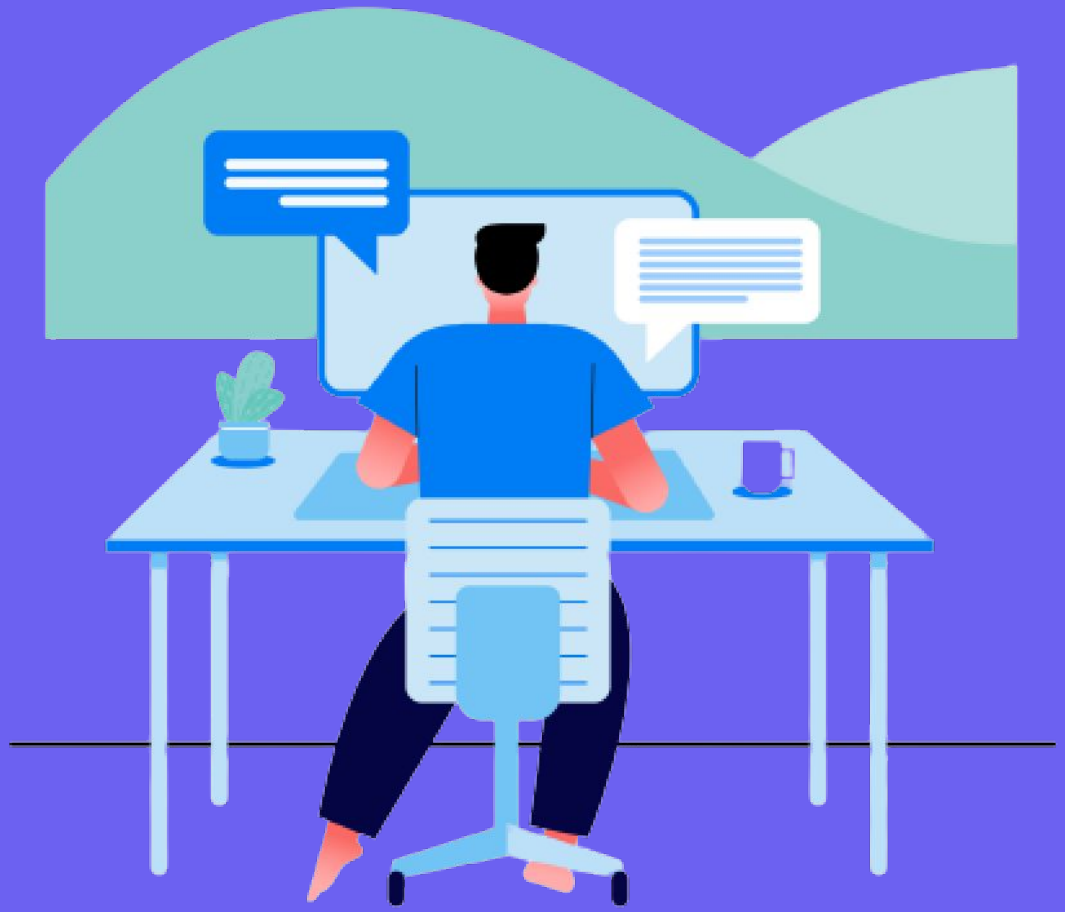
Date Time Function

Relevel
by Unacademy



Why do we need date-time manipulation?

Most of the data in the real world include date-time data. For most of the analysis, we need to perform data manipulation on date columns to perform research (not limited) to:

- Daily/weekly/monthly trends(for example, sales trends for an e-commerce website)
- Perform forecast (Eg: Demand for cabs in a city)
- Analysis using time data(Eg: Number of days since a customer made a transaction, time is taken between two cab

We need to learn SQL functions for manipulating data for all the analyses mentioned above.

