BT5110: Tutorial 7 — Aggregate and Nested Queries

Pratik Karmakar

School of Computing,
National University of Singapore

AY25/26 S1

Scenario

Students at the **National University of Ngendipura (NUN)** buy, lend, and borrow books.
NUNStA commissions *Apasaja Private Limited* to implement an online book exchange system that records:

- Students: name, faculty, department, **email** (identifier), join year.
- Books: title, authors, publisher, year, edition, **ISBN10**, **ISBN13**.
- Loans: `borrowed` date, `returned` date (may be `NULL`).

Auditing preserves data for graduated students and copies with loan records.
This tutorial uses the schema/data created in "Creating and Populating Tables".

Important Constraints

This tutorial:

Focus on **aggregate** and **nested** queries.
Practice writing equivalent formulations and ensure readability.

Questions — Aggregate

1. Aggregate Queries

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

(b) For each faculty, print the number of loans that involve an owner and borrower from this faculty.

(c) What are the average and standard deviation of loan duration (in days)?

Questions — Nested

2. Nested Queries

(a) Print titles of books that have never been borrowed (use nested query).

(b) Print names of students who own a copy of a book never lent.

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

(d) Print emails and names of students who borrowed all books authored by Adam Smith.

1(a). Number of loans in same department

```sql
SELECT COUNT(*)
FROM loan l, student s1, student s2
WHERE l.owner = s1.email
  AND l.borrower = s2.email
  AND s1.department = s2.department;
```

1(b). Number of loans within faculty

```sql
SELECT d1.faculty, COUNT(*)
FROM loan l, student s1, student s2,
     department d1, department d2
WHERE l.owner = s1.email
  AND l.borrower = s2.email
  AND s1.department = d1.department
  AND s2.department = d2.department
  AND d1.faculty = d2.faculty
GROUP BY d1.faculty;
```

1(c). Average and Stddev of Loan Duration

Alternative #1 (inline CASE):

```
1  SELECT CEIL(AVG((CASE
2      WHEN l.returned ISNULL
3        THEN CURRENT_DATE
4      ELSE l.returned END) - l.borrowed + 1)),
5    CEIL(STDDEV_POP((CASE
6      WHEN l.returned ISNULL
7        THEN CURRENT_DATE
8      ELSE l.returned END) - l.borrowed + 1))
9  FROM loan l;
```

1(c). Alternative Query (Subquery)

```
1  SELECT CEIL(AVG(temp.duration)),
2         CEIL(STDDEV_POP(temp.duration))
3  FROM (
4    SELECT ((CASE
5       WHEN l.returned ISNULL THEN CURRENT_DATE
6       ELSE l.returned END) - l.borrowed + 1) AS duration
7    FROM loan l
8  ) AS temp;
```

2(a). Books never borrowed

Option #1 (NOT IN):

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

2(a). Alternative (<> ALL)

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

2(b). Students who never lent their copy

```sql
SELECT s.name
FROM student s
WHERE s.email IN (
  SELECT c.owner
  FROM copy c
  WHERE NOT EXISTS (
    SELECT *
    FROM loan l
    WHERE l.owner = c.owner
      AND l.book = c.book
      AND l.copy = c.copy));
```

2(c). Top lenders in each department

```sql
SELECT s.department, s.name, COUNT(*)
FROM student s, loan l
WHERE l.owner = s.email
GROUP BY s.department, s.email, s.name
HAVING COUNT(*) >= ALL (
  SELECT COUNT(*)
  FROM student s1, loan l1
  WHERE l1.owner = s1.email
    AND s.department = s1.department
  GROUP BY s1.email);
```

2(d). Borrowed all Adam Smith books

```sql
SELECT s.email, s.name
FROM student s
WHERE NOT EXISTS (
  SELECT *
  FROM book b
  WHERE authors = 'Adam Smith'
    AND NOT EXISTS (
      SELECT *
      FROM loan l
      WHERE l.book = b.ISBN13
        AND l.borrower = s.email));
```

2(d). Borrowed all Adam Smith books

Student	Truth Value
s1	T
s2	T
s3	F
s4	F
s5	T
s6	F
s7	T

Students who have borrowed all books by Adam Smith - Students who have truth value T in the table shown - But how do we assign the truth values?

The logical statement to evaluate:

If there is a book by Adam Smith, the student has borrowed it.

P: there is a book by Adam Smith

Q: the student has borrowed it

The Boolean Logic Form:

$P \rightarrow Q$ (P implies Q) $\equiv \neg P \vee Q$

P	Q	$P \rightarrow Q$	$\neg P \vee Q$
F	F	T	T
F	T	T	T
T	F	F	F
T	T	T	T

2(d). Borrowed all Adam Smith books

```
SELECT s.email, s.name
FROM student s
WHERE NOT EXISTS (
    SELECT *
    FROM book b
    WHERE authors = 'Adam Smith'
        AND NOT EXISTS (
            SELECT *
            FROM loan l
            WHERE l.book = b.isbn13
            AND l.borrower = s.email));
```

NOT Gate (¬)

A book by Adam Smith exists (P)

AND Gate (∧)

The student borrows the book (Q)

$$\neg(P \wedge \neg Q) \equiv \neg P \vee \neg(\neg Q) \equiv \neg P \vee Q \equiv P \rightarrow Q$$

Guidelines & Marking Tips

- **No hardcoding.** Queries must work on any dataset consistent with the schema.
- Use **quantifiers** (`ALL`/`ANY`) explicitly in subqueries.
- Avoid using **TOP N / LIMIT** for max/min — prefer ALL with GROUP BY.
- Be cautious of duplicate elimination; use `DISTINCT` only when needed.
- Write queries in **readable, indented style**.

Questions?
Drop a mail at: pratik.karmakar@u.nus.edu