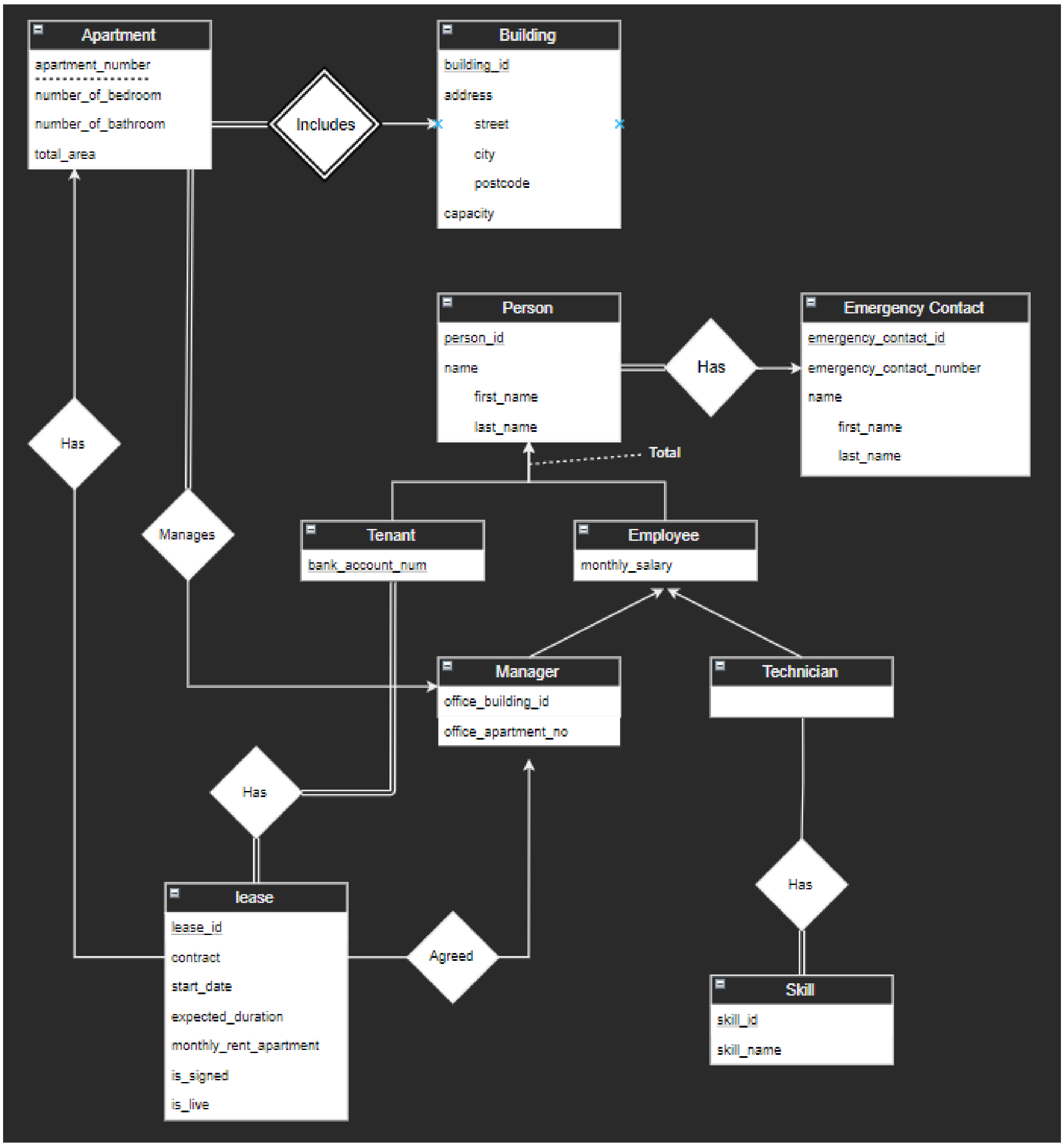
CSC 2041/42

Group #32

Name	Student ID	Student e-mail
Conor Browne	40234668	cbrowne27@qub.ac.uk
David Hamilton	40231146	dhamilton32@qub.ac.uk
George Jenkinson	40226329	gjenkinson01@qub.ac.uk
Adam McNeill	40204945	amcneill21@qub.ac.uk
Josh Beatty	40230451	jbeatty05@qub.ac.uk