Date/Time Data Types

There are four main data types:

DATA TYPE	DESCRIPTION	EXAMPLE	OUTPUT
TIMESTAMP	date and time	`TIMESTAMP '2021-08-09 13:57:40'`	2021-08-09T13:57:40
DATE	date (no time)	`DATE '2021-08-09 13:57:40'`	2021-08-09
TIME	time (no day)	`TIME '2021-08-09 13:57:40'`	13:57:40
INTERVAL	interval between two date/times	`INTERVAL '1 day 2 hours 10 seconds'`	1 day, 2:00:10

We'll go over more about each of these.

Understanding Data Time Functions

In the coming slides, we will understand the usage of each function with a demonstration. **We will use the below dataset for demonstration:**

Ride_share

id	start_terminal	end_terminal	start_time	end_time
1	31245	31109	2012-01-01 00:04:00	2012-01-01 00:11:00
2	31400	31103	2012-01-01 00:10:00	2012-01-01 00:29:00
3	31400	31103	2012-01-01 00:10:00	2012-01-01 00:29:00
4	31101	31602	2012-01-01 00:15:00	2012-01-01 00:23:00
5	31102	31109	2012-01-01 00:15:00	2012-01-01 00:23:00
6	31017	31014	2012-01-01 00:17:00	2012-01-01 00:23:00
7	31236	31404	2012-01-01 00:18:00	2012-01-01 00:47:00
8	31101	31201	2012-01-01 00:22:00	2012-01-01 00:27:00
9	31014	31237	2012-01-01 00:24:00	2012-01-01 00:33:00
10	31101	31218	2012-01-01 00:25:00	2012-01-01 00:40:00

NOW() Function

It returns the current date and time.

Query

```
SELECT *, NOW() AS current_date_time FROM ride_share
```

Output

id	start_terminal	end_terminal	start_time	end_time	current_date_time
1	31245	31109	2012-01-01 00:04:00	2012-01-01 00:11:00	2022-03-09 05:28:27
2	31400	31103	2012-01-01 00:10:00	2012-01-01 00:29:00	2022-03-09 05:28:27
3	31400	31103	2012-01-01 00:10:00	2012-01-01 00:29:00	2022-03-09 05:28:27
4	31101	31602	2012-01-01 00:15:00	2012-01-01 00:23:00	2022-03-09 05:28:27
5	31102	31109	2012-01-01 00:15:00	2012-01-01 00:23:00	2022-03-09 05:28:27
6	31017	31014	2012-01-01 00:17:00	2012-01-01 00:23:00	2022-03-09 05:28:27
7	31236	31404	2012-01-01 00:18:00	2012-01-01 00:47:00	2022-03-09 05:28:27
8	31101	31201	2012-01-01 00:22:00	2012-01-01 00:27:00	2022-03-09 05:28:27
9	31014	31237	2012-01-01 00:24:00	2012-01-01 00:33:00	2022-03-09 05:28:27
10	31101	31218	2012-01-01 00:25:00	2012-01-01 00:40:00	2022-03-09 05:28:27

CURRENT_DATE Function

It returns the current date

Query

```
SELECT *, CURRENT_DATE AS current_date FROM ride_share
```

Output

id	start_terminal	end_terminal	start_time	end_time	current_date
1	31245	31109	2012-01-01 00:04:00	2012-01-01 00:11:00	2022-03-09 00:00:00
2	31400	31103	2012-01-01 00:10:00	2012-01-01 00:29:00	2022-03-09 00:00:00
3	31400	31103	2012-01-01 00:10:00	2012-01-01 00:29:00	2022-03-09 00:00:00
4	31101	31602	2012-01-01 00:15:00	2012-01-01 00:23:00	2022-03-09 00:00:00
5	31102	31109	2012-01-01 00:15:00	2012-01-01 00:23:00	2022-03-09 00:00:00
6	31017	31014	2012-01-01 00:17:00	2012-01-01 00:23:00	2022-03-09 00:00:00
7	31236	31404	2012-01-01 00:18:00	2012-01-01 00:47:00	2022-03-09 00:00:00
8	31101	31201	2012-01-01 00:22:00	2012-01-01 00:27:00	2022-03-09 00:00:00
9	31014	31237	2012-01-01 00:24:00	2012-01-01 00:33:00	2022-03-09 00:00:00
10	31101	31218	2012-01-01 00:25:00	2012-01-01 00:40:00	2022-03-09 00:00:00

