GRIN-Global SQL Library

Revision Date
May 31, 2022

A change to the Server (2.0.5) – April, 2021, has impacted some of the SQL query examples included in this document. Until this document is completely edited and the queries are revised if necessary, in general, whenever owned_date or created_by is used, try substituting owned_date or owned_by.

Editor
Martin Reisinger

Contributors
Many contributors, but especially Kurt Endress at the National Plant Germplasm System DBMU.

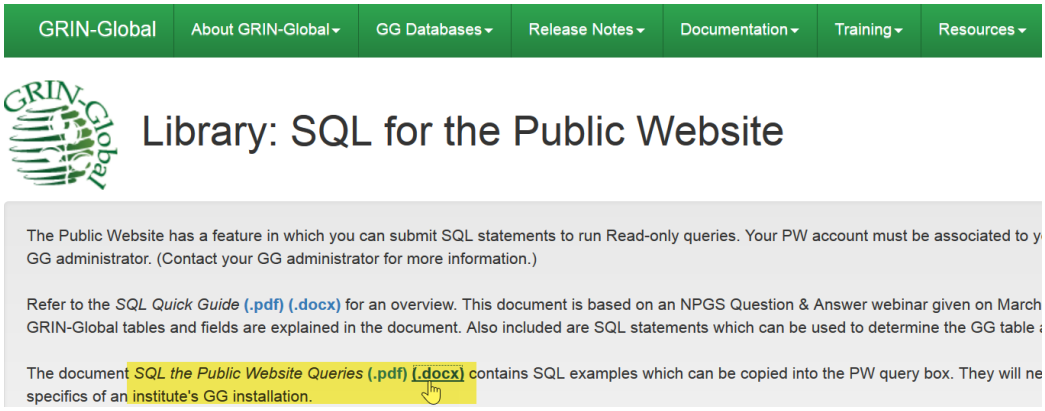
If a query is not delivering results as expected, please contact marty.reisinger@usda.gov. Stuff happens!

If you have a query that you think GG users will find useful, please send it to me along with a brief description – in advance, thanks!

--marty reisinger

Contents

This is an online document stored at:



The direct URL is: https://www.grin-global.org/sql_examples.htm

A companion document, "Quick Guide to SQL with GRIN-Global," is available online at https://www.grin-global.org/docs/gg_sql_quick_guide.docx

Items in red are used as examples and should be replaced with data relative to your GG database / query.
/* indicates a comment */ also a double-dash comments the remainder of the line

SQL comments are embedded in paired delimiters. Example:

/* WHERE s.site_short_name = 'NC7' */

Comments may also be included at the end of the line and are indicated with a --
Example:

WHERE final_recipient_cooperator_id =128348 -- coop id for Svalbard orders

Basic Queries

Display all fields for table *xyz*

In this example, list all the records (and all the records' fields – the * indicated that) after a specified number of days. Optionally, instead of using the asterisk, specify the field names. To determine the fields for a table, see Display *Table* Field Names, Data Type,…,

```
SELECT *
FROM
web_order_request_action
WHERE
created_date > (GETUTCDATE() -2)
```

```
SELECT
web_order_request_id,  action_code,  acted_date,  action_for_id,  note
FROM
web_order_request_action
WHERE
```

Site or Accession Data

Curators and their Crops & Species

```
SELECT distinct ts.name AS Species_name, c.last_name AS Curator, crop.name AS Crop_Name
FROM taxonomy_species ts

    JOIN taxonomy_crop_map tcm
        ON  ts.taxonomy_species_id = tcm.taxonomy_species_id
    JOIN crop
        ON  crop.crop_id = tcm.crop_id
    JOIN crop_trait ct
         ON crop.crop_id = ct.crop_id
    JOIN cooperator c
         ON ct.owned_by = c.cooperator_id
    JOIN site s
         ON c.site_id = s.site_id
WHERE  s.site_short_name = 'NC7'
ORDER BY ts.name
```

```
SELECT distinct a.accession_number_part1, a.accession_number_part2, a.accession_number_part3,
ts.name AS Species_name, c.last_name AS Curator, crop.name AS Crop_Name
FROM accession a

JOIN taxonomy_species ts
ON a. taxonomy_species_id =  ts.taxonomy_species_id

    JOIN taxonomy_crop_map tcm
        ON  ts.taxonomy_species_id = tcm.taxonomy_species_id
    JOIN crop
        ON  crop.crop_id = tcm.crop_id
    JOIN crop_trait ct
         ON crop.crop_id = ct.crop_id
    JOIN cooperator c
         ON ct.owned_by = c.cooperator_id
    JOIN site s
         ON c.site_id = s.site_id
/* WHERE  s.site_short_name = 'NC7' */
WHERE  ts.name = 'Phaseolus vulgaris'
ORDER BY ts.name
```

Curators and their Genera

```
SELECT DISTINCT  genus_name, s.site_short_name, last_name, first_name

  FROM taxonomy_genus tg
  JOIN taxonomy_species ts
        ON ts.taxonomy_genus_id = tg.taxonomy_genus_id
  JOIN accession a
        ON a.taxonomy_species_id = ts.taxonomy_species_id
  JOIN cooperator c
        ON a.owned_by = c.cooperator_id
  JOIN site s
        ON c.site_id = s.site_id

WHERE s.site_short_name = 'NC7'
AND a.status_code = 'ACTIVE'

/*  or by
WHERE  genus_name = 'Genus'
*/

ORDER BY genus_name
```

Crops with Observations for a Specified Site

```
SELECT crop.name AS Crop,
     COUNT(*) AS Total_obs
FROM   crop
    JOIN crop_trait ct
          ON crop.crop_id = ct.crop_id
    JOIN crop_trait_observation cto
          ON ct.crop_trait_id = cto.crop_trait_id
    JOIN cooperator c
          ON ct.owned_by = c.cooperator_id
    JOIN site s
          ON c.site_id = s.site_id
WHERE  s.site_short_name = 'S9'
GROUP  BY crop.name
ORDER  BY crop.name
```

List Your Site's Top Accession Names

```
SELECT
  a.accession_number_part1,
  a.accession_number_part2,
  a.accession_number_part3,
  ain.category_code,
  ain.plant_name,
  ain.plant_name_rank

FROM accession a
/* see note below re site_id */
JOIN cooperator ac ON a.owned_by = ac.cooperator_id AND ac.site_id = 29
JOIN accession_inv_name ain ON ain.accession_inv_name_id =
  (SELECT TOP 1 accession_inv_name_id
   FROM accession_inv_name
  JOIN inventory ON accession_inv_name.inventory_id = inventory.inventory_id
   WHERE inventory.accession_id = a.accession_id
   ORDER BY accession_inv_name.plant_name_rank
  )
```

Determine your site id by using the Site dataview in the CT. In the example above, **ac.site_id = 29** represents Potato Germplasm Introduction Station (NR6).

Display Inventory Fields for a Specific Accession

```
SELECT
inventory_id,
inventory_number_part1,
inventory_number_part2,
inventory_number_part3,
form_type_code,
distribution_default_form_code
FROM inventory
WHERE accession_id = 1927546
```

Count of accessions with more than one available, distributable inventory

Count of accession with more than one available, distributable inventory by site

```
SELECT count(*) AS Accessions_With_Multiple_Avail_Inv, site_short_name
/* Count of accession with more than one available, distributable inventory by site */


FROM accession a
JOIN cooperator c ON c.cooperator_id = a.owned_by
JOIN site s ON s.site_id = c.site_id
WHERE (SELECT count(*) FROM inventory WHERE accession_id = a.accession_id AND is_distributable =
'Y' AND is_available = 'Y') > 1
GROUP BY site_short_name
```

List Unavailable Accessions

This SQL is explained in the online SQL Guide, in the section “Exists (and subqueries)”

```
SELECT a.*
FROM accession a
JOIN taxonomy_species ts ON ts.taxonomy_species_id = a.taxonomy_species_id
WHERE ts.name like 'Glycine%'
  AND NOT EXISTS (SELECT * FROM inventory I
WHERE i.accession_id = a.accession_id
AND is_distributable = 'Y' AND is_available = 'Y')
```

