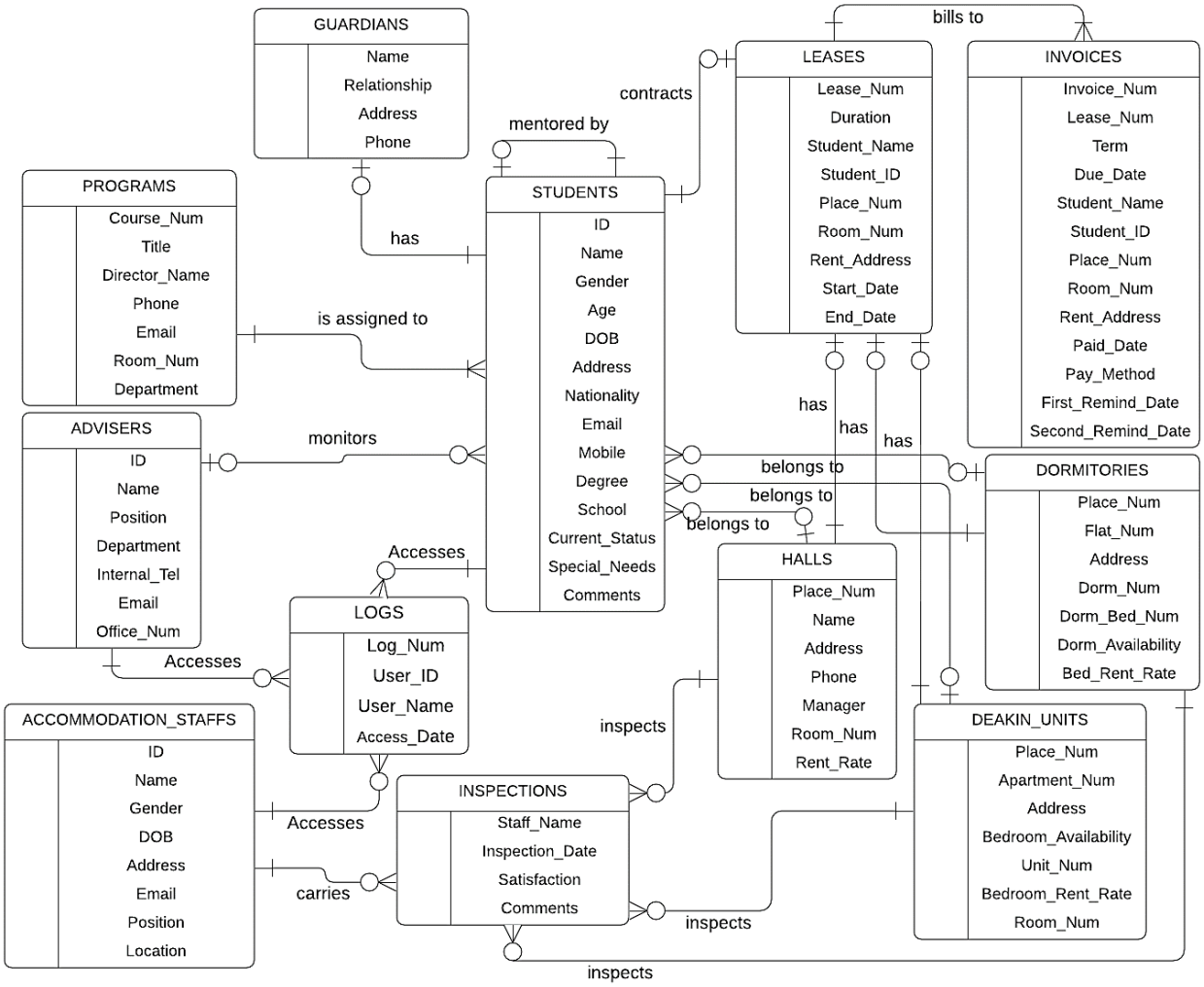
SQL Database Design and Information Retrieval

Part A:

Question A-1: Define the entities, attributes and draw ERD

Each table in the below diagram represents an entity whose name is indicated on the top row of the table. The attributes of each entity are listed in the right side of each table. All entities are connected by their relationships which indicated by the lines. The labels on those lines tell the name of the relationships while the symbols at either end of the lines show the type of relationships.

ERD diagram before normalization:



Question A-2: List the Functional and Transitive dependencies in each relation

Some entities contain functional and transitive dependencies which require reconstruction during normalization processes. Table below contains the functional and transitive dependencies of each entity:

Entity	Functional Dependency	Transitive Dependency
STUDENTS	ID → Name, Gender, Age, DOB, Address, Nationality, Email, Mobile, Degree, School, Current_Status, Special_Needs, Comments	Current_Status → Special_Needs, Comments
GUARDIANS	Name → Relationship, Address, Phone	