CURRENT_TIME Function

It returns the current time

Query

```
SELECT *, CURRENT_TIME AS current_time FROM ride_share
```

Output

id	start_terminal	end_terminal	start_time	end_time	current_time
1	31245	31109	2012-01-01 00:04:00	2012-01-01 00:11:00	05:35:32
2	31400	31103	2012-01-01 00:10:00	2012-01-01 00:29:00	05:35:32
3	31400	31103	2012-01-01 00:10:00	2012-01-01 00:29:00	05:35:32
4	31101	31602	2012-01-01 00:15:00	2012-01-01 00:23:00	05:35:32
5	31102	31109	2012-01-01 00:15:00	2012-01-01 00:23:00	05:35:32
6	31017	31014	2012-01-01 00:17:00	2012-01-01 00:23:00	05:35:32
7	31236	31404	2012-01-01 00:18:00	2012-01-01 00:47:00	05:35:32
8	31101	31201	2012-01-01 00:22:00	2012-01-01 00:27:00	05:35:32
9	31014	31237	2012-01-01 00:24:00	2012-01-01 00:33:00	05:35:32
10	31101	31218	2012-01-01 00:25:00	2012-01-01 00:40:00	05:35:32

DATE() Function

It extracts the date part from a date_time expression.

Query

```
SELECT *, CURRENT_TIME AS current_time FROM ride_share
```

Output

id	start_terminal	end_terminal	start_time	end_time	ride_start_date
1	31245	31109	2012-01-01 00:04:00	2012-01-01 00:11:00	2012-01-01 00:00:00
2	31400	31103	2012-01-01 00:10:00	2012-01-01 00:29:00	2012-01-01 00:00:00
3	31400	31103	2012-01-01 00:10:00	2012-01-01 00:29:00	2012-01-01 00:00:00
4	31101	31602	2012-01-01 00:15:00	2012-01-01 00:23:00	2012-01-01 00:00:00
5	31102	31109	2012-01-01 00:15:00	2012-01-01 00:23:00	2012-01-01 00:00:00
6	31017	31014	2012-01-01 00:17:00	2012-01-01 00:23:00	2012-01-01 00:00:00
7	31236	31404	2012-01-01 00:18:00	2012-01-01 00:47:00	2012-01-01 00:00:00
8	31101	31201	2012-01-01 00:22:00	2012-01-01 00:27:00	2012-01-01 00:00:00
9	31014	31237	2012-01-01 00:24:00	2012-01-01 00:33:00	2012-01-01 00:00:00
10	31101	31218	2012-01-01 00:25:00	2012-01-01 00:40:00	2012-01-01 00:00:00

EXTRACT() Function

It returns a single part from a date/time datatype.

Syntax of Extract Function

EXTRACT (field from date/time source)

This function takes two inputs:

1. **Field:** It refers to the value you want to extract from the date/time datatype

1. **Date/time source:** It refers to the date/time column from which we want to extract the field

EXTRACT() Function – Possible inputs for Field

Potential input for field

Field Value	TIMESTAMP	Interval
CENTURY	The century	The number of centuries
DAY	The day of the month (1-31)	The number of days
DECADE	The decade that is the year divided by 10	Sames as TIMESTAMP
DOW	The day of week Sunday (0) to Saturday (6)	N/A
DOY	The day of year that ranges from 1 to 366	N/A
EPOCH	The number of seconds since 1970-01-01 00:00:00 UTC	The total number of seconds in the interval
HOUR	The hour (0-23)	The number of hours
ISODOW	Day of week based on ISO 8601 Monday (1) to Sunday (7)	N/A
ISOYEAR	ISO 8601 week number of year	N/A
MICROSECONDS	The seconds field, including fractional parts, multiplied by 1000000	Sames as TIMESTAMP
MILLENNIUM	The millennium	The number of millennium
MILLISECONDS	The seconds field, including fractional parts, multiplied by 1000	Sames as TIMESTAMP