Determine Accessions for a Taxon w/out Distributable Inventory (“Non-Exists”)

```
SELECT accession_number_part1, accession_number_part2,accession_number_part3
FROM accession a
INNER JOIN taxonomy_species ts ON a.taxonomy_species_id = ts.taxonomy_species_id
WHERE ts.name like 'Cicer%'
AND NOT EXISTS (SELECT * FROM inventory i
   WHERE a.accession_id = i.accession_id
      AND i.is_distributable = 'Y')
```

```
SELECT a.accession_number_part1, a.accession_number_part2, a.accession_number_part3,
plant_name, ts.name
FROM taxonomy_species ts
JOIN accession a ON ts.taxonomy_species_id = a.taxonomy_species_id
JOIN inventory i ON a.accession_id = i.accession_id
JOIN accession_inv_name invn ON invn.inventory_id = i.inventory_id
WHERE ts.name LIKE 'Sorghum%'
AND NOT EXISTS (SELECT * FROM inventory i
   WHERE a.accession_id = i.accession_id
      AND i.is_distributable = 'Y')
```

Determine Accessions at a Site Needing Backup

SQL lists all the accessions for the taxon **Nic%** that either have not been backed up or have only one backup location.

```
SELECT ts.name,
accession_number_part1 AS Prefix,
accession_number_part2 AS AccNumb,
accession_number_part3 AS Suffix,
is_backed_up,
CASE backup_location1_site_id WHEN 25 THEN 'TOB' WHEN 42 THEN 'NSSL' ELSE '' END AS BKUP1,
CASE backup_location2_site_id WHEN 25 THEN 'TOB' WHEN 42 THEN 'NSSL' ELSE '' END AS BKUP2

FROM accession a
INNER JOIN taxonomy_species ts ON a.taxonomy_species_id = ts.taxonomy_species_id
WHERE
ts.name like 'Nic%' AND (a.backup_location1_site_id IS NULL OR a.backup_location2_site_id IS NULL)
```

Determine Accessions with Multiple (Distributable & Available) Inventory Types

```
SELECT accession_id, accession_number_part1, accession_number_part2, accession_number_part3
FROM accession a
WHERE a.status_code = 'ACTIVE' AND a.is_web_visible = 'Y'
AND (SELECT COUNT(distinct form_type_code) FROM inventory WHERE accession_id = a.accession_id
AND is_distributable = 'Y' AND is_available = 'Y') > 1
```

Crop-Related

Calculating Most Recent Viability

```
SELECT MAX(vci.percent_viable) as max_percent_viable, a.accession_id
FROM accession a
INNER JOIN taxonomy_species ts ON ts.taxonomy_species_id = a.taxonomy_species_id
INNER JOIN cooperator c ON c.cooperator_id = a.owned_by
INNER JOIN inventory i ON i.accession_id = a.accession_id
INNER JOIN vc_inventory vci ON vci.inventory_id = i.inventory_id
WHERE i.availability_status_code='AVAIL'
AND ts.name like 'Cucurbita%'
AND a.status_code='ACTIVE'
AND c.site_id=22
GROUP BY a.accession_id
HAVING MAX(vci.percent_viable) < 60
```

List Traits for a Crop

```
SELECT crop.name AS Crop, ct.coded_name
FROM   crop
    JOIN crop_trait ct
        ON crop.crop_id = ct.crop_id

/* remove the following two lines to see all traits; as is, there must be observations recorded for the
trait */
    JOIN crop_trait_observation cto
        ON ct.crop_trait_id = cto.crop_trait_id

/* Remove comments and edit red text for a specific crop:
WHERE crop.name = 'Alfalfa'
*/
GROUP  BY crop.name, ct.coded_name
ORDER  BY crop.name, ct.coded_name
```

List Accessions That Have Observations for a Specified Trait

```
SELECT crop.name AS Crop, ct.coded_name, cto.inventory_id, inv.accession_id,
a.accession_number_part1, a.accession_number_part2, a.accession_number_part3

FROM   crop
    JOIN crop_trait ct
        ON crop.crop_id = ct.crop_id
    JOIN crop_trait_observation cto
        ON ct.crop_trait_id = cto.crop_trait_id
    JOIN inventory inv
        ON inv.inventory_id = cto.inventory_id
    JOIN accession a
        ON a.accession_id = inv.accession_id
    JOIN crop_trait_code ctc
        ON ct.crop_trait_id =ctc.crop_trait_id
WHERE crop.name = 'Phaseolus'
AND
ct.coded_name = 'LEAF SHAPE'

GROUP  BY crop.name, ct.coded_name, cto.inventory_id, inv.accession_id, a.accession_number_part1,
a.accession_number_part2, a.accession_number_part3

ORDER  BY crop.name, ct.coded_name
```

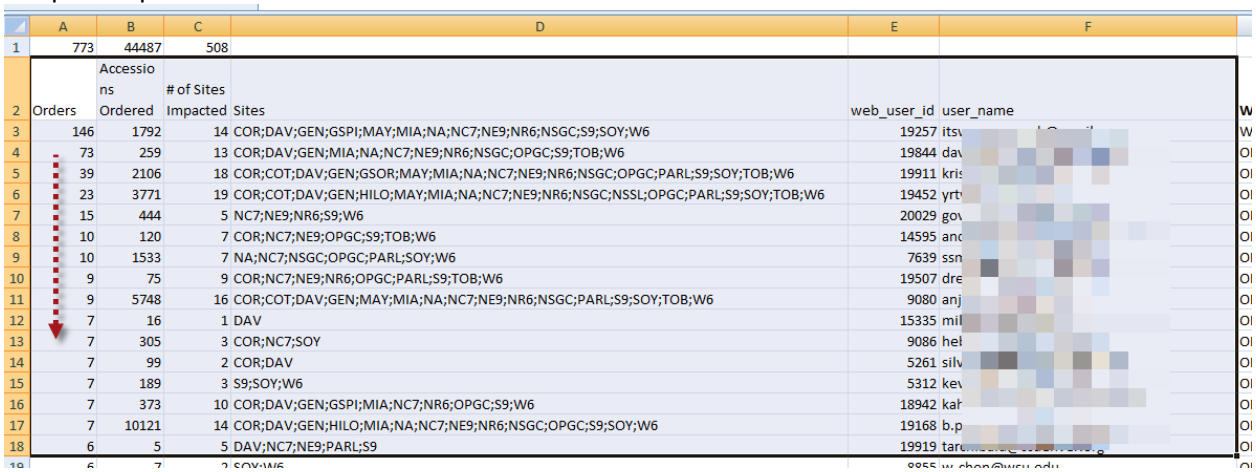
Cooperator & Order-Related

Display List of Repeat Requestors from the Recent Quarter

To determine requesters who have been making multiple orders in the previous 90 days, run the following SQL query. It counts orders, items, and number of sites impacted for the highest repeat customers over the previous 90 day period.

```
DECLARE @days int = 90;
WITH recentWu AS (
SELECT created_by AS web_user_id, count(*) AS worCount FROM web_order_request
WHERE datediff(Day, owned_date, getutcdate()) < @days
GROUP BY created_by
)
SELECT
(SELECT count(*) FROM web_order_request WHERE created_by = wu.web_user_id AND datediff(Day,
owned_date, getutcdate()) < @days) AS NumOfOrders
,(SELECT count(*) FROM web_order_request wor JOIN web_order_request_item wori ON
wori.web_order_request_id = wor.web_order_request_id
    WHERE wor.created_by = wu.web_user_id AND datediff(Day, wor.owned_date, getutcdate()) <
@days) AS Items_Requested
,(SELECT count(distinct c.site_id)
    FROM web_order_request wor
    JOIN web_order_request_item wori ON wori.web_order_request_id = wor.web_order_request_id
    JOIN accession a ON a.accession_id = wori.accession_id
    JOIN cooperator c ON c.cooperator_id = a.owned_by
      JOIN site s ON s.site_id = c.site_id
    WHERE wor.created_by = wu.web_user_id AND datediff(Day, wor.owned_date, getutcdate()) <
@days) AS Sites
,stuff((SELECT distinct ';' + s.site_short_name
    FROM web_order_request wor
    JOIN web_order_request_item wori ON wori.web_order_request_id = wor.web_order_request_id
    JOIN accession a ON a.accession_id = wori.accession_id
    JOIN cooperator c ON c.cooperator_id = a.owned_by
      JOIN site s ON s.site_id = c.site_id
    WHERE wor.created_by = wu.web_user_id AND datediff(Day, wor.owned_date, getutcdate()) <
@days
      FOR XML PATH('')),1,1,'')  AS Sites_Impacted
,wu.web_user_id, wu.user_name
FROM recentWu JOIN web_user wu ON wu.web_user_id = recentWu.web_user_id
WHERE worCount > 2
ORDER BY 2 DESC
```

Sample Output:



	A	B	C	D	E	F
1	773	44487	508			
2	Orders	Accessions Ordered	# of Sites Impacted	Sites	web_user_id	user_name
3	146	1792	14	COR;DAV;GEN;GSPI;MAY;MIA;NA;NC7;NE9;NR6;NSGC;S9;SOY;W6	19257	itsv
4	73	259	13	COR;DAV;GEN;MIA;NA;NC7;NE9;NR6;NSGC;OPGC;S9;TOB;W6	19844	dav
5	39	2106	18	COR;COT;DAV;GEN;GSOR;MAY;MIA;NA;NC7;NE9;NR6;NSGC;OPGC;PARL;S9;SOY;TOB;W6	19911	kris
6	23	3771	19	COR;COT;DAV;GEN;HILO;MAY;MIA;NA;NC7;NE9;NR6;NSGC;NSSL;OPGC;PARL;S9;SOY;TOB;W6	19452	yrt
7	15	444	5	NC7;NE9;NR6;S9;W6	20029	gov
8	10	120	7	COR;NC7;NE9;OPGC;S9;TOB;W6	14595	anc
9	10	1533	7	NA;NC7;NSGC;OPGC;PARL;SOY;W6	7639	ssn
10	9	75	9	COR;NC7;NE9;NR6;OPGC;PARL;S9;TOB;W6	19507	dre
11	9	5748	16	COR;COT;DAV;GEN;MAY;MIA;NA;NC7;NE9;NR6;NSGC;PARL;S9;SOY;TOB;W6	9080	anj
12	7	16	1	DAV	15335	mil
13	7	305	3	COR;NC7;SOY	9086	hel
14	7	99	2	COR;DAV	5261	silv
15	7	189	3	S9;SOY;W6	5312	kev
16	7	373	10	COR;DAV;GEN;GSPI;MIA;NC7;NR6;OPGC;S9;W6	18942	kah
17	7	10121	14	COR;DAV;GEN;HILO;MIA;NA;NC7;NE9;NR6;NSGC;OPGC;S9;SOY;W6	19168	b.p
18	6	5	5	DAV;NC7;NE9;PARL;S9	19919	tar

Two of the leading requesters ordered 146 orders and 10121 items…

You can then use the user_name (email address) to search in other ways, using the various SQL queries in this document.

In the output above, the email addresses have been intentionally blurred, but when you run the SQL, you will see the requestors' email addresses.

Using the following SQL, you can search by an email address to list the orders that were requested by the same user. In the second example below, the site code was added to the SQL and multiple email addresses were included:

```
SELECT wor.web_order_request_id, o.order_request_id, wor.ordered_date, wu.user_name,
wc.first_name, wc.last_name, tg.genus_name, s.site_short_name, ori.status_code, count(*) AS items
FROM web_order_request wor
JOIN web_cooperator wc
    ON wor.web_cooperator_id = wc.web_cooperator_id
JOIN web_user wu
    ON wor.created_by = wu.web_user_id
JOIN web_order_request_item wori
    ON wor.web_order_request_id = wori.web_order_request_id
JOIN accession a
    ON wori.accession_id = a.accession_id
JOIN cooperator c
    ON a.owned_by = c.cooperator_id
JOIN site s
    ON c.site_id = s.site_id
JOIN taxonomy_species ts
     ON a.taxonomy_species_id = ts.taxonomy_species_id
JOIN taxonomy_genus tg
    ON ts.taxonomy_genus_id = tg.taxonomy_genus_id
LEFT JOIN order_request o
```

```
	ON wor.web_order_request_id = o.web_order_request_id
LEFT JOIN order_request_item ori
	ON o.order_request_id = ori.order_request_id
	AND ori.inventory_id IN (SELECT inventory_id FROM inventory WHERE accession_id =
a.accession_id)

WHERE wc.email='email_address'

GROUP BY wor.web_order_request_id, o.order_request_id, wor.ordered_date, wu.user_name,
wc.first_name, wc.last_name, tg.genus_name, ori.status_code, s.site_short_name
ORDER  BY site_short_name, web_order_request_id
```

_____

```
SELECT wor.web_order_request_id, o.order_request_id, wor.ordered_date, wu.user_name,
wc.first_name, wc.last_name, tg.genus_name, s.site_short_name, ori.status_code, count(*) AS items
FROM web_order_request wor
JOIN web_cooperator wc
	ON wor.web_cooperator_id = wc.web_cooperator_id
JOIN web_user wu
	ON wor.created_by = wu.web_user_id
JOIN web_order_request_item wori
	ON wor.web_order_request_id = wori.web_order_request_id
JOIN accession a
	ON wori.accession_id = a.accession_id
JOIN cooperator c
	ON a.owned_by = c.cooperator_id
JOIN site s
	ON c.site_id = s.site_id
JOIN taxonomy_species ts
	 ON a.taxonomy_species_id = ts.taxonomy_species_id
JOIN taxonomy_genus tg
	ON ts.taxonomy_genus_id = tg.taxonomy_genus_id
LEFT JOIN order_request o
	ON wor.web_order_request_id = o.web_order_request_id
LEFT JOIN order_request_item ori
	ON o.order_request_id = ori.order_request_id
	AND ori.inventory_id IN (SELECT inventory_id FROM inventory WHERE accession_id =
a.accession_id)

WHERE site_short_name='NA'   AND
(
wc.email='wmnswrk@gmail.com'
OR wc.email='dvdjhn@yahoo.com'
)

GROUP BY wor.web_order_request_id, o.order_request_id, wor.ordered_date, wu.user_name,
wc.first_name, wc.last_name, tg.genus_name, ori.status_code, s.site_short_name
```

```
ORDER  BY site_short_name, web_order_request_id
```

Alternatively, IN could have been used, such as

```
…
WHERE site_short_name='NA'  AND
wc.email in (
mwrk@gmail.com,
dvdhntl@yahoo.com,
…
krwlls@yahoo.com
)
…
```

Determine Web Orders which have had multiple Inventory items requested

(for the same accession)

Beginning in Server Release 2.1.0, it is possible to make requests by inventory (rather than by accession). Hence a requestor can now request multiple items for an accession that has more than one distributable form.

```
SELECT DISTINCT web_order_request_id, CONVERT(date, created_date) AS date
FROM web_order_request_item
WHERE created_date > '2021-06-21'
AND accession_id IN (SELECT accession_id
FROM accession a
WHERE (SELECT count(*) FROM inventory WHERE accession_id = a.accession_id and is_distributable =
'Y' AND is_available = 'Y') > 1)
```

List Web Orders for a Requestor (by requestor's email address or name)

```
SELECT wor.web_order_request_id, wor.ordered_date, wu.user_name, wc.last_name, wc.first_name
FROM web_order_request wor
JOIN web_cooperator wc
      ON wor.web_cooperator_id = wc.web_cooperator_id
JOIN web_user wu
      ON wor.owned_by = wu.web_user_id

WHERE  wu.user_name = 'abc@ualberta.ca'

/*  WHERE wc.last_name = 'reisinger'  AND wc.first_name = 'martin'     */
/* to search by names, use the above line
/* text between the slash asterisk and the asterisk slash are comments */
/* use either WHERE clause above, based on preference */

GROUP BY wor.web_order_request_id,  wor.ordered_date, wu.user_name, wc.first_name,
wc.last_name
```

Related Web Orders and Orders

```
WITH bystat AS (
SELECT ori.order_request_id, s.site_short_name, count(*) AS item_count
  , COALESCE(ori.status_code,wori.status_code) AS status
  , CONVERT(nvarchar(10), wor.ordered_date) AS date
  , CONCAT (wor.web_order_request_id, ' ', wc.first_name, ' ', wc.last_name, ' ', wu.user_name) AS
web_order
FROM web_order_request wor
JOIN web_cooperator wc ON wor.web_cooperator_id = wc.web_cooperator_id
JOIN web_user wu ON wor.owned_by = wu.web_user_id
JOIN web_order_request_item wori ON wor.web_order_request_id = wori.web_order_request_id
LEFT JOIN order_request_item ori ON ori.web_order_request_item_id =
wori.web_order_request_item_id
JOIN accession a ON wori.accession_id = a.accession_id
JOIN cooperator c ON a.owned_by = c.cooperator_id
JOIN site s ON c.site_id = s.site_id
/* list of web_order_request_IDs*/
WHERE wor.web_order_request_id IN ( 49474, 49481 )
/*  WHERE wu.user_name = 'mar@rrginc.com'  *

GROUP BY wor.web_order_request_id, ori.order_request_id, wor.ordered_date, wu.user_name,
wc.first_name, wc.last_name, s.site_short_name, COALESCE(ori.status_code,wori.status_code)
)
```

