COMM1822

Term 2 2022

Introduction to Databases for Business Analytics

Week 5 Relational Algebra and SQL Joins

Lecturer-in-Charge: Kam-Fung (Henry) Cheung
Email: kf.cheung@unsw.edu.au
Tutors: Theresa Tran
Liam Li Chen
Kathy Xu
PASS Leader: Srilekha Chandrashekara Kolaki

UNSW SYDNEY

Copyright

- There are some file-sharing websites that specialise in buying and selling academic work to and from university students.

- If you upload your original work to these websites, and if another student downloads and presents it as their own either wholly or partially, **you might be found guilty of collusion — even years after graduation**.

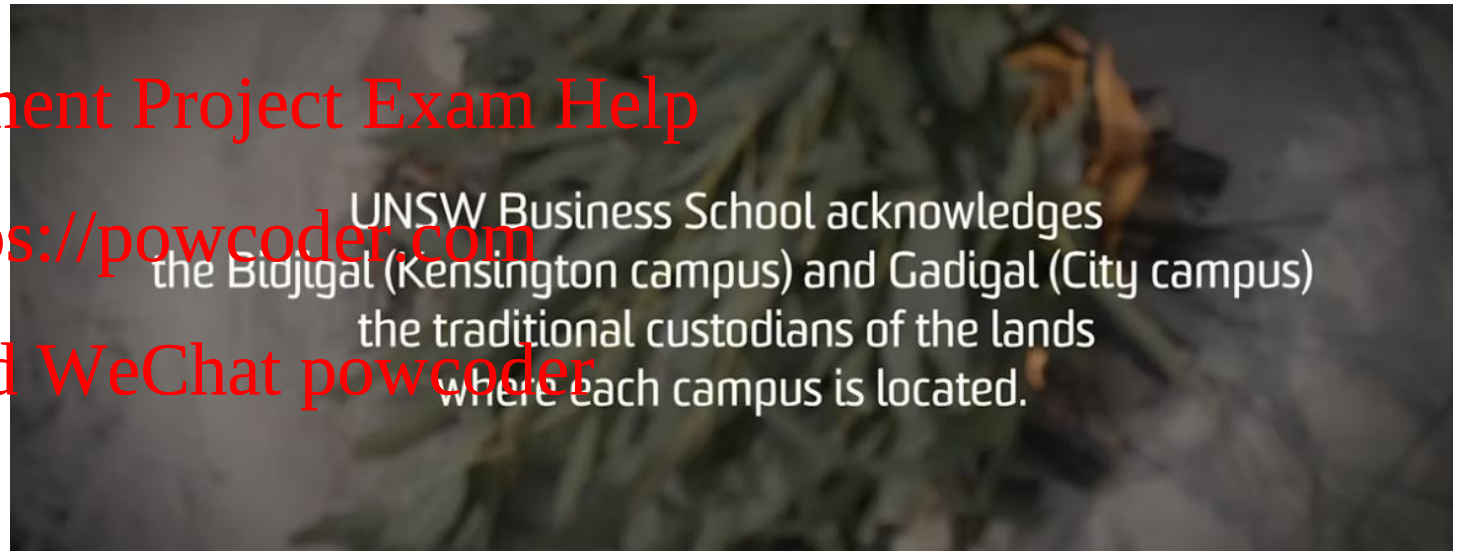
- These file-sharing websites may also accept purchase of course materials, **such as copies of lecture slides and tutorial handouts. By law, the copyright on course materials, developed by UNSW staff in the course of their employment, belongs to UNSW. It constitutes copyright infringement, if not academic misconduct, to trade these materials.**

Acknowledgement of Country

UNSW Business School acknowledges the Bidjigal (Kensington campus) and Gadigal (City campus) the traditional custodians of the lands where each campus is located.

