How to SQL

Part 2

Jeremy Goldstein: Minuteman Library Network

Ray Voelker: The Public Library of Cincinnati and Hamilton County

Tuesday, March 23 – Thursday, March 25

Meet Jeremy and Ray...and Rufus and Audrey

Agenda

Part 1

- SQL Basics
  - Relational Databases
  - Joins
  - SQL Overview
- Statements
  - SELECT
  - WHERE
  - GROUP BY / HAVING
- Subqueries
- CASE

Part 2

- Sierra_view schema
  - Previewing data
- Data Types
  - Casting
- Functions
  - Aggregate/Filter
  - String Functions
  - Window Functions
- Combining Queries
  - EXISTS
  - INTERSECT/EXCEPT/UNION

Sierra_View

Overview

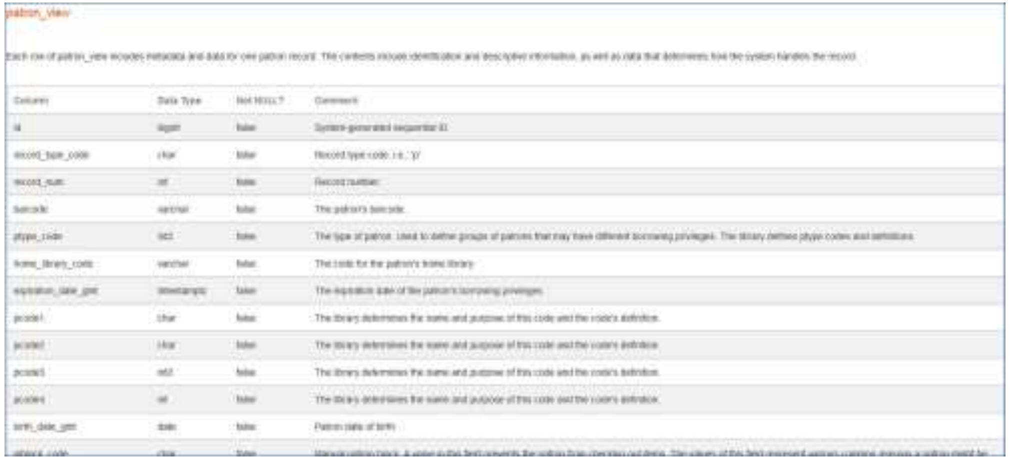
- Views Not Tables
- 360 tables
  - 2780 columns
- How to reference a field
  - schema.table.column
  - sierra_view.item_record.id
    - Only other schema you can access is pg_catalog

Sierra DNA

Sierra DNA (Database NAvigator)

http://techdocs.iii.com/sierradna/

Uses the same username & password as Supportal / CSDirect



patron_view

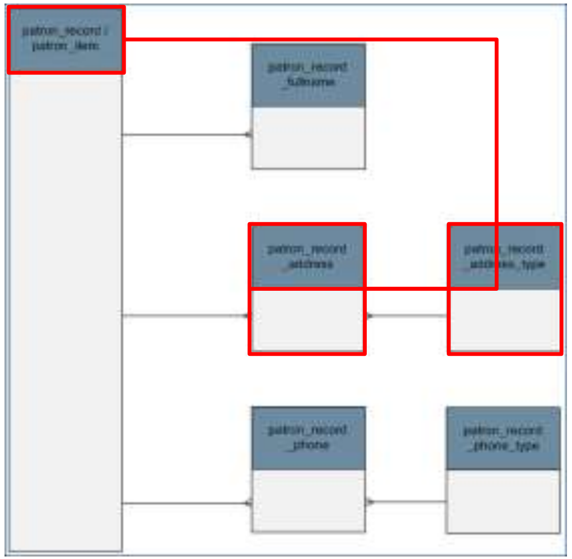
Each row of patron_view includes metadata and data for one patron record. The contents include identification and descriptive information, as well as data that determines how the system handles the record.

Column	Data Type	Not NULL?	Comment
id	bigint	false	System-generated sequential ID
record_type_code	char	false	Record type code, i.e., 'p'
record_num	int	false	Record number
barcode	varchar	false	The patron's barcode
ptype_code	int2	false	The type of patron. Used to define groups of patrons that may have different borrowing privileges. The library defines ptype codes and definitions.
home_library_code	varchar	false	The code for the patron's home library
expiration_date_gmt	timestamptz	false	The expiration date of the patron's borrowing privileges
pcode1	char	false	The library determines the name and purpose of this code and the code's definition.
pcode2	char	false	The library determines the name and purpose of this code and the code's definition.
pcode3	int2	false	The library determines the name and purpose of this code and the code's definition.
pcode4	int	false	The library determines the name and purpose of this code and the code's definition.
birth_date_gmt	date	false	Patron date of birth
mblock_code	char	false	Manual patron block. A value in this field prevents the patron from checking out items. The values of this field represent various reasons a patron might be [illegible]

The Sierra DNA describes all the SQL tables and the columns in those tables.

Documentation

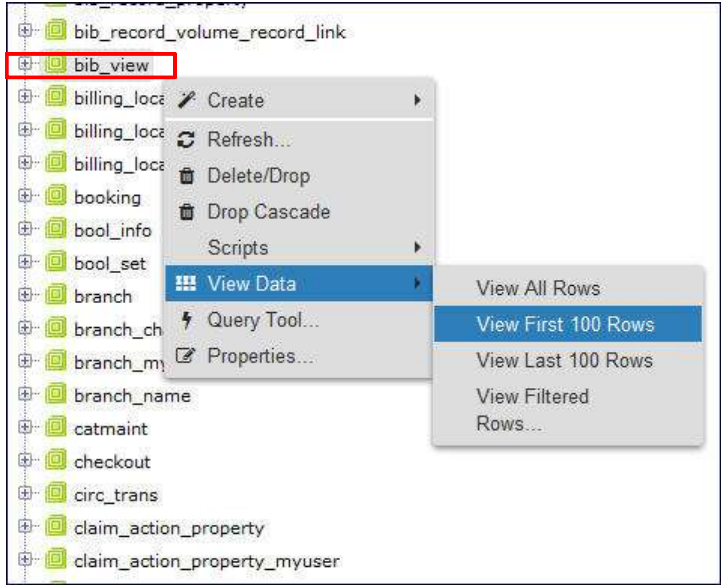
ERD (Entity Relationship Diagram) View


patron_record / patron_item
patron_record _fullname
patron_record _address
patron_record _address_type
patron_record _phone
patron_record _phone_type

This shows which tables are linkable – there is an column in one table that matches a column in another.

In this example, you can link patron_record_address_type to patron_record via patron_record_address

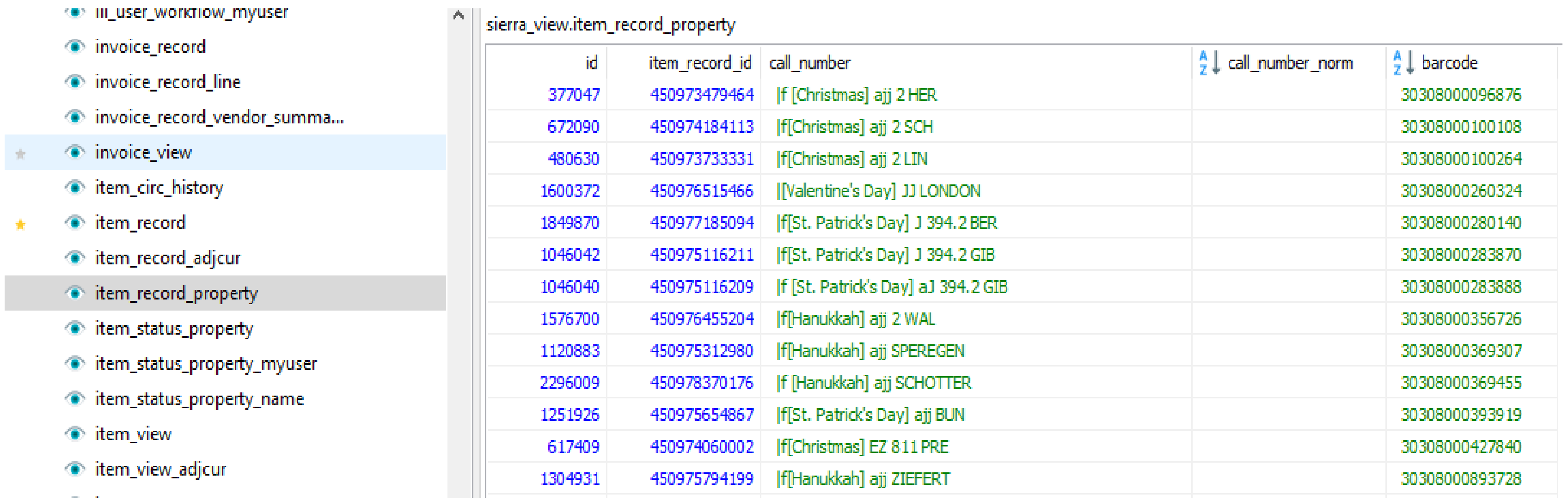
Previewing Your Data



To see sample data from a table, right click on the table (bib_view in this example) and then View Data and then View Top 100 Rows

