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. As on

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. As on
   - You should not post any solutions/results/ideas/interpretations related to assessment items (including assignm
     Wattle discussion forum.
   - Additional drop-in sessions will be availab
     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. As on
   - https://eduassistpro.github.i
   - You should not post any solutions/results/ideas/interpretations related to assessment items (including assignm
     Wattle discussion forum.
   - Additional drop-in sessions will be availab
     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.
2. S
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.

459	Fran	11/09/1987	frankk@gmail.com

COURSE

No	Cname	
COMP1130	Introduction to Advanced C	
COMP2400	Relational Datab	

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));
```

- Wi
- IN
  VA

Insert Statement – Example


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

- Wi
- IN
  VA

Yes.

Insert Statement – Example


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

- Wi
- IN
  VA

  Yes.
- INSERT INTO STUDENT(StudentID)
  VALUES (450);

Insert Statement – Example


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

- Wi
- IN
  VA

  Yes.
- INSERT INTO STUDENT(StudentID)
  VALUES (450);

  Yes. The values for Name, DoB and Email will be

Insert Statement – Example


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

- Wi
- IN
  VA

Yes.

- `INSERT INTO STUDENT(StudentID)`
  `VALUES (450);`

Yes. The values for Name, DoB and Email will be

- `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));
```

- Wi
- IN
  VA

  Yes.
- `INSERT INTO Student(StudentID)`
  `VALUES (450);`

  Yes. The values for Name, DoB and Email will be
- `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

- W

```
UPDATE STUDENT SET Name='Tom Lee', Em                    '
WHERE StudentID=456;
```

Update Statement – Example

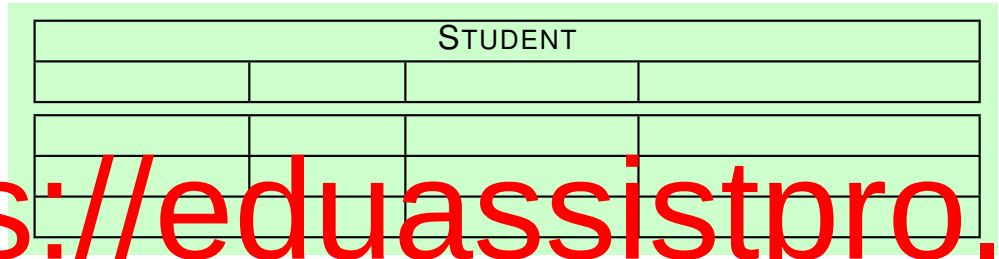
STUDENT			
StudentID	Name	DoB	Email

- W

```
UPDATE STUDENT SET Name='Tom Lee', Em                    '
WHERE StudentID=456;
```


STUDENT			
StudentID	Name	DoB	
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



- W

```
DELETE FROM STUDENT WHERE StudentID=4
```

Delete Statement – Example

STUDENT			

- W

```
DELETE FROM STUDENT WHERE StudentID=4
```


STUDENT			
StudentID	Name	DoB	
458	Peter	23/05/19	
459	Fran	11/09/1987	frankk@gmail.com

Delete Statement – Example

STUDENT

StudentID	Name	DoB	Email

- w

```
DELETE FROM STUDENT;
```

Delete Statement – Example

STUDENT			
StudentID	Name	DoB	Email

-

```
DELETE FROM STUDENT;
```

STUDENT			
StudentID	Name		

Delete Statement – Example

STUDENT			
StudentID	Name	DoB	Email

- w

```
DELETE FROM STUDENT;
```

STUDENT			
StudentID	Name		


```
DROP TABLE STUDENT;
```

The Table STUDENT is deleted.

Delete Statement – Example

STUDENT			
StudentID	Name	DoB	Email

- w

```
DELETE FROM STUDENT;
```

STUDENT			
StudentID	Name		


```
DROP TABLE STUDENT;
```

The Table STUDENT is deleted.

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

Delete Statement – Example

- Consider the following foreign key defined on ENROL:

  FOREIGN KEY(StudentID) REFERENCES STUDENT(StudentID)