Task I: Entity Relationship Diagram

E-R diagram based on the Queen's Accommodation scenario detailed in the assignment.pdf



Task I: Constraints and Assumptions

Constraints

Cardinality Constraints

1. [Apartment INCLUDES Building] - INCLUDES is a *One or Many - to - One* relationship.
2. [Person HAS Emergency Contact] - HAS is a *One or Many - to - One* relationship.
3. [Manager MANAGES Apartment] - MANAGES is a *One - to - One or Many* relationship***[1]**.
4. [Lease HAS Apartment] - HAS is a *Many- to - One* relationship***[2]**.
5. [Manager AGREES Lease] - AGREES is a *One - to - Many* relationship.
6. [Tenant HAS Lease] - AGREES is a *One or Many - to - One or Many* relationship.
7. [Technician HAS Skill] - HAS is a *Many - to - One or Many* relationship.

Generalisations Specialization Constraints

8. [Employee / Tenant ISA Person] - ISA is a *total disjoint generalisation****[3]**
9. [Manager / Technician ISA Employee] - ISA is an *overlapping generalisation****[4]**

Constraint Explanations

- ***[1]** - A manager manages at least one apartment - his office - therefore there is a *One or Many* constraint on the MANAGES relationship. Specialized attributes of the Manager Entity specify which apartment is the manager's office.
- ***[2]** - A lease must be related to an apartment in order to exist.
- ***[3]** - As specified in the context statement, "QA manages two disjoint categories of people: tenants and employees". Total disjoint generalisation ensures all people are at least and at most one of the two categories of people QA manages.
- ***[4]** - As specified in the context statement, "Employees may be managers or technicians (or both)". An overlapping generalisation ensures an employee can be either of the specialisations or both.

Assumptions

1. Assuming the **generalisation of people must be a disjoint relationship** based on the extract *"QA manages two disjoint categories of people: tenants and employees"* from the context statement.
2. Assuming the **apartment entity must be a weak entity** - apartments with the same apartment number can exist in separate buildings.
3. Assuming the statements *"Each manager manages at least one apartment"* and *"Each manager has an office in one of the apartments"* implies the **minimum one apartment is the manager's office** - . To specify the office details, attributes have been added to the Manager specialization.

Task II - The Database design

Key:
Tablename(Primary Key, @ForeignKey, attribute)

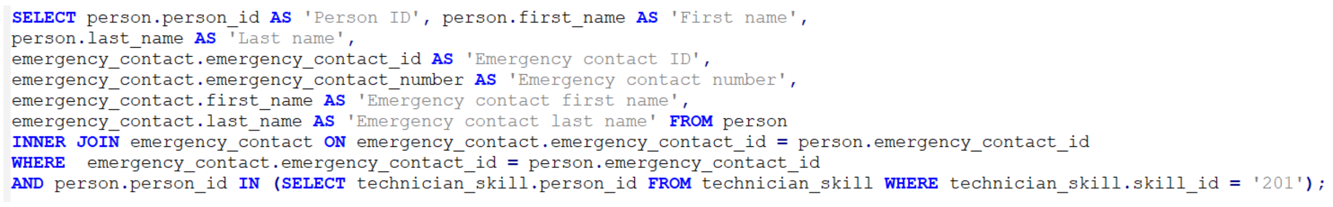
Building (building_id, street, city, postcode, capacity)
Apartment (apartment_number, @building_id, num_of_bed, num_of_bath, total_area, @person_id) *Building_ID is used because apartment number isn't unique unless paired with the building that the apartments within. Person ID references the manager which manages the apartment.*
Person (person_id, first_name, last_name, @emergency_contact_id) *Emergency_contact_id foreign key links person to their emergency contact*
Emergency_Contact (emergency_contact_id, emergency_contact_number, first_name, last_name)
Tenant (@person_id, back_acc_num) *Person_id foreign key links tenant to the person they relate to*
Employee (@person_id, monthly_sal) *Person_id foreign key links employee to the person they relate to*
Manager (@person_id, office_apartment_number, office_building_id)
Technician (@person_id) *Person_id foreign key links technician to the person they relate to*
Skill (skill_id, skill_name)
Technician_Skill (@person_id, @skill_id) *These foreign keys are working as a link between a technician and which skills they possess*
Lease (lease_id, contract, start_date, expected_duration, monthly_rent_apartment, is_signed, is_live)
Tenant_Lease (@lease_id, @person_id) *Act as a link between tenants and the leases they hold*
Manager_Lease (@lease_id, @person_id) *Act as a link between a manager and the leases they've signed off on*
Apartment_Lease (@lease_id, @(apartment_number, building_id)) *Acts as a link between a lease and the apartment related to it. '(apartment_number, building_id)' is a composite foreign key because it needs to reference both the apartment number and the building that apartment is in.*