Screen using PGAdmin https://www.pgadmin.org/

Explore your data



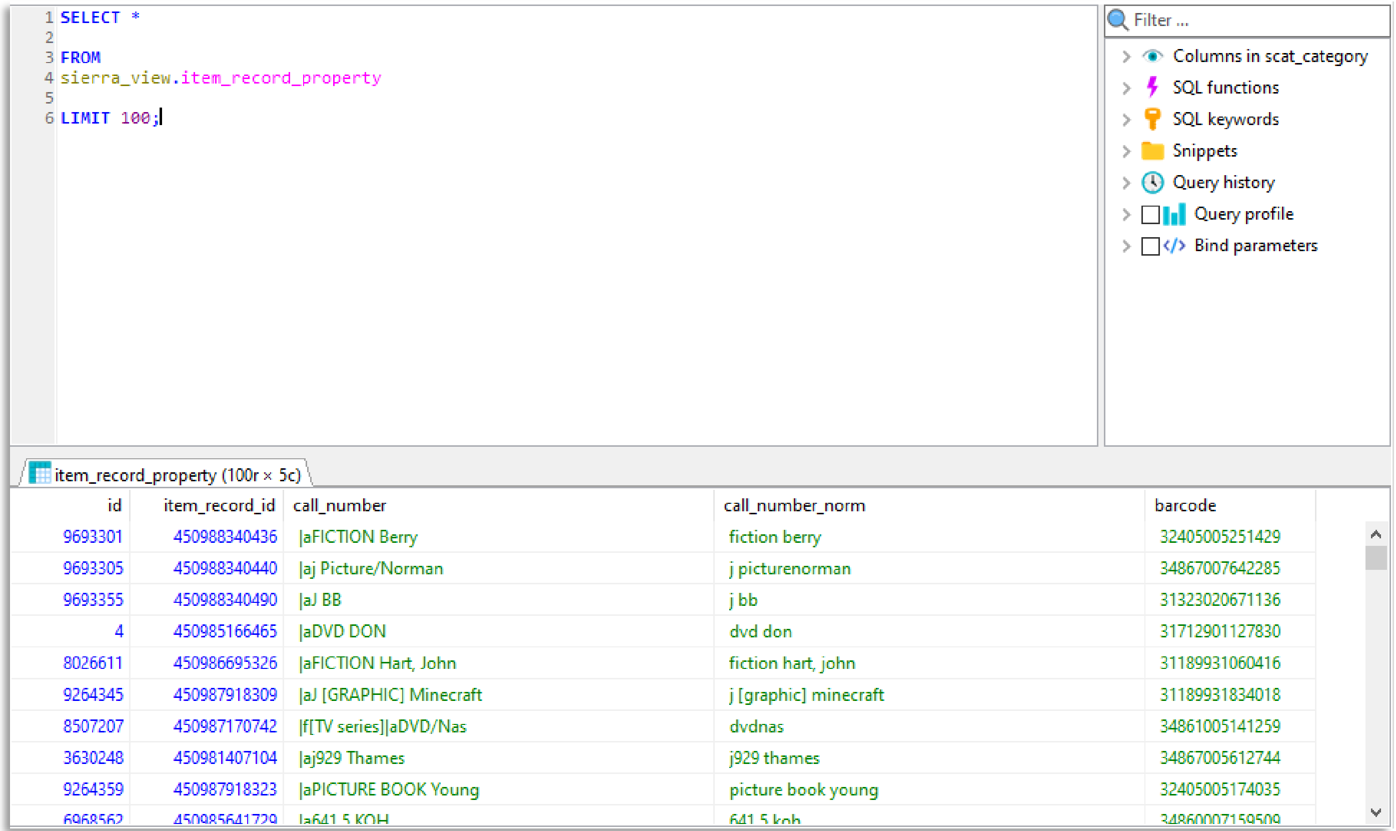
- iii_user_workflow_myuser
- invoice_record
- invoice_record_line
- invoice_record_vendor_summa...
- invoice_view
- item_circ_history
- item_record
- item_record_adjcur
- item_record_property
- item_status_property
- item_status_property_myuser
- item_status_property_name
- item_view
- item_view_adjcur

sierra_view.item_record_property

id	item_record_id	call_number	call_number_norm	barcode
377047	450973479464	|f [Christmas] ajj 2 HER		30308000096876
672090	450974184113	|f[Christmas] ajj 2 SCH		30308000100108
480630	450973733331	|f[Christmas] ajj 2 LIN		30308000100264
1600372	450976515466	|[Valentine's Day] JJ LONDON		30308000260324
1849870	450977185094	|f[St. Patrick's Day] J 394.2 BER		30308000280140
1046042	450975116211	|f[St. Patrick's Day] J 394.2 GIB		30308000283870
1046040	450975116209	|f [St. Patrick's Day] aJ 394.2 GIB		30308000283888
1576700	450976455204	|f[Hanukkah] ajj 2 WAL		30308000356726
1120883	450975312980	|f[Hanukkah] ajj SPEREGEN		30308000369307
2296009	450978370176	|f [Hanukkah] ajj SCHOTTER		30308000369455
1251926	450975654867	|f[St. Patrick's Day] ajj BUN		30308000393919
617409	450974060002	|f[Christmas] EZ 811 PRE		30308000427840
1304931	450975794199	|f[Hanukkah] ajj ZIEFERT		30308000893728

Screens using HeidiSQL https://www.heidisql.com/

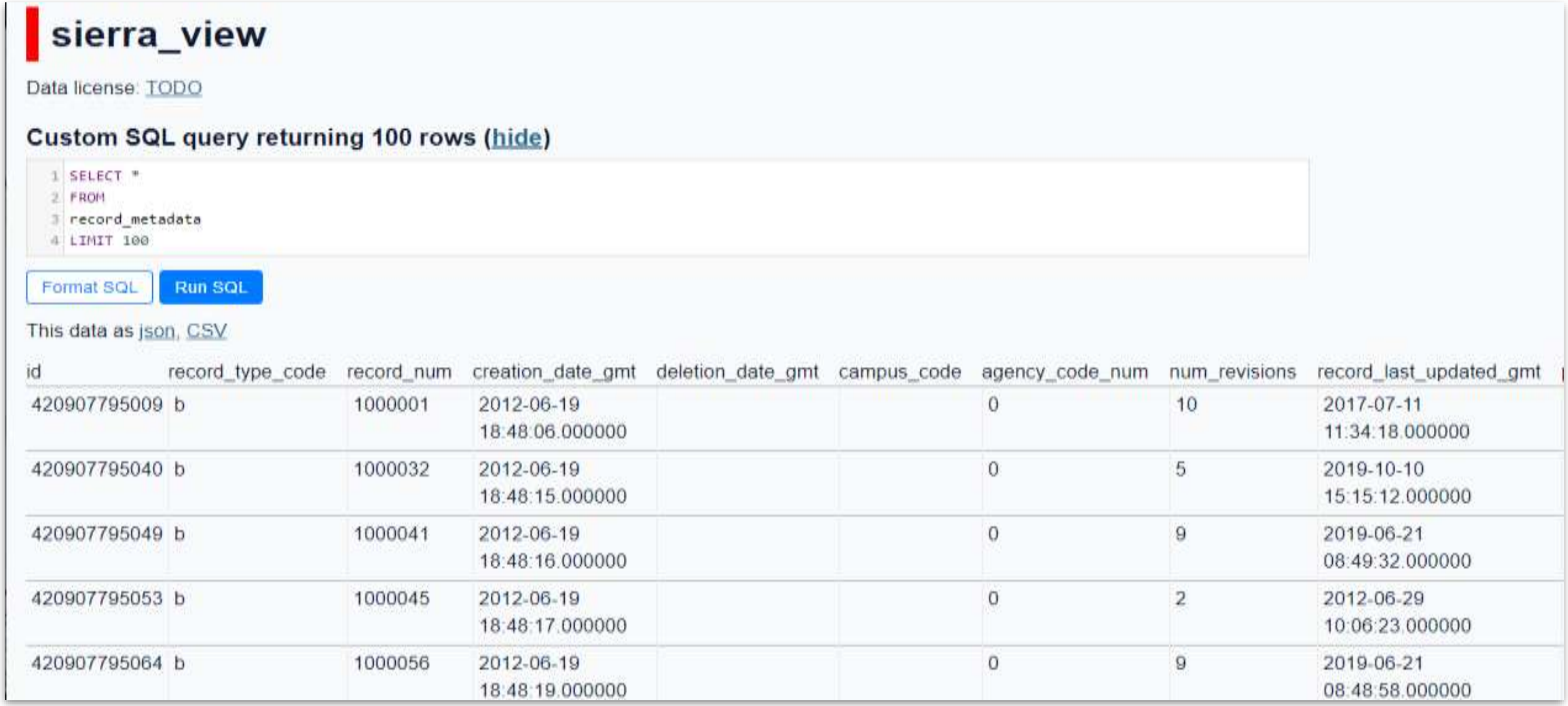
The Query Based Approach



```
SELECT *

FROM
sierra_view.item_record_property

LIMIT 100;
```

id	item_record_id	call_number	call_number_norm	barcode
9693301	450988340436	|aFICTION Berry	fiction berry	32405005251429
9693305	450988340440	|aj Picture/Norman	j picturenorman	34867007642285
9693355	450988340490	|aJ BB	j bb	31323020671136
4	450985166465	|aDVD DON	dvd don	31712901127830
8026611	450986695326	|aFICTION Hart, John	fiction hart, john	31189931060416
9264345	450987918309	|aJ [GRAPHIC] Minecraft	j [graphic] minecraft	31189931834018
8507207	450987170742	|f[TV series]|aDVD/Nas	dvdnas	34861005141259
3630248	450981407104	|aj929 Thames	j929 thames	34867005612744
9264359	450987918323	|aPICTURE BOOK Young	picture book young	32405005174035
6968562	450985641729	|a641.5 KOH	641.5 koh	34860007159509

The Query Based Approach



sierra_view

Data license: TODO

Custom SQL query returning 100 rows (hide)

```
SELECT *
FROM
record_metadata
LIMIT 100
```

Format SQL Run SQL

This data as json, CSV

id	record_type_code	record_num	creation_date_gmt	deletion_date_gmt	campus_code	agency_code_num	num_revisions	record_last_updated_gmt
420907795009	b	1000001	2012-06-19 18:48:06.000000			0	10	2017-07-11 11:34:18.000000
420907795040	b	1000032	2012-06-19 18:48:15.000000			0	5	2019-10-10 15:15:12.000000
420907795049	b	1000041	2012-06-19 18:48:16.000000			0	9	2019-06-21 08:49:32.000000
420907795053	b	1000045	2012-06-19 18:48:17.000000			0	2	2012-06-29 10:06:23.000000
420907795064	b	1000056	2012-06-19 18:48:19.000000			0	9	2019-06-21 08:48:58.000000

Works with the SQL Sandbox from Part 1 too

A Shameless Plug

For more see 2020 presentation:

The Unofficial Guide to Sierra's SQL Views

Casting and Data Types

Data Types

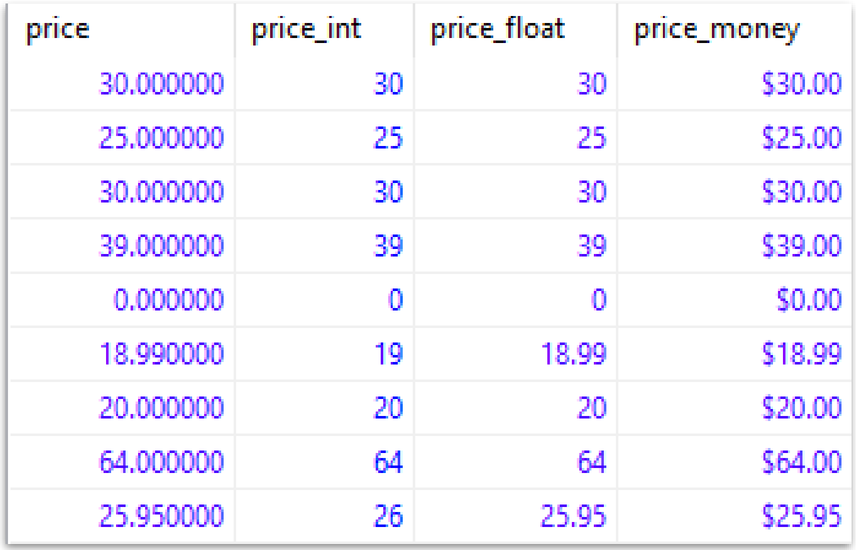
https://www.postgresql.org/docs/current/datatype.html

- Some important and common PostgreSQL data types to understand
  - **INTEGER**: signed, four-byte integer (`1`, `-1`, `42`, etc)
  - **NUMERIC**: real number or **NUMERIC(p,s)** with p digits with s number after the decimal point
    - **MONEY**: Numeric value to 2 decimals places including dollar sign
  - **CHAR**: single character, or `CHAR(n)` fixed-length of `n` characters with *space padded*
  - **VARCHAR(n)**: variable-length character string of `n` characters with *no space padded*
  - **TEXT**: character string with unlimited length
  - **BOOLEAN**: true or false values (can use special `IS TRUE` or `IS FALSE` clause to test)

CAST()

- CAST() will allow you to change the data type of a field
  - :: is a shortcut for the CAST() function

```
SELECT
i.price,
CAST(i.price AS INT) AS price_int,
i.price::FLOAT AS price_float,
i.price::MONEY AS price_money

FROM
sierra_view.item_record i
```



price	price_int	price_float	price_money
30.000000	30	30	$30.00
25.000000	25	25	$25.00
30.000000	30	30	$30.00
39.000000	39	39	$39.00
0.000000	0	0	$0.00
18.990000	19	18.99	$18.99
20.000000	20	20	$20.00
64.000000	64	64	$64.00
25.950000	26	25.95	$25.95

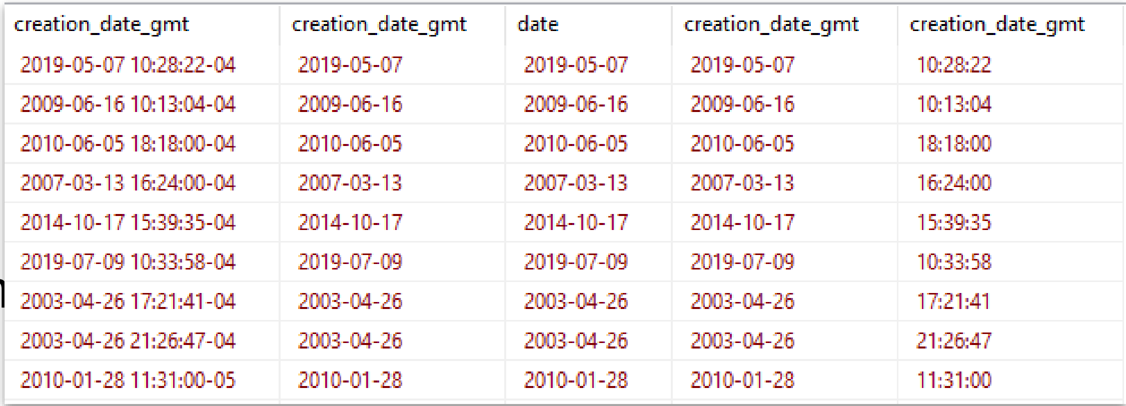
Date Types

https://www.postgresql.org/docs/current/datatype-datetime.html

- Date / Time Types:
  - **DATE**: ISO 8601 (`YYYY-MM-DD`):
    `2019-03-17`
  - **TIMESTAMP**: ISO 8601 date with time (24-hour clock):
    `2019-03-17 11:41:13.979849`
    Time zone is optional
  - **TIMESTAMP WITH TIME ZONE**:
    `2019-03-17 11:41:13.979849-04`
  - **INTERVAL**: defines periods of time
    - Traditional Postgres format:
      `1 year 2 months 3 days 4 hours 5 minutes 6 seconds`

Timestamps

```
SELECT
rm.creation_date_gmt,
CAST(rm.creation_date_gmt AS DATE),
DATE(rm.creation_date_gmt),
rm.creation_date_gmt::DATE,
rm.creation_date_gmt::TIME

FROM
sierra_view.record_metadata r
```

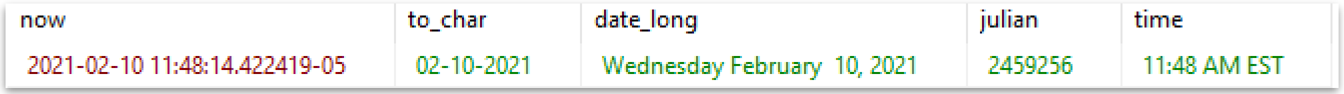


creation_date_gmt	creation_date_gmt	date	creation_date_gmt	creation_date_gmt
2019-05-07 10:28:22-04	2019-05-07	2019-05-07	2019-05-07	10:28:22
2009-06-16 10:13:04-04	2009-06-16	2009-06-16	2009-06-16	10:13:04
2010-06-05 18:18:00-04	2010-06-05	2010-06-05	2010-06-05	18:18:00
2007-03-13 16:24:00-04	2007-03-13	2007-03-13	2007-03-13	16:24:00
2014-10-17 15:39:35-04	2014-10-17	2014-10-17	2014-10-17	15:39:35
2019-07-09 10:33:58-04	2019-07-09	2019-07-09	2019-07-09	10:33:58
2003-04-26 17:21:41-04	2003-04-26	2003-04-26	2003-04-26	17:21:41
2003-04-26 21:26:47-04	2003-04-26	2003-04-26	2003-04-26	21:26:47
2010-01-28 11:31:00-05	2010-01-28	2010-01-28	2010-01-28	11:31:00

TO_CHAR()

- **NOW()** will return current timestamp
- **TO_CHAR()** can be used for date and timestamp formatting

```
SELECT
NOW(),
TO_CHAR(NOW(), 'MM-DD-YYYY'),
TO_CHAR(NOW(), 'Day Month DD, YYYY') AS date_long,
TO_CHAR(NOW(), 'J') AS julian,
TO_CHAR(NOW(), 'HH:MI AM TZ') AS time
```

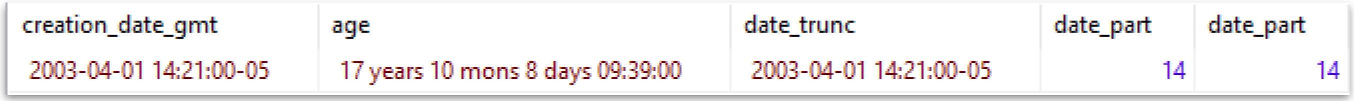


now	to_char	date_long	julian	time
2021-02-10 11:48:14.422419-05	02-10-2021	Wednesday February 10, 2021	2459256	11:48 AM EST

- Template Patterns for Date/Time Formatting can be found here: https://www.postgresql.org/docs/current/functions-formatting.html

Additional Datetime Functions

```
SELECT
rm.creation_date_gmt,
AGE(rm.creation_date_gmt),
DATE_TRUNC('minute', rm.creation_date_gmt),
DATE_PART('hour', rm.creation_date_gmt),
EXTRACT(HOUR FROM rm.creation_date_gmt)
FROM
sierra_view.record_metadata rm
```



creation_date_gmt	age	date_trunc	date_part	date_part
2003-04-01 14:21:00-05	17 years 10 mons 8 days 09:39:00	2003-04-01 14:21:00-05	14	14

- List of available datetime functions can be found here: https://www.postgresql.org/docs/9.1/functions-datetime.html

Functions

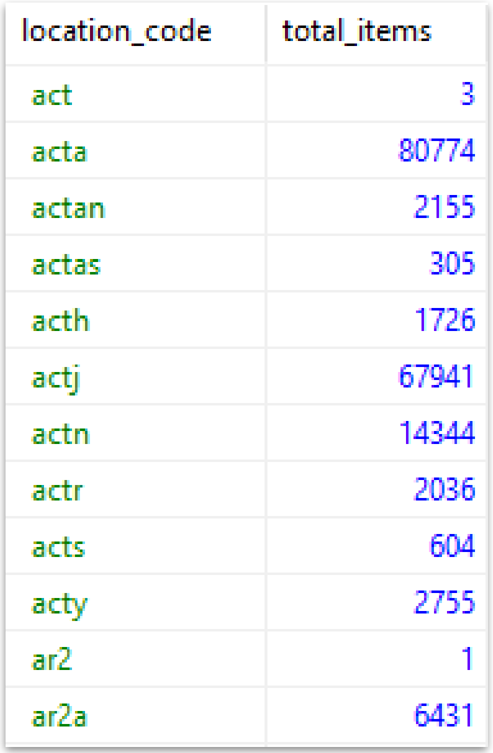
Functions()

