1

Last Session :
Happy and *On Fire* !!!

1

We start
from the *ZERO*

3

Don’t feeling *left behind*
you are going at your *own pace*

DQLab

live Class

FUNDAMENTAL PYTHON

DATA SCIENCE

FUNDAMENTAL SQL

DATA VISUALIZATION

PORTFOLIO SESSION

DQLab

Live Class

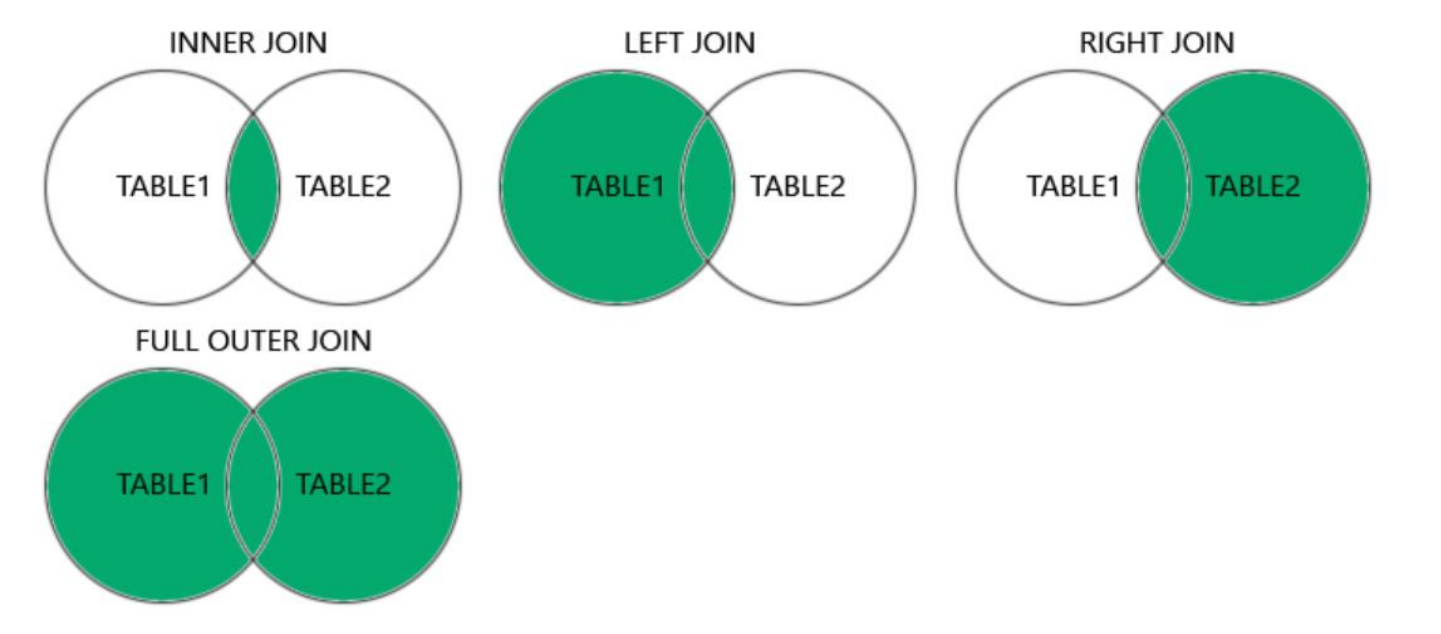
Query Join

Sesi 5 - Bootcamp Data Analyst with SQL and Python using Google Platform

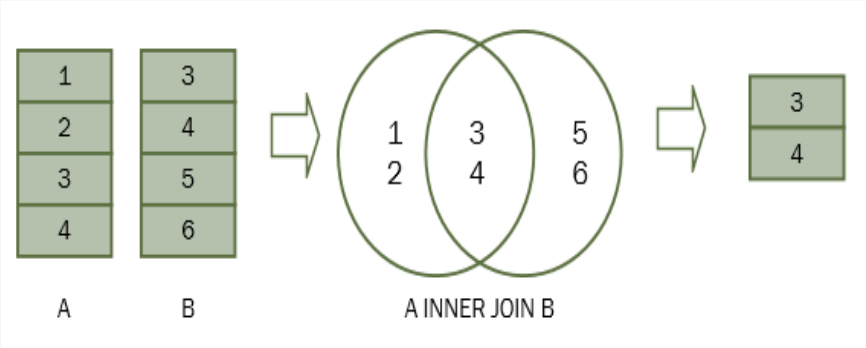
Fundamental SQL Menggunakan Google Big Query