PROGRAMS	Course_Num → Title, Director, Phone, Email, Room_Num, Department	Director → Phone, Email
ADVISERS	ID → Name, Position, Department, Internal_Tel, Email, Office_Num	
ACCOMMODATION _STAFFS	ID → Name, Gender, DOB, Address, Email, Position, Location	
INSPECTIONS	Staff_Name, Inspection_Date → Satisfaction, Comments	
HALLS	Place_Num → Name, Address, Phone, Manager, Room_Num, Rent_Rate	Name, Address → Phone, Manager
DEAKIN_UNITS	Place_Num → Apartment_num, Address, Bedroom_Availability, Unit_Num, Bedroom_Rent_Rate, Room_Num	Apartment_Num → Address, Bedroom_Availability Unit_Num → Room_Num, Bedroom_Rent_Rate,
DORMITORIES	Place_Num → Flat_Num, Address, Dorm_Availability, Dorm_Num, Dorm_Bed_Num, Bed_Rent_Rate	Flat_Num → Address, Dorm_Availability
INVOICES	Invoice_Num → Lease_Num, Term, Due_Date, Student_Name, Student_ID, Place_Num, Room_Num, Rent_Address, Paid_Date, Pay_Method, First_Remind_Data, Second_Remind_Date	Lease_Num → Place_Num, Room_Num, Rent_Address Place_Num → Room_Num, Rental_Address Student_ID → Student_Name
LEASES	Lease_Num → Duration, Student_Name, Student_ID, Place_Num, Room_Num, Rent_Address, Start_Data, End_Date	Student_ID → Student_Name Place_Num → Room_Num, Rent_Addresss
LOGS	Log_Num → User_ID, User_Name, Access_Date	User_ID → User_Name

Question A-3 Normalize each relation to 3NF

1st Normalization Form:

Remove repeated or redundant data and anomalies by splitting the entity:

Entity Name	Full Attributes	New Entity Name	Split Attributes
STUDENTS	ID → Name, Gender, Age, DOB, Address, Nationality, Email, Mobile, Degree, School, Current_Status, Special_Needs, Comments	**STUDENTS**	ID, Name, Gender, Age, DOB, Address, Nationality, Email, Mobile, Degree, School
		Q_LIST	Current_Status, Special_Needs, Comments
PROGRAMS	Course_Num, Title, Director_Name, Phone, Email, Room_Num, Department	**PROGRAMS**	Course_Num, Title, Room_Num, Department, Director
		DIRECTORS	Director_Name, Phone, Email
HALLS	Place_Num, Name, Address, Phone, Manager, Room_Num, Rent_Rate	**HALLS**	Name, Address, Phone, Manager, Place_Num
		HALL_ROOMS	Place_Num, Room_Num, Rent_Rate, Name, Address
DEAKIN_UNITS	Place_Num, Apartment_num, Address, Bedroom_Availability, Unit_Num, Bedroom_Rent_Rate, Room_Num	**DEAKIN_UNITS**	Apartment_Num, Address, Bedroom_Availability

		DEAKIN_UNIT_ROOMS	Place_Num, Unit_Num, Room_Num, Bedroom_Rent_Rate, Apartment_Num
DORMITORIES	Place_Num, Flat_Num, Address, Dorm_Availability, Dorm_Num, Dorm_Bed_Num, Bed_Rent_Rate	**DORMITORIES**	Flat_Num, Address, Dorm_Availability
		DORMITORY_ROOMS	Place_Num, Dorm_Num, Dorm_Bed_Num, Bed_Rent_Rate, Flat_Num
INVOICES	Invoice_Num, Lease_Num, Term, Due_Date, Student_Name, Student_ID, Place_Num, Room_Num, Rent_Address, Paid_Date, Pay_Method, First_Remind_Date, Second_Remind_Date	**INVOICES**	Invoice_Num, Term, Due_Date, Paid_Date, Pay_Method, First_Remind_Date, Second_Remind_Date, Lease_Num, Place_Num, Sudent_ID
		INVOICE_LOCATIONS	Place_Num, Room_Num, Rental_Address
		INVOICE_RECIPIENTS	Student_ID, Student_Name
LEASES	Lease_Num, Duration, Student_Name, Student_ID, Place_Num, Room_Num, Rent_Address, Start_Data, End_Date	**LEASES**	Lease_Num, Duration, Start_Data, End_Date, Place_Num, Student_ID
		LEASE_LOCATIONS	Place_Num, Room_Num, Rent_Address
		LEASSEES	Student_ID, Student_Name
LOGS	Log_Num, User_ID, User_Name, Date	**LOGS**	Log_Num, User_ID, Access_Date
		USERS	User_ID, User_Name

NOTE: STUDENTS: [Age] is the repetition of [DOB], so [Age] is removed.

2nd Normalization Form:

Identify the primary keys and foreign keys of all entities:

Some entities may not have their unique primary keys or foreign keys yet. In these cases, artificial keys will be added to form those keys in order to be able to build the relationships between entities.

Entity Names	**Primary Keys** (other attributes) **Foreign Keys**
STUDENTS	**ID** (Name, Gender, DOB, Address, Nationality, Email, Mobile, Degree, School) **Mentor_ID, Adviser_ID, Course_Num, Place_Num**
Q_LIST	**Q_Num** (Current_Status, Special_Needs, Comments) **Student_ID**
GUARDIANS	**ID** (Name, Relationship, Address, Phone) **Student_ID**
PROGRAMS	**Course_Num** (Title, Room_Num, Department) **Director_ID**
DIRECTORS	**ID** (Director_Name, Phone, Email)
ADVISERS	**ID** (Name, Position, Email, Department, Internal_Tel, Office_Num)
ACCOMMODATION_STAFFS	**ID** (Name, Gender, DOB, Address, Email, Position, Location)
INSPECTIONS	**Inspection_Num** (Inspection_Date, Satisfaction, Comments) **Staff_ID, Place_Num**

