ECE30030/ITP30010 Database Systems

Advanced SQL

Reading: Chapters 4-5

Charmgil Hong

charmgil@handong.edu

Spring, 2023

Handong Global University

Agenda

- Join
- Views
- Window functions
- Keys

Join Operations

- Join operations take two relations and return another relation
  - A join is a Cartesian product that requires tuples in the two relations match
    - It also specifies the attributes that are present in the result of the join (project)
  - Typically used as subquery expressions in the **FROM** clause

  - Join types
    - **INNER JOIN**
    - **OUTER JOIN**

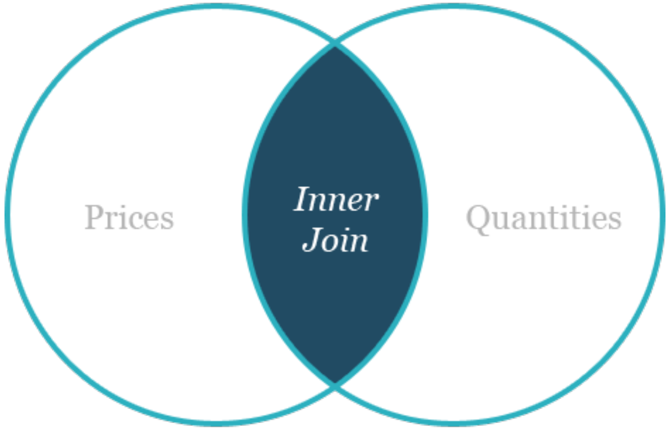
  - Join conditions
    - **NATURAL**
    - **ON** <predicate>
    - **USING (**A_1, A_2, …, A_n**)**

Inner & Outer Join

- Join: Compare and combine
  - Inner join: Returns matching data from tables
  - Outer join: Returns matching & some dissimilar data from tables

- Inner join

Table 1: prices		Table 2: quantities	
product	price	product	quantity
Potatoes	3.00	Potatoes	45
Avocados	4.00	Onions	20
Oranges	5.00	Broccoli	27

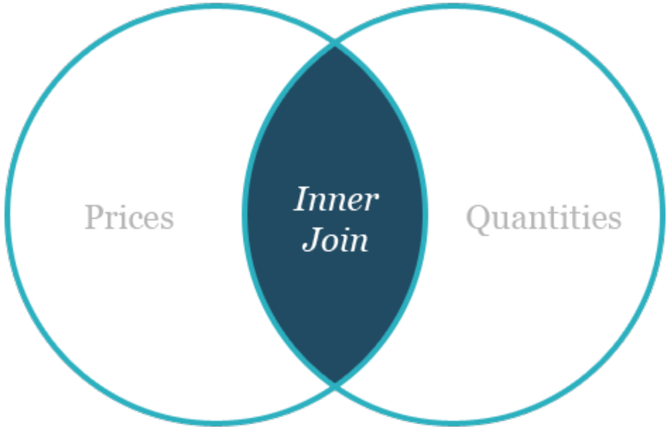


* Source: https://www.diffen.com/difference/Inner_Join_vs_Outer_Join

Inner & Outer Join

- Inner join: *matching data*

Table 1: prices		Table 2: quantities	
product	price	product	quantity
Potatoes	3.00	Potatoes	45
Avocados	4.00	Onions	20
Oranges	5.00	Broccoli	27



```
SELECT * FROM prices, quantities
WHERE prices.product = quantities.product;
```

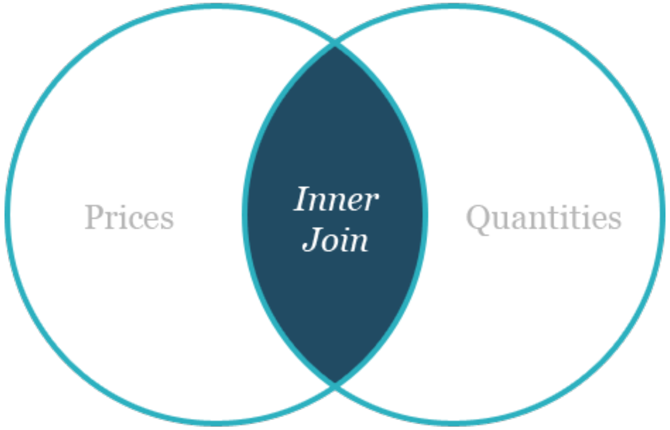
product	price	product	quantity
Potatoes	3.00	Potatoes	45

* Source: https://www.diffen.com/difference/Inner_Join_vs_Outer_Join

Inner & Outer Join

- Inner join: *matching data*

Table 1: prices		Table 2: quantities	
product	price	product	quantity
Potatoes	3.00	Potatoes	45
Avocados	4.00	Onions	20
Oranges	5.00	Broccoli	27



```
SELECT * FROM prices
JOIN quantities ON prices.product=quantities.product;
```

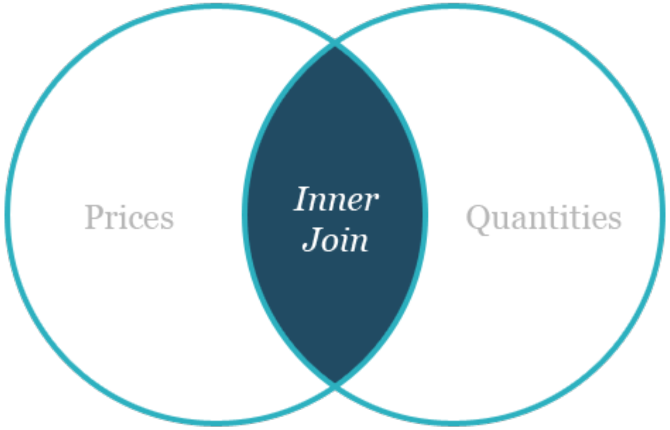
product	price	product	quantity
Potatoes	3.00	Potatoes	45

* Source: https://www.diffen.com/difference/Inner_Join_vs_Outer_Join

Inner & Outer Join

- Inner join: *matching data*

Table 1: prices		Table 2: quantities	
product	price	product	quantity
Potatoes	3.00	Potatoes	45
Avocados	4.00	Onions	20
Oranges	5.00	Broccoli	27



```
SELECT * FROM prices
NATURAL JOIN quantities;
```

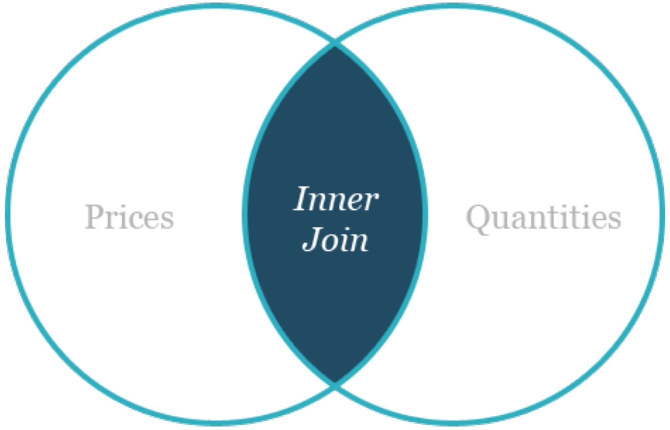
product	price	quantity
Potatoes	3.00	45

* Source: https://www.diffen.com/difference/Inner_Join_vs_Outer_Join

Inner & Outer Join

- Inner join: *matching data*

Table 1: prices		Table 2: quantities	
product	price	product	quantity
Potatoes	3.00	Potatoes	45
Avocados	4.00	Onions	20
Oranges	5.00	Broccoli	27



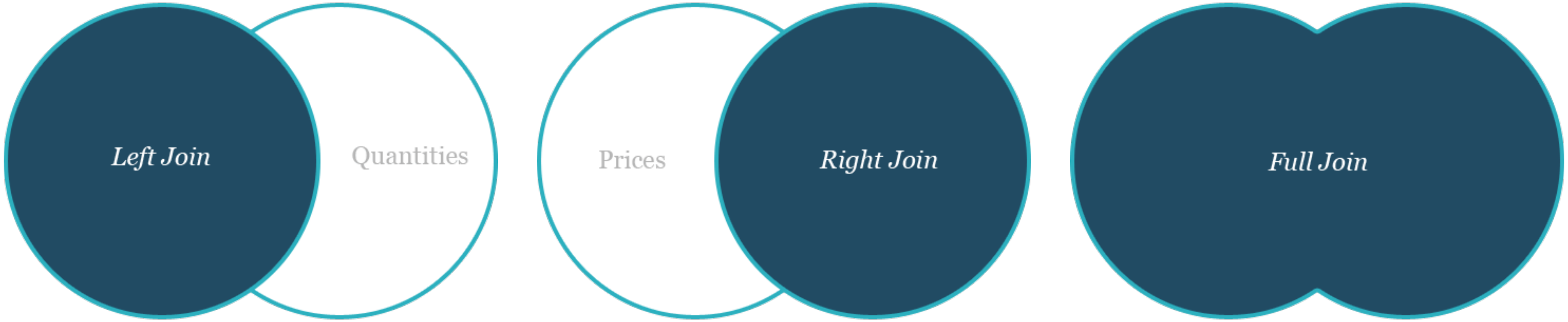
```
SELECT * FROM prices
JOIN quantities USING (product);
```

product	price	quantity
Potatoes	3.00	45

* Source: https://www.diffen.com/difference/Inner_Join_vs_Outer_Join

Inner & Outer Join

- Outer join: *matching & some dissimilar data*
  - Returns a set of records that include what an inner join would return + other rows for which no corresponding match is found in the other table
  - Left (outer) join: ”Left” records with no corresponding entry on the “right”
  - Right (outer) join: ”Right” records with no corresponding entry on the “left”
  - Full (outer) join: “All” records with no corresponding entry on the other table

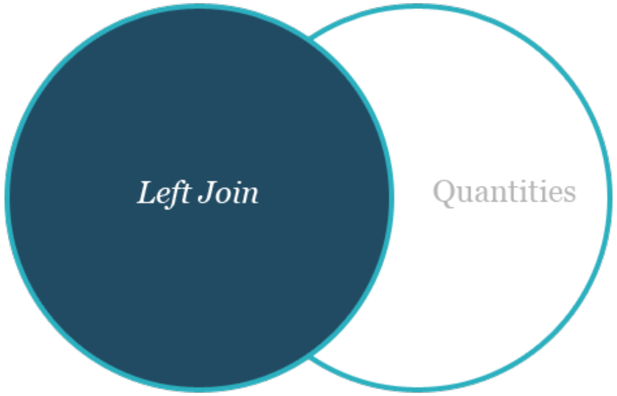


* Source: https://www.diffen.com/difference/Inner_Join_vs_Outer_Join

Inner & Outer Join

- Outer join

Table 1: prices		Table 2: quantities	
product	price	product	quantity
Potatoes	3.00	Potatoes	45
Avocados	4.00	Onions	20
Oranges	5.00	Broccoli	27



```
SELECT * FROM prices
LEFT OUTER JOIN quantities
   ON prices.product=quantities.product;
```

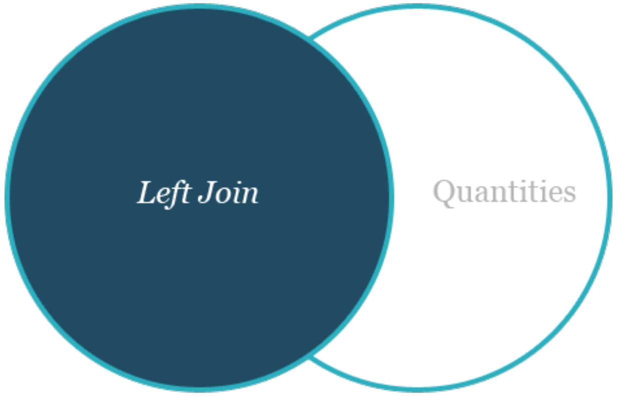
product	price	product	quantity
Potatoes	3.00	Potatoes	45
Avocados	4.00	NULL	NULL
Oranges	5.00	NULL	NULL

* Source: https://www.diffen.com/difference/Inner_Join_vs_Outer_Join

Inner & Outer Join

- Outer join

Table 1: prices		Table 2: quantities	
product	price	product	quantity
Potatoes	3.00	Potatoes	45
Avocados	4.00	Onions	20
Oranges	5.00	Broccoli	27



```
SELECT * FROM prices
LEFT OUTER JOIN quantities
   ON prices.product=quantities.product;
```

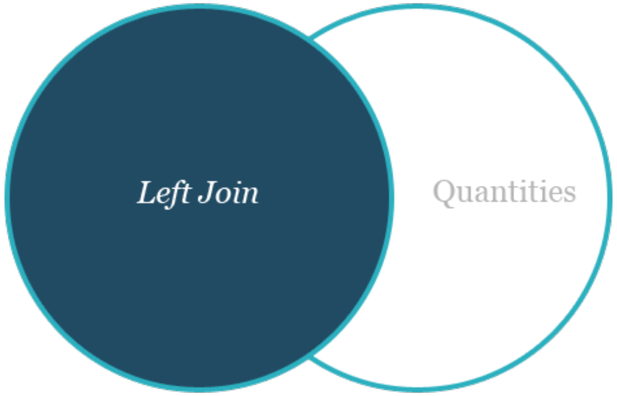
product	price	product	quantity
Potatoes	3.00	Potatoes	45
Avocados	4.00	NULL	NULL
Oranges	5.00	NULL	NULL

* Source: https://www.diffen.com/difference/Inner_Join_vs_Outer_Join

Inner & Outer Join

- Outer join

Table 1: prices		Table 2: quantities	
product	price	product	quantity
Potatoes	3.00	Potatoes	45
Avocados	4.00	Onions	20
Oranges	5.00	Broccoli	27



SELECT * FROM prices
NATURAL LEFT OUTER JOIN quantities;