_____

the above query replaced:

```
SELECT wor.web_order_request_id, o.order_request_id, wor.ordered_date, wu.user_name,
wc.last_name, wc.first_name
FROM web_order_request wor
JOIN web_cooperator wc
   ON wor.web_cooperator_id = wc.web_cooperator_id
JOIN web_user wu
   ON wor.owned_by = wu.web_user_id
LEFT JOIN order_request o
   ON wor.web_order_request_id = o.web_order_request_id
JOIN web_order_request_item wori
   ON wor.web_order_request_id = wori.web_order_request_id
JOIN accession a
   ON wori.accession_id = a.accession_id
JOIN cooperator c
   ON a.owned_by = c.cooperator_id
JOIN site s
   ON c.site_id = s.site_id
/* Edit date and site
WHERE wor.web_order_request_id = 23135 */
```

```
WHERE wu.user_name = 'mar@rrginc.com'

GROUP BY wor.web_order_request_id, o.order_request_id, wor.ordered_date, wu.user_name,
wc.first_name, wc.last_name
```

Determine the Highest Web Order Number*

```
SELECT MAX(web_order_request_id) AS Max Number
FROM web_order_request
```

* similar queries for any table can be made by using the correct ID field and corresponding table name (see the dictionary if needed [https://docs.google.com/spreadsheet/ccc?key=0AvdWZS-UqEE7dHFaRnRsR1RxOUx0em9KZmhNZTVlRnc&hl=en#gid=2]

List New Web Orders

This SQL searches for incoming *web orders* NEW to a site (specified in the WHERE clause). When a web order is a multi-site order, records in the resulting report may display an *Order* status of **ACCEPTED**, which indicates that at least one other site has already accepted the web order. (If the Order Status is **SUBMITTED**, then the weborder hasn't been accepted yet by any site.)

```
SELECT o.web_order_request_id AS WebOrdr#,FORMAT(ordered_date,'MM/dd/yyyy', 'en-US') AS
Ordered, First_name, Last_name, Organization, g.country_code as 'ShipTo', COUNT(*) AS Items,
intended_use_code AS Intended_Use, Intended_use_note,  Special_Instruction, o.status_code AS
Order_Status, o.web_cooperator_id AS WebCoop#

FROM web_order_request o
  JOIN web_order_request_item oi
        ON oi.web_order_request_id = o.web_order_request_id
  JOIN accession_view a
        ON a.accession_id=oi.accession_id
  JOIN web_cooperator c
        ON c.web_cooperator_id=o.web_cooperator_id
JOIN web_user wu
       ON wu.web_cooperator_id = c.web_cooperator_id
JOIN web_user_shipping_address wus
       ON wu.web_user_id = wus.web_user_id
  JOIN geography g
       ON wus.geography_id = g.geography_id

WHERE
a.site_short_name='NC7' AND oi.status_code = 'NEW'

GROUP BY
o.web_order_request_id, o.web_cooperator_id,ordered_date,
first_name,last_name,organization, intended_use_code,intended_use_note,
o.status_code,special_instruction, o.owned_date, g.country_code
```

Listing Web Orders Using SQL (…MIXED)

In the Public Website **Tools | Web Query** utility, use the following SQL for looking for incoming web orders with a status of **MIXED**

```
SELECT wor.web_order_request_id, o.order_request_id, wor.ordered_date, wu.user_name,
wc.last_name, wc.first_name
FROM web_order_request wor
JOIN web_cooperator wc
    ON wor.web_cooperator_id = wc.web_cooperator_id
JOIN web_user wu
    ON wor.created_by = wu.web_user_id
LEFT JOIN order_request o
    ON wor.web_order_request_id = o.web_order_request_id
JOIN web_order_request_item wori
    ON wor.web_order_request_id = wori.web_order_request_id
JOIN accession a
    ON wori.accession_id = a.accession_id
JOIN cooperator c
    ON a.owned_by = c.cooperator_id
JOIN site s
    ON c.site_id = s.site_id
/* Edit date and site */
WHERE wor.ordered_date >= '2017-07-30'
/* AND  wor.ordered_date < '2016-06-01' */
AND  s.site_short_name = 'GEN'
AND wor.status_code = 'MIXED'
/* or change to search by username (email address of requestor
WHERE  wu.user_name = 'requestor email address'  */

GROUP BY wor.web_order_request_id, o.order_request_id, wor.ordered_date, wu.user_name,
wc.first_name, wc.last_name
```

_____

Web Order Request NRR Statistics

The following query is a good example in general of a query joining many GG tables in order to get to the desired data. The rationale for this SQL was to obtain statistics by site for web orders that had been reviewed and then rejected by the NRR Committee. Also desired was a count of the number of web items involved in these WORs. Other columns were generated.

```
SELECT s.site_short_name
    , count(DISTINCT woac.web_order_request_action_id) as action_count
    , count(DISTINCT wor.web_order_request_id) AS web_order_count
    , count(DISTINCT wor.created_by) AS web_user_count
    , count(DISTINCT wori.web_order_request_item_id) AS web_item_count
    , count(DISTINCT a.accession_id) AS accession_count
FROM web_order_request_action woac
JOIN web_order_request wor ON wor.web_order_request_id = woac.web_order_request_id
JOIN web_order_request_item wori ON wori.web_order_request_id = wor.web_order_request_id
LEFT JOIN accession a ON a.accession_id = wori.accession_id
LEFT JOIN cooperator c ON c.cooperator_id = a.owned_by
LEFT JOIN site s ON s.site_id = c.site_id
WHERE woac.action_code = 'NRR_REJECT' AND wor.created_date > '8/6/2021'
GROUP BY s.site_short_name
```

_____

For a Web Order, List the Taxon, Curators, and Crops

When a web order number is provided to the WHERE clause, the resulting report lists all of the taxon that were ordered as well as their corresponding curator, crops, and inventory maintenance policy.

```
SELECT distinct
wor.web_order_request_id AS Web_Order, s.site_short_name AS Site, c.last_name AS Curator, ts.name
AS Taxon, crop.name as Crop_Name, a.accession_number_part1 AS Prefix, a.accession_number_part2
AS AccNumb, a.accession_number_part3  AS Suffix,
imp.maintenance_name AS Inv_Maintenance_Policy
FROM web_order_request wor
JOIN web_order_request_item wori ON wor.web_order_request_id = wori.web_order_request_id
JOIN accession a ON a.accession_id = wori.accession_id
JOIN cooperator c ON c.cooperator_id = a.owned_by
JOIN site s ON s.site_id = c.site_id
JOIN taxonomy_species ts ON ts.taxonomy_species_id = a.taxonomy_species_id
LEFT JOIN taxonomy_crop_map tcm ON tcm.taxonomy_species_id = ts.taxonomy_species_id AND
tcm.alternate_crop_name != 'N/A'
LEFT JOIN crop ON crop.crop_id = tcm.crop_id
LEFT JOIN inventory i ON i.accession_id = a.accession_id AND i.owned_by = a.owned_by AND
i.is_distributable = 'Y'
LEFT JOIN inventory_maint_policy imp ON imp.inventory_maint_policy_id = i.inventory_maint_policy_id
WHERE wor.web_order_request_id = 17648
ORDER BY Curator
```

List the Number of Orders Sent to a Location by Genus

```
SELECT o.country,o.state,SUBSTRING(ISNULL(taxon,'NULL'),1,charindex(' ',taxon)-1) AS species,
COUNT(distinct acid) AS items
FROM oi i join order_request_view o
ON o.order_request_id=i.orno join cooperator c
ON c.cooperator_id=o.final_recipient_cooperator_id

WHERE order_type_code IN ('DI','RP')
AND     o.site_short_name='GEN'
AND      i.status  IN ('INSPECT','PSHIP','SHIPPED')
AND     year(i.acted) > 2008

GROUP BY o.country,o.state,substring(isnull(taxon,'NULL'),1,charindex(' ',taxon)-1)
ORDER by 1,2
```

Finding Counts of Shipments by Genus for a Specified Time Range

```
SELECT taxon,year(status_date) year, count(*) AS CT
FROM order_request_item_view oi
WHERE  order_type in ('DI','RP')
AND status_code IN ('INSPECT','PSHIP','SHIPPED')
AND taxon LIKE 'Ficus%'
AND status_date BETWEEN '01/01/2009' AND '12/31/2009'
GROUP BY TAXON,year(status_date)
ORDER BY 1,2
```

