

# BT5110 Data Management and Warehousing

## Tutorial 3: Aggregate and Nested Queries

Mark Meng Huasong

School of Computing  
National University of Singapore

6 - 10 Sep, 2021



All the materials within presentation slides are protected by copyrights.  
It is forbidden by NUS to upload these materials to the Internet.

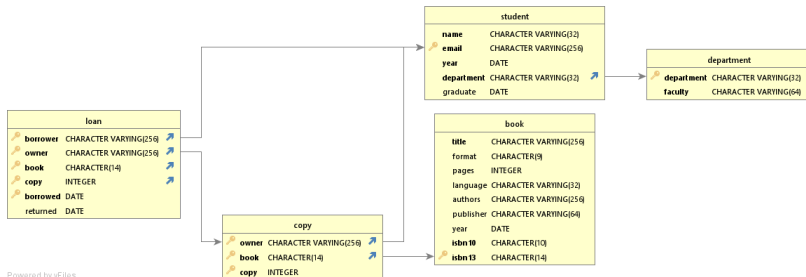
# Quick Recap: End of Last Tutorial

What we have done in the last week:

- (1) Write simple a query with arithmetic, comparison & logical operators;
- (2) Write a table joining query (CROSS JOIN and INNER JOIN);
- (4) Write simple a query with the set operators.

Summary of our database (# of rows):

book	student	copy	loan	department
311	105	1244	4976	13



Powered by yFiles

(plotted by DbVisualizer)

# Question 1 Aggregate Queries (a)

How many loans involve an owner and a borrower from the same department?

## Solution:

---

```
1  SELECT COUNT(*)
2  FROM loan l, student s1, student s2
3  WHERE l.owner = s1.email
4  AND l.borrower = s2.email
5  AND s1.department = s2.department;
```

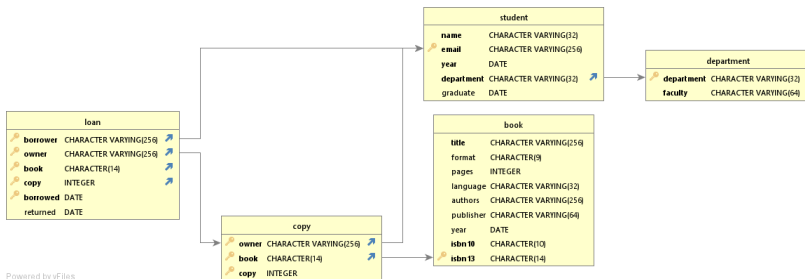
---

You should have **502** observed in the output.

# Question 1 (b)

For each faculty, print the number of loans that involve an owner and a borrower from this faculty?

Which tables are involved in this query?



Powered by yFiles

(plotted by DbVisualizer)

## Question 1 (b) Cont.

For each faculty, print the number of loans that involve an owner and a borrower from this faculty?

### Solution:

---

```
1  SELECT d1.faculty, COUNT(*)
2  FROM loan l, student s1, student s2, department d1, department d2
3  WHERE l.owner = s1.email
4         AND l.borrower = s2.email
5         AND s1.department = d1.department
6         AND s2.department = d2.department
7         AND d1.faculty = d2.faculty
8  GROUP by d1.faculty;
```

---

## Question 1 (b) Cont.

The output should be like as follows (for your reference).

faculty	count
Faculty of Arts and Social Science	379
Faculty of Engineering	82
Faculty of Science	239
School of Computing	529

(4 rows)

## Question 1 (c)

What are the average and the standard deviation of the duration of a loan? (assume today is 2010-12-31)

**Recap:** Q1 (d) of Tutorial 2: print the duration of each loan:

---

```
1  SELECT l.book, ((CASE WHEN l.returned ISNULL
2     THEN '2010-12-31'
3     ELSE l.returned END) - l.borrowed + 1) AS duration
4  FROM loan l
5  ORDER BY l.book ASC, l.duration ASC;
```

---

How can we reuse this code?

