Table of Contents

[EXCEL operations 4](#_Toc83222712)

[Add PST string to datetime in Excel 4](#_Toc83222713)

[BASIC Database and Table operations 4](#_Toc83222714)

[PostgreSQL download 4](#_Toc83222715)

[Check size of tables and objects 4](#_Toc83222716)

[Copy CSV file to table 4](#_Toc83222717)

[Copy table to CSV file 4](#_Toc83222718)

[Copy table approaches 4](#_Toc83222719)

[1. To copy a table completely 4](#_Toc83222720)

[2. To copy a table structure without data 5](#_Toc83222721)

[3. To copy a table with partial data from an existing table 5](#_Toc83222722)

[Proper names need quotes 5](#_Toc83222723)

[Find duplicates in a table, id by some columns (Stackoverflow) 5](#_Toc83222724)

[Select distinct on 5](#_Toc83222725)

[Create database table with Primary Key 5](#_Toc83222726)

[Add a geometry column from lat/long points in table 6](#_Toc83222727)

[Add an index 6](#_Toc83222728)

[Set table SRID 6](#_Toc83222729)

[Setting parameter based upon nearest match 6](#_Toc83222730)

[Convert (long,lat) to projected points 7](#_Toc83222731)

[Check your geometry in the classic tuple 7](#_Toc83222732)

[Merge data from tables on approximate match 7](#_Toc83222733)

[Selection boxes 8](#_Toc83222734)

[1. geog\_geometry && ST\_MakeEnvelope 8](#_Toc83222735)

[2. ST\_Within and ST\_MakeEnvelope 8](#_Toc83222736)

[3. ST\_Within and ST\_MakeBox2D 9](#_Toc83222737)

[TABLE JOINS 9](#_Toc83222738)

[Select information with outer left join 9](#_Toc83222739)

[Select points within polygon in Salish Sea (add semi-colon) 9](#_Toc83222740)

[Join Bottle data and header 9](#_Toc83222741)

[dfo\_bottle\_data\_2017 10](#_Toc83222742)

[Create ctd linestrings with text as output 11](#_Toc83222743)

[Add a geometry column 17](#_Toc83222744)

[LINESTRING AND POLYGON OPERATIONS 17](#_Toc83222745)

[Forming linestring from points 17](#_Toc83222746)

[Commands to create gps point table and then linestrings 17](#_Toc83222747)

[Points from GPS to lines 1 17](#_Toc83222748)

[Points from GPS to lines 2 18](#_Toc83222749)

[Create table for Sidney and James Islands 18](#_Toc83222750)

[Create database table with line information 19](#_Toc83222751)

[Create ship tracks 20](#_Toc83222752)

[Midpoint on a MultiLineString 20](#_Toc83222753)

[PostGIS Query for a Point Within a Polygon 20](#_Toc83222754)

[Find point in one table nearest to polygon in another table 20](#_Toc83222755)

[Updating fields in specific rows with Geometry text 21](#_Toc83222756)

[WATER PROPERTY TABLE from tables of header and cast data 21](#_Toc83222757)

[Create table cascade for osd ctd data 22](#_Toc83222758)

[CREATE TABLE osd\_ctd\_southern 22](#_Toc83222759)

[CREATE TABLE osd\_ctd\_arctic 22](#_Toc83222760)

[CREATE TABLE osd\_ctd\_atlantic 22](#_Toc83222761)

[CREATE TABLE osd\_ctd\_indian 22](#_Toc83222762)

[CREATE TABLE osd\_ctd\_chinasea 22](#_Toc83222763)

[CREATE TABLE osd\_ctd\_pac\_west 22](#_Toc83222764)

[CREATE TABLE osd\_ctd\_pac\_noreast 22](#_Toc83222765)

[CREATE TABLE osd\_ctd\_pac\_soueast 22](#_Toc83222766)

[CREATE TABLE osd\_ctd\_ssea 23](#_Toc83222767)

[Rows in partitioned osd\_ctd\_2017 tables 23](#_Toc83222768)

[SHOREZONE 23](#_Toc83222769)

[Add shorezone table 24](#_Toc83222770)

[Create Shorezone track from a series of gps points 24](#_Toc83222771)

[PostgreSQL getting daily, weekly, and monthly averages of the occurrences of an event in one query 26](#_Toc83222772)

[Select using “distinct on” and “extract” 26](#_Toc83222773)

[Daily averages 26](#_Toc83222774)

[Weekly averages 27](#_Toc83222775)

[Daily averages 27](#_Toc83222776)

[Daily averages 28](#_Toc83222777)

[Weekly averages from joined tables 28](#_Toc83222778)

[Weekly averages from joined tables 29](#_Toc83222779)

[CITIZEN SCIENCE TABLES 29](#_Toc83222780)

[Drop existing table and create new 29](#_Toc83222781)

[Load table from csv, add geometry column from lat/long 29](#_Toc83222782)

[WEATHER TABLES 30](#_Toc83222783)

[DROP TABLE IF EXISTS, Create table 30](#_Toc83222784)

[DROP TABLE IF EXISTS, Create table 30](#_Toc83222785)

[FORAGE FISH TABLES 31](#_Toc83222786)

[Create forage fish table 31](#_Toc83222787)

# EXCEL operations

## Add PST string to datetime in Excel

=TEXT(E2,"yyyy-mm-dd hh:mm:ss")&"-8:00"

# BASIC Database and Table operations

PostgreSQL download

https\_\_\_download.postgresql.org\_pub\_repos\_yum\_9.6\_redhat\_rhel-6-x86\_64\_pgdg-redhat96-9.6-3.noarch.rpm

## Check size of tables and objects

Check size of tables and objects in PostgreSQL database

SELECT

relname as "Table",

pg\_size\_pretty(pg\_total\_relation\_size(relid)) As "Size",

pg\_size\_pretty(pg\_total\_relation\_size(relid) - pg\_relation\_size(relid)) as "External Size"

FROM pg\_catalog.pg\_statio\_user\_tables ORDER BY pg\_total\_relation\_size(relid) DESC;

## Copy CSV file to table

\COPY nodc\_ctd\_locations FROM F\_\gisData\2014nodcData\ctd\files4export\nodcCtdLocs.csv DELIMITERS , CSV HEADER;

Sample point (no commas or quotes)\_ POINT(-110 30)

## Copy table to CSV file

COPY hab\_positioned TO '/home/terry/hab\_positioned.csv' DELIMITER ',' CSV HEADER;

\*\*\* since I signed in as postgres, I required pre-existing hab\_positioned.csv file (created with VI) and owned by postgres. Also gave full permissions.

## Copy table approaches

1. To copy a table completely, including both table structure and data, you use the following statement:

CREATE TABLE new\_table AS

TABLE existing\_table;

2. To copy a table structure without data, you add the WITH NO DATA clause to the CREATE TABLE statement as follows:

CREATE TABLE new\_table AS

TABLE existing\_table

WITH NO DATA;

3. To copy a table with partial data from an existing table, you use the following statement:

CREATE TABLE new\_table AS

SELECT \*

FROM existing\_table

WHERE condition;

## Proper names need quotes

Select t1.level4name AS name,long,lat,x\_utm10n as x,y\_utm10n AS y,

t2."VisitDate"::date AS date,t2."StartTime"::TIME AS start, t2."EndTime"::TIME AS end

FROM level4 AS t1

INNER JOIN "Visits" AS t2

ON t1.level4id = t2."Level4ID";

LIMIT 5;

## Find duplicates in a table, id by some columns (Stackoverflow)

SELECT castid,depth, count(\*)

FROM ctd\_2017

GROUP BY castid,depth

HAVING count(\*) > 1;

## Select distinct on

select distinct on (left(castid,7)) \* from osd\_ctd\_2017

where ST\_Within(geog\_\_geometry, ST\_GeomFromEWKT(SRID=4326;POLYGON((-126 49,-122 49, -122 50,-126 50, -126 49)))) limit 20;