Retrieve Order Information to Streamline Inventory Loading Activities

This SQL provides companion fields for loading new inventory when regenerating such as using the order note as the new inventory note, order item number as plot number, inventory as parent inventory, and Quantity Shipped to load in PLANTED action.

```
SELECT
ori.sequence_number AS SeqNumber,
imp.maintenance_name AS Inv_Maintenance_Policy,
ori.order_request_item_id AS OrderItemNum,
a.accession_number_part1 AS AccPrefix,
a.accession_number_part2 AS AccNo,
a.accession_number_part3 AS AccSufx,
ts.name AS TaxSpeciesName,
i.inventory_id AS InvID,
i.inventory_number_part1 AS InvPrefix,
i.inventory_number_part2 AS InvNumber,
i.inventory_number_part3 AS InvSuffix,
i.form_type_code AS InvType,
i.inventory_number_part1 + ' ' + CAST (i.inventory_number_part2 AS VARCHAR) + ' ' +
i.inventory_number_part3 +' ' + i.form_type_code AS Parent_Inventory,
i.accession_id AS Accession,
i.is_distributable AS IsDefault,
i.is_auto_deducted AS AutoDeduct,
i.is_available AS IsAvailable,
i.availability_status_code AS AvailabilityStatus,
i.pollination_method_code AS PollinationMethod,

ori.quantity_shipped AS QtyShipped,
ori.note AS OrderItemNote
/* FROM accession a */
FROM taxonomy_species ts
JOIN  accession a ON ts.taxonomy_species_id = a.taxonomy_species_id
LEFT JOIN inventory i
      ON i.accession_id = a.accession_id
LEFT JOIN inventory_maint_policy imp
      ON imp.inventory_maint_policy_id = i.inventory_maint_policy_id
JOIN order_request_item ori
      ON ori.inventory_id = i.inventory_id
JOIN order_request req
      ON req.order_request_id = ori.order_request_id

/* change the order request id number for each order */
WHERE
req.order_request_id = 275946
ORDER BY
/* Inv_Maintenance_Policy */
SeqNumber
```

Web User (PW Profile Exist?)

Answers if the web user account exists for the Public Website … date or name…

```
SELECT web_user_id, user_name, is_enabled
FROM  web_user
WHERE
user_name LIKE '%mar@rrg%'

/* created_date >= '06/10/2021' */
user_name LIKE '%marty.reisinger@usda.gov%' */
```

Display Basic Web Cooperator Info when eMail Address is Known

```
SELECT web_cooperator_id, first_name, last_name, primary_phone, organization, email, owned_date
FROM  web_cooperator
WHERE email = 'mar@rrginc.com'
```

Determine Shipping Address(es) by the Web Cooperator's eMail Address

```
SELECT last_name, first_name, email, primary_phone, organization, wu.web_user_id,
wus.address_name,
wus.address_line1 AS 'SHIPTO ADDRESSLINE1',
wus.address_line2 AS 'SHIPTO ADDRESS2',
wus.address_line3 AS 'SHIPTO ADDRESS3',
wus.city AS 'SHIP TO CITY', g.country_code

FROM  web_cooperator wc
JOIN web_user wu
      ON wu.web_cooperator_id = wc.web_cooperator_id
JOIN web_user_shipping_address wus
       ON wu.web_user_id = wus.web_user_id
JOIN geography g
       ON wus.geography_id = g.geography_id
/* substitute correct email address */
WHERE wc.email =' valid_email_address'
```

Find Related Orders When the WebOrder # is Known

This SQL will query much quicker than the next SQL example. The results here are much simpler: primarily the web order number and any order numbers related to the web order.

```
SELECT wor.web_order_request_id, o.order_request_id, wor.ordered_date, wu.user_name,
wc.last_name, wc.first_name

FROM web_order_request wor
JOIN web_cooperator wc
      ON wor.web_cooperator_id = wc.web_cooperator_id
JOIN web_user wu
      ON wor.created_by = wu.web_user_id
LEFT JOIN order_request o
      ON wor.web_order_request_id = o.web_order_request_id
/*  **web order number below**  */
WHERE wor.web_order_request_id = 17389
GROUP BY wor.web_order_request_id, o.order_request_id, wor.ordered_date, wu.user_name,
wc.first_name, wc.last_name
```

List History of Weborders & Orders by the Web Cooperator's email Address

or WebOrder Number

Use this SQL query when more information such as Taxonomy is needed; otherwise use the previous SQL to determine orders related to a web order.

```
SELECT wor.web_order_request_id, o.order_request_id, wor.ordered_date, wu.user_name,
wc.first_name, wc.last_name, tg.genus_name, s.site_short_name, ori.status_code, count(*) AS items
FROM web_order_request wor
JOIN web_cooperator wc
      ON wor.web_cooperator_id = wc.web_cooperator_id
JOIN web_user wu
      ON wor.owned_by = wu.web_user_id
JOIN web_order_request_item wori
      ON wor.web_order_request_id = wori.web_order_request_id
JOIN accession a
      ON wori.accession_id = a.accession_id
JOIN cooperator c
      ON a.owned_by = c.cooperator_id
JOIN site s
      ON c.site_id = s.site_id
JOIN taxonomy_species ts
       ON a.taxonomy_species_id = ts.taxonomy_species_id
JOIN taxonomy_genus tg
      ON ts.taxonomy_genus_id = tg.taxonomy_genus_id
LEFT JOIN order_request o
      ON wor.web_order_request_id = o.web_order_request_id
LEFT JOIN order_request_item ori
```

```
        ON o.order_request_id = ori.order_request_id
       AND ori.inventory_id IN (SELECT inventory_id FROM inventory WHERE accession_id =
a.accession_id)

WHERE wc.email='qwerty@gmail.com'
/* can restrict by site code as shown below; remove it not desired*/
AND s.site_short_name ='NA'

/* alternative search by web order number
WHERE wor.web_order_request_id = 17389
*/

WHERE wor.ordered_date >= '2016-04-30'
AND  wor.ordered_date < '2016-06-01'
AND  s.site_short_name = 'MIA'


GROUP BY wor.web_order_request_id, o.order_request_id, wor.ordered_date, wu.user_name,
wc.first_name, wc.last_name, tg.genus_name, ori.status_code, s.site_short_name
ORDER  BY web_order_request_id
```

Display Web Orders and Related Orders

(by date and site – or by requestor email address, or by web order #)

```
SELECT wor.web_order_request_id, o.order_request_id, wor.ordered_date, wu.user_name,
wc.last_name, wc.first_name

FROM web_order_request wor
JOIN web_cooperator wc
        ON wor.web_cooperator_id = wc.web_cooperator_id
JOIN web_user wu
        ON wor.created_by = wu.web_user_id
LEFT JOIN order_request o
        ON wor.web_order_request_id = o.web_order_request_id
JOIN web_order_request_item wori
        ON wor.web_order_request_id = wori.web_order_request_id
JOIN accession a
        ON wori.accession_id = a.accession_id
JOIN cooperator c
        ON a.owned_by = c.cooperator_id
JOIN site s
        ON c.site_id = s.site_id

/* Edit date and site */
WHERE wor.ordered_date >= '2016-04-30'
AND  wor.ordered_date < '2016-06-01'
AND  s.site_short_name = 'MIA'
```

```
/* or search by Web Order Number
WHERE wor.web_order_request_id = 23135
*/

/* or change to search by username (email address of requestor
WHERE  wu.user_name = 'requestor email address'
*/
GROUP BY wor.web_order_request_id, o.order_request_id, wor.ordered_date, wu.user_name,
wc.first_name, wc.last_name
```

Display the Weborder & Order (with Item Counts, Genus, and Shipping Status)

```
SELECT wor.web_order_request_id, o.order_request_id, wor.ordered_date, wu.user_name,
wc.first_name, wc.last_name, tg.genus_name, s.site_short_name, ori.status_code, count(*) AS items

FROM web_order_request wor
JOIN web_cooperator wc
      ON wor.web_cooperator_id = wc.web_cooperator_id
JOIN web_user wu
      ON wor.created_by = wu.web_user_id
JOIN web_order_request_item wori
      ON wor.web_order_request_id = wori.web_order_request_id
JOIN accession a
      ON wori.accession_id = a.accession_id
JOIN cooperator c
      ON a.owned_by = c.cooperator_id
JOIN site s
      ON c.site_id = s.site_id
JOIN taxonomy_species ts
       ON a.taxonomy_species_id = ts.taxonomy_species_id
JOIN taxonomy_genus tg
      ON ts.taxonomy_genus_id = tg.taxonomy_genus_id
LEFT JOIN order_request o
      ON wor.web_order_request_id = o.web_order_request_id
LEFT JOIN order_request_item ori
      ON o.order_request_id = ori.order_request_id
     AND ori.inventory_id IN (SELECT inventory_id FROM inventory WHERE accession_id =
a.accession_id)

WHERE wc.email=' requestor email address '

GROUP BY wor.web_order_request_id, o.order_request_id, wor.ordered_date, wu.user_name,
wc.first_name, wc.last_name, tg.genus_name, ori.status_code, s.site_short_name
```

