Australian National University

Week 3 – Data Manipulation Language

W3 – Housekeeping

1. Using own computer (in Labs or Campus) - please log in to partch

   - Instruction in Lab 1
2. Thanks Patrik for providing More SQL exercises

   - Find in Wee3 learning materials on Wattle
3. Drop-in session on Tuesday 11am-12pm at CSIT N19.

Week 3

- Quiz feedback
- Insert, delete, update
- Select: simple SQL queries
- Select: advanced SQL queries

Australian National University

Quiz-Q5

Q5 Version 1 (latest)

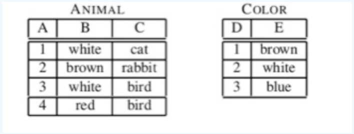
Question 1

Not yet answered

Marked out of 0.10

Given the following two relations, and the SQL query

SELECT *
FROM ANIMAL INNER JOIN COLOR
ON ANIMAL.B = COLOR.E;



ANIMAL

A	B	C
1	white	cat
2	brown	rabbit
3	white	bird
4	red	bird

COLOR

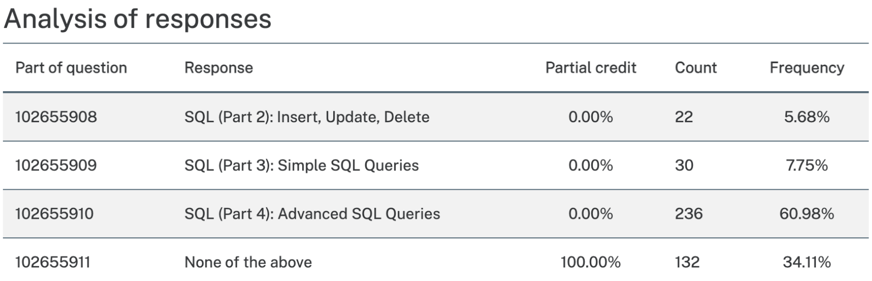
D	E
1	brown
2	white
3	blue

How many rows will be in the resulting table of the above SQL query?

- ○ a. 4
- ○ b. None of the above
- ○ c. 5
- ○ d. 3

Australian National University

Quiz-Q6 statistics



Analysis of responses

Part of question	Response	Partial credit	Count	Frequency
102655908	SQL (Part 2): Insert, Update, Delete	0.00%	22	5.68%
102655909	SQL (Part 3): Simple SQL Queries	0.00%	30	7.75%
102655910	SQL (Part 4): Advanced SQL Queries	0.00%	236	60.98%
102655911	None of the above	100.00%	132	34.11%

Week 3

- Quiz feedback
- Insert, delete, update
- Select: simple SQL queries
- Select: advanced SQL queries

Insert, Update, Delete

- The INSERT statement is used to add tuples into a relation.

```
INSERT INTO table_name
            [(attribute_name,...,attribute_name)]
     VALUES (value,...,value),...,(value,...,value);
```

- The UPDATE statement is used to modify attribute values of one or more selected tuples.

```
UPDATE table_name
   SET attribute_name = value,...,attribute_name = value
[WHERE selection_condition];
```

- The DELETE statement is used to remove tuples from a relation.

```
DELETE FROM table_name
     [WHERE selection_condition];
```

Australian National University

Insert – Example

```
CREATE TABLE STUDENT(StudentID INT PRIMARY KEY, Name VARCHAR(50),
DoB DATE, Email VARCHAR(100));
```

- Will the following Insert statements work?

-
  ```
  INSERT INTO STUDENT
  VALUES (456, 'Tom', '25/01/1988', 'tom@gmail.com');
  ```
  Yes.

-
  ```
  INSERT INTO STUDENT(StudentID)
  VALUES (459);
  ```
  Yes. The values for Name, DoB and Email will be NULL.

-
  ```
  INSERT INTO STUDENT(Name, DoB, Email)
  VALUES ('John', '15/11/1998', 'john@gmail.com');
  ```
  No. The primary key value cannot be NULL.

Australian National University

Update – Example

STUDENT			
StudentID	Name	DoB	Email
456	Tom	25/01/1988	tom@gmail.com
458	Peter	23/05/1993	peter@gmail.com
459	Fran	11/09/1987	frankk@gmail.com

- What is the resulting table after executing the following statement?

```
UPDATE STUDENT SET Name='Tom Lee', Email='tom.lee@yahoo.com'
WHERE StudentID=456;
```

STUDENT			
StudentID	Name	DoB	Email
456	**Tom Lee**	25/01/1988	**tom.lee@yahoo.com**
458	Peter	23/05/1993	peter@gmail.com
459	Fran	11/09/1987	frankk@gmail.com

Australian National University

Delete – Example

STUDENT			
StudentID	Name	DoB	Email
456	Tom	25/01/1988	tom@gmail.com
458	Peter	23/05/1993	peter@gmail.com
459	Fran	11/09/1987	frankk@gmail.com

- What is the resulting table after executing the following statement?

```
DELETE FROM STUDENT WHERE StudentID=456;
```

STUDENT			
StudentID	Name	DoB	Email
458	Peter	23/05/1993	peter@gmail.com
459	Fran	11/09/1987	frankk@gmail.com

```
DELETE FROM STUDENT;
```

STUDENT			
StudentID	Name	DoB	Email

```
DROP TABLE STUDENT;
```

The Table STUDENT is deleted.