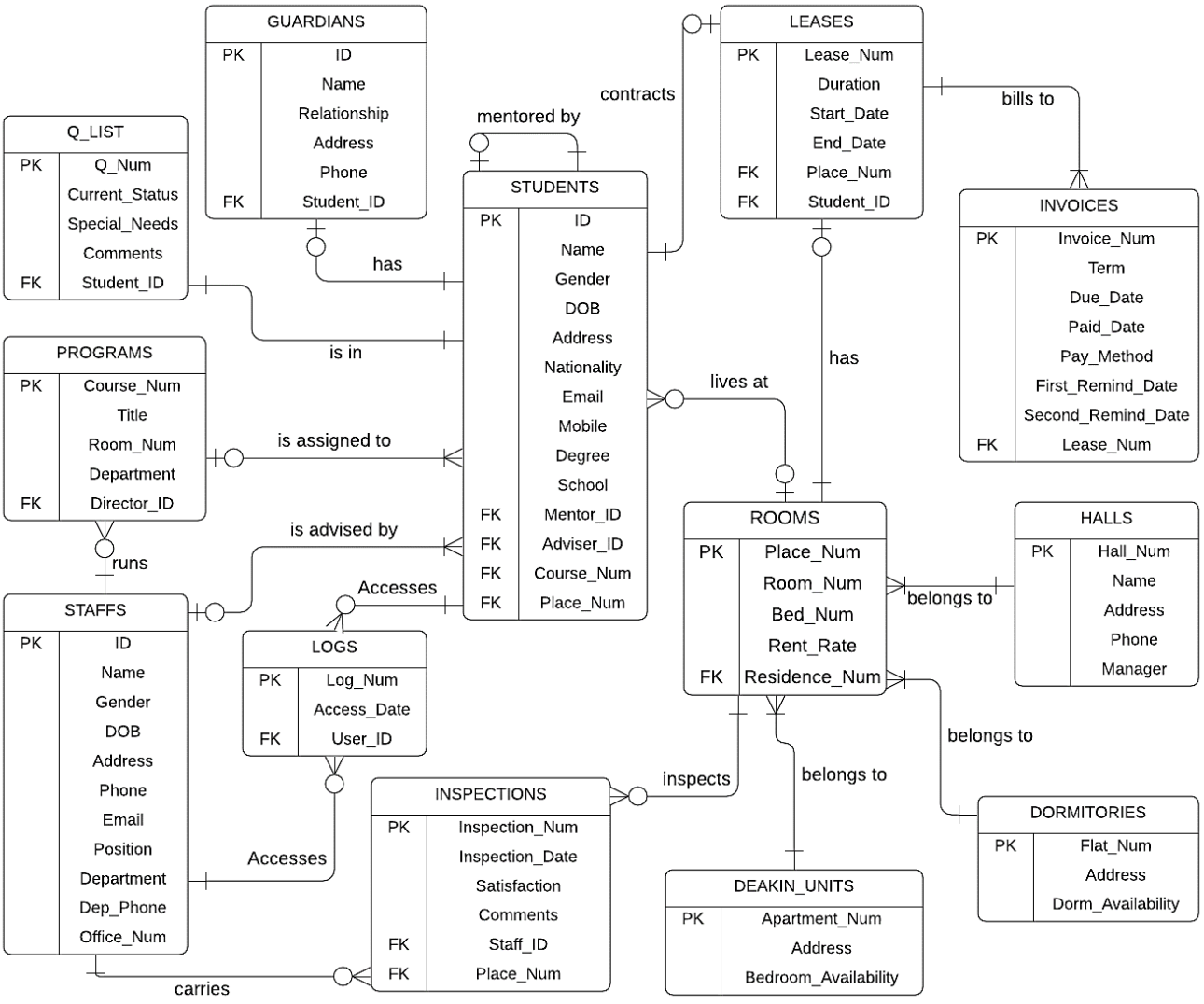
HALLS	**Hall_Num** (Name, Address, Phone, Manager)
HALL_ROOMS	**Place_Num** (Room_Num, Rent_Rate) **Hall_Num**
DEAKIN_UNITS	**Apartment_Num** (Address, Bedroom_Availability)
DEAKIN_UNIT_ROOMS	**Place_Num** (Room_Num, Bedroom_Rent_Rate) **Apartment_Num**
DORMITORIES	**Flat_Num** (Address, Dorm_Availability)
DORMITORY_ROOMS	**Place_Num (**Dorm_Num, Dorm_Bed_Num, Bed_Rent_Rate) **Flat_Num**
INVOICES	**Invoice_Num** (Term, Due_Date, Paid_Date, Pay_Method, First_Remind_Date, Second_Remind_Date) **Lease_Num, Place_Num, Sudent_ID**
INVOICE_LOCATIONS	**Place_Num (**Room_Num, Rental_Address)
INVOICE_RECIPIENTS	**Student_ID** (Student_Name)
LEASES	**Lease_Num** (Duration, Start_Data, End_Date) **Place_Num, Student_ID**
LEASE_LOCATIONS	**Place_Num** (Room_Num, Rent_Address)
LEASSEES	**Student_ID** (Student_Name)
LOGS	**Log_Num** (Access_Date) **User_ID**
USERS	**User_ID** (User_Name)

3rd Normalization Form:

At this point, all entities have been developed without having transitive dependency problems because we have solved it at A-2. Now we can merge the entities whose attributes are inherited from same family to reduce the unnecessary variation of entities.