  ON DELETE NO ACTION

_____	_____	_____		

STUDENT

StudentID	Name	DoB	
456	Tom	25/01/198	
458	Peter	20/02/199	

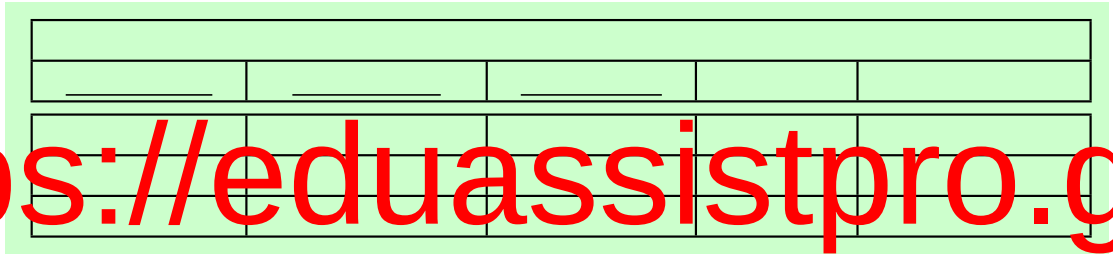
- What will happen if we execute the following state

```
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



STUDENT

StudentID	Name	DoB	
456	Tom	25/01/198	
458	Peter	20/02/199	

- What will happen if we execute the following state

```
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

STUDENT

StudentID	Name	DoB	
456	Tom	25/01/198	
458	Peter	20/02/199	

Delete Statement – Example

- Consider the following foreign key defined on ENROL:

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

_____	_____	_____		

STUDENT

StudentID	Name	DoB	
456	Tom	25/01/198	
458	Peter	20/02/199	

- What will happen if we execute the following state

```
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

_____	_____	_____		

STUDENT

StudentID	Name	DoB	
456	Tom	25/01/198	
458	Peter	20/02/199	

- What will happen if we execute the following state

```
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 Statement

- Th

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

Select Statement

STUDENT

StudentID	Name	DoB	Email
456	Tom	23/01/1988	tom@hotmail.com
458	Peter	23/05/1993	peter@gmail.com

- W
s

Select Statement

STUDENT

StudentID	Name	DoB	Email
456	Tom	23/01/1988	tom@hotmail.com
458	Peter	23/05/1993	peter@gmail.com

- W
S

StudentID	Name	DoB	
458	Peter	23/05/19	
459	Fran	11/09/19	

Select Statement

STUDENT			
StudentID	Name	DoB	Email
456	Tom	23/01/1988	tom@hotmail.com
458	Peter	23/05/1993	peter@gmail.com

- W
S

StudentID	Name	DoB	
458	Peter	23/05/19	
459	Fran	11/09/19	


```
SELECT StudentID FROM Student WHER                    m';
```

Select Statement

STUDENT

StudentID	Name	DoB	Email
456	Tom	23/01/1988	tom@hotmail.com
458	Peter	23/05/1993	peter@gmail.com

- W
S

StudentID	Name	DoB	
458	Peter	23/05/19	
459	Fran	11/09/19	


```
SELECT StudentID FROM STUDENT WHER                    m';
```

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

- W
s

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

- W
  s

STUDENT

StudentID	Name	DoB	
460	Peter	03/09/19	

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

- W
  S

STUDENT

StudentID	Name	DoB	
460	Peter	03/09/19	

```
SELECT * FROM STUDENT WHERE lower(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

- W
  s

STUDENT			
StudentID	Name	DoB	
460	Peter	03/09/19	

```
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.
- A
int

  - `AVG` returns the average of argument values
  - `MIN` returns the minimum value of the argu
  - `MAX` returns the maximum value of the argu
  - `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.
- A
int
  -
  - `AVG` returns the average of argument values
  - `MIN` returns the minimum value of the argu
  -
  - `MAX` returns the maximum value of the argu
  - `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

		it
COMP3600	Algo	

Aggregate Functions – Example

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

		it
COMP3600	Algo	


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

Aggregate Functions – Example

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

		it
COMP3600	Algo	


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


- The query result will be:

COUNT	MAX
3	6

Select + Group By – Example

STUDY		
StudentID	CourseNo	Hours
111	ECON2102	120
333	BUSN	

- What would happen for the following SELECT + Gr

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

Select + Group By – Example

Group	STUDY		
StudentID	StudentID	CourseNo	Hours
333	333		
	333		

- What would happen for the following SELECT + Gr

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

Select + Group By – Example

Group	STUDY		
StudentID	StudentID	CourseNo	Hours
333			
	333		

- What is the result for the following SELECT + Group

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

Select + Group By – Example

Group	STUDY		
StudentID	StudentID	CourseNo	Hours
333			
	333		

- What is the result for the following SELECT + Group

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

111
222
333

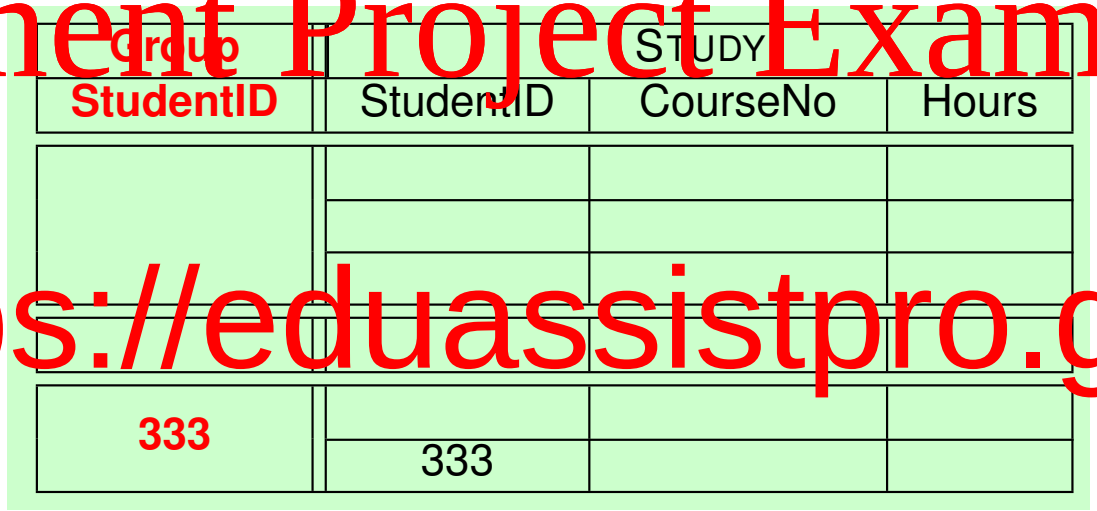
Select + Group By – Example

Group StudentID	STUDY StudentID	CourseNo	Hours
333			
	333		

- What is the result for the following SELECT + Group

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

Select + Group By – Example



- What is the result for the following SELECT + Group

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

	T
111	3
222	1
333	2

Select + Group By – Example

Group

StudentID
333

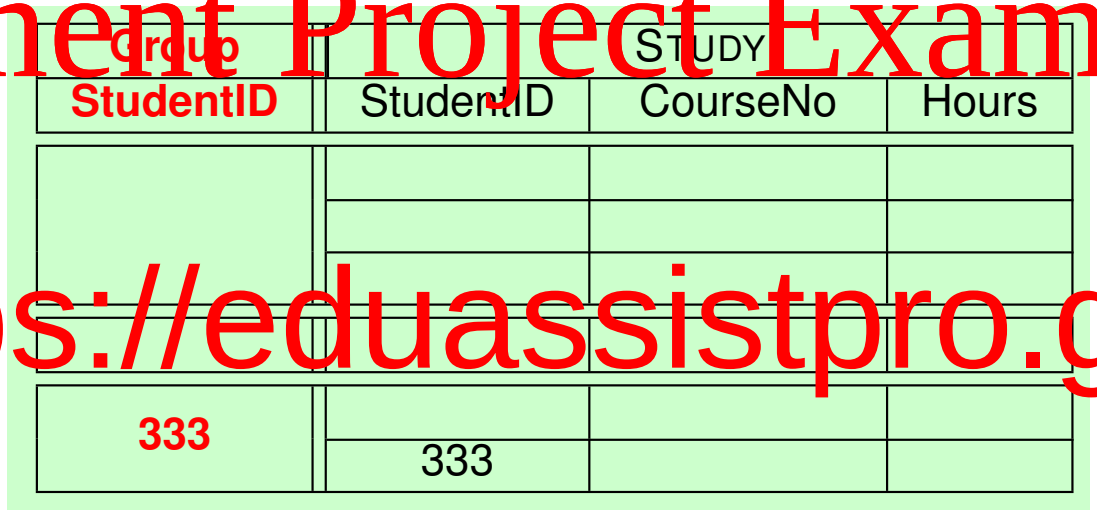
STUDY

StudentID	CourseNo	Hours
333		

- What is the result for the following SELECT + Group

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

Select + Group By – Example