## Create database table with Primary Key

CREATE TABLE dfo\_cast\_info(

castID varchar(15) PRIMARY KEY,

mission varchar(20),

event varchar(10),

station varchar(20),

latitude numeric(12,5),

longitude numeric(12,5),

datetime timestamptz,

waterDepth real,

country varchar(20),

project varchar(60),

region varchar(60),

PI varchar(30),

platform varchar(30),

agency varchar(60),

probe\_type varchar(30),

instrument varchar(60),

channels smallint,

records smallint,

the\_geog geography(POINT, 4326) );

\*\*\* "the\_geog" wording is important to QGIS (but not GN or GS)\*\*\*

### Add a geometry column from lat/long points in table

* add a geometry column (quotes for vars),
* add (lat, long) points from the table,
* add index\_

SELECT AddGeometryColumn (schema,table,geom,4326,POINT,2);

UPDATE table SET geom = ST\_SetSRID(ST\_MakePoint(long, lat), 4326);

### Add an index

\*\*\* NEED AN INDEX to find lat\_long with GS

CREATE INDEX jacksonco\_streets\_gix ON jacksonco\_streets USING GIST (the\_geom);

## Set table SRID

ALTER TABLE mytable

ALTER COLUMN geom

TYPE geometry(Point, 4326)

USING ST\_SetSRID(geom, 4326);

## Setting parameter based upon nearest match

with another table field (needs coords with same geom)

UPDATE mabrri\_cmn mc

SET phyident = (

SELECT u.phyident

FROM unit\_lines u

ORDER BY mc.geom2 <-> u.geom

LIMIT 1

);

Explanation:

-----------

Update on mabrri\_cmn (cost=0.00..683.67 rows=832 width=1977)

-> Seq Scan on mabrri\_cmn (cost=0.00..683.67 rows=832 width=1977)

SubPlan 1

-> Limit (cost=0.28..0.61 rows=1 width=46)

-> Index Scan using unit\_lines\_gix on unit\_lines u (cost=0.28..20245.02 rows=61837 width=46)

Order By: (geom <-> mabrri\_cmn.geom2), where <-> means the distance between the arguments

## Convert (long,lat) to projected points

With the ST\_MakePoint function. It takes the longitude, latitude, and altitude and returns a geometry of type point. Example of such a request:

SELECT ST\_MakePoint(longitude,latitude) as geom FROM list\_points

Specify the coordinate system (SRID) of your points with the ST\_SetSRID function. Example (4326):

SELECT ST\_SetSRID(ST\_MakePoint(longitude,latitude),4326) as geom FROM list\_points

Project your data in a specific coordinate system. Example:

SELECT ST\_Transform(ST\_SetSRID(ST\_MakePoint(longitude,latitude),4326),27561) as geom FROM list\_points

## Check your geometry in the classic tuple

To check your geometry in the classic tuple (longitude,latitude), use the ST\_AsText function. For example:

SELECT ST\_AsText(geom) as points FROM list\_geom

Examine output of converted coordinates with specified SRID, as text

SELECT ST\_X(geom) as longitude, ST\_Y(geom) as latitude FROM list\_geom

## Merge data from tables on approximate match

SELECT

h.island,

h.location,

h.beachnumb,

h.x\_utm10n,

h.y\_utm10n,

h.species,

h.habitat,

h.length,

h.zone\_width,

h.beach\_slope,

h.sed\_p,

h.sed\_s1,

h.sed\_s2,

h.sed\_depth,

h.sediment AS "sed\_continuity",

h.neg\_sed,

h.grain\_size,

h.vegetation,

h.oh\_shading,

h.fs\_lu,h.fs\_mod,h.fs\_struct,

h.bs\_lu,h.bs\_mod,h.bs\_struct,

h.anthro,

h.collection,

sq.island AS s\_island,

sq.location\_g,sq.location\_s,

sq.x\_utm10n AS s\_x,

sq.y\_utm10n AS s\_y,

sq.gps\_date,

sq.gps\_time,

sq.longitude,

sq.latitude,

sq.date\_collected,

sq.time\_collected,

sq.field\_surveyor AS surv,

sq.field\_surveyor2 AS surv2,

sq.sediment\_c

FROM habitat\_descriptors AS h

JOIN LATERAL (

SELECT

s.island,

s.location\_g,s,location\_s,

s.x\_utm10n,s.y\_utm10n,

s.gps\_date,s.gps\_time,

s.longitude,s.latitude,

s.date\_collected,s.time\_collected,

s.field\_surveyor,s.field\_surveyor2,

s.sediment\_c

FROM gps\_static AS s

ORDER BY h.geom\_h <-> s.geom\_s LIMIT 1

) AS sq ON TRUE;

## Selection boxes

All perform quickly, with slight differences between && and ST\_Within[ENVELOPE best]

### geog\_geometry && ST\_MakeEnvelope

SELECT distinct on (left(castid,7)) \*,st\_x(geog\_\_geometry) AS long,st\_y(geog\_\_geometry) AS lat from osd\_ctd\_ssea

WHERE geog\_\_geometry && ST\_MakeEnvelope(-126, 49,-122,50,4326) AND depth between 4 and 8 limit 20;

### ST\_Within and ST\_MakeEnvelope

select distinct on (left(castid,7)) \*,st\_x(geog\_\_geometry) AS long,st\_y(geog\_\_geometry) AS lat from osd\_ctd\_ssea

WHERE ST\_Within(geog\_\_geometry, ST\_MakeEnvelope(-126, 49,-122,50,4326)) AND depth between 4 and 8 limit 20;

### ST\_Within and ST\_MakeBox2D

select distinct on (left(castid,7)) \*,st\_x(geog\_\_geometry) AS long,st\_y(geog\_\_geometry) AS lat from osd\_ctd\_ssea WHERE ST\_Within(geog\_\_geometry, ST\_SetSRID(ST\_MakeBox2D(ST\_Point(-126, 49),ST\_Point(-122,50)),4326)) AND depth between 4 and 8 limit 20;

# TABLE JOINS

## Select information with outer left join

SELECT c.mission AS mission,c.datetime,

d.pressure AS P,d.temperature AS T

FROM dfo\_cast\_info AS c JOIN dfo\_ctd\_2017 AS d USING (castid)

GROUP BY mission,datetime,d.pressure,d.temperature limit 20;

## Select points within polygon in Salish Sea (add semi-colon)

SELECT count(\*) from dfo\_cast\_info where

ST\_Within( geog\_\_geometry, ST\_GeomFromText(POLYGON ( ( -125.65 50.15, -124.83 51.25, -121.16 47.87, -122.45 47, -123.29 47, -124.86 48.29, -124.27 49.08, -125.65 50.15 ) ), 4326 ) )

## Join Bottle data and header

\COPY osd\_bottle\_data finalBottle.csv CSV HEADER;

select d.castid,d.depth\_metres AS depth,

d.pressurereversing\_decibar AS "Prev", d.pressure\_decibar AS "P",

d.temperaturereversing\_degC\_ITS90 AS "Trev",d.temperature\_degC\_ITS90 AS "T",

d.salinitybottle\_pss78 AS "Sbott",d.salinity\_pss78 AS "S",

d.oxygendissolved\_umol\_per\_l AS "DO",

h.latitude\_deg AS lat, h.longitude\_deg AS long

from osd\_bottle\_data\_2017 AS d

inner join osd\_bottle\_header\_2017 AS h ON d.castid=h.castid

where left(d.castid,4)>'2000' limit 30;

select d.\*,

h.latitude\_deg AS lat, h.longitude\_deg AS long

from osd\_bottle\_data\_2017 AS d

inner join osd\_bottle\_header\_2017 AS h ON d.castid=h.castid

where left(d.castid,4)>'2000' limit 30;

select d.\*,

h.latitude\_deg AS lat, h.longitude\_deg AS long,

h.geom AS geom

from osd\_bottle\_data\_2017 AS d

inner join osd\_bottle\_header\_2017 AS h ON d.castid=h.castid

WHERE h.latitude\_deg BETWEEN %latmin% AND %latmax%

AND h.longitude\_deg BETWEEN %lonmin% AND %lonmax%

SELECT d.castid, d.depth\_metres AS "D",

d.temperature\_reversing\_degc AS "Trev",d.temperature\_degc AS "T",

d.salinity\_bottle\_psu AS "Sbott",d.salinity\_psu AS "S",

d.oxygen\_dissolved\_umol\_l AS "DO",

h.latitude\_deg AS lat, h.longitude\_deg AS long

FROM osd\_bottle\_data\_2017 AS d

INNER JOIN osd\_bottle\_header\_2017 AS h

ON d.castid=h.castid

WHERE

h.geom && ST\_MakeEnvelope(-125,47,-122,50,4326)