1. DDL : Create, Drop
2. Query Select and combination
3. DML : Insert, Update, Delete
4. Query Date, Subquery and Case When
5. Query Join

QUERY JOIN

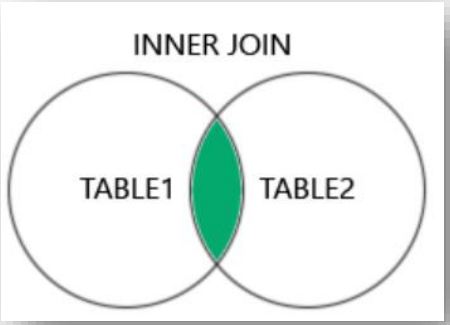

INNER JOIN
TABLE1
TABLE2
LEFT JOIN
TABLE1
TABLE2
RIGHT JOIN
TABLE1
TABLE2
FULL OUTER JOIN
TABLE1
TABLE2

SQL INNER JOIN




```
SELECT a
FROM A
INNER JOIN B ON b = a;
```

```
SELECT
  A.n
FROM A
INNER JOIN B ON B.n = A.n
INNER JOIN C ON C.n = A.n;
```




```
SELECT column_name(s)
FROM table1
INNER JOIN table2
ON table1.column_name = table2.column_name
;
```

```
SELECT t1.ProductID, t1.ProductName
, t1.CategoryID, t1.Price
, t2.CategoryName
FROM Products t1
INNER JOIN Categories t2
ON t1.CategoryID = t2.CategoryID
;
```

1

ProductID	ProductName	CategoryID	Price
1	Chais	1	18
2	Chang	1	19
3	Aniseed Syrup	2	10
10	Ikura	8	31

2

CategoryID	CategoryName	Description
1	Beverages	Soft drinks, coffees, teas, beers, and ales
2	Condiments	Sweet and savory sauces, relishes, spreads, and seasonings

3

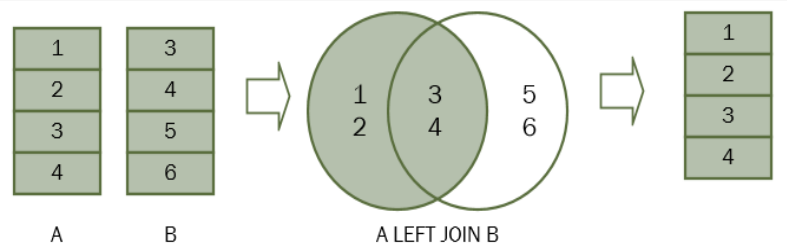
Number of Records: 3

ProductID	ProductName	CategoryID	Price	CategoryName
1	Chais	1	18	Beverages
2	Chang	1	19	Beverages
3	Aniseed Syrup	2	10	Condiments

SQL LEFT/RIGHT JOIN

Right Join atau Left Join adalah hal yang sama, yaitu untuk mengembalikan semua baris dari tabel kiri/kanan dan mencari apakah ada baris yang cocok atau tidak di tabel sebaliknya.

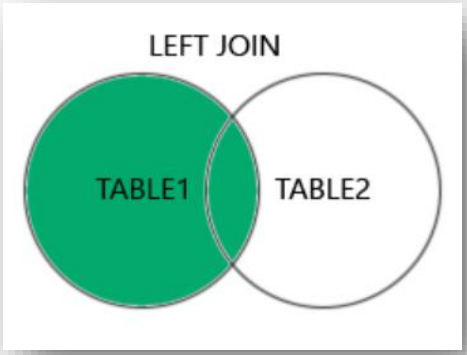
Saat kita menggabungkan tabel A dengan tabel B, semua baris pada tabel A (tabel kiri) dimasukkan ke dalam himpunan hasil apakah ada baris yang cocok pada tabel B atau tidak.



