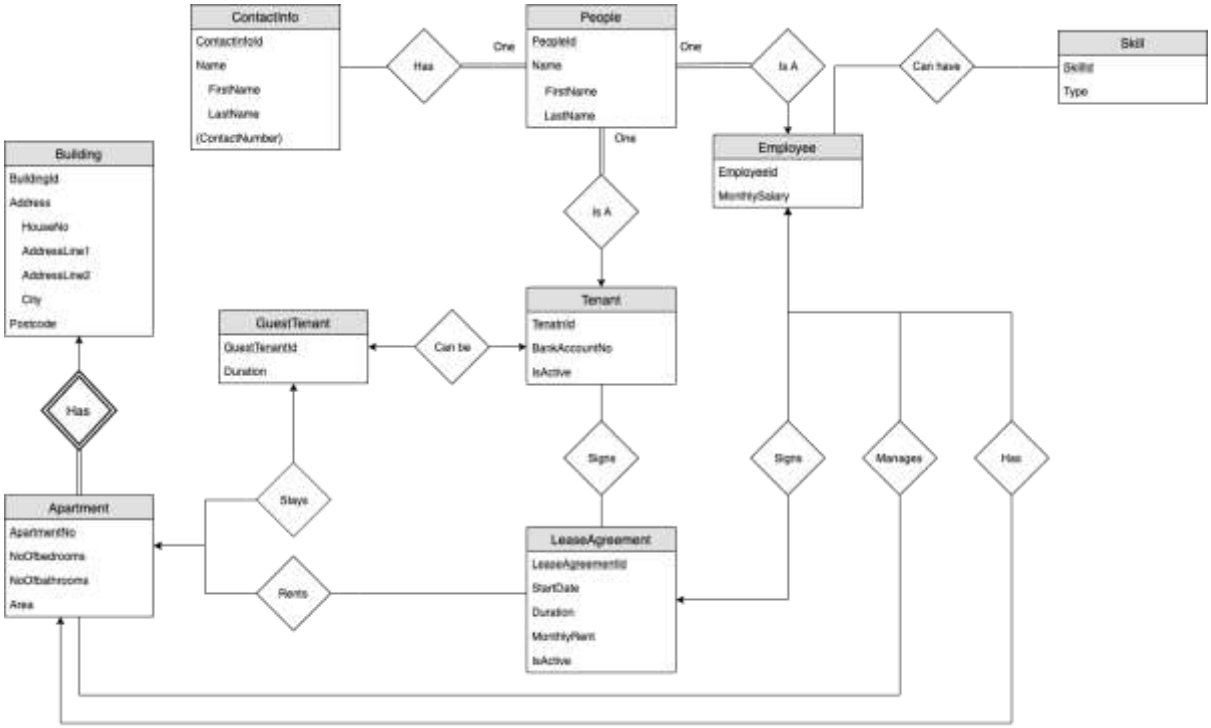
Team 49 Information Management / Modelling Group Assignment

Name	Student Number	Email Address
Christian Antonny	40251583	cantonny01@qub.ac.uk
James Earls	40206210	jearls01@qub.ac.uk
Cormac McGrath	40183351	cmcgrath61@qub.ac.uk
Elise Burton	40227633	eburton04@qub.ac.uk
Ben Black	40235191	bblack713@qub.ac.uk

Team 49 E-R Diagram



Assumptions / Constraints

ContactInfo

We assumed that each person saved in the ‘People’ table could have on or more emergency contacts, but there must be only one person in the ‘People’ table that an emergency contact is for i.e. you can’t have contact information for the next of kin of someone not in the database.

People

We enforced participation constraints between the ‘is-a’ relationships between the ‘People’ table and the ‘Tenant’ table, as well as the ‘Employee’ table, because every employee and every tenant must be a person in the ‘People’ table.

We also used a many to one relationship between the ‘People’ table and the ‘Tenant’ and ‘Employee’ tables because one person can’t be multiple tenants/multiple employees.

Employee

We assumed that every employee that is saved in the ‘Employee’ table is either a manager or a technician or both, so we’ve used participation constraints between these tables.