Task III: SQL Querying

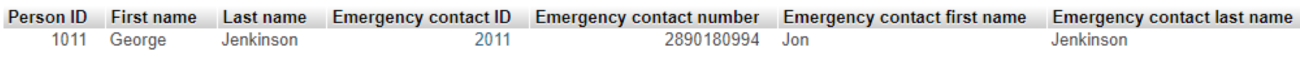
Query 1

Justification: Queen's accommodation wants to know the emergency contact of all of the plumbers in their employment. Return the person_id, first name, and last name as well as the emergency contact information of all plumbers.

Overview: This Query links emergency contact to person while checking that the contact details only relate to somebody with the plumbing skill. The nested query returns any of the plumbers by searching for the employee id of anybody with the skill '201'.



```
SELECT person.person_id AS 'Person ID', person.first_name AS 'First name',
person.last_name AS 'Last name',
emergency_contact.emergency_contact_id AS 'Emergency contact ID',
emergency_contact.emergency_contact_number AS 'Emergency contact number',
emergency_contact.first_name AS 'Emergency contact first name',
emergency_contact.last_name AS 'Emergency contact last name' FROM person
INNER JOIN emergency_contact ON emergency_contact.emergency_contact_id = person.emergency_contact_id
WHERE  emergency_contact.emergency_contact_id = person.emergency_contact_id
AND person.person_id IN (SELECT technician_skill.person_id FROM technician_skill WHERE technician_skill.skill_id = '201');
```



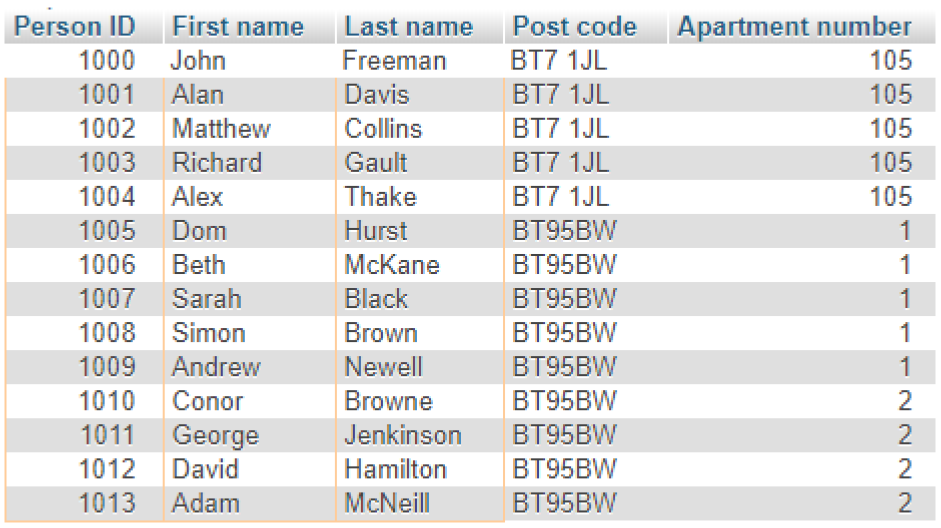
Person ID	First name	Last name	Emergency contact ID	Emergency contact number	Emergency contact first name	Emergency contact last name
1011	George	Jenkinson	2011	2890180994	Jon	Jenkinson

Query 2

Justification: A manager at Queen's Accommodation is moving on, they would like to alert any tenants living in an apartment managed by 'Matthew Collins' of this fact. This query will return the full name, person id, apartment number and building postcode for anybody living in apartments managed by 'Matthew Collins'.