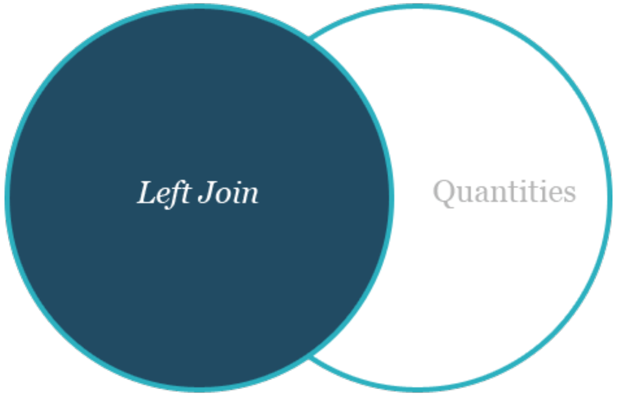
product	**price**	**quantity**
Potatoes	3.00	45
Avocados	4.00	NULL
Oranges	5.00	NULL

* Source: https://www.diffen.com/difference/Inner_Join_vs_Outer_Join

Inner & Outer Join

- Outer join

Table 1: prices		Table 2: quantities	
product	price	product	quantity
Potatoes	3.00	Potatoes	45
Avocados	4.00	Onions	20
Oranges	5.00	Broccoli	27



SELECT * **FROM** prices
LEFT OUTER JOIN quantities **USING** (product);

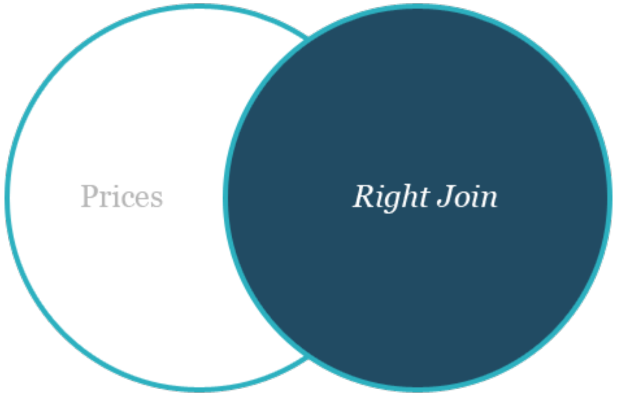
product	price	quantity
Potatoes	3.00	45
Avocados	4.00	NULL
Oranges	5.00	NULL

* Source: https://www.diffen.com/difference/Inner_Join_vs_Outer_Join

Inner & Outer Join

- Outer join

Table 1: prices		Table 2: quantities	
product	price	product	quantity
Potatoes	3.00	Potatoes	45
Avocados	4.00	Onions	20
Oranges	5.00	Broccoli	27



```
SELECT * FROM prices
RIGHT OUTER JOIN quantities
   ON prices.product=quantities.product;
```

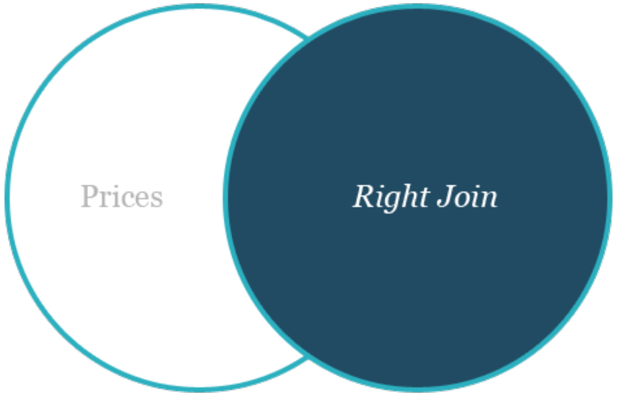
product	price	product	quantity
Potatoes	3.00	Potatoes	45
NULL	NULL	Onions	20
NULL	NULL	Broccoli	27

* Source: https://www.diffen.com/difference/Inner_Join_vs_Outer_Join

Inner & Outer Join

- Outer join

Table 1: prices		Table 2: quantities	
product	price	product	quantity
Potatoes	3.00	Potatoes	45
Avocados	4.00	Onions	20
Oranges	5.00	Broccoli	27



SELECT * FROM prices
NATURAL RIGHT OUTER JOIN quantities;

⇔

SELECT * FROM prices
RIGHT OUTER JOIN quantities
USING (product**);**

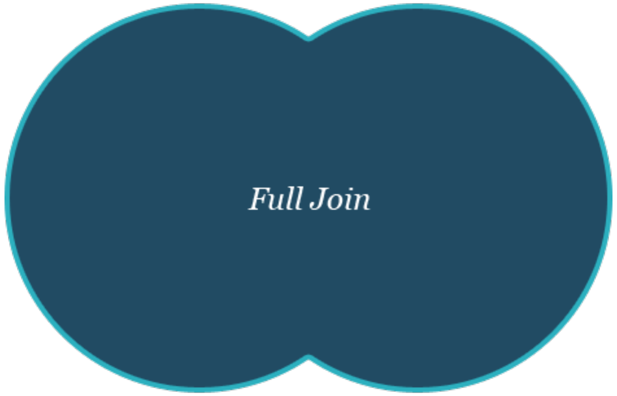
product	quantity	price
Potatoes	45	3.00
Onions	20	NULL
Broccoli	27	NULL

* Source: https://www.diffen.com/difference/Inner_Join_vs_Outer_Join

Inner & Outer Join

- Outer join

Table 1: prices		Table 2: quantities	
product	price	product	quantity
Potatoes	3.00	Potatoes	45
Avocados	4.00	Onions	20
Oranges	5.00	Broccoli	27



```
SELECT * FROM prices
LEFT OUTER JOIN quantities
   ON prices.product=quantities.product
UNION
SELECT * FROM prices
RIGHT OUTER JOIN quantities
   ON prices.product=quantities.product;
```

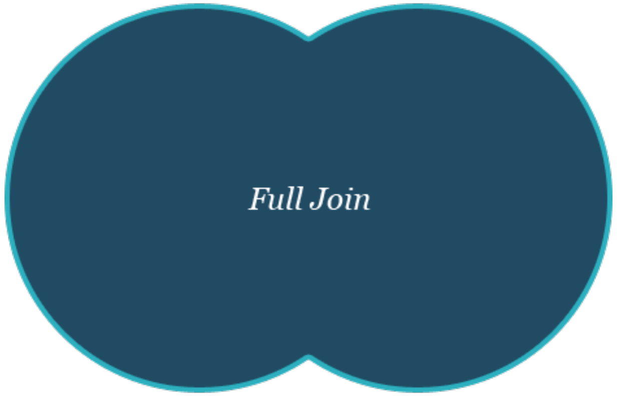
product	price	product	quantity
Potatoes	3.00	Potatoes	45
Avocados	4.00	NULL	NULL
Oranges	5.00	NULL	NULL
NULL	NULL	Onions	20
NULL	NULL	Broccoli	27

* Source: https://www.diffen.com/difference/Inner_Join_vs_Outer_Join

Inner & Outer Join

- Outer join

Table 1: prices		Table 2: quantities	
product	price	product	quantity
Potatoes	3.00	Potatoes	45
Avocados	4.00	Onions	20
Oranges	5.00	Broccoli	27



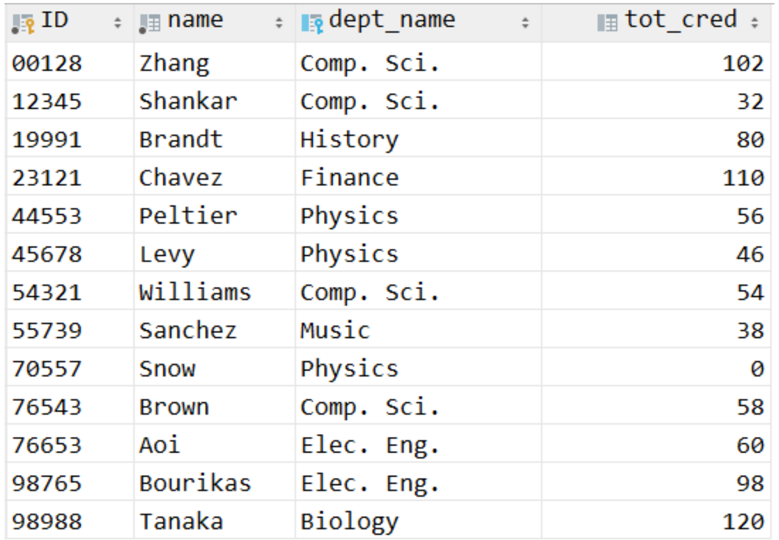
```
SELECT * FROM prices
FULL OUTER JOIN quantities
   ON prices.product=quantities.product;
```

product	price	product	quantity
Potatoes	3.00	Potatoes	45
Avocados	4.00	NULL	NULL
Oranges	5.00	NULL	NULL
NULL	NULL	Onions	20
NULL	NULL	Broccoli	27

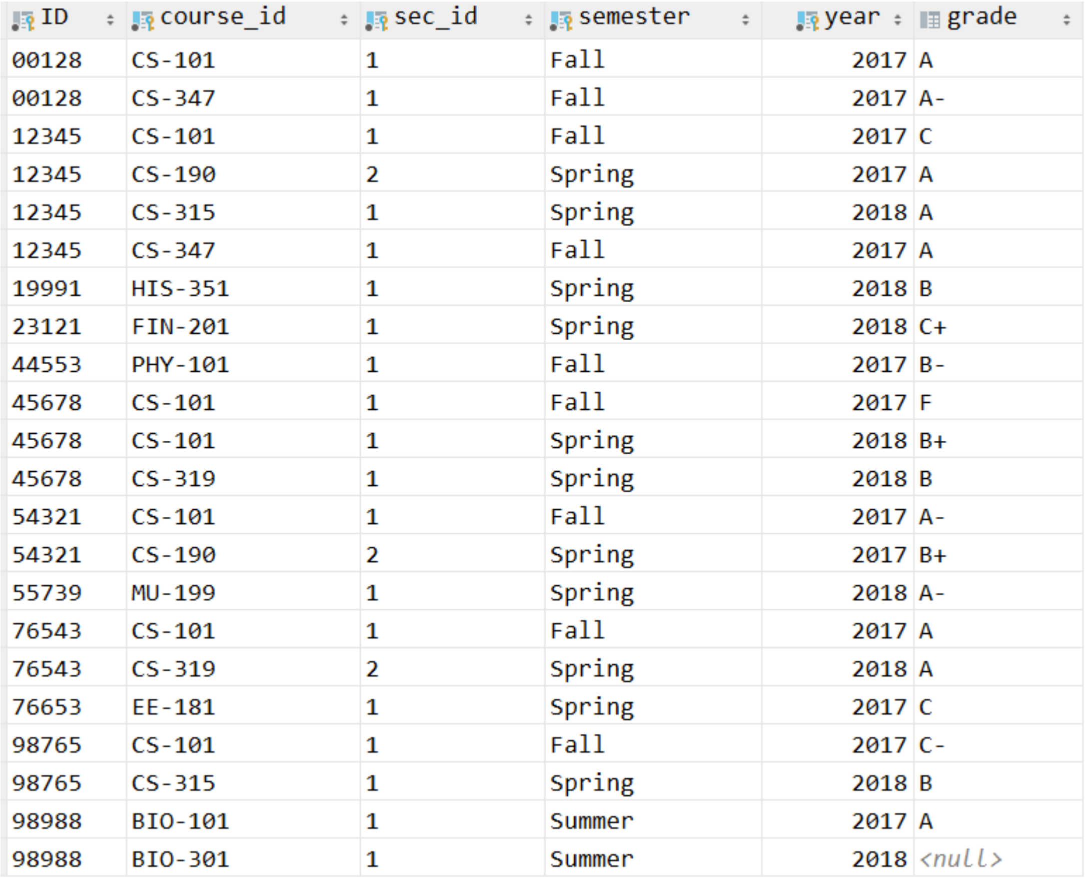
* Source: https://www.diffen.com/difference/Inner_Join_vs_Outer_Join

Running Example

- Relations: student, takes



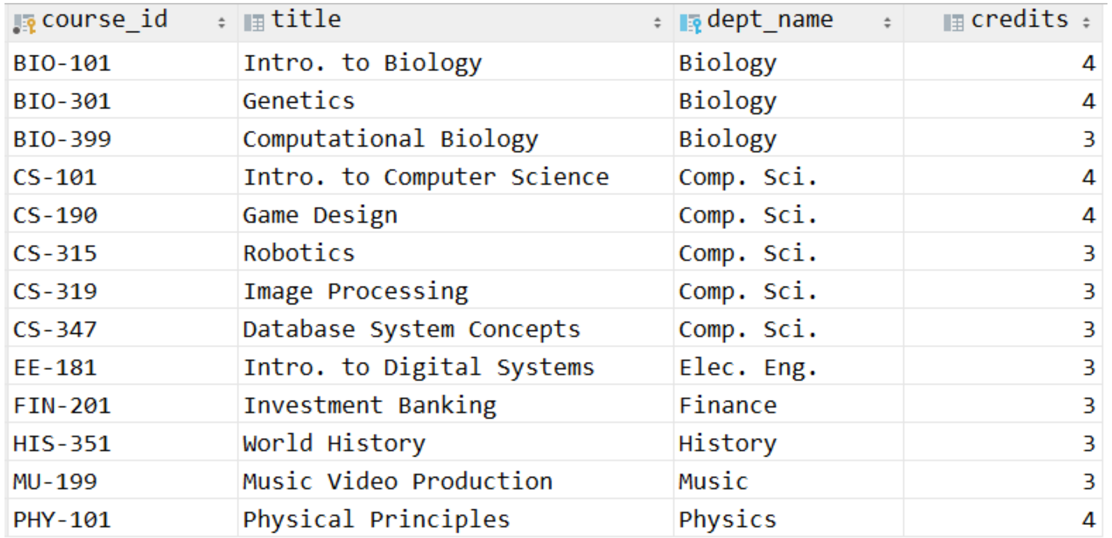
ID	name	dept_name	tot_cred
00128	Zhang	Comp. Sci.	102
12345	Shankar	Comp. Sci.	32
19991	Brandt	History	80
23121	Chavez	Finance	110
44553	Peltier	Physics	56
45678	Levy	Physics	46
54321	Williams	Comp. Sci.	54
55739	Sanchez	Music	38
70557	Snow	Physics	0
76543	Brown	Comp. Sci.	58
76653	Aoi	Elec. Eng.	60
98765	Bourikas	Elec. Eng.	98
98988	Tanaka	Biology	120