Tenant

We assumed that every tenant ‘Is A’ Person, every tenant ‘Can be’ a Guest Tenant and that every tenant will ‘Sign’ a Lease, these are the assumptions that we made for relationships between tenant to other tables. We also assumed a tenant has an Attribute of a Boolean function which is the ‘IsActive’ attribute inside the Tenant table.

GuestTenant

We assumed that there is a many to many relationships with GuestTenant and Tenant and that a GuestTenant ‘Can be’ a Tenant. We also assumed that a GuestTenant ‘Stays’ within an apartment and that there are many apartments to many GuestTenants. We also added a Duration attribute to keep track of how long the GuestTenant will be staying there.

Skill

In the skill table we have a one to one relationship with employee and the relationship tag ‘Can have’ and we have an attribute for the Type of skill they will have out of the three choices.

LeaseAgreement

We used a many to many relationship for defining the ‘signing’ relationship set between a LeaseAgreement and Tenants. Each LeaseAgreement is also ‘signed’ by a ‘Manager’ which is a many to one relationship. Every LeaseAgreement has a many to one relationship with ‘Apartment’ defining the ‘rents’ relationship set.

Building

We’ve assumed each building will have many apartments and we have used a ‘has’ identifying relationship between the ‘Building’ and ‘Apartment’ entity sets to represent that each building may have multiple apartments that need to be uniquely identified if it is required to get the address and postcode of an individual apartment.

Apartment

We have assumed that employees and more specifically managers have the ability to manage one or more apartments.

Database Design

Building (**BuildingId**, HouseNo, AddressLine1, AddressLine2, City, Postcode)

Apartment (**ApartmentNo**, *BuildingId*, *ManagerId*, NoOfBedrooms, NoOfBathrooms, Area)

LeaseAgreement (**LeaseAgreementId**, *ManagerId*, StartDate, Duration, MonthlyRent, IsActive, *ApartmentNo*, *BuildingId*)

Employee (**EmployeeId**, *PeopleId*, Contact Number, AddressLine1, AddressLine2, City, Postcode, Monthly Salary)

Technician (***EmployeeId***, ***SkillId***)

Skill (**SkillId**, Type)

Tenant (**TenantId,** *PeopleId,* BankAccountNo, IsActive)

LeaseTenant (***TenantId***, ***LeaseID****)*

GuestTenant (**GuestTenantId**, Duration, *apartmentNo*, *buildingId*)

Manager (**ManagerId**, *employeeId*, *apartmentNo*, *buildingId)*

People (**PeopleId**, name_firstName, name_lastName)

ContactInfo (**ContactInfoId**, *PeopleId*, name_firstName, name_lastName, teleNumber)

SQL Query 1 - Produce a query showing all guest tenants, their apartment numbers, the building they are in and their emergency contact phone number

SQL QUERY

```
SELECT guestTenant.guestTenantId AS GuestID,
apartment.apartmentNo AS ApartmentNumber,
apartment.buildingId AS BuildingID,
contactInfo.teleNumber AS Telephone_Number,
contactInfo.name_firstName AS FirstName,
contactInfo.name_lastName AS LastName
FROM guestTenant
INNER JOIN apartment ON guestTenant.apartmentNo = apartment.apartmentNo AND
guestTenant.buildingId = apartment.buildingId
INNER JOIN tenant ON guestTenant.tenantId = tenant.tenantId
INNER JOIN people ON tenant.peopleId = people.peopleId
INNER JOIN contactInfo ON people.peopleId = contactInfo.peopleId
ORDER BY guestTenant.guestTenantId ASC;
```

This query's objective is to pull from the database four pieces of data from the database and present it, The SQL query pulls the GuestID, BuildingID, Telephone Number, FirstName & Last Name and orders it in ascending order from the database.