EXTRACT() Function – Possible inputs for Field

Potential input for field

Field Value	TIMESTAMP	Interval
MINUTE	The minute (0-59)	The number of minutes
MONTH	Month, 1-12	The number of months, modulo (0-11)
QUARTER	Quarter of the year	The number of quarters
SECOND	The second	The number of seconds
TIMEZONE	The timezone offset from UTC, measured in seconds	N/A
TIMEZONE_HOUR	The hour component of the time zone offset	N/A
TIMEZONE_MINUTE	The minute component of the time zone offset	N/A
WEEK	The number of the ISO 8601 week-numbering week of the year	N/A
YEAR	The year	Sames as TIMESTAMP

EXTRACT() Function – An Example

Query

```
SELECT *,
EXTRACT(DAY FROM start_time) AS ride_start_day,
EXTRACT(MONTH FROM start_time) AS ride_start_month,
EXTRACT(YEAR FROM start_time) AS ride_start_year
FROM ride_share
```

Output

id	start_terminal	end_terminal	start_time	end_time	ride_start_day	ride_start_month	ride_start_year
1	31245	31109	2012-01-01 00:04:00	2012-01-01 00:11:00	1	1	2012
2	31400	31103	2012-01-01 00:10:00	2012-01-01 00:29:00	1	1	2012
3	31400	31103	2012-01-01 00:10:00	2012-01-01 00:29:00	1	1	2012
4	31101	31602	2012-01-01 00:15:00	2012-01-01 00:23:00	1	1	2012
5	31102	31109	2012-01-01 00:15:00	2012-01-01 00:23:00	1	1	2012
6	31017	31014	2012-01-01 00:17:00	2012-01-01 00:23:00	1	1	2012
7	31236	31404	2012-01-01 00:18:00	2012-01-01 00:47:00	1	1	2012
8	31101	31201	2012-01-01 00:22:00	2012-01-01 00:27:00	1	1	2012
9	31014	31237	2012-01-01 00:24:00	2012-01-01 00:33:00	1	1	2012
10	31101	31218	2012-01-01 00:25:00	2012-01-01 00:40:00	1	1	2012

Instructions for practice questions

- Log into https://mode.com/
- Create a new report
- Access database tutorial.dc_bikeshare_q1_2012, tutorial.crunchbase_companies_clean_date, tutorial.crunchbase_acquisitions_clean_date

Practice Question - 1

Find the number of rides per month.

Instructions:

Use tutorial.dc_bikeshare_q1_2012 table. Extract the month from start_terminal and then count the id.

Solution - 1

```
SELECT
        EXTRACT(month FROM start_time) AS month,
      COUNT(id) AS num_rides
FROM
        tutorial.dc_bikeshare_q1_2012
GROUP BY  1
ORDER BY
        month
```

Solution - 1

Output

```sql
SELECT
  EXTRACT(month FROM start_time) AS month,
  COUNT(id) AS num_rides
FROM
  tutorial.dc_bikeshare_q1_2012
GROUP BY
  1
ORDER BY 1
```

3 rows | 48B returned in 636ms

month	num_rides
1	96744
2	103137
3	164875

Practice Question - 2

Find the number of rides per month per rider type.

Instructions:

Use tutorial.dc_bikeshare_q1_2012 table. Extract the month from start_terminal and then count the id and group by rider type.