Determine Curators Responsible For Web Order Items Across All Sites

```
SELECT distinct
wor.web_order_request_id AS Web_Order, s.site_short_name AS Site, c.last_name AS Curator, ts.name
AS Taxon, crop.name as Crop_Name, a.accession_number_part1 AS Prefix, a.accession_number_part2
AS AccNumb, a.accession_number_part3  AS Suffix,
imp.maintenance_name AS Inv_Maintenance_Policy
FROM web_order_request wor
JOIN web_order_request_item wori ON wor.web_order_request_id = wori.web_order_request_id
JOIN accession a ON a.accession_id = wori.accession_id
JOIN cooperator c ON c.cooperator_id = a.owned_by
JOIN site s ON s.site_id = c.site_id
JOIN taxonomy_species ts ON ts.taxonomy_species_id = a.taxonomy_species_id
LEFT JOIN taxonomy_crop_map tcm ON tcm.taxonomy_species_id = ts.taxonomy_species_id AND
tcm.alternate_crop_name != 'N/A'
LEFT JOIN crop ON crop.crop_id = tcm.crop_id
LEFT JOIN inventory i ON i.accession_id = a.accession_id AND i.owned_by = a.owned_by AND
i.is_distributable = 'Y'
LEFT JOIN inventory_maint_policy imp ON imp.inventory_maint_policy_id = i.inventory_maint_policy_id
WHERE wor.web_order_request_id = 19849
/* To get a site specific list of the curators involved in a web order, substitute the above WHERE clause
for this one:
WHERE wor.web_order_request_id = 19849 AND s.site_short_name IN ('NC7')
*/
ORDER BY Curator
```

_____

Duplicate Cooperators (Finding)

```
-- only consider active cooperators with a last name and address line 1
WITH candidate AS (
	SELECT * FROM cooperator WHERE current_cooperator_id = cooperator_id
	AND TRIM(COALESCE(last_name, '')) != '' AND TRIM(COALESCE(address_line1, ''))!= ''
), pairs AS (
	SELECT c1.cooperator_id AS id1, c2.cooperator_id AS id2
	  FROM candidate c1 JOIN candidate c2 ON c2.cooperator_id != c1.cooperator_id
	 WHERE c1.last_name = c2.last_name AND c1.first_name = c2.first_name
	UNION SELECT c1.cooperator_id AS id1, c2.cooperator_id AS id2
	  FROM candidate c1 JOIN candidate c2 ON c2.cooperator_id != c1.cooperator_id
	 WHERE c1.last_name = c2.first_name AND c1.first_name = c2.last_name
)
SELECT * INTO #pairs FROM pairs;

-- Create temp table that merges the cooperator address into a single field
SELECT first_name, last_name, cooperator_id, city, email
	, (SELECT s.site_short_name FROM cooperator cz JOIN site s ON s.site_id = cz.site_id WHERE
cz.cooperator_id = c.owned_by) AS site
	, CONCAT (first_name+' ', last_name+' ', organization+' '
	, address_line1+' ', address_line2+' ', address_line3+' ', city+' '
	, CASE WHEN postal_index LIKE '%-%' AND country_code = 'USA'
		THEN LEFT(postal_index, 5)+' ' ELSE postal_index + ' ' END
	, adm2+' ', adm1+' ', country_code) AS oaddr
	, CONVERT(nvarchar(1000), ' ') AS caddr, 0 AS acnt, 0 AS ncnt
INTO #ca
FROM cooperator c JOIN geography g ON g.geography_id = c.geography_id
WHERE cooperator_id IN (SELECT id1 FROM #pairs)

-- Clean address with abbreviations
UPDATE #ca SET caddr = REPLACE(REPLACE(REPLACE(REPLACE(REPLACE(
	REPLACE(REPLACE(REPLACE(REPLACE(REPLACE(REPLACE(REPLACE(REPLACE(
	REPLACE(REPLACE(REPLACE(REPLACE(REPLACE(REPLACE(REPLACE(oaddr
	, ', ', ' '), '. ', ' '), 'university', 'UNIV'), 'department', 'DEPT'), ' P.O ', ' PO ')
	, ' street ', ' ST '), ' road ', ' RD '), ' drive ', ' DR '), ' avenue ', ' AVE ')
	, ' Lane ', ' LN '), ' Court ', ' CT '), ' Highway ', ' HWY '), ' Parkway ', ' PKWY ')
	, ' North ', ' N '), ' East ', ' E '), ' South ', ' S '), ' WEST ', ' W ')
	, '  ', ' '), '  ', ' '), '  ', ' ')

-- Count unique alpha token/word in address
UPDATE #ca SET acnt = (SELECT count(distinct TRIM(token.value))
    FROM cooperator c CROSS APPLY STRING_SPLIT(caddr, ' ') AS token
    WHERE c.cooperator_id = #ca.cooperator_id AND token.value NOT LIKE '%[0-9]%')
FROM #ca;

-- Count unique numeric token/word in address
```

```
UPDATE #ca SET ncnt = (SELECT count(distinct TRIM(token.value))
      FROM cooperator c CROSS APPLY STRING_SPLIT(caddr, ' ') AS token
      WHERE c.cooperator_id = #ca.cooperator_id AND token.value LIKE '%[0-9]%')
FROM #ca;

DELETE FROM #pairs WHERE id1 <= id2;

-- Count the matchhing words in pairs of addresses
WITH cc AS (SELECT u2.last_name, u2.first_name
          , u1.cooperator_id AS id1, u1.site AS site1
          , u2.cooperator_id AS id2, u2.site AS site2
          --, u1.acnt AS acnt1, u2.acnt AS acnt2, u1.ncnt AS ncnt1, u2.ncnt AS ncnt2
          , u1.acnt + u2.acnt - 2*(SELECT count(DISTINCT t1.value) FROM #ca d
                    CROSS APPLY STRING_SPLIT(d.caddr, ' ') AS t1
                    CROSS APPLY STRING_SPLIT(u2.caddr, ' ') AS t2
                    WHERE d.cooperator_id = u1.cooperator_id AND t1.value = t2.value AND t1.value NOT
LIKE '%[0-9]%'
          ) as amiss
          , u1.ncnt + u2.ncnt - 2*(SELECT count(DISTINCT t1.value) FROM #ca d
                    CROSS APPLY STRING_SPLIT(d.caddr, ' ') AS t1
                    CROSS APPLY STRING_SPLIT(u2.caddr, ' ') AS t2
                    WHERE d.cooperator_id = u1.cooperator_id AND t1.value = t2.value AND t1.value LIKE
'%[0-9]%'
          ) as nmiss
          , u2.oaddr AS addr1, u1.oaddr AS addr2
          FROM #pairs p
          JOIN #ca u1 ON u1.cooperator_id = p.id1
          JOIN #ca u2 ON u2.cooperator_id = p.id2
)
SELECT * FROM cc WHERE nmiss + amiss between 0 and 2
/* nmiss means numeric differences; amiss indicates alpha differences

*/
```

GRIN-Global System Information

Display Current Data Dictionary

```
SELECT
sd. database_area_code AS Area,
sd.dataview_name AS Dataview_Name,
sdf.sort_order AS Sort_order,
st.table_name AS Table_Name,
sdf.field_name AS Field_Name,
stfl.title AS Title,
stfl.description AS Description
  FROM sys_dataview sd
  JOIN sys_dataview_field sdf
        ON sd.sys_dataview_id = sdf.sys_dataview_id
  LEFT JOIN sys_table_field stf
        ON sdf.sys_table_field_id = stf.sys_table_field_id
  LEFT JOIN sys_table st
        ON stf.sys_table_id = st.sys_table_id
  LEFT JOIN sys_table_field_lang stfl
        ON stf.sys_table_field_id = stfl.sys_table_field_id
        AND  stfl.sys_lang_id = 1
WHERE
  sd.category_code = 'Client'
ORDER BY sd.dataview_name, sdf.sort_order
```

Display Table and Field Name Information

```
SELECT table_name, column_name, is_nullable, data_type, character_maximum_length
FROM information_schema.columns
/* to see all tables, omit the WHERE clause */
WHERE table_name = 'inventory'
ORDER BY table_name, ordinal_position
```