- Take the form of function_name(argument(s))
- Allow you to perform actions on your data
- Introduced Aggregate Functions in Part 1
    - Used along with GROUP
        - COUNT()
        - SUM()
        - STRING_AGG()

- Full list of Postgres Aggregate functions available here: https://www.postgresql.org/docs/9.5/functions-aggregate.html

COUNT()

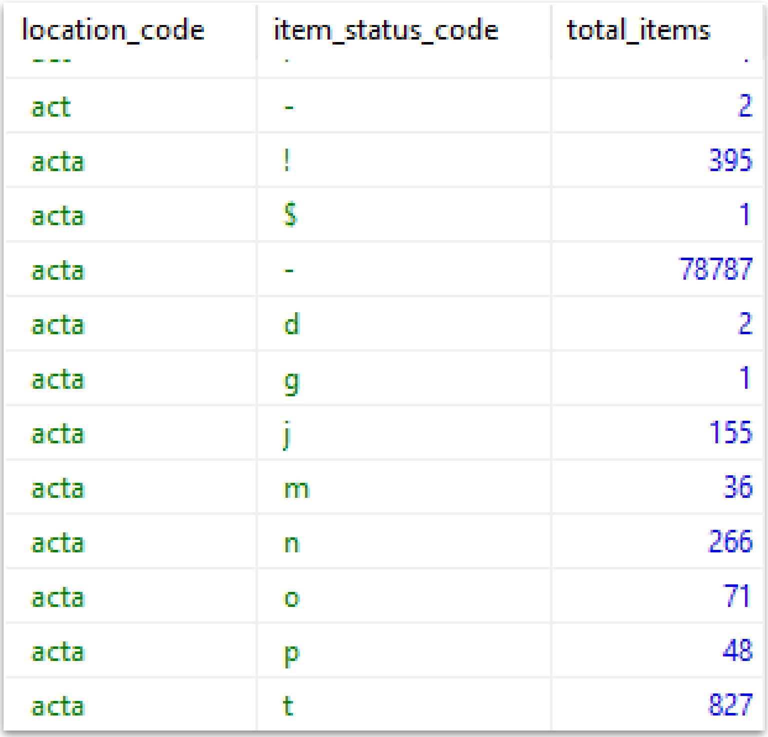
```
SELECT
    i.location_code,
    COUNT(i.id) AS total_items

FROM
    sierra_view.item_record  i

GROUP BY 1
ORDER BY 1;
```



location_code	total_items
act	3
acta	80774
actan	2155
actas	305
acth	1726
actj	67941
actn	14344
actr	2036
acts	604
acty	2755
ar2	1
ar2a	6431

COUNT() Count By Location and Status

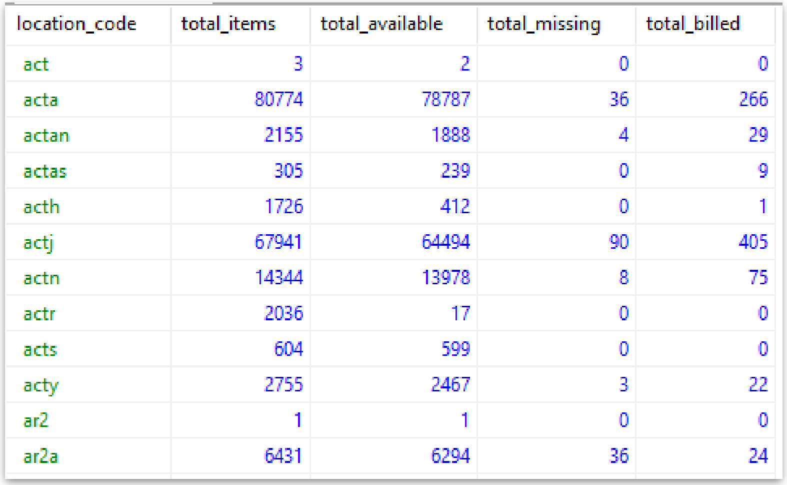
```
SELECT
    i.location_code,
    i.item_status_code,
    COUNT(i.id) AS total_items

FROM
    sierra_view.item_record  i

GROUP BY 1,2
ORDER BY 1;
```



location_code	item_status_code	total_items
act	-	2
acta	!	395
acta	$	1
acta	-	78787
acta	d	2
acta	g	1
acta	j	155
acta	m	36
acta	n	266
acta	o	71
acta	p	48
acta	t	827

FILTER()

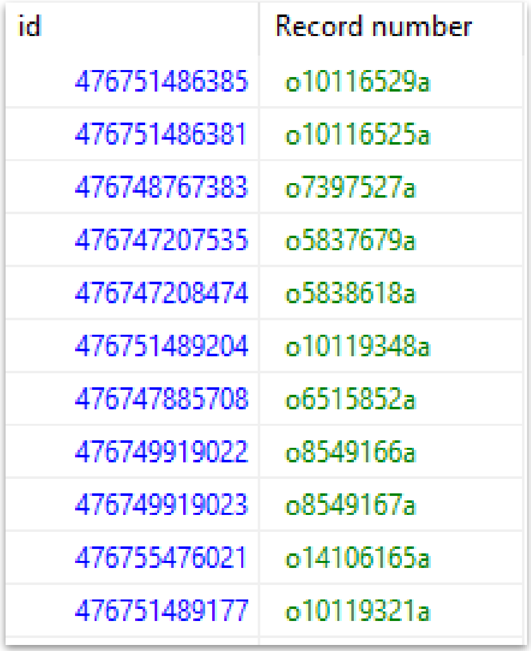
```
SELECT
    i.location_code,
    COUNT(i.id) AS total_items,
    COUNT(i.id) FILTER(WHERE i.item_status_code = '-') AS total_available,
    COUNT(i.id) FILTER(WHERE i.item_status_code = 'm') AS total_missing,
    COUNT(i.id) FILTER(WHERE i.item_status_code = 'n') AS total_billed

FROM
    sierra_view.item_record  i

GROUP BY 1
ORDER BY 1;
```



location_code	total_items	total_available	total_missing	total_billed
act	3	2	0	0
acta	80774	78787	36	266
actan	2155	1888	4	29
actas	305	239	0	9
acth	1726	412	0	1
actj	67941	64494	90	405
actn	14344	13978	8	75
actr	2036	17	0	0
acts	604	599	0	0
acty	2755	2467	3	22
ar2	1	1	0	0
ar2a	6431	6294	36	24

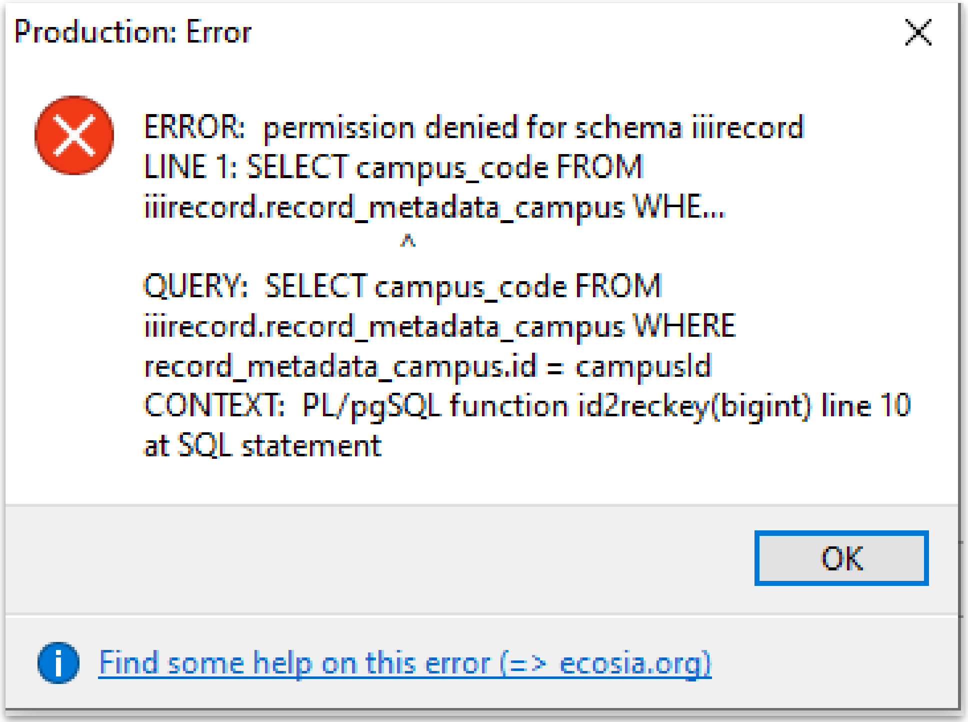
ID2RECKEY()

```
SELECT
  o.id,
  ID2RECKEY(o.record_id) AS "Record number"
FROM
    sierra_view.order_record  o;
```



id	Record number
476751486385	o10116529a
476751486381	o10116525a
476748767383	o7397527a
476747207535	o5837679a
476747208474	o5838618a
476751489204	o10119348a
476747885708	o6515852a
476749919022	o8549166a
476749919023	o8549167a
476755476021	o14106165a
476751489177	o10119321a

ID2RECKEY()


Production: Error

```
ERROR:  permission denied for schema iiirecord
LINE 1: SELECT campus_code FROM
iiirecord.record_metadata_campus WHE...
                                    ^
QUERY:  SELECT campus_code FROM
iiirecord.record_metadata_campus WHERE
record_metadata_campus.id = campusId
CONTEXT:  PL/pgSQL function id2reckey(bigint) line 10
at SQL statement
```

String Functions()

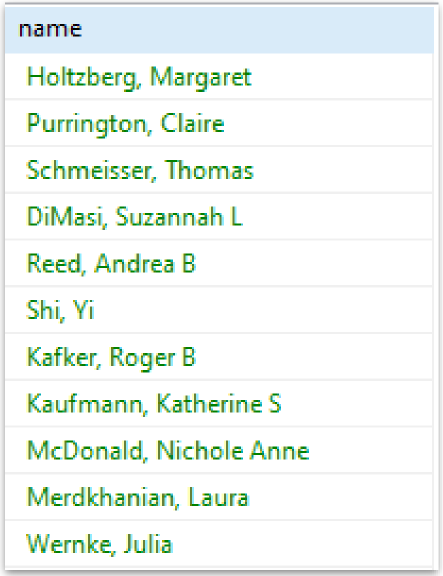
- Take the form of function_name(argument(s))
- Allow you to perform actions on your data
- A few examples
  - LOWER(string) - convert to lowercase
  - LENGTH(string) - count characters in string
  - REPLACE(string, from text, to text) - replace all