AND \_ST\_Within(geom, ST\_MakeEnvelope(-125,47,-122,50,4326) )

ORDER BY castid limit 10;

SELECT d.\*,

h.latitude\_deg AS lat, h.longitude\_deg AS long

FROM osd\_bottle\_data\_2017 AS d

INNER JOIN osd\_bottle\_header\_2017 AS h

ON d.castid=h.castid

WHERE

h.geom && ST\_MakeEnvelope(-125,47,-122,50,4326)

AND \_ST\_Within(geom, ST\_MakeEnvelope(-125,47,-122,50,4326) )

ORDER BY castid limit 20;

## dfo\_bottle\_data\_2017

SELECT h.datetime, h.latitude\_deg AS lat, h.longitude\_deg AS long,

d.\*, h.project, h.pi AS "PI", h.platform, h.station,

h.instType, h.geom

FROM osd\_bottle\_data\_2017 AS d

INNER JOIN osd\_bottle\_header\_2017 AS h

ON d.castid=h.castid

WHERE

h.geom && ST\_MakeEnvelope(%lonmin%,%latmin%,%lonmax%,%latmax%, 4326)

AND \_ST\_Within(geom, ST\_MakeEnvelope(%lonmin%,%latmin%,%lonmax%,%latmax%, 4326) )

ORDER BY h.castid,depth\_metres

Data from the "bottle" folders in IOS Archive. Primary data is the nutrient data from water samples. Non-bottle CTD data is from the up-cast. The default latmin, lonmin, laatmax and lonmax parameters can be changed to encompass all readings performed by IOS globally - currently set to Vancouver Island. The first data row contains the count of the number of records in that field, as an indicator of fields to download.

Create ctd linestrings with text as output

CREATE TABLE temp\_ctd\_ssea AS (

SELECT profile.castid,ST\_AsText(profile.geom) AS geom,

ST\_AsText(ST\_MakeLine(ST\_MakePoint(profile.depth, profile.pressure) ORDER BY depth)) AS P,

ST\_AsText(ST\_MakeLine(ST\_MakePoint(profile.depth, profile.temperature) ORDER BY depth)) AS T,

ST\_AsText(ST\_MakeLine(ST\_MakePoint(profile.depth, profile.salinity) ORDER BY depth)) AS S,

ST\_AsText(ST\_MakeLine(ST\_MakePoint(profile.depth, profile.sigma\_t) ORDER BY depth)) AS sigma\_t,

ST\_AsText(ST\_MakeLine(ST\_MakePoint(profile.depth, profile.sigma\_stp) ORDER BY depth)) AS sigma\_stp,

ST\_AsText(ST\_MakeLine(ST\_MakePoint(profile.depth, profile.oxygen) ORDER BY depth)) AS oxygen,

ST\_AsText(ST\_MakeLine(ST\_MakePoint(profile.depth, profile.fluor) ORDER BY depth)) AS fluor,

ST\_AsText(ST\_MakeLine(ST\_MakePoint(profile.depth, profile.xmiss) ORDER BY depth)) AS xmiss,

ST\_AsText(ST\_MakeLine(ST\_MakePoint(profile.depth, profile.par) ORDER BY depth)) AS par

FROM osd\_ctd\_ssea AS profile

GROUP BY profile.castid,profile.geom

);

\*\*\* Create ctd linestrings with hex as output

CREATE TABLE temp\_ctd\_ssea AS (

SELECT profile.castid,profile.geom AS geom,

ST\_MakeLine(ST\_MakePoint(profile.depth, profile.pressure) ORDER BY depth) AS P,

ST\_MakeLine(ST\_MakePoint(profile.depth, profile.temperature) ORDER BY depth) AS T,

ST\_MakeLine(ST\_MakePoint(profile.depth, profile.salinity) ORDER BY depth) AS S,

ST\_MakeLine(ST\_MakePoint(profile.depth, profile.sigma\_t) ORDER BY depth) AS sigma\_t,

ST\_MakeLine(ST\_MakePoint(profile.depth, profile.sigma\_stp) ORDER BY depth) AS sigma\_stp,

ST\_MakeLine(ST\_MakePoint(profile.depth, profile.oxygen) ORDER BY depth) AS oxygen,

ST\_MakeLine(ST\_MakePoint(profile.depth, profile.fluor) ORDER BY depth) AS fluor,

ST\_MakeLine(ST\_MakePoint(profile.depth, profile.xmiss) ORDER BY depth) AS xmiss,

ST\_MakeLine(ST\_MakePoint(profile.depth, profile.par) ORDER BY depth) AS par

FROM osd\_ctd\_ssea AS profile

GROUP BY profile.castid,profile.geom

);

CREATE TABLE ios\_bottle\_2017 (

castID varchar(30),

datetime timestamptz,

project varchar(50),

area varchar(50),

mission varchar(10),

PI varchar(30),

platform varchar(30),

station varchar(40),

event varchar(10),

longitude\_deg real,

latitude\_deg real,

waterDepth\_metres varchar(20),

agency varchar(50),

country varchar(20),

dataDesc varchar(60),

instType varchar(50),

instModel varchar(30),

instSN varchar(10),

chans smallint,

recs smallint,

bin\_records smallint,

Sample\_Number varchar(10),

Depth\_metres real [ ],

Pressure\_decibar real [ ],

Temperature\_degC\_ITS90 real [ ],

"Temperature\_Reversing\_deg C (IPTS68)" real [ ],

"Temperature\_BT\_deg C" real [ ],

"Temperature\_Bucket\_deg C (IPTS68)" real [ ],

"Temperature\_DoDraw\_deg C (ITS90)" real [ ],

"Salinity\_Bottle\_PSS-78" real [ ],

Salinity real [ ],

"Oxygen\_Dissolved\_Mass\_umol\_kg" real [ ],

"Oxygen\_Dissolved\_mL\_L" real [ ],

"Oxygen\_Dissolved\_Saturation\_%" real [ ],

"Oxygen\_Isotope\_18\_\_mille" real [ ],

"Transmissivity\_%\_metre" real [ ],

"Turbidity\_Seapoint\_FTU" real [ ],

"Total\_Suspended\_Solids\_mg\_m3" real [ ],

"PAR\_uE\_m^2\_sec" real [ ],

"PAR\_Reference\_uE\_m^2\_sec" real [ ],

pH real [ ],

"Alkalinity\_Total\_CRM\_Corrected\_umol\_kg" real [ ],

"Alkalinity\_Carbonate\_umol\_kg" real [ ],

"Calcium\_Total\_mmol\_kg" real [ ],

"Chlorophyll\_Extracted\_mg\_m^3" real [ ],

"Phaeo-Pigment\_Extracted\_mg\_m^3" real [ ],

"Chlorophyll\_plus\_Phaeo-Pigment\_Extracted\_mg\_m^3" real [ ],

"Chlorophyll\_Extracted\_>0.7um\_mg\_m^3" real [ ],

"Chlorophyll\_Extracted\_>5.0um\_mg\_m^3" real [ ],

"Chlorophyll\_Extracted\_<5.0um\_mg\_m^3" real [ ],

"CDOM\_mg\_m^3" real [ ],

"Fluorescence\_mg\_m^3" real [ ],

"Phosphate\_umol\_L" real [ ],

"Silicate\_umol\_L" real [ ],

"Silicate\_Acidified\_umol\_L" real [ ],

"Ammonium\_umol\_L" real [ ],

"Nitrate\_umol\_L" real [ ],

"Nitrite\_umol\_L" real [ ],

"Nitrate\_plus\_Nitrite\_umol\_L" real [ ],

"Nitrogen\_Particulate\_Total\_umol\_kg" real [ ],

"Nitrogen\_Particulate\_Organic\_mg\_m^3" real [ ],

"Nitrogen\_Dissolved\_Organic\_umol\_L" real [ ],

"Carbon\_Particulate\_Total\_umol\_L" real [ ],

"Carbon\_Particulate\_Organic\_mg\_m^3" real [ ],

"Carbon\_Dissolved\_Organic\_umol\_L" real [ ],

"Carbon\_Dissolved\_Inorganic\_umol\_kg" real [ ],

"Carbon\_Total\_Organic\_umol\_kg" real [ ],

"Carbon\_Isotope\_13\_\_mille" real [ ],

"Carbon\_Isotope\_14\_\_mille" real [ ],

"Production\_Primary\_day\_mgC\_m^3\_day" real [ ],

"Production\_Primary\_hr\_mgC\_m^3\_hour" real [ ],

"Bacteria1\_10^6\_mL" real [ ],

"Bacteria2\_\_mL" real [ ],

"Picophytoplankton\_\_mL" real [ ],

"Nanophytoplankton\_\_mL" real [ ]