## Question 1 (c) Cont.

### Solution:

```
1 SELECT AVG((CASE WHEN l.returned ISNULL
2     THEN '2010-12-31'
3     ELSE l.returned END) - borrowed + 1),
4     STDDEV_POP((CASE WHEN l.returned ISNULL
5     THEN '2010-12-31'
6     ELSE l.returned END) - borrowed + 1)
7 FROM loan l;
```

The output should be as follows.

avg	stddev_pop
41.4963826366559486	38.4206806387009364

(1 row)

### Attention

STDDEV\_SAMP and STDDEV are equivalent, which calculate Sample Covariance. While STDDEV\_POP calculate Population Covariance. Here we need to use the latter.

**Extra:** Can we use nested query to simplify it?



## Question 1 (c) Cont.

### Solution (nested version):

---

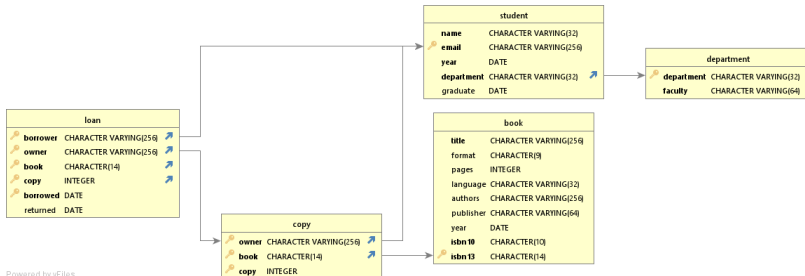
```
1  SELECT AVG(temp.duration), STDDEV_POP(temp.duration)
2  FROM (SELECT ((CASE
3           WHEN l.returned ISNULL
4           THEN '2010-12-31'
5           ELSE l.returned END) - l.borrowed + 1) AS duration
6  FROM loan l) AS temp;
```

---

## Question 2 Nested Queries (a)

Print the titles of different books that have never been borrowed. Use a nested query.

Which tables are involved in this nested query?



Powered by yFiles

(plotted by DbVisualizer)

## Question 2 (a) Cont.

### Solution:

---

```
1  SELECT b.title
2  FROM book b
3  WHERE b.ISBN13 NOT IN (
4      SELECT l.book
5      FROM loan l);
```

---

...or, equivalently,

---

```
1  SELECT b.title
2  FROM book b
3  WHERE b.ISBN13 <> ALL (
4      SELECT l.book
5      FROM loan l);
```

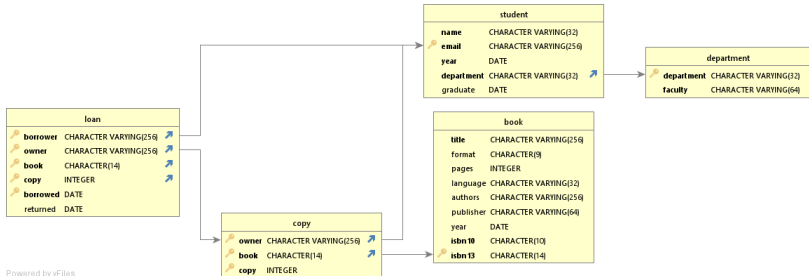
---

You should observe **no** record given in the output.

## Question 2 (b)

Print the name of the different students who own a copy of a book that they have never lent to anybody.

Which tables are involved in this nested query?



Powered by yFiles

(plotted by DbVisualizer)

## Question 2 (b) Cont.

### Solution:

---

```
1  SELECT s.name
2  FROM student s
3  WHERE s.email IN
4      (SELECT c.owner
5       FROM copy c
6       WHERE NOT EXISTS (
7           SELECT *
8           FROM loan l
9           WHERE l.owner = c.owner
10              AND l.book = c.book
11              AND l.copy = c.copy));
```

---

## Question 2 (b) Cont.

...or, equivalently,

---

```
1  SELECT s.name
2  FROM student s
3  WHERE s.email = ANY (
4      SELECT c.owner
5      FROM copy c
6      WHERE NOT EXISTS (
7          SELECT *
8          FROM loan l
9          WHERE l.owner = c.owner
10             AND l.book = c.book
11             AND l.copy = c.copy));
```