HALL_ROOMS:	(**Place_Num**, Room_Num, Rent_Rate, **Hall_Num**)
DEAKIN_UNIT_ROOMS:	(**Place_Num**, Room_Num, Bedroom_Rent_Rate, **Apartment_Num**)
DORMITORY_ROOMS:	(**Place_Num**, Dorm_Num, Dorm_Bed_Num, Bed_Rent_Rate, **Flat_Num**)
These entities are merged to the parent ROOMS: (**Place_Num**, Room_Num, Bed_Num, Rent_Rate, **Residence_Num**)	
INVOICE_LOCATIONS:	(**Place_Num,** Room_Num, Rental_Address)
LEASE_LOCATIONS**:**	(**Place_Num,** Room_Num, Rent_Address)
These entities are merged to the parent ROOMS	
INVOICE_RECIPIENTS:	(**Student_ID,** Student_Name)
LEASSEES:	(**Student_ID,** Student_Name)
These entities are merged to the parent **STUDENTS**	
DIRECTORS:	(**ID,** Director_Name, Phone, Email)
ADVISERS:	(**ID,** Name, Position, Email, Department, Internal_Tel, Office_Num)
ACCOMMODATION_STAFFS:	(**ID,** Name, Gender, DOB, Address, Email, Position, Location)
These entities are merged to the parent **STAFFS**: (**ID**, Name, Gender, DOB, Address, Phone, Email, Position, Department, Dep_Phone, Office_Num)	
USERS:	**User_ID** (User_Name)
This entity is merged to the parent **STUDENTS** and **STAFFS**	

Final ERD diagram after normalization:



Question A-4: Create the database using SQL queries and submit database.sql

(database.sql is at the assignment folder)

Question A-5: Insert the data (6 records in each table) into "database" and submit mydata.sql

(mydata.sql is at the assignment folder)

Question A-6: Complete the following queries and provide query and its output.

a) Present a report listing the Manager's name and telephone number for each hall of residence
NOTE: One hall has multiple rooms, so we have inserted only two halls in [mydata.sql]

Query:

```
select manager, phone from halls;
```

Result: (2 records)

MANAGER	PHONE
Paul A.	0838 293 838
Dianel E.	0848 233 838

b) Present a report listing the names and student id with the details of their lease agreements.

Query:

```
select id, name, lease_num, duration, start_date, end_date, leases.place_num, room_num,
residence_address from students, leases,
        (select place_num as PLACE, room_num, address as residence_address from rooms
inner join
        (select hall_num as address_num, address from halls
        union
        select apartment_num as address_num, address from deakin_units
        union
        select flat_num as address_num, address from dormitories)
on residence_num = address_num)
where students.id = leases.student_id and leases.place_num = PLACE
order by id;
```

Result: (10 records)

ID	NAME	LEASE_NUM	DURATION	START_DATE	END_DATE	PLACE_NUM	ROOM_NUM	RESIDENCE_ADDRESS
1	Marry B.	5	3	10-Jul-20	10-Jul-21	2	3	Boxhill Melbourne 2273
2	Jack L.	4	2	10-Jul-20	10-Feb-21	1	1	Boxhill Melbourne 2273
3	Morgan K.	6	3	05-Mar-20	10-Feb-21	3	7	Boxhill Melbourne 2273
4	Denny M.	10	1	10-Nov-20	10-Feb-21	12	4	Blackburn Melbourne 2272
6	Rot B.	9	3	01-Jan-20	01-Jan-21	9	2	Burwood Melbourne 2271
7	Mike B.	7	2	20-Feb-20	20-Nov-20	7	1	Burwood Melbourne 2271
8	David S.	3	1	10-Jul-20	10-Nov-20	6	2	Mountainhill Melbourne 2275
9	Karl B.	1	3	10-Feb-20	10-Feb-21	5	8	Mountainhill Melbourne 2275
10	Mally .K	2	3	20-Feb-20	20-Feb-21	4	3	Mountainhill Melbourne 2275
11	Daren P.	11	3	02-Jun-20	02-Jun-21	10	4	Blackburn Melbourne 2272

c) Display the details of lease agreements that include the summer semester

NOTE: Leases.duration is recorded as number of semesters (1, 2, or 3). So, duration = 3 means 1 Year period which includes summer semester. A lease that starts from October or November is assumed to include summer semester. Further, a lease that ends at January or February is also assumed to include summer semester.

Query:

```
Select * from leases
Where duration > 2 or
Extract (month from start_date) in (10, 11) or
Extract (month from end_date) in (1, 2);
```

Result: (8 records)

LEASE_NUM	DURATION	START_DATE	END_DATE	PLACE_NUM	STUDENT_ID
1	3	10-Feb-20	10-Feb-21	5	9
2	3	20-Feb-20	20-Feb-21	4	10
4	2	10-Jul-20	10-Feb-2021	1	2
5	3	10-Jul-20	10-Jul-21	2	1
6	3	05-Mar-20	05-Mar-21	3	3
8	1	02-Nov-20	02-Feb-21	9	6
9	1	10-Nov-20	10-Feb-21	12	4
10	2	02-Nov-20	02-Jun-21	10	11

d) Display the details of the total rent paid by a given student.
Lets' assume the student we want to find has ID = 8
Query:

```
Select id, name, lease_num, room_num, payment
From students, rooms, leases,
        (Select id as person_id, sum((End_date - start_date) * rooms.rent_rate) as payment
        from leases, rooms, students, invoices
        where   leases.place_num = rooms.place_num and
                leases.student_id = students.id and
                leases.lease_num = invoices.lease_num and
                students.id = 8 and
                invoices.paid_date is not null
        group by id)
where   leases.student_id = person_id and
        leases.place_num = rooms.place_num and
        students.id = person_id;
```

Result: (1 records)

ID	NAME	LEASE_NUM	ROOM_NUM	PAYMENT
8	David S.	3	2	6851.1

e) Present a report on students who have not paid their invoices by a given date.
Lets' assume the given date we want to find = 01-JUN-2020
Query:

```
Select id, name from
students, leases,
        (select lease_num as lease_code from invoices
        where paid_date > '01-JUN-2020' or
        paid_date is null)
where leases.lease_num = lease_code and
        leases.student_id = id;
```

