BT5110: Tutorial 6 — Simple 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 that records:

- Student info: name, faculty, department, **email** (identifier), join date (year).

- Book info: title, authors, publisher, year, edition, **ISBN10**, **ISBN13** (industry IDs; *unique*).

- Loans: borrowed date, returned date (may be NULL).

Auditing keeps data for (i) copies with loans and (ii) graduated students with loaned books.

This tutorial uses the schema/data created in "Creating and Populating Tables".

Important Constraints for This Tutorial

Use *simple queries only*

No **nested** or **aggregate** queries in answers.
Focus on **single**-table and **multi**-table joins and set operators.

We'll present **equivalent** formulations (e.g., CROSS JOIN vs INNER JOIN, UNION/INTERSECT/EXCEPT) and discuss readability best practices.

Questions — Single-table

1. Single-Table Queries

(a) Print the different departments.

(b) Print the different departments in which students are enrolled.

(c) For each copy that has been borrowed and returned, print the ISBN13 and the loan **duration**. Order by ISBN13 (ASC) then duration (DESC). Use a **single** table.

2. Multi-Table Queries

(a) For each unreturned loan of a book published by 'Wiley', print: book title, owner name+faculty, borrower name+faculty.

(b) Print emails of students who *borrowed or lent* a copy **before** they joined the University.

(c) Print emails of students who *borrowed or lent* a copy **on the day** they joined.

(d) Print emails of students who *borrowed and lent* a copy **on the day** they joined.

(e) Print emails of students who *borrowed but did not lend* a copy **on the day** they joined.

(f) Print ISBN13 of books that have **never** been borrowed.

1(a). Different departments (single table)

Good practice: alias tables; refer to columns via the alias.

```
1  SELECT d.department
2  FROM department AS d;
```

Why no DISTINCT? department is the **PRIMARY KEY** of table
department, so duplicates cannot occur.

1(b). Departments with enrolled students

Query the student table; many students share the same department, so we need DISTINCT.

```
1   SELECT DISTINCT s.department
2   FROM student AS s;
```

Wording matters. If we asked "*department of the different students*", then DISTINCT should **not** be on department (students are identified by email, not department).

1(c). Loan duration from a single table (returned only)

```
1  SELECT l.book, l.returned - l.borrowed + 1 AS duration
2  FROM loan AS l
3  WHERE l.returned IS NOT NULL
4  ORDER BY l.book ASC, duration DESC;
```

1(c). Loan duration including unreturned

Using COALESCE

```
1  SELECT l.book,
2         (COALESCE(l.returned, CURRENT_DATE)
3          - l.borrowed + 1) AS duration
4  FROM loan AS l
5  ORDER BY l.book ASC, duration DESC;
```

Using CASE

```
1  SELECT l.book,
2         ((CASE WHEN l.returned IS NULL
3                THEN CURRENT_DATE
4                ELSE l.returned END)
5          - l.borrowed + 1) AS duration
6  FROM loan AS l
7  ORDER BY l.book ASC, duration DESC;
```

2(a). Unreturned Wiley loans (with COPY join)

```
1   SELECT b.title,
2          s1.name AS ownerName,  d1.faculty AS ownerFaculty,
3          s2.name AS borrowerName, d2.faculty AS borrowerFaculty
4   FROM loan AS l, book AS b, copy AS c,
5        student AS s1, student AS s2,
6        department AS d1, department AS d2
7   WHERE l.book = b.ISBN13
8     AND c.book = l.book AND c.copy = l.copy AND c.owner = l.owner
9     AND l.owner = s1.email AND l.borrower = s2.email
10    AND s1.department = d1.department AND s2.department =
       d2.department
11    AND b.publisher = 'Wiley'
12    AND l.returned IS NULL;
```

2(a). Unreturned Wiley loans (omit COPY via PK–FK)

```
1   SELECT  b.title,
2           s1.name AS ownerName,  d1.faculty AS ownerFaculty,
3           s2.name AS borrowerName, d2.faculty AS borrowerFaculty
4   FROM loan AS l, book AS b,
5        student AS s1, student AS s2,
6        department AS d1, department AS d2
7   WHERE l.book = b.ISBN13
8     AND l.owner = s1.email   AND l.borrower = s2.email
9     AND s1.department = d1.department AND s2.department =
        d2.department
10    AND b.publisher = 'Wiley' AND l.returned IS NULL;
```