---

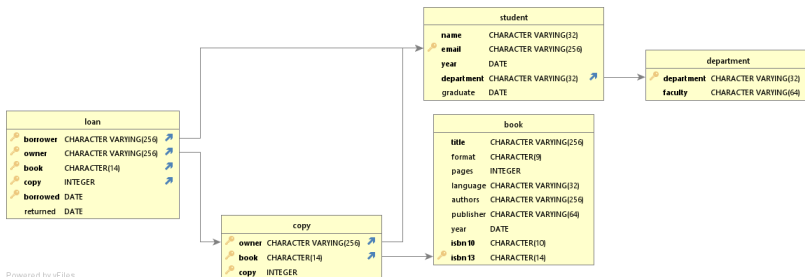
You should observe **no** record given in the output.

### Notice

We can always use “<> ALL” as an equivalence of “NOT IN”. Similarly, we can use “= ANY” as an equivalence of “IN”

## Question 2 (c)

For each department, print the names of the students who lent the most.



Powered by yFiles

(plotted by DbVisualizer)

## Question 2 (c) Cont.

### Solution:

---

```
1  SELECT  s.department, s.name, count(*)
2  FROM    student s, loan l
3  WHERE   l.owner = s.email
4  GROUP BY s.department, s.email
5  HAVING  count(*) >= ALL
6          (SELECT count(*)
7           FROM    student s1, loan l1
8           WHERE   l1.owner = s1.email
9           AND     s.department = s1.department
10          GROUP BY s1.email);
```

---

It is better to add "ORDER BY s.department" at the end for better output display, although it is not required by the question.



## Question 2 (c) Cont.

You should observe the output as follows.

department	name	count
Geography	SEAH TECK KEE	64
Geography	ZHANG YUZHAO	64
EE	WANG NA	60
CE	TAY YONG MING	68
Math	HUANG WENXIN	76
CS	LIU ZHENCAI	76
Geography	DAVID HALL	64
ME	GE DUO	64
Language	NEELAM DEOL	68
Biology	PENG JIAYUAN	80
IS	ZHANG HONG	88
History	ZENG YIHUI	60
Economics	LI YUZHAO	60
Physics	NI HANRAN	60
Chemistry	XIE XIN	56

(15 rows)

(without ORDER BY)

department	name	count
Biology	PENG JIAYUAN	80
CE	TAY YONG MING	68
Chemistry	XIE XIN	56
Computer Science	LIU ZHENCAI	76
EE	WANG NA	60
Economics	LI YUZHAO	60
Geography	SEAH TECK KEE	64
Geography	ZHANG YUZHAO	64
Geography	DAVID HALL	64
History	ZENG YIHUI	60
IS	ZHANG HONG	88
Language	NEELAM DEOL	68
ME	GE DUO	64
Math	HUANG WENXIN	76
Physics	NI HANRAN	60

(15 rows)

(with ORDER BY s.department)

### Warning

Notice that there are three such students in the department of **Geography** (that is why one should almost never use TOP N queries).

## Question 2 (c) Cont. - *Extra(1)*

**Extra(1):** What if I insert the query below before we execute the solution?

```
1 INSERT INTO department VALUES('Business_Analytics', 'School_of_Computing');
2 INSERT INTO student VALUES('MARK_MENG', 'mark@biz.edu', '2010-01-01', 'Business_Analytics', '2014-01-01');
```

We observe the output as:

department	name	count
Geography	SEAH TECK KEE	64
Geography	ZHANG YUZHAO	64
EE	WANG NA	60
CE	TAY YONG MING	68
Math	HUANG WENXIN	76
CS	LIU ZHENCAI	76
Geography	DAVID HALL	64
ME	GE DUO	64
Language	NEELAM DEOL	68
Biology	PENG JIAYUAN	80
IS	ZHANG HONG	88
History	ZENG YIHUI	60
Economics	LI YUZHAO	60
Physics	NI HANRAN	60
Chemistry	XIE XIN	56
(15 rows)		