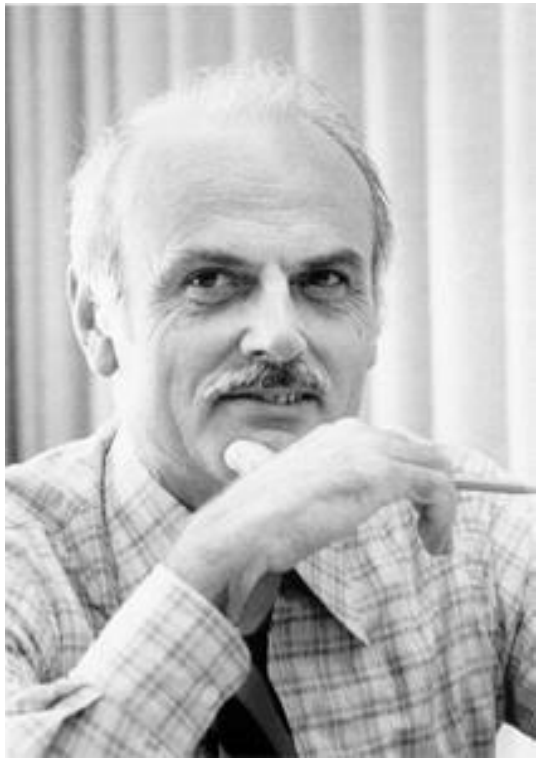
We acknowledge all Aboriginal and Torres Strait Islander Elders, past and present and their communities who have shared and practiced their teachings over thousands of years including business practices.

We recognise Aboriginal and Torres Strait Islander people's ongoing leadership and contributions, including to business, education and industry.



UNSW Business School. (2022, May 7). *Acknowledgement of Country* [online video]. Retrieved from https://vimeo.com/369229957/d995d8087f

Relational Languages



- Codd (1970, 1971)'s **relation model** is the conceptual basis for relational databases. The relational model includes **two relational languages**:

1. **Relational algebra** is a **non-procedural, high-level language** that provides a declarative way to specify database queries. (Relational algebra "declares a definition" to get to certain data.)
2. **Relational calculus** is a **procedural, low-level language** that provides a procedural way for specifying queries. (Relational calculus provides a "order of steps" to get to certain data.)

- For every expression in the relational algebra there is an equivalent expression in the relational calculus, and vice versa. They are **logically equivalent**.
- **Relational algebra and relational calculus** are **not very user friendly**.
- **SQL** was developed as **user-friendly query** to work with RDBMS.

Relational Algebra

The relational algebra has **operations**. These operations fall into **three main categories**:

1. **Union**, **Intersection** and **Difference**: Boolean operations to define a new relation based on two existing relations.

2. **Selection** and **Projection**: Operations that remove parts of a relation.
3. **Cartesian Product / Joins**: Operations that combine the tuples of two relations.

1) Union, Intersection and Difference

- **Union, Intersection and Difference** are operations (or "set operations") on two relations (R and S), both relations should have schemas with **identical sets of attributes** and **identical order of the attributes**.
- **UNION: R ∪ S**
  - The union of R and S is the set of all tuples that are in R and S.
  - In short: *merge the two sets of tuples!*
- **INTERSECTION: R ∩ S**
  - The intersection of R and S is the set of tuples that appear in both tables.
  - In short: *find the common tuples!*
- **DIFFERENCE**: **R - S**
  - The difference of R and S, is the set of tuples that are in R but not in S.
  - In short: *subtract the tuples in S from the tuples in R!*

Question: Is R - S the same as S - R?