Overview: This query links together the person_id to their lease and connects the corresponding manager_id. The nested query then searches for the manager_id of 'Matthew Collins' allowing the query to only return the details of people in apartments managed by Matthew.

```
SELECT person.person_id AS 'Person ID',
person.first_name AS 'First name',
person.last_name AS 'Last name',
building.postcode AS 'Post code',
apartment_lease.apartment_number AS 'Apartment number' FROM person
INNER JOIN tenant_lease ON person.person_id = tenant_lease.person_id
INNER JOIN manager_lease ON tenant_lease.lease_id = manager_lease.lease_id
INNER JOIN apartment_lease ON apartment_lease.lease_id = manager_lease.lease_id
INNER JOIN building ON apartment_lease.building_id = building.building_id
WHERE person.person_id = tenant_lease.person_id
AND manager_lease.lease_id = tenant_lease.lease_id
AND apartment_lease.lease_id = manager_lease.lease_id
AND apartment_lease.building_id = building.building_id
AND manager_lease.person_id
IN (SELECT person.person_id FROM person WHERE person.first_name = 'Matthew' AND person.last_name = 'Collins');
```

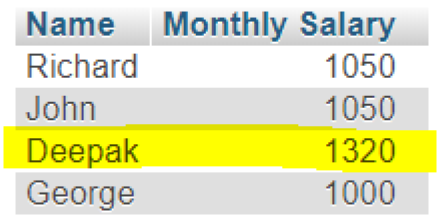


Person ID	First name	Last name	Post code	Apartment number
1000	John	Freeman	BT7 1JL	105
1001	Alan	Davis	BT7 1JL	105
1002	Matthew	Collins	BT7 1JL	105
1003	Richard	Gault	BT7 1JL	105
1004	Alex	Thake	BT7 1JL	105
1005	Dom	Hurst	BT95BW	1
1006	Beth	McKane	BT95BW	1
1007	Sarah	Black	BT95BW	1
1008	Simon	Brown	BT95BW	1
1009	Andrew	Newell	BT95BW	1
1010	Conor	Browne	BT95BW	2
1011	George	Jenkinson	BT95BW	2
1012	David	Hamilton	BT95BW	2
1013	Adam	McNeill	BT95BW	2

Query 3

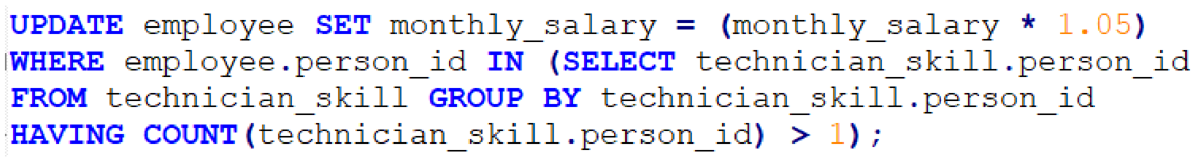
Justification: In order to recognise the efforts and adaptability of technicians who bring more than 1 skill to the job, an option to increase their salary by a percentage should be in place - without affecting the salary of those who only have 1 skill, or none at all.

Overview: With 5 current employees, only 4 of them have skills. This query considers the skill-set of an employee by selecting their 'ID' and grouping each employee who possess more than one *skill_id* together. Once the employee/s have been grouped, an update will occur, raising their current monthly salary by 5%.
In this instance, the employee 'Deepak' is the only person who holds more than 1 skill.



