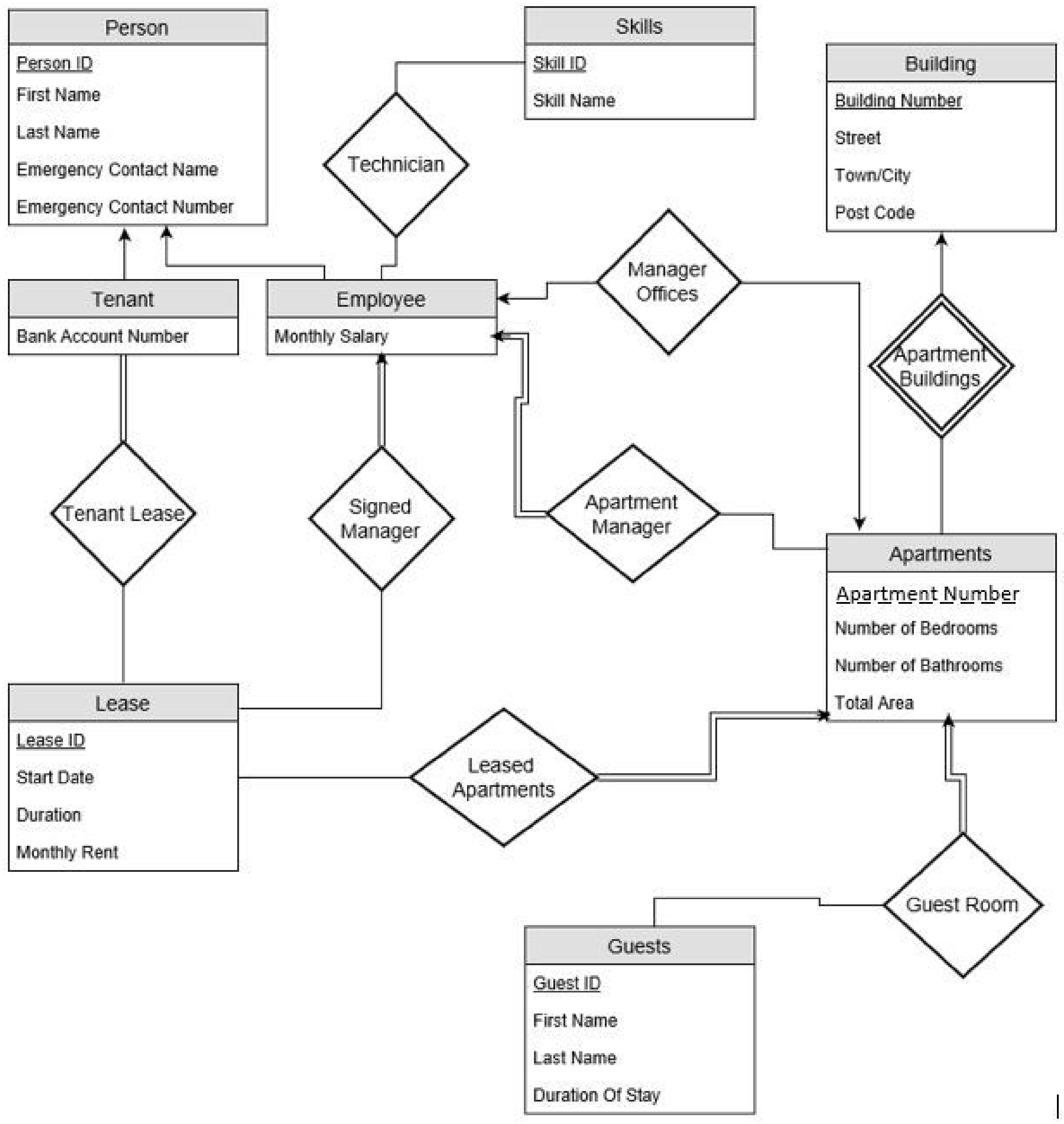
# CSC2042 Information Modelling Group Assignment