Solution - 2

```
SELECT
        EXTRACT(month FROM start_time) AS month,
        rider_type,
        COUNT(id) AS num_rides
FROM
         tutorial.dc_bikeshare_q1_2012
GROUP BY  1, rider_type
ORDER BY
        month
```

Solution - 2

Output

```
SELECT
   EXTRACT(month FROM start_time) AS month,
   rider_type,
   COUNT(id) AS num_rides
FROM
   tutorial.dc_bikeshare_q1_2012
GROUP BY  1, rider_type
ORDER BY
  month
```

rows | 144B returned in 612ms

month	rider_type	num_rides
1	Casual	8969
1	Registered	87775
2	Registered	94416
2	Casual	8721
3	Registered	133257
3	Casual	31618

Adding time interval to the date/time column

We can add a time interval to a date/time column. The interval value can be minute, hour, day, week, month.

Syntax

Date_column + INTERVAL ‘mention the interval value’

* Refer to the field values discussed earlier to find the interval values

Adding time interval to the date/time column

Query

```
SELECT *,
 id,  start_time,  start_time + INTERVAl '1 DAY' AS start_date_day_add,  start_time + INTERVAl '1 HOUR' AS start_date_hour_add,
start_time + INTERVAl '1 WEEK' AS start_date_week_add,  start_time + INTERVAl '1 MONTH' AS start_date_month_add
FROM ride_share
```

Output

id	start_time	start_date_day_add	start_date_hour_add	start_date_week_add	start_date_month_add
1	2012-01-01 00:04:00	2012-01-02 00:04:00	2012-01-01 01:04:00	2012-01-08 00:04:00	2012-02-01 00:04:00
2	2012-01-01 00:10:00	2012-01-02 00:10:00	2012-01-01 01:10:00	2012-01-08 00:10:00	2012-02-01 00:10:00
3	2012-01-01 00:10:00	2012-01-02 00:10:00	2012-01-01 01:10:00	2012-01-08 00:10:00	2012-02-01 00:10:00
4	2012-01-01 00:15:00	2012-01-02 00:15:00	2012-01-01 01:15:00	2012-01-08 00:15:00	2012-02-01 00:15:00
5	2012-01-01 00:15:00	2012-01-02 00:15:00	2012-01-01 01:15:00	2012-01-08 00:15:00	2012-02-01 00:15:00
6	2012-01-01 00:17:00	2012-01-02 00:17:00	2012-01-01 01:17:00	2012-01-08 00:17:00	2012-02-01 00:17:00
7	2012-01-01 00:18:00	2012-01-02 00:18:00	2012-01-01 01:18:00	2012-01-08 00:18:00	2012-02-01 00:18:00
8	2012-01-01 00:22:00	2012-01-02 00:22:00	2012-01-01 01:22:00	2012-01-08 00:22:00	2012-02-01 00:22:00
9	2012-01-01 00:24:00	2012-01-02 00:24:00	2012-01-01 01:24:00	2012-01-08 00:24:00	2012-02-01 00:24:00
10	2012-01-01 00:25:00	2012-01-02 00:25:00	2012-01-01 01:25:00	2012-01-08 00:25:00	2012-02-01 00:25:00

Subtracting time interval from date/time column

We can add a time interval to a date/time column. Interval value can be minute, hour, day, week, month.

Syntax of Extract Function

Date_column - INTERVAL ‘mention the interval value’

* Refer to the field values discussed earlier to find the interval values

Subtracting time interval from date/time column

Query

```
SELECT *,
 id,  start_time, start_time - INTERVAl '1 DAY' AS start_date_day_diff,  start_time -  INTERVAl '1 HOUR' AS start_date_hour_diff,  start_time - INTERVAl '1 WEEK' AS start_date_week_diff,  start_time - INTERVAl '1 MONTH' AS start_date_month_diff
FROM ride_share
```