);

\COPY ios\_bottle\_2017 FROM F\_\gisData\2017Oct\_osd\osdResult\20171126\_163757\_compressed.csv CSV HEADER;

CREATE TABLE osd\_bottle\_header (

castID varchar(30),

datetime timestamptz,

project varchar(60),

area varchar(60),

mission varchar(10),

PI varchar(30),

platform varchar(30),

station varchar(40),

event varchar(10),

longitude\_deg real,

latitude\_deg real,

waterDepth\_metres varchar(20),

agency varchar(60),

country varchar(20),

dataDesc varchar(60),

instType varchar(50),

instModel varchar(30),

instSN varchar(10),

chans smallint,

recs smallint,

bin\_records varchar(10),

Depth\_metres real [ ],

Pressure\_Reversing\_decibar real [ ],

Pressure\_decibar real [ ],

Temperature\_Reversing\_degC real [ ],

Temperature\_degC real [ ],

Temperature\_BT\_degC real [ ],

Temperature\_Bucket\_degC real [ ],

Temperature\_Draw\_degC real [ ],

Salinity\_Bottle\_PSU real [ ],

Salinity\_PSU real [ ],

Sigma\_t real [ ],

Turbidity\_Seapoint\_FTU real [ ],

Total\_Suspended\_Solids\_mg\_L real [ ],

Transmissivity\_pc\_metre real [ ],

Oxygen\_Dissolved\_umol\_L real [ ],

Oxygen\_Dissolved\_Mass\_umol\_kg real [ ],

Oxygen\_Dissolved\_Saturation\_pc real [ ],

Oxygen\_Isotope\_18\_\_mille real [ ],

PAR\_uE\_m2\_sec real [ ],

PAR\_Reference\_uE\_m2\_sec real [ ],

pH real [ ],

Fluorescence\_mg\_m3 real [ ],

CDOM\_mg\_m3 real [ ],

Chlorophyll\_Extracted\_mg\_m3 real [ ],

Chlorophyll\_Extracted\_lt5um\_mg\_m3 real [ ],

Chlorophyll\_Extracted\_gtpoint7um\_mg\_m3 real [ ],

Chlorophyll\_Extracted\_gt5um\_mg\_m3 real [ ],

Chlorophyll\_Extracted\_point3um\_mg\_m3 real [ ],

Phaeo\_Pigment\_Extracted\_mg\_m3 real [ ],

Phaeo\_Pigment\_Extracted\_point3um\_mg\_m3 real [ ],

Chlorophyll\_plus\_Phaeo\_Pigment\_Extracted\_mg\_m3 real [ ],

Nitrogen\_Dissolved\_Organic\_umol\_L real [ ],

Nitrogen\_Particulate\_Organic\_mg\_m3 real [ ],

Nitrogen\_Particulate\_Total\_umol\_L real [ ],

Nitrate\_umol\_L real [ ],

Nitrite\_umol\_L real [ ],

Nitrate\_plus\_Nitrite\_umol\_L real [ ],

Phosphate\_umol\_L real [ ],

Silicate\_umol\_L real [ ],

Silicate\_Acidified\_umol\_L real [ ],

Ammonium\_umol\_L real [ ],

Alkalinity\_Total\_umol\_kg real [ ],

Alkalinity\_Total\_Colorimetric\_umol\_kg real [ ],

Alkalinity\_Total\_Potentiometric\_umol\_kg real [ ],

Alkalinity\_Carbonate\_umol\_kg real [ ],

Production\_Primary\_mgC\_m3\_day real [ ],

Bacteria\_million\_mL real [ ],

Carbon\_Dissolved\_Inorganic\_umol\_kg real [ ],

Carbon\_Dissolved\_Organic\_umol\_L real [ ],

Carbon\_Particulate\_Organic\_umol\_L real [ ],

Carbon\_Particulate\_Total\_umol\_L real [ ],

Carbon\_Total\_Organic\_umol\_L real [ ],

Carbon\_Isotope\_13\_\_mille real [ ],

Carbon\_Isotope\_14\_\_mille real [ ],

Aluminum\_Dissolved\_pmol\_L real [ ],

Calcium\_Total\_mmol\_kg real [ ],

Iron\_Filtered\_Buffered\_point\_03\_nmol\_L real [ ],

Iron\_Filtered\_Buffered\_point\_1\_nmol\_L real [ ],

Iron\_Filtered\_Buffered\_point\_22\_nmol\_L real [ ],

Iron\_Filtered\_Buffered\_point\_45\_nmol\_L real [ ],

Iron\_Filtered\_Buffered\_200kDalton\_nmol\_L real [ ],

Iron\_Filtered\_StrongAcid\_point\_22\_nmol\_L real [ ],

Iron\_Filtered\_StrongAcid\_Geo\_point\_22\_nmol\_L real [ ],

Iron\_Filtered\_StrongAcid\_MV\_point\_22\_nmol\_L real [ ],

Iron\_Unfiltered\_Buffered\_nmol\_L real [ ],

Iron\_Unfiltered\_Geo\_StrongAcid\_nmol\_L real [ ],

Iron\_Unfiltered\_StrongAcid\_nmol\_L real [ ],

Iron\_Unfiltered\_StrongAcid\_MV\_nmol\_L real [ ],

Sample\_Number varchar(10)

);