We expect the output to be like:

department	name	count
Geography	SEAH TECK KEE	64
Geography	ZHANG YUZHAO	64
EE	WANG NA	60
CE	TAY YONG MING	68
Math	HUANG WENXIN	76
CS	LIU ZHENCAI	76
Geography	DAVID HALL	64
ME	GE DUO	64
Language	NEELAM DEOL	68
Biology	PENG JIAYUAN	80
IS	ZHANG HONG	88
History	ZENG YIHUI	60
Economics	LI YUZHAO	60
Physics	NI HANRAN	60
Chemistry	XIE XIN	56
Business Analytics	MARK MENG	0
(16 rows)		

Let's think about it... (out of scope of this tutorial)

## Question 2 (c) Cont. *Extra(1)*

To simplify this question, let's create a VIEW first:

---

```
1 CREATE VIEW lender AS
2 SELECT s.department, s.name, count(*) AS count
3 FROM student s, loan l
4 WHERE l.owner = s.email
5 GROUP BY s.department, s.email;
```

---

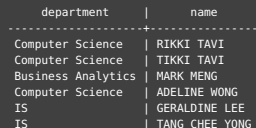
How to print those departments & students without any student lending a book?

---

```
1 SELECT s.department, s.name
2 FROM student s LEFT OUTER JOIN loan l
3 ON l.owner = s.email
4 WHERE l.borrower ISNULL;
```

---

The output will be like follows.



department	name
Computer Science	RIKKI TAVI
Computer Science	TIKKI TAVI
Business Analytics	MARK MENG
Computer Science	ADELIN WONG
IS	GERALDINE LEE
IS	TANG CHEE YONG

## Question 2 (c) Cont. *Extra(1)*

Let's create the view that includes those department without a lender?

---

```
1  DROP VIEW lender;
```

---

```
1  CREATE VIEW lender AS
2  (SELECT  s.department, s.name, count(*) AS count
3    FROM student s, loan l
4   WHERE l.owner = s.email
5   GROUP BY s.department, s.email)
6  UNION
7  (SELECT s.department, s.name, 0 AS count
8    FROM student s LEFT OUTER JOIN loan l
9     ON l.owner = s.email
10   WHERE l.borrower ISNULL);
```

---

## Question 2 (c) Cont. *Extra(1)*

Now let's recap the solution of Q2(c) (on the **left**), and convert it to the code on the **right**.

---

```
1 SELECT s.department, s.name, count(*)
2 FROM student s, loan l
3 WHERE l.owner = s.email
4 GROUP BY s.department, s.email
5 HAVING count(*) >= ALL
6     (SELECT count(*)
7      FROM student s1, loan l1
8      WHERE l1.owner = s1.email
9      AND s.department = s1.department
10     GROUP BY s1.email)
11 ORDER BY s.department;
```

---

---

```
1 SELECT l.department, l.name, l.count
2 FROM lender l
3 GROUP BY l.department, l.name, l.count
4 HAVING l.count >= ALL
5     (SELECT l2.count
6      FROM lender l2
7      WHERE l.department = l2.department)
8 ORDER BY l.department;
```

---

The output below is better, and is in fact the CORRECT one. (see next slide)

## Question 2 (c) *Extra(1)*

The output:

department	name	count
Biology	PENG JIAYUAN	80
Business Analytics	MARK MENG	0
CE	TAY YONG MING	68
Chemistry	XIE XIN	56
Computer Science	LIU ZHENCAI	76
EE	WANG NA	60
Economics	LI YUZHAO	60
Geography	DAVID HALL	64
Geography	SEAH TECK KEE	64
Geography	ZHANG YUZHAO	64
History	ZENG YIHUI	60
IS	ZHANG HONG	88
Language	NEELAM DEOL	68
ME	GE DUO	64
Math	HUANG WENXIN	76
Physics	NI HANRAN	60
(16 rows)		

### Notice

If you have run those code, don't forget to delete ME from student, as those record are not in the scope of our tutorials.