What is the difference?: DELETE/ DROP, UPDATE/ALTER

Australian National University

Update and Delete - Referential Actions

ENROL				
StudentID	CourseNo	Semester	Status	EnrolDate
456	COMP1130	2016 S1	active	25/02/2016
458	COMP1130	2016 S1	active	25/02/2016
456	COMP2400	2016 S2	active	09/03/2016

STUDENT			
StudentID	Name	DoB	Email
456	Tom	25/01/1988	tom@gmail.com
458	Peter	20/02/1991	peter@hotmail.com

- Referential actions specify what happens in case of deleting or updating referenced tuples. SQL offers the following possibilities:
  - `NO ACTION` (default) will throw an error if one tries to delete a row (or update the primary key value) referenced.
  - `CASCADE` will force the referencing tuples to be deleted (or updated with new primary key value).
  - `SET NULL` will force the corresponding values in the referencing tuples to be set to a null value (i.e., unknown).
  - `SET DEFAULT` will force the corresponding values in the referencing tuples to be set to a specified default value.

Delete – Example

- Consider the following foreign key defined on ENROL:

```
FOREIGN KEY(StudentID) REFERENCES STUDENT(StudentID)
      ON DELETE NO ACTION
```

ENROL				
StudentID	CourseNo	Semester	Status	EnrolDate
456	COMP1130	2016 S1	active	25/02/2016
458	COMP1130	2016 S1	active	25/02/2016
456	COMP2400	2016 S2	active	09/03/2016

STUDENT			
StudentID	Name	DoB	Email
456	Tom	25/01/1988	tom@gmail.com
458	Peter	20/02/1991	peter@hotmail.com

```
DELETE FROM STUDENT WHERE StudentID=456;
```

- The deletion of a student who has enrolled at least one course will throw out an error concerning the foreign key.

Australian National University

Delete – Example

- Consider the following foreign key defined on ENROL:

```
FOREIGN KEY(StudentID) REFERENCES STUDENT(StudentID)
    ON DELETE CASCADE
```

ENROL

StudentID	CourseNo	Semester	Status	EnrolDate
456	COMP1130	2016 S1	active	25/02/2016
458	COMP1130	2016 S1	active	25/02/2016
456	COMP2400	2016 S2	active	09/03/2016

STUDENT

StudentID	Name	DoB	Email
456	Tom	25/01/1988	tom@gmail.com
458	Peter	20/02/1991	peter@hotmail.com

```
DELETE FROM STUDENT WHERE StudentID=456;
```

- We would have ENROL below after deleting the student 456.

StudentID	CourseNo	Semester	Status	EnrolDate
458	COMP1130	2016 S1	active	25/02/2016

Week 3

- Quiz feedback
- Insert, delete, update
- Select: simple SQL queries
- Select: advanced SQL queries

Select Statement

- **Select Statement**

```
SELECT *
FROM World
WHERE "Someone"
LIKE '%You%'
```

Select Statement

- The SELECT statement has the following basic form:

```
   SELECT attribute_list
     FROM table_list
   [WHERE condition]
[GROUP BY attribute_list [HAVING group_condition]]
[ORDER BY attribute_list];
```

Australian National University

Select Statement

STUDENT			
StudentID	Name	DoB	Email
456	Tom	25/01/1988	tom@hotmail.com
458	Peter	23/05/1993	peter@gmail.com
459	Fran	11/09/1987	frankk@gmail.com

- What is the result for the following Select statement?

```
SELECT * FROM STUDENT WHERE Email like '%@gmail.com';
```

StudentID	Name	DoB	Email
458	Peter	23/05/1993	peter@gmail.com
459	Fran	11/09/1987	frankk@gmail.com

```
SELECT StudentID FROM STUDENT WHERE Email like '%@gmail.com';
```

StudentID
458
459

Select Statement

STUDENT			
StudentID	Name	DoB	Email
456	Tom	25/01/1988	tom@hotmail.com
458	peter	23/05/1993	peter@gmail.com
459	Fran	11/09/1987	frankk@gmail.com
460	Peter	03/09/1992	Peter@Github.com

- What is the result for the following Select statement?

```
SELECT * FROM STUDENT WHERE Name = 'Peter';
```

STUDENT			
StudentID	Name	DoB	Email
460	Peter	03/09/1992	Peter@Github.com

```
SELECT * FROM STUDENT WHERE lower(Name) = 'peter';
```

STUDENT			
StudentID	Name	DoB	Email
458	peter	23/05/1993	peter@gmail.com
460	Peter	03/09/1992	Peter@Github.com

Australian National University

Select + Group By

- GROUP BY *attribute_list* groups tuples for each value combination in the attribute_list.
- Aggregate functions can be applied to aggregate a group of attribute values into a single value, e.g.,
  - COUNT returns the total number of argument values
  - AVG returns the average of argument values
  - MIN returns the minimum value of the arguments
  - MAX returns the maximum value of the arguments
  - SUM returns the sum of the argument values
- We can use HAVING *condition* to add the condition on the groups.

Select + Group By – Example

Study		
StudentID	CourseNo	Hours
111	COMP2400	120
222	COMP2400	115
333	STAT2001	120
111	BUSN2011	110
111	ECON2102	120
333	BUSN2011	130

- What would happen for the following SELECT + Group By StudentID?

```
SELECT ...
FROM STUDY
Group By StudentID;
```

Select + Group By – Example

Group	STUDY		
StudentID	StudentID	CourseNo	Hours
111	111	COMP2400	120
	111	BUSN2011	110
	111	ECON2102	120
222	222	COMP2400	115
333	333	STAT2001	120
	333	BUSN2011	130

- What is the result for the following SELECT + Group By StudentID?

```
SELECT StudentID
FROM STUDY
Group By StudentID;
```

StudentID
111
222
333

Australian National University

Select + Group By – Example

Group StudentID	STUDY StudentID	CourseNo	Hours
111	111	COMP2400	120
	111	BUSN2011	110
	111	ECON2102	120
222	222	COMP2400	115
333	333	STAT2001	120
	333	BUSN2011	130

- What is the result for the following SELECT + Group By StudentID?

```
SELECT StudentID, COUNT(*)
FROM STUDY
Group By StudentID;
```