The way that the query achieves this is by starting off selecting the all of the different tables and the attributes inside each of the tables, it achieves this by selecting the table its under and then selects the column. E.g. guestTenant.guestTenantID;

The next part of the select SQL statement is the "AS" statement, this part will give the Column a alias name, So when selected it makes it more readable for the user, e.g. for the guestTenant.guestTenantId selection it was GuestID instead so easier to read.

All of these statements then go above the "FROM" statement, this simply says that the first table its pulling information from is guestTenant so just the starting default table information is being taken from and the table which will be linking to the rest of the tables via joins.

I then chose to use a left join onto apartment, The reason for this is because a left join on SQL will pull information from guestTenant and then only take information that is requested (this is the select statement which is apartment.apartmentNo) so by using a left join on this statement I'll only be pulling in the apartmentNo piece of information.

The next two lines are the same command in SQL (Inner join) although I had done it with the last two tables which are building and contactInfo, I had chosen to use a inner join on both of those because I only required one column from the building but for contact I required more columns. But I'm still fine to use a left join in this scenario to get all of those columns for contactInfo I required the First Name, Last Name & the Telephone Number.

The way that the left join actually links the two tables together is by linking two attributes together and making them = eachother, this is after the "ON" section of the LEFT JOIN statement. A good example of this is guestTenant.apartmentNo = apartment.apartmentNo this part of the statement is simply linking apartmentNo's in both tables together.

SQL Query 2 - Create a query to find the number of active tenants in each building and display the results under a column named "Number of Tenants"

```
SELECT
   building.address_addressLine1 AS 'Building',
   COUNT(leaseTenant.tenantId) AS 'Number of Tenants'
FROM
   leaseTenant
      INNER JOIN
   leaseAgreement ON leaseTenant.leaseAgreementId = leaseAgreement.leaseAgreementId
      INNER JOIN
   building ON leaseAgreement.buildingId = building.buildingId
WHERE
   leaseAgreement.isActive = TRUE
GROUP BY leaseAgreement.buildingId
ORDER BY building.address_addressLine1;
```

First, I have selected the building addresses to show the building that the Tenants are staying in and used the AS key word to make the attribute headings more user friendly and condense the name from “address_addressLine1” to “Building”.

I have then used the “COUNT” key word to pull the number of tenant Id’s from the leaseTenant table and display this count under “Number of Tenants using the AS keyword again.

Using the “FROM” key word I then select the tables that are required to display the queries. In the case of this query, we require use of the LeaseTenant table, the Building table and the Lease Agreement table but then needed to use Join’s to combine the data together.

I then used an “INNER JOIN” to join leaseAgreement to leaseTenant in order to find Tenants whose leases are active which will determine if they are currently staying in the building or not and another “INNER JOIN” building to leaseAgreement in order to retrieve the address of the building.

The WHERE key word will allow us to only count the tenants that have a true Boolean value for being active and therefore will only count tenants that actually are

I have also used the “GROUP BY” key word to groups rows that have the same values into summary rows grouped the results by buildingId and used the “ORDER BY” key word to sort the buildings by ascending alphabetical order.

SQL Query 3 - Find a list of technicians that currently live in an apartment

```
SELECT
   e.`employeeId` AS 'Employee ID',
   p.`name_firstName` AS 'First Name',
   p.`name_lastName` AS 'Last Name',
   (e.`monthlySalary` - (la.`monthlyRent` / (SELECT
         COUNT(`tenantId`)
      FROM
         leaseTenant
      WHERE
         `leaseAgreementId` = la.`leaseAgreementId`))) AS 'Salary After Rent'
FROM
   people p
      INNER JOIN
   employee e ON p.peopleId = e.peopleId
      INNER JOIN
   tenant t ON p.peopleId = t.peopleId
      INNER JOIN
   leaseTenant lt ON lt.tenantId = t.tenantId
      INNER JOIN
   leaseAgreement la ON la.leaseAgreementId = lt.leaseAgreementId
WHERE
   e.employeeId IN (SELECT DISTINCT
         t.employeeId
      FROM
         technician t)
      AND t.isActive = 1
      AND la.isActive = 1
ORDER BY p.peopleId ASC;
```

