



Date/time types and formats

Christina Maimone
Data Scientist



Main types

date

- YYYY-MM-DD
- example: 2018-12-30

timestamp

- YYYY-MM-DD HH:MM:SS
- example: 2018-12-30 13:10:04.3



Intervals

interval examples:

```
6 days 01:48:08

00:51:03

1 day 21:57:47

07:48:46

406 days 00:31:56
```



Date/time format examples

1pm on January 10, 2018

01/10/18 1:00

10/01/18 01:00:00

01/10/2018 1pm

January 10th, 2018 1pm

10 Jan 2018 1:00

01/10/18 01:00:00

01/10/18 13:00:00



ISO 8601

ISO = International Organization for Standards

YYYY-MM-DD HH:MM:SS

Example: 2018-01-05 09:35:15



UTC and timezones

UTC = Coordinated Universal Time

Timestamp with timezone:

YYYY-MM-DD HH:MM:SS+HH

Example: 2004-10-19 10:23:54+02



Date and time comparisons

```
Compare with >, <, =
```

```
SELECT '2018-01-01' > '2017-12-31';
```

now(): current timestamp

```
SELECT now() > '2017-12-31';
```



Date subtraction

```
SELECT now() - '2018-01-01';

343 days 21:26:32.710898

SELECT now() - '2015-01-01';

1439 days 21:32:22.616076
```



Date addition

```
SELECT '2010-01-01'::date + 1;

2010-01-02

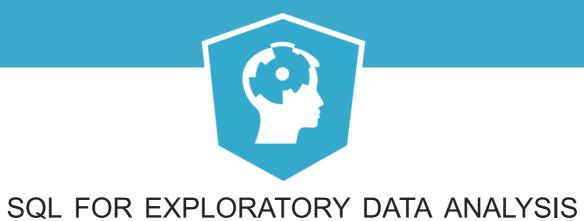
SELECT '2018-12-10'::date + '1 year'::interval;

2019-12-10 00:00:00

SELECT '2018-12-10'::date + '1 year 2 days 3 minutes'::interval;

2019-12-12 00:03:00
```





Let's practice!





Date/time components and aggregation

Christina Maimone
Data Scientist



Common date/time fields

Date/Time Functions and Operators Documentation

Fields

- century: 2019-01-01 = century 21
- decade: 2019-01-01 = decade 201
- year, month, day
- hour, minute, second
- week
- dow: day of week



Extracting fields



Extract to summarize by field

Individual sales

```
SELECT *
   FROM sales
   WHERE date >= '2010-01-01'
   AND date < '2017-01-01';
```

By month

```
SELECT date_part('month', date)

AS month,

sum(amt)

FROM sales

GROUP BY month

ORDER BY month;
```



Truncating dates



Truncate to keep larger units

Individual sales

```
SELECT *
   FROM sales
   WHERE date >= '2017-06-01'
   AND date < '2019-02-01';
```

```
date
                      amt
2019-01-25 06:58:32
2018-03-14 18:07:03
2018-04-15 19:03:33
2018-08-10 14:30:49
                       28
2018-02-02 09:52:28
2017-08-08 08:40:35
                       20
2018-07-05 02:05:52
2018-07-28 17:49:16
                       16
2018-08-01 20:57:40
                         8
2018-03-04 08:56:04
```

By month with year

```
SELECT date_trunc('month', date)

AS month

sum(amt)

FROM sales

GROUP BY month

ORDER BY month;
```

```
month
                        sum
2017-06-01 00:00:00
                       594
2017-07-01 00:00:00
                      3824
2017-08-01 00:00:00
                       482
2017-09-01 00:00:00
                      1384
2017-10-01 00:00:00
                      3058
                       259
2017-11-01 00:00:00
2017-12-01 00:00:00
                       874
2018-01-01 00:00:00 | 1225
```





Time to practice extracting and aggregating dates





Aggregating with date/time series

Christina Maimone
Data Scientist



Generate series

```
SELECT generate series(from, to, interval);
SELECT generate_series('2018-01-01',
                       '2018-01-15',
                       '2 days'::interval);
  generate_series
2018-01-01 00:00:00
2018-01-03 00:00:00
2018-01-05 00:00:00
2018-01-07 00:00:00
2018-01-09 00:00:00
2018-01-11 00:00:00
2018-01-13 00:00:00
2018-01-15 00:00:00
(8 rows)
```



Generate series



Generate series from the beginning



Generate series from the beginning



Normal aggregation

```
SELECT * FROM sales;
```

```
date
                       amount
2018-04-23 09:13:14
2018-04-23 13:57:53
2018-04-23 12:05:44
                           23
2018-04-23 09:07:33
2018-04-23 10:31:40
                           18
2018-04-23 09:35:16
                           19
2018-04-23 12:17:43
2018-04-23 12:57:49
2018-04-23 10:12:35
                           13
2018-04-23 13:21:30
(10 rows)
```

```
SELECT date_trunc('hour', date)

AS hour,

count(*)

FROM sales

GROUP BY hour

ORDER BY hour;
```