StudentID	COUNT
111	3
222	1
333	2

Australian National University

Select + Group By – Example

Group	STUDY		
StudentID	StudentID	CourseNo	Hours
111	111	COMP2400	120
	111	BUSN2011	110
	111	ECON2102	120
222	222	COMP2400	115
333	333	STAT2001	120
	333	BUSN2011	130

- What is the result for the following SELECT + Group By StudentID?

```
SELECT StudentID, MAX(hours)
FROM STUDY
Group By StudentID;
```

StudentID	MAX
111	120
222	115
333	130

Australian National University

Select + Group By – Example

Group	STUDY		
StudentID	StudentID	CourseNo	Hours
111	111	COMP2400	120
	111	BUSN2011	110
	111	ECON2102	120
222	222	COMP2400	115
333	333	STAT2001	120
	333	BUSN2011	130

- What is the result for the following SELECT + Group By StudentID?

```
SELECT StudentID, COUNT(StudentID)
FROM STUDY
Group By StudentID;
```

StudentID	COUNT
111	3
222	1
333	2

Select + Group By – Example

Group	STUDY		
StudentID	StudentID	CourseNo	Hours
111	111	COMP2400	120
	111	BUSN2011	110
	111	ECON2102	120
222	222	COMP2400	115
333	333	STAT2001	120
	333	BUSN2011	130

- What is the result for the following SELECT + Group By StudentID?

```
SELECT StudentID, CourseNo
FROM STUDY
Group By StudentID;
```

Error Message.

Australian National University

Select + Group By – Example

Group	STUDY		
StudentID	StudentID	CourseNo	Hours
111	111	COMP2400	120
	111	BUSN2011	110
	111	ECON2102	120
222	222	COMP2400	115
333	333	STAT2001	120
	333	BUSN2011	130

- What is the result for the following SELECT + Group By StudentID?

```
SELECT *
FROM STUDY
Group By StudentID;
```

Error Message.

Select + Group By – Example

Group	STUDY		
StudentID	StudentID	CourseNo	Hours
111	111	COMP2400	120
	111	BUSN2011	110
	111	ECON2102	120
222	222	COMP2400	115
333	333	STAT2001	120
	333	BUSN2011	130

- What is the result for the following SELECT + Group By StudentID?

```
SELECT COUNT(*)
FROM STUDY
Group By StudentID;
```

COUNT
3
1
2

Australian National University

Select + Group By + Having – Example

Group	STUDY		
CourseNo	StudentID	CourseNo	Hours
BUSN2011	111	BUSN2011	110
	333	BUSN2011	130
COMP2400	111	COMP2400	120
	222	COMP2400	115
ECON2102	111	ECON2102	120
STAT2001	333	STAT2001	120

- What is the result for the following SELECT + Group By + Having?

```
SELECT CourseNo
FROM STUDY
Group By CourseNo
Having MAX(Hours) > 120;
```

CourseNo
BUSN2011

Australian National University

Select + Group By + Having – Example

Group	STUDY		
CourseNo	StudentID	CourseNo	Hours
BUSN2011	111	BUSN2011	110
	333	BUSN2011	130
COMP2400	111	COMP2400	120
	222	COMP2400	115
ECON2102	111	ECON2102	120
STAT2001	333	STAT2001	120

- What is the result for the following SELECT + Group By + Having?

```
SELECT CourseNo
FROM STUDY
Group By CourseNo
Having COUNT(*) > 1;
```

CourseNo
BUSN2011
COMP2400

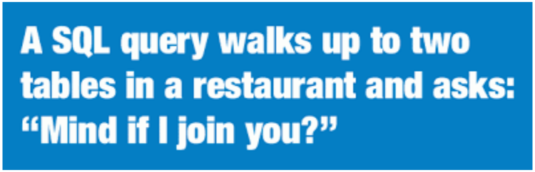
Week 3

- Quiz feedback
- Insert, delete, update
- Select: simple SQL queries
- Select: advanced SQL queries
  – set operations, join operations, subqueries

Australian National University

A Bunch of Tables

- **A Bunch of Tables**


A SQL query walks up to two tables in a restaurant and asks: "Mind if I join you?"

Set Operations

- SQL incorporates several set operations: `UNION` (set union) and `INTERSECT` (set intersection), and sometimes `EXCEPT` (set difference / minus).

- Set operations result in return of a relation of tuples (no duplicates).

- Set operations apply to relations that have the same attribute types appearing in the same order.

Australian National University

Set Operations

Study		
StudentID	CourseNo	Hours
111	COMP2400	120
222	COMP2400	115
333	STAT2001	120
111	BUSN2011	110
111	ECON2102	120
333	BUSN2011	130

- What is the result for the following SQL query?

```
SELECT StudentID FROM STUDY
WHERE CourseNo='COMP2400'
UNION
SELECT StudentID FROM STUDY
WHERE CourseNo='ECON2102';
```

StudentID
111
222

UNION

StudentID
111

Set Operations

Study		
StudentID	CourseNo	Hours
111	COMP2400	120
222	COMP2400	115
333	STAT2001	120
111	BUSN2011	110
111	ECON2102	120
333	BUSN2011	130

- What is the result for the following SQL query?

```
SELECT StudentID FROM Study
WHERE CourseNo='COMP2400'
UNION
SELECT StudentID FROM Study
WHERE CourseNo='ECON2102';
```

StudentID
111
222

Australian National University

Set Operations

STUDY		
StudentID	CourseNo	Hours
111	COMP2400	120
222	COMP2400	115
333	STAT2001	120
111	BUSN2011	110
111	ECON2102	120
333	BUSN2011	130

- What is the result for the following SQL query?

```
SELECT CourseNo FROM STUDY
WHERE StudentID=111
 EXCEPT
SELECT CourseNo FROM STUDY
WHERE StudentID=222;
```

