Postgres Cheatsheet

Contents

[Postgres Cheatsheet 1](#_Toc452434619)

[Admin Functions 3](#_Toc452434620)

[What users are using the database? 3](#_Toc452434621)

[Aggregate Functions 3](#_Toc452434622)

[Aggregate Functions for Statistics 3](#_Toc452434623)

[Average 4](#_Toc452434624)

[Conditionals 4](#_Toc452434625)

[Case Statements 4](#_Toc452434626)

[Data Definition Language (DDL) 5](#_Toc452434627)

[Creae a Table 5](#_Toc452434628)

[Create a Table from a Query 5](#_Toc452434629)

[Drop a Table 5](#_Toc452434630)

[Dates and Times 5](#_Toc452434631)

[Current Date and Time (UTC) 5](#_Toc452434632)

[Extract a Date part 5](#_Toc452434633)

[Formatting Dates and Times 5](#_Toc452434634)

[Convert to timestamp without time zone 10](#_Toc452434635)

[Interval between two dates 10](#_Toc452434636)

[Interval in hours 12](#_Toc452434637)

[Command Line Queries 13](#_Toc452434638)

[Execute a query on the command line 13](#_Toc452434639)

[Errors 13](#_Toc452434640)

[Error: out of memory for query result 13](#_Toc452434641)

[Functions 13](#_Toc452434642)

[Change Function Ownership 13](#_Toc452434643)

[Grant Function Privileges 13](#_Toc452434644)

[Geography 14](#_Toc452434645)

[Distance between two points 14](#_Toc452434646)

[Importing Data 14](#_Toc452434647)

[Importing CSV File into a PosgreSQL Table 14](#_Toc452434648)

[Mathematical Functions 14](#_Toc452434649)

[Strings 17](#_Toc452434650)

[String length 17](#_Toc452434651)

[Tables 17](#_Toc452434652)

[Change table ownership 17](#_Toc452434653)

[Describe a table 17](#_Toc452434654)

[Get column names from a table 17](#_Toc452434655)

# Admin Functions

## What users are using the database?

SELECT datname,usename,procpid,client\_addr,waiting,query\_start,current\_query

FROM pg\_stat\_activity;

# Aggregate Functions



## Aggregate Functions for Statistics



## Average

avg(expression)

# Conditionals

## Case Statements

select origin, orig\_poi.long\_name as orig\_plant,

destination,

case when (status\_lat = 34.0592 and status\_lng = -118.2481) then 'Union Station'

when (status\_lat = 34.108889 and status\_lng = -117.298056) then 'San Bernardino'

when (status\_lat = 34.0000 and status\_lng = -118.1589) then 'East LA'

else 'Surprise'

end as end\_train\_station,

shipment\_count

from …

# Data Definition Language (DDL)

## Creae a Table

create table training\_carrier\_blacklist (SCAC\_CODE varchar(10),

PRIMARY KEY (SCAC\_CODE))

;

## Create a Table from a Query

create table model\_train\_prod1 as

select distinct a.sap\_shipment\_id, a.carrier\_scac

from carrier\_edi\_events a

where a.sap\_shipment\_id != '0' and a.status\_code!='AG';

## Drop a Table

drop table if exists model\_train\_prod2;

# Dates and Times

## Current Date and Time (UTC)

select now();

Output:

"2016-05-31 11:59:55.04674+00"

## Extract a Date part

select status\_lat, status\_lng,

extract(HOUR from status\_dt\_utc) as hr

from model\_train17

where sap\_shipment\_id > 0

and orig\_lat is not null

and orig\_lon is not null

and dest\_lat is not null

and dest\_lon is not null

and stops = 1

and actual\_shipment\_end\_date\_utc is not null

and status\_lat != 0

and status\_lng != 0

;