Result: (4 records)

ID	NAME
1	Marry B.
2	Jack L.
8	David S.
11	Daren P.

f) Display the details of apartment inspections where the property was found to be in an unsatisfactory condition
Query:

```
Select inspection_num, name as staff_name, inspection_date, place_num as property_num,
satisfaction, comments
From inspections, staffs
Where satisfaction = 'N' and
        staff_id = staffs.id;
```

Result: (2 records)

INSPECTION_NUM	STAFF_NAME	INSPECTION_DATE	PROPERTY_NUM	SATISFACTION	COMMENTS
3	John B.	10-Mar-20	6	N	Wholes on walls
6	Danel D.	10-Jun-20	2	N	Broken furniture

g) Present a report of the names and ID of students with their room number and place number in a particular hall of residence

Lets' assume, the hall we want to find is hall 2 (Hall_num = 2)
Query:

```
Select id, students.name as name, room_num, rooms.place_num as place_num, hall_num
From students, rooms, halls
Where  students.place_num = rooms.place_num and
        Rooms.residence_num = halls.hall_num and
        Hall_num = 2;
```

Result: (3 records)

ID	NAME	ROOM	PLACE_NUM	HALL_NUM
5	Jen B.	2	1	2
6	Rot B.	3	4	2
9	Karl B.	8	10	2

h) Present a report listing the details of all students currently on the waiting list for accommodation; that is; who were not placed.
Query:

```
Select id, name, gender, DOB, address, nationality, email, mobile, degree, school
From students inner join q_list
on current_status = 'Waiting' and
id = student_ID;
```

Result: (2 records)

ID	NAME	GENDER	DOB	ADDRESS	NATIOANLITY	MOBILE	EMAIL	DEGREE	SCHOOL
5	Jen B.	F	21-Feb-04	Stone Melbourne 2266	Australian	0480 382 837	Jen@deakin.edu.au	Undergraduate	SIT
12	Cortney L.	F	12-Feb-95	Cody Melbourne 2273	American	0484 382 838	Kortney@deakin.edu.au	Postgraduate	SC

i) Display the total number of students in each Study level
Query:

```
Select degree, count(degree) as total
From students
Group by degree;
```

Result: (2 records)

DEGREE	TOTAL
Undergraduate	7
Postgraduate	5

j) Present a report of the names and ID numbers for all students who have not supplied details of their Guardian.

Query:

```
Select students.id as id, students.name as name
From students
Where students.id not in (select student_id from guardians)
Order by id;
```

Result:

ID	NAME
1	Marry B.

4	Denny M.
5	Jen B.
6	Rot B.
9	Karl B.
10	Mally K.

k) Display the name and internal telephone number of the Adviser for a particular student.

Lets' assume, the adviser who we want to find is in charge of student with id = 4
Query:

```
Select staffs.name as adviser_name, dep_phone as internal_phone
From students, staffs
Where  students.adviser_id = staffs.id and
       Students.id = 4;
```

Result: (1 record)

ADVISER_NAME	INTERNAL_PHONE
Jacky D.	1223

l) Display the minimum, maximum, and average monthly rent for rooms in residence halls.
Query:

```
Select min(rent_rate)*30 as minimum, max(rent_rate)*30 as maximum,
avg(rent_rate)*30 as average
From rooms, halls
Where rooms.residence_num = hall_num;
```

Result:

MINIMUM	MAXIMUM	AVERAGE
1458	1713	1600

m) Display the total number of places in each residence hall.
Query:

```
Select hall_num, count(residence_num) as total_places
From rooms, halls
Where rooms.residence_num = halls.hall_num
Group by hall_num;
```

Result: (2 records)

HALL_NUM	TOTAL_PLACES
1	3
2	3

n) Display the staff number, name, age, and current location of all members of the residence staff who are over 60 years old today.

Query:

```
Select id, name, age, current_location
From
        (Select id, name, (extract(year from sysdate) - extract(year from dob)) as age,
        department as current_location, position from staffs)
Where  age > 60 and
        Position = 'Accommodation staff';
```

Result:

ID	NAME	AGE	CURRENT_LOCATION
301	John B.	63	Residence Hall
303	Danel D.	69	Dormitory

o) List each student and his mentor who lives in either Victoria Hall or DeakinUnit.

Query:

```
Select name, mentor_name
From students, (select id as mentor_code, name as mentor_name, place_num as
mentor_place_num from students)
Where  students.mentor_id = mentor_code
and
        id not in (Select student_id from q_list where current_status = 'Waiting')
and
        mentor_code not in (Select student_id from q_list where current_status = 'Waiting')
and
        Students.place_num in (select place_num as student_residence from rooms
        where   rooms.residence_num not in (select flat_num from dormitories))
and
        Mentor_place_num in (select place_num as student_residence from rooms
        where   rooms.residence_num not in (select flat_num from dormitories));
```

Result:

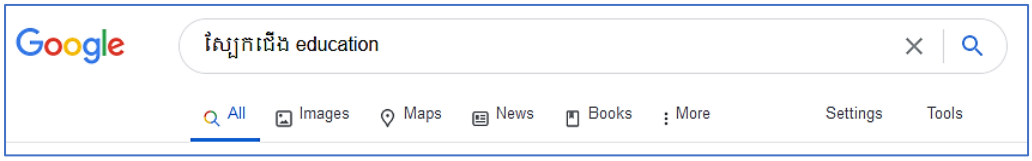
NAME	MENTOR_NAME
Jack L.	David S.