CourseNo
COMP2400
BUSN2011
ECON2102

EXCEPT

CourseNo
COMP2400

Set Operations

STUDY		
StudentID	CourseNo	Hours
111	COMP2400	120
222	COMP2400	115
333	STAT2001	120
111	BUSN2011	110
111	ECON2102	120
333	BUSN2011	130

- What is the result for the following SQL query?

```
SELECT CourseNo FROM STUDY
WHERE StudentID=111
 EXCEPT
SELECT CourseNo FROM STUDY
WHERE StudentID=222;
```

CourseNo
BUSN2011
ECON2102

Australian National University

Set Operations

STUDY		
StudentID	CourseNo	Hours
111	COMP2400	120
222	COMP2400	115
333	STAT2001	120
111	BUSN2011	110
111	ECON2102	120
333	BUSN2011	130

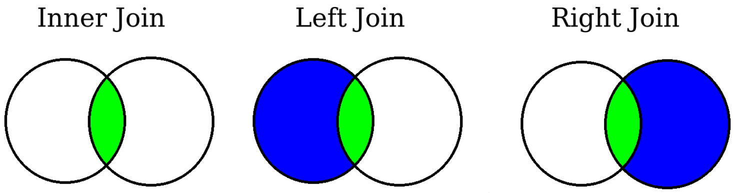
- What is the result for the following SQL query?

```
SELECT CourseNo FROM STUDY
WHERE StudentID=111
EXCEPT
SELECT StudentID FROM STUDY
WHERE CourseNo='ECON2102';
```

ERROR MESSAGE

Join Operations

- When we want to retrieve data from *more than one relations*, we often need to use **join** operations.
- **Inner Join**: tuples are included in the result only if there is at least one matching in both relations.
- **Left/Right Join**: all tuples of the left/right table are included in the result, even if there are no matches in the relations.



Australian National University

Inner Join – Example

COURSE		
No	Cname	Unit
COMP2400	Relational Databases	6
BUSN2011	Management Accounting	6
ECON2102	Macroeconomics	6

ENROL			
StudentID	CourseNo	Semester	Status
111	BUSN2011	2016 S1	active
222	COMP2400	2016 S1	active
111	COMP2400	2016 S2	active

- What is the result for the following INNER JOIN statement?

```
SELECT COURSE.No
FROM COURSE INNER JOIN ENROL ON COURSE.No=ENROL.CourseNo;
```

COURSE			ENROL			
No	Cname	Unit	StudentID	CourseNo	Semester	Status
COMP2400	Relational Databases	6	222	COMP2400	2016 S1	active
COMP2400	Relational Databases	6	111	COMP2400	2016 S2	active
BUSN2011	Management Accounting	6	111	BUSN2011	2016 S1	active

Australian National University

Inner Join – Example

Course		
No	Cname	Unit
COMP2400	Relational Databases	6
BUSN2011	Management Accounting	6
ECON2102	Macroeconomics	6

Enrol			
StudentID	CourseNo	Semester	Status
111	BUSN2011	2016 S1	active
222	COMP2400	2016 S1	active
111	COMP2400	2016 S2	active

- What is the result for the following INNER JOIN statement?

```
SELECT Course.No
FROM Course INNER JOIN Enrol ON Course.No=Enrol.CourseNo;
```

No
COMP2400
COMP2400
BUSN2011

Australian National University

Left Join – Example

COURSE		
No	Cname	Unit
COMP2400	Relational Databases	6
BUSN2011	Management Accounting	6
ECON2102	Macroeconomics	6

ENROL			
StudentID	CourseNo	Semester	Status
111	BUSN2011	2016 S1	active
222	COMP2400	2016 S1	active
111	COMP2400	2016 S2	active

- What is the result for the following LEFT JOIN statement?

```
SELECT Course.No
FROM COURSE LEFT JOIN ENROL ON COURSE.No=ENROL.CourseNo;
```

COURSE			ENROL			
No	Cname	Unit	StudentID	CourseNo	Semester	Status
COMP2400	Relational Databases	6	222	COMP2400	2016 S1	active
COMP2400	Relational Databases	6	111	COMP2400	2016 S2	active
BUSN2011	Management Accounting	6	111	BUSN2011	2016 S1	active
ECON2102	Macroeconomics	6	NULL	NULL	NULL	NULL

Australian National University

Left Join – Example

Course		
No	Cname	Unit
COMP2400	Relational Databases	6
BUSN2011	Management Accounting	6
ECON2102	Macroeconomics	6

Enrol			
StudentID	CourseNo	Semester	Status
111	BUSN2011	2016 S1	active
222	COMP2400	2016 S1	active
111	COMP2400	2016 S2	active

- What is the result for the following LEFT JOIN statement?

```
SELECT Course.No
FROM Course LEFT JOIN Enrol ON Course.No=Enrol.CourseNo;
```

No
COMP2400
COMP2400
BUSN2011
ECON2102

Natural Join

- A natural join is considered as one kind of inner join.
- In a natural join, two relations are joined implicitly by comparing all attributes of the same names in both relations.
- A natural join retains all the data of the two tables for only the matched rows, without duplication.