R

P_CODE
345678
345679

S

P_CODE
123456
123457
123458

Exercise 1

1

P_CODE
123456
123457
123458

UNION

P_CODE
345678
345679

3

F_NAME
Kaiser
Julia
Alex
Asa
Evan

DIFFERENCE

F_NAME
Julia
Shane
Evan
Ben

R S

2

F_NAME
Kaiser
Julia
Alex
Asa
Evan

INTERSECT

F_NAME
Julia
Shane
Evan
Ben

R S

4

F_NAME
Julia
Shane
Evan
Ben

DIFFERENCE

F_NAME
Kaiser
Julia
Alex
Asa
Evan

S R

Solution to Exercise 1

1

P_CODE
123456
123457
123458

UNION

P_CODE
345678
345679

yields

P_CODE
123456
123457
123458
345678
345679

3

F_NAME
Kaiser
Julia
Alex
Asa
Evan

R

DIFFERENCE

F_NAME
Julia
Shane
Evan
Ben

S

yields

F_NAME
Kaiser
Alex
Asa

R - S

2

F_NAME
Kaiser
Julia
Alex
Asa
Evan

R

INTERSECT

F_NAME
Julia
Shane
Evan
Ben

S

yields

F_NAME
Julia
Evan

4

F_NAME
Julia
Shane
Evan
Ben

S

DIFFERENCE

F_NAME
Kaiser
Julia
Alex
Asa
Evan

R

yields

F_NAME
Shane
Ben

S - R

2) Selection and Projection

❑ **Selection and Projection** operations are applied to a single relation (R).

❑ SELECTION

- Selection (SELECT) returns a relation that contains only those tuples (i.e., rows in a table) from a specified relation (R) that satisfy a specified condition (horizontal subset of a table).
- Relational operator is σ. $\sigma_{\text{predicate}}$ R (σ = “sigma”)

❑ PROJECTION

- Projection (PROJECT) returns a relation that contains a list of tuples for selected attributes from a specified relation (R), eliminating duplicates (vertical subset of a table).
- Relational operator is π. $\pi_{\text{attribute 1, … attribute n}}$ R (π = “pi”)

Exercise 2

ProductCode	ProductDescription	Price
213345	9V battery	1.92
311452	Power drill	34.99
254467	100W bulb	1.47

1. What is the result of: $\sigma_{price < 2.00} R$?
   ("Selection with Price less than 2.00 of R"; "List all tuples with a price less than 2.00")
2. What is the result of $\pi_{Product\ Description,\ Price}\ R$?
   ("Projection with Product Description, Price from R"; "List all tuples showing only description and price")

Solution to Exercise 2

ProductCode	ProductDescription	Price
213345	9v battery	1.92
311452	Power drill	34.99
254467	100W bulb	1.47

- Selection: $\sigma_{price < 2.00} R$ ("List all tuples with a price less than 2.00")

213345	9v battery	1.92
254467	100w bulb	1.47

- Projection: $\pi_{Product\ Description,\ Price} R$ ("List all tuples showing only description and price")

9v battery	1.92
Power drill	34.99
100w bulb	1.47

3) Cartesian Product and Joins

- **Cross Join** joins (creates and returns) the **Cartesian Product** of two relations R and S. ("$R * S$", "all possible tuple combinations of two relations", "everything joined to everything").
- **Inner Join** returns combined tuples from R and S that **fulfil a certain criterion**. This is the most common, the **default join type**.
  - An **Equi Join** joins tuples from R and S based on **equality of values for specified attributes**. The join is called a **Theta Join** if a comparison other than "equality" (=) is used, such as "smaller/less than" (<).
  - A **Natural Join** joins tuples from R and S that **agree in value for whatever attributes are common** to the schemas of R and S. The attributes are not explicitly specified. Hence, "naturally", **attributes in common are used for the join**.
- A Full **Outer Join** returns tuples from both relations with their matching values in the respective other relation (i.e., tuples with no match in the other relation still appears, with NULL values instead of matching values).