CREATE TABLE osd\_bottle\_data\_2017 (

castID varchar(30),

Depth\_metres numeric(10,1),

Pressure\_Reversing\_decibar numeric(10,1),

Pressure\_decibar numeric(10,1),

Temperature\_Reversing\_degC numeric(10,3),

Temperature\_degC numeric(10,3),

Temperature\_BT\_degC numeric(10,3),

Temperature\_Bucket\_degC numeric(10,3),

Temperature\_Draw\_degC numeric(10,3),

Salinity\_Bottle\_PSU numeric(10,3),

Salinity\_PSU numeric(10,3),

Sigma\_t numeric(10,3),

Turbidity\_Seapoint\_FTU numeric(10,1),

Total\_Suspended\_Solids\_mg\_L numeric(10,1),

Transmissivity\_pc\_metre numeric(10,2),

Oxygen\_Dissolved\_umol\_L numeric(10,2),

Oxygen\_Dissolved\_Mass\_umol\_kg numeric(10,1),

Oxygen\_Dissolved\_Saturation\_pc real,

Oxygen\_Isotope\_18\_\_mille real,

PAR\_uE\_m2\_sec numeric(10,3),

PAR\_Reference\_uE\_m2\_sec numeric(10,3),

pH numeric(10,4),

Fluorescence\_mg\_m3 numeric(10,3),

CDOM\_mg\_m3 numeric(10,3),

Chlorophyll\_Extracted\_mg\_m3 numeric(10,3),

Chlorophyll\_Extracted\_lt5um\_mg\_m3 numeric(10,3),

Chlorophyll\_Extracted\_gtpoint7um\_mg\_m3 numeric(10,3),

Chlorophyll\_Extracted\_gt5um\_mg\_m3 numeric(10,3),

Chlorophyll\_Extracted\_point3um\_mg\_m3 numeric(10,3),

Phaeo\_Pigment\_Extracted\_mg\_m3 numeric(10,3),

Phaeo\_Pigment\_Extracted\_point3um\_mg\_m3 numeric(10,3),

Chlorophyll\_plus\_Phaeo\_Pigment\_Extracted\_mg\_m3 numeric(10,3),

Nitrogen\_Dissolved\_Organic\_umol\_L numeric(10,3),

Nitrogen\_Particulate\_Organic\_mg\_m3 numeric(10,3),

Nitrogen\_Particulate\_Total\_umol\_L numeric(10,3),

Nitrate\_umol\_L numeric(10,3),

Nitrite\_umol\_L numeric(10,3),

Nitrate\_plus\_Nitrite\_umol\_L numeric(10,3),

Phosphate\_umol\_L numeric(10,3),

Silicate\_umol\_L numeric(10,3),

Silicate\_Acidified\_umol\_L numeric(10,3),

Ammonium\_umol\_L numeric(10,3),

Alkalinity\_Total\_umol\_kg numeric(10,3),

Alkalinity\_Total\_Colorimetric\_umol\_kg numeric(10,3),

Alkalinity\_Total\_Potentiometric\_umol\_kg numeric(10,3),

Alkalinity\_Carbonate\_umol\_kg numeric(10,3),

Production\_Primary\_mgC\_m3\_day numeric(10,3),

Bacteria\_million\_mL numeric(10,3),

Carbon\_Dissolved\_Inorganic\_umol\_kg numeric(10,3),

Carbon\_Dissolved\_Organic\_umol\_L numeric(10,3),

Carbon\_Particulate\_Organic\_umol\_L numeric(10,3),

Carbon\_Particulate\_Total\_umol\_L numeric(10,3),

Carbon\_Total\_Organic\_umol\_L numeric(10,3),

Carbon\_Isotope\_13\_\_mille numeric(10,3),

Carbon\_Isotope\_14\_\_mille numeric(10,3),

PhytoplanktonVolume\_mm3\_per\_m3 numeric(10,3),

Nanophytoplankton\_per\_mL numeric(10,3),

Picophytoplankton\_per\_mL numeric(10,3),

Aluminum\_Dissolved\_pmol\_L numeric(10,3),

Calcium\_Total\_mmol\_kg numeric(10,3),

Iron\_Filtered\_Buffered\_point\_03\_nmol\_L numeric(10,3),

Iron\_Filtered\_Buffered\_point\_1\_nmol\_L numeric(10,3),

Iron\_Filtered\_Buffered\_point\_22\_nmol\_L numeric(10,3),

Iron\_Filtered\_Buffered\_point\_45\_nmol\_L numeric(10,3),

Iron\_Filtered\_Buffered\_200kDalton\_nmol\_L numeric(10,3),

Iron\_Filtered\_StrongAcid\_point\_22\_nmol\_L numeric(10,3),

Iron\_Filtered\_StrongAcid\_Geo\_point\_22\_nmol\_L numeric(10,3),

Iron\_Filtered\_StrongAcid\_MV\_point\_22\_nmol\_L numeric(10,3),

Iron\_Unfiltered\_Buffered\_nmol\_L numeric(10,3),

Iron\_Unfiltered\_Geo\_StrongAcid\_nmol\_L numeric(10,3),

Iron\_Unfiltered\_StrongAcid\_nmol\_L numeric(10,3),

Iron\_Unfiltered\_StrongAcid\_MV\_nmol\_L numeric(10,3),

Sample\_Number varchar(10)

);

\*\*\*

## Add a geometry column

* add a geometry column,
* add (lat, long) points from the table,
* add index\_

SELECT AddGeometryColumn ('data','osd\_bottle\_2017','geom',4326,'POINT',2);

UPDATE table SET geom = ST\_SetSRID(ST\_MakePoint(long, lat), 4326);

# LINESTRING AND POLYGON OPERATIONS

## Forming linestring from points

-- If you are using PostgreSQL 9.0+

-- (you can use the new ORDER BY support for aggregates)

-- this is a guaranteed way to get a correctly ordered linestring

-- Your order by part can order by more than one column if needed

SELECT gps.gps\_track, ST\_MakeLine(gps.the\_geom ORDER BY gps\_time) As newgeom

FROM gps\_points As gps

GROUP BY gps.gps\_track;

## Commands to create gps point table and then linestrings

### Points from GPS to lines 1

drop table gps\_dynamic ;

create table gps\_dynamic(

Island varchar,

LineNo integer,

Point\_ID varchar,

GPS\_Date date,

GPS\_Time time,

Latitude numeric(12,5),

Longitude numeric(12,5),

GNSS\_Heigh varchar,

X\_UTM10N numeric(10,1),

Y\_UTM10N numeric(10,1),

Max\_PDOP varchar,

Max\_HDOP varchar,

Datafile varchar,

Corr\_Type varchar,

Rcvr\_Type varchar,

Update\_Sta varchar,

Feat\_Name varchar,

Unfilt\_Pos varchar,

Filt\_Pos varchar,

Vert\_Prec varchar,

Horz\_Prec varchar,

dist real,

deldist real,

deltime real,

LineFlag numeric(5,0)

);

\COPY gps\_dynamic from './20190914\_dynamic.csv' DELIMITER ',' CSV HEADER;

alter table gps\_dynamic add column gid serial primary key;

alter table gps\_dynamic add column geom\_d geometry(Point,26910);

update gps\_dynamic set geom\_d = ST\_SetSRID(ST\_MakePoint(x\_utm10n,y\_utm10n),26910);

create index dyn\_gix ON gps\_dynamic USING GIST(geom\_d);

### Points from GPS to lines 2

CREATE TABLE gps\_tracks AS SELECT gps.island AS island,gps.lineno,min(gps.gps\_date) AS date\_start, min(gps.gps\_time) AS time\_start, max(gps.gps\_date) AS date\_end, max(gps.gps\_time) AS time\_end,gps.datafile,ST\_MakeLine(gps.geom\_d ORDER BY gps\_time) As geom2

FROM gps\_dynamic As gps

GROUP BY (gps.island,gps.lineno,gps.gps\_date,gps.datafile);

delete from gps\_tracks where time\_start=time\_end;

alter table gps\_tracks add column geom geometry(LineString, 26910);

update gps\_tracks set geom = ST\_Transform(geom2, 26910);

alter table gps\_tracks drop column geom2;

\d gps\_tracks

select island,lineno,date\_start AS date,time\_start,time\_end,(time\_end-time\_start) AS del\_t,datafile,left(ST\_AsText(geom),50)

FROM gps\_tracks

Order BY lineno;

alter table gps\_tracks add column gid serial primary key;

create index gps\_tracks\_gix on gps\_tracks using gist(geom);

\d gps\_tracks

### Create table for Sidney and James Islands

create table isle\_trust\_sidney\_james (

OBJECTID varchar,

Species varchar,

BeachNumb varchar,

Island varchar,

Location varchar,

Sample\_Date date,

Sample\_Time time,

WP\_POSITION varchar,

UTM\_Zone varchar,

y\_utm10n numeric(10,1),

x\_utm10n numeric(10,1),

GPS\_Model varchar,

Habitat varchar,

Sediment varchar,

Anthro varchar,

Neg\_Sed varchar,

Sed\_P varchar,

Sed\_S1 varchar,

Sed\_S2 varchar,

OH\_Shading varchar,

Vegetation varchar,

FS\_Mod varchar,

FS\_Struct varchar,

FS\_LU varchar,

BS\_Mod varchar,

BS\_Struct varchar,

BS\_LU varchar,

Beach\_Slope varchar,

Zone\_Width numeric(10,1),

Beach\_Length numeric(10,1),

Sed\_Depth varchar,

Tide\_Height varchar,

Description varchar,

SHAPE\_Length varchar,

geom geometry(Linestring, 26910)

);

\COPY isle\_trust\_sidney\_james from './sidney\_james\_wkt\_geom\_26910.csv' DELIMITER ',' CSV HEADER;

alter table isle\_trust\_sidney\_james add column gid serial primary key;

alter table isle\_trust\_sidney\_james add column geom\_pt geometry(Point,26910);

update isle\_trust\_sidney\_james set geom\_pt =ST\_SetSRID(ST\_MakePoint(x\_utm10n,y\_utm10n),26910);

### Create database table with line information

CREATE TABLE dfo\_missions AS

SELECT mission, min(datetime) AS start, max(datetime) AS finish,

project, investigator AS pi, platform,

ST\_MakeLine((geog\_\_geometry) ORDER BY datetime) As shiptrack

FROM dfo\_cast\_info GROUP BY mission,project,pi,platform;