Australian National University

Natural Join – Example

COURSE		
CourseNo	Cname	Unit
COMP2400	Relational Databases	6
BUSN2011	Management Accounting	6
ECON2102	Macroeconomics	6

ENROL			
StudentID	CourseNo	Semester	Status
111	BUSN2011	2016 S1	active
222	COMP2400	2016 S1	active
111	COMP2400	2016 S2	active

- What is the result for the following NATURAL JOIN statement?

```
SELECT CourseNo
FROM Course NATURAL JOIN Enrol;
```

	COURSE		ENROL		
CourseNo	Cname	Unit	StudentID	Semester	Status
COMP2400	Relational Databases	6	222	2016 S1	active
COMP2400	Relational Databases	6	111	2016 S2	active
BUSN2011	Management Accounting	6	111	2016 S1	active

Australian National University

Natural Join – Example

COURSE		
CourseNo	Cname	Unit
COMP2400	Relational Databases	6
BUSN2011	Management Accounting	6
ECON2102	Macroeconomics	6

ENROL			
StudentID	CourseNo	Semester	Status
111	BUSN2011	2016 S1	active
222	COMP2400	2016 S1	active
111	COMP2400	2016 S2	active

- What is the result for the following NATURAL JOIN statement?

```
SELECT CourseNo
FROM COURSE NATURAL JOIN ENROL;
```

CourseNo
COMP2400
COMP2400
BUSN2011

Australian National University

Natural Join – Example

COURSE		
No	Cname	Unit
COMP2400	Relational Databases	6
BUSN2011	Management Accounting	6
ECON2102	Macroeconomics	6

ENROL			
StudentID	**CourseNo**	Semester	Status
111	BUSN2011	2016 S1	active
222	COMP2400	2016 S1	active
111	COMP2400	2016 S2	active

- What is the result for the following NATURAL JOIN statement?

```
SELECT *
FROM COURSE NATURAL JOIN ENROL;
```

If there are no matching attributes in two tables for NATURAL JOIN,

```
SELECT *
FROM COURSE, ENROL;
```

Australian National University

Natural Join – Example

COURSE		
CourseNo	Cname	Unit
COMP2400	Relational Databases	6
BUSN2011	Management Accounting	6
ECON2102	Macroeconomics	6

ENROL			
StudentID	**CourseNo**	Semester	Status
111	BUSN2011	2016 S1	active
222	COMP2400	2016 S1	active
111	COMP2400	2016 S2	active

- What is the result for the following NATURAL JOIN statement?

```
SELECT *
FROM COURSE NATURAL JOIN ENROL ON COURSE.CourseNo=ENROL.CourseNo;
```

  ERROR MESSAGE because a NATURAL JOIN **implicitly** compares all attributes of the same names in two table.

- Inner Join, Left Join, Right Join, Natural Join, and Cartesian Product, what's their difference?

Australian National University

Join – More Examples

STUDENT			
StudentID	Name	DoB	Email

COURSE		
No	Cname	Unit

ENROL		
StudentID	CourseNo	Status

- List all information of students who have enrolled in a course with CourseNo='X' and the CourseNo of these courses.
  1. Use SELECT + FROM (Cartesian Product) + WHERE
  2. Use SELECT + FROM (INNER JOIN) + ON
  3. Use SELECT + FROM (INNER JOIN) + ON + WHERE
  4. Use SELECT + FROM (NATURAL JOIN) + WHERE

Australian National University

Join – More Examples

STUDENT			
StudentID	Name	DoB	Email

ENROL		
StudentID	CourseNo	Status

- List all information of students who have enrolled in a course with CourseNo='X' and the CourseNo of these courses.
- (1) Use SELECT + FROM (Cartesian Product) + WHERE

```
SELECT STUDENT.*, ENROL.CourseNo
FROM STUDENT, ENROL
WHERE (STUDENT.StudentID=ENROL.StudentID)
      AND (ENROL.CourseNo = 'X');
```

Join – More Examples

STUDENT			
<u>StudentID</u>	Name	DoB	Email

ENROL		
<u>StudentID</u>	<u>CourseNo</u>	Status

- List all information of students who have enrolled in a course with CourseNo='X' and the CourseNo of these courses.
- (2) Use SELECT + FROM (INNER JOIN) + ON

```
SELECT STUDENT.*, ENROL.CourseNo
FROM STUDENT INNER JOIN ENROL
ON (STUDENT.StudentID=ENROL.StudentID)
    AND (ENROL.CourseNo = 'X');
```

Australian National University

Join – More Examples

STUDENT			
StudentID	Name	DoB	Email

ENROL		
StudentID	CourseNo	Status

- List all information of students who have enrolled in a course with CourseNo='X' and the CourseNo of these courses.
- (3) Use SELECT + FROM (INNER JOIN) + ON + WHERE

```
SELECT STUDENT.*, ENROL.CourseNo
FROM STUDENT INNER JOIN ENROL
ON STUDENT.StudentID=ENROL.StudentID
WHERE ENROL.CourseNo = 'X';
```

Australian National University

Join – More Examples

STUDENT			
StudentID	Name	DoB	Email

ENROL		
StudentID	CourseNo	Status

- List all information of students who have enrolled in a course with CourseNo='X' and the CourseNo of these courses.
- (4) Use SELECT + FROM (NATURAL JOIN) + WHERE

```
SELECT STUDENT.*, ENROL.CourseNo
FROM STUDENT NATURAL JOIN ENROL
WHERE ENROL.CourseNo = 'X';
```

Subqueries

- **Subqueries** can be viewed as temporary tables (usually in conjunction with aliases and renaming, exist only for the query).
- Subqueries can be specified within the `FROM`-clause.
- Subqueries can also be specified within the `WHERE`-clause, e.g.,
  - `IN` *subquery* tests if tuple occurs in the temporary table of the subquery.
  - `EXISTS` *subquery* tests whether the temporary table of the subquery is empty or not.
  - using `ALL`, `SOME` or `ANY` before a subquery makes subqueries usable in comparison formulae (`SOME` and `ANY` are interchangeable).
  - in all these cases the condition involving the subquery can be negated using a preceding `NOT`.