## Group 39 Details

| Name | Student Number | Email |
|---|---|---|
| Beth McAvoy | 40247407 | bmcavoy01@qub.ac.uk |
| Cliona O’Doherty | 40199292 | codoherty18@qub.ac.uk |
| Donal Maguire | 40179005 | dmaguire31@qub.ac.uk |
| Edel Crawford | 40206223 | ecrawford16@qub.ac.uk |
| James Ferguson | 40205873 | jferguson43@qub.ac.uk |
| Tracy O’Hagan | 40199221 | tohagan05@qub.ac.uk |

## ER Diagram



Person
Person ID
First Name
Last Name
Emergency Contact Name
Emergency Contact Number
Skills
Skill ID
Skill Name
Building
Building Number
Street
Town/City
Post Code
Technician
Tenant
Bank Account Number
Employee
Monthly Salary
Manager Offices
Apartment Buildings
Tenant Lease
Signed Manager
Apartment Manager
Apartments
Apartment Number
Number of Bedrooms
Number of Bathrooms
Total Area
Lease
Lease ID
Start Date
Duration
Monthly Rent
Leased Apartments
Guest Room
Guests
Guest ID
First Name
Last Name
Duration Of Stay

## List of Constraints and Assumptions in ER Diagram

**ASSUMPTIONS**

1. We are assuming that regardless of whether you are an employee, tenant or other, you are considered a Person in the database.

2. We are assuming that an employee can also be a tenant, therefore an employee can lease an apartment.

3. We are assuming that guests do not pay rent because they are not leasing the apartment they are staying in and therefore we only link Guest with Apartment and not lease.

**CONSTRAINTS**

1. Due to the lease table holding both current and expired leases, there is no way of telling which of the leases are expired and what are ongoing without creating a query.

2. As we have not linked the Guest with a lease, the system will therefore allow us to assign a Guest to an apartment which is occupied with enough tenants to fill each room.

## Database Design

1. BUILDINGS (Building_Number, Street, Town_City, Postcode)

2. APARTMENT (Building_Number, Apartment_Number, Bedroom_Number, Bathroom_Number, Total_Area, Manager_ID,)
   Building_Number is a foreign key to Building_Number in the Building Table
   Manager_ID is a foreign key to Person_ID in the Person Table

3. TENANTS (Tenant_ID, Bank_Account_No)
   Tenant_ID is a foreign key to Person_ID in Person Table

4. LEASE (Lease_ID, Building_Number, Apartment_Number, Start_Date, Duration_in_Months,Monthly_Rent, Manager_ID)
   (Building_Number, Apartment_Number) is a foreign key to (Building_Number, Apartment_Number) in Apartment Table

5. TENANT_LEASE (Tenant_ID, Lease ID)
   Tenant_ID is a foreign key to Person_ID in Person Table
   Lease_ID is a foreign key to Lease_ID in Lease Table

6. EMPLOYEES (Employee_ID, Monthly_Salary)
   Employee_ID is a foreign key to Person_ID in Person Table

7. PERSON (Person_ID, First_Name_Last_name, Emergency_Contact_Name, Emergency_Contact_Number)

8. SKILLS (Skill ID, Skill_Description)

9. TECHNICIAN (Employee_ID, Skill_ID)
   Employee_ID is a foreign key to Person_ID in Person Table
   Skill_ID is a foreign key to Skill_ID in Skills Table

10. GUESTS (Guest_ID, First_Name, Last_Name, Nights_of_Stay, Building_Staying_In, Apartment_Staying_In)
    (Building_Staying_In, Apartment_Staying_In) is a foreign key to (Building_Number, Apartment_Number) in Apartment Table

11. MANAGEROFFICES (Manager_ID, Building_Number, Apartment_Number)
    Manager_ID is a foreign key to Person_ID in Person Table
    (Building_Number, Apartment_Number) is a foreign key to (Building_Number, Apartment_Number) in Apartment Table

## SQL Querying

### Query One

**Description:**

In this query scenario, we want to list the Manager IDs, Tenant's names and their monthly rents, ordering by the manager ID and the tenant last name in ascending order.