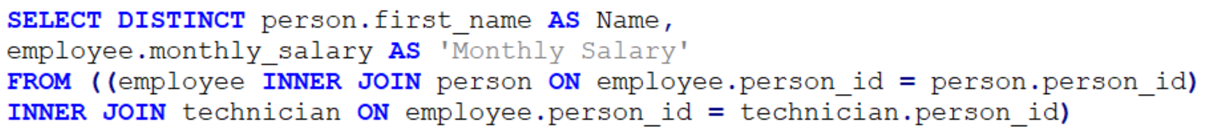
Name	Monthly Salary
Richard	1050
John	1050
Deepak	1320
George	1000

Deepak's current monthly salary

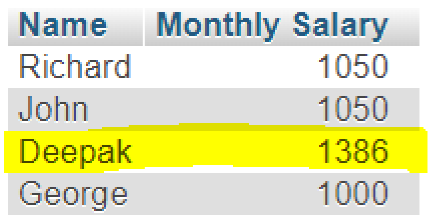


```
UPDATE employee SET monthly_salary = (monthly_salary * 1.05)
WHERE employee.person_id IN (SELECT technician_skill.person_id
FROM technician_skill GROUP BY technician_skill.person_id
HAVING COUNT(technician_skill.person_id) > 1);
```

Using a select query, you can view the current employees with skills. From the diagram above and below, it is clear that only Deepak's salary has increased by 5%



```
SELECT DISTINCT person.first_name AS Name,
employee.monthly_salary AS 'Monthly Salary'
FROM ((employee INNER JOIN person ON employee.person_id = person.person_id)
INNER JOIN technician ON employee.person_id = technician.person_id)
```



Name	Monthly Salary
Richard	1050
John	1050
Deepak	1386
George	1000

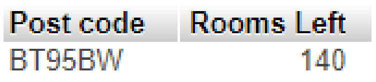
Deepak's new monthly salary

Query 4

Justification: A common question for prospective tenants and an insightful statistic for advertising, would be to check how many apartments are remaining to rent for accommodation in a given building. Particularly useful for displaying on the Queen’s Accommodation website.

Overview: Considers all apartments in a particular building where the apartment isn’t occupied. This is achieved by finding all of the records of tenant leases for a particular apartment where the lease is live, and subtracting this from the capacity of the building.

```sql
SELECT building.postcode AS 'Post code',
(building.capacity - COUNT(building.postcode)) AS 'Rooms Left'  FROM building
INNER JOIN apartment ON apartment.building_id = building.building_id
INNER JOIN apartment_lease ON apartment.building_id = apartment_lease.building_id
INNER JOIN lease ON apartment_lease.lease_id = lease.lease_id
WHERE building.building_id = apartment_lease.building_id
AND apartment_lease.lease_id = lease.lease_id
AND lease.is_live = 1
AND building.postcode = 'BT95BW'
GROUP BY building.postcode;
```



```
Post code   Rooms Left
BT95BW             140
```

Task IV - Coping with possible Changes

Change
With the expansion of Queen’s Accommodation, changes have been made to the pricing schemes of different apartments. Pricing for apartments could vary based on number of people sharing a kitchen, or inclusion of rates and amenities. Queen’s Accommodation would also like to keep track of which tenant occupies which room in an apartment, which is not supported by the current design and implementation of the DBMS.

Potential implementation
We may implement a new table called ‘apartment_type’ to specify what each apartment includes and their monthly rate. A foreign key would need to be added to the ‘apartment’ linking these two tables so that the apartment’s details can be obtained.

To address the new monthly costs, we could remove ‘monthly_rate’ from ‘lease’ and instead have it be calculated based on its connection to an apartment and an apartment type through ‘apartment_lease’ to ‘apartment’ and ‘apartment’ to ‘apartment_type’.

Finally, to keep track of a tenant’s room within an apartment, we could create a new attribute in ‘tenant_lease’ specifying the tenant’s room number. This lease would still be connected to the apartment but would now allow for the tenant’s room to be known.

Complexity
The change to include the apartment types is fairly complex as it requires the creation of a new table and a link to the apartment table using a foreign key. The change to keep track of the room numbers is fairly simple as it could be achieved with just the addition of a single attribute. There is no guarantee that these changes could be made in such a fashion, this is just a concept of how these changes could be implemented.