This query's task is to retrieve records from the database that consist of EmployeeId, First name, Last name and Salary after rent.
This is done using joins and other SQL statements to get the count or get a list of Ids to compare against. Firstly, I renamed the attributes to make the final result easier to read and understand, this done using the 'AS' keyword. It allowed me to rename the columns to something more readable. After this I used another query to get the count of tenants on a specific lease agreement using the leaseTenant table and counting the ids that match the leaseAgreementId I passed to the query. This was used to divide the monthly rent to get an even split of what even tenant pays per month. Finally, I took this value away from the employee's monthly salary to gage their income after rent.
From this I needed to use several 'INNER JOIN' statements in order to access the appropriate tables to find if the tenants were still active and also if the they had any lease agreements and that they were still active.
In order to make sure that my query only used employees I had to query the technician table to get a list of employeeIds. This meant that because I had used 'INNER JOIN' statements I could access the employeeId from the employee table using its foreign key from the people table. I used the 'IN' statement to compare each employeeId of my query results and check if it was within the list returned from the technician query results. Finally, I used an 'ORDER BY' statement followed by the 'ASC' keyword to logically order my query results.

SQL Query 4 – Produce a query to display the average salary of a technician employed at Queen's Accomodation based on their skills

```
SELECT
   technician.skillId,
   skill.type AS Skill,
   COUNT(employee.employeeId) AS Number_Of_Skilled_Employees,
   ROUND(AVG(monthlySalary) , 2) AS Average_Salary_for_skill_type
FROM
   employee
      INNER JOIN
   technician ON employee.employeeId = technician.employeeId
      INNER JOIN
   skill ON technician.skillId = skill.skillId
GROUP BY skill.type
ORDER BY COUNT(employeeId) DESC;
```

QUERY RESULT

skillId	Skill	Number_Of_Skilled_Employees	Average_Salary_for_skill_type
2	Carpentry	5	535.70
1	Electrical	3	563.28
3	Plumbing	2	544.72

The SQL query uses the 'AVG()' function to calculate the average monthly salary from the 'monthlySalary' attribute in the 'emoployee' table and then the 'AS' keyword changes the column header to 'Average_Salary_for_skill_type' which is more suitable and easier to read.
I wanted to present the salaries in a suitable form because they represent currency so I used the ROUND() function to round each salary to two decimal places.
The COUNT() function is used to count the amount of employees who possess each skill, this column was renamed to 'Number_Of_Skilled_Employees' using the 'AS' keyword and this is also the attribute that is used to order the table using 'ORDER BY' with the highest number first to show the most popular skill amongst employees, this is achieved using the 'DESC' keyword.
Data from three different tables 'employee', 'technician' & 'skill' are required in this query so I used two INNER JOINs to join the tables together. The 'technician' and 'employee' tables where joined by equating 'employee.employeeId' and tables 'technician' and 'skill' were then joined by equating 'technician.skillId' and 'skill.skillId'.
The keyword 'GROUP BY' groups all data by the skill type, thus splits up the average monthly salary and number of skilled employees into the 4 rows.

Coping with Changes - Queen's Accommodation Database Management System (DBMS) Future Expansion Plans

Queen's Accommodation (QA) is widely known for its outstanding service for their clients, e.g. high-quality accommodations, reliable customer service response and their ever so welcoming reception at the entrance of the campus. In terms of placing, Queen's Accommodation easily lands the top 100 school accommodation ranking. However, this is not the end of its business ventures. One of the expansion plans of QA is increasing the facilities in the campus. The highlight of these facilities is implementing a shopping mall in the heart of the campus.

The basic management of a shopping mall requires various information such as data on the shops, shop owners, monthly/annually rental and possibly season parking for certain authorised personnel. This new batch of information will be keyed into the QA's Database, which means that there must be a change in the design of the system. To date, the database includes People, contactInfo, Employee, Technician, Tenant, GuestTenant, Manager, Skill, Lease Tenant, Lease Agreement, Apartment and Building. Should QA take action on this particular expansion plan, (People, ContactInfo, Employee, Tenant, Manager, Lease Tenant, Lease Agreement and Building) tables will be affected.