Subqueries IN – Example 1

STUDENT			
StudentID	Name	DoB	Email

ENROL		
StudentID	CourseNo	Status

- List all the information of students who have enrolled in a course with CourseNo='X' and the CourseNo of these courses, we have:

```
SELECT STUDENT.*, ENROL.CourseNo
FROM STUDENT NATURAL JOIN ENROL
WHERE ENROL.CourseNo = 'X';
```

Subqueries IN – Example 1

- List all the information of students who have enrolled in a course *that has less than 10 students enrolled in it* and the CourseNo of these courses.
  - List the CourseNo of the courses in Enrol *that have less than 10 students enrolled*

    ```
    SELECT CourseNo
    FROM Enrol
    GROUP BY CourseNo
    HAVING COUNT(*)<10;
    ```

- List all the information of students who have enrolled in a course with CourseNo='X' and the CourseNo of these courses

```
SELECT Student.*, Enrol.CourseNo
FROM Student NATURAL JOIN Enrol
WHERE Enrol.CourseNo = 'X';
```

Australian National University

Subqueries IN – Example 1

- List all the information of students who have enrolled in a course *that has less than 10 students enrolled* and the CourseNo of these courses.

```
SELECT STUDENT.*, CourseNo
FROM STUDENT NATURAL JOIN ENROL
WHERE CourseNo IN (SELECT CourseNo
                    FROM ENROL
                    GROUP BY CourseNo
                    HAVING COUNT(*)<10);
```

- Does the above query look confusing?

  It is better to distinguish two ENROL tables.

Australian National University

Subqueries IN – Example 1

- List all the information of students who have enrolled in a course *that has less than 10 students enrolled* and the CourseNo of these courses.
    - Use aliases e1 and e2 for ENROL to distinguish two ENROL tables.

```
SELECT STUDENT.*,e1.CourseNo
FROM STUDENT NATURAL JOIN ENROL e1
WHERE e1.CourseNo IN (SELECT e2.CourseNo
                      FROM ENROL e2
                      GROUP BY e2.CourseNo
                      HAVING COUNT(*)<10);
```

Australian National University

Subqueries IN – Example 1

- List all the information of students who have enrolled in a course *that has less than 10 students enrolled* and the CourseNo of these courses.

```
SELECT STUDENT.*,e1.CourseNo
FROM STUDENT NATURAL JOIN ENROL e1
WHERE e1.CourseNo IN (SELECT e2.CourseNo, COUNT(*)
                      FROM ENROL e2
                      GROUP BY e2.CourseNo
                      HAVING COUNT(*)<10);
```

- Is the above query correct?

  No. `IN` *subquery* tests if tuple occurs in the temporary table of the subquery.

- **expression** IN (**subquery**)

  The right-hand side is a parenthesized subquery, which must return exactly one column. (refer to Subquery Expressions in PostgreSQL Documentation)

Australian National University

Subqueries EXISTS – Example 2

STUDENT	
StudentID	Name
111	Tom
222	Emily
333	John

ENROL		
StudentID	CourseNo	Semester
111	BUSN2011	2016 S1
222	COMP2400	2016 S1
111	COMP2400	2016 S2

- Count the number of students who have enrolled in at least one course?

```
SELECT COUNT(*)
FROM STUDENT s
WHERE EXISTS (SELECT *
             FROM ENROL e
             WHERE s.StudentID=e.StudentID);
```

1st tuple of STUDENT, EXISTS

StudentID	CourseNo	Semester
111	BUSN2011	2016 S1
111	COMP2400	2016 S2

2st tuple of STUDENT, EXISTS

StudentID	CourseNo	Semester
222	COMP2400	2016 S1

- **The above query (returning 2) is correct!**

Australian National University

Subqueries EXISTS – Example 2

STUDENT	
StudentID	Name
111	Tom
222	Emily
333	John

ENROL		
StudentID	CourseNo	Semester
111	BUSN2011	2016 S1
222	COMP2400	2016 S1
111	COMP2400	2016 S2

- Count the number of students who have enrolled in at least one course?

```
SELECT COUNT(*)
FROM ENROL e
WHERE EXISTS (SELECT *
              FROM STUDENT s
              WHERE e.StudentID=s.StudentID);
```

```
1st tuple in ENROL, EXISTS
2nd tuple in ENROL, EXISTS
3rd tuple in ENROL, EXISTS
```

StudentID	Name
111	Tom

StudentID	Name
222	Emily

StudentID	Name
111	Tom

- **The above query (returning 3 instead of 2) is incorrect!**

Australian National University

Subqueries EXISTS – Example 2

STUDENT	
StudentID	Name
111	Tom
222	Emily
333	John

ENROL		
StudentID	CourseNo	Semester
111	BUSN2011	2016 S1
222	COMP2400	2016 S1
111	COMP2400	2016 S2

- Count the number of students who have enrolled in at least one course?

```
SELECT COUNT(*)
FROM STUDENT s
WHERE EXISTS (SELECT *
             FROM ENROL e
             WHERE s.StudentID=e.StudentID);
```

```
SELECT COUNT(*)
FROM STUDENT s
WHERE EXISTS (SELECT StudentID
             FROM ENROL e
             WHERE s.StudentID=e.StudentID);
```

- **Both queries are correct!** EXISTS *subquery* tests whether the temporary table of the subquery is empty or not.

Australian National University

Using Cartesian Product – Example 2

STUDENT	
StudentID	Name
111	Tom
222	Emily
333	John

ENROL		
StudentID	CourseNo	Semester
111	BUSN2011	2016 S1
222	COMP2400	2016 S1
111	COMP2400	2016 S2

- Count the number of students who have enrolled in at least one course?

```
SELECT COUNT(*)
FROM STUDENT, ENROL
WHERE STUDENT.StudentID=ENROL.StudentID;
```

STUDENT		ENROL		
StudentID	Name	StudentID	CourseNo	Semester
111	Tom	111	BUSN2011	2016 S1
111	Tom	111	COMP2400	2016 S2
222	Emily	222	COMP2400	2016 S1

- **The above query is incorrect!**
  We should use COUNT(DISTINCT StudentID) instead of COUNT(*).