Task V: Most Challenging Aspects of the Project

Name	Comment
Conor Browne	Personally, I found creating the queries the most difficult aspect of the project. Queries is something I've always found a bit tricky, when we went off to work on ideas for our 4 queries, I did struggle to get some of the queries working as intended to get the results we wanted. But this is a good thing for me to be aware of now as it will prepare me for the exam in the later parts of the course.
David Hamilton	The hardest part of the project was the time constraints of constantly updating and changing the ERD to meet our requirements. After seeming relatively simple at the beginning, it was only when implementing the database that we discovered certain relationships weren't quite right and the database wasn't behaving exactly as it should have been. The diagram was something that we continuously kept referring back to, to update or change.
George Jenkinson	I found the ER diagram the most challenging aspect of the project. We went though many redesigns when we realised that the diagram did not support our original assumptions or how were implementing the design in the schema. Even up to the queries stage we were noticing issues with regards to the ER design in which we had to change the diagram and thus the schema to achieve our desired queries.
Adam McNeill	I found the SQL queries to be the most difficult part of the task. I felt that they needed to be reasonably complex in order to satisfy the task, and this meant that we first had to think of examples of queries that would be complex enough, and then be able to implement a query with that complexity. This involved improving my SQL skills.
Josh Beatty	One of the areas we remodelled the most in our ER Diagram stage was the relationship sets related to the Lease entity, involving the; Apartment, Manager and Tenant entities. There were many constraints to consider in addition to one of the entities being a weak entity (apartment). We revisited and updated our ER Diagram several times during the early stages of the project, requiring us to re-evaluate our work in subsequent tasks.

Individual contributions record

Student ID	Task (i)	Task (ii)	Task (iii)	(Task iv)	Group member Total
40234668	20	20	20	20	80
40231146	20	20	20	20	80
40226329	20	20	20	20	80
40204945	20	20	20	20	80
40230451	20	20	20	20	80
TOTAL	100	100	100	100	400

Individual Contributions Narrative

Name	Individual Contribution
Conor Browne	I contributed my own ideas for the ERD diagram to compare it with the rest of the team to put together the final ERD for the project. I created some of the tables within the schema and populated these tables with appropriate data. I also assisted with reviewing the query's as well as contributing ideas for the queries. I contributed ideas for task 4 outlining expansions to the database to accommodate students from other universities. I consistently attended and contributed to all group meetings.
David Hamilton	I created my own version of the ERD to discuss alongside everyone else's versions to bring all of our ideas together and put it into one diagram. As well as this, I brought a few constraints and assumptions forward to include. As we split the database design evenly across the group, my role was to take the about a fifth of the tables and create a schema for each. I then implemented an SQL schema for creating the tables and adding the insert statements for each. I wrote and implemented the third query which was the *update* statement. George helped me refine this query. I was consistently at group meetings contributing ways to improve and progress with the project, as well as bringing ways to cope with possible changes to the requirements.
George Jenkinson	I created my own ERD with some inclusion of constraints and assumptions which we incorporated into our final ERD. I created and populated one fifth of the tables in the schema, and assisted in reviewing the completed schema when we brought all of our work together. I created query 2 and assisted in creating the other queries where my team mates were having difficulty. I also contributed ideas to what these query's should be. I contributed ideas as to how we could implement the changes to the database that we decided on. I attended each team meeting and contributed well to each of them.
Adam McNeill	I made an ERD based on the specification and integrated it with the rest of the team's ideas, which resulted in the completion of our collaborative ERD. I then completed a fifth of the schema, including the create and insert statements. I created the first query with assistance from George, and helped the team generate the ideas for the queries during a team meeting. I attended and contributed to each team meeting actively.
Josh Beatty	I created my own ERD version and combined sections of it and ideas to the final version. I converted some of the entities to the schema form and created the relevant create schema and insert statements for test data. Constructed Query 4 with assistance from George. Wrote up sections of the report relevant to the work I completed or had input in. Attended group meetings with high attendance and revised schema affected by ER, relationship or constraint changes.