- Full list of Postgres String functions available here: https://www.postgresql.org/docs/9.1/functions-string.html

CONCAT()

- Use CONCAT() to combine strings into a single string

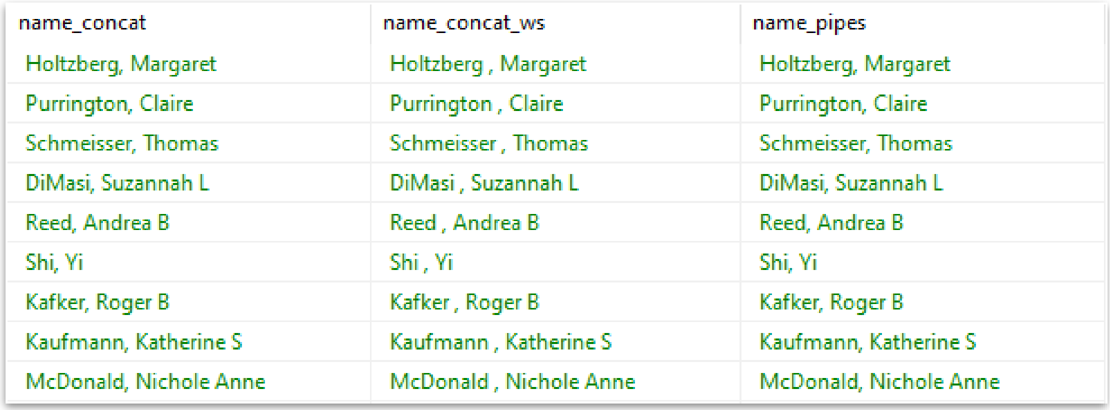
```
SELECT
    CONCAT(p.last_name,', ',p.first_name, ' ', p.middle_name)
        AS name
FROM
    sierra_view.patron_record_fullname  p;
```



name
Holtzberg, Margaret
Purrington, Claire
Schmeisser, Thomas
DiMasi, Suzannah L
Reed, Andrea B
Shi, Yi
Kafker, Roger B
Kaufmann, Katherine S
McDonald, Nichole Anne
Merdkhanian, Laura
Wernke, Julia

More on Concatenation

```
SELECT
 CONCAT(p.last_name,', ',p.first_name, ' ', p.middle_name) AS NAME_concat,
 CONCAT_WS(' ',p.last_name,',',p.first_name, p.middle_name) AS name_concat_ws,
 p.last_name||', '||p.first_name||' '||p.middle_name AS name_pipes

FROM
    sierra_view.patron_record_fullname  p;
```

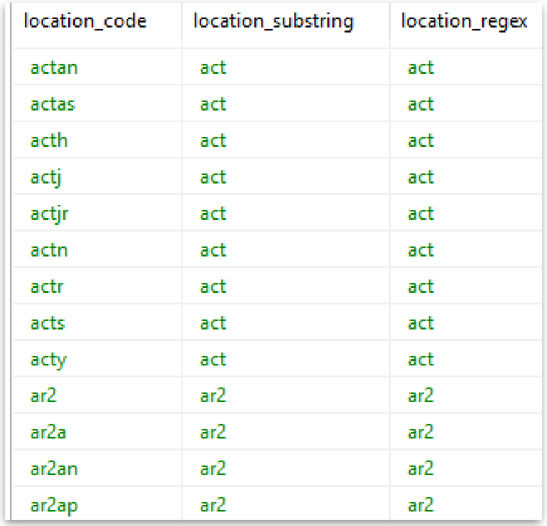


name_concat	name_concat_ws	name_pipes
Holtzberg, Margaret	Holtzberg , Margaret	Holtzberg, Margaret
Purrington, Claire	Purrington , Claire	Purrington, Claire
Schmeisser, Thomas	Schmeisser , Thomas	Schmeisser, Thomas
DiMasi, Suzannah L	DiMasi , Suzannah L	DiMasi, Suzannah L
Reed, Andrea B	Reed , Andrea B	Reed, Andrea B
Shi, Yi	Shi , Yi	Shi, Yi
Kafker, Roger B	Kafker , Roger B	Kafker, Roger B
Kaufmann, Katherine S	Kaufmann , Katherine S	Kaufmann, Katherine S
McDonald, Nichole Anne	McDonald , Nichole Anne	McDonald, Nichole Anne

SUBSTRING()

- Use SUBSTRING() to pull out parts of a string by their position

```
SELECT
i.location_code
SUBSTRING(i.location_code,1,3) AS location_substring,
SUBSTRING(i.location_code,'^.{3}') AS location_regex

FROM sierra_view.item_record i
ORDER BY 1;
```



location_code	location_substring	location_regex
actan	act	act
actas	act	act
acth	act	act
actj	act	act
actjr	act	act
actn	act	act
actr	act	act
acts	act	act
acty	act	act
ar2	ar2	ar2
ar2a	ar2	ar2
ar2an	ar2	ar2
ar2ap	ar2	ar2

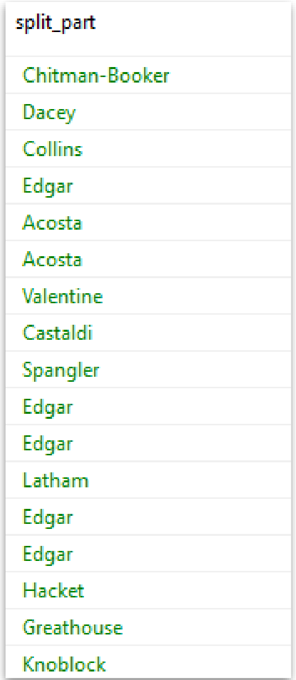
SPLIT_PART(): Author Last names

- Use SPLIT_PART() to parse strings on a specified delimiter

SELECT

SPLIT_PART(b.best_author, ‘, ‘, 1) AS last_name

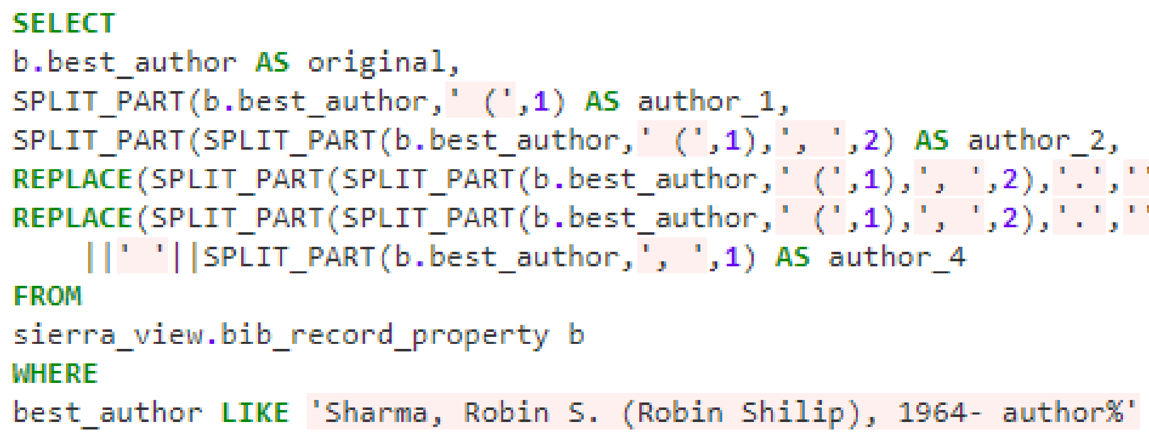
FROM sierra_view.bib_record_property b;



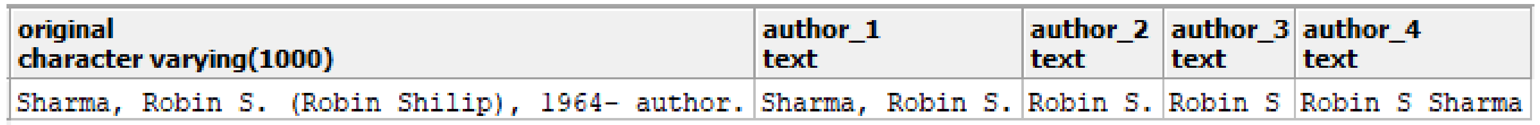
split_part
Chitman-Booker
Dacey
Collins
Edgar
Acosta
Acosta
Valentine
Castaldi
Spangler
Edgar
Edgar
Latham
Edgar
Edgar
Hacket
Greathouse
Knoblock

Nesting String Functions

Using string functions to display an author in first name, last name order



```
SELECT
b.best_author AS original,
SPLIT_PART(b.best_author,' (',1) AS author_1,
SPLIT_PART(SPLIT_PART(b.best_author,' (',1),', ',2) AS author_2,
REPLACE(SPLIT_PART(SPLIT_PART(b.best_author,' (',1),', ',2),'.','') AS author_3,
REPLACE(SPLIT_PART(SPLIT_PART(b.best_author,' (',1),', ',2),'.','')
    ||' '||SPLIT_PART(b.best_author,', ',1) AS author_4
FROM
sierra_view.bib_record_property b
WHERE
best_author LIKE 'Sharma, Robin S. (Robin Shilip), 1964- author%'
```



original character varying(1000)	author_1 text	author_2 text	author_3 text	author_4 text
Sharma, Robin S. (Robin Shilip), 1964- author.	Sharma, Robin S.	Robin S.	Robin S	Robin S Sharma

