SQL: QUERY DATE, SUBQUERY, CASE WHEN, AND JOIN

Anita Mila Oktafani

DQLab Live Class Data Analyst with SQL & Python in Google Platform

Table of Content

1. Query Date
   - Current_Date
   - Date
   - Date_Add
   - Date_Diff
   - Date_Sub
   - Extract
   - Format_Date
   - Parse_Date
   - Datetime
2. SubQuery
3. Case When
4. Join
   - Left/Right Join
   - Inner Join
   - Full Outer Join
   - Cross Join

QUERY DATE

Date Function - Current_Date()

Returns the current date as a DATE object. Parentheses "()" are optional when called without arguments.

```
------ Query FUNCTION CURRENT DATE
select current_date() as the_date;
```

Row	the_date
1	2023-12-27

Date Function - DATE() Structure

Create or extract dates. This function supports the following arguments:

- YEAR
- MONTH
- DAY
- TIMESTAMP_EXPRESSION
- TIME_ZONE_EXPRESSION
- DATETIME_EXPRESSION

```
------ Query FUNCTION DATE()
select
  DATE(2016, 12, 25) as date_ymd,
  DATE(DATETIME '2016-12-25 23:59:59') as date_dt,
  DATE(TIMESTAMP '2016-12-25 05:30:00+07', 'America/Los_Angeles') as date_tstz;
```

Row	date_ymd	date_dt	date_tstz
1	2016-12-25	2016-12...	2016-12-24

Date Function - DATE_ADD

Adds a specified time interval to DATE. DATE_ADD supports the following date_part values:

- DAY
- WEEK. Equivalent to 7 DAYS.
- MONTH
- QUARTER
- YEAR

Date Function - DATE_ADD

DATE_ADD(date_expression, INTERVAL int64_expression date_part)

```
------ Query FUNCTION DATE_ADD()
select
  date_add(DATE '2023-10-22', interval 5 day) as fivedayslater,
  date_add(DATE '2023-10-22', interval 10 day) as tendayslater,
  date_add(DATE '2023-10-22', interval -10 day) as tendayback
;
```

Row	fivedayslater	tendayslater	tendayback
1	2023-10-27	2023-11-01	2023-10-12

```
select
  date_add(DATE '2023-10-22', interval -1 month) as a_month_back,
  date_add(DATE '2023-10-22', interval -2 week) as two_weeks_back,
  date_add(DATE '2023-10-22', interval 1 year) as a_year_later
;
```

Row	a_month_back	two_weeks_back	a_year_later
1	2023-09-22	2023-10-08	2024-10-22

Date Function - DATE_DIFF

DATE_DIFF returns the total number of specified date_part intervals between two DATE objects (date_expression_a - date_expression_b). If the first DATE is earlier than the second, the result is negative. DATE_DIFF supports the following date_part values:

- DAY
- WEEK This date section starts on Sunday.
- WEEK(<WEEKDAY>): This date section starts on WEEKDAY. Valid values for WEEKDAY are SUNDAY, MONDAY, TUESDAY, WEDNESDAY, THURSDAY, FRIDAY, and SATURDAY
- MONTH, unless the first two arguments are TIMESTAMP objects.
- QUARTER
- YEAR

DATE_DIFF(date_expression_a, date_expression b, date_part)

```
------ Query FUNCTION DATE_DIFF()
select
  date_diff(DATE '2023-10-25', DATE '2023-10-23', day) as days_diff,
  date_diff(DATE '2023-10-25', DATE '2023-10-20', week) as weeks_diff,
  date_diff(DATE '2023-10-25', DATE '2023-09-20', month) as month_diff,
  date_diff(DATE '2023-10-25', DATE '2020-10-20', year) as year_diff
;
```

days_diff	weeks_diff	month_diff	year_diff
2	1	1	3

DATE_DIFF with the date part WEEK returns 1 because DATE_DIFF counts the number of date part boundaries in this range of dates.

Each WEEK begins on Sunday, so there is one date part boundary between Wednesday, 2023-10-25 and Saturday, 2023-10-20.

```
select
  date_diff('2023-10-25', '2023-10-23', week) as weeks_diff,
  date_diff('2023-10-25', '2023-10-23', week(wednesday)) as week_weekday_diff
;
```

Row	weeks_diff	week_weekday_diff
1	0	1

Date Function - DATE_SUB

DATE_SUB(date_expression, INTERVAL int64_expression date_part)

Subtracts a specified time interval from DATE.