Australian National University

Using INNER JOIN – Example 2

STUDENT	
StudentID	Name
111	Tom
222	Emily
333	John

ENROL		
StudentID	CourseNo	Semester
111	BUSN2011	2016 S1
222	COMP2400	2016 S1
111	COMP2400	2016 S2

- Count the number of students who have enrolled in at least one course?

```
SELECT COUNT(*)
FROM STUDENT s INNER JOIN ENROL e
ON s.StudentID=e.StudentID;
```

s		e		
StudentID	Name	StudentID	CourseNo	Semester
111	Tom	111	BUSN2011	2016 S1
111	Tom	111	COMP2400	2016 S2
222	Emily	222	COMP2400	2016 S1

- **The above query is incorrect!**
  We should use COUNT(DISTINCT StudentID) instead of COUNT(*).

Using NATURAL JOIN – Example 2

STUDENT	
StudentID	Name
111	Tom
222	Emily
333	John

ENROL		
StudentID	CourseNo	Semester
111	BUSN2011	2016 S1
222	COMP2400	2016 S1
111	COMP2400	2016 S2

- Count the number of students who have enrolled in at least one course?

```
SELECT COUNT(*)
FROM STUDENT NATURAL JOIN ENROL;
```

	STUDENT	ENROL	
StudentID	Name	CourseNo	Semester
111	Tom	BUSN2011	2016 S1
111	Tom	COMP2400	2016 S2
222	Emily	COMP2400	2016 S1

- **The above query is incorrect!**
  We should use COUNT(DISTINCT StudentID) instead of COUNT(*).

A Simple Solution – Example 2

STUDENT	
StudentID	Name
111	Tom
222	Emily
333	John

ENROL		
StudentID	CourseNo	Semester
111	BUSN2011	2016 S1
222	COMP2400	2016 S1
111	COMP2400	2016 S2

- Count the number of students who have enrolled in at least one course?

```
SELECT COUNT(DISTINCT StudentID)
FROM ENROL;
```

- **The above query is correct!**

Subqueries – Example 3

- List the courses that have the largest number of students enrolled in Semester 2 2016
  - List the CourseNo and the corresponding number of students enrolled for all courses in Semester 2 2016

```
SELECT CourseNo, COUNT(*) AS NoOfStudents
FROM ENROL
WHERE Semester = '2016 S2'
GROUP BY CourseNo;
```

  - List **the largest number of students enrolled** in a course in Semester 2 2016

```
SELECT MAX(NoOfStudents)
FROM (SELECT CourseNo, COUNT(*) AS NoOfStudents
      FROM ENROL
      WHERE Semester = '2016 S2'
      GROUP BY CourseNo);
```

Australian National University

Subqueries – Combination and Aliases – Example 3

- List the courses that have **the largest number of students enrolled** in Semester 2 2016

```
SELECT CourseNo
FROM (SELECT CourseNo, COUNT(*) AS NoOfStudents
      FROM ENROL
      WHERE Semester = '2016 S2'
      GROUP BY CourseNo)
WHERE NoOfStudents =
            (SELECT MAX(NoOfStudents)
            FROM (SELECT CourseNo, COUNT(*) AS NoOfStudents
                  FROM ENROL
                  WHERE Semester = '2016 S2'
                  GROUP BY CourseNo));
```

- **ERROR**: Subqueries specifying a derived table must be enclosed in parentheses and must be assigned a table alias name.

Subqueries – Combination and Aliases – Example 3

- List the courses that have **the largest number of students enrolled** in Semester 2 2016

```
SELECT e.CourseNo
FROM (SELECT e1.CourseNo, COUNT(*) AS NoOfStudents
      FROM ENROL e1
      WHERE e1.Semester = '2016 S2'
      GROUP BY e1.CourseNo) e
WHERE e.NoOfStudents =
            (SELECT MAX(e2.NoOfStudents)
            FROM (SELECT e1.CourseNo, COUNT(*) AS NoOfStudents
                  FROM ENROL e1
                  WHERE e1.Semester = '2016 S2'
                  GROUP BY e1.CourseNo) e2);
```

- Which alias(es) are essential in the above query?
  The aliases e and e2 are essential but e1 is not.

Subqueries – Use “With” – Example 3

- List the courses that have **the largest number of students enrolled** in Semester 2 2016

  Use “WITH” to break down complicated queries into simpler parts.[1]

```
WITH Sem2Students AS
     (SELECT e1.CourseNo, COUNT(*) AS NoOfStudents
      FROM ENROL e1
      WHERE e1.Semester = '2016 S2'
      GROUP BY e1.CourseNo)
SELECT e.CourseNo
FROM Sem2Students e
WHERE e.NoOfStudents =
             (SELECT MAX(e2.NoOfStudents)
              FROM Sem2Students e2);
```

- Which alias(es) are essential in the above query?
  None of the aliases e, e1 and e2 are essential.