Part B:

Question B-1:

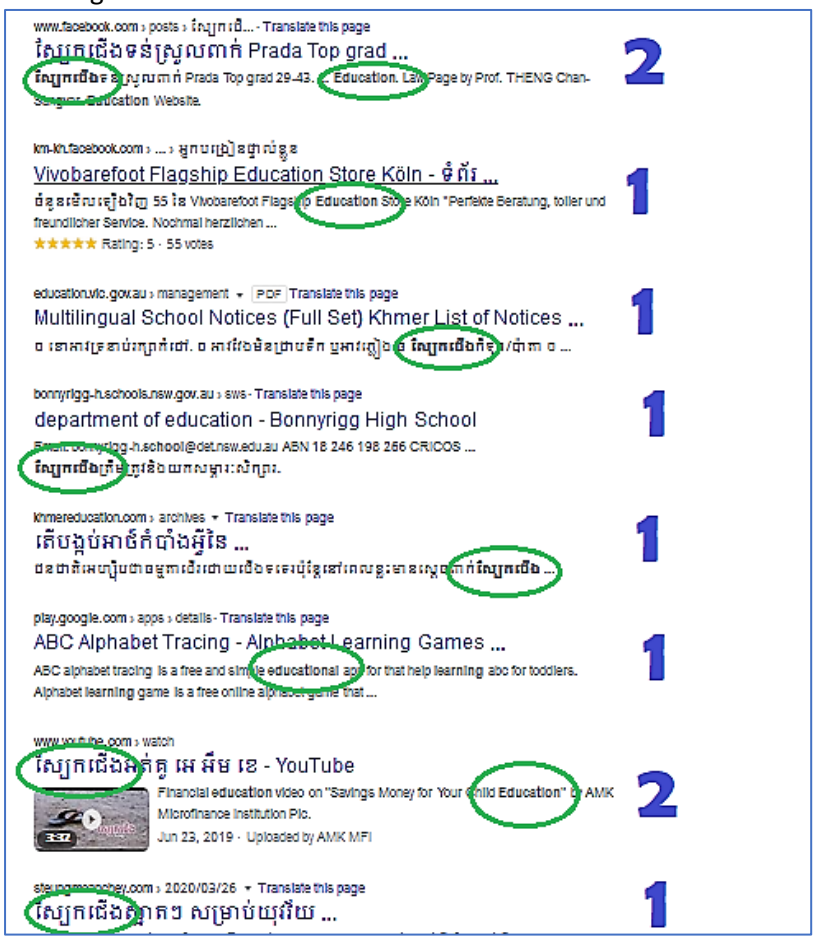
I could not find English terms for this experiment, so I decided to use Khmer term + English term to test this concept. So, I used “ស្បែកជើង” (means shoes) for term 1 and “Education” for term 2. I try to use the terms that do not go along, so to expect less search results.

Here is the screen shot of the two terms:



(i) Take screenshot of the first page of Google results (or more if you want to) and mark each result with 2 (both terms occur on the page), 1 (one term occurs on the page) or 0 (neither term occurs on the page)

First Page:



Last page:

(ii) Based on this evidence, does Google interpret all queries as a Boolean conjunction?
Analyzing the results, google tries to fit the best match at the first research result. But it seems that from the second to the sixth result, google only gives weight to one only term. Interestingly, on the seventh result, it provides a mixture of the two-terms result again. The last few results come without matching the two terms at all.

My personal conclusion is Google does not use Boolean method to evaluation my query because many of the results favor only one term over the other which is not the characteristics of Boolean method. Google even sends back some results without matching any terms but the meaning of the terms. Boolean method would always balance the existence of the two terms and discard any page results that do not contains the queried terms. Boolean method determines the clear cut whether the terms exist and sends only relevant pages that meet the requirement. Therefore, Google must have used other technique but not a pure Boolean method.

Question B-2: (6)

(a) Compute Recall, Precision and precision@5:
Accuracy and Recall table for Q1:

Q1	No	1	2	3	4	5	6	7	8	9	10	11
	Precision	0.0000	0.0000	0.3333	0.2500	0.4000	0.3333	0.2857	0.3750	0.3333	0.3000	0.2727
	Recall	0.0000	0.0000	0.1111	0.1111	0.2222	0.2222	0.2222	0.3333	0.3333	0.3333	0.3333
	Status			R		R			R			

The precision@5 is 0.4

Accuracy and Recall table for Q2:

Q2	No	1	2	3	4	5	6	7	8	9	10
	Precision	1.0000	0.5000	0.3333	0.5000	0.6000	0.5000	0.5714	0.6250	0.5556	0.5000
	Recall	0.1250	0.1250	0.1250	0.2500	0.3750	0.3750	0.5000	0.6250	0.6250	0.6250
	Status	R			R	R		R	R		

The precision@5 is 0.6

Accuracy and Recall table for Q3:

Q3	No	1	2	3	4
	Precision	1.0000	0.5000	0.6667	0.7500
	Recall	0.0833	0.0833	0.1667	0.2500
	Status	R		R	R

Since there is no document 5 retrieved, so it is considered as irrelevant retrieval.
Then the precision@5 is 3/5 = 0.6

(b)

- Provide and Justify two information-seeking tasks where precision may be considerably more important than recall:

  There situations that precision becomes more important than recall measurement. Such case of searching for a definition of a word, the user only looks for precision of the results. Only a few relevant results on the top of the result list would be very satisfying. Any other case is when a user wants to find a factual information of an object such as a person, thing, or place. In the two situations above, the information sought is general and there are normally many of relevant documents that contains similar information the user is looking for. Therefore, receiving a few of them or many of them won't make any much difference, and thus precision is prioritized.