DATE_SUB supports the following date_part values:

- DAY
- WEEK
- MONTH
- QUARTER
- YEAR

```
------ Query FUNCTION DATE_SUB
select date_sub(DATE '2008-12-25', interval 5 day) as five_days_ago;
```

Row	five_days_ago
1	2008-12-20

```
select
  date_sub(DATE '2023-10-22', interval 5 day) as five_days_ago,
  date_add(DATE '2023-10-22', interval -5 day) as five_days_ago
;
```

Row	five_days_ago	five_days_ago_1
1	2023-10-17	2023-10-17

DATE_SUB with interval **5 day** and **DATE_ADD** with interval **-5 day** have the same results

Date Function - EXTRACT

Returns the value corresponding to the specified date section. The section must be one of:

- DAYOFWEEK
- DAY
- DAYOFYEAR
- WEEK
- WEEK(<WEEKDAY>)
- MONTH
- QUARTER
- YEAR

EXTRACT(part FROM date_expression)

```
------ Query FUNCTION EXTRACT DATE
SELECT
  extract(DAY from DATE '2023-10-23') as the_day,
  extract(WEEK from DATE '2023-10-23') as the_week,
  extract(MONTH from DATE '2023-10-23') as the_month,
  extract(YEAR from DATE '2023-10-23') as the_year
;
```

Row	the_day	the_week	the_month	the_year
1	23	43	10	2023

FORMAT_DATE

Formats date_expr according to the specified format_string.
Return Data Type (STRING)

```
------ Query FUNCTION FORMAT_DATE
select format_date('%x', DATE '2008-12-25') as us_format;
```

Row	us_format
1	12/25/08

```
select format_date('%b-%d-%Y', DATE '2008-12-25') as formatted;
```

Row	formatted
1	Dec-25-2008

```
select
  format_date('%b-%d-%Y', current_date()) as formatted,
  format_date('%d-%b-%Y', current_date()) as formatted,
  format_date('%Y-%b-%d', current_date()) as formatted,
  format_date('%d-%m-%Y', current_date()) as formatted,
;
```

Row	formatted	formatted_1	formatted_2	formatted_3
1	Dec-27-2023	27-Dec-2023	2023-Dec-27	27-12-2023

PARSE_DATE

Converts a string representation of a date to a DATE object. format_string contains format elements that determine how the date_string is formatted. Each element in date_string must have a corresponding element in format_string. The location of each element in format_string must match the location of each element in date_string.

This work because elements on both sides match

```
select parse_date('%A %b %e %Y', 'Thursday Dec 25 2008') as the_date;
```

Row	the_date
1	2008-12-25

This produces an error because the year element is in different locations

```
select parse_date('%Y %A %b %e', 'Thursday Dec 25 2008') as the_date;
```

Failed to parse input string "Thursday Dec 25 2008"

This produces an error because one of the year element is missing

```
select parse_date('%A %b %e', 'Thursday Dec 25 2008') as the_date;
```

Failed to parse input string "Thursday Dec 25 2008"

This work because %F can find all matching elements in date_string

```
select parse_date('%F', '2000-12-30') as the_date;
```

Row	the_date
1	2000-12-30

DATETIME

- Creates a DATETIME object using INT64 values representing year, month, day, hour, minute, and second.

Return Data Type (DATETIME)

a DATE object and an optional TIME object

```
------ Query FUNCTION DATETIME
select
  datetime(2008, 12, 25, 05, 30, 00) as datetime_ymdhms,
  datetime(TIMESTAMP "2008-12-25 05:30:00+00", "America/Los_Angeles") as datetime_tstz
;
```

a TIMESTAMP object

optional specify time zone

Row	datetime_ymdhms	datetime_tstz
1	2008-12-25T05:30:00	2008-12-24T21:30:00

DATETIME_ADD

Adds an int64_expression part unit to the DATETIME object.
DATETIME_ADD supports the following values for some:

- MICROSECONDS
- MILLISECOND
- SECOND
- MINUTES
- HOURS
- DAY
- WEEK. Equivalent to 7 DAYS.
- MONTH
- QUARTER
- YEAR

Return Data Type (DATETIME)

```
------ Query FUNCTION DATETIME_ADD
select
  datetime("2008-12-25 05:30:00") as original_date,
  datetime_add(datetime "2008-12-25 05:30:00", interval 10 minute) as later
;
```