### Create ship tracks

CREATE TABLE nodc\_ship\_tracks AS

SELECT nodcid, accessionNo, country, institute, cruiseid AS orig cruise ID,

project, investigator AS PI, platform, min(datetime) AS start, max(datetime) AS finish,

instrument, ST\_MakeLine(the\_geog\_\_geometry ORDER BY datetime) As track

FROM nodc\_ctd\_info

GROUP BY nodcid,cruiseid,project,PI,platform,instrument,accessionNo,country,institute;

### Midpoint on a MultiLineString

select organization,

ST\_X(ST\_Transform(ST\_LineInterpolatePoint(ST\_LineMerge(geom),0.5),4326)) AS "Long",

ST\_Y(ST\_Transform(ST\_LineInterpolatePoint(ST\_LineMerge(geom),0.5),4326)) AS "Lat",

ST\_XMIN(geom),ST\_YMIN(geom),sed\_continuity from ff\_it limit 100;

### PostGIS Query for a Point Within a Polygon

CREATE or replace function in\_neighborhood(coords varchar(255))

returns varchar(255)

as $$

declare

myid varchar(255);

begin

SELECT id FROM zones

WHERE ST\_Contains(zones.geometry, ST\_Transform(

ST\_GeomFromText(concat(POINT(, coords, )), 4326), 4326)

)

into myid;

return myid;

end;

$$

language plpgsql;

### Find point in one table nearest to polygon in another table

PostGIS - For each polygon/linestring/point in table A, find nearest point in table B and update table A with value

UPDATE tableA tA1

SET pc = (SELECT near\_point.pc

FROM tableA tA2, LATERAL

(SELECT pc

FROM tableB

ORDER BY tableB.geom <-> tA2.geom

LIMIT 1) near\_point

WHERE tA2.id = tA1.id)

UPDATE table

SET column1 = value1,

column2 = value2 ,...

WHERE

condition;

UPDATE dfo\_cast\_info SET investigator=value1 WHERE condition;

## Updating fields in specific rows with Geometry text

shorezone=> update ff\_spawning\_beach\_monitoring

shorezone-> set geom=ST\_TRANSFORM(ST\_GeomFromEWKT('SRID=4326;MULTILINESTRING((-124.198203173111 49.2964658229472,-124.198282979567 49.2963568746907,-124.198283741357 49.2962488586613,-124.198338085901 49.2961763414833,-124.198114217154 49.2958820011725,-124.198002368637 49.29562215362,-124.198001085456 49.2954783818015,-124.198058873572 49.2954056340535,-124.198170994402 49.2952606479666,-124.198619254441 49.2950399758336,-124.198675663273 49.2949584813958,-124.198843377444 49.2949297784894,-124.199069534956 49.2948556152432,-124.199180902105 49.2948183743756,-124.199461729513 49.2947074285786,-124.199909539913 49.2946307732101,-124.200020895371 49.2945938013618,-124.20047007067 49.2945261778479,-124.200583152644 49.2944889551128,-124.200806192267 49.2944507549769,-124.201200415059 49.2943385964874,-124.201424528541 49.2942283936728,-124.201535892642 49.2941911413462,-124.201760004614 49.2940809378113,-124.202210595402 49.2939322999747,-124.20237862541 49.2939395975428,-124.202603055723 49.2938654072241,-124.203051185515 49.2938244748507,-124.203163180172 49.2938595387274,-124.203997546921 49.2938236130465))'),26910)

WHERE phyident='01/10/0099/00';

# WATER PROPERTY TABLE from tables of header and cast data

Water property casts via a profiling CTD always include castid, datetime, longitude, latitude, depth, pressure and temperature. It almost always includes salinity, sigma-t and sigma-Stp (density anomaly). It often includes oxygen and fluorescence. It may include PAR.

CREATE TABLE temp\_ctd\_2017 AS

SELECT c.castID AS castid,depth,pressure,temperature,salinity,sigma\_t,sigma\_stp,oxygen,fluor,xmiss\_m AS xmiss,par,geog from dfo\_cast\_info AS c INNER JOIN ctd\_2017 USING (castid)

PRIMARY KEY (castid, depth)

);

## Create table cascade for osd ctd data

### CREATE TABLE osd\_ctd\_southern

(LIKE temp\_ctd\_2017 INCLUDING DEFAULTS INCLUDING CONSTRAINTS);

ALTER TABLE osd\_ctd\_southern ADD CONSTRAINT ocean\_southern

CHECK (ST\_Y(geog\_\_geometry) <-45);

alter table osd\_ctd\_southern add column fid serial PRIMARY KEY

### CREATE TABLE osd\_ctd\_arctic

(LIKE temp\_ctd\_2017 INCLUDING DEFAULTS INCLUDING CONSTRAINTS);

ALTER TABLE osd\_ctd\_arctic ADD CONSTRAINT ocean\_arctic

CHECK (ST\_Y(geog\_\_geometry) >65 );

### CREATE TABLE osd\_ctd\_atlantic

(LIKE temp\_ctd\_2017 INCLUDING DEFAULTS INCLUDING CONSTRAINTS);

ALTER TABLE osd\_ctd\_atlantic ADD CONSTRAINT ocean\_atlantic

CHECK ( (ST\_Y(geog\_\_geometry) BETWEEN -45 AND +65) AND

(ST\_X(geog\_\_geometry) between -80.001 and +21) );

### CREATE TABLE osd\_ctd\_indian

(LIKE temp\_ctd\_2017 INCLUDING DEFAULTS INCLUDING CONSTRAINTS);

ALTER TABLE osd\_ctd\_indian ADD CONSTRAINT ocean\_indian

CHECK ( (ST\_Y(geog\_\_geometry) BETWEEN -45 AND +65) AND

(ST\_X(geog\_\_geometry) between 21.001 and 100) );

### CREATE TABLE osd\_ctd\_chinasea

(LIKE temp\_ctd\_2017 INCLUDING DEFAULTS INCLUDING CONSTRAINTS);

ALTER TABLE osd\_ctd\_chinasea ADD CONSTRAINT ocean\_chinasea

CHECK ( (ST\_Y(geog\_\_geometry) BETWEEN -45 AND +65) AND

(ST\_X(geog\_\_geometry) between 122.001 and 125) );

### CREATE TABLE osd\_ctd\_pac\_west

(LIKE osd\_ctd\_2017 INCLUDING DEFAULTS INCLUDING CONSTRAINTS);

ALTER TABLE osd\_ctd\_pac\_west ADD CONSTRAINT ocean\_pacw

CHECK ( (ST\_Y(geog\_\_geometry) BETWEEN -45 AND +65) AND (ST\_X(geog\_\_geometry) BETWEEN 125.001 and 180) );

### CREATE TABLE osd\_ctd\_pac\_noreast

(LIKE osd\_ctd\_2017 INCLUDING DEFAULTS INCLUDING CONSTRAINTS);

ALTER TABLE osd\_ctd\_pac\_noreast ADD CONSTRAINT ocean\_pacne

CHECK ( (ST\_Y(geog\_\_geometry) BETWEEN 51.5 AND +65) AND (ST\_X(geog\_\_geometry) BETWEEN -180 and -80) );

### CREATE TABLE osd\_ctd\_pac\_soueast

(LIKE osd\_ctd\_2017 INCLUDING DEFAULTS INCLUDING CONSTRAINTS);

ALTER TABLE osd\_ctd\_pac\_soueast ADD CONSTRAINT ocean\_pacse

CHECK ( (ST\_Y(geog\_\_geometry) BETWEEN -45 AND +51.499999) AND (ST\_X(geog\_\_geometry) BETWEEN -180 and -80) AND NOT ST\_Within( geog\_\_geometry, ST\_GeomFromText(POLYGON ( ( -125.65 50.15, -124.83 51.25, -121.16 47.87, -122.45 47, -123.29 47, -124.86 48.29, -124.27 49.08, -125.65 50.15 ) ), 4326 ) ) );

### CREATE TABLE osd\_ctd\_ssea

(LIKE temp\_ctd\_2017 INCLUDING DEFAULTS INCLUDING CONSTRAINTS);