Window Functions

Window Functions

Window Functions allow you to perform calculations across related rows

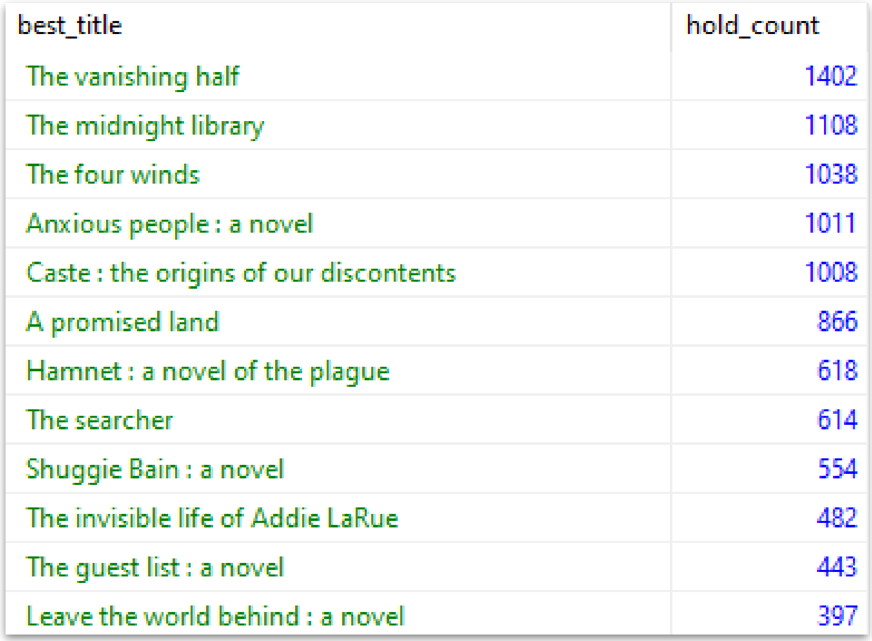
Use the Syntax [function]() OVER (field)

Some examples of window functions are:

- row_number()
- rank()
- ntile()
    - The list of available window functions can be found here: https://www.postgresql.org/docs/9.3/functions-window.html

Top Requested Titles

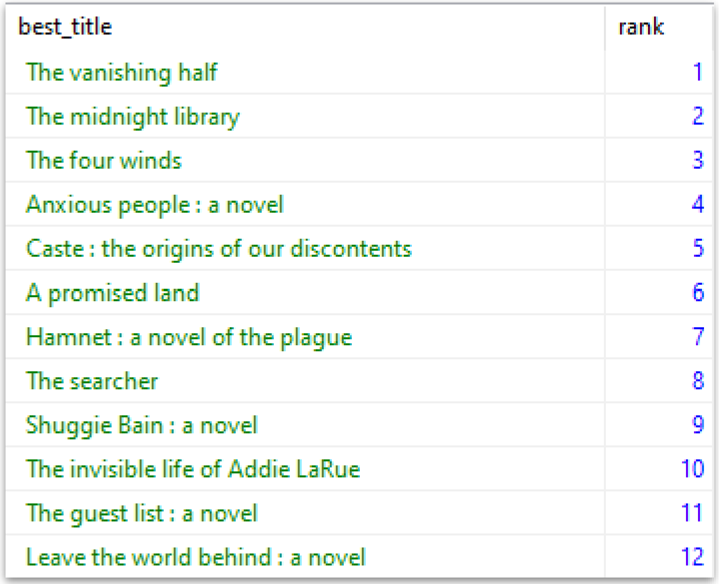
```
SELECT
b.best_title,
COUNT(h.id) AS hold_count

FROM
sierra_view.hold h
JOIN
sierra_view.bib_record_property b
ON h.record_id = b.bib_record_id

GROUP BY 1
ORDER BY 2 DESC
```



best_title	hold_count
The vanishing half	1402
The midnight library	1108
The four winds	1038
Anxious people : a novel	1011
Caste : the origins of our discontents	1008
A promised land	866
Hamnet : a novel of the plague	618
The searcher	614
Shuggie Bain : a novel	554
The invisible life of Addie LaRue	482
The guest list : a novel	443
Leave the world behind : a novel	397

RANK()

```
SELECT
b.best_title,
RANK() OVER (ORDER BY COUNT(h.id) DESC) AS rank

FROM
sierra_view.hold h
JOIN
sierra_view.bib_record_property b
ON h.record_id = b.bib_record_id

GROUP BY 1
ORDER BY 2
```



best_title	rank
The vanishing half	1
The midnight library	2
The four winds	3
Anxious people : a novel	4
Caste : the origins of our discontents	5
A promised land	6
Hamnet : a novel of the plague	7
The searcher	8
Shuggie Bain : a novel	9
The invisible life of Addie LaRue	10
The guest list : a novel	11
Leave the world behind : a novel	12

PARTITION

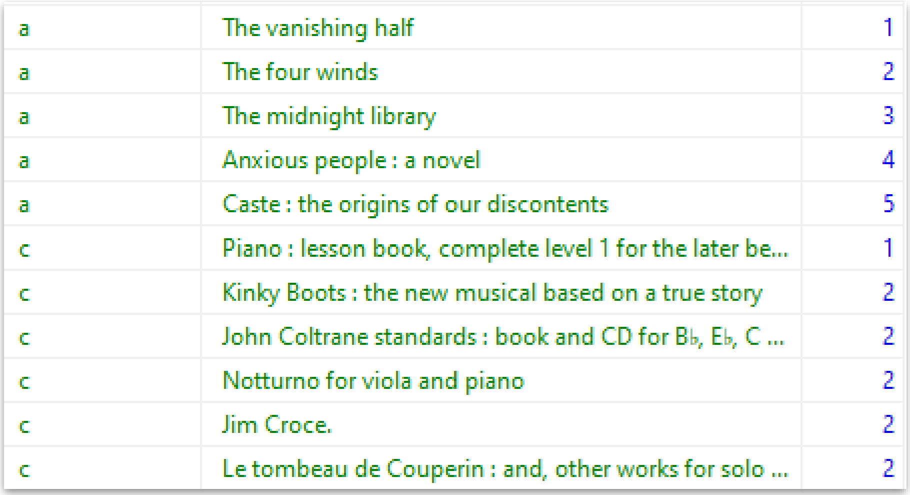
The PARTITION clause allow us to subdivide a table into smaller sets of rows

In combination with a window function we can then apply that function to subsets of our data

```
RANK() OVER (
    PARTITION BY b.material_code
    ORDER BY COUNT(h.id) DESC
) AS rank
```

Top Requested Titles By Format

```
SELECT *
FROM (
SELECT
b.material_code, b.best_title,
RANK() OVER (PARTITION BY b.material_code ORDER BY COUNT(h.id) DESC) AS rank
FROM
sierra_view.hold h
JOIN
sierra_view.bib_record_property b
ON h.record_id = b.bib_record_id
GROUP BY 1,2
)inner_query

WHERE inner_query.rank < 6
ORDER BY 1,3
```



a	The vanishing half	1
a	The four winds	2
a	The midnight library	3
a	Anxious people : a novel	4
a	Caste : the origins of our discontents	5
c	Piano : lesson book, complete level 1 for the later be...	1
c	Kinky Boots : the new musical based on a true story	2
c	John Coltrane standards : book and CD for B♭, E♭, C ...	2
c	Notturno for viola and piano	2
c	Jim Croce.	2
c	Le tombeau de Couperin : and, other works for solo ...	2

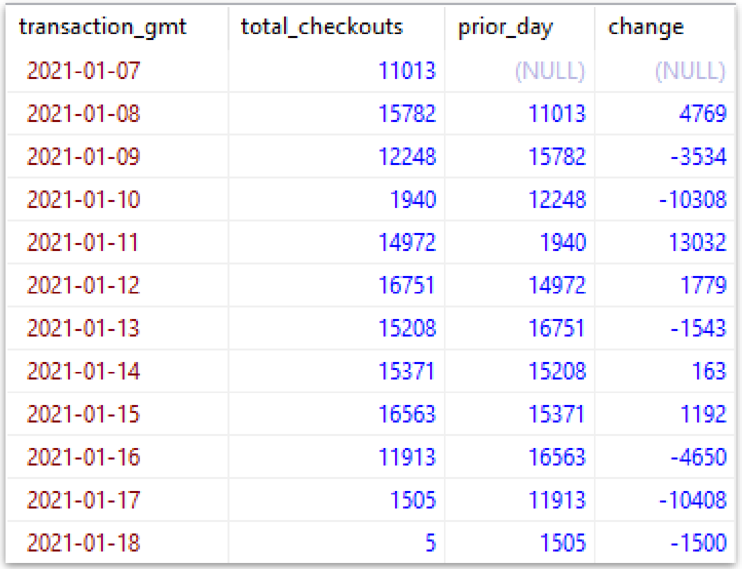
LAG() & LEAD()

- LAG() & LEAD() allow you to utilize a field from a neighboring row

- LAG(COUNT(id), 1)
  - Retrieves the value of the id field from 1 row prior.