- Provide and Justify two information-seeking tasks where recall may be more important than precision:

Recall may be more important when it comes to locating a research idea such as finding whether a research topic has been covered by other researchers. A few relevant results will not stop the user from continuing the exploration of other documents. An exhaustive search is required to ensure all relevant documents have been discovered. Another case also prioritizes recall when we are searching for a how-to method or remedy of a problem. In this case, there are possibly many different ways to solve the problem which make the variation of relevant results critical. We simply try to look through all possible ways, even the information is widespread across many different documents. Therefore, in the two cases above, the recall of the search result becomes more important than precision because the user needs all the relevant documents.

(c) How the search engine tries to guess without asking, whether user cares more about precision than recall, or vice versa?

The only way the search engine can guess whether a user prefer precision to recall or vice versa is through the amount of the relevant documents. This can be related to the number of query terms input by user. Normally the greater number of query terms is, the fewer relevant results will be, and this effect is opposite when the number of query terms is less. For instance, when query terms are very few (1 or 2), it can be interpreted as a general search. There be large amount relevant documents that match with the search terms and their combination. This large number of relevant documents makes it impossible for the user to locate all of them. In this sense, the system must focus on the precision in order to optimize its support.

In the opposite way, the system will emphasize on recall when there are large numbers of query terms. Relating it to the question (b) above, if a user searches for a research topic, he/she will produce at least four to five key terms in order to limit the search boundary. The combination of the query terms will make the query more specific. As the result, the number of relevant results become relatively small. In this situation, system knows that user is likely to be able to check through every relevant result, and thus it weighs more to the recall measurement.

Question B-3:
Which document(s) (if any) match each of the following queries where each expression within quotes is a phrase query? At which positions do the queries match?

(i) “fools rush in”
It matches with document 3, at position (13, 14, 15).

(ii) “Devil is fool” AND “you are angel”
No match found!

(iii) “Where are you angle”.
No match found!

(iv) There is something wrong with this positional index. What is the problem?
There are index numbers of different terms that have the same numbers. Below are the findings:
- All index numbers of term “in” and “is” are the same for document 1, 2 and 3.
- Term “are” and “You” in document 2 have the same index <13>.
- Term “where” and “Rush” in document 3 have the same index <14>.
- Term “Angel” and “in” in document 4 have the same index <17>, “are” and “where” have the same index <98>.

There is a character “h” in document 2, index number <14> of term ”where”.

Question B-4:

(i) Compute the document scores and the ranking associated with the query (houses OR for OR sale OR in OR Geelong OR Melbourne)

Calculate the IDF of the terms

house	for	sale	in	Geelong	Melbourne
0.00	0.00	0.00	0.00	0.42	0.00

Calculate the TF-IDF of each document:

	house	for	sale	in	Geelong	Melbourne
Doc 1	0.00	0.00	0.00	0.00	9.13	0.00
Doc 2	0.00	0.00	0.00	0.00	0.00	0.00
Doc 3	0.00	0.00	0.00	0.00	8.72	0.00
Doc 4	0.00	0.00	0.00	0.00	1.25	0.00

Since every logical operation is OR, the term evidence (TE) takes the max value of TF-IDF. Therefore, the rank is:

	Scores	Rank
Doc 1	9.13	1st
Doc 2	0.00	4th
Doc 3	8.72	2nd
Doc 4	1.25	3rd

(ii) How is the ranking produced probably sub-optimal and why does this happen?

There are so many TF-IDF scores have values of zeros. This flat scores of many terms turns most of the document scores to identical values. As the result, it makes Boolean ranking method becomes sub-optimal because the ranking depends only on a few available scores. In our case, only feature "Geelong" of the vector holds the scores for this ranking. Additionally, Doc 2 loses its score values completely.

(iii) Compute the document scores and the ranking associated with the query (houses AND for AND sale AND in AND Geelong OR Melbourne)

Due to logical operator precedence, AND operation must be computed before OR. So, we have:

(houses AND for AND sale AND in AND Geelong)
For logical AND, TE takes the min of TF-IDF.
Result:

	Scores
Doc 1	0.00
Doc 2	0.00
Doc 3	0.00
Doc 4	0.00

OR

Melbourne

	Scores
Doc 1	0.00
Doc 2	0.00
Doc 3	0.00
Doc 4	0.00

Based on the above TF-IDF, the final score and rank is:

	Scores	Rank
Doc 1	0.00	1st
Doc 2	0.00	1st
Doc 3	0.00	1st
Doc 4	0.00	1st

(iv) How is the ranking produced probably sub-optimal and why does this happen?

Large number of terms scores are zeros. It minimizes the identity scores of each document. Therefore, the ranking is not fully optimized. In our case, the query generates zero scores for every document. In this situation, every document will have the same score.

(v) How would you extend the Boolean retrieval model to handle AND NOT constraints (e.g., houses AND NOT Geelong)? Your proposed solution should give a higher score to documents that contain fewer occurrences of the term to the right of the AND NOT (e.g., Geelong). Please be as mathematical as possible. In other words, saying: "I would reduce the score for documents that contain the word to the right of AND NOT." is too vague.

The formula should discourage the high value of the term to the right, but instead promote the high value of the term to the left. Therefore, we propose a score formula as follow:

Score = [Term to the left] – [Term to the right]

The higher the score, the better rank of the document is.