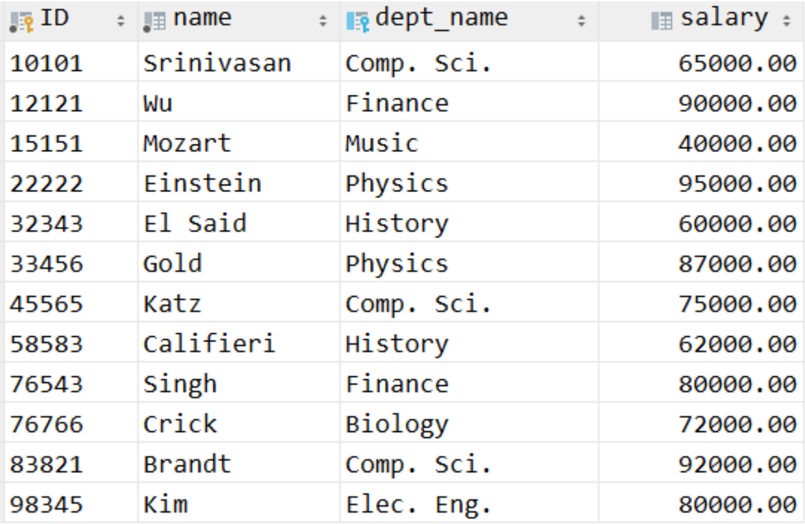
ID	course_id	sec_id	semester	year	grade
00128	CS-101	1	Fall	2017	A
00128	CS-347	1	Fall	2017	A-
12345	CS-101	1	Fall	2017	C
12345	CS-190	2	Spring	2017	A
12345	CS-315	1	Spring	2018	A
12345	CS-347	1	Fall	2017	A
19991	HIS-351	1	Spring	2018	B
23121	FIN-201	1	Spring	2018	C+
44553	PHY-101	1	Fall	2017	B-
45678	CS-101	1	Fall	2017	F
45678	CS-101	1	Spring	2018	B+
45678	CS-319	1	Spring	2018	B
54321	CS-101	1	Fall	2017	A-
54321	CS-190	2	Spring	2017	B+
55739	MU-199	1	Spring	2018	A-
76543	CS-101	1	Fall	2017	A
76543	CS-319	2	Spring	2018	A
76653	EE-181	1	Spring	2017	C
98765	CS-101	1	Fall	2017	C-
98765	CS-315	1	Spring	2018	B
98988	BIO-101	1	Summer	2017	A
98988	BIO-301	1	Summer	2018	<null>

Running Example

- Relations: course, instructor



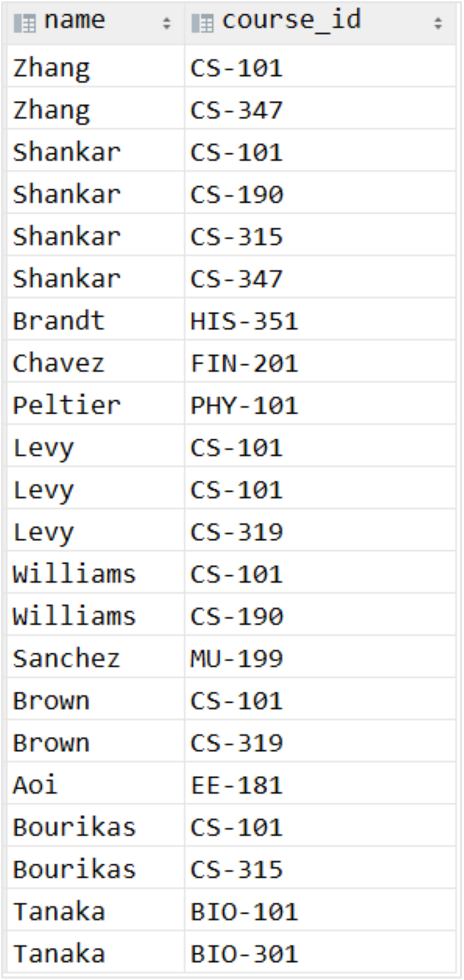
course_id	title	dept_name	credits
BIO-101	Intro. to Biology	Biology	4
BIO-301	Genetics	Biology	4
BIO-399	Computational Biology	Biology	3
CS-101	Intro. to Computer Science	Comp. Sci.	4
CS-190	Game Design	Comp. Sci.	4
CS-315	Robotics	Comp. Sci.	3
CS-319	Image Processing	Comp. Sci.	3
CS-347	Database System Concepts	Comp. Sci.	3
EE-181	Intro. to Digital Systems	Elec. Eng.	3
FIN-201	Investment Banking	Finance	3
HIS-351	World History	History	3
MU-199	Music Video Production	Music	3
PHY-101	Physical Principles	Physics	4



ID	name	dept_name	salary
10101	Srinivasan	Comp. Sci.	65000.00
12121	Wu	Finance	90000.00
15151	Mozart	Music	40000.00
22222	Einstein	Physics	95000.00
32343	El Said	History	60000.00
33456	Gold	Physics	87000.00
45565	Katz	Comp. Sci.	75000.00
58583	Califieri	History	62000.00
76543	Singh	Finance	80000.00
76766	Crick	Biology	72000.00
83821	Brandt	Comp. Sci.	92000.00
98345	Kim	Elec. Eng.	80000.00

Natural Join

- Natural join matches tuples with the same values for all common attributes, and retains only one copy of each common column
  - *E.g.*, List the names of students along with the course ID of the courses that they took
    - **SELECT** *name*, *course_id*
      FROM *student*, *takes*
      WHERE *student.ID* = *takes.ID*;
  - Same query in SQL with natural join:
    - **SELECT** *name*, *course_id*
      FROM *student* **NATURAL JOIN** *takes*;



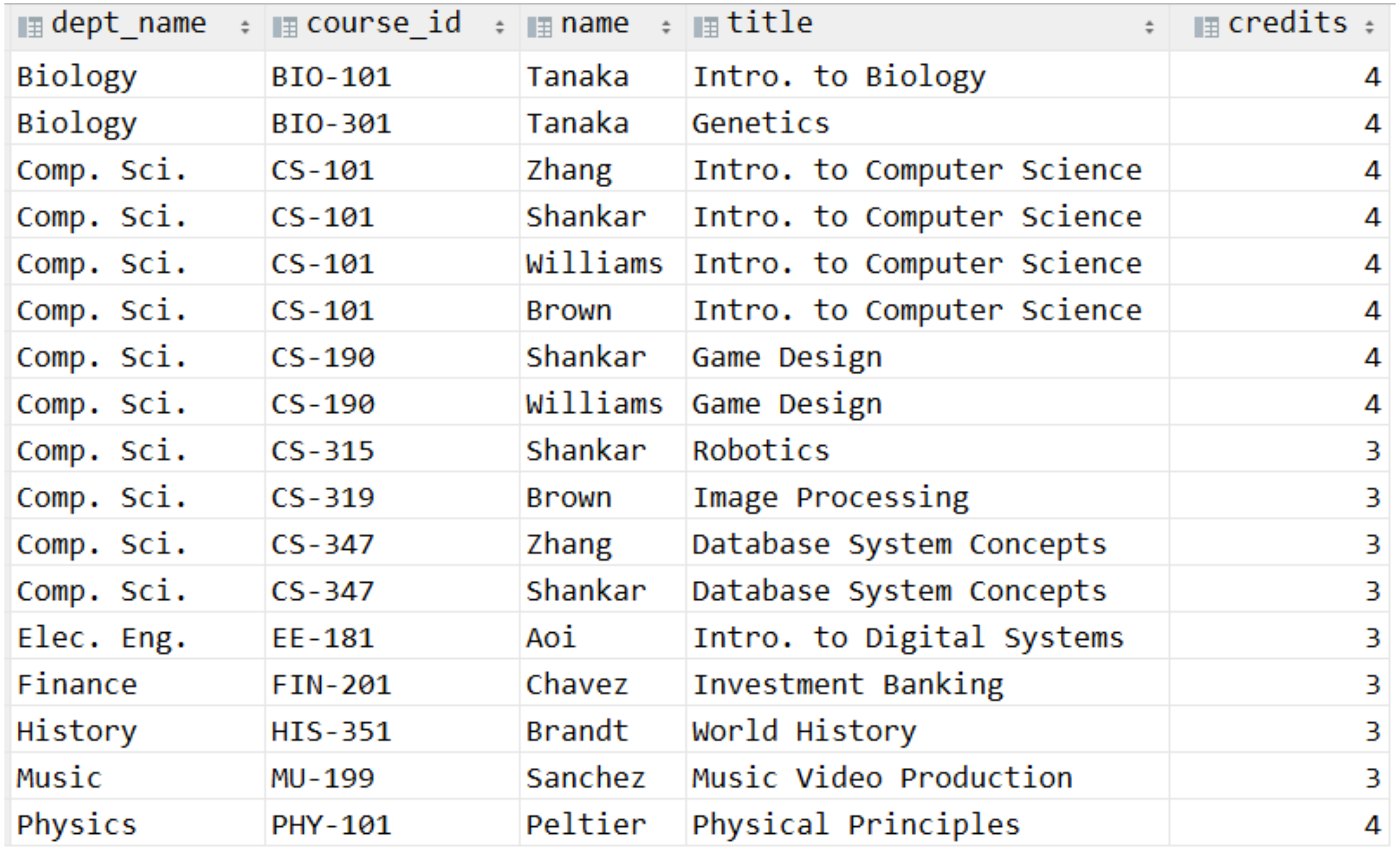
name	course_id
Zhang	CS-101
Zhang	CS-347
Shankar	CS-101
Shankar	CS-190
Shankar	CS-315
Shankar	CS-347
Brandt	HIS-351
Chavez	FIN-201
Peltier	PHY-101
Levy	CS-101
Levy	CS-101
Levy	CS-319
Williams	CS-101
Williams	CS-190
Sanchez	MU-199
Brown	CS-101
Brown	CS-319
Aoi	EE-181
Bourikas	CS-101
Bourikas	CS-315
Tanaka	BIO-101
Tanaka	BIO-301

Natural Join

- The **FROM** clause can have multiple relations combined using natural join:
  - **SELECT** $A_1, A_2, \ldots A_n$
    FROM r_1 **NATURAL JOIN** r_2 **NATURAL JOIN** … **NATURAL JOIN** r_n
    WHERE P ;

Caveat

- *E.g.*, **(Incorrect)**
  SELECT *dept_name, course_id, name, title, credits*
  FROM *student* **NATURAL JOIN** *takes* **NATURAL JOIN** *course*;



dept_name	course_id	name	title	credits
Biology	BIO-101	Tanaka	Intro. to Biology	4
Biology	BIO-301	Tanaka	Genetics	4
Comp. Sci.	CS-101	Zhang	Intro. to Computer Science	4
Comp. Sci.	CS-101	Shankar	Intro. to Computer Science	4
Comp. Sci.	CS-101	Williams	Intro. to Computer Science	4
Comp. Sci.	CS-101	Brown	Intro. to Computer Science	4
Comp. Sci.	CS-190	Shankar	Game Design	4
Comp. Sci.	CS-190	Williams	Game Design	4
Comp. Sci.	CS-315	Shankar	Robotics	3
Comp. Sci.	CS-319	Brown	Image Processing	3
Comp. Sci.	CS-347	Zhang	Database System Concepts	3
Comp. Sci.	CS-347	Shankar	Database System Concepts	3
Elec. Eng.	EE-181	Aoi	Intro. to Digital Systems	3
Finance	FIN-201	Chavez	Investment Banking	3
History	HIS-351	Brandt	World History	3
Music	MU-199	Sanchez	Music Video Production	3
Physics	PHY-101	Peltier	Physical Principles	4

Caveat

- Beware of unrelated attributes with same name getting equated incorrectly
  - *E.g.*, List the names of students along with the titles of courses that they have taken
    - Correct

      SELECT *name*, *title*
      FROM *student* **NATURAL JOIN** *takes*, *course*
      WHERE *takes*.*course_id* = *course*.*course_id*;

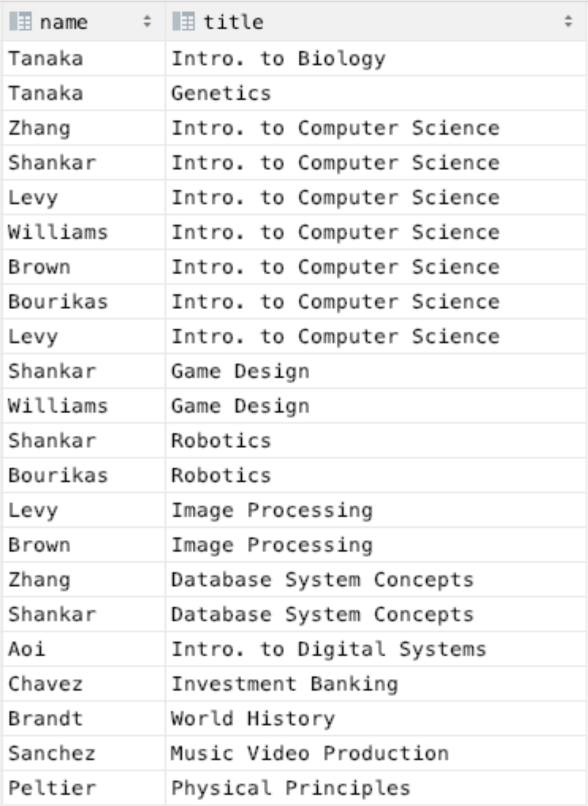
    - Incorrect

      SELECT *name*, *title*
      FROM *student* **NATURAL JOIN** *takes* **NATURAL JOIN** *course*;

      - This query omits all (student name, course title) pairs where the student takes a course in a department other than the student's own department

