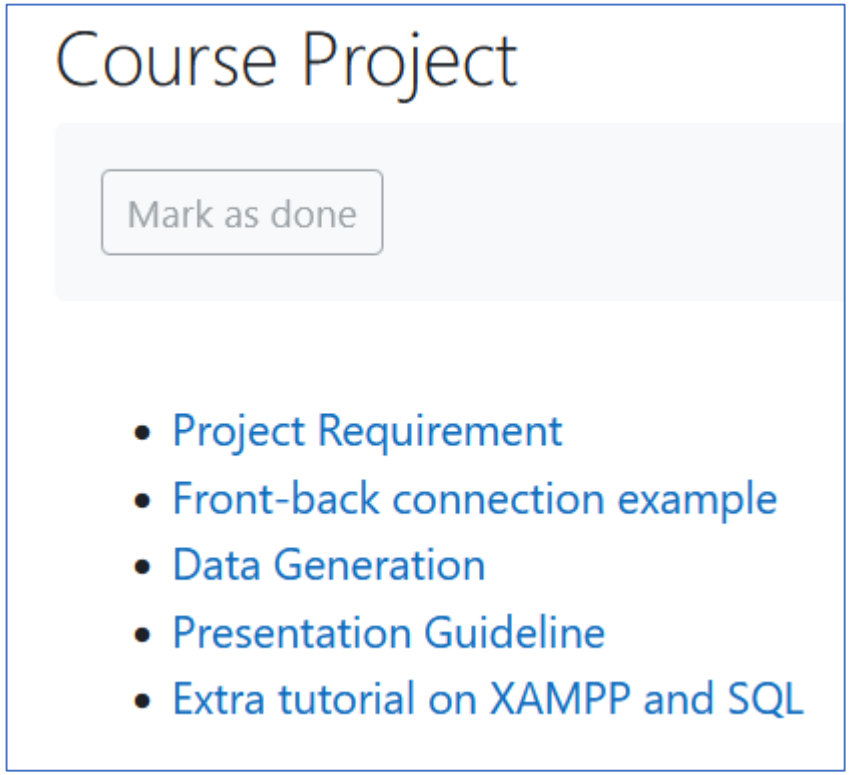
Database Lab 10
Correlated Subquery and Division
Addendum

Dr. Jefferson Fong

Project

- Static (HTML and CSS) web pages are due on Sun Nov 30.
  - You can use other software tools, but no extra credit will be given on how nice you web pages look.
  - Our focus is on the database. See the Course Project links for details.
  - The front end needs to be good enough to demonstrate the database requirements.



Make-up Midterm

- Students who are approved to take the make-up midterm exam have been notified. Please join the WeCom chat group.

Slide 9 LHS bottom:

- To test programme – temp = ∅, NOT EXISTS is used. It returns TRUE if it is followed by an empty set (in the blue **ellipses** … box).

Slide 9-10

- Difficult to derive the algorithm.
- To help you to remember this correlated subquery:
  - Use example for catalogue ÷ programme,
  - There are 3 SELECT's, with C1 appearing in the 1st and 3rd SELECT.
  - Has NOT EXISTS and EXCEPT between two SELECT's.
  - Catalogue is the **top dividend**, in C1 and C2 in 1st and 3rd SELECT
  - Programme is the **bottom divisor**, in the 2nd SELECT.

- Demonstrate the algorithm (Slide 9) on our example (Slide 4).
- 1st: SELECT C1.c_name FROM catalogue AS C1
- From table catalogue (Slide 4), we have

  { (Database, CST), (OS, CST), (DATABASE, DS), (Data Analytics, DS) }
  - Feed (Database, CST) into the algorithm:
    - 3rd: SELECT p_name FROM catalogue AS C2 WHERE C2.c_name=C1.c_name
      - C_name is Database, we have (Database, CST) and (Database, DS) from C2; so p_name is {CST, DS}
    - 2nd: SELECT p_name FROM programme
      - {CST, DS}
    - So 2nd select – 3rd select = ∅ (empty set)
    - 1st: WHERE NOT EXISTS (empty set) is true.
    - So c_name Database in (Database, CST) belongs to catalogue ÷ programme.

- Feed (OS, CST) into the algorithm:
  - 3rd: SELECT p_name FROM catalogue AS C2 WHERE C2.c_name=C1.c_name
    - C_name is OS, we have (OS, CST) from C2; so p_name is {CST}
  - 2nd: SELECT p_name FROM programme
    - {CST, DS}
  - So 2nd select – 3rd select = {DS}
  - 1st: WHERE NOT EXISTS is not empty, returns false.
  - So c_name OS in (OS, CST) does not belongs to catalogue ÷ programme.
- Similarly for (DATABASE, DS) is in, and (Data Analytics, DS) is not.

- The algorithm works in our example.

Slide 10-11

- If your version of MySQL does not support EXCEPT, use NOT IN instead.

Example from Lab 10 PPT, Slides 9-11:

Query: Find course names that are taught by all programmes (or every programme).

Solution: Catelogue(c_name, p_name) ÷ Programme(p_name)

- Note the schema result is c_name.

Example in Slide 12

Query: Find customer_id who rented film from every staff.

Solution: rental(customer_id, staff_id) ÷ staff(staff_id)

So in Slide 11's query, change

Catalogue to Rental,
Programme to Staff,
c_name to customer_id,
p_name to staff_id.

```
SELECT DISTINCT customer_id
FROM rental AS r1
WHERE NOT EXISTS(
  SELECT *
  FROM staff
  WHERE staff_id NOT IN (
    SELECT staff_id
    FROM rental AS r2
    WHERE r2.customer_id=r1.customer_id
  )
);
```

Exercise 1: Find customers who rented films from all stores.

- The **apparent solution** seems to be

  Customer(customer_id, store_id) ÷ Store(store_id)

- But this query **returns empty rows!**

- The table Customer(customer_id, store_id, first_name, last_name, …) is misleading.
- The store_id refers to the store where the customer **registered**, **not** where he or she **rented** films.
  - Check the contents of the customer table in Sakila.
  - The apparent solution is for "find customers who **registered** in all stores", not what we want.

Apparent solution:

```
SELECT c1.customer_id
FROM customer AS c1
WHERE NOT EXISTS(
  SELECT *
  FROM store
  WHERE store_id NOT IN (
    SELECT c2.store_id
    FROM customer AS c2
    WHERE
      c1.customer_id=c2.customer_id
  )
);
```

Change:
Catalogue to Customer,
Programme to Store,
c_name to customer_id,
p_name to store_id.

Exercise 1: Find customers who rented films from all stores.

Correct solution:

To find out which film the customer rented, we must go to the rental table.

Rental(rental_id, inventory_id, customer_id).

Walk to table Inventory(inventory_id, film_id, **store_id**). This store_id refers to the location of the film, from where the customer rented.

So join tables rental and inventory using inventory_id to get the correct store_id.

The innermost subquery is then (with customer AS c1)

```
SELECT store_id
FROM (rental AS r1 JOIN inventory USING (inventory_id))
WHERE c1.customer_id=r1.customer_id
```

Replace the **innermost subquery** marked in **red** in the **apparent solution** in the last slide.


"That's all Folks!"