## Formatting Dates and Times

Table 9-21. Template Patterns for Date/Time Formatting

| **Pattern** | **Description** |
| --- | --- |
| HH | hour of day (01-12) |
| HH12 | hour of day (01-12) |
| HH24 | hour of day (00-23) |
| MI | minute (00-59) |
| SS | second (00-59) |
| MS | millisecond (000-999) |
| US | microsecond (000000-999999) |
| SSSS | seconds past midnight (0-86399) |
| AM or A.M. or PM or P.M. | meridian indicator (uppercase) |
| am or a.m. or pm or p.m. | meridian indicator (lowercase) |
| Y,YYY | year (4 and more digits) with comma |
| YYYY | year (4 and more digits) |
| YYY | last 3 digits of year |
| YY | last 2 digits of year |
| Y | last digit of year |
| IYYY | ISO year (4 and more digits) |
| IYY | last 3 digits of ISO year |
| IY | last 2 digits of ISO year |
| I | last digits of ISO year |
| BC or B.C. or AD or A.D. | era indicator (uppercase) |
| bc or b.c. or ad or a.d. | era indicator (lowercase) |
| MONTH | full uppercase month name (blank-padded to 9 chars) |
| Month | full mixed-case month name (blank-padded to 9 chars) |
| month | full lowercase month name (blank-padded to 9 chars) |
| MON | abbreviated uppercase month name (3 chars in English, localized lengths vary) |
| Mon | abbreviated mixed-case month name (3 chars in English, localized lengths vary) |
| mon | abbreviated lowercase month name (3 chars in English, localized lengths vary) |
| MM | month number (01-12) |
| DAY | full uppercase day name (blank-padded to 9 chars) |
| Day | full mixed-case day name (blank-padded to 9 chars) |
| day | full lowercase day name (blank-padded to 9 chars) |
| DY | abbreviated uppercase day name (3 chars in English, localized lengths vary) |
| Dy | abbreviated mixed-case day name (3 chars in English, localized lengths vary) |
| dy | abbreviated lowercase day name (3 chars in English, localized lengths vary) |
| DDD | day of year (001-366) |
| DD | day of month (01-31) |
| D | day of week (1-7; Sunday is 1) |
| W | week of month (1-5) (The first week starts on the first day of the month.) |
| WW | week number of year (1-53) (The first week starts on the first day of the year.) |
| IW | ISO week number of year (The first Thursday of the new year is in week 1.) |
| CC | century (2 digits) (The twenty-first century starts on 2001-01-01.) |
| J | Julian Day (days since January 1, 4712 BC) |
| Q | quarter |
| RM | month in Roman numerals (I-XII; I=January) (uppercase) |
| rm | month in Roman numerals (i-xii; i=January) (lowercase) |
| TZ | time-zone name (uppercase) |
| tz | time-zone name (lowercase) |

Certain modifiers may be applied to any template pattern to alter its behavior. For example, FMMonth is the Month pattern with the FM modifier. [Table 9-22](http://www.postgresql.org/docs/8.2/static/functions-formatting.html#FUNCTIONS-FORMATTING-DATETIMEMOD-TABLE) shows the modifier patterns for date/time formatting.

Table 9-22. Template Pattern Modifiers for Date/Time Formatting

| **Modifier** | **Description** | **Example** |
| --- | --- | --- |
| FM prefix | fill mode (suppress padding blanks and zeroes) | FMMonth |
| TH suffix | uppercase ordinal number suffix | DDTH |
| th suffix | lowercase ordinal number suffix | DDth |
| FX prefix | fixed format global option (see usage notes) | FX Month DD Day |
| TM prefix | translation mode (print localized day and month names based on lc\_messages) | TMMonth |
| SP suffix | spell mode (not yet implemented) | DDSP |

Usage notes for date/time formatting:

* FM suppresses leading zeroes and trailing blanks that would otherwise be added to make the output of a pattern be fixed-width.
* TM does not include trailing blanks.
* to\_timestamp and to\_date skip multiple blank spaces in the input string if the FX option is not used. FX must be specified as the first item in the template. For example to\_timestamp('2000    JUN', 'YYYY MON') is correct, but to\_timestamp('2000    JUN', 'FXYYYY MON') returns an error, because to\_timestamp expects one space only.
* Ordinary text is allowed in to\_char templates and will be output literally. You can put a substring in double quotes to force it to be interpreted as literal text even if it contains pattern key words. For example, in '"Hello Year "YYYY', the YYYY will be replaced by the year data, but the single Y in Year will not be.
* If you want to have a double quote in the output you must precede it with a backslash, for example E'\\"YYYY Month\\"'. (Two backslashes are necessary because the backslash already has a special meaning when using the escape string syntax.)
* The YYYY conversion from string to timestamp or date has a restriction if you use a year with more than 4 digits. You must use some non-digit character or template after YYYY, otherwise the year is always interpreted as 4 digits. For example (with the year 20000): to\_date('200001131', 'YYYYMMDD') will be interpreted as a 4-digit year; instead use a non-digit separator after the year, like to\_date('20000-1131', 'YYYY-MMDD') or to\_date('20000Nov31', 'YYYYMonDD').
* In conversions from string to timestamp or date, the CC field is ignored if there is a YYY, YYYY or Y,YYY field. If CC is used with YY or Y then the year is computed as (CC-1)\*100+YY.
* Millisecond (MS) and microsecond (US) values in a conversion from string to timestamp are used as part of the seconds after the decimal point. For example to\_timestamp('12:3', 'SS:MS') is not 3 milliseconds, but 300, because the conversion counts it as 12 + 0.3 seconds. This means for the format SS:MS, the input values 12:3, 12:30, and 12:300 specify the same number of milliseconds. To get three milliseconds, one must use 12:003, which the conversion counts as 12 + 0.003 = 12.003 seconds.

Here is a more complex example: to\_timestamp('15:12:02.020.001230', 'HH:MI:SS.MS.US') is 15 hours, 12 minutes, and 2 seconds + 20 milliseconds + 1230 microseconds = 2.021230 seconds.

* to\_char's day of the week numbering (see the 'D' formatting pattern) is different from that of the extract function.
* to\_char(interval) formats HH and HH12 as hours in a single day, while HH24 can output hours exceeding a single day, e.g. >24.

## Convert to timestamp without time zone

to\_timestamp(e.edi\_xfer\_date || ' ' || e.edi\_xfer\_time, 'YYYY-MM-DD HH24:MI:SS')::timestamp without time zone

## Interval between two dates

select s.sap\_shipment\_id, status\_code, status\_dt\_utc, edi\_xfer\_date, edi\_xfer\_time, s.actual\_shipment\_end\_date\_utc,

(s.actual\_shipment\_end\_date\_utc - status\_dt\_utc) as delta

from edi\_msgs e,

shipment\_msgs s

where status\_code = 'X1'

and s.sap\_shipment\_id = e.sap\_shipment\_id

and s.actual\_shipment\_end\_date\_utc is not null

and s.actual\_shipment\_end\_date\_utc <> status\_dt\_utc

limit 20;



### Interval in hours

select leg\_start\_dt\_utc, leg\_end\_dt\_utc,

EXTRACT(epoch from (leg\_end\_dt\_utc - leg\_start\_dt\_utc)) / 3600 as duration

from model\_train\_dev18

limit 5;



## Subtract an Interval from a Date

select now(), now() - INTERVAL '24 hours';

Output:

"2016-05-31 12:13:54.183421+00" "2016-05-30 12:13:54.183421+00"

# Command Line Queries

## Execute a query on the command line

c:\psqltest>psql --host=prod-savi-analytics-db.c9yercffrnb2.us-east-1.rds.amazonaws.com --port=5432 --username=bbeauchamp --dbname=dbanalytics --file=psqltest.sql

# Errors

## Error: out of memory for query result

This appears to be a client-side error which occurs for large queries. A work-around is to create a temporary table which contains the entire output for the desired query, and then just use a simple “SELECT \* from <temp\_table\_name>” query to extract the data.