E.g., DELETE FROM student WHERE name='MARK MENG';  
DELETE FROM department WHERE department='Business Analytics';

## Question 2 (c) *Extra(2)*

**Extra(2):** What if the question asks to print the top **3** lenders in each department?

**Hint:** This time we may seek help from the troublesome “LEFT OUTER JOIN”.

## Question 2 (c) *Extra(2)*

### Solution of Extra(2):

---

```
1  SELECT winner.department, winner.name, winner.count
2  FROM lender winner LEFT OUTER JOIN lender loser
3      ON winner.department=loser.department
4      AND winner.name<>loser.name
5      AND winner.count<loser.count
6  GROUP BY winner.department, winner.name, winner.count
7  HAVING COUNT(loser.name) < 3
8  ORDER BY winner.department;
```

---



## Question 2 (c) *Extra(2)*

The output will be as follows.

department	name	count
Biology	CHOY YI TING	56
Biology	GOH HUI YING	72
Biology	PENG JIAYUAN	80
CE	ANUPAMA ANGHAN	64
CE	DING KUAN CHONG	60
CE	TAY YONG MING	68
Chemistry	LISA SMITH	40
Chemistry	LIU ZHANPENG	52
Chemistry	XIE XIN	56
Computer Science	ANG JIA YI	64
Computer Science	LIU ZHENCAI	76
Computer Science	SIOW CAO KHOA	72
EE	LIU JUN	48
EE	WANG NA	60
EE	ZHANG CONG	48
Economics	JERRY BROWN	56
Economics	LI YUZHAO	60
Economics	LIU LINXI	44
Geography	DAVID HALL	64
Geography	SEAH TECK KEE	64
Geography	ZHANG YUZHAO	64

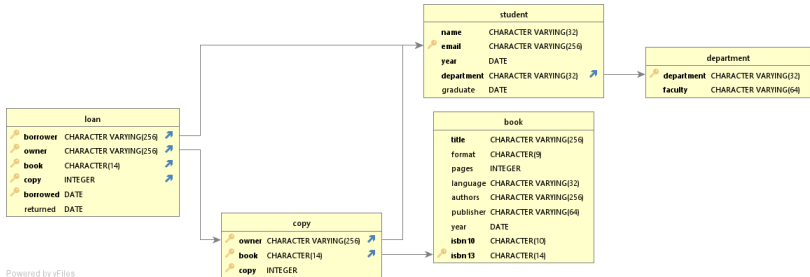
(Continue)		
Geography	ZHANG YUZHAO	64
History	DING YANG	56
History	ZENG YIHUI	60
History	ZHU CHANG	48
IS	HUANG QI	60
IS	SUBRAMANIAM GHANTASALA	76
IS	ZHANG HONG	88
Language	ANNIE CHAPMAN	60
Language	NEELAM DEOL	68
Language	NEHAL KANWAT	64
ME	DENNIS PALMER	32
ME	GE DUO	64
ME	LIU DANNI	32
ME	NG ANH QUANG	32
Math	HUANG WENXIN	76
Math	ZHANG ZHANPENG	72
Math	ZHANG ZHUO	52
Physics	GE YUWEI	52
Physics	NI HANRAN	60
Physics	TAN CHENG HAN	44
Physics	TSO HUI LIN	44
(41 rows)		

Nightmare ends here!

## Question 2 (d)

Print the emails and the names of the different students who borrowed all the books authored by Adam Smith.

Can we make use of negation with nested query?