Firstly, the shopping mall will be a building owned by QA, which will have its own buildingId. The building will be managed by a Manager. This means that each shop will have its own shopId and their own unique rental price. Since the building belongs to QA, there can be lease on the shops which have various rent prices and they are due payable either monthly or annually. The DBMS will specify under Tenant that it will need the information of the tenant (Id, peopleId, BankAccountNo), and to be added in new attributes (Monthly/Annually Rent) and will also be referenced to Lease Tenant as well as Lease Agreement (The Lease Agreement table will state whether the lease is for an apartment or a shop). In terms of Employee, the shopping mall will need a general manager (Manager), and a body to govern over the whole shopping mall. This includes new tables under Employees. Technicians will not only be hired for the purpose of the accommodations but also the management of the machines to be implemented and installed in the shopping mall. Janitors and Security Guards will be hired as well. Each table under employee (Manager, Technician, Security Guard, Janitor) will include their employeeId.

In conclusion, Queen's Accommodation might have plans on expanding their services. With the addition of shopping mall in the campus, the database management system will have to adjust its ER Diagram. These adjustments include additional attributes in various tables, mainly People, ContactInfo, Employee, Tenant, Manager, Lease Tenant, Lease Agreement and Building. It also requires for the design of the database to have new tables under Employee, such as Janitor and Security Guard. Another factor is a new table labelled Shops under Building as 'Building' could now mean apartment as well as shops. This also means that ShopId will be an attribute under the table Shops. Nevertheless, this course of actions is one of the main focus of QA's expansion plan as it not only generates a higher revenue for QA, but also improves the quality of life for the clients of the accommodation.

Answer to Hardness Question

The ‘Manager’ table was the most difficult to model, because of the complex relationships between multiple tables and having to use multiple foreign keys to model this.

Individual Contributions Record

Group Member Student number	Task (i)	Task (ii)	Task (iii)	Task (iv)	Group Member Totals
40251583	5	10	0	100	115
40206210	23.75	20	25	0	68.75
40183351	23.75	30	30	0	83.75
40227633	23.75	20	22.5	0	66.25
40235191	23.75	20	22.5	0	66.25
Total	100	100	100	100	

Individual Contributions Narrative

Narrative (Cormac McGrath):
I have contributed to my group by helping organise some of the group meetings and I spent considerable time designing the ER Diagram. I then also setup a Trello board for managing the project. I also setup a Gitlab repository for sharing our SQL statements as well creating a shared database on AWS for use during the project. I also designed, created and inserted data into the Building, Apartment and LeaseAgreement. I wrote a detailed query (Query 3) for finding employee's renting on site and the profit they take home after rent. I also spent time helping anyone in my group who needed it, especially when it came to MySQL and MySQL Workbench.

Narrative (James Earls):
In this project I have Contributed to my group by helping with the design of the ER Diagram. I then also did the database design for the Employee, Technician and Skill tables and create statements for these tables too. I also contributed to the inserts for these tables and wrote a query that finds the number of active tenants in a building.

Narrative (Benjamin Black):
I have contributed in the starter stage where we all worked together to do the ER Diagram, I added In the relational tables for Manager, GuestTenant tables and contributed the SQL Query at the end for taking from guestTenant, Apartment, Building and ContactInfo and organising them all I also think I had a good hand in organising times for team meetings.

Narrative (Elise Burton):
I have contributed to my group by helping to design the ER Diagram which will we all worked on collectively over several team meetings. I wrote assumptions and constraints for the 'ContactInfo', 'Employee' and 'People' tables. I created tables 'tenant' and 'leaseTenant' in the schema and wrote insert statements to populate them with data. My SQL was Query 4 that finds the average monthly salary of technicians grouped by their skill.

Narrative (Christian Antonny):
While I spent time in task 2 with creating and inserting the data for People and Contact Info, I also minimally helped my group with the designing the ER diagram with the exception of assumptions and constraints. My key contribution to the group was completing task 4, Coping with Changes.