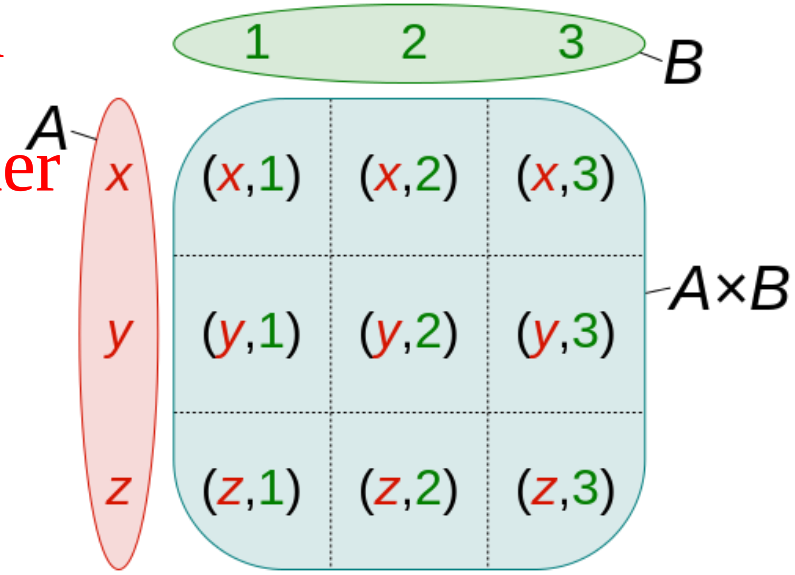
Cross Join (Cartesian Product)

- **Cartesian** = "relating to **René Descartes (1596-1650)** and his ideas". The word comes from the Latinised version (Renatus Cartesius) of the name (René Descartes).

- Descartes made major progress in **analytical geometry**.

A = [x , y , z]

B = [1, 2 , 3]



Cross Join (Cartesian Product)

- **Cross Join (Cartesian Product)**: Select all possible combinations of tuples in R with tuples in S ("$R * S$", "all possible tuple combinations of two relations", "everything joined to everything").

In SQL:

SELECT * FROM R **CROSS JOIN** S; this is an **explicit** cross join

SELECT * FROM R, S; this is an **implicit** cross join

Question: Is a **Cross Join** of R, S identical to a **Union** of R, S? Why (not)?

Cross Join (Cartesian Product)

G

	Name	City
x	Mary	Boston
y	Susan	Chicago
z	Betty	Chicago

B

	Name	City
a	Sam	Chicago
b	James	Dallas

Cartesian Product:

SELECT * FROM G, B;

SELECT * FROM G CROSS JOIN B;

	G.Name	G.City	B.Name	B.City
x, a	Mary	Boston	Sam	Chicago
y, a	Susan	Chicago	Sam	Chicago
z, a	Betty	Chicago	Sam	Chicago
x, b	Mary	Boston	James	Dallas
y, b	Susan	Chicago	James	Dallas
z, b	Betty	Chicago	James	Dallas

Compare to Union

G

Name	City
Mary	Boston
Susan	Chicago
Betty	Chicago

B

Name	City
Sam	Chicago
James	Dallas

Union:

```
SELECT * FROM G
UNION [ALL]
SELECT * FROM B;
```

Name	City
Mary	Boston
Susan	Chicago
Betty	Chicago
Sam	Chicago
James	Dallas

Inner Join

An **Inner Join** returns combined tuples from two relations that have the same value for a defined attribute (match on the attribute). This is the default join type, the most common join type.

SELECT * FROM R **INNER JOIN** S this is an **explicit** inner join

ON R.attribute = S.attribute

Tip: One way to think of an Inner Join is as a Cross Join (Cartesian Product) with all tuples removed that do *not* match on the defined attribute.

Inner Join

G

Name	City
Mary	Boston
Susan	Chicago
Betty	Chicago

B

Name	City
Sam	Chicago
James	Dallas

Inner Join/Equi-Join:

SELECT * FROM G INNER JOIN B ON G.CITY = B.CITY;

G.Name	G.City	B.Name	B.City
Susan	Chicago	Sam	Chicago
Betty	Chicago	Sam	Chicago

SELECT * FROM G INNER JOIN B ON G.CITY = B.CITY;

- An **equi join** is a **join** with a **join** condition containing an equality operator.
- A **theta join** is when other comparison operators are used (<=, >=, <, >).