To prove it, please see the simplified examples of cases below:

Term1 AND NOT Term 2		
Term 1	Term 2	Score = Term1 -Term2
10	10	0
10	9	1
10	8	2
10	7	3
10	6	4
10	5	5
10	4	6
10	3	7
10	2	8
10	1	9

Term1 AND NOT Term 2		
Term 1	Term 2	Score = Term1 - Term2
5	10	-5
5	9	-4
5	8	-3
5	7	-2
5	6	-1
5	5	0
5	4	1
5	3	2
5	2	3
5	1	4

Term1 AND NOT Term 2		
Term 1	Term 2	Score = Term1 - Term2
1	10	-9
1	9	-8
1	8	-7
1	7	-6
1	6	-5
1	5	-4
1	4	-3
1	3	-2
1	2	-1
1	1	0

These three tables give a thorough set of cases between term 1 and term 2. As we can see, this formula gives the higher score when term 1 holds the highest value (10) while term 2 holds the lowest value (1). Therefore, the document that contains the higher score (9) is rank on the top.

(vi) Using the index, what would be the Boolean retrieval model scores given to documents 1-4 by your proposed scoring method for the query "houses AND NOT Geelong"?

In this case, the score will look like:

	"house" AND NOT "Geelong"			
	"house"	"Geelong"	Score	Rank
Doc 1	0.00	9.13	-9.13	4th
Doc 2	0.00	0.00	0.00	1st
Doc 3	0.00	8.72	-8.72	3rd
Doc 4	0.00	1.25	-1.25	2nd

As seen, Doc 2 will be the top, followed by Doc 4. The 3rd goes to Doc 3 and finally the lowest rank goes to Doc 1.

Question B-5:

(i) Explain the similarity scores of both Q1 and Q2.

Generate the frequency table:

- Remove the stop word dictionary and grammatical variations

Q 1	love	book	good			
Q2	read	good	book	make	feel	better

Doc 1	book	consider	good	book	make	reader	feel	good	book
Doc 2	love	read	good	book	feel	better			
Doc 3	one	can	feel	better	after	read	Tom	recent	book

- Merge the terms, count and sort them:

	after	better	book	can	consider	feel	good	love	make	one	read	reader	recent	Tom
Q 1	0	0	1	0	0	0	1	1	0	0	0	0	0	0
Q2	0	1	1	0	0	1	1	0	1	0	1	0	0	0

	after	better	book	can	consider	feel	good	love	make	one	read	reader	recent	Tom
Doc 1	0	0	3	0	1	1	2	0	1	0	0	1	0	0
Doc 2	0	1	1	0	0	1	1	1	0	0	1	0	0	0
Doc 3	1	1	1	1	0	1	0	0	0	1	1	0	1	1

Get IDF of the terms:

after	better	book	can	consider	feel	good	love	make	one	read	reader	recent	Tom
1.58	0.58	0.00	1.58	1.58	0.00	0.58	1.58	1.58	1.58	0.58	1.58	1.58	1.58

Get IDF of the documents

	after	better	book	can	consider	feel	good	love	make	one	read	reader	recent	Tom
Doc1	0.00	0.00	0.00	0.00	1.58	0.00	1.17	0.00	1.58	0.00	0.00	1.58	0.00	0.00
Doc2	0.00	0.58	0.00	0.00	0.00	0.00	0.58	1.58	0.00	0.00	0.58	0.00	0.00	0.00
Doc3	1.58	0.58	0.00	1.58	0.00	0.00	0.00	0.00	0.00	1.58	0.58	0.00	1.58	1.58

By using Cosine similarity score, we get the final scores:

	Q1	Q2
Doc1	0.2263	0.3769
Doc2	0.6660	0.3809
Doc3	0.0000	0.1312

Based on the similarity score table, both Q1 and Q2 will produce the same result. The search result will place Doc 2 on the top, followed by Doc 1, and lastly Doc 3.

We also see the difference between Q1 and Q2 scores. Q1 score tents to be more extreme than Q2. While Q1 weighs high score to Doc 1, it does not give any score to Doc 3. Distinctively, Q2 seems to balance all documents with moderate scores.

(ii) How would the result changes using TF-IDF instead of TF as Query?

Get TF-IDF scores of the two queries:

	after	better	book	can	consider	feel	good	love	make	one	read	reader	recent	Tom
Q1	0.00	0.00	0.00	0.00	0.00	0.00	0.58	1.58	0.00	0.00	0.00	0.00	0.00	0.00
Q2	0.00	0.58	0.00	0.00	0.00	0.00	0.58	0.00	1.58	0.00	0.58	0.00	0.00	0.00

By using Cosine similarity score, we get final scores:

	Q1	Q2
Doc1	0.1357	0.5694
Doc2	0.8981	0.2901
Doc3	0.0000	0.1000

As we can see, Q1 still produces the same result as before, but Q2 changes the rank from the Doc 2 (being the top previously) to Doc 1. Doc 3 is still at the lowest rank. We also notice that the scores become more distinctive than before. The reason is TF-IDF has removed the weights of common terms, i.e. “feel” and “book” and given more score to the rare terms, i.e. “love” and “make”.

(iii) What do prefer using TF or TF-IDF as Query (Support your claim using F-score).

I personally prefer TF-IDF to TF. The reason is TF-IDF provides clear distinction between the level of relevance between documents. More importantly, it mitigates the effect of common words that exist in most documents. In return, it focuses more on important terms which leads to a more preferable result.

I count not get the F-score because there is no clear count of how many relevant documents.

The end