Aggregation with series



Aggregation with series: result



Aggregation with bins

```
-- Create bins
WITH bins AS (
      SELECT generate series ('2018-04-23 09:00:00',
                             '2018-04-23 15:00:00',
                             '3 hours'::interval) AS lower,
             generate series('2018-04-23 12:00:00',
                              '2018-04-23 18:00:00',
                              '3 hours'::interval) AS upper)
-- Count values in each bin
SELECT lower, upper, count (date)
  -- left join keeps all bins
  FROM bins
       LEFT JOIN sales
              ON date >= lower
             AND date < upper
 -- Group by bin bounds to create the groups
 GROUP BY lower, upper
 ORDER BY lower;
```



Bin result

| lower | upper | count |
|---------------------|---|-----------------|
| 2018-04-23 12:00:00 | 2018-04-23 12:00:00 2018-04-23 15:00:00 2018-04-23 18:00:00 | 5 5 0 |





Practice generating series!





Time between events

Christina Maimone
Data Scientist



The problem

```
SELECT *
FROM sales
ORDER BY date;
```

```
date
                      amount
2018-04-23 09:07:33 |
                          31
2018-04-23 09:13:14 |
                         12
2018-04-23 09:35:16
                         18
2018-04-23 10:12:35
                          13
2018-04-23 10:31:40
                          23
2018-04-23 12:05:44
2018-04-23 12:17:43
                          19
2018-04-23 12:57:49
                          32
2018-04-23 13:21:30 |
2018-04-23 13:57:53 |
(10 rows)
```



Lead and lag

```
SELECT date,
lag(date) OVER (ORDER BY date),
lead(date) OVER (ORDER BY date)
FROM sales;
```

```
date
                              lag
                                                   lead
                                            2018-04-23 09:13:14
2018-04-23 09:07:33
2018-04-23 09:13:14
                      2018-04-23 09:07:33 | 2018-04-23 09:35:16
2018-04-23 09:35:16
                      2018-04-23 09:13:14
                                            2018-04-23 10:12:35
2018-04-23 10:12:35
                      2018-04-23 09:35:16
                                          | 2018-04-23 10:31:40
2018-04-23 10:31:40
                      2018-04-23 10:12:35 | 2018-04-23 12:05:44
2018-04-23 12:05:44
                     2018-04-23 10:31:40
                                          | 2018-04-23 12:17:43
2018-04-23 12:17:43 | 2018-04-23 12:05:44 | 2018-04-23 12:57:49
2018-04-23 12:57:49 | 2018-04-23 12:17:43 | 2018-04-23 13:21:30
2018-04-23 13:21:30 | 2018-04-23 12:57:49 | 2018-04-23 13:57:53
2018-04-23 13:57:53 | 2018-04-23 13:21:30
(10 rows)
```



Lead and lag

```
SELECT date,
    lag(date) OVER (ORDER BY date),
    lead(date) OVER (ORDER BY date)
FROM sales;
```



Time between events

```
SELECT date,
date - lag(date) OVER (ORDER BY date) AS gap
FROM sales;
```



Average time between events

```
SELECT avg(gap)
FROM (SELECT date - lag(date) OVER (ORDER BY date) AS gap
FROM sales) AS gaps;
```

```
avg
------
00:32:15.555556
(1 row)
```



Change in a time series

```
SELECT date,
amount,
lag(amount) OVER (ORDER BY date),
amount - lag(amount) OVER (ORDER BY date) AS change
FROM sales;
```

```
date
                   | amount | lag | change
2018-04-23 09:07:33
2018-04-23 09:13:14
                                      -19
                             12 |
2018-04-23 09:35:16
                             18
2018-04-23 10:12:35
2018-04-23 10:31:40
                             13 |
                                     18
2018-04-23 12:05:44
2018-04-23 12:17:43
                        19 |
                             23 |
                                     -4
2018-04-23 12:57:49
                                     13
2018-04-23 13:21:30
                             32 |
                                      -26
2018-04-23 13:57:53
                                       35
(10 rows)
```





On to the exercises!





Wrap-up

Christina Maimone
Data Scientist



Download the data

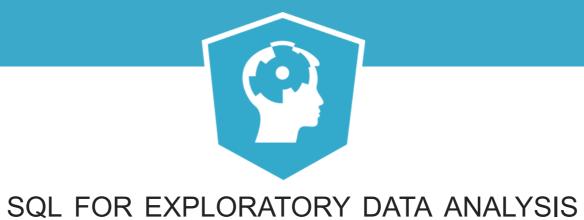
Links on the course landing page!



Parting tips

- Spend time exploring your data
- Use the PostgreSQL documentation
- Be curious
- Check data distributions first





Start exploring!