**Approach:**

We firstly identify each the IDs of the manager that manages the apartment in which tenants are staying in and then gather their name and monthly rents. This is then sorted by the managers ID and each tenants last name in ascending order.

**SQL:**

```
SELECT Lease.Manager_ID, Person.first_name, Person.Last_Name, Lease.Monthly_Rent

FROM Person              JOIN TenantLease ON
Person.Person_ID = TenantLease.Tenant_ID
JOIN Lease ON
TenantLease.Lease_ID = Lease.Lease_ID

ORDER BY Lease.Manager_ID, Person.Last_Name ASC
```

## Query Two

**Description:**

In this query scenario, we want to find the apartments where the number of tenants staying in it is less than the number of bedrooms it has ie. where there is a free bedroom in the apartment.

**Approach:**

This query contains a nested SQL query. The nested query is used to find the number of tenants on each lease, it does this by counting the number of tenant ids and then grouping the table by lease id. Using this nested query we join the returned table to the left of a table which is the Lease table joined to the left of the Apartment table. We then check for which Apartments have a bedroom number greater than the counted tenant ids from the inner query.

**SQL:**

```
SELECT Apartment.Building_Number, Apartment.Apartment_Number, Lease.Lease_ID,
countTable1.countID Number_In_Flat, Apartment.Bedroom_Number

FROM Apartment       LEFT JOIN Lease ON
                              Apartment.Building_Number=Lease.Building_Number AND
                              Apartment.Apartment_Number=Lease.Apartment_Number
                     LEFT JOIN (SELECT Lease_ID, COUNT(Tenant_ID) countID
                     FROM TenantLease GROUP BY Lease_ID) countTable1 ON
                              Lease.Lease_ID=countTable1.Lease_ID

WHERE countID < Apartment.Bedroom_Number

ORDER BY Lease.Lease_ID;
```

## Query Three

**Description:**

Output the monthly salary, PersonID and Last name of a technician with the skill "Electrical", a last name starting with "S" and a monthly salary more than or equal to £1,200 sorting it by the technician's last name.

**Approach:**

We identify firstly the technician's with the skill "Electrical", then the technicians' last names starting with "S" and then lastly the monthly salary for the technician's which is more than or equal to £1,200.

**SQL:**

```
SELECT Person.Person_ID,Person.Last_Name

FROM Person         INNER JOIN Employee ON
                            Person.Person_ID = Employee.Employee_ID
                    INNER JOIN Technician ON
                            Person.Person_ID = Technician.Employee_ID
                    INNER JOIN Skill ON
                            Technician.Skill_ID = Skill.Skill_ID

WHERE Skill.Skill_Name = "Electrical" AND Person.Last_Name LIKE "S%" AND
Employee.Monthly_Salary >=1200

ORDER BY Person.Last_Name ASC;
```

## Query Four

**Description:**

In this query scenario, we want to display the Skill ID,  Employee's IDs and Employee's first names whose monthly salary is greater than £3,000 but less than £5,000.  We then want to group by Skill_ID and sort this in ascending order of monthly wage.