Natural Join with USING Clause

- To avoid the danger of equating attributes erroneously, use the **USING** construct
  - USING: allows us to specify exactly which columns should be equated
  - *E.g.*, **SELECT** *name*, *title*
    FROM (*student* **NATURAL JOIN** *takes*) **JOIN** *course* **USING** (*course_id*)



name	title
Tanaka	Intro. to Biology
Tanaka	Genetics
Zhang	Intro. to Computer Science
Shankar	Intro. to Computer Science
Levy	Intro. to Computer Science
Williams	Intro. to Computer Science
Brown	Intro. to Computer Science
Bourikas	Intro. to Computer Science
Levy	Intro. to Computer Science
Shankar	Game Design
Williams	Game Design
Shankar	Robotics
Bourikas	Robotics
Levy	Image Processing
Brown	Image Processing
Zhang	Database System Concepts
Shankar	Database System Concepts
Aoi	Intro. to Digital Systems
Chavez	Investment Banking
Brandt	World History
Sanchez	Music Video Production
Peltier	Physical Principles

JOIN … ON

- The **ON** condition allows a general predicate over the relations being joined
  - Written like a **WHERE** clause predicate
  - *E.g.*, **SELECT** *
    FROM *student* **JOIN** *takes* **ON** *student.ID* = *takes.ID*
    - The **ON** condition specifies that a tuple from *student* matches a tuple from *takes* if their *ID* values are equal
    - Equivalent to:
      SELECT *name*, *course_id*
      FROM *student, takes*
      WHERE *student.ID* = *takes.ID*;

Inner Join

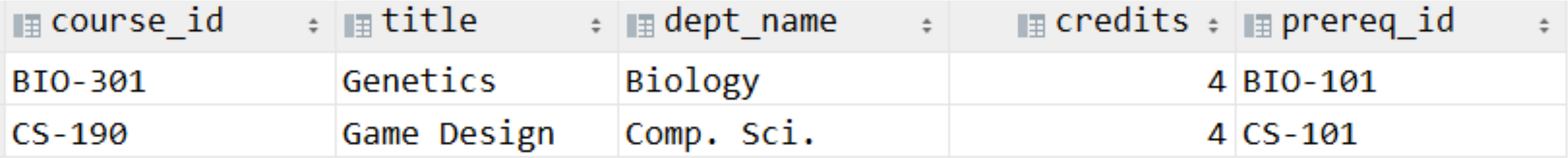
- Inner join: Does not preserve nonmatched tuples
  - Tables are joined based on common columns mentioned in the **ON** or **USING** clause
  - One can specify the condition with an **ON** or **USING** construct

- *C.f.*, Natural join: assumes the join condition to be where same-named columns in both tables match
  - One cannot use **ON** or **USING**
  - In the result of a natural join, repeated columns are avoided

Natural Join

- Natural join: Some tuples in either or both relations being joined may be lost

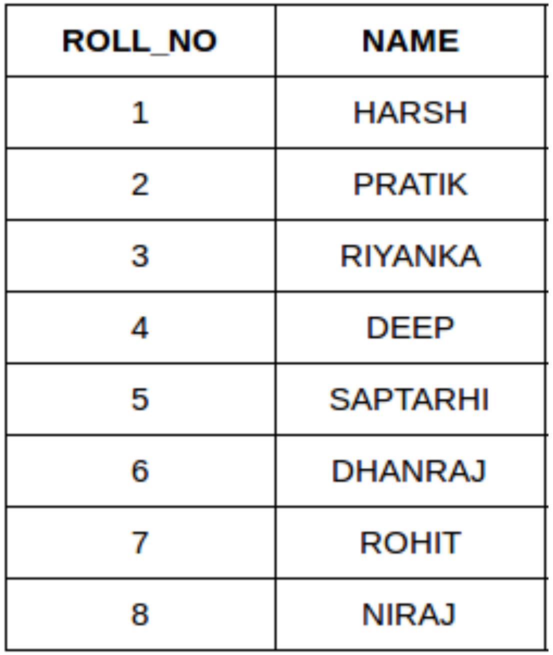
- **SELECT** *
  FROM *course* **NATURAL JOIN** *prereq*;



course_id	title	dept_name	credits	prereq_id
BIO-301	Genetics	Biology	4	BIO-101
CS-190	Game Design	Comp. Sci.	4	CS-101

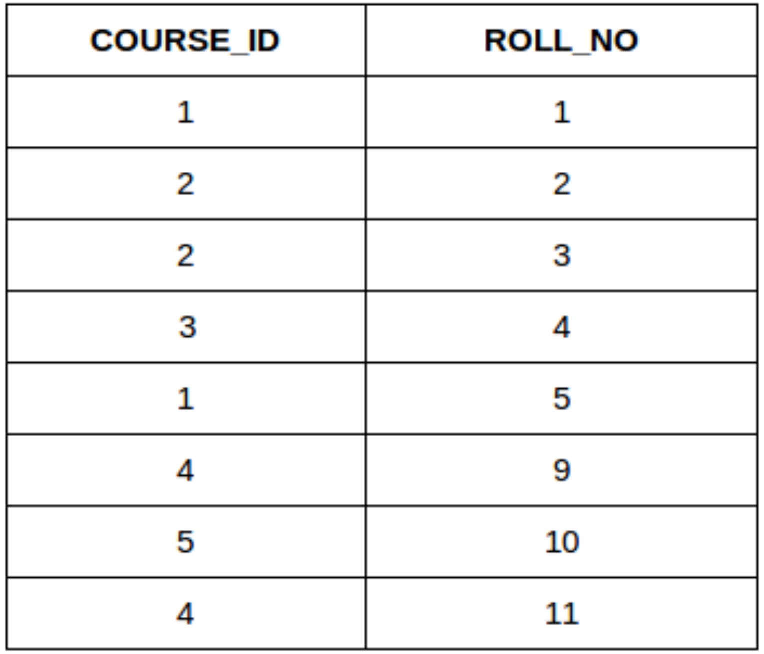
Examples

- Tables



ROLL_NO	NAME
1	HARSH
2	PRATIK
3	RIYANKA
4	DEEP
5	SAPTARHI
6	DHANRAJ
7	ROHIT
8	NIRAJ

Student



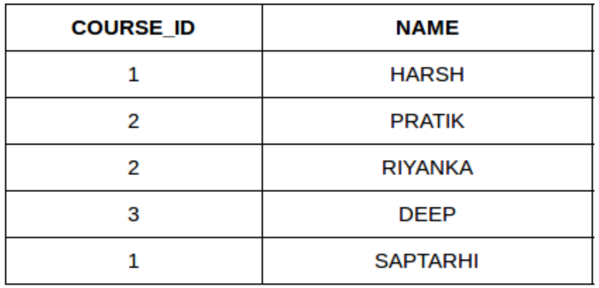
COURSE_ID	ROLL_NO
1	1
2	2
2	3
3	4
1	5
4	9
5	10
4	11