Inner Join & Natural Join

A **natural join** joins tuples based on all attributes with identical names in the two relations.

```
TableA                          TableB
+-----------+----------+        +--------------------+
|Column1    | Column2  |        |Column1  |  Column3 |
+-----------+----------+        +--------------------+
| 1         |  2       |        | 1       |   3      |
+-----------+----------+        +---------+----------+
```



```
+-----------+----------+----------+
|Column1    | Column2  | Column3  |
+-----------+----------+----------+
| 1         |  2       |   3      |
+-----------+----------+----------+
```

Natural Join
(All common columns)
Here only Column1

A **Natural Join** joins 2 tables on the basis of all common columns

An **Inner Join** joins 2 tables on the basis of common columns mentioned in the ON clause

```
+-----------+-----------+----------------------+
| a.Column1 | a.Column2 | b.Column1| b.Column3|
+-----------+-----------+----------+----------+
| 1         |  2        | 1        |   3      |
+-----------+-----------+----------+----------+
```

Inner Join
on Column1

Full Outer Join

Full Outer Join: Selects and joins tuples from two tables that match on a defined attribute. If there is no match for a tuple, the tuple will still appear with missing attributes shown as NULL.


```
SELECT * FROM R
FULL OUTER JOIN S
ON R.attribute = S.attribute
```

Full Outer Join

G

Name	City
Mary	Boston
Susan	Chicago
Betty	Chicago
Nancy	Null
Anne	Denver

B

Name	City
Sam	Chicago
James	Dallas
John	Boston
Henry	Boston
George	Null

Full Outer Join:

```
SELECT * FROM G FULL OUTER JOIN B USING (CITY);
SELECT * FROM G FULL OUTER JOIN B ON (G.CITY = B.CITY);
```

G.Name	G.City	B.Name	B.City
Mary	Boston	John	Boston
Mary	Boston	Henry	Boston
Susan	Chicago	Sam	Chicago
Betty	Chicago	Sam	Chicago
Anne	Denver	Null	Null
Nancy	Null	Null	Null
Null	Null	James	Dallas
Null	Null	George	Null

Left Outer Join

Left Outer Join : Select and joins tuple from the “left” table (R) with tuples from the “right” table (S) on defined attributes. If there is no match, the attributes from the right side will contain NULL values.


```
SELECT * FROM R
LEFT OUTER JOIN S
ON R.attribute = S.attribute
```

Left Outer Join

G

Name	City
Mary	Boston
Susan	Chicago
Betty	Chicago
Nancy	Null
Anne	Denver

B

Name	City
Sam	Chicago
James	Dallas
John	Boston
Henry	Boston
George	Null

Left Outer Join:

SELECT * FROM G LEFT OUTER JOIN B USING (CITY);

G.Name	G.City	B.Name	B.City
Mary	Boston	John	Boston
Mary	Boston	Henry	Boston
Susan	Chicago	Sam	Chicago
Betty	Chicago	Sam	Chicago
Anne	Denver	Null	Null
Nancy	Null	Null	Null

Right Outer Join

Right Outer Join : Select and joins tuple from the “left” table (R) with tuples from the “right” table (S) on defined attributes. If there is no match, the attributes from the left side will contain NULL values.


```
SELECT * FROM R
RIGHT OUTER JOIN S
ON R.attribute = S.attribute
```

Right Outer Join

G

Name	City
Mary	Boston
Susan	Chicago
Betty	Chicago
Nancy	Null
Anne	Denver

B

Name	City
Sam	Chicago
James	Dallas
John	Boston
Henry	Boston
George	Null

Right Outer Join:

SELECT * FROM G RIGHT OUTER JOIN B USING (CITY);