Output

id	start_time	start_date_day_diff	start_date_hour_diff	start_date_week_diff	start_date_month_diff
1	2012-01-01 00:04:00	2011-12-31 00:04:00	2011-12-31 23:04:00	2011-12-25 00:04:00	2011-12-01 00:04:00
2	2012-01-01 00:10:00	2011-12-31 00:10:00	2011-12-31 23:10:00	2011-12-25 00:10:00	2011-12-01 00:10:00
3	2012-01-01 00:10:00	2011-12-31 00:10:00	2011-12-31 23:10:00	2011-12-25 00:10:00	2011-12-01 00:10:00
4	2012-01-01 00:15:00	2011-12-31 00:15:00	2011-12-31 23:15:00	2011-12-25 00:15:00	2011-12-01 00:15:00
5	2012-01-01 00:15:00	2011-12-31 00:15:00	2011-12-31 23:15:00	2011-12-25 00:15:00	2011-12-01 00:15:00
6	2012-01-01 00:17:00	2011-12-31 00:17:00	2011-12-31 23:17:00	2011-12-25 00:17:00	2011-12-01 00:17:00
7	2012-01-01 00:18:00	2011-12-31 00:18:00	2011-12-31 23:18:00	2011-12-25 00:18:00	2011-12-01 00:18:00
8	2012-01-01 00:22:00	2011-12-31 00:22:00	2011-12-31 23:22:00	2011-12-25 00:22:00	2011-12-01 00:22:00
9	2012-01-01 00:24:00	2011-12-31 00:24:00	2011-12-31 23:24:00	2011-12-25 00:24:00	2011-12-01 00:24:00
10	2012-01-01 00:25:00	2011-12-31 00:25:00	2011-12-31 23:25:00	2011-12-25 00:25:00	2011-12-01 00:25:00

Practice Question - 3

Find the difference (in hours) between the first ride and the last ride per terminal. Show terminal_id, first ride start time, last ride start time, and the difference in hours.

Instructions:

Use tutorial.dc_bikeshare_q1_2012 table. Use the Max and Min functions to find the last and first ride. Subtract the first and last ride time using date_part function.

Solution - 3

```
SELECT
        start_terminal,
        first_ride_start,
        last_ride_start,
        DATE_PART('day',last_ride_start - first_ride_start) *24 + DATE_PART('hour',last_ride_start - first_ride_start) AS diff_hours
FROM(
        SELECT
                start_terminal,
                MIN(start_time) AS first_ride_start,
                MAX(start_time) AS last_ride_start
        FROM
                tutorial.dc_bikeshare_q1_2012
        GROUP BY  start_terminal
) AS a
```

Solution - 3

Output

```
SELECT
  start_terminal,
  first_ride_start,
  last_ride_start,
  DATE_PART('day',last_ride_start - first_ride_start) *24 + DATE_PART('hour',last_ride_start - first_ride_start) AS diff_hours
FROM
(
SELECT
   start_terminal,
   MIN(start_time) AS first_ride_start,
   MAX(start_time) AS last_ride_start
FROM
   tutorial.dc_bikeshare_q1_2012
GROUP BY
  start_terminal
) AS a
```

) rows | 320B returned in 1s

start_terminal	first_ride_start	last_ride_start	diff_hours
31235	2012-01-01 11:41:00	2012-03-31 21:22:00	2169
31003	2012-01-01 12:30:00	2012-03-31 19:13:00	2166
31009	2012-01-01 09:37:00	2012-03-31 18:21:00	2168
31210	2012-01-03 18:38:00	2012-03-31 18:02:00	2111
31616	2012-01-01 10:31:00	2012-03-31 23:37:00	2173
31608	2012-01-01 15:45:00	2012-03-31 23:32:00	2167
31204	2012-01-01 09:54:00	2012-03-31 23:49:00	2173
31604	2012-01-01 01:04:00	2012-03-31 23:31:00	2182
31615	2012-01-01 01:56:00	2012-03-31 23:50:00	2181
31603	2012-01-01 01:13:00	2012-03-31 23:40:00	2182

Subtracting two dates

We can achieve this using the date_part function.