StudentCourse

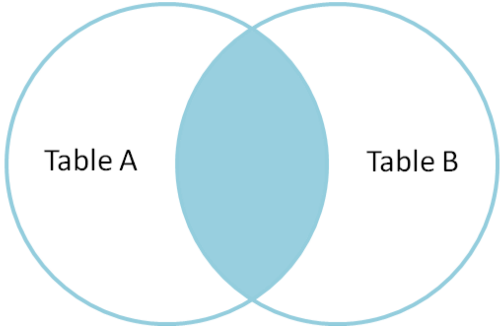
Examples

- Inner join

  -
    ```
    SELECT StudentCourse.COURSE_ID, Student.NAME
    FROM Student
    INNER JOIN StudentCourse
    ON Student.ROLL_NO = StudentCourse.ROLL_NO;
    ```



COURSE_ID	NAME
1	HARSH
2	PRATIK
2	RIYANKA
3	DEEP
1	SAPTARHI



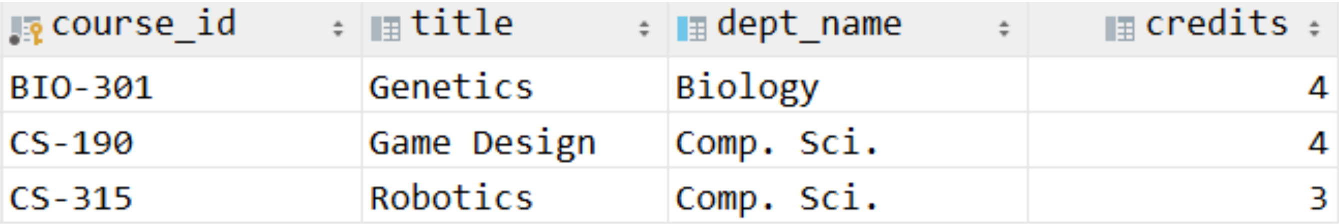
Outer Join

- An extension of the join operation that avoids loss of information
  - Outer join preserves those tuples that would be lost in a join by creating tuples in the result containing null values
  - Computes the join and then adds tuples form one relation that does not match tuples in the other relation to the result of the join

- Three forms of outer join:
  - **LEFT OUTER JOIN**
  - **RIGHT OUTER JOIN**
  - **FULL OUTER JOIN**

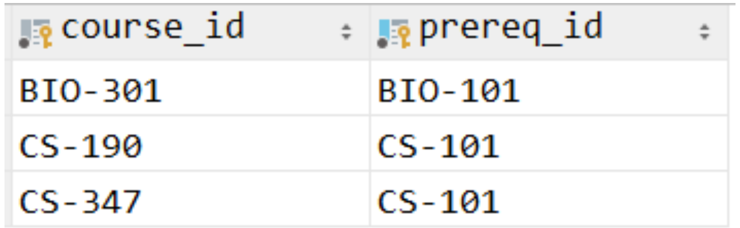
Running Example

- Relation *course*



course_id	title	dept_name	credits
BIO-301	Genetics	Biology	4
CS-190	Game Design	Comp. Sci.	4
CS-315	Robotics	Comp. Sci.	3

- Relation *prereq*



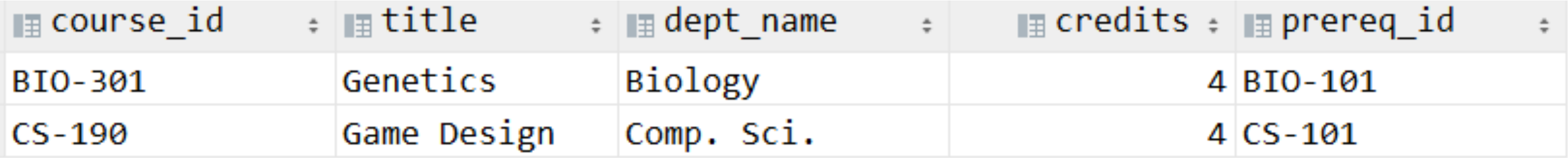
course_id	prereq_id
BIO-301	BIO-101
CS-190	CS-101
CS-347	CS-101

  - *course* is missing CS-347
  - *prereq* is missing CS-315

Inner Join with NATURAL

- Natural join: Some tuples in either or both relations being joined may be lost

- **SELECT ***
  FROM *course* **NATURAL JOIN** *prereq*;

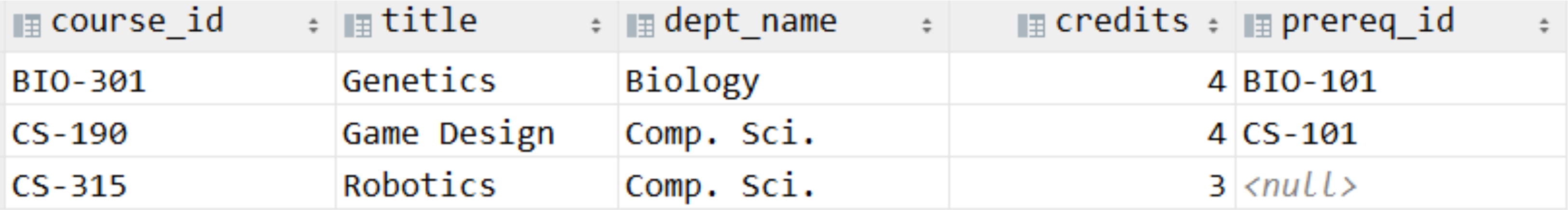


course_id	title	dept_name	credits	prereq_id
BIO-301	Genetics	Biology	4	BIO-101
CS-190	Game Design	Comp. Sci.	4	CS-101

Left Outer Join with NATURAL

- Left outer join: Preserves tuples only in the relation named before (to the left of) the operation

- **SELECT ***
  FROM *course* **NATURAL LEFT OUTER JOIN** *prereq*;

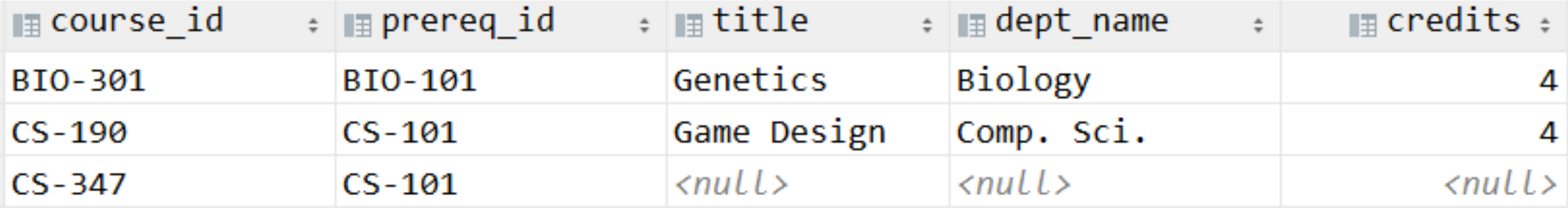


course_id	title	dept_name	credits	prereq_id
BIO-301	Genetics	Biology	4	BIO-101
CS-190	Game Design	Comp. Sci.	4	CS-101
CS-315	Robotics	Comp. Sci.	3	<null>

Right Outer Join with NATURAL

- Right outer join: Preserves tuples only in the relation named after (to the right of) the operation

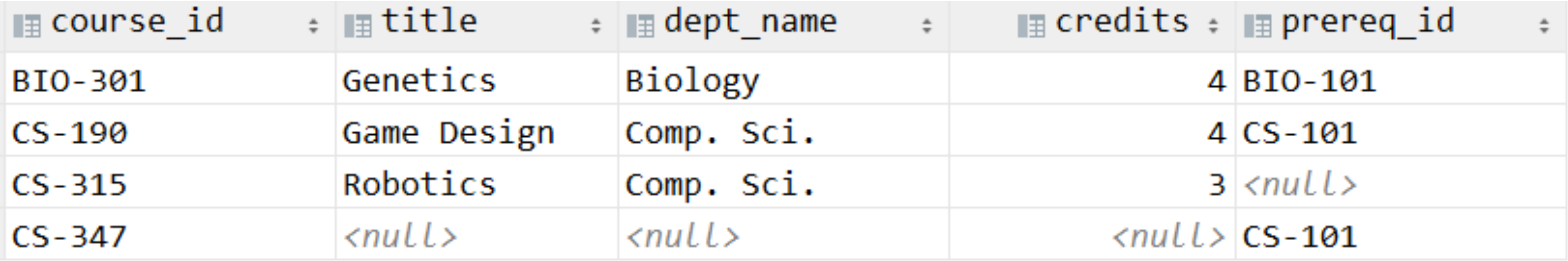
- **SELECT ***
  FROM *course* **NATURAL RIGHT OUTER JOIN** *prereq*;



course_id	prereq_id	title	dept_name	credits
BIO-301	BIO-101	Genetics	Biology	4
CS-190	CS-101	Game Design	Comp. Sci.	4
CS-347	CS-101	<null>	<null>	<null>

Full Outer Join with NATURAL

- **SELECT ***
  FROM *course* **NATURAL FULL OUTER JOIN** *prereq;*

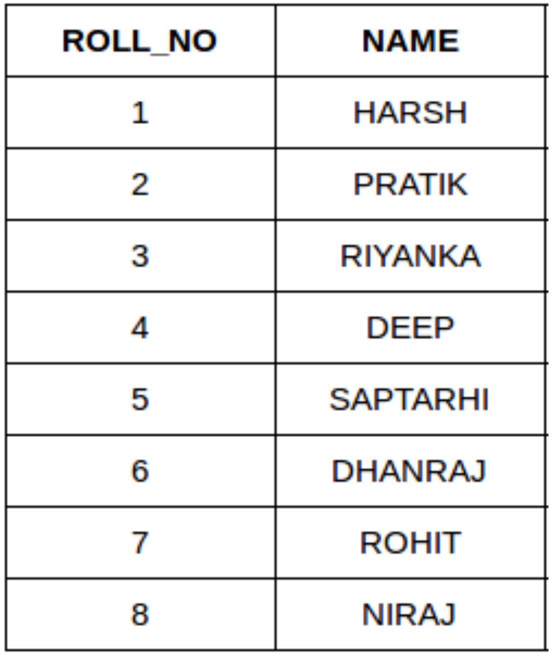


course_id	title	dept_name	credits	prereq_id
BIO-301	Genetics	Biology	4	BIO-101
CS-190	Game Design	Comp. Sci.	4	CS-101
CS-315	Robotics	Comp. Sci.	3	<null>
CS-347	<null>	<null>	<null>	CS-101

  - MySQL does NOT support FULL join
    - Alternative: use the **UNION** of left and right joins
      SELECT course_id, title, dept_name, credits, prereq_id
      FROM course **NATURAL LEFT OUTER JOIN** prereq
      UNION
      SELECT course_id, title, dept_name, credits, prereq_id
      FROM course **NATURAL RIGHT OUTER JOIN** prereq;
      - In order to perform UNION properly, the attributes of both join queries must be aligned

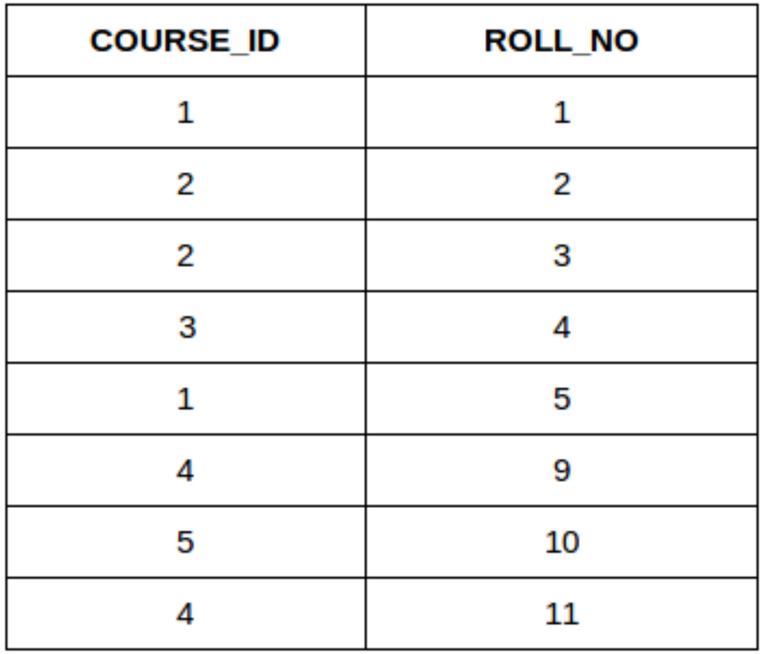
Examples

- Tables



ROLL_NO	NAME
1	HARSH
2	PRATIK
3	RIYANKA
4	DEEP
5	SAPTARHI
6	DHANRAJ
7	ROHIT
8	NIRAJ

Student



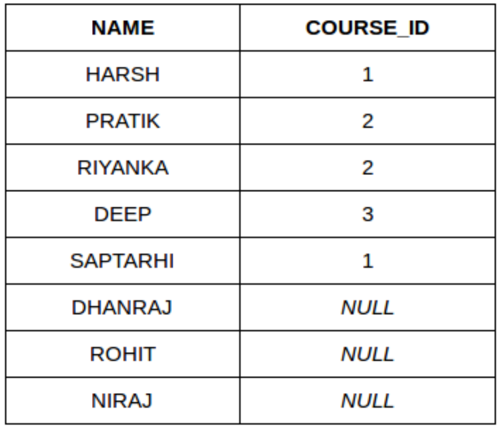
COURSE_ID	ROLL_NO
1	1
2	2
2	3
3	4
1	5
4	9
5	10
4	11

StudentCourse

Examples

- Left join

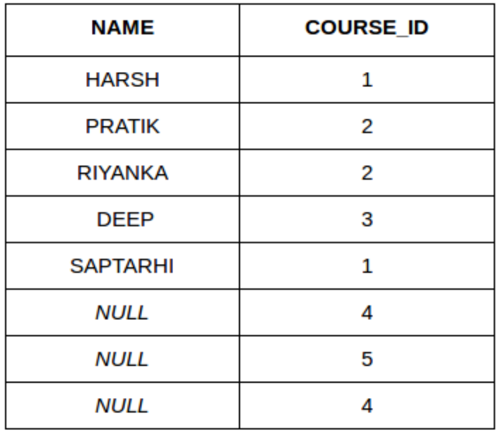
  -
    ```
    SELECT Student.NAME, StudentCourse.COURSE_ID
    FROM Student
    LEFT JOIN StudentCourse
    ON StudentCourse.ROLL_NO = Student.ROLL_NO;
    ```



NAME	COURSE_ID
HARSH	1
PRATIK	2
RIYANKA	2
DEEP	3
SAPTARHI	1
DHANRAJ	*NULL*
ROHIT	*NULL*
NIRAJ	*NULL*

- Right join

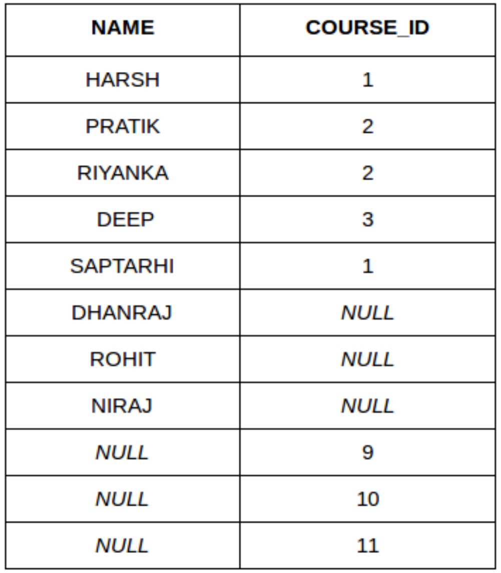
  -
    ```
    SELECT Student.NAME, StudentCourse.COURSE_ID
    FROM Student
    RIGHT JOIN StudentCourse
    ON StudentCourse.ROLL_NO = Student.ROLL_NO;
    ```



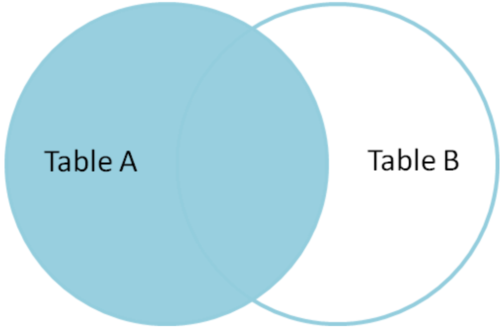
NAME	COURSE_ID
HARSH	1
PRATIK	2
RIYANKA	2
DEEP	3
SAPTARHI	1
NULL	4
NULL	5
NULL	4

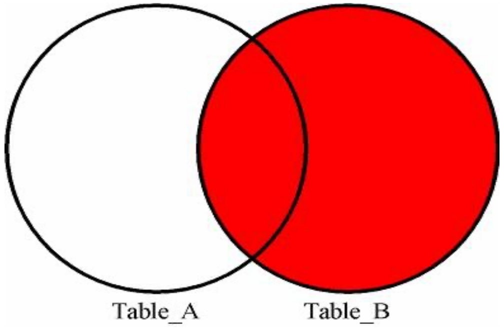
Examples

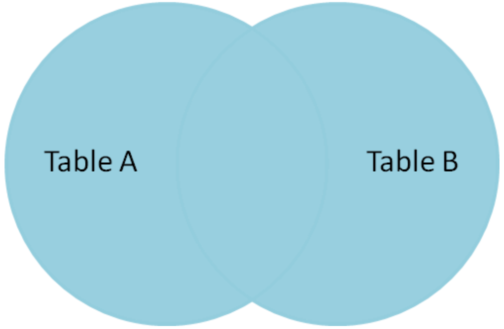
- Full join
  - **SELECT** *Student.NAME, StudentCourse.COURSE_ID*
FROM *Student*
FULL JOIN *StudentCourse*
ON *StudentCourse.ROLL_NO = Student.ROLL_NO;*



NAME	COURSE_ID
HARSH	1
PRATIK	2
RIYANKA	2
DEEP	3
SAPTARHI	1
DHANRAJ	*NULL*
ROHIT	*NULL*
NIRAJ	*NULL*
NULL	9
NULL	10
NULL	11





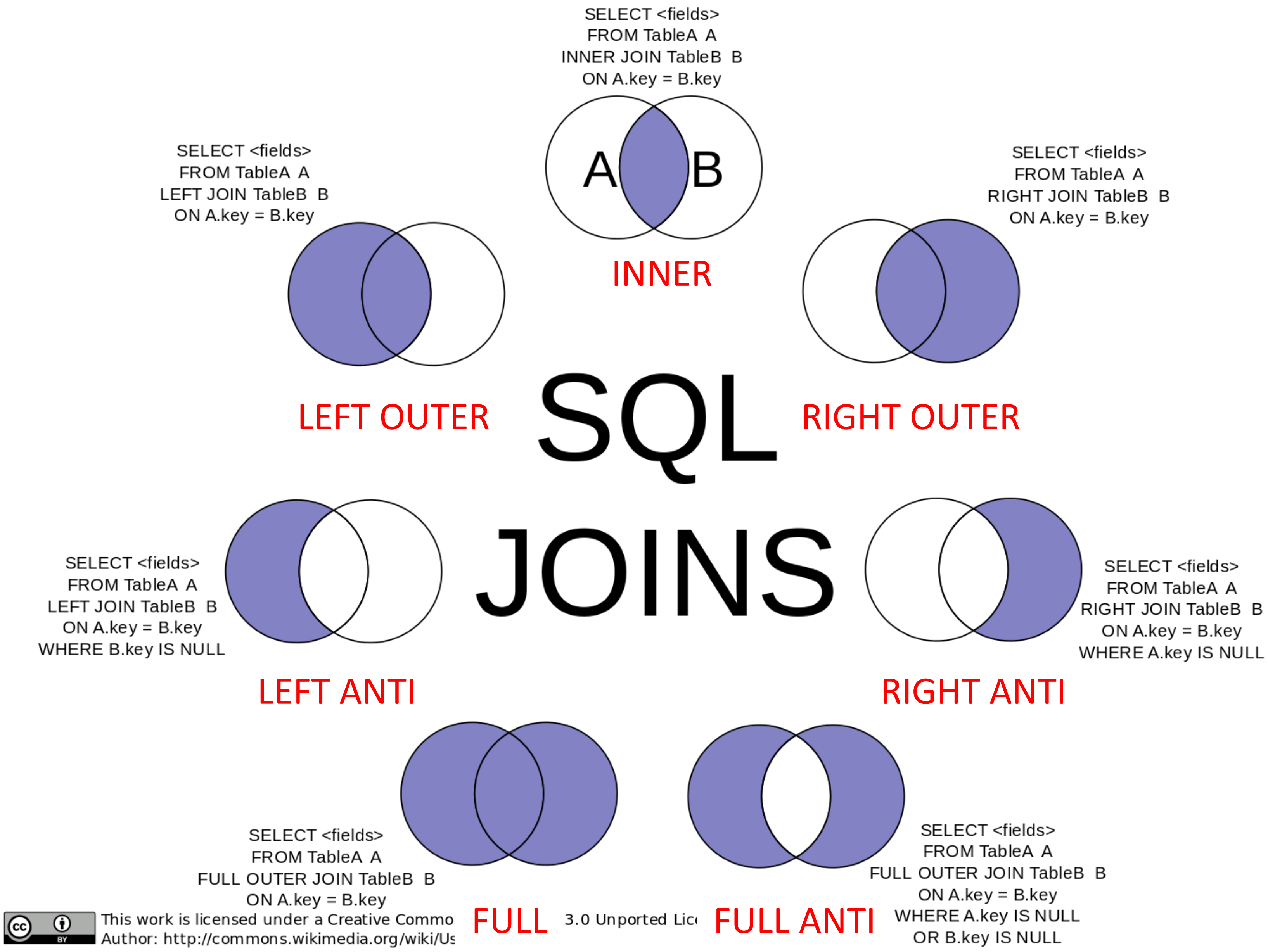


Join Types and Conditions

- Join type: Defines how tuples in each relation that do not match any tuples in the other relation are treated
  - **INNER JOIN**
  - **LEFT OUTER JOIN**
  - **RIGHT OUTER JOIN**
  - **FULL OUTER JOIN**

- Join condition: Defines which tuples in the two relations match
  - **NATURAL**
  - **ON** <predicate>
  - **USING (**A_1, A_2, …, A_n**)**

Join Types



INNER