2(a). Using `INNER JOIN` (clear ON vs WHERE)

```
1   SELECT b.title,
2          s1.name AS ownerName,    d1.faculty AS ownerFaculty,
3          s2.name AS borrowerName, d2.faculty AS borrowerFaculty
4   FROM loan AS l
5   INNER JOIN book       AS b  ON l.book     = b.ISBN13
6   INNER JOIN student    AS s1 ON l.owner    = s1.email
7   INNER JOIN student    AS s2 ON l.borrower = s2.email
8   INNER JOIN department AS d1 ON s1.department = d1.department
9   INNER JOIN department AS d2 ON s2.department = d2.department
10  WHERE b.publisher = 'Wiley' AND l.returned IS NULL;
```

Convention: `ON` for PK–FK join predicates; `WHERE` for additional filters.

2(b). Borrowed or lent *before* joining

Either role (owner or borrower) and **date check**:

```
1  SELECT DISTINCT s.email
2  FROM loan AS l, student AS s
3  WHERE (s.email = l.borrower OR s.email = l.owner)
4    AND l.borrowed < s.year;
```

Distributed condition (equivalent):

```
1  SELECT DISTINCT s.email
2  FROM loan AS l, student AS s
3  WHERE (s.email = l.borrower AND l.borrowed < s.year)
4     OR (s.email = l.owner    AND l.borrowed < s.year);
```

2(c). Borrowed or lent *on* joining day

Direct form (needs DISTINCT):

```
1  SELECT DISTINCT s.email
2  FROM loan AS l, student AS s
3  WHERE (s.email = l.borrower OR s.email = l.owner)
4    AND l.borrowed = s.year;
```

Using UNION (deduplicates by default):

```
1  SELECT s.email
2  FROM loan AS l, student AS s
3  WHERE s.email = l.borrower AND l.borrowed = s.year
4  UNION
5  SELECT s.email
6  FROM loan AS l, student AS s
7  WHERE s.email = l.owner AND l.borrowed = s.year;
```

2(d). Borrowed *and* lent *on* joining day

Using `INTERSECT`:

```
1  SELECT s.email
2  FROM loan AS l, student AS s
3  WHERE s.email = l.borrower AND l.borrowed = s.year
4  INTERSECT
5  SELECT s.email
6  FROM loan AS l, student AS s
7  WHERE s.email = l.owner AND l.borrowed = s.year;
```

Without `INTERSECT` (two aliases of `loan`):

```
1  SELECT DISTINCT s.email
2  FROM loan AS l1, loan AS l2, student AS s
3  WHERE s.email = l1.borrower AND l1.borrowed = s.year
4    AND s.email = l2.owner    AND l2.borrowed = s.year;
```

2(e). Borrowed *but did not lend on* joining day

```
1   SELECT s.email
2   FROM loan AS l, student AS s
3   WHERE s.email = l.borrower AND l.borrowed = s.year
4   EXCEPT
5   SELECT s.email
6   FROM loan AS l, student AS s
7   WHERE s.email = l.owner AND l.borrowed = s.year;
```

2(f). Books *never* borrowed

Using EXCEPT:

```
1  SELECT b.ISBN13
2  FROM book AS b
3  EXCEPT
4  SELECT l.book
5  FROM loan AS l;
```

Using LEFT OUTER JOIN + IS NULL:

```
1  SELECT b.ISBN13
2  FROM book AS b
3  LEFT OUTER JOIN loan AS l
4    ON b.ISBN13 = l.book
5  WHERE l.book IS NULL;
```

Guidelines & Marking Tips

- **No hardcoding.** Queries must work on any dataset consistent with the schema.

- **Constants only if stated.** If the question names a constant (e.g., 'Wiley'), you may use it; otherwise avoid.

- **Readable style.** Use table aliases, qualify columns, and uppercase SQL keywords.

- **Set operators** (UNION/INTERSECT/EXCEPT) inherently deduplicate; DISTINCT is redundant with them.

- **Partial marks:** (i) query executes, (ii) correct columns (names, types, order), (iii) minimal row differences.

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