RIGHT JOIN

3
4
5
6

```
SELECT
        A.n
FROM
        A
LEFT JOIN B ON B.n = A.n;
```




```
SELECT column_name(s)
FROM table1
LEFT JOIN table2
ON table1.column_name = table2.column_name
;
```

1

ProductID	ProductName	CategoryID	Price
1	Chais	1	18
2	Chang	1	19
3	Aniseed Syrup	2	10
10	Ikura	8	31

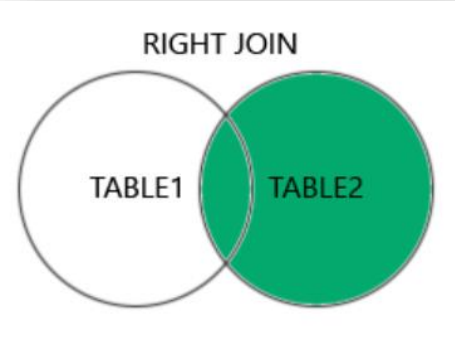
2

CategoryID	CategoryName	Description
1	Beverages	Soft drinks, coffees, teas, beers, and ales
2	Condiments	Sweet and savory sauces, relishes, spreads, and seasonings
8	Seafood	Seaweed and fish

```
SELECT t1.ProductID, t1.ProductName
, t1.CategoryID, t1.Price
, t2.CategoryName
FROM Products t1
LEFT JOIN Categories t2
ON t1.CategoryID = t2.CategoryID
;
```

3

ProductID	ProductName	CategoryID	Price	CategoryName
1	Chais	1	18	Beverages
2	Chang	1	19	Beverages
3	Aniseed Syrup	2	10	Condiments
10	Ikura	8	31	Seafood




```
SELECT column_name(s)
FROM table1
RIGHT JOIN table2
ON table1.column_name = table2.column_name
;
```

```
SELECT t1.ProductID, t1.ProductName
, t1.CategoryID, t1.Price
, t2.CategoryName
FROM Products t1
RIGHT JOIN Categories t2
ON t1.CategoryID = t2.CategoryID
;
```

1

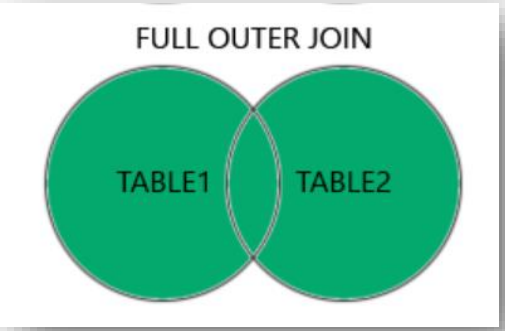
ProductID	ProductName	CategoryID	Price
1	Chais	1	18
2	Chang	1	19
3	Aniseed Syrup	2	10
10	Ikura	8	31

2

CategoryID	CategoryName	Description
1	Beverages	Soft drinks, coffees, teas, beers, and ales
2	Condiments	Sweet and savory sauces, relishes, spreads, and seasonings
8	Seafood	Seaweed and fish

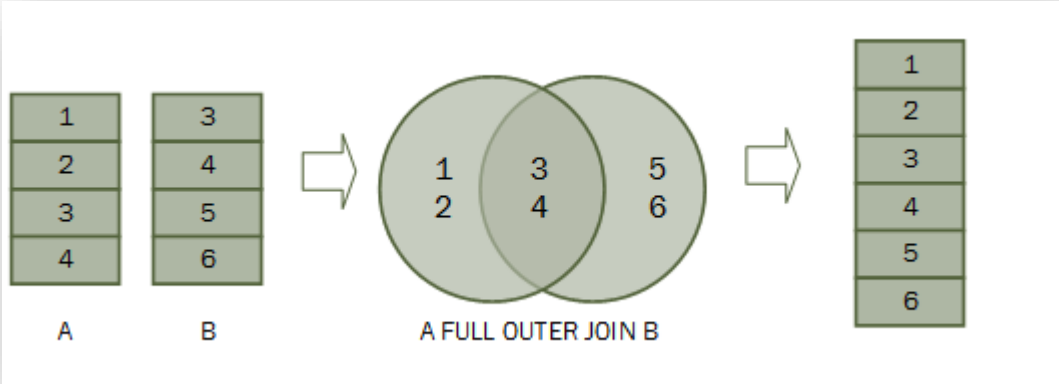
3

ProductID	ProductName	CategoryID	Price	CategoryName
1	Chais	1	18	Beverages
2	Chang	1	19	Beverages
3	Aniseed Syrup	2	10	Condiments
10	Ikura	8	31	Seafood



Secara teori, full outer join adalah kombinasi dari left join dan right join. Gabungan luar penuh mencakup semua baris dari tabel gabungan apakah tabel lain memiliki baris yang cocok atau tidak.