```
SELECT <fields>
FROM TableA  A
INNER JOIN TableB  B
ON A.key = B.key
```

LEFT OUTER

```
SELECT <fields>
FROM TableA  A
LEFT JOIN TableB  B
ON A.key = B.key
```

RIGHT OUTER

```
SELECT <fields>
FROM TableA  A
RIGHT JOIN TableB  B
ON A.key = B.key
```

LEFT ANTI

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

RIGHT ANTI

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

FULL

```
SELECT <fields>
FROM TableA  A
FULL OUTER JOIN TableB  B
ON A.key = B.key
```

FULL ANTI

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

Join Condition

- Join condition
  - **NATURAL**: Joins two tables based on same attribute name and datatypes
    - **SELECT** * **FROM** *course* **NATURAL JOIN** *prereq;*
  - **ON** <predicate>: Joins two tables based on the column(s) explicitly specified in the **ON** clause
    - **SELECT** * **FROM** *course*
JOIN *prereq* **ON** *course.course_id = prereq.prereq_id*;
  - **USING** (A_1, A_2, …, A_n): Joins two tables based on common attribute name(s) listed next to **USING**
    - **SELECT** * **FROM** *course*
JOIN *prereq* **USING** (*course_id*)

Inner Join vs. Natural Join

INNER JOIN	NATURAL JOIN
Joins two tables on the basis of the column which is explicitly specified in the ON clause	Joins two tables based on same attribute name and datatypes
The resulting table will contain all the attribute of both the tables (including duplicate columns)	The resulting table will contain all the attribute of both the tables but keep only one copy of each common column
Only those records will return which exists in both tables	If there is no indication of LEFT, RIGHT, or FULL, it returns the rows based on the common column

* Source: https://www.geeksforgeeks.org/difference-between-natural-join-and-inner-join-in-sql/

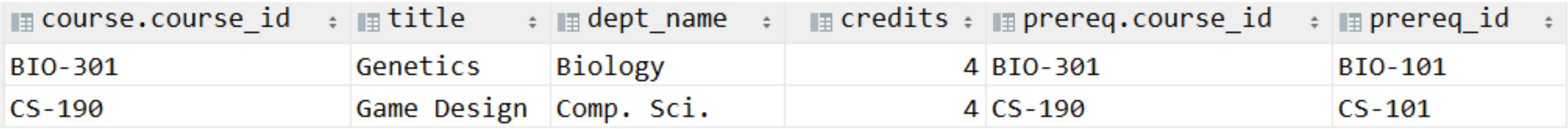
Inner Join vs. Natural Join

- Inner join
  - **SELECT** * **FROM** *course*
    INNER JOIN *prereq* **ON** *course.course_id = prereq.prereq_id*;

- Natural join
  - **SELECT** *
    FROM *course* **NATURAL JOIN** *prereq*
    ON *course.course_id = prereq.prereq_id*; ← NOT VALID!

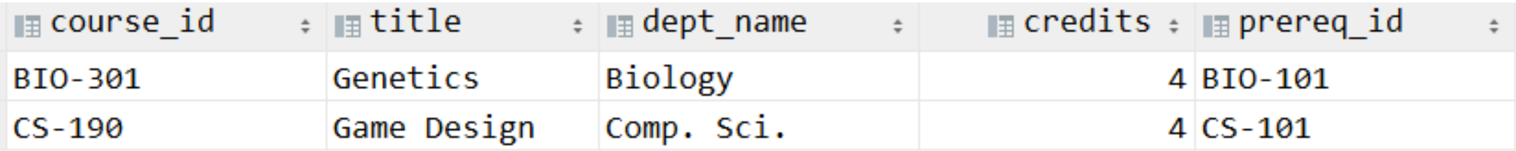
Inner Join vs. Natural Join

- Inner join
    - **SELECT * FROM** *course*
      INNER JOIN *prereq* **ON** *course.course_id = prereq.course_id*;
        - Equivalent to:
          SELECT * FROM *course*
          JOIN *prereq* **ON** *course.course_id = prereq.course_id*;



course.course_id	title	dept_name	credits	prereq.course_id	prereq_id
BIO-301	Genetics	Biology	4	BIO-301	BIO-101
CS-190	Game Design	Comp. Sci.	4	CS-190	CS-101

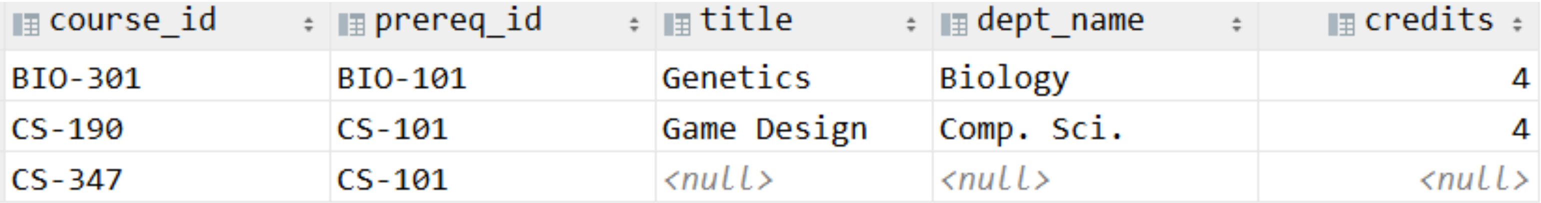
- Natural join
    - **SELECT ***
      FROM *course* **NATURAL JOIN** *prereq*;



course_id	title	dept_name	credits	prereq_id
BIO-301	Genetics	Biology	4	BIO-101
CS-190	Game Design	Comp. Sci.	4	CS-101

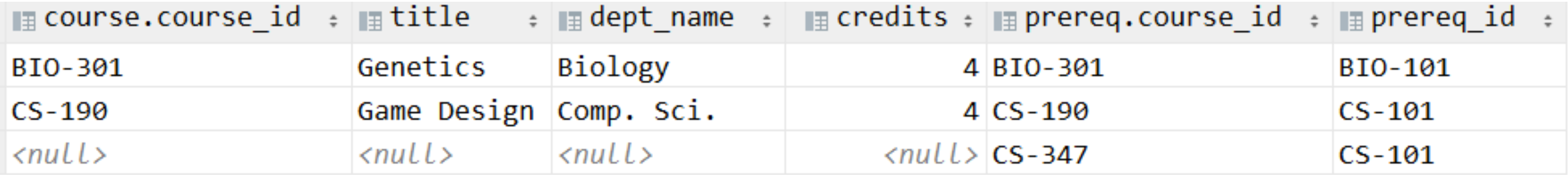
Outer Join vs. Natural Join

- Right outer join
  - **SELECT ***
    FROM *course* **NATURAL RIGHT OUTER JOIN** *prereq*;
    - Equivalent to:
      SELECT *
      FROM *course* **RIGHT OUTER JOIN** *prereq*
      USING (*course_id*);



course_id	prereq_id	title	dept_name	credits
BIO-301	BIO-101	Genetics	Biology	4
CS-190	CS-101	Game Design	Comp. Sci.	4
CS-347	CS-101	<null>	<null>	<null>

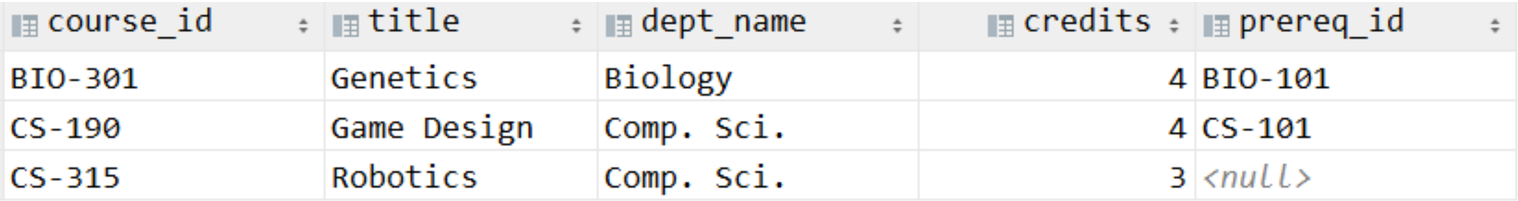
- **SELECT ***
  FROM *course* **RIGHT OUTER JOIN** *prereq*
  ON *course.course_id* = *prereq.course_id*;



course.course_id	title	dept_name	credits	prereq.course_id	prereq_id
BIO-301	Genetics	Biology	4	BIO-301	BIO-101
CS-190	Game Design	Comp. Sci.	4	CS-190	CS-101
<null>	<null>	<null>	<null>	CS-347	CS-101

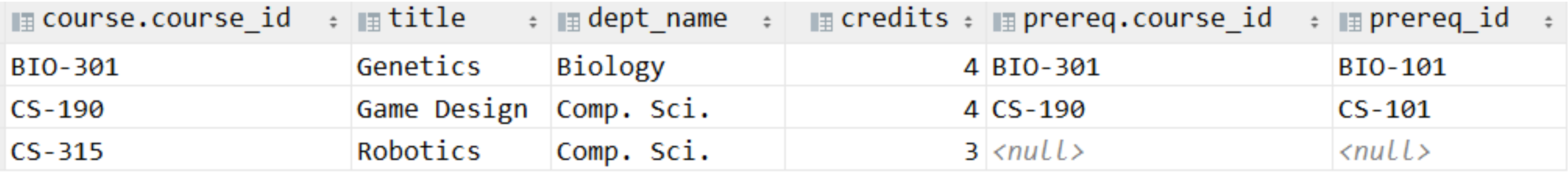
Outer Join vs. Natural Join

- Left outer join
  - **SELECT ***
    FROM *course* **NATURAL LEFT OUTER JOIN** *prereq*;
    - Equivalent to:
      SELECT *
      FROM *course* **LEFT OUTER JOIN** *prereq*
      USING (*course_id*);



course_id	title	dept_name	credits	prereq_id
BIO-301	Genetics	Biology	4	BIO-101
CS-190	Game Design	Comp. Sci.	4	CS-101
CS-315	Robotics	Comp. Sci.	3	<null>

- **SELECT ***
  FROM *course* **LEFT OUTER JOIN** *prereq*
  ON *course.course_id* = *prereq.course_id*;



course.course_id	title	dept_name	credits	prereq.course_id	prereq_id
BIO-301	Genetics	Biology	4	BIO-301	BIO-101
CS-190	Game Design	Comp. Sci.	4	CS-190	CS-101
CS-315	Robotics	Comp. Sci.	3	<null>	<null>

Natural Joins Are Often Avoided

- *Natural joins are often avoided in practice*, because:
  - Natural joins are not particularly readable (by most SQL coders) and possibly not supported by various tools/libraries
  - Natural joins are not informative; you cannot tell what columns are being joined on without referring to the schema
  - Your join conditions are invisibly vulnerable to schema changes
    - Even if there are multiple natural join columns and one such column is removed from a table, the query will still execute
    - But the result may not be correct and this change in behavior will be silent
  - Hardly worth the effort; you are only saving about 10 seconds by not typing specific conditions

ECE30030/ITP30010 Database Systems

Term Project

Charmgil Hong

charmgil@handong.edu

Spring, 2023

Handong Global University

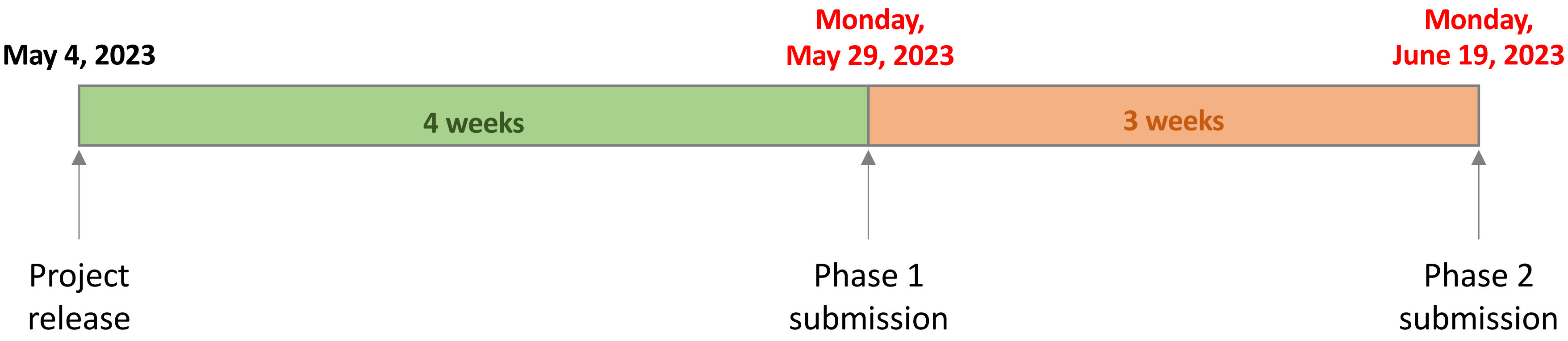
Term Project

- Goals
  - To practice the concepts and underlying mechanisms of database management system with an actual database instance
  - To represent database designs in modeling languages and analyze the designs with respect to given constraints
  - To articulate the relational database language (structured query language)
  - To exercise the optimization and evaluation of the database performance

- In this project, each team will be given a large chunk of data that is completely unnormalized
  - Your objective is to design a "good" database schema that can accommodate the provided data without any loss of information
    - "Good" in that…
      - Efficient in terms of space and time complexity

Term Project Overview

- Planned timeline



- Phase 1 "space" submission: Monday, May 29, 2023
- Phase 2 "time" submission: Monday, June 19, 2023

KUBiC: Korean Unification Bigdata Center

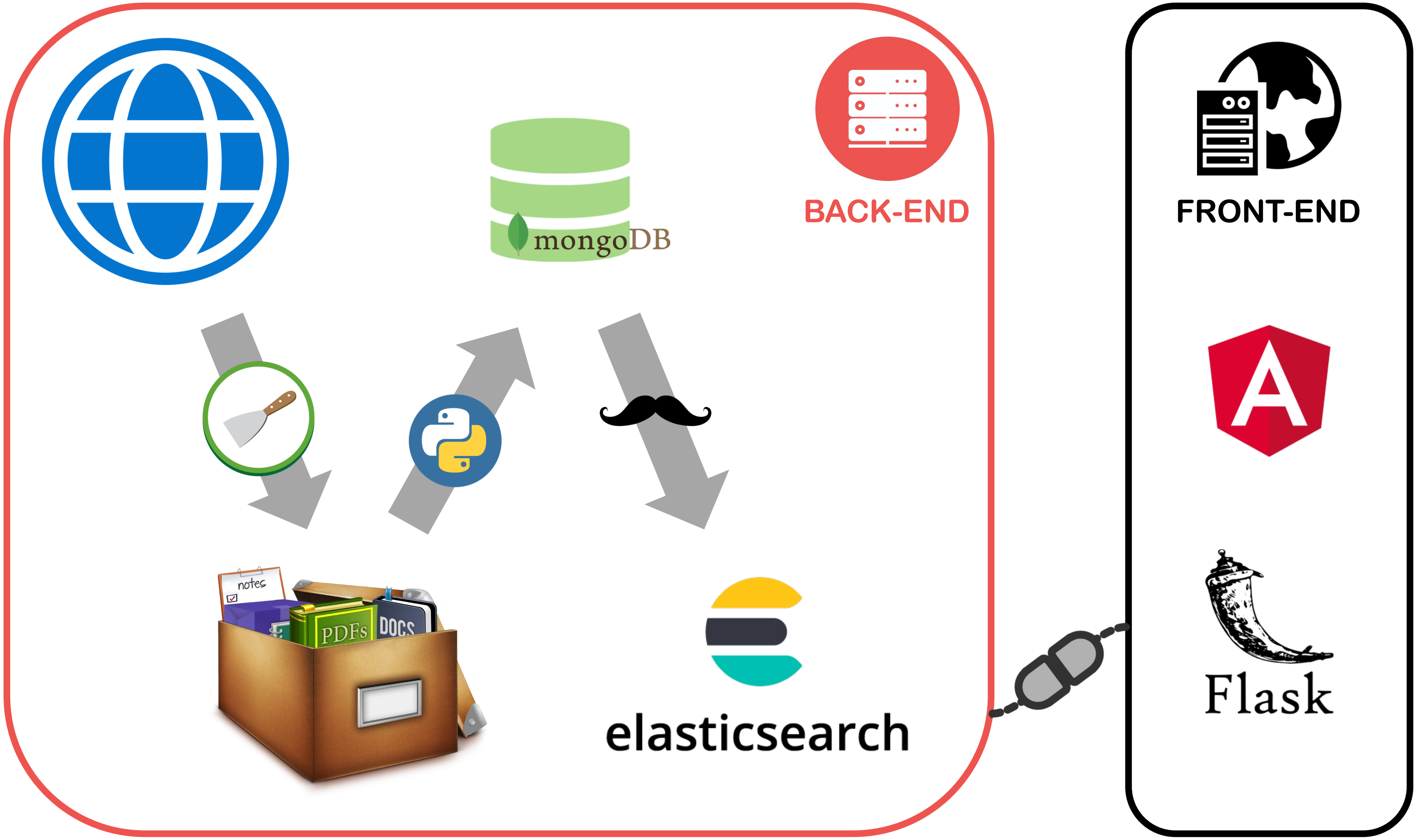
- **Term-Project data is provided by the KUBiC project team**
- A government-funded project on a data-center development focusing on the Korean unification
  - URL: https://kubic.handong.edu/
  - Data archive + search engine + web-based analysis tools, specialized on the Korean unification and North Korea research
  - Contains a lot of academic papers and government reports on the relevant topics

KUBiC
Korea Unification Bigdata Center

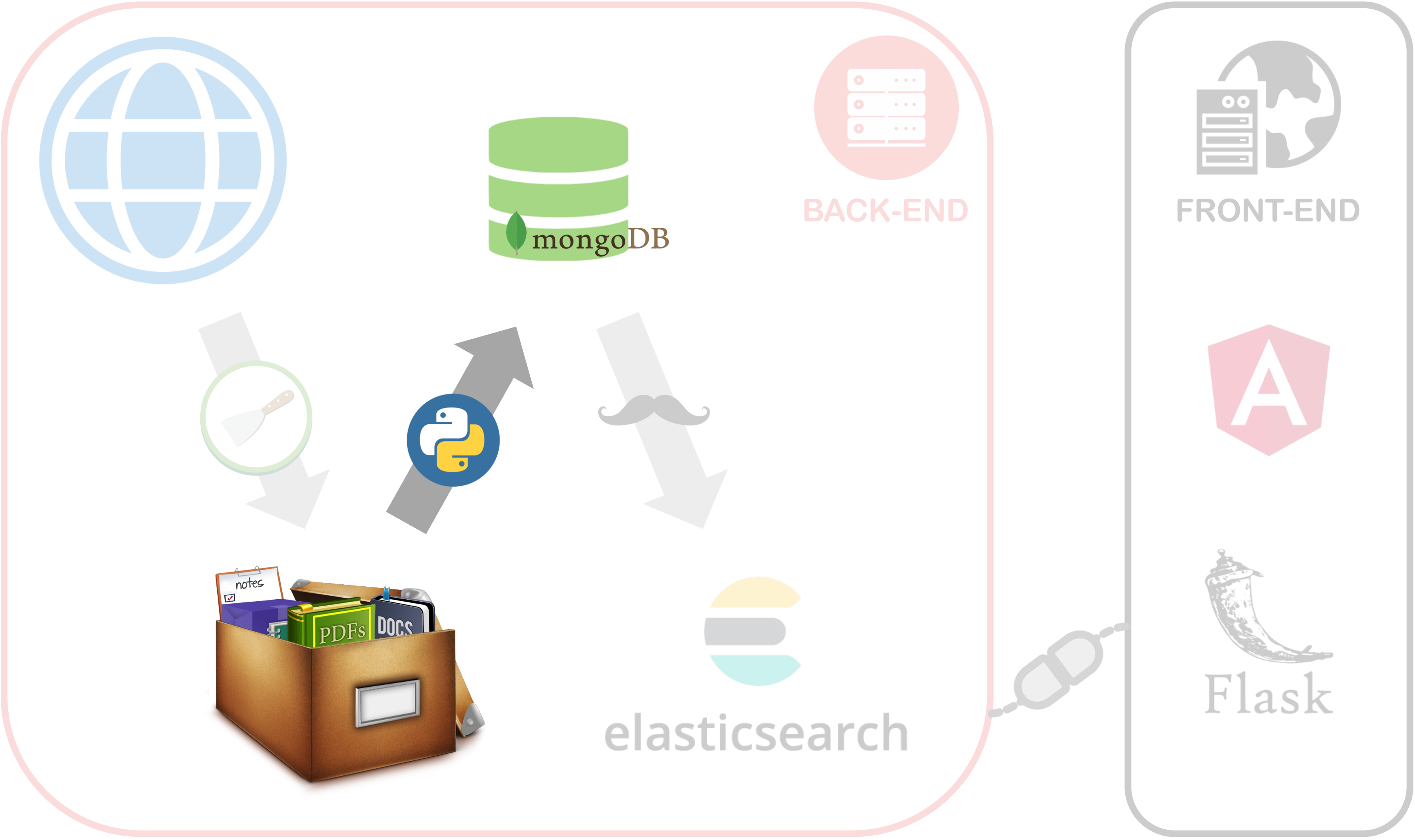
한동대학교
HANDONG GLOBAL UNIVERSITY

통일부
Ministry of Unification

KUBiC: Korean Unification Bigdata Center



KUBiC: Korean Unification Bigdata Center

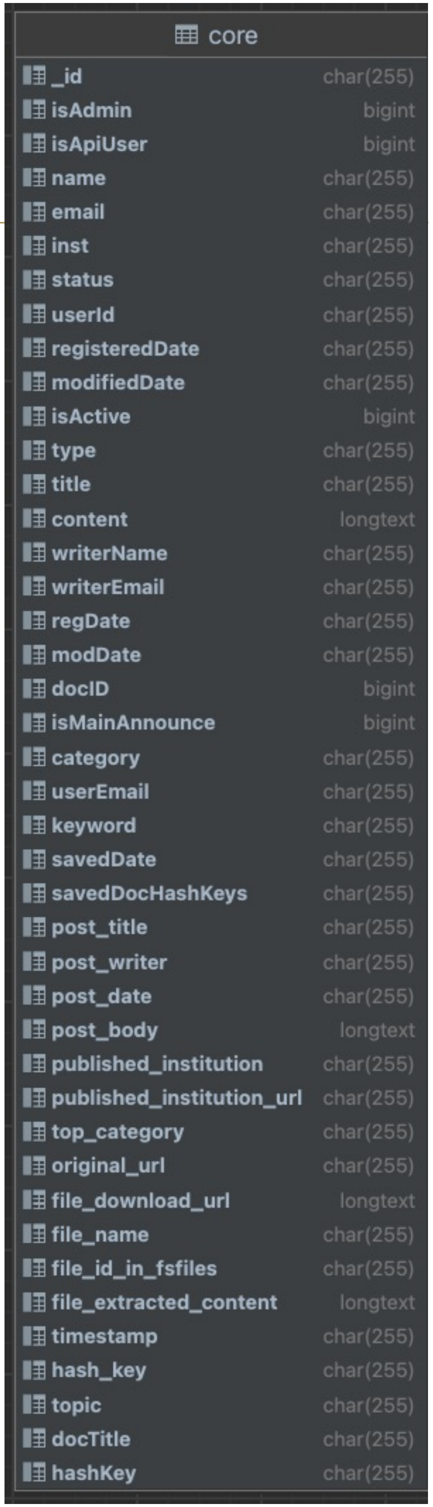


Term Project

- Background
  - You will be given large chunks of data snapshot from the KUBIC database, that consist of one SQL dump file and two csv files
    - core.sql
      - 116,320 records, 42 columns (approx. 2.45 GB)
      - Completely unnormalized
    - tfidf.csv
      - TF-IDF analysis of the service documents
      - 877,490 records, 4 columns (approx. 170.6 MB)
    - rcmd.csv
      - Cosince similarity analysis of the service documents
      - 1,000,000 records, 3 columns (approx. 126.8 MB)
  - SQL dump file: Ordinary text file, written in the SQL syntax
    - Contains a record of the table structure and/or the data from a database