Row	original_date	later
1	2008-12-25T05:30:00	2008-12-25T05:40:00

DATETIME_DIFF

datetime_expression_a

datetime_expression_b

datetime_expression_a - datetime_expression_b

```
------ Query FUNCTION DATETIME_DIFF
select
  datetime("2010-07-07 10:20:00") as first_datetime,
  datetime("2008-12-25 15:30:00") as second_datetime,
  datetime_diff(datetime "2010-07-07 10:20:00", datetime "2008-12-25
15:30:00", day) as difference
;
```

Row	first_datetime	second_datetime	difference
1	2010-07-07T10:20:00	2008-12-25T15:30:00	559

```
select
  datetime_diff(datetime "2017-10-15 00:00:00", datetime "2017-10-14 00:00:00", day) as days_diff,
  datetime_diff(datetime "2017-10-15 00:00:00", datetime "2017-10-14 00:00:00", week) as weeks_diff
;
```

Row	days_diff	weeks_diff
1	1	1

SUBQUERY

SUBQUERY

SubQuery is a combination of more than one query to get a certain output.

```
------ SUB QUERY
select *
from `dqlab-boootcamp.data_analytic.super store`
where Ship_Mode = "Standard Class";
```

Main Query

```
select Product_Name, Sub_Category, City
from `dqlab-boootcamp.data_analytic.super store`
where Postal_Code in (50701, 43017, 44052)
order by Sub_Category;
```

SubQuery

```
select Product_Name, Sub_Category, City
from `dqlab-boootcamp.data_analytic.super store`
where Postal_Code in (select Postal_Code
from `dqlab-boootcamp.data_analytic.super store`
where Ship_Mode = "Standard Class")
order by Sub_Category;
```

Combined

Row	Product_Name	Sub_Category	City
1	Imation Swivel Flash Drive USB...	Accessories	Dublin
2	Logitech Wireless Performance...	Accessories	Dublin
3	Logitech G602 Wireless Gamin...	Accessories	Lorain
4	Belkin F8E887 USB Wired Ergo...	Accessories	Akron

CASE WHEN

CASE WHEN

Case when is used to create conditional statements or rules from a column

Conditional1

Conditional2

```
------ CASE WHEN
select Sub_Category, Sales,
CASE
  WHEN Sales > 100 THEN 'The sales is greater than 100'
  WHEN Sales = 100 THEN 'The sales is 100'
  ELSE 'The sales is under 100'
end as SalesText
from `dqlab-boootcamp.data_analytic.super store`;
```