G.Name	G.City	B.Name	B.City
Mary	Boston	John	Boston
Mary	Boston	Henry	Boston
Susan	Chicago	Sam	Chicago
Betty	Chicago	Sam	Chicago
Null	Null	James	Dallas
Null	Null	George	Null

Full Outer Join, Left Outer Join and Right Outer Join

G

Name	City
Mary	Boston
Susan	Chicago
Betty	Chicago
Nancy	Null
Anne	Denver

B

Name	City
Sam	Chicago
James	Dallas
John	Boston
Henry	Boston
George	Null

Full Outer Join:

SELECT * FROM G FULL OUTER JOIN B USING (CITY);

SELECT * FROM G FULL OUTER JOIN B ON (G.CITY = B.CITY);

G.Name	G.City	B.Name	B.City
Mary	Boston	John	Boston
Mary	Boston	Henry	Boston
Susan	Chicago	Sam	Chicago
Betty	Chicago	Sam	Chicago
Anne	Denver	Null	Null
Nancy	Null	Null	Null
Null	Null	James	Dallas
Null	Null	George	Null

Left Outer Join:

SELECT * FROM G LEFT OUTER JOIN B USING (CITY);

G.Name	G.City	B.Name	B.City
Mary	Boston	John	Boston
Mary	Boston	Henry	Boston
Susan	Chicago	Sam	Chicago
Betty	Chicago	Sam	Chicago
Anne	Denver	Null	Null
Nancy	Null	Null	Null

Right Outer Join:

SELECT * FROM G RIGHT OUTER JOIN B USING (CITY);

G.Name	G.City	B.Name	B.City
Mary	Boston	John	Boston
Mary	Boston	Henry	Boston
Susan	Chicago	Sam	Chicago
Betty	Chicago	Sam	Chicago
Null	Null	James	Dallas
Null	Null	George	Null

Exercise 3

Table: R

P_CODE	PRICE
AA	5.99
BB	22.75

Table: S

STORE	AISLE	SHELF
23	W	5
24	K	9
25	Z	6

Dimension of T: Dim(T) = (No. of rows in T, No. of columns in T)

Dim(R) = (2, 2)

Dim(S) = (3, 3)

Dim($R * S$) = (2 x 3, 2 + 3) = (6, 5)

Build the Cartesian Product of $R * S$.

Solution to Exercise 3

Table: R

P_CODE	PRICE
AA	5.99
BB	22.75

Dim(R) = (2, 2)

Table: S

STORE	AISLE	SHELF
23	W	5
24	K	9
25	Z	6

Dim(S) = (3, 3)

Table: $R * S$

P_CODE	PRICE	STORE	AISLE	SHELF
AA	5.99	23	W	5
AA	5.99	24	K	9
AA	5.99	25	Z	6
BB	22.75	23	W	5
BB	22.75	24	K	9
BB	22.75	25	Z	6

Dim($R * S$) = (6, 5)

Exercise 4

- Apply natural (inner) join, left outer join, right outer join and full outer join on ***Std_Name***.

Table 1

Std_Name	Tutor_Name
Mr. Brown	Reed R.
Mr. Green	Yeo, J.
Ms. White	Yeo, J.

Table 2

Std_Name	Subject
Mr. Brown	SADF
Ms. White	BDM
Ms. Pink	BDM

Solution to Exercise 4

Std_Name	Tutor_Name
Mr. Brown	Reed R.
Mr. Green	Yeo, J.
Ms. White	Yeo, J.

Std_Name	Subject
Mr. Brown	SADF
Ms. White	BDM
Ms. Pink	BDM

Inner Join/ Natural join

Student_Name	Tutor_Name	Subjects
Mr. Brown	Reed, G.	SADF
Ms. White	Yeo, J.	BDM

Left Outer Join

Student_Name	Tutor_Name	Subjects
Mr. Brown	Reed, G.	SADF
Ms. White	Yeo, J.	BDM
Mr. Green	Yeo, J.	NULL

Solution to Exercise 4

Std_Name	Tutor_Name
Mr. Brown	Reed R.
Mr. Green	Yeo, J.
Ms. White	Yeo, J.