    - Often used for backing up a database so that its contents can be restored in the event of data loss

Provided Data

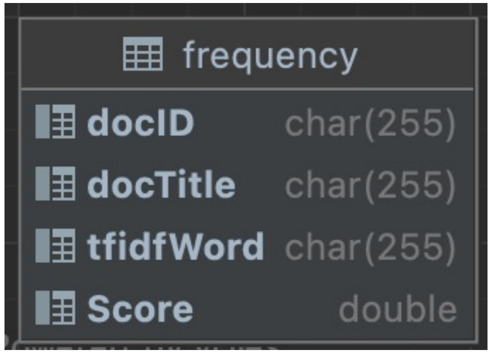
- Core
  - Collection of core meta-data about the web-documents that KUBIC contains
  - Also contains the bulletin boards, user information, saved documents of each user
  - 116,320 records, 42 columns (2.45GB)
  - Completely unnormalized



core	
_id	char(255)
isAdmin	bigint
isApiUser	bigint
name	char(255)
email	char(255)
inst	char(255)
status	char(255)
userId	char(255)
registeredDate	char(255)
modifiedDate	char(255)
isActive	bigint
type	char(255)
title	char(255)
content	longtext
writerName	char(255)
writerEmail	char(255)
regDate	char(255)
modDate	char(255)
docID	bigint
isMainAnnounce	bigint
category	char(255)
userEmail	char(255)
keyword	char(255)
savedDate	char(255)
savedDocHashKeys	char(255)
post_title	char(255)
post_writer	char(255)
post_date	char(255)
post_body	longtext
published_institution	char(255)
published_institution_url	char(255)
top_category	char(255)
original_url	char(255)
file_download_url	longtext
file_name	char(255)
file_id_in_fsfiles	char(255)
file_extracted_content	longtext
timestamp	char(255)
hash_key	char(255)
topic	char(255)
docTitle	char(255)
hashKey	char(255)

Provided Data

- tfidf
  - TF-IDF analysis of the service documents
  - For Phase 1, this table is supposed to be empty

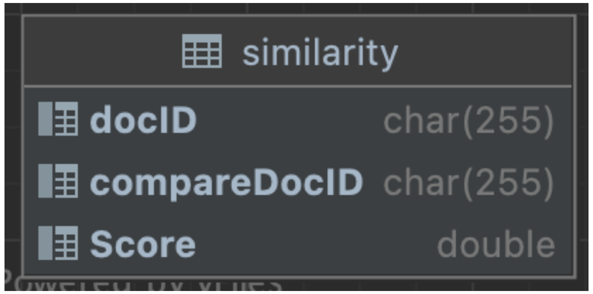


frequency

docID	char(255)
docTitle	char(255)
tfidfWord	char(255)
Score	double

Provided Data

- simscores
  - Cosince similarity analysis of the service documents
  - For Phase 1, this table is supposed to be empty



similarity

docID	char(255)
compareDocID	char(255)
Score	double

Term Project

- *Phase 1 requirements*
  - *Design and implment a database that can effectively accommodate the entire data without any loss*
    - *You and your team will need to draw E-R diagrams and conduct a number of normalization processes*
  - *Import the data; there should be no missing portion*
    - *You will be asked to create and submit views*
  - *Make the database size as small as possible!*

- *Phase 2 requirements*
  - *Optimize the database using*
    - *Denormalization*
    - *Indexing*

Data Files

- Core
  - https://drive.google.com/file/d/1BUTHZv0AgZPUEaOna3IoxUklSXO8VfZ5/view?usp=sharing

- Tfidf
  - https://drive.google.com/file/d/1MUNteBF58NZHNLOf31ZN90BkE0MSUS8H/view?usp=sharing

- Rcmds
  - https://drive.google.com/file/d/14QpCNHPQEucieDK6iWBYjY_Xflz2DWKW/view?usp=sharing

Technical Resources

- Upon completion, submit your result to LMS. Each submission should have the following items:
  - Dump of the database (in `.sql`)
    - How to create a SQL dump?
      - https://dev.mysql.com/doc/refman/8.0/en/mysqldump.html
      - https://dev.mysql.com/doc/refman/8.0/en/mysqldump-sql-format.html
  - Report Documents (in `.pdf`)
    - How to attack this problem?
    - DDL query and result for View instruction
    - ER Diagram of your database
  - Submission should be one `.zip` file

TA’s are up for help

- Chanju Lee (이찬주), Seohwee (박서휘): Data-specific questions
- Jihyeon Song (송지현), Harim Kim (김하림): SQL and DBMS functionalities-related questions

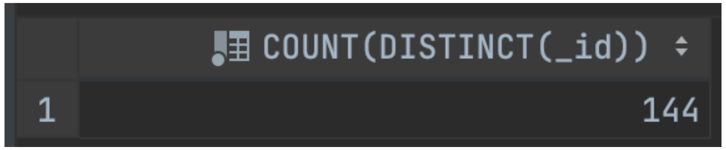
Phase 1: Database Modeling

- Goal: Design and implement a database instance that is efficient in space
  - You are expected to conduct a database design using ERD and apply the normalization theory
  - We will check the correctness and completeness of your data by examining the output of the views suggested in next slides
  - The database size on the physical storage will be estimated; the smallest 10% teams will earn bonus points (maximum +7%)

  - Before the submission, each team is expected to run several iterations of design, implement, data import, and internal evaluation

Phase 1: Database Modeling

- Views to create (and submit)
  1. View: userCount
     - Count the number of users in the database
     - **SELECT** * **FROM** userCount

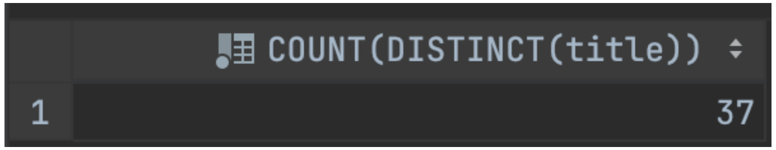


	COUNT(DISTINCT(_id))
1	144

  - The column name may vary

Phase 1: Database Modeling

- Views to create (and submit)
  2. View: boardCount
     - Count the number of bulletins on the board
     - **SELECT** * **FROM** boardCount

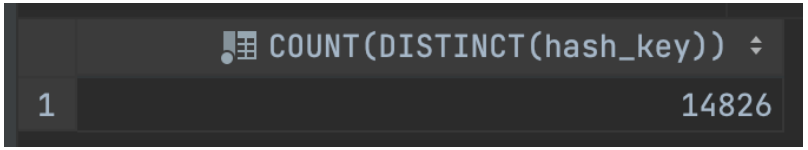


	COUNT(DISTINCT(title))
1	37

- The column name may vary

Phase 1: Database Modeling

- Views to create (and submit)
  3. View: docCount
     - Count the number of documents that are stored
     - **SELECT** * **FROM** docCount



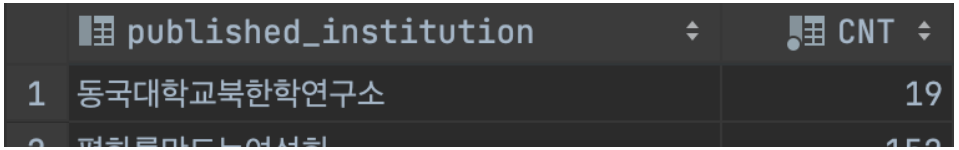
	COUNT(DISTINCT(hash_key))
1	14826

  - The column name may vary

Phase 1: Database Modeling

- Views to create (and submit)

  4. View: instPubInfo
     - List the names of publisher institutes and their numbers of publications (sort the results in ascending order of the number of publications)
     - **SELECT** * **FROM** instPubInfo



	published_institution	CNT
1	동국대학교북한학연구소	19
2	평화를만드는여성회	152

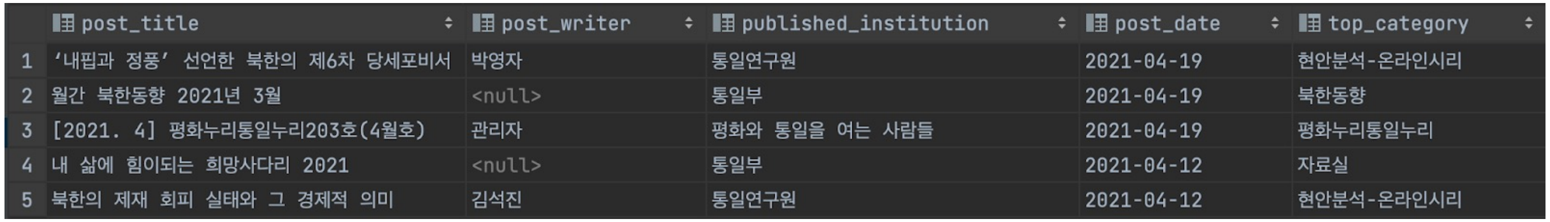
: :

: :

Phase 1: Database Modeling

- Views to create (and submit)

  5. View: docInfo
     - List the posting title, post author name and affiliation, posted date, and top category tag
     - **SELECT * FROM** docInfo



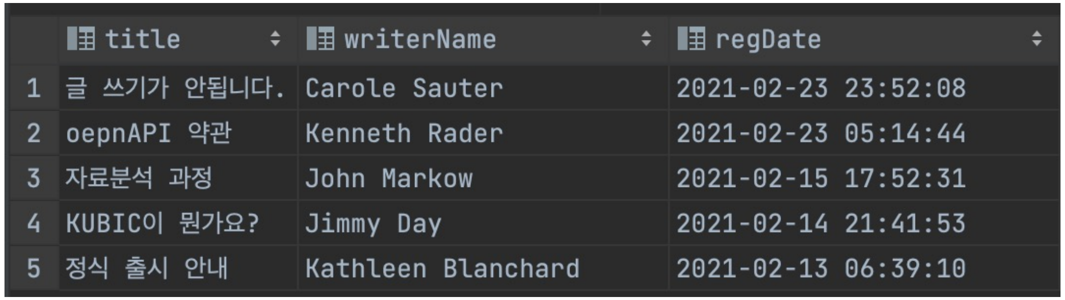
	post_title	post_writer	published_institution	post_date	top_category
1	‘내핍과 정풍’ 선언한 북한의 제6차 당세포비서	박영자	통일연구원	2021-04-19	현안분석-온라인시리
2	월간 북한동향 2021년 3월	<null>	통일부	2021-04-19	북한동향
3	[2021. 4] 평화누리통일누리203호(4월호)	관리자	평화와 통일을 여는 사람들	2021-04-19	평화누리통일누리
4	내 삶에 힘이되는 희망사다리 2021	<null>	통일부	2021-04-12	자료실
5	북한의 제재 회피 실태와 그 경제적 의미	김석진	통일연구원	2021-04-12	현안분석-온라인시리

:   :

:   :

Phase 1: Database Modeling

- Views to create (and submit)
  6. View: bulletinSummary
     - List all bulletin titles, author names (writer names), and posted dates
     - **SELECT** * **FROM** bulletinSummary

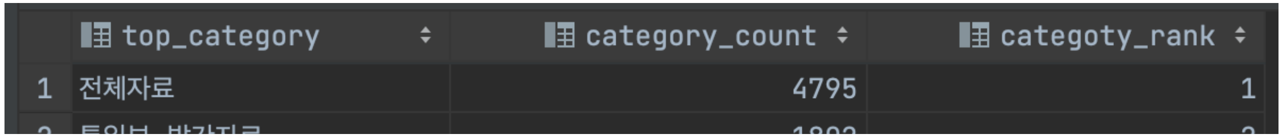


	title	writerName	regDate
1	글 쓰기가 안됩니다.	Carole Sauter	2021-02-23 23:52:08
2	oepnAPI 약관	Kenneth Rader	2021-02-23 05:14:44
3	자료분석 과정	John Markow	2021-02-15 17:52:31
4	KUBIC이 뭔가요?	Jimmy Day	2021-02-14 21:41:53
5	정식 출시 안내	Kathleen Blanchard	2021-02-13 06:39:10

⋮ ⋮

Phase 1: Database Modeling

- Views to create (and submit)
  7. View: docSummary
     - Count the number of documents per each of top category values; show the results in descending order of the counts and put their ranks
     - **SELECT** * **FROM** docSummary



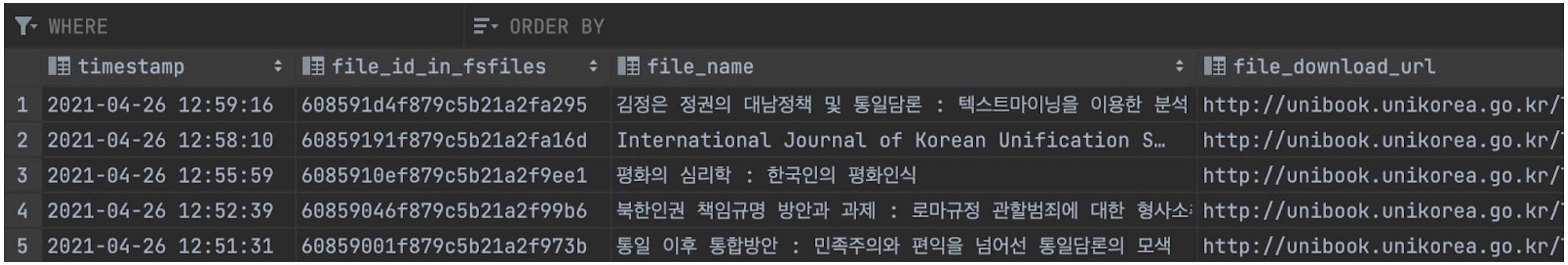
	top_category	category_count	categoty_rank
1	전체자료	4795	1
2	통일부_발간자료	[illegible]	2
	⋮	⋮	

Phase 1: Database Modeling

- Views to create (and submit)

8. View: fileSummary

    - Show the attached file information by summarizing their timestamp, file ID, filename, and download url
    - **SELECT** * **FROM** fileSummary



	timestamp	file_id_in_fsfiles	file_name	file_download_url
1	2021-04-26 12:59:16	608591d4f879c5b21a2fa295	김정은 정권의 대남정책 및 통일담론 : 텍스트마이닝을 이용한 분석	http://unibook.unikorea.go.kr/
2	2021-04-26 12:58:10	60859191f879c5b21a2fa16d	International Journal of Korean Unification S…	http://unibook.unikorea.go.kr/
3	2021-04-26 12:55:59	6085910ef879c5b21a2f9ee1	평화의 심리학 : 한국인의 평화인식	http://unibook.unikorea.go.kr/
4	2021-04-26 12:52:39	60859046f879c5b21a2f99b6	북한인권 책임규명 방안과 과제 : 로마규정 관할범죄에 대한 형사소	http://unibook.unikorea.go.kr/
5	2021-04-26 12:51:31	60859001f879c5b21a2f973b	통일 이후 통합방안 : 민족주의와 편익을 넘어선 통일담론의 모색	http://unibook.unikorea.go.kr/

: :

: :

Phase 1: Database Modeling

- A query to check the size of your database instance