ALTER TABLE osd\_ctd\_ssea ADD CONSTRAINT ocean\_ssea

CHECK ( ST\_Within( geog\_\_geometry, ST\_GeomFromText(POLYGON ( ( -125.65 50.15, -124.83 51.25, -121.16 47.87, -122.45 47, -123.29 47, -124.86 48.29, -124.27 49.08, -125.65 50.15 ) ), 4326 ) ) );

INSERT INTO osd\_ctd\_pac\_west(castid,depth,pressure,temperature,salinity,sigma\_t,sigma\_stp,oxygen,fluor,xmiss,par,geog)

SELECT castid,depth,pressure,temperature,salinity,sigma\_t,sigma\_stp,oxygen,fluor,xmiss,par,geog

FROM osd\_ctd\_pacific

WHERE ( (ST\_Y(geog\_\_geometry) BETWEEN -45 AND +65) AND (ST\_X(geog\_\_geometry) BETWEEN 125.001 and 180) );

### Rows in partitioned osd\_ctd\_2017 tables

osd\_ctd\_2017 = 6 744 812

osd\_ctd\_arctic = 3 932 330

osd\_ctd\_atlantic = 23 434

osd\_ctd\_chinasea = 0

osd\_ctd\_indian = 0

osd\_ctd\_pac\_noreast = 3 436 687

osd\_ctd\_pac\_soueast = 16 834 965

osd\_ctd\_pac\_west = 1 876 355

osd\_ctd\_southern = 83 641

osd\_ctd\_ssea = 2 705 407

osd\_ctd\_pacific = 22 148 007

# SHOREZONE

ShoreZone is an aerial imaging, coastal habitat classification and mapping system used to inventory alongshore and across-shore geomorphological and biological attributes of the shoreline. The georeferenced, oblique, low altitude aerial imagery is acquired during the lowest tides of the year and then used to classify habitat attributes into a searchable database. This data is used for coastal planning, identification of vulnerable resources, oil spill response planning, habitat modeling, recreational planning and scientific research.

Each Shorezone bioband is an assemblage of coastal biota, and are visible in aerial imagery as patterns of colour and texture across the shoreline. Biobands are generally associated with characteristic wave energies, substrate conditions and across-shore elevations, and are often seen as spatially distinct alongshore and across-shore patterns. Each bioband is named for the dominant species or group.

## Add shorezone table

CREATE TABLE shorezone\_track\_sample (

FID integer,

LAT real,

LON real,

DATE\_UTC date,

TIME\_UTC time,

DATETIME timestamp,

TAPE\_NO varchar,

PLAY\_VIDEO varchar,

PHOTONAME varchar,

VIEW\_PHOTO varchar,

DISPLAY varchar );

## Create Shorezone track from a series of gps points

\COPY shorezone\_track\_sample FROM 'F:\gisData\2017\_11\_17 CORI\_BcShoreZone\trackline\bc15\_sh\_ShoreZone\_track.csv' CSV HEADER;

This example takes a sequence of GPS points and creates one record for each gps travel where the geometry field is a line string composed of the gps points in the order of the travel.

-- If you are using PostgreSQL 9.0+

-- (you can use the new ORDER BY support for aggregates)

-- this is a guaranteed way to get a correctly ordered linestring

-- Your order by part can order by more than one column if needed

SELECT gps.gps\_track, ST\_MakeLine(gps.the\_geom ORDER BY gps\_time) As newgeom

FROM gps\_points As gps

GROUP BY gps.gps\_track;

SELECT ST\_MakeLine(gps.geom ORDER BY datetime) As newgeom

FROM shorezone\_track\_sample As gps

GROUP BY gps.track;

CREATE TABLE cori\_tracks AS

SELECT gps.tape\_no, min(datetime) AS start, max(datetime) AS finish,

ST\_MakeLine(gps.geom ORDER BY gps.datetime) As trackgeom

FROM shorezone\_track\_sample As gps

GROUP BY gps.tape\_no;

SELECT l.unit\_id,b.\*,l.geom

FROM unit\_lines AS l

INNER JOIN bioband AS b

ON l.phyident=b.phyident

# PostgreSQL getting daily, weekly, and monthly averages of the occurrences of an event in one query

(https://stackoverflow.com/questions/38226788/postgresql-getting-daily-weekly-and-monthly-averages-of-the-occurrences-of-an/38227758)

CURRENTLY: I have this rather large query that works by

\* Aggregating the daily, weekly, monthly counts into intermediate tables by taking the count() of an event grouped by the event name and the date.

\* Selecting the avg count over each intermediate table by doing avg() group by just event, doing a union of the results,

and because I want to have a separate column for daily, weekly, monthly, putting a filler value of 0 into empty columns.

\* I then sum over all the columns, and the 0s basically act as a no-op, which gives me just a single value for each event.

ANSWER: In 9.5+ use grouping sets

The data selected by the FROM and WHERE clauses is grouped separately by each specified grouping set, aggregates computed for each group just as for simple GROUP BY clauses, and then the results returned

select event,

avg(total) filter (where day is not null) as avg\_day,

avg(total) filter (where week is not null) as avg\_week,

avg(total) filter (where month is not null) as avg\_month

from (

select

event,

date\_trunc('day', created\_at) as day,

date\_trunc('week', created\_at) as week,

date\_trunc('month', created\_at) as month,

count(\*) as total

from tracking\_stuff

where event in ('thing','thing2','thing3')

group by grouping sets ((event, 2), (event, 3), (event, 4))

) s

group by event

## Select using “distinct on” and “extract”

### Daily averages

select distinct on (extract(day from date)) s.stnid,s.station,

longitude AS "lon",latitude AS "lat",date,temp,temp\_sd,press,p\_sd,

dew\_pt,"RH",wind\_min,wind\_max,avg AS wind\_avg,stddev AS wind\_sd,vis

from weather\_daily w inner join weather\_stations s on (w.stnid=s.stnid)

order by lat desc

limit 100;

### Weekly averages

select distinct ON (extract(week from to\_date(dt,'yyyy-mm-dd')) )

stnid,station,

to\_date(dt,'yyyy-mm-dd') AS "date",

min(wind) AS "wind\_min",max(wind) AS "wind\_max",

SQRT(POWER(AVG(-wind\*COS(RADIANS(wind\_dir\_10s \* 10 ) ) ),2 ) +

POWER(AVG(-wind\*SIN(RADIANS(wind\_dir\_10s \* 10 ) ) ),2 ) )::numeric(7,1) AS "wind\_avg",

90 - DEGREES(ATAN2(AVG(-wind\*COS(RADIANS(wind\_dir\_10s \* 10 ))),

AVG(-wind\*SIN(RADIANS(wind\_dir\_10s \* 10 )))) )::numeric(7,1) AS "wind\_to\_dir",

AVG(-wind\*SIN(RADIANS(wind\_dir\_10s \* 10 ) ) )::numeric(7,1) AS "wnd\_eward",

AVG(-wind\*COS(RADIANS(wind\_dir\_10s \* 10 ) ) )::numeric(7,1) AS "wnd\_nthward"

from weather\_hourly

group by stnid, station,to\_date(dt,'yyyy-mm-dd')

limit 100;

### Daily averages

CREATE TABLE weather\_daily AS

select stnid,station,

to\_date(dt,'yyyy-mm-dd') AS "date",

AVG(temp)::numeric(7,2) AS "temp",stddev(temp)::numeric(5,1) AS "temp\_sd",

AVG(dew\_point)::numeric(5,1) AS "dew\_pt", AVG(rel\_hum\_pc)::numeric(5,1) AS "rh",

AVG(press)::numeric(5,1) AS "press",stddev(press)::numeric(5,1) AS "p\_sd",

AVG(visibility)::numeric(7,1) AS "vis",

min(wind) AS "wind\_min",max(wind) AS "wind\_max",

SQRT(POWER(AVG(-wind\*COS(RADIANS(wind\_dir\_10s \* 10 ) ) ),2 ) +

POWER(AVG(-wind\*SIN(RADIANS(wind\_dir\_10s \* 10 ) ) ),2 ) )::numeric(7,1) AS "wind\_avg",