Display Table and Name In formation with Field Sizes & Types

```
USE gringlobal;
SELECT
        st.table_name,
        stf.field_name,
        stfl.title as field_title,
        stfl.description as field_description,
        stf.max_length,
        stf.field_type,
```

```
/*      sl2.title as language_title,
        stfl.sys_table_field_lang_id,
*/
stf.is_primary_key,
stf.is_foreign_key,
stf.is_nullable

FROM
        sys_table st
        JOIN sys_table_field stf
                ON st.sys_table_id = stf.sys_table_id
        LEFT JOIN sys_table_field_lang stfl
                ON stf.sys_table_field_id = stfl.sys_table_field_id
        LEFT JOIN sys_lang sl2
                ON stfl.sys_lang_id = sl2.sys_lang_id
        LEFT JOIN sys_table_lang stl
                ON st.sys_table_id = stl.sys_table_id
        LEFT JOIN sys_lang sl1
                ON stl.sys_lang_id = sl1.sys_lang_id
WHERE
sl2.sys_lang_id = 1

ORDER BY
        st.table_name, stf.field_ordinal
```

Locate Lowercase Data

The database typically supporting the GG database is Microsoft SQL Server. It is typically configured that it does not distinguish upper- from lower-case characters. You cannot search for lowercase characters in the Search Tool, but you can in a SQL query. See the example below:

```
SELECT * FROM accession WHERE accession_number_part3  = 'rrg'
COLLATE SQL_Latin1_General_CP1_CS_AS
```

Display *Table* Field Names, Data Type, Lengths, and Column Headings (Schema)

SQL displays the database table and field names as well as the field titles (column headings) and descriptions used by the Curator Tool and Public Website. Note that the sl2.sys_lang_id = 1 i the WHERE clause indicates the English language; substitute the respective value to display another language.

```
USE gringlobal;
SELECT
        st.table_name,
        stf.field_name,
        stf.max_length,
        stf.field_type,
/*      sl2.title as language_title,
        stfl.sys_table_field_lang_id,
*/
        stfl.title as field_title,
        stfl.description as field_description
FROM
        sys_table st
        JOIN sys_table_field stf
                ON st.sys_table_id = stf.sys_table_id
        LEFT JOIN sys_table_field_lang stfl
                ON stf.sys_table_field_id = stfl.sys_table_field_id
        LEFT JOIN sys_lang sl2
                ON stfl.sys_lang_id = sl2.sys_lang_id
        LEFT JOIN sys_table_lang stl
                ON st.sys_table_id = stl.sys_table_id
        LEFT JOIN sys_lang sl1
                ON stl.sys_lang_id = sl1.sys_lang_id
WHERE
sl2.sys_lang_id = 1
/*  for one specific table – example here: accession inventory name,
or a family of related dataviews, such as CROP
AND  st.table_name = 'accession_inv_name'
AND  st.table_name  LIKE 'crop%'
*/
ORDER BY
        st.table_name, stf.field_ordinal
```

Owner parent relationships in GG

```
SELECT ost.table_name AS parent_table, st.table_name AS child_table
FROM sys_table_relationship str
JOIN sys_table_field stf ON stf.sys_table_field_id = str.sys_table_field_id
JOIN sys_table st ON st.sys_table_id = stf.sys_table_id
JOIN sys_table_field ostf ON ostf.sys_table_field_id = str.other_table_field_id
JOIN sys_table ost ON ost.sys_table_id = ostf.sys_table_id
WHERE str.relationship_type_tag = 'OWNER_PARENT'
ORDER BY 1,2
```

parent_table	child_table
accession	accession_action
accession	accession_ipr
accession	accession_pedigree
accession	accession_quarantine
accession	accession_source
accession_inv_group	accession_inv_group_attach
accession_inv_group	accession_inv_group_map
accession_source	accession_source_map
crop	genetic_marker
crop_trait	crop_trait_code
inventory	accession_inv_annotation
inventory	accession_inv_attach
inventory	accession_inv_name
inventory	accession_inv_voucher
inventory	crop_trait_observation
inventory	genetic_observation
inventory	geneva_site_inventory
inventory	inventory_action
inventory	inventory_quality_status
inventory	inventory_viability
inventory	nc7_site_inventory
inventory	ne9_site_inventory
inventory	nssl_site_inventory
inventory	opgc_site_inventory
inventory	parl_site_inventory
inventory	s9_site_inventory
inventory	w6_site_inventory
inventory_maint_policy	inventory
method	method_attach
order_request	order_request_action
order_request	order_request_attach
order_request	order_request_item
taxonomy_family	taxonomy_genus

Display *Table* Field Names (and their Dataview Areas)

```
SELECT
sdf.field_name,
sd.database_area_code, sd.dataview_name,
     st.table_name, stfl.title,  stfl.description
  FROM sys_dataview sd
  JOIN sys_dataview_field sdf
        ON sd.sys_dataview_id = sdf.sys_dataview_id
  LEFT JOIN sys_table_field stf
        ON sdf.sys_table_field_id = stf.sys_table_field_id
  LEFT JOIN sys_table st
        ON stf.sys_table_id = st.sys_table_id
  LEFT JOIN sys_table_field_lang stfl
        ON stf.sys_table_field_id = stfl.sys_table_field_id AND  stfl.sys_lang_id = 1
WHERE
  sd.category_code = 'Client'
ORDER BY sdf.field_name, sd.dataview_name
```

Display *Dataview* Field Names

```
/*
(Primarily used by GG administrators; displays data stored in sys_dataview_field_lang)
*/
USE gringlobal;
SELECT
          sd.dataview_name,
          sdl.title AS dataview_title,
          sdf.field_name,
          sl.title AS language_title,
          sdfl.title AS field_title,
          sdfl.description AS field_description
FROM
          sys_dataview sd
         JOIN sys_dataview_field sdf
                   ON sd.sys_dataview_id = sdf.sys_dataview_id
          LEFT JOIN sys_dataview_field_lang sdfl
                   ON sdf.sys_dataview_field_id = sdfl.sys_dataview_field_id
          LEFT JOIN sys_lang sl
                   ON sdfl.sys_lang_id = sl.sys_lang_id
          LEFT JOIN sys_dataview_lang sdl
                   ON sd.sys_dataview_id = sdl.sys_dataview_id AND sdl.sys_lang_id = sdfl.sys_lang_id
WHERE
sl.sys_lang_id = 1 AND sd.category_code IN ('Client')
ORDER BY
          sd.dataview_name, sdf.sort_order
```

Determine Date when Fields, Dataviews, & Tables were Introduced

Handy for administrators if a question comes up about when a field was introduced into the database.

Database Fields

```
SELECT table_name, field_name, stf.owned_date
  FROM sys_table st
  JOIN sys_table_field stf ON stf.sys_table_id = st.sys_table_id
ORDER BY stf.owned_date DESC
```

Database Tables

```
SELECT
    OBJECT_NAME(sc.object_id) AS [table]
    ,sc.name as [column]
    ,so.modify_date
    ,so.create_date
  FROM sys.columns sc
  JOIN sys.objects so ON sc.object_id = so.object_id
WHERE OBJECT_SCHEMA_NAME(sc.object_id) = 'dbo'
 AND sc.object_id NOT IN (SELECT object_id FROM sys.views)
ORDER BY so.create_date DESC
```

Dataviews

```
SELECT dataview_name, category_code, owned_date
FROM sys_dataview
ORDER BY owned_date DESC
```

Display Codes and Code Groups

Code Groups Only

```
SELECT DISTINCT
    cv.group_name as group_name
  FROM
  code_value cv
```

Groups & Codes

```
SELECT
  cv.code_value_id,
  cv.group_name as group_name,
  cv.value,  cvl.title, cvl.description
FROM
  code_value cv
  LEFT JOIN code_value_lang cvl ON cv.code_value_id = cvl.code_value_id
                          AND cvl.sys_lang_id = 1
/*  use the WHERE clause to search for a specific code – examples:  */
/*  WHERE  cv.group_name  = ' ORDER_REQUEST_ITEM_STATUS'  */
/*  WHERE  cv.group_name  LIKE  'germplasm%'   */

ORDER BY  group_name,   cv.value
```

List the Unique Indexes and Their Fields

```
SELECT t.name as tablename, i.name as indexname, tc.name as fieldname
FROM sys.tables t
        JOIN sys.indexes i
            ON t.object_id = i.object_id
        JOIN sys.index_columns ic
            ON i.object_id = ic.object_id AND i.index_id = ic.index_id
        JOIN sys.columns tc
            ON i.object_id = tc.object_id AND ic.column_id = tc.column_id
WHERE i.is_unique = 1
ORDER BY t.name, i.name, ic.index_column_id
```