Syntax of Extract Function

Date_part('interval', date1-date2) [date1>date2]

* Refer to the field values discussed earlier to find the interval values

Subtracting two dates

Query

```
SELECT *,
 DATE_PART('minute',end_time - start_time) AS diff_minutes
FROM
ride_share
```

Output

id	start_terminal	end_terminal	start_time	end_time	diff_minutes
1	31245	31109	2012-01-01 00:04:00	2012-01-01 00:11:00	7
2	31400	31103	2012-01-01 00:10:00	2012-01-01 00:29:00	19
3	31400	31103	2012-01-01 00:10:00	2012-01-01 00:29:00	19
4	31101	31602	2012-01-01 00:15:00	2012-01-01 00:23:00	8
5	31102	31109	2012-01-01 00:15:00	2012-01-01 00:23:00	8
6	31017	31014	2012-01-01 00:17:00	2012-01-01 00:23:00	6
7	31236	31404	2012-01-01 00:18:00	2012-01-01 00:47:00	29
8	31101	31201	2012-01-01 00:22:00	2012-01-01 00:27:00	5
9	31014	31237	2012-01-01 00:24:00	2012-01-01 00:33:00	9
10	31101	31218	2012-01-01 00:25:00	2012-01-01 00:40:00	15

Practice Question - 4

Find the time taken to acquire a company.

Instructions:

Join tutorial.crunchbase_companies_clean_date and **tutorial.crunchbase_acquisitions_clean_date** . Take a difference between the founded date and acquired date.

Solution - 4

```
SELECT
        companies.permalink,
        companies.founded_at_clean,
        acquisitions.acquired_at_cleaned,
        acquisitions.acquired_at_cleaned - companies.founded_at_clean::timestamp AS time_to_acquisition
FROM
        tutorial.crunchbase_companies_clean_date companies
JOIN
        tutorial.crunchbase_acquisitions_clean_date acquisitions
ON acquisitions.company_permalink = companies.permalink
WHERE founded_at_clean IS NOT NULL
```

Solution - 4

Output

```
SELECT companies.permalink,
       companies.founded_at_clean,
       acquisitions.acquired_at_cleaned,
       acquisitions.acquired_at_cleaned -
         companies.founded_at_clean::timestamp AS time_to_acquisition
  FROM tutorial.crunchbase_companies_clean_date companies
  JOIN tutorial.crunchbase_acquisitions_clean_date acquisitions
    ON acquisitions.company_permalink = companies.permalink
 WHERE founded_at_clean IS NOT NULL
```

100 rows | 5KB returned in 551ms

permalink	founded_at_clean	acquired_at_cleaned	time_to_acquisition
/company/waywire	2012-06-01	2013-10-17 00:00:00	503 days
/company/1000memories	2010-07-01	2012-10-03 00:00:00	825 days
/company/12society	2012-01-01	2013-07-03 00:00:00	549 days
/company/3leaf	2004-06-01	2011-02-19 00:00:00	2454 days
/company/3x-systems	2007-11-01	2012-10-25 00:00:00	1820 days
/company/4home	2006-01-01	2010-12-01 00:00:00	1795 days
/company/5to1	2009-01-01	2011-05-01 00:00:00	850 days
/company/60mo	2009-06-01	2012-05-03 00:00:00	1067 days

Practice Question - 5

Write a query to find how long ago a company is founded.

Instructions:

Use tutorial.crunchbase_companies_clean_date. Use NOW() function and subtract it with the founded date.

Solution - 5

```
SELECT
        companies.permalink,
        companies.founded_at_clean,
        NOW() - companies.founded_at_clean::timestamp AS founded_time_ago
FROM
        tutorial.crunchbase_companies_clean_date companies
WHERE
        founded_at_clean IS NOT NULL
```

Solution - 5

Output

```
SELECT companies.permalink,
       companies.founded_at_clean,
       NOW() - companies.founded_at_clean::timestamp AS founded_time_ago
  FROM tutorial.crunchbase_companies_clean_date companies
 WHERE founded_at_clean IS NOT NULL
```