Daily Checkout Comparison

```
SELECT
        c.transaction_gmt::DATE,
        COUNT(c.id) AS total_checkouts,
        LAG(COUNT(c.id),1)
          OVER (ORDER BY c.transaction_gmt::DATE) AS prior_day,
        COUNT(c.id) - LAG(COUNT(c.id),1)
          OVER (ORDER BY c.transaction_gmt::DATE) AS change
FROM sierra_view.circ_trans c
WHERE c.op_code = 'o'
GROUP BY 1
ORDER BY 1;
```



transaction_gmt	total_checkouts	prior_day	change
2021-01-07	11013	(NULL)	(NULL)
2021-01-08	15782	11013	4769
2021-01-09	12248	15782	-3534
2021-01-10	1940	12248	-10308
2021-01-11	14972	1940	13032
2021-01-12	16751	14972	1779
2021-01-13	15208	16751	-1543
2021-01-14	15371	15208	163
2021-01-15	16563	15371	1192
2021-01-16	11913	16563	-4650
2021-01-17	1505	11913	-10408
2021-01-18	5	1505	-1500

Combining Queries

EXISTS / NOT EXISTS

The Exists operator tests for the existence of a row in a subquery

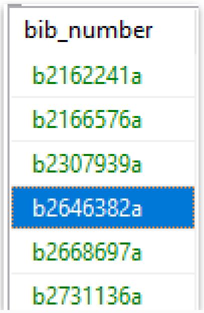
- If there is a result then TRUE else FALSE

Use it within a WHERE clause to limit results based on a subquery

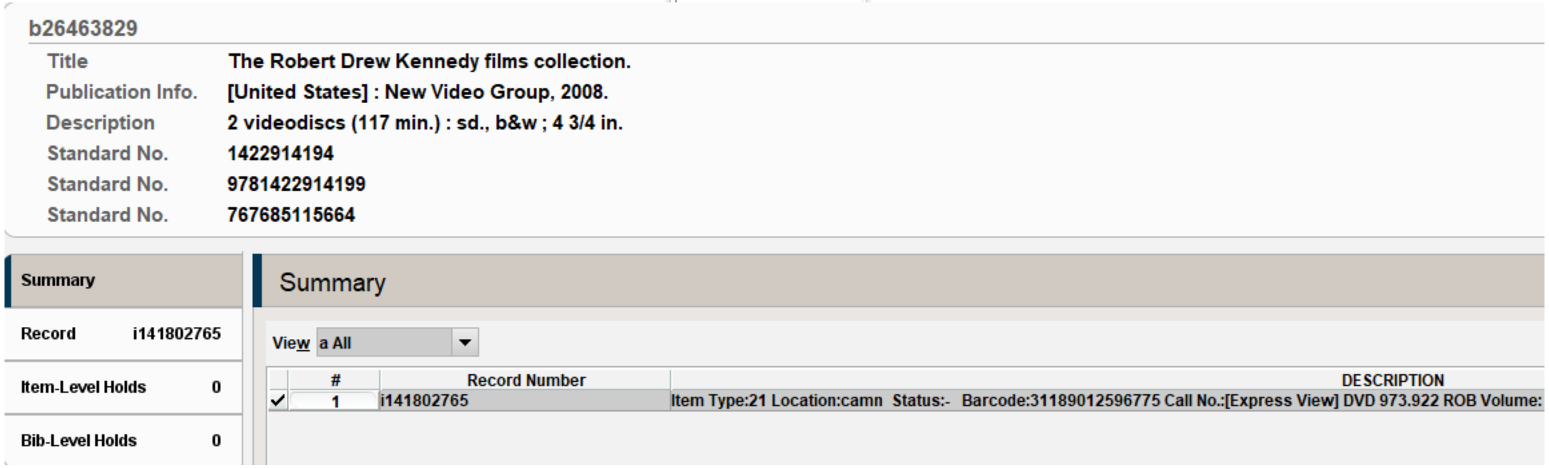
Titles Where All Items Share an Itype

```
SELECT id2reckey(b.id)||'a' AS bib_number
  FROM sierra_view.bib_record b
WHERE EXISTS (
  SELECT l.id
  FROM  sierra_view.bib_record_item_record_link l
  JOIN   sierra_view.item_record i ON l.item_record_id = i.id
  WHERE  b.id = l.bib_record_id AND i.itype_code_num = '21')
AND NOT EXISTS (
  SELECT l.id
  FROM sierra_view.bib_record_item_record_link l
  JOIN sierra_view.item_record i ON l.item_record_id = i.id
  WHERE  b.id = l.bib_record_id AND i.itype_code_num != '21')
ORDER BY 1
```

Titles Where All Items Share an Itype



bib_number
b2162241a
b2166576a
b2307939a
b2646382a
b2668697a
b2731136a



b26463829

Title	The Robert Drew Kennedy films collection.
Publication Info.	[United States] : New Video Group, 2008.
Description	2 videodiscs (117 min.) : sd., b&w ; 4 3/4 in.
Standard No.	1422914194
Standard No.	9781422914199
Standard No.	767685115664

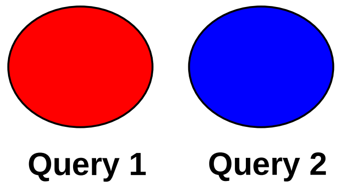
Record	i141802765
Item-Level Holds	0
Bib-Level Holds	0

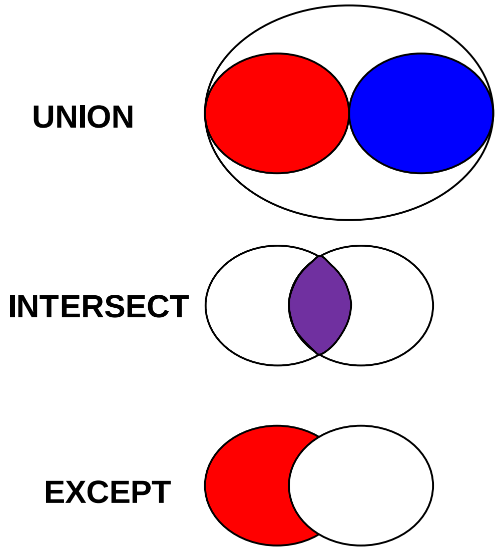
Summary

View a All

	#	Record Number	DESCRIPTION
✓	1	i141802765	Item Type:21 Location:camn Status:- Barcode:31189012596775 Call No.:[Express View] DVD 973.922 ROB Volume:

Intersect/ Except/ Union





UNION returns the combined results of the two queries.
INTERSECT returns the results shared by the two queries
EXCEPT returns the results in the first query, but not in the second

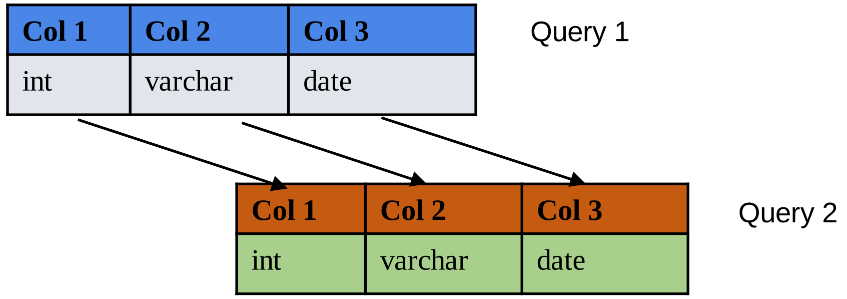
All 3 follow two rules – the queries must have the same number of columns and each column must match data type.

Intersect/ Except/ Union

To combine the two queries, we insert UNION between them. This takes the results of each query and displays the results of both as if they were part of one query. In order for this to work, UNION the combined queries must follow two rules:

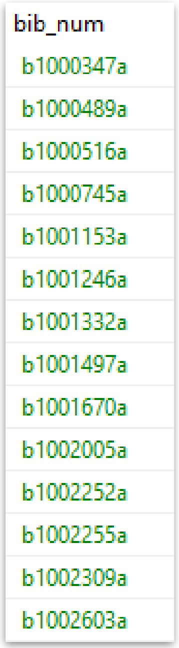
1. The queries must produce the same number of columns
2. Each column must match on data type.

One ORDER BY command may be applied to the combined results at the end of the last query to sort the entire set of results.



INTERSECT: Bibs with both Items and Orders

```
SELECT
      ID2RECKEY(il.bib_record_id)||’a’ AS bib_num
FROM
sierra_view.bib_record_item_record_link il
   INTERSECT
   SELECT
            ID2RECKEY(ol.bib_record_id)||’a’ AS bib_num
            FROM sierra_view.bib_record_order_record_link ol
ORDER BY bib_num;
```

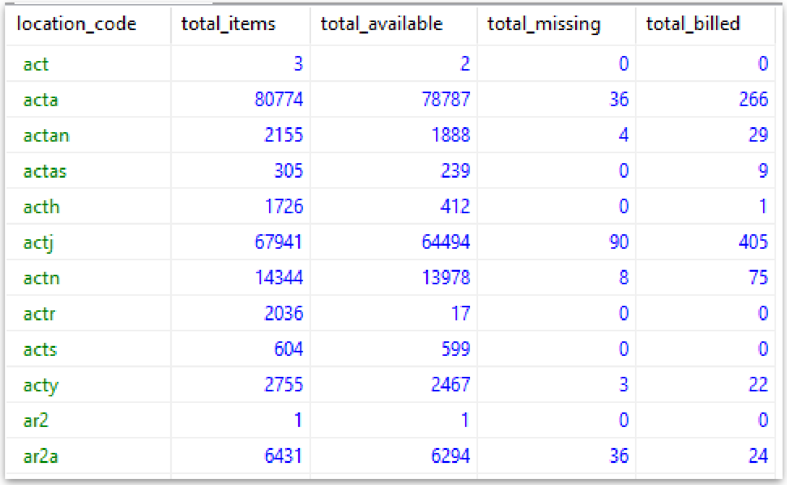


bib_num
b1000347a
b1000489a
b1000516a
b1000745a
b1001153a
b1001246a
b1001332a
b1001497a
b1001670a
b1002005a
b1002252a
b1002255a
b1002309a
b1002603a