Powered by yFiles

(plotted by DbVisualizer)

## Question 2 (d) Cont.

### Solution:

```
1  SELECT s.email, s.name
2  FROM student s
3  WHERE NOT EXISTS (
4      SELECT *
5      FROM book b
6      WHERE authors = 'Adam_Smith' AND NOT EXISTS (
7          SELECT *
8          FROM loan l
9          WHERE l.book = b.isbn13 AND l.borrower = s.email));
```

```
      email      |      name
-----+-----
yeojiahao1989@yahoo.com | YE0 JIA HAO
(1 row)
```

## Question 2 (d) Cont.

**Interpretation:** We are going to print the emails and the names of students who have borrowed all Adam's book.

Our query must start with:

```
SELECT s.email, s.name FROM student s...
```

Then how to use DOUBLE NEGATION to describe this student?

---

```
1  SELECT s.email, s.name
2  FROM student s  -- if the student has not missed borrowing any book of Adam, that
                   means the student has borrowed all Adam's books.
3  -- next, let's describe the student by using double negation
4  WHERE NOT EXISTS ( -- it means 'no such a book by Adam Smith ...'
5      SELECT *
6      FROM book b
7      WHERE authors = 'Adam_Smith'
8      AND NOT EXISTS ( -- '... has not been borrowed by the student'
9          SELECT *
10         FROM loan l
11         WHERE l.book = b.isbn13 AND l.borrower = s.email));
```

---

## Question 2 (d) - Extra

Nightmare comes again!!!

**Extra:** Can you spot the difference between the 2 queries below:

```
1  SELECT s.email, s.name
2  FROM student s
3  WHERE NOT EXISTS (
4      SELECT *
5      FROM book b
6      WHERE authors = 'Adam_Smith'
7      AND NOT EXISTS (
8          SELECT *
9          FROM loan l
10         WHERE l.book = b.isbn13
11         AND l.borrower = s.email));
```

```
1  SELECT s.email, s.name
2  FROM student s
3  WHERE NOT EXISTS (
4      SELECT *
5      FROM loan l
6      WHERE l.borrower = s.email
7      AND NOT EXISTS (
8          SELECT *
9          FROM book b
10         WHERE b.authors = 'Adam_Smith'
11         AND l.book = b.isbn13));
```

email	name
yeojiahao1989@yahoo.com	YEO JIA HAO

(1 row)

email	name
glee@msn.com	GERALDINE LEE
tcy@hotmail.com	TANG CHEE YONG
awong007@msn.com	ADELIN WONG
tikki@gmail.com	TIKKI TAVI
rikki@gmail.com	RIKKI TAVI

(5 rows)

## Question 2 (d) - Extra

How to interpret the RHS query?

---

```
1 SELECT s.email, s.name
2 FROM student s
3 WHERE NOT EXISTS (
4     SELECT *
5     FROM book b
6     WHERE authors = 'Adam_Smith'
7     AND NOT EXISTS (
8         SELECT *
9         FROM loan l
10        WHERE l.book = b.isbn13
11        AND l.borrower = s.email));
```

---

---

```
1 SELECT s.email, s.name
2 FROM student s
3 WHERE NOT EXISTS (
4     SELECT *
5     FROM loan l
6     WHERE l.borrower = s.email
7     AND NOT EXISTS (
8         SELECT *
9         FROM book b
10        WHERE b.authors = 'Adam_Smith'
11        AND l.book = b.isbn13));
```

---

Print the names and emails of students who have left NO borrowing record in the loan table, of any book NOT written by Adam Smith

Output 5 students who actually have never borrowed any book according to the loan table.

## Question 2 (d) - Extra (2)

There always exists alternative solution, either simpler or more complicated.

Can you guy come up with another solution (or idea) to solve this question rather than using DOUBLE NEGATION.

First of all, we may calculate how many books are written by Adam Smith -> 2 books

Then select distinct those students who has borrowed at least one book of Adam Smith (i.e., the email of student has ever appeared in loan table as the borrower of Adam Smith's books)

Wrap the first two queries into a bigger query, that using HAVING to enforce the aggregate condition to print students who has borrow (2) different books written by Adam Smith according to all records of the loan table

For any further question, please feel free to email me:

[huasong.meng@u.nus.edu](mailto:huasong.meng@u.nus.edu)

Copyright 2021 Mark H. Meng. All rights reserved.