Std_Name	Subject
Mr. Brown	SADF
Ms. White	BDM
Ms. Pink	BDM

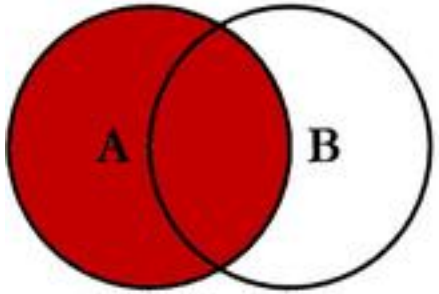
Right Outer Join

Student_Name	Tutor_Name	Subjects
Mr. Brown	Reed, G.	SADF
Ms. White	Yeo, J.	BDM
Ms. Pink	NULL	BDM

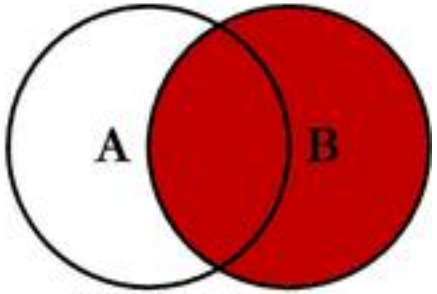
Full Outer Join

Student_Name	Tutor_Name	Subjects
Mr. Brown	Reed, G.	SADF
Ms. White	Yeo, J.	BDM
Mr. Green	Yeo, J.	NULL
Ms. Pink	NULL	BDM

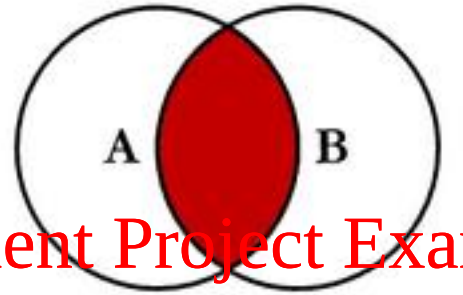
SQL JOINS



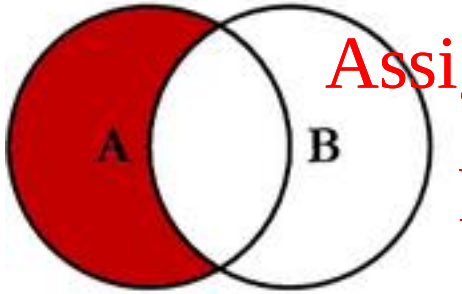

```
SELECT <select_list>
FROM TableA A
LEFT JOIN TableB B
ON A.Key = B.Key
```



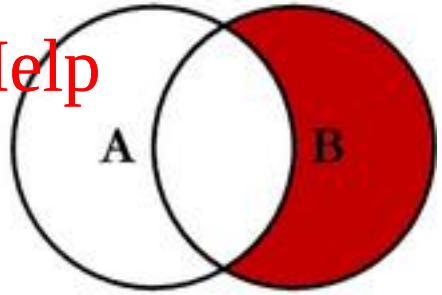

```
SELECT <select_list>
FROM TableA A
RIGHT JOIN TableB B
ON A.Key = B.Key
```



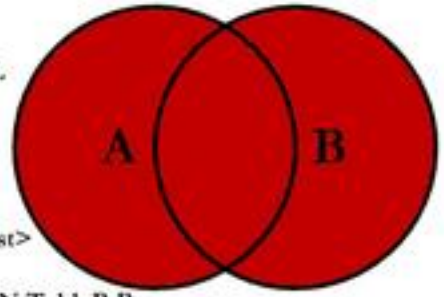

```
SELECT <select_list>
FROM TableA A
ON A.Key = B.Key
```