Time for One Last Query

Previously: Item Count By Location and Status

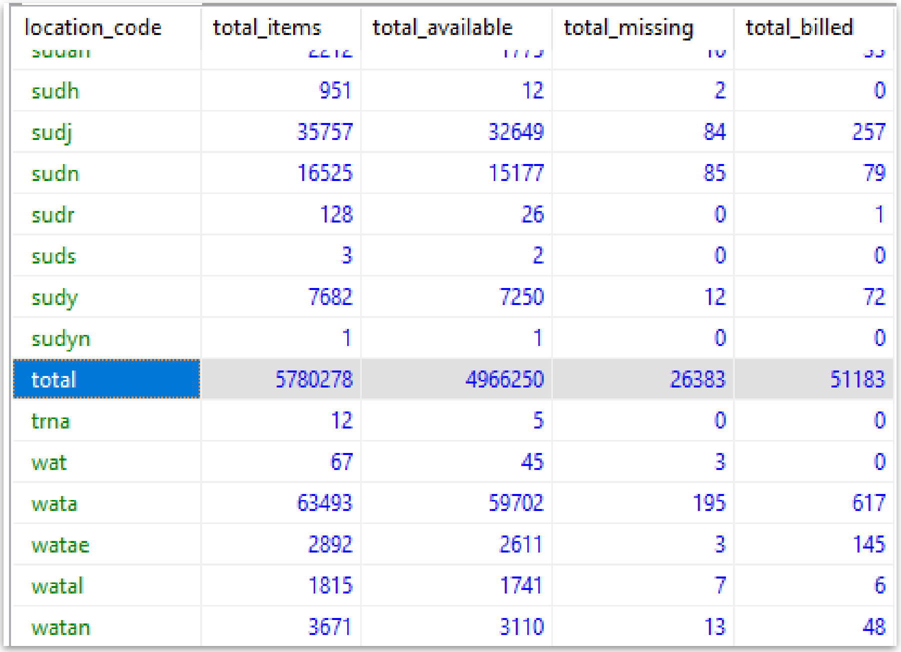
```
SELECT
    i.location_code,
    COUNT(i.id) AS total_items,
    COUNT(i.id) FILTER(WHERE i.item_status_code = '-') AS total_available,
    COUNT(i.id) FILTER(WHERE i.item_status_code = 'm') AS total_missing,
    COUNT(i.id) FILTER(WHERE i.item_status_code = 'n') AS total_billed

FROM
    sierra_view.item_record  i

GROUP BY 1
ORDER BY 1;
```



location_code	total_items	total_available	total_missing	total_billed
act	3	2	0	0
acta	80774	78787	36	266
actan	2155	1888	4	29
actas	305	239	0	9
acth	1726	412	0	1
actj	67941	64494	90	405
actn	14344	13978	8	75
actr	2036	17	0	0
acts	604	599	0	0
acty	2755	2467	3	22
ar2	1	1	0	0
ar2a	6431	6294	36	24

UNION: Adding a Total Row

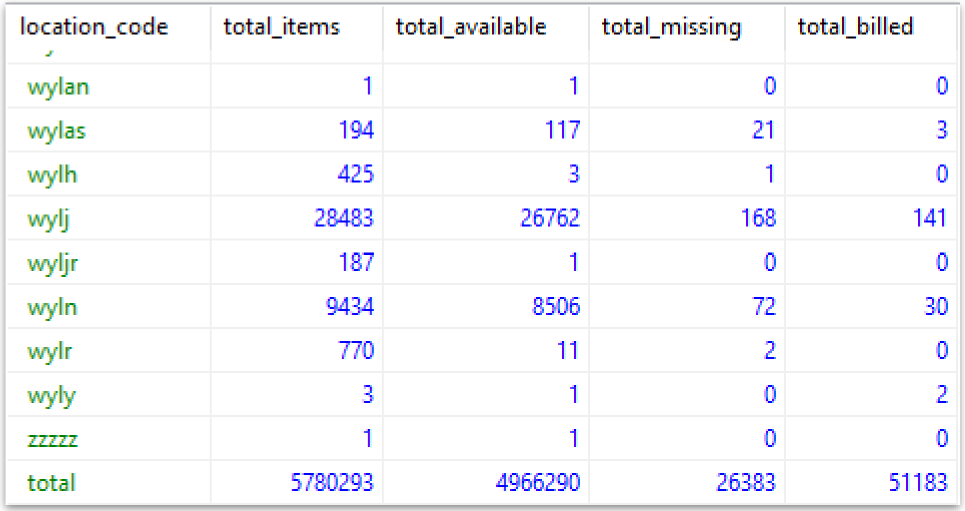
```
SELECT
    i.location_code,
    COUNT(i.id) AS total_items,
    COUNT(i.id) FILTER(WHERE i.item_status_code = '-') AS total_available,
    COUNT(i.id) FILTER(WHERE i.item_status_code = 'm') AS total_missing,
    COUNT(i.id) FILTER(WHERE i.item_status_code = 'n') AS total_billed

FROM
    sierra_view.item_record  i

GROUP BY 1

UNION

SELECT
   'total',
   COUNT(i.id) AS total_items,
   COUNT(i.id) FILTER(WHERE i.item_status_code = '-') AS total_available,
   COUNT(i.id) FILTER(WHERE i.item_status_code = 'm') AS total_missing,
   COUNT(i.id) FILTER(WHERE i.item_status_code = 'n') AS total_billed

FROM
   sierra_view.item_record  i

ORDER BY location_code
```



location_code	total_items	total_available	total_missing	total_billed
[illegible]	[illegible]	[illegible]	[illegible]	[illegible]
sudh	951	12	2	0
sudj	35757	32649	84	257
sudn	16525	15177	85	79
sudr	128	26	0	1
suds	3	2	0	0
sudy	7682	7250	12	72
sudyn	1	1	0	0
total	5780278	4966250	26383	51183
trna	12	5	0	0
wat	67	45	3	0
wata	63493	59702	195	617
watae	2892	2611	3	145
watal	1815	1741	7	6
watan	3671	3110	13	48

UNION: Adding a Total Row

```
SELECT *
FROM(
SELECT
   i.location_code,
   COUNT(i.id) AS total_items,
   COUNT(i.id) FILTER(WHERE i.item_status_code = '-') AS total_available,
   COUNT(i.id) FILTER(WHERE i.item_status_code = 'm') AS total_missing,
   COUNT(i.id) FILTER(WHERE i.item_status_code = 'n') AS total_billed

FROM
   sierra_view.item_record  i

GROUP BY 1

UNION

SELECT
  'total',
  COUNT(i.id) AS total_items,
  COUNT(i.id) FILTER(WHERE i.item_status_code = '-') AS total_available,
  COUNT(i.id) FILTER(WHERE i.item_status_code = 'm') AS total_missing,
  COUNT(i.id) FILTER(WHERE i.item_status_code = 'n') AS total_billed

FROM
  sierra_view.item_record  i
)inner_query

ORDER BY CASE
                        WHEN location_code = 'total' THEN 2
                        ELSE 1
          END,
          location_code
```



location_code	total_items	total_available	total_missing	total_billed
wylan	1	1	0	0
wylas	194	117	21	3
wylh	425	3	1	0
wylj	28483	26762	168	141
wyljr	187	1	0	0
wyln	9434	8506	72	30
wylr	770	11	2	0
wyly	3	1	0	2
zzzzz	1	1	0	0
total	5780293	4966290	26383	51183

Additional Resources

Additional Resources

- Presentation Site
  - https://site-checker.cincy.pl/iug2021/
- PostgreSQL Official Documentation
  - https://www.postgresql.org/docs/
- Stackoverflow
  - https://stackoverflow.com/
- SQL Cookbook by Anthony Molinaro
  - O'Reilly, 2005
- SQL Murder Mystery
  - https://mystery.knightlab.com/

Find Us on Slack

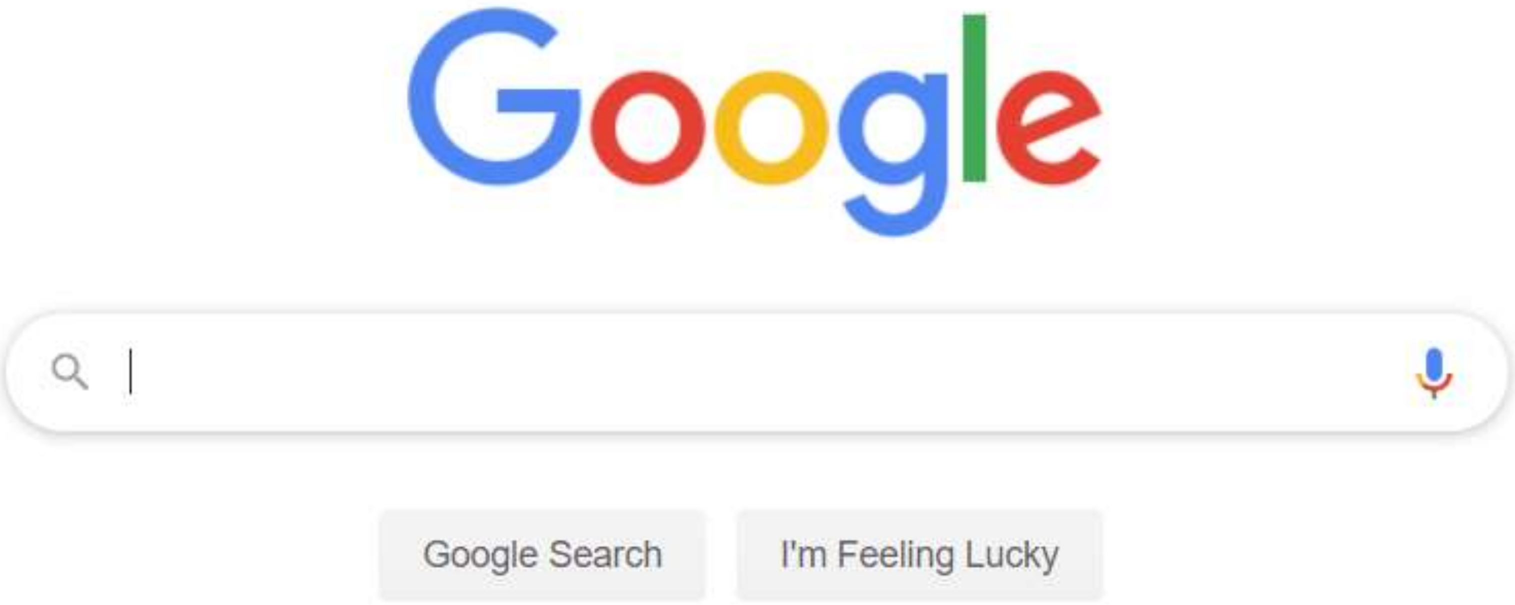
Jeremy & Ray can be found along with many other Sierra SQL experts, on the Sierra-ILS Slack workspace

Invite link will be available on the presentation site page:

https://howtosql.cincy.pl/iug2021/

Or e-mail Jeremy or Ray

JGI



Thank You!

Jeremy Goldstein
jgoldstein@minlib.net

Ray Voelker
ray.voelker@cincinnatilibrary.org