# Functions

## Change Function Ownership

ALTER FUNCTION round\_even\_two(nnum double precision) OWNER TO datascience;

## Grant Function Privileges

grant all on FUNCTION round\_even\_two(nnum double precision) to datascience

# Geography

## Distance between two points

SELECT ST\_Distance(

ST\_Transform(ST\_GeomFromText('POINT(-121.069 38.896)',4326),2163),

ST\_Transform(ST\_GeomFromText('POINT(-120.842 37.501)', 4326),2163)

) / 1000 as dist\_km;



# Importing Data

## Importing CSV File into a PosgreSQL Table

See <http://www.postgresqltutorial.com/import-csv-file-into-posgresql-table/>

# Mathematical Functions



| **Function** | **Return Type** | **Description** | **Example** | **Result** |
| --- | --- | --- | --- | --- |
| abs(*x*) | (same as *x*) | absolute value | abs(-17.4) | 17.4 |
| cbrt(dp) | dp | cube root | cbrt(27.0) | 3 |
| ceil(dp or numeric) | (same as input) | smallest integer not less than argument | ceil(-42.8) | -42 |
| ceiling(dp or numeric) | (same as input) | smallest integer not less than argument (alias for ceil) | ceiling(-95.3) | -95 |
| degrees(dp) | dp | radians to degrees | degrees(0.5) | 28.6478897565412 |
| exp(dp or numeric) | (same as input) | exponential | exp(1.0) | 2.71828182845905 |
| floor(dp or numeric) | (same as input) | largest integer not greater than argument | floor(-42.8) | -43 |
| ln(dp or numeric) | (same as input) | natural logarithm | ln(2.0) | 0.693147180559945 |
| log(dp or numeric) | (same as input) | base 10 logarithm | log(100.0) | 2 |
| log(b numeric, x numeric) | numeric | logarithm to base b | log(2.0, 64.0) | 6.0000000000 |
| mod(y, x) | (same as argument types) | remainder of y/x | mod(9,4) | 1 |
| pi() | dp | "π" constant | pi() | 3.14159265358979 |
| power(a dp, b dp) | dp | a raised to the power of b | power(9.0, 3.0) | 729 |
| power(a numeric, b numeric) | numeric | a raised to the power of b | power(9.0, 3.0) | 729 |
| radians(dp) | dp | degrees to radians | radians(45.0) | 0.785398163397448 |
| random() | dp | random value between 0.0 and 1.0 | random() |  |
| round(dp or numeric) | (same as input) | round to nearest integer | round(42.4) | 42 |
| round(v numeric, s int) | numeric | round to s decimal places | round(42.4382, 2) | 42.44 |
| setseed(dp) | int | set seed for subsequent random() calls | setseed(0.54823) | 1177314959 |
| sign(dp or numeric) | (same as input) | sign of the argument (-1, 0, +1) | sign(-8.4) | -1 |
| sqrt(dp or numeric) | (same as input) | square root | sqrt(2.0) | 1.4142135623731 |
| trunc(dp or numeric) | (same as input) | truncate toward zero | trunc(42.8) | 42 |
| trunc(v numeric, s int) | numeric | truncate to s decimal places | trunc(42.4382, 2) | 42.43 |
| width\_bucket(op numeric, b1 numeric, b2 numeric, count int) | int | return the bucket to which operand would be assigned in an equidepth histogram with count buckets, an upper bound of b1, and a lower bound of b2 | width\_bucket(5.35, 0.024, 10.06, 5) | 3 |

# Strings

## String length

select \*

from delivery\_msgs

where length(x\_box\_load\_id) > 0;

# Tables

## Change table ownership

alter table public.model\_train\_dev17 OWNER to analytics;

## Describe a table

select column\_name, data\_type, character\_maximum\_length

from INFORMATION\_SCHEMA.COLUMNS where table\_name = 'delivery\_msgs';

## Get column names from a table

select column\_name from information\_schema.columns

where table\_name = 'model\_train17'