Jika baris dalam tabel gabungan tidak cocok, kumpulan hasil gabungan luar penuh berisi nilai NULL untuk setiap kolom tabel yang tidak memiliki bari yang cocok.




```
SELECT column_list
FROM A
FULL OUTER JOIN B ON B.n = A.n;
```

SQL FULL OUTER JOIN – Contoh

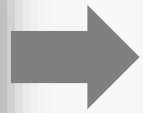
Pertama, buat dua tabel baru: keranjang dan buah untuk demonstrasi. Setiap keranjang menyimpan nol buah atau lebih dan setiap buah dapat disimpan dalam nol atau satu keranjang.

```
INSERT INTO baskets (basket_id, basket_name)
VALUES
        (1, 'A'),
        (2, 'B'),
        (3, 'C');
```

```
INSERT INTO fruits (
        fruit_id,
        fruit_name,
        basket_id
)
VALUES
        (1, 'Apple', 1),
        (2, 'Orange', 1),
        (3, 'Banana', 2),
        (4, 'Strawberry', NULL);
```

Untuk menggabungkan data dari kedua tabel ini, Anda menggunakan klausa gabungan dalam sebagai query berikut:

```
SELECT
        basket_name,
        fruit_name
FROM
        fruits
FULL OUTER JOIN baskets ON baskets.basket_id = fruits.basket_id;
```



```
basket_name | fruit_name
------------+------------
A           | Apple
A           | Orange
B           | Banana
(null)      | Strawberry
C           | (null)
```

SQL CROSS JOIN

Cross join adalah operasi join yang menghasilkan produk Cartesian dari dua tabel atau lebih. Dalam Matematika, produk Cartesian adalah operasi matematika yang mengembalikan rangkaian produk dari beberapa rangkaian.

A

n
1
2
3

B

c
x
y
z

```
SELECT *
FROM A
CROSS JOIN B
```

A x B

n	c
1	x
1	y
1	z
2	x
2	y
2	z
3	x
3	y
3	z

```sql
SELECT column_list
FROM A
CROSS JOIN B;
```

SQL CROSS JOIN – Contoh

Misalkan perusahaan memiliki dua organisasi penjualan yaitu Domestik dan Ekspor, yang bertanggung jawab atas penjualan di pasar domestik dan internasional.

```
INSERT INTO sales_organization (sales_org_id, sales_org)
VALUES
        (1, 'Domestic'),
        (2, 'Export');
```

```
INSERT INTO sales_channel (channel_id, channel)
VALUES
        (1, 'Wholesale'),
        (2, 'Retail'),
        (3, 'eCommerce'),
        (4, 'TV Shopping');
```

Untuk menemukan semua saluran penjualan yang mungkin dimiliki organisasi penjualan, Anda menggunakan CROSS JOIN untuk menggabungkan tabel sales_organization dengan tabel sales_channel sbb :

```
SELECT
        sales_org,
        channel
FROM
        sales_organization
CROSS JOIN sales_channel;
```

sales_org	channel
Domestic	Wholesale
Export	Wholesale
Domestic	Retail
Export	Retail
Domestic	eCommerce
Export	eCommerce
Domestic	TV Shopping
Export	TV Shopping

DQLab

live Class

Fundamental SQL Menggunakan Google Big Query

1. DDL : Create, Drop
2. Query Select and combination
3. DML : Insert, Update, Delete
4. Query Date, Subquery and Case When
5. Query Join

PostgreSQL

MariaDB Foundation

CLOUDERA

MySQL

DQLab

live Class

CURIOSITY

+

PASSIONATE

READY !!!

Thanks!

Let's Review Together

Dataset : Titanic Passenger

JOB INFORMATION | RESULTS | CHART PREVIEW | JSON | EXECUTION DETAILS | EXECUTION GRAPH

Row	PassengerId	Survived	Pclass	Name	Sex	Age	SibSp	Parch	Ticket	Fare	Cabin
1	180	0	3	Leonard, Mr. Lionel	male	36.0	0	0	LINE	0.0	null
2	264	0	1	Harrison, Mr. William	male	40.0	0	0	112059	0.0	B94
3	278	0	2	Parkes, Mr. Francis "...	male	null	0	0	239853	0.0	null
4	303	0	3	Johnson, Mr. William...	male	19.0	0	0	LINE	0.0	null
5	414	0	2	Cunningham, Mr. Alfr...	male	null	0	0	239853	0.0	null
6	467	0	2	Campbell, Mr. William	male	null	0	0	239853	0.0	null
7	482	0	2	Frost, Mr. Anthony W...	male	null	0	0	239854	0.0	null
8	598	0	3	Johnson, Mr. Alfred	male	49.0	0	0	LINE	0.0	null

Value

Atribut/ Field/ Column

Record/ Row

Entity

DQLab

live Class

Thanks!

Let's Prepare the Table first !

Create *Customer Profile* Table

```
drop table if exists `bootcamp-402414.data_analytic.customer_profile`;

create table `bootcamp-402414.data_analytic.customer_profile` as
select customer_id, customer_name, postal_code, product_id, order_date,
ship_date
from `bootcamp-402414.data_analytic.super_store`
;
select * from  `bootcamp-402414.data_analytic.customer_profile` ;
```

Create *Customer Location* Table

```
drop table if exists `bootcamp-402414.data_analytic.customer_location`;

create table `bootcamp-402414.data_analytic.customer_location` as
select postal_code, city, state, country_region, region
from `bootcamp-402414.data_analytic.super_store`
group by 1,2,3,4,5
;

select * from  `bootcamp-402414.data_analytic.customer_location` ;
```

Create *Product Catalog* Table

```
drop table if exists `bootcamp-402414.data_analytic.product_catalog`;

create table `bootcamp-402414.data_analytic.product_catalog` as
select product_id, category, sub_category, product_name
from `bootcamp-402414.data_analytic.super_store`
group by 1,2,3,4
;

select * from  `bootcamp-402414.data_analytic.product_catalog` ;
```

Create *Customer Segment* Table

```
drop table if exists `bootcamp-402414.data_analytic.customer_segment`;

create table `bootcamp-402414.data_analytic.customer_segment` as
select customer_name, segment
from `bootcamp-402414.data_analytic.super_store`
group by 1,2
;

select * from  `bootcamp-402414.data_analytic.customer_segment` ;
```

DQLab

live Class

Thanks!

Let's Explore The Data !

Query *SELECT* and *WHERE + SORTING*

```
select * from `bootcamp-402414.data_analytic.product_catalog` ;

select distinct sub_category from `bootcamp-402414.data_analytic.product_catalog` ;

select distinct sub_category from `bootcamp-402414.data_analytic.product_catalog`
order by sub_category asc ;

select * from `bootcamp-402414.data_analytic.product_catalog`
WHERE SUB_CATEGORY = 'Art' ;
```

Query *SELECT* and *WHERE + SORTING*

```
select * from  `bootcamp-402414.data_analytic.customer_profile` ;

select * from  `bootcamp-402414.data_analytic.customer_profile`
where order_date = date '2017-08-30'

select * from  `bootcamp-402414.data_analytic.customer_profile`
where order_date < date '2017-08-30' ;

select * from  `bootcamp-402414.data_analytic.customer_profile`
where product_id like '%OFF%' ;

select * from  `bootcamp-402414.data_analytic.customer_profile`
where product_id not like '%OFF%' ;
```

Query AGGREGATION + HAVING

```
select order_date
, count(*) as total_trx
from  `bootcamp-402414.data_analytic.customer_profile`
where product_id like '%OFF%' and order_date > date '2017-08-30'
group by order_date
order by order_date asc;
```

```
select order_date
, count(*) as total_trx
from  `bootcamp-402414.data_analytic.customer_profile`
where product_id like '%OFF%' and order_date > date '2017-08-30'
group by order_date
having total_trx > 10
order by order_date asc ;
```

Query *FUNCTION* *DATE_ADD + DATE_DIFF*

```
select customer_id
, product_id
, order_date
, ship_date
, DATE_ADD(ship_date, INTERVAL 10 DAY) AS ten_days_later_from_ship_date
, DATE_ADD(ship_date, INTERVAL 2 MONTH) AS two_months_later_from_ship_date
, DATE_ADD(ship_date, INTERVAL 1 YEAR) AS a_year_later_from_ship_date
, DATE_DIFF(ship_date, order_date, DAY) AS days_diff
FROM `bootcamp-402414.data_analytic.customer_profile` ;
```

Query *FUNCTION* *DATE_SUB + DATE_DIFF*

```
select customer_id
, product_id
, order_date
, ship_date
, DATE_SUB(ship_date, INTERVAL 10 DAY) AS ten_days_back_from_ship_date
, DATE_SUB(ship_date, INTERVAL 2 MONTH) AS two_months_back_from_ship_date
, DATE_SUB(ship_date, INTERVAL 1 YEAR) AS a_year_back_from_ship_date
, DATE_DIFF(ship_date, order_date, DAY) AS days_diff
, DATE_DIFF(ship_date, order_date, WEEK) AS days_diff
FROM `bootcamp-402414.data_analytic.customer_profile` ;
```

Query *FUNCTION EXTRACT DATE*

```
SELECT customer_id
, ship_date
, extract(DAY from ship_date) as ship_date_day
, extract(WEEK from ship_date) as ship_date_WEEK
, extract(MONTH from ship_date) as ship_date_MONTH
, extract(YEAR from ship_date) as ship_date_YEAR
FROM `bootcamp-402414.data_analytic.customer_profile`
limit 10 ;
```

Query *LEFT JOIN (1)*

```
select * from `bootcamp-402414.data_analytic.product_catalog`
;

select t1.*, t2.category, t2.sub_category, t2.product_name
from `bootcamp-402414.data_analytic.customer_profile` t1
left join  `bootcamp-402414.data_analytic.product_catalog` t2
on t1.product_id = t2.product_id
;
```

Query *LEFT JOIN (2)*

```
select * from  `bootcamp-402414.data_analytic.customer_location` ;

select t1.*, t2.city, t2.state, t2.country_region, t2.region
from `bootcamp-402414.data_analytic.customer_profile` t1
left join `bootcamp-402414.data_analytic.customer_location` t2
on t1.postal_code = t2.postal_code
;
```

Query *LEFT JOIN (3)*

```
select t1.*
, t2.category, t2.sub_category, t2.product_name
, t3.city, t3.state, t3.country_region, t3.region
, t4.segment
from `bootcamp-402414.data_analytic.customer_profile` t1
left join `bootcamp-402414.data_analytic.product_catalog` t2 on t1.product_id = t2.product_id
left join `bootcamp-402414.data_analytic.customer_location` t3 on t1.postal_code = t3.postal_code
left join `bootcamp-402414.data_analytic.customer_segment` t4 on t1.customer_name = t4.customer_name
;
```

Query *INNER JOIN*

```
select t1.customer_id, t1.customer_name, t2.segment
from `bootcamp-402414.data_analytic.customer_profile` t1
left join `bootcamp-402414.data_analytic.customer_segment` t2
on t1.customer_name = t2.customer_name
;

select t1.customer_id, t1.customer_name, t2.segment
from `bootcamp-402414.data_analytic.customer_profile` t1
inner join
  (select * from `bootcamp-402414.data_analytic.customer_segment`
  where segment = 'Consumer') t2
on t1.customer_name = t2.customer_name
;
```

Query *CROSS JOIN*

```
select * from
  (
  select distinct customer_name
  from `bootcamp-402414.data_analytic.customer_profile`
  where customer_name in ('Annie Thurman','Ben Ferrer','Henry Goldwyn')
  ) t0
cross join
  (
  select distinct product_id
  from `bootcamp-402414.data_analytic.customer_profile`
  where product_id in ('OFF-BI-10002432','TEC-PH-10004447','OFF-BI-10000285')
  ) t1
;
```

Thanks!

It’s time to QUIZ ☺

Thanks!

Congratulations for the winner ☺


Thanks!