[1] https://www.postgresql.org/docs/current/static/queries-with.html

Australian National University

Subqueries – Result – Example 3

- List the courses that have **the largest number of students enrolled** in Semester 2 2016

Input:

ENROL		
StudentID	CourseNo	Semester
111	BUSN2011	2016 S2
111	COMP1100	2016 S2
111	COMP2400	2016 S2
111	ECON2102	2016 S2
222	BUSN2011	2016 S2
222	COMP2400	2016 S2
333	BUSN2011	2016 S2
333	COMP2400	2016 S2
333	ECON2102	2016 S2

Output:

CourseNo
COMP2400
BUSN2011

Subqueries – Example 4

- List all students' IDs and names who are under-enrolled ($<$ 4 courses) in Semester 2 2016.
  - List the students' IDs and the corresponding number of enrolled courses in Semester 2 2016

```
SELECT e.StudentID, COUNT(*) AS NoOfEnrols
FROM ENROL e
WHERE e.Semester = '2016 S2'
GROUP BY e.StudentID;
```

Australian National University

Subqueries – Example 4

- List all students' IDs and names who are under-enrolled ($<$ 4 courses) in Semester 2 2016.

```
SELECT s.StudentID, s.Name
FROM (SELECT e.StudentID, COUNT(*) AS NoOfEnrols
      FROM Enrol e
      WHERE e.Semester = '2016 S2'
      GROUP BY e.StudentID) ne INNER JOIN Student s
ON (s.StudentID = ne.StudentID) AND (ne.NoOfEnrols < 4);
```

Use "With"

```
WITH StudEnrols AS (
      SELECT e.StudentID, COUNT(*) AS NoOfEnrols
      FROM Enrol e
      WHERE e.Semester = '2016 S2'
      GROUP BY e.StudentID)
SELECT s.StudentID, s.Name
FROM Student s INNER JOIN StudEnrols ne
ON (s.StudentID = ne.StudentID) AND (ne.NoOfEnrols < 4);
```

Australian National University

Subqueries - The Query Result - Example 4

- List all students' IDs and names who are under-enrolled ($<$ 4 courses) in Semester 2 2016.

ENROL		
StudentID	CourseNo	Semester
111	BUSN2011	2016 S2
111	COMP1100	2016 S2
111	COMP2400	2016 S2
111	ECON2102	2016 S2
222	BUSN2011	2016 S2
222	COMP2400	2016 S2
333	BUSN2011	2016 S2
333	COMP2400	2016 S2
333	ECON2102	2016 S2

Result:

StudentID	Name
222	Emily
333	John

STUDENT	
StudentID	Name
111	Tom
222	Emily
333	John

Australian National University

Subqueries – What About The Following Scenario? – Example 4

- List all students' IDs and names who are under-enrolled ($<$ 4 courses) in Semester 2 2016.

ENROL		
StudentID	CourseNo	Semester
111	BUSN2011	2016 S2
111	COMP1100	2016 S2
111	COMP2400	2016 S2
111	ECON2102	2016 S2
222	BUSN2011	2016 S2
222	COMP2400	2016 S2
333	BUSN2011	2016 S2
333	COMP2400	2016 S2
333	ECON2102	2016 S2

Result (still correct?):

StudentID	Name
222	Emily
333	John

STUDENT	
StudentID	Name
111	Tom
222	Emily
333	John
444	Ana

Australian National University

Subqueries – Use LEFT/RIGHT JOIN? – Example 4

- List all students' IDs and names who are under-enrolled ($<$ 4 courses) in Semester 2 2016.

```
SELECT s.StudentID, s.Name
FROM (SELECT e.StudentID, COUNT(*) AS NoOfEnrols
      FROM ENROL e
      WHERE e.Semester = '2016 S2'
      GROUP BY e.StudentID) ne RIGHT JOIN STUDENT s
ON (s.StudentID = ne.StudentID) AND (ne.NoOfEnrols < 4);
```

```
WITH StudEnrols AS (
      SELECT e.StudentID, COUNT(*) AS NoOfEnrols
      FROM ENROL e
      WHERE e.Semester = '2016 S2'
      GROUP BY e.StudentID)
SELECT s.StudentID, s.Name
FROM STUDENT s LEFT JOIN StudEnrols ne
ON (s.StudentID = ne.StudentID) AND (ne.NoOfEnrols < 4);
```

Australian National University

Subqueries – Using LEFT/RIGHT JOIN Is Still Incorrect! – Example 4

ENROL

StudentID	CourseNo	Semester
111	BUSN2011	2016 S2
111	COMP1100	2016 S2
111	COMP2400	2016 S2
111	ECON2102	2016 S2
222	BUSN2011	2016 S2
222	COMP2400	2016 S2
333	BUSN2011	2016 S2
333	COMP2400	2016 S2
333	ECON2102	2016 S2

STUDENT

StudentID	Name
111	Tom
222	Emily
333	John
444	Ana

Result (still incorrect?):

StudentID	Name
111	Tom
222	Emily
333	John
444	Ana

- The reason why “111, Tom” is incorrectly included in the final result is due to “Query Processing and Optimisation”, which will be discussed in Week 8. We will revisit this issue in Week 8.

Australian National University

Subqueries – Use Set Operations – Example 4

- List all students' IDs and names who are under-enrolled ($<$ 4 courses) in Semester 2 2016.

  The set of all students EXCEPT the set of students enrolled in at least 4 courses in Semester 2 2016.

```
SELECT s.StudentID, s.Name
FROM (SELECT StudentID
      FROM STUDENT
      EXCEPT
      SELECT e.StudentID
      FROM ENROL e
      WHERE e.Semester = '2016 S2'
      GROUP BY e.StudentID
      HAVING COUNT(*) > 3) e4 INNER JOIN Student s
ON (e4.StudentID = s.StudentID);
```

Australian National University

Subqueries – Using Set Operations Works – Example 4

- List all students' IDs and names who are under-enrolled ($<$ 4 courses) in Semester 2 2016.

ENROL

StudentID	CourseNo	Semester
111	BUSN2011	2016 S2
111	COMP1100	2016 S2
111	COMP2400	2016 S2
111	ECON2102	2016 S2
222	BUSN2011	2016 S2
222	COMP2400	2016 S2
333	BUSN2011	2016 S2
333	COMP2400	2016 S2
333	ECON2102	2016 S2

Result:

StudentID	Name
222	Emily
333	John
444	Ana

STUDENT

StudentID	Name
111	Tom
222	Emily
333	John
444	Ana