90 - DEGREES(ATAN2(AVG(-wind\*COS(RADIANS(wind\_dir\_10s \* 10 ))),

AVG(-wind\*SIN(RADIANS(wind\_dir\_10s \* 10 )))) )::numeric(7,1) AS "wind\_to\_dir",

AVG(-wind\*SIN(RADIANS(wind\_dir\_10s \* 10 ) ) )::numeric(7,1) AS "wnd\_eward",

AVG(-wind\*COS(RADIANS(wind\_dir\_10s \* 10 ) ) )::numeric(7,1) AS "wnd\_nthward"

from weather\_hourly

group by stnid, station,to\_date(dt,'yyyy-mm-dd')

### Daily averages

select \*

from weather\_stations s, weather\_daily d

where d.stnid=s.stnid

limit 300;

### Weekly averages from joined tables

select distinct on (extract(week from date) )

station, longitude AS "lon", latitude AS "lat", elevation\_m AS "ht",

s.stnid, date,temp, dew\_pt, rh, press,

vis, wind\_min, wind\_max, wind\_avg,

wind\_to\_dir, wnd\_eward, wnd\_nthward

from marine\_weather\_stations s, weather\_daily d

where d.stnid=s.stnid;

### Weekly averages from joined tables

select s.stnid, station, longitude AS "lon", latitude AS "lat", elevation\_m AS "ht",

stnid, datetime, epoch, temp, dew\_point, rel\_hum\_pc, wind\_dir\_10s, wind,

visibility, press, humidity, wind\_chill, weather, geom

from marine\_weather\_stations s, weather\_hourly h

where d.stnid=s.stnid;

# CITIZEN SCIENCE TABLES

## Drop existing table and create new

DROP TABLE IF EXISTS <tablename>;CREATE TABLE citsci\_stations (

gid integer,

Line varchar(12),

crew varchar(10),

datetime timestamptz,

epoch integer,

station varchar(12),

long numeric(10,3),

lat numeric(10,3),

x\_albers numeric(10,1),

y\_albers numeric(10,1),

dist\_kms numeric(10,3),

tdist\_mins numeric(10,3),

Log\_Man varchar(12),

depth\_min numeric(7,1),

depth\_max numeric(7,1),

date date,

time time,

castid varchar(25)

);

## Load table from csv, add geometry column from lat/long

\COPY citsci\_stations ./CitSciCastInfo\_20180417.csv CSV HEADER;

SELECT AddGeometryColumn ('data','citsci\_stations','geom',4326,'POINT',2);

UPDATE table SET geom = ST\_SetSRID(ST\_MakePoint(long, lat), 4326);

# WEATHER TABLES

Name climateID longitude latitude timestamp Temp\_degC Temp\_Flag Dew\_Point\_degC Dew\_Point\_Flag Rel\_Hum\_pc Rel\_Hum\_Flag Wind\_Dir\_10s\_deg Wind\_Dir\_Flag Wind\_Spd\_kmh Wind\_Spd\_Flag Visibility\_km Visibility\_Flag Press\_kPa Press\_Flag Hmdx Hmdx\_Flag Wind\_Chill Wind\_Chill\_Flag Weather

## DROP TABLE IF EXISTS, Create table

DROP TABLE IF EXISTS weather\_hourly;

CREATE TABLE weather\_hourly (

Name varchar(30),

climateID varchar(15),

longitude numeric(7,2),

latitude numeric(7,2),

timestamp timestamp with time zone,

Temp\_degC numeric(5,1),

Temp\_Flag varchar(5),

Dew\_Point\_degC numeric(5,1),

Dew\_Point\_Flag varchar(5),

Rel\_Hum\_pc numeric(5,1),

Rel\_Hum\_Flag varchar(5),

Wind\_Dir\_10s\_deg smallint,

Wind\_Dir\_Flag varchar(5),

Wind\_Spd\_kmh smallint,

Wind\_Spd\_Flag varchar(5),

Visibility\_km numeric(5,1),

Visibility\_Flag varchar(5),

Press\_kPa numeric(5,1),

Press\_Flag varchar(5),

Hmdx smallint,

Hmdx\_Flag varchar(5),

Wind\_Chill numeric(5,1),

Wind\_Chill\_Flag varchar(5),

Weather varchar(100)

);

## DROP TABLE IF EXISTS, Create table

DROP TABLE IF EXISTS weather\_daily2;

CREATE TABLE weather\_daily2 AS

select climateid,name,longitude,latitude,

timestamp::date AS "date",

AVG(temp\_degC)::numeric(7,2) AS "temp",stddev(temp\_degC)::numeric(5,1) AS "temp\_sd",

AVG(dew\_point\_degC)::numeric(5,1) AS "dew\_pt", AVG(rel\_hum\_pc)::numeric(5,1) AS "rh",

AVG(press\_kpa)::numeric(5,1) AS "press",stddev(press\_kpa)::numeric(5,1) AS "p\_sd",

AVG(visibility\_km)::numeric(7,1) AS "vis",

min(wind\_spd\_kmh) AS "wind\_min",max(wind\_spd\_kmh) AS "wind\_max",

SQRT(POWER(AVG(-wind\_spd\_kmh\*COS(RADIANS(wind\_dir\_10s\_deg \* 10 ) ) ),2 ) +

POWER(AVG(-wind\_spd\_kmh\*SIN(RADIANS(wind\_dir\_10s\_deg \* 10 ) ) ),2 ) )::numeric(7,1) AS "wind\_avg",

90 - DEGREES(ATAN2(AVG(-wind\_spd\_kmh\*COS(RADIANS(wind\_dir\_10s\_deg \* 10 ))),

AVG(-wind\_spd\_kmh\*SIN(RADIANS(wind\_dir\_10s\_deg \* 10 )))) )::numeric(7,1) AS "wind\_to\_dir",

AVG(-wind\_spd\_kmh\*SIN(RADIANS(wind\_dir\_10s\_deg \* 10 ) ) )::numeric(7,1) AS "wnd\_eward",

AVG(-wind\_spd\_kmh\*COS(RADIANS(wind\_dir\_10s\_deg \* 10 ) ) )::numeric(7,1) AS "wnd\_nthward"

from weather\_hourly2

group by climateid,name,longitude,latitude,date;

# FORAGE FISH TABLES

## Create forage fish table

CREATE TABLE data.foragefish\_rll\_it\_cmn\_pss (

station character varying,

lat numeric(12,5),

long numeric(12,5),

observation\_date date,

sampling\_time time without time zone,

species\_sampled character varying,

site\_sample\_number character varying,

sample\_type character varying,

sed\_primary character varying,

sed\_sec\_1 character varying,

sed\_sec\_2 character varying,

shading character varying,

vegetation character varying,

foreshore\_mod character varying,

foreshore\_structure character varying,

foreshore\_landuse character varying,

backshore\_mod character varying,

backshore\_structure character varying,

backshore\_landuse character varying,

tidal\_land\_mark character varying,

distance\_from\_landmark numeric(10,3),

beach\_slope character varying,

calculated\_tidal\_elevation numeric(10,3),

width\_of\_zone character varying,

length\_avail\_habitat character varying,

sample\_volume character varying,

spawn\_density character varying,

spawn\_present character varying,

comment\_general character varying,

comment\_sediment character varying,

time\_start time without time zone,

time\_end time without time zone,

landmass character varying,

municipality character varying,

location character varying,

gps\_unit character varying,

weather\_conditions character varying,

air\_temp numeric(10,1),

beach\_aspect character varying,

compass\_direction\_beach\_faces character varying,

reference\_tide\_height character varying,

last\_high\_tide\_height character varying,

tide\_height\_at\_time\_start numeric(10,3),

exposure character varying,

fetch1 character varying,

phyident character varying,

transect character varying,

sed\_date\_processed date,

sediment\_appearance character varying,

number\_eggs character varying,

comment\_eggs character varying,

organization character varying,

primary\_field\_surveyor character varying,

sed\_processor character varying,

data\_entry character varying,

objectid character varying,

sed\_continuity character varying,

grain\_size character varying,

sed\_depth character varying,

neg\_sed character varying,

habitat character varying,

anthro character varying,

geom2 public.geometry(MultiLineString,3005),

gid integer NOT NULL

);