Result1

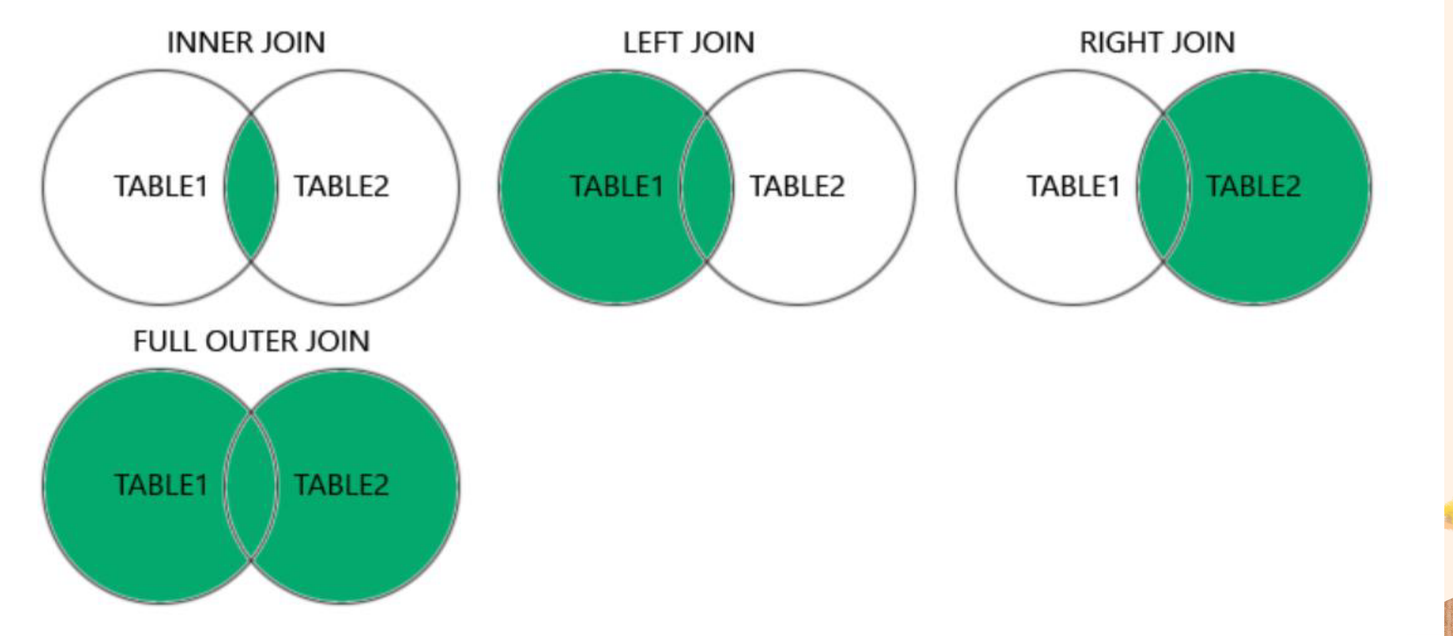
Result2

Result

Row	Sub_Category	Sales	SalesText
1	Art	30.32	The sales is under 100
2	Paper	9.99	The sales is under 100
3	Binders	15.24	The sales is under 100
4	Chairs	1408.1	The sales is greater than 100
5	Phones	263.96	The sales is greater than 100
6	Paper	106.32	The sales is greater than 100

QUERY JOIN

JOIN


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

JOIN

First, we create new table named Customer Profile (t1) which contains customer_id, customer_name, postal_code, product_id, order_date, ship_date

Second, we create new table named Customer Location (t2) which contains postal_code, city, state, country_region, region

Third, we create new table named Product Catalog (t3) which contains product_id, category, sub_category, product_name

Fourth, we create new table named Customer Segment (t4) which contains customer_name, segment

all contents of the new table are taken from the superstore table

LEFT/RIGHT JOIN

column_name(s)

table1

```
---- left join table customer profile & table product catalog
select t1.*, t2.category, t2.sub_category, t2.product_name
from `dqlab-boootcamp.data_analytic.customer_pofile` t1
left join `dqlab-boootcamp.data_analytic.product_catalog` t2
on t1.product_id = t2.product_id;
```

table2

table1.column_name = t2.column_name

Row	customer_id	customer_name	postal_code	product_id	order_date	ship_date	category	sub_category	product_name
1	AT-10735	Annie Thurman	6010	OFF-ST-10000563	2017-08-30	2017-09-04	Office Supplies	Storage	Fellowes Bank...
2	AT-10735	Annie Thurman	6010	OFF-BI-10002432	2017-08-30	2017-09-04	Office Supplies	Binders	Wilson Jones ...
3	BF-11170	Ben Ferrer	60623	TEC-PH-100044...	2018-05-13	2018-05-18	Technology	Phones	Toshiba IPT20...
4	BF-11170	Ben Ferrer	60623	OFF-FA-10002280	2018-05-13	2018-05-18	Office Supplies	Fasteners	Advantus Plast...
5	HG-14965	Henry Goldwyn	39503	OFF-BI-10000285	2019-01-24	2019-01-26	Office Supplies	Binders	XtraLife ClearV...

```sql
---- right join all table customer profile & table customer loca
select t1.*, t2.city, t2.country_region, t2.region
from `dqlab-boootcamp.data_analytic.customer_pofile` t1
right join `dqlab-boootcamp.data_analytic.customer_location` t2
on t1.postal_code = t2.postal_code;
```