  -
    ```
    SELECT table_schema AS 'DatabaseName’,
           ROUND(SUM(data_length+index_length)/1024, 1) AS 'Size(KB)’
    FROM information_schema.tables
    WHERE table_schema = 'YOUR DATABASE NAME’
    GROUP BY table_schema;
    ```

- A query to check each table size from your database

  -
    ```
    SELECT TABLE_SCHEMA, TABLE_NAME,
           ROUND(DATA_LENGTH/(1024), 1) AS 'data(KB)’,
           ROUND(INDEX_LENGTH/(1024), 1) AS 'idx(KB)’
    FROM information_schema.tables
    WHERE TABLE_TYPE = 'BASE TABLE’
           AND TABLE_SCHEMA = 'YOUR DATABASE SIZE';
    ```

Phase 1: Database Modeling

- What to submit
  - A report including
    - ER diagram of the implemented database
    - List of all tables and their attributes with precise notions of data types and integrity constraints
    - Description of the requested views
      - Size of the resulting table (in counts)
      - The screenshots of the table header and first five records
    - Summary of the database size and table sizes (in Kilobytes)
  - A zipped MySQL dump file containing all the database implementations including the database schema, records, views, *etc*.

Phase 1: Database Modeling

- Resources
  - How to create a dump file
    - MySQL Workbench https://dev.mysql.com/doc/workbench/en/wb-admin-export-import-management.html
    - DataGrip https://www.jetbrains.com/help/datagrip/export-data-in-ide.html
    - HeidiSQL https://www.heidisql.com/screenshots.php?which=export_sql
    - SequelAce https://sequelpro.com/docs/ref/working-with-data