**Approach:**

We identify the Skill IDs and Employee ID from the technician table, the person's first name from the Person table and the Monthly Salary from the Employee Table. We do this by using left joins to joining the Technician and Person tables to the Employee Table

**SQL:**

```
SELECT Technician.Skill_ID, Technician.Employee_ID, Person.First_Name,
Employee.Monthly_Salary


FROM Technician       LEFT JOIN Employee ON

                               Technician.Employee_ID = Employee.Employee_ID

                      LEFT JOIN Person ON

                               Employee.Employee_ID = Person.Person_ID


WHERE Employee.Monthly_Salary > 3500 AND Employee.Monthly_Salary < 5000

ORDER BY Technician.Skill_ID, Employee.Monthly_Salary ASC
```

## Coping with Changes

### Scenario:

Queen’s Accommodation would like to evolve as time goes by, and significantly expand its service, to do this Queen’s Accommodation has decided on a few new expansions, including renovating a number of their buildings to have different categories including premium ensuite, premium standard, standard ensuite and standard and these different types of apartments should have a set monthly rent rate. Furthermore, they also want to record the details of all bookings made by guests to include the dates of signing in and out for each apartment and building. Lastly they also wish to record all maintenance jobs carried out by the technicians for each apartment. The nature of the job carried out and the apartment in which the work was done should be recorded.

### Report:

These expansion plans could be addressed with a few changes to the database and some expansions are complex and others more simple. Starting off with the modification for the different types of apartments, how we would handle this is firstly modifying the building table to include an apartment type field and this would be used to store whether the building is either premium ensuite, premium standard, standard ensuite and standard apartment. Then we would add a new table to the database called “Rental_Rates” which would store the set monthly rent for each type of apartment and because of this new table, we would remove the field “Monthly_Rent” from the lease table as this would no longer be required.

To address the bookings made by guests, we would firstly add a “Booking” table to include “Booking_ID”, “Sign_In_Date”, “Sign_Out_Date”, “Guest_Name”, “Guest_Number”, “Building_ID”, “Apartment_ID” where the “Booking_ID” would be a Primary key for the table and the “Building_ID” and “Apartment_ID” would be the foreign keys linked to the “Building” and “Apartment” table, linking the “Booking” table to the specific apartment for each booking that the guest is staying in. This alteration to our database system would be relatively difficult as we would have to consider the foreign key relationships to the table and ensure they are linked up correctly to avoid any error within the code and ensure data integrity.

Another relatively complex change would be addressing the need to record maintenance jobs. The details of all maintenance jobs carried out on each specific apartment would be recorded within the database. To do this, we would create a table “Maintenance_Jobs” with the following fields “Maintenance_ID”, “Job_Type”, “Date”, “Apartment_ID” and “Technician_ID”. The “Apartment_ID” would be a foreign key linking from the “Apartment” table and the “Technician_ID” would be a foreign key from the “Technician” table. This table has a primary key “Maintenance_ID” to allow each maintenance job to be uniquely identified.

## Hardest to Model

We found the relationship between tenants and their lease particularly difficult as we were unable to include the “Person_ID” of the tenant inside the “Lease” table as one or more tenants can lease an apartment and this would result in a multi-valued attribute within the table and therefore this would result in bad database design and poor data integrity as the accuracy and consistency of the data being entered into the records would be inaccurate.

Therefore to overcome this problem, we created a separate table called “TenantLease” to include the “Person_ID” of the tenant and “Lease_ID” of the lease, having them as foreign keys from the “Person” and “Lease” tables and this would ensure that more than one tenant can lease the same apartment therefore resolving the problem.

Despite finding this task difficult, we worked as a group to overcome the problem and find a suitable solution.

## Individual Contribution Record

| Group Member<br>Student Number | Task (i)<br>ER | Task (ii)<br>Design | Task (iii)<br>Query | Task (iv)<br>Changes | Total |
|---|---|---|---|---|---|
| 40247407 | 20 | 20 | 20 | 20 | 80 |
| 40199221 | 20 | 20 | 20 | 20 | 80 |
| 40206223 | 20 | 20 | 20 | 20 | 80 |
| 40205873 | 20 | 20 | 20 | 20 | 80 |
| 40199292 | 20 | 20 | 20 | 20 | 80 |
| 40179005 | 0 | 0 | 0 | 0 | 0 |
| | 100 | 100 | 100 | 100 | 400 |