Default Values for the Coded Fields

In the CT, whenever there is a dropdown, that file is using codes from a code group. Roll the mouse over the column heading to determine which code group.) The following SQL is used to list the default values for fields using these codes.

```
SELECT table_name, field_name, default_value
FROM sys_table st
JOIN sys_table_field stf ON stf.sys_table_id = st.sys_table_id
WHERE default_value IS NOT NULL AND default_value != '{DBNull.Value}' AND default_value != ''
ORDER BY 1,2
```

Display Source Habitat Descriptors Codes and Code Groups

Source Habitat Descriptors

```
SELECT DISTINCT
    sd.coded_name AS "S H Descriptors"
  FROM
  source_descriptor sd
```

Source Habitat Descriptors & Codes

```
SELECT
coded_name, sdl.title AS 'Desc-Title', sdc.code, sdcl.title, sdcl.description
FROM
source_descriptor sd
JOIN source_descriptor_code sdc
ON sdc.source_descriptor_id = sd.source_descriptor_id
JOIN source_descriptor_lang sdl
ON sdl.source_descriptor_id = sd.source_descriptor_id
JOIN source_descriptor_code_lang sdcl
ON sdcl.source_descriptor_code_id = sdc.source_descriptor_code_id
```

Display the Current Client Dataviews (with all of their fields)

This SQL pulls the Client dataview and table fields. The extracted spreadsheet displays the dataviews, the dataview's Area, their fields, titles, and descriptions. It can be used as the basis for a dataview dictionary. In the example below, the language identifier is "1" (stfl.sys_lang_id = 1) Substitute the ID value of the desired language.

```
SELECT
sdf.field_name,
sd.dataview_name,
sd. database_area_code,
st.table_name,
stfl.title,
stfl.description
  FROM sys_dataview sd
  JOIN sys_dataview_field sdf
        ON sd.sys_dataview_id = sdf.sys_dataview_id
  LEFT JOIN sys_table_field stf
        ON sdf.sys_table_field_id = stf.sys_table_field_id
  LEFT JOIN sys_table st
        ON stf.sys_table_id = st.sys_table_id
  LEFT JOIN sys_table_field_lang stfl
        ON stf.sys_table_field_id = stfl.sys_table_field_id
        AND  stfl.sys_lang_id = 1
WHERE
  sd.category_code = 'Client'
ORDER BY sdf.field_name, sd.dataview_name
```

COUNT Number of records in a table

```
SELECT COUNT ( DISTINCT crop_trait_id )
AS  "Number of Trait Records"
FROM crop_trait
```

Determine the Number of traits per crop

```
SELECT COUNT (*) AS trait_count, name
FROM crop
JOIN crop_trait ct ON
ct.crop_id = crop.crop_id
GROUP BY crop.name
ORDER BY 1 DESC
```

Germination Results for the Distribution Lots of a Given Inventory Maintenance Policy

This query lets you look at the last germination result for the distribution lots of a given inventory maintenance policy. You can select the year. (This is a less than statement – it allows you look at distribution lots that need to be pulled for maintenance germination tests). In the example below: the user wants to know which distribution lots in the NC7-brassica sitecrop have not been germinated in the last ten years (they are due a maintenance germ).

```
SELECT i.inventory_number_part1, i.inventory_number_part2, i.inventory_number_part3,
i.form_type_code,
      iv.percent_viable, FORMAT(iv.tested_date, iv.tested_date_code) AS tested_date
  FROM inventory i
  JOIN inventory_viability iv ON iv.inventory_id = i.inventory_id
  JOIN inventory_maint_policy imp ON i.inventory_maint_policy_id = imp.inventory_maint_policy_id
  JOIN cooperator c ON c.cooperator_id = i.owned_by
WHERE NOT EXISTS (SELECT * FROM inventory_viability WHERE inventory_id = i.inventory_id AND
tested_date > iv.tested_date)
   AND c.site_id = 16
   AND i.is_distributable = 'Y'
   AND imp.maintenance_name = 'NC7-brassica'
   AND iv.tested_date < '2007'
```

Full Text Indexing – Indicate if it is On or Off

```
SELECT fulltextserviceproperty('IsFullTextInstalled')
As answer
/* 1 indicates Full Text Indexing is installed; 0 indicates that it is not) */
```

Display Indexed Fields (Fields with Full Text Search Indexes)

The following SQL displays a list of the full text indexes used with the database.

```
SELECT DISTINCT
   object_name(fic.[object_id]) AS table_name,
   [name]
FROM
   sys.fulltext_index_columns fic
   INNER JOIN sys.columns c
      ON c.[object_id] = fic.[object_id]
      AND c.[column_id] = fic.[column_id]
```

List the "Autofields" Used in the Search Box

```
SELECT table_name, field_name
  FROM sys_search_autofield ssa
  JOIN sys_table_field stf ON stf.sys_table_field_id = ssa.sys_table_field_id
  JOIN sys_table st ON st.sys_table_id = stf.sys_table_id
  ORDER BY 1,2
```

Find which Code Groups are used in which Fields:

```
SELECT stf.group_name, st.table_name, stf.field_name
FROM sys_table_field stf
JOIN sys_table st ON st.sys_table_id = stf.sys_table_id
WHERE group_name IS NOT NULL
ORDER BY stf.group_name
```

Produces results similar to:

group_name	table_name	field_name
ACCESSION_ACTION	accession_action	action_name_code
ACCESSION_LIFE_FORM	accession	life_form_code
ACCESSION_LIFE_FORM	taxonomy_species	life_form_code
ACCESSION_NAME_TYPE	accession_inv_name	category_code
ACCESSION_QUARANTINE_STATUS	accession_quarantine	progress_status_code
ACCESSION_QUARANTINE_TYPE	accession_quarantine	quarantine_type_code
ACCESSION_RESTRICTION_TYPE	accession_ipr	type_code
ACCESSION_SOURCE_HABITAT_TYPE	accession_source	acquisition_source_code
ACCESSION_SOURCE_TYPE	accession_source	source_type_code
ACCESSION_STATUS	accession	status_code
ANNOTATION_TYPE	accession_inv_annotation	annotation_type_code

The above tells you that the ACCESSION_RESTRICTION_TYPE group is used in the type_code field of the accession_ipr table. If you wanted to find if any of the codes were unused you could use SQL such as:

```
SELECT group_name, value
FROM code_value
WHERE group_name = 'ACCESSION_RESTRICTION_TYPE'
AND value NOT IN (SELECT DISTINCT type_code FROM accession_ipr)
```

You can even write a SQL statement that will generate SQL needed to check the codes:

```
SELECT
    'SELECT group_name, value FROM code_value WHERE group_name = '''
    + stf.group_name
    + ''' AND value NOT IN (SELECT DISTINCT ' + stf.field_name
    + ' FROM ' + st.table_name + ')'
        AS SQL
FROM sys_table_field stf
JOIN sys_table st ON st.sys_table_id = stf.sys_table_id
WHERE group_name IS NOT NULL
ORDER BY stf.group_name
```

Site & User Information

List the Site's CT User Accounts

```
SELECT user_name, first_name, last_name, su.created_date, su.cooperator_id
FROM sys_user su
JOIN cooperator c
ON c.cooperator_id = su.cooperator_id
WHERE c.site_id = 42
```

Note: A site ID can be determined by opening the Site dataview in the Curator Tool.

Display CT User Accounts

```
Select user_name, modified_date
FROM sys_user
WHERE user_name
LIKE 'jill%'
```

Determine if the PW Account is Enabled

```
Select user_name, wu.is_enabled, wu.modified_date
FROM web_user wu
JOIN web_cooperator wc
ON  wu.web_cooperator_id = wc.web_cooperator_id
WHERE wu.user_name
LIKE '%gayle%'
```

```
Select wc.last_name,  wc.modified_date
FROM web_cooperator wc
WHERE wc.email
LIKE '%therese.bengtsson@slu.se%'
```

Determine Which Permission Policies a User Has Been Included

Curator Tool users within the organization can create permission policies in which they typically indicate update and delete permission for specific records they own (or record types, such as accessions, inventory, order requests). When creating the policies, the user, via the Security Wizard, indicates which of their fellow users will be given these permissions. For more details on security policies and permissions, see https://www.grin-global.org/docs/gg_security.pdf .

```
SELECT su.user_name, sg.group_tag, sp.permission_tag, st.table_name, CONCAT(c.first_name, ' ',
c.last_name) AS owner
FROM sys_user su
JOIN sