Week 3 Workshop

Housekeeping

1. Thank you again for providing us with your valuable feedback!

Housekeeping

1. Thank you again for providing us with your valuable feedback!
2. Refer to the post in Wattle News Forum for makeup information for the CECS teaching pause.

Housekeeping

1. Thank you again for providing us with your valuable feedback!
2. Refer to the post in Wattle News Forum for makeup information for the CECS teaching pause.
3. Assessment on SQL (Assignment 1) will be available on Wattle at 11:59pm on Aug 20 (Friday) and due at 11:59pm on Sep 3 (Friday).

Housekeeping

1. Thank you again for providing us with your valuable feedback!
2. Refer to the post in Wattle News Forum for makeup information for the CECS teaching pause.
3. Assessment on SQL (Assignment 1) will be available on Wattle at 11:59pm on Aug 20 (Friday) and due at 11:59pm on Sep 3 (Friday).
    - This assessment should be done individually and no group work is allowed.
    - You should not post any solutions/results/ideas/interpretations related to assessment items (including assignments, quizzes, tests) on the Wattle discussion forum.
    - Additional drop-in sessions will be available in Week 5 if you need any further clarification for this assignment.

Housekeeping

1. Thank you again for providing us with your valuable feedback!
2. Refer to the post in Wattle News Forum for makeup information for the CECS teaching pause.
3. Assessment on SQL (Assignment 1) will be available on Wattle at 11:59pm on Aug 20 (Friday) and due at 11:59pm on Sep 3 (Friday).
   - This assessment should be done individually and no group work is allowed.
   - You should not post any solutions/results/ideas/interpretations related to assessment items (including assignments, quizzes, tests) on the Wattle discussion forum.
   - Additional drop-in sessions will be available in Week 5 if you need any further clarification for this assignment.
4. Here are our course representatives for COMP2400/6240 in S2 2021
   - Julian Crosby, Julian.Crosby@anu.edu.au
   - Yixin Liu, Yixin.Liu@anu.edu.au
   - Navdeep Gill, u7275100@anu.edu.au
   - Xueqi Lin, Xueqi.Lin@anu.edu.au

Outline

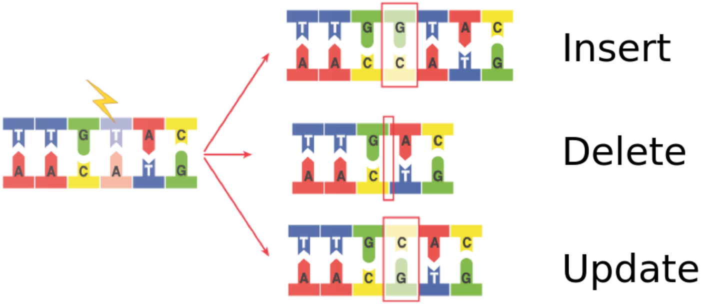
1. **Insert, Update, Delete Statements**
   v.s. **Relational Database State**

2. **Select Statements**

3. **A Bunch of Tables**

Insert, Update, Delete Statements

- **Insert, Delete, Update Statements**
  v.s. **Relational Database State**



Relational Database State – Example

- A **relational database state** of *S* is a set of relations such that
  - there is just one relation for each relation schema in *S*, and
  - all the relations satisfy the integrity constraints *IC*.

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

COURSE

No	Cname	Unit
COMP1130	Introduction to Advanced Computing I	6
COMP2400	Relational Databases	6

ENROL

StudentID	CourseNo	Semester	Status	EnrolDate
456	COMP2400	2016 S2	active	25/05/2016
458	COMP1130	2016 S1	active	20/02/2016
459	COMP2400	2016 S2	active	11/06/2016

Insert Statement – 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');
  ```

Insert Statement – 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 Statement – 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);
  ```

Insert Statement – 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 Statement – 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');
  ```

Insert Statement – 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.

Update Statement – 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;
```

Update Statement – 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

Delete Statement – 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;
```

Delete Statement – 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 Statement – 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;
```

Australian National University

Delete Statement – 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;
```

STUDENT			
StudentID	Name	DoB	Email

Delete Statement – 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;
```

STUDENT			
StudentID	Name	DoB	Email

```
DROP TABLE STUDENT;
```

The Table STUDENT is deleted.

Delete Statement – 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;
```

STUDENT			
StudentID	Name	DoB	Email

```
DROP TABLE STUDENT;
```

The Table STUDENT is deleted.

- Note the difference between the Delete and Drop Table statements.

Australian National University

Delete Statement – 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

- What will happen if we execute the following statement?

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

Delete Statement – 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

- What will happen if we execute the following statement?

```
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.

Delete Statement – 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 Statement – 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

- What will happen if we execute the following statement?

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

Delete Statement – 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

- What will happen if we execute the following statement?

```
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

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];
```

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';
```

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 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';
```

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';
```

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

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
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';
```

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
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

Select + Group By

- `GROUP BY` *attribute_list* groups tuples for each value combination in the attribute_list.

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

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.

Aggregate Functions – Example

- List the total number of courses, the sum of the units of courses, the minimum unit in COURSE

COURSE		
No	Cname	Unit
COMP1130	Introduction to Advanced Computing I	6
COMP2400	Relational Databases	6
COMP3600	Algorithms	4

Aggregate Functions – Example

- List the total number of courses, the sum of the units of courses, the minimum unit in COURSE

COURSE		
No	Cname	Unit
COMP1130	Introduction to Advanced Computing I	6
COMP2400	Relational Databases	6
COMP3600	Algorithms	4

```
SELECT COUNT(unit), MAX(unit) FROM COURSE;
```

Aggregate Functions – Example

- List the total number of courses, the sum of the units of courses, the minimum unit in COURSE

COURSE		
No	Cname	Unit
COMP1130	Introduction to Advanced Computing I	6
COMP2400	Relational Databases	6
COMP3600	Algorithms	4

```
SELECT COUNT(unit), MAX(unit) FROM COURSE;
```

- The query result will be:

COUNT	MAX
3	6

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 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;
```

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

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

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

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;
```

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

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;
```

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

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, CourseNo
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, CourseNo
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 *
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 *
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 COUNT(*)
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 COUNT(*)
FROM Study
Group By StudentID;
```

COUNT
3
1
2

Australian National University

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 CourseNo?

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

Australian National University

Select + Group By – 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 would happen for the following SELECT + Group By CourseNo?

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

Australian National University

Select + Group By – 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 CourseNo?

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

Select + Group By – 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 CourseNo?

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

CourseNo	COUNT
BUSN2011	2
COMP2400	2
ECON2102	1
STAT2001	1

Select + Group By – 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 CourseNo?

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

Australian National University

Select + Group By – 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 CourseNo?

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

Error Message.

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;
```

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

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;
```

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

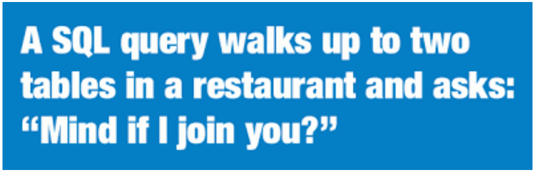
- What is the result for the following SELECT + Group By + Having?

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

CourseNo
BUSN2011
COMP2400

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.

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';
```

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

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

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;
```

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

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
BUSN2011
ECON2102

Set Operations

STUDY		
<u>StudentID</u>	<u>CourseNo</u>	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';
```

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';
```

CourseNo
COMP2400
BUSN2011
ECON2102

EXCEPT

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

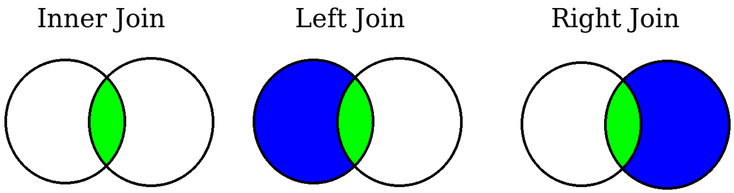
- What is the result for the following SQL query?

```
SELECT 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.



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 would happen for the following INNER JOIN statement?

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

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 would happen for the following INNER JOIN statement?

```
SELECT ...
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

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

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

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 would happen for the following LEFT JOIN statement?

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

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 would happen for the following LEFT JOIN statement?

```
SELECT ...
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

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

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.

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 would happen for the following NATURAL JOIN statement?

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

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 would happen for the following NATURAL JOIN statement?

```
SELECT ...
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;
```

	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

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

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;
```

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;
```

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.

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

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

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			
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.
- (2) Use SELECT + FROM (INNER JOIN) + ON

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.
- (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');
```

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

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';
```

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

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

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, we have:

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

- Now if we want to list all information of students who have enrolled in a course *that has less than 10 students enrolled* and the CourseNo of these courses.

Subqueries IN – Example

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

Subqueries IN – Example

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

Subqueries IN – Example

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

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

Subqueries IN – Example

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

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

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

Subqueries IN – Example

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

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

  - List all 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';
    ```

Subqueries IN – Example

- List all 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
                      FROM ENROL e2
                      GROUP BY e2.CourseNo
                      HAVING COUNT(*)<10);
```

Subqueries IN – Example

- List all 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
                      FROM ENROL e2
                      GROUP BY e2.CourseNo
                      HAVING COUNT(*)<10);
```

- Why do we use aliases e1 and e2 for ENROL?

Subqueries IN – Example

- List all 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
                      FROM ENROL e2
                      GROUP BY e2.CourseNo
                      HAVING COUNT(*)<10);
```

- Why do we use aliases e1 and e2 for ENROL?

  Distinguish two ENROL tables.

Subqueries IN – Example

- List all 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);
```

Subqueries IN – Example

- List all 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?

Subqueries IN – Example

- List all 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.

Subqueries EXISTS – Example

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?

Subqueries EXISTS – Example

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);
```

Subqueries EXISTS – Example

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!**

Subqueries EXISTS – Example

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);
```

Subqueries EXISTS – Example

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!**

Subqueries EXISTS – Example

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);
```

Subqueries EXISTS – Example

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.

Using Cartesian Product – Same Example

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;
```

Using Cartesian Product – Same Example

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

Using Cartesian Product – Same Example

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!**

Using Cartesian Product – Same Example

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 – Same Example

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;
```

Using INNER JOIN – Same Example

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

Using INNER JOIN – Same Example

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!**

Using INNER JOIN – Same Example

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(*).

Australian National University

Using NATURAL JOIN – Same Example

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;
```

Using NATURAL JOIN – Same Example

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

Using NATURAL JOIN – Same Example

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!**

Australian National University

Using NATURAL JOIN – Same Example

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 – Same Example

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;
```

A Simple Solution – Same Example

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!**

A Simple Solution – Same Example

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!**
  - Is this the shortest query to answer the above question? Refer to the last slide on "[Credit Cookie] The Shortest Code/Program?".

Subqueries – More Examples

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

Subqueries – More Examples

- 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

Australian National University

Subqueries – More Examples

- 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;
```

Subqueries – More Examples

- 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

Subqueries – More Examples

- 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);
    ```

Subqueries – More Complicated

- 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);
```

Subqueries – More Complicated

- 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]

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

Subqueries – More Complicated

- 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);
```

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

Subqueries – More Complicated

- 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

Subqueries – More Complicated

- 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 – More Examples

- List all the courses that have more students enrolled than at least one other course in Semester 2 2016

Subqueries – More Examples

- List all the courses that have more students enrolled than at least one other course 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
      > ANY (SELECT e2.NoOfStudents
            FROM (SELECT e1.CourseNo, COUNT(*) AS NoOfStudents
                 FROM ENROL e1
                 WHERE e1.Semester = '2016 S2'
                 GROUP BY e1.CourseNo) e2);
```

Subqueries – More Examples

- List all the courses that have more students enrolled than at least one other course in Semester 2 2016

Subqueries – More Examples

- List all the courses that have more students enrolled than at least one other course in Semester 2 2016

```
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
            > ANY (SELECT e2.NoOfStudents
                   FROM Sem2Students e2);
```

Subqueries – More Complicated

- List all the courses that have more students enrolled than at least one other course 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

Subqueries – More Complicated

- List all the courses that have more students enrolled than at least one other course 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
ECON2102

Subqueries – More Examples

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

Subqueries – More Examples

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

Subqueries – More Examples

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

Subqueries – More Examples

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

Subqueries – More Examples

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

```
SELECT s.StudentID, s.Name, ne.NoOfEnrols
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);
```

Subqueries – More Examples

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

```
SELECT s.StudentID, s.Name, ne.NoOfEnrols
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);
```

```
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, ne.NoOfEnrols
FROM STUDENT s INNER JOIN StudEnrols ne
ON (s.StudentID = ne.StudentID) AND (ne.NoOfEnrols < 4);
```

Subqueries – More Examples

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

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

Subqueries – More Examples

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

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

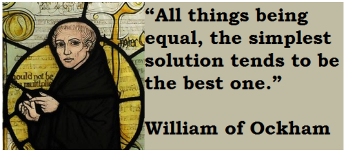
Result:

StudentID	Name	NoOfEnrols
222	Emily	2
333	John	3

STUDENT	
StudentID	Name
111	Tom
222	Emily
333	John

[Credit Cookie] The Shortest Code/Program?

- Occam's razor is the problem-solving principle that "entities should not be multiplied beyond necessity".

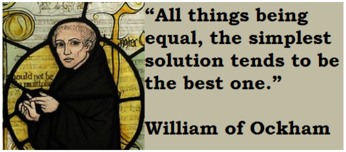


"All things being equal, the simplest solution tends to be the best one."

William of Ockham

[Credit Cookie] The Shortest Code/Program?

- Occam's razor is the problem-solving principle that "entities should not be multiplied beyond necessity".


"All things being equal, the simplest solution tends to be the best one."
William of Ockham

- The minimum description length of a data set (i.e., Kolmogorov complexity) cannot be computed.

https://en.wikipedia.org/wiki/Andrey_Kolmogorov