100 rows | 5KB returned in 410ms

permalink	founded_at_clean	founded_time_ago
/company/21e6	2013-01-01	3354 days 07:32:28.817704
/company/club-domains	2011-10-10	3803 days 07:32:28.817704
/company/pay-mobile-checkout	2011-05-01	3965 days 07:32:28.817704
/company/waywire	2012-06-01	3568 days 07:32:28.817704
/company/0-6-com	2007-01-01	5546 days 07:32:28.817704
/company/0xdata	2011-01-01	4085 days 07:32:28.817704
/company/1-800-doctors	1984-01-01	13947 days 07:32:28.817704
/company/10-20-media	2001-01-01	7737 days 07:32:28.817704
/company/1000memories	2010-07-01	4269 days 07:32:28.817704
/company/1000museums-com	2008-01-01	5181 days 07:32:28.817704
/company/1001-menus	2010-11-20	4127 days 07:32:28.817704

Practice Question - 6

Write a query that counts the number of companies acquired within 3 years, 5 years, and 10 years of being founded (in 3 separate columns). Include a column for total companies acquired as well. Group by the category and limit to only rows with a founding date.

Instructions:

Use tutorial.crunchbase_companies_clean_date. Use NOW() function and subtract it with the founded date.

Solution - 6

```
SELECT
        companies.category_code,
        COUNT(CASE WHEN acquisitions.acquired_at_cleaned <= companies.founded_at_clean::timestamp + INTERVAL '3 years' THEN 1
ELSE NULL END) AS acquired_3_yrs,
        COUNT(CASE WHEN acquisitions.acquired_at_cleaned <= companies.founded_at_clean::timestamp + INTERVAL '5 years' THEN 1
ELSE NULL END) AS acquired_5_yrs,
        COUNT(CASE WHEN acquisitions.acquired_at_cleaned <= companies.founded_at_clean::timestamp + INTERVAL '10 years' THEN 1
ELSE NULL END) AS acquired_10_yrs,
        COUNT(1) AS total
FROM
        tutorial.crunchbase_companies_clean_date companies
JOIN
        tutorial.crunchbase_acquisitions_clean_date acquisitions
ON acquisitions.company_permalink = companies.permalink
WHERE founded_at_clean IS NOT NULL
GROUP BY 1
ORDER BY 5 DESC
```

Solution - 6

Output

```
SELECT
  companies.category_code,
  COUNT(CASE WHEN acquisitions.acquired_at_cleaned <= companies.founded_at_clean::timestamp + INTERVAL
  COUNT(CASE WHEN acquisitions.acquired_at_cleaned <= companies.founded_at_clean::timestamp + INTERVAL
  COUNT(CASE WHEN acquisitions.acquired_at_cleaned <= companies.founded_at_clean::timestamp + INTERVAL
  COUNT(1) AS total
FROM
  tutorial.crunchbase_companies_clean_date companies JOIN tutorial.crunchbase_acquisitions_clean_date a
  ON acquisitions.company_permalink = companies.permalink
  WHERE founded_at_clean IS NOT NULL
```

0 rows | 2KB returned in 820ms

category_code	acquired_3_yrs	acquired_5_yrs	acquired_10_yrs	total
software	36	72	165	240
web	63	90	124	129
enterprise	17	38	82	115
mobile	22	53	84	109
advertising	17	40	63	71
games_video	19	42	60	69
ecommerce	26	38	50	60

Curious Case of Date Time Function

Unlike aggregate and windows functions, time date functions depend significantly on the database type. The functions which work on one database might not work on another.

For Example:

- In the PostgreSQL database, we use the date_part() function to take the difference between two dates. In the MySQL database, we can use the DATEDIFF function.

- Similarly, MYSQL uses CUR_TIME CUR_DATE to find the current time or date. Whereas PostgreSQL use CURRENT_TIME and CURRENT_DATE

In this session, we have taught functions by PostgreSQL. We will advise referring to the database type before using the date/time function.

In the next class we will study:

String Functions and Pivoting Data

THANK YOU