Row	customer_id	customer_name	postal_code	product_id	order_date	ship_date	city	country_region	region
1	EH-14125	Eugene Hildebra...	26003	OFF-PA-10001...	2020-06-21	2020-06-25	Wheeling	United States	East
2	EH-14125	Eugene Hildebra...	26003	OFF-BI-100028...	2020-06-21	2020-06-25	Wheeling	United States	East
3	EH-14125	Eugene Hildebra...	26003	OFF-BI-100001...	2020-06-21	2020-06-25	Wheeling	United States	East
4	NF-18385	Natalie Fritzler	26003	FUR-TA-10001...	2020-10-12	2020-10-12	Wheeling	United States	East
5	VM-21685	Valerie Mitchum	5401	TEC-PH-10002...	2019-04-06	2019-04-10	Burlingt...	United States	East

```sql
---- left join all table t1 t2 t3 t4
select t1.*
, t2.category, t2.sub_category, t2.product_name
, t3.city, t3.state, t3.country_region, t3.region
, t4.segment
from `dqlab-boootcamp.data_analytic.customer_pofile` t1
left join `dqlab-boootcamp.data_analytic.product_catalog` t2 on t1.product_id = t2.product_id
left join `dqlab-boootcamp.data_analytic.customer_location` t3 on t1.postal_code = t3.postal_code
left join `dqlab-boootcamp.data_analytic.customer_segment` t4 on t1.customer_name = t4.customer_name
;
```

Row	customer_id	customer_name	postal_code	product_id	order_date	ship_date	category	sub_category	product_name	city	state	country_regi
1	AT-10735	Annie Thurman	6010	OFF-ST-10000563	2017-08-30	2017-09-04	Office Suppl...	Storage	Fellowes Ba...	Bristol	Connecticut	United State
2	AT-10735	Annie Thurman	6010	OFF-BI-10002432	2017-08-30	2017-09-04	Office Suppl...	Binders	Wilson Jone...	Bristol	Connecticut	United State
3	BF-11170	Ben Ferrer	60623	TEC-PH-100044...	2018-05-13	2018-05-18	Technology	Phones	Toshiba IPT...	Chica...	Illinois	United State
4	BF-11170	Ben Ferrer	60623	OFF-FA-10002280	2018-05-13	2018-05-18	Office Suppl...	Fasteners	Advantus Pl...	Chica...	Illinois	United State
5	HG-14965	Henry Goldwyn	39503	OFF-BI-10000285	2019-01-24	2019-01-26	Office Suppl...	Binders	XtraLife Cle...	Gulfp...	Mississippi	United State
6	AW-10930	Arthur Wiediger	19143	OFF-LA-100001...	2019-10-07	2019-10-10	Office Suppl...	Labels	Avery 511	Philad...	Pennsylva...	United State

INNER JOIN

```
------ Query INNER JOIN
select t1.customer_id, t1.customer_name, t2.segment
from `dqlab-boootcamp.data_analytic.customer_pofile` t1
inner join `dqlab-boootcamp.data_analytic.customer_segment` t2
on t1.customer_name = t2.customer_name;
```

Row	customer_id	customer_name	segment
1	AT-10735	Annie Thurman	Consumer
2	AT-10735	Annie Thurman	Consumer
3	BF-11170	Ben Ferrer	Home Office
4	BF-11170	Ben Ferrer	Home Office
5	HG-14965	Henry Goldwyn	Corporate
6	AW-10930	Arthur Wiediger	Home Office

FULL OUTER JOIN

```
------ Query FULL OUTER JOIN
select t1.postal_code, t3.product_name
from `dqlab-boootcamp.data_analytic.customer_pofile` t1
full outer join `dqlab-boootcamp.data_analytic.product_catalog` t3
on t1.product_id = t3.product_id;
```

Row	postal_code	product_name
1	26003	Xerox 1881
2	94122	Heavy-Duty E-Z-D Binders
3	60440	Geemarc AmpliPOWER60
4	10009	Xerox 1881
5	90004	Satellite Sectional Post Binders

CROSS JOIN

```
------ Query CROSS JOIN
select * from
  (
    select distinct customer_name
    from `dqlab-boootcamp.data_analytic.customer_pofile`
    where customer_name in ('Annie Thurman','Ben Ferrer','Henry Goldwyn')
  ) t0
  cross join
  (
    select distinct product_id
    from `dqlab-boootcamp.data_analytic.customer_pofile`
    where product_id in ('OFF-BI-10002432','TEC-PH-10004447','OFF-BI-10000285')
  ) t1
;
```

Row	customer_name	product_id
1	Henry Goldwyn	TEC-PH-10004447
2	Henry Goldwyn	OFF-BI-10000285
3	Henry Goldwyn	OFF-BI-10002432
4	Annie Thurman	TEC-PH-10004447
5	Annie Thurman	OFF-BI-10000285
6	Annie Thurman	OFF-BI-10002432

Thank You

LinkedIn: https://www.linkedin.com/in/anitamilaoktafani/

DQLab Live Class Data Analyst with SQL & Python in Google Platform