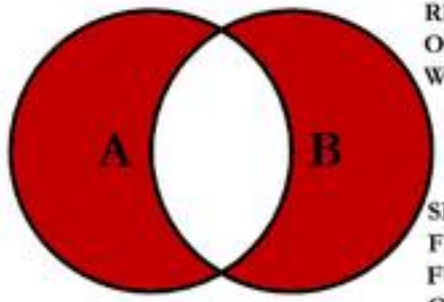

```
SELECT <select_list>
FROM TableA A
LEFT JOIN TableB B
ON A.Key = B.Key
WHERE B.Key IS NULL
```




```
SELECT <select_list>
FROM TableA A
RIGHT JOIN TableB B
ON A.Key = B.Key
WHERE A.Key IS NULL
```

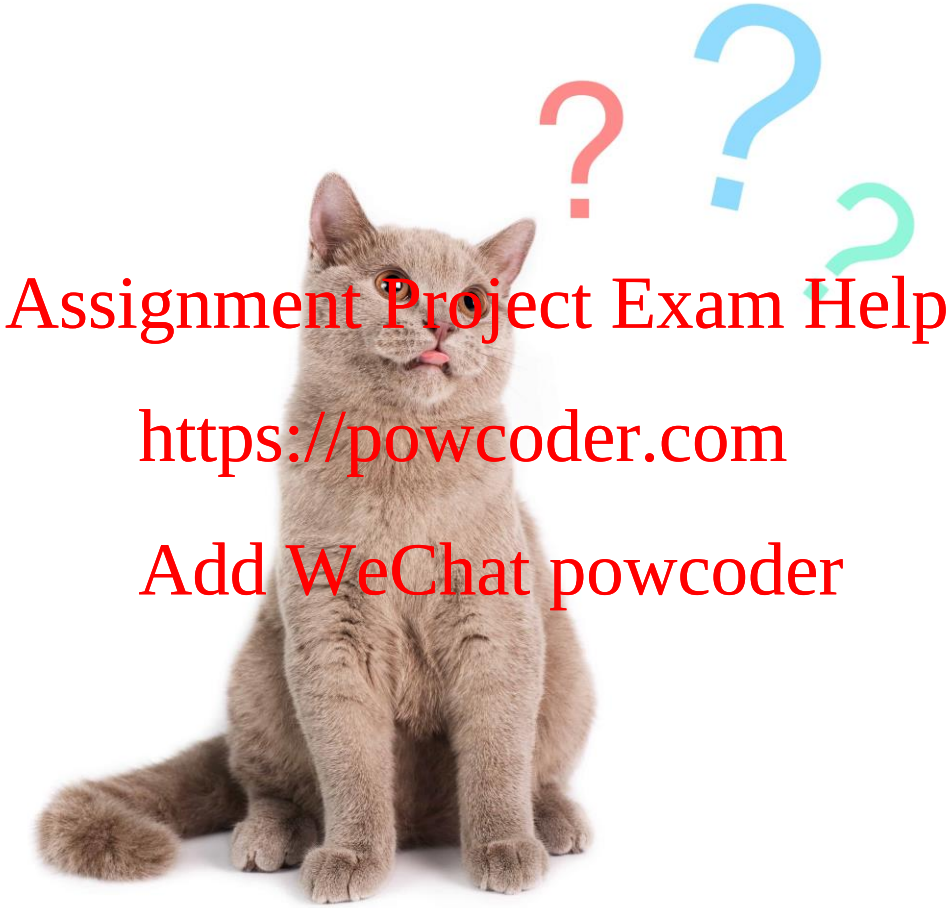



```
SELECT <select_list>
FROM TableA A
FULL OUTER JOIN TableB B
ON A.Key = B.Key
```




```
SELECT <select_list>
FROM TableA A
FULL OUTER JOIN TableB B
ON A.Key = B.Key
WHERE A.Key IS NULL
OR B.Key IS NULL
```

Questions



Source: petcare.com.au