- What is the result for the following SELECT + Group

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

111	120
222	115
333	130

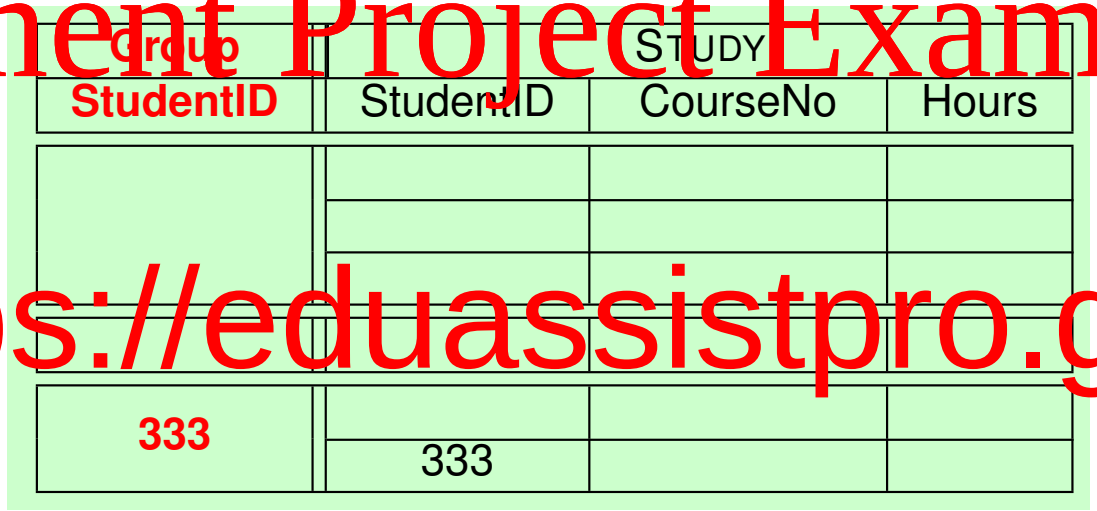
Select + Group By – Example

Group StudentID	STUDY StudentID	CourseNo	Hours
333			
	333		

- What is the result for the following SELECT + Group

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

Select + Group By – Example



- What is the result for the following SELECT + Group

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

	T
111	3
222	1
333	2

Select + Group By – Example

Group StudentID	STUDY		
	StudentID	CourseNo	Hours
333	333	STAT2001	120
	333		

- What is the result for the following SELECT + Group

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

Select + Group By – Example

Group StudentID	STUDY StudentID	CourseNo	Hours
333	333	STAT2001	120
	333		

- What is the result for the following SELECT + Group

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

Error Message.

Select + Group By – Example

Group	STUDY		
StudentID	StudentID	CourseNo	Hours
333	333	STAT2001	120
	333		

- What is the result for the following SELECT + Group

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

Select + Group By – Example

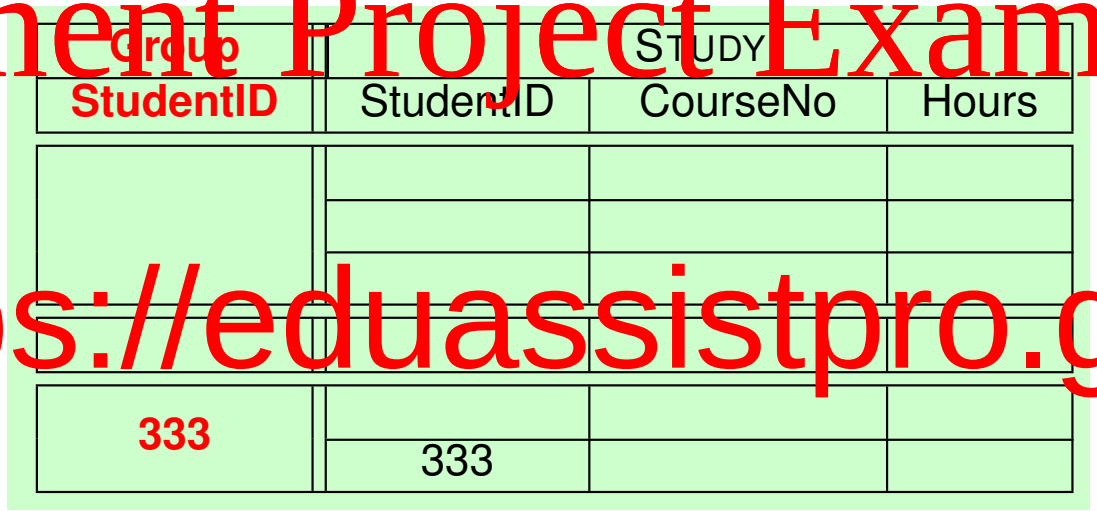
Group	STUDY		
StudentID	StudentID	CourseNo	Hours
333	333	STAT2001	120
	333		

- What is the result for the following SELECT + Group

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

Error Message.

Select + Group By – Example



- What is the result for the following SELECT + Group

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

Select + Group By – Example

Group	STUDY		
StudentID	StudentID	CourseNo	Hours
333			
	333		

- What is the result for the following SELECT + Group

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

3
1
2

Select + Group By – Example

STUDY		
StudentID	CourseNo	Hours
111	ECON2102	120
333	BUSN	

- What would happen for the following SELECT + Gr

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

Select + Group By – Example

Group	STUDY		
CourseNo	StudentID	CourseNo	Hours
ECON2102	111	ECON2102	120
STAT2001	333		

- What would happen for the following SELECT + Gr

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

Select + Group By – Example

Group CourseNo	STUDY StudentID	CourseNo	Hours
	111	BUSN2011	110
STAT2001	333		

- What is the result for the following SELECT + Group

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

Select + Group By – Example

Group CourseNo

	STUDY		
	StudentID	CourseNo	Hours
	111	BUSN2011	110
STAT2001	333		

- What is the result for the following SELECT + Group

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

	T
BUSN2011	2
COMP2400	2
ECON2102	1
STAT2001	1

Select + Group By – Example

Group	STUDY		
CourseNo	StudentID	CourseNo	Hours
ECON2102	111	ECON2102	120
STAT2001	333		

- What is the result for the following SELECT + Group

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

Select + Group By – Example

Group	STUDY		
CourseNo	StudentID	CourseNo	Hours
ECON2102	111	ECON2102	120
STAT2001	333		

- What is the result for the following SELECT + Group

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

Error Message.

Select + Group By + Having – Example

Group	Study		
CourseNo	StudentID	CourseNo	Hours
STAT2001	333		

- What is the result for the following SELECT + Group

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

Select + Group By + Having – Example

Group	STUDY		
CourseNo	StudentID	CourseNo	Hours
STAT2001	333		

- What is the result for the following SELECT + Group

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

CourseNo
BUSN2011

Select + Group By + Having – Example

Group	STUDY		
CourseNo	StudentID	CourseNo	Hours
	111	BUSN2011	110
STAT2001	333		

- What is the result for the following SELECT + Group

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

Select + Group By + Having – Example

Group	STUDY		
CourseNo	StudentID	CourseNo	Hours
	111	BUSN2011	110
STAT2001	333		

- What is the result for the following SELECT + Group

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

CourseNo
BUSN2011
COMP2400

A Bunch of Tables

- A Bunch of Tables

Set Operations

- S
(s INTERSECT

- Set operations result in return of a relation of tuples (n

- 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

- 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

- What is the result for the following SQL query?


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

222

UNION

StudentID
111

Set Operations

STUDY

StudentID	CourseNo	Hours
111	COMP2400	120

- What is the result for the following SQL query?


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

111
222

Set Operations

STUDY

StudentID	CourseNo	Hours
111	COMP2400	120

- What is the result for the following SQL query?


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

Set Operations

STUDY

StudentID	CourseNo	Hours
111	COMP2400	120

- What is the result for the following SQL query?


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

ECON2102

EXCEPT

CourseNo
COMP2400

Set Operations

STUDY

StudentID	CourseNo	Hours
111	COMP2400	120

- What is the result for the following SQL query?


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

BUSN2011
ECON2102

Set Operations

STUDY

StudentID	CourseNo	Hours
111	COMP2400	120

- 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

- What is the result for the following SQL query?


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

ECON2102

EXCEPT

StudentID
111

Set Operations

STUDY

StudentID	CourseNo	Hours
111	COMP2400	120

- 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.
- **In**
  m
- **Le**
  ev

Inner Join – Example

COURSE

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

111	COMP2400	2016 S2	active

- What would happen for the following INNER JOIN

```
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

111	COMP2400	2016 S2	active

- What would happen for the following INNER JOIN


```
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

111	COMP2400	2016 S2	active

- What is the result for the following INNER JOIN stat

```
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	Name	Unit
COMP2400	Relational Databases	6
BUSN2011	Management Accounting	6
ECON2102	Macroeconomics	6

111	COMP2400	2016 S2	active

- What is the result for the following INNER JOIN stat

```
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

111	COMP2400	2016 S2	active

- What would happen for the following LEFT JOIN st

```
SELECT ...
FROM Course LEFT JOIN Enrol ON Cour
```

Left Join – Example

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

111	COMP2400	2016 S2	active

- What would happen for the following LEFT JOIN st


```
SELECT ...
FROM Course LEFT JOIN Enrol ON Cour
```

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

111	COMP2400	2016 S2	active

- What is the result for the following LEFT JOIN state


```
SELECT Course.No
FROM COURSE LEFT JOIN ENROL ON COUR
```

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

111	COMP2400	2016 S2	active

- What is the result for the following LEFT JOIN state


```
SELECT Course.No
FROM Course LEFT JOIN Enrol ON Cour
```

No
COMP2400
COMP2400
BUSN2011
ECON2102

Natural Join

- A n

- In a natural join, two relations are joined implicitly by c
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

111	COMP2400	2016 S2	active

- What would happen for the following NATURAL J

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

Natural Join – Example

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

111	COMP2400	2016 S2	active

- What would happen for the following NATURAL J

```
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

Natural Join – Example

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

111	COMP2400	2016 S2	active

- What is the result for the following NATURAL JOIN


```
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

111	COMP2400	2016 S2	active

- What is the result for the following NATURAL JOIN

```
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

111	COMP2400	2016 S2	active

- What is the result for the following NATURAL JOIN

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

Natural Join – Example

COURSE

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

111	COMP2400	2016 S2	active

- What is the result for the following NATURAL JOIN

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

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

```
SELECT *
FROM Course, Enrol;
```

Natural Join – Example

COURSE

CourseNo	Cname	Unit
COMP2400	Relational Databases	6
BUSN2011	Management Accounting	6

222	COMP2400	2016 S1	active
111	COMP2400		

- What is the result for the following NATURAL JOIN

```
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

222	COMP2400	2016 S1	active
111	COMP2400		

- What is the result for the following NATURAL JOIN

```
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

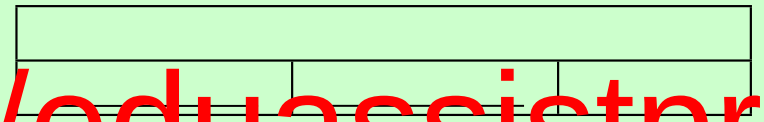
StudentID	Name	DoB	Email

StudentID	Cours	

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

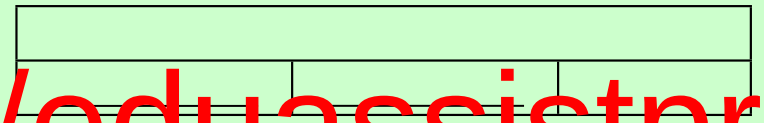
STUDENT			
StudentID	Name	DoB	Email



- Li
  CourseNo='X' and the CourseNo of these course
- (1) Use SELECT + FROM (Cartesian Product) + W

Join – More Examples

STUDENT			
StudentID	Name	DoB	Email

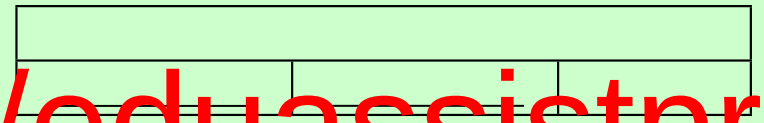


- Li
  CourseNo='X' and the CourseNo of these course
- (1) Use SELECT + FROM (Cartesian Product) + W


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

Join – More Examples

STUDENT			
StudentID	Name	DoB	Email

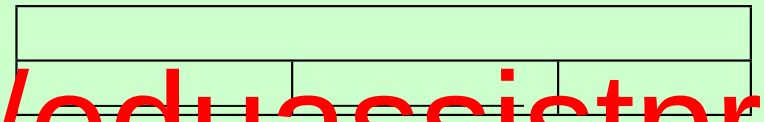


- Li
  CourseNo='X' and the CourseNo of these course
- (2) Use SELECT + FROM (INNER JOIN) + ON

Join – More Examples

STUDENT

StudentID	Name	DoB	Email



- Li
  CourseNo='X' and the CourseNo of these course
- (2) Use SELECT + FROM (INNER JOIN) + ON


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

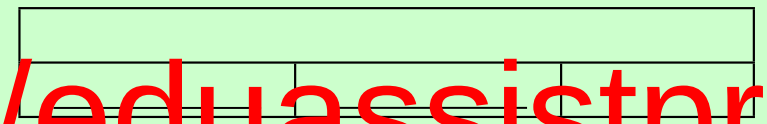
Join – More Examples

STUDENT			
StudentID	Name	DoB	Email

- Li
  CourseNo='X' and the CourseNo of these course
- (3) Use SELECT + FROM (INNER JOIN) + ON + WH

Join – More Examples

STUDENT			
StudentID	Name	DoB	Email



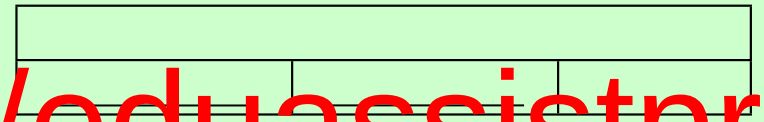
- Li
  CourseNo='X' and the CourseNo of these course
- (3) Use SELECT + FROM (INNER JOIN) + ON + WH


```
SELECT Student.*, Enrol.Cours
FROM Student INNER JOIN Enrol
ON Student.StudentID=Enrol.StudentID
WHERE Enrol.CourseNo = 'X';
```

Join – More Examples

STUDENT

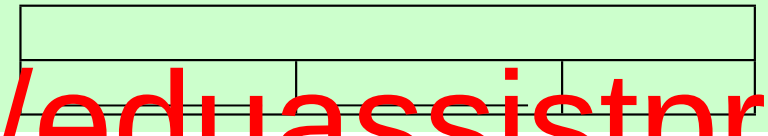
StudentID	Name	DoB	Email



- Li
  CourseNo='X' and the CourseNo of these course
- (4) Use SELECT + FROM (NATURAL JOIN) + WH

Join – More Examples

STUDENT			
StudentID	Name	DoB	Email



- Li
  CourseNo='X' and the CourseNo of these course
- (4) Use SELECT + FROM (NATURAL JOIN) + WH


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

Subqueries

- **Subqueries** can be viewed as temporary tables (usually in conjunction with aliases and renaming, exist only for the query).
- S
- S

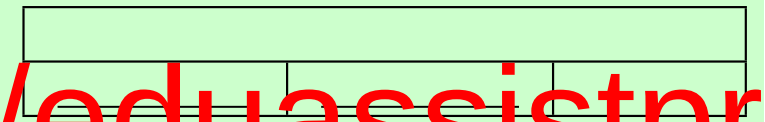
  - `IN` *subquery* tests if tuple occurs in the temporary table of the subquery.
  - `EXISTS` *subquery* tests whether the tem empty or not.

  - 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



- Li
  CourseNo='X' and the CourseNo of these course


```
SELECT STUDENT.*, ENROL.Cours
FROM STUDENT NATURAL JOIN ENR
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.

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.


```
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.


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


- List all information of students who have en
  CourseNo='X' and the CourseNo of these c

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.


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


- List all information of students who have en
  CourseNo='X' and the CourseNo of these c

```
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.


```
FROM Enrol e2
GROUP BY e2
HAVING CO
```

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.


```
FROM ENROL e2
GROUP BY e2
HAVING CO
```


- 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.


```
FROM ENROL e2
GROUP BY e2
HAVING CO
```


- 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.


```
FROM Enrol e2
GROUP BY e2
HAVING CO
```

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.


```
FROM Enrol e2
GROUP BY e2
HAVING CO
```


- 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.


```
FROM Enrol e2
GROUP BY e2
HAVING CO
```


- 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	John
222	Emily
333	John

ENROL

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

- C

Subqueries EXISTS – Example

STUDENT

StudentID	Name
111	John
222	Emily
333	John

ENROL

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

- C


```
WHERE EXISTS (SELECT *
              FROM ENROL e
              WHERE s.StudentID=e
```

Subqueries EXISTS – Example

STUDENT

StudentID	Name
111	John
222	Emily
333	John

ENROL

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

- C


```
WHERE EXISTS (SELECT *
              FROM ENROL e
              WHERE s.StudentID=e
```


1st tuple of STUDENT, EXISTS

		emester
		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	John
222	Emily
333	John

ENROL

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

- C


```
WHERE EXISTS (SELECT *
              FROM STUDENT s
              WHERE e.StudentID=s
```

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

- C


```
WHERE EXISTS (SELECT *
              FROM STUDENT s
              WHERE e.StudentID=s
```


1st tuple in ENROL, EXISTS

2nd tuple in ENROL, EXISTS

3rd tuple in ENROL, EXISTS

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

- C


```
WHERE EXISTS (SELECT *
              FROM ENROL e
              WHERE s.StudentID=e
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

- C


```
WHERE EXISTS (SELECT *
              FROM ENROL e
              WHERE s.StudentID=e
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

- C


```
FROM Student, Enrol
WHERE Student.StudentID=Enrol.
```

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

- C


```
FROM Student, Enrol
WHERE Student.StudentID=Enrol.
```


Student		Enrol		
StudentID	Name	StudentID		
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

- C


```
FROM STUDENT, ENROL
WHERE STUDENT.StudentID=ENROL.
```


STUDENT		ENROL		
StudentID	Name	StudentID		
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

- C


```
FROM Student, Enrol
WHERE Student.StudentID=Enrol.
```


Student		Enrol		
StudentID	Name	StudentID		
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 INNER JOIN – Same Example

Student	
StudentID	Name
111	Tom
222	Emily

Enrol		
StudentID	CourseNo	Semester
111	BUSN2011	2016 S1
222	COMP2400	2016 S1
		16 S2

- C


```
FROM Student s INNER JOIN Enro
ON s.StudentID=e.StudentID;
```

Using INNER JOIN – Same Example

Student	
StudentID	Name
111	Tom
222	Emily

Enrol		
StudentID	CourseNo	Semester
111	BUSN2011	2016 S1
222	COMP2400	2016 S1
		16 S2

- C


```
FROM Student s INNER JOIN Enro
ON s.StudentID=e.StudentID;
```


s				
StudentID	Name	StudentID		
111	Tom	111		
111	Tom	111	COMP2400	2016 S2
222	Emily	222	COMP2400	2016 S1

Using INNER JOIN – Same Example

Student	
StudentID	Name
111	Tom
222	Emily

Enrol		
StudentID	CourseNo	Semester
111	BUSN2011	2016 S1
222	COMP2400	2016 S1
		16 S2

- C

```
FROM Student s INNER JOIN Enro
ON s.StudentID=e.StudentID;
```


s				
StudentID	Name	StudentID		
111	Tom	111		
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

Enrol		
StudentID	CourseNo	Semester
111	BUSN2011	2016 S1
222	COMP2400	2016 S1
		16 S2

- C

```
FROM Student s INNER JOIN Enro
ON s.StudentID=e.StudentID;
```


s				
StudentID	Name	StudentID		
111	Tom	111		
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 – Same Example

STUDENT

StudentID	Name
111	Tom

ENROL

StudentID	CourseNo	Semester
111	BUSN2011	2016 S1
		16 S1
		16 S2

- C

```
SELECT COUNT(*)
FROM STUDENT NATURAL JOIN ENRO
```

Using NATURAL JOIN – Same Example

STUDENT

StudentID	Name
111	Tom

ENROL

StudentID	CourseNo	Semester
111	BUSN2011	2016 S1
		16 S1
		16 S2

- C

```
SELECT COUNT(*)
FROM STUDENT NATURAL JOIN ENRO
```


StudentID	Name	C	
111	Tom	BU	
111	Tom	COMP2400	2016 S2
222	Emily	COMP2400	2016 S1

Using NATURAL JOIN – Same Example

STUDENT

StudentID	Name
111	Tom

ENROL

StudentID	CourseNo	Semester
111	BUSN2011	2016 S1
		16 S1
		16 S2

- C

```
SELECT COUNT(*)
FROM STUDENT NATURAL JOIN ENRO
```


StudentID	Name	C	
111	Tom	BU	
111	Tom	COMP2400	2016 S2
222	Emily	COMP2400	2016 S1

- **The above query is incorrect!**

Using NATURAL JOIN – Same Example

STUDENT

StudentID	Name
111	Tom

ENROL

StudentID	CourseNo	Semester
111	BUSN2011	2016 S1
		16 S1
		16 S2

- C

```
SELECT COUNT(*)
FROM STUDENT NATURAL JOIN ENRO
```


StudentID	Name	C	
111	Tom	BU	
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	
	Name

Enrol		
	CourseNo	Semester
		16 S1
		16 S1
		16 S2

- C

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

A Simple Solution – Same Example

STUDENT	
	Name

ENROL		
	CourseNo	Semester
		16 S1
		16 S1
		16 S2

- C

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


- **The above query is correct!**

A Simple Solution – Same Example

STUDENT	
	Name

ENROL		
	CourseNo	Semester
		16 S1
		16 S1
		16 S2

- C

```
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

Subqueries – More Examples

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


```
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


```
WHERE Semester = '2016 S2'
GROUP BY CourseNo;
```


- List the largest number of students enr Semester 2 2016

Subqueries – More Examples

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


```
WHERE Semester = '2016 S2'
GROUP BY CourseNo;
```


- List the largest number of students enrolled 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


```
WHERE e1.Semester = '201
GROUP BY e1.CourseNo) e
WHERE e.NoOfStudents =
      (SELECT MAX(e2.No
      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

U [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

U [1]


```
        FROM ENROL e1
        WHERE e1.Semester = '201
        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	COMP1400	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 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


```
        WHERE e1.Semester = '201
        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 co

Subqueries – More Examples

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


```
        FROM Enrol e1
        WHERE e1.Semester = '201
        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

Subqueries – More Examples

- List students (ses) in

Subqueries – More Examples

- Li
  S
  se
  s) in
  - List the students' IDs and the corresponding number of enrolled courses in Semester 2 2016

Subqueries – More Examples

- Li
  S
  se
) in

  - List the students' IDs and the corresponding number of enrolled courses in Semester 2 2016


```
SELECT e.StudentID, COUNT(*) AS N
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
```



```
s
ON (s.StudentID = ne.StudentID) AND (ne
```

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
ON (s.StudentID = ne.StudentID) AND (ne
WITH StudEnrols AS (
        SELECT e.StudentID, COUNT(*
        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.

222	BUSN2011	2016 S2
222	COMP2400	2016 S2
333	BUSN2011	2016 S2
333	COMP2400	2016 S2
333	ECON2002	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.

222	BUSN2011	2016 S2
222	COMP2400	2016 S2
333	BUSN2011	2016 S2
333	COMP2400	2016 S2
333	ECON2102	2016 S2

		ols

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".

[Credit Cookie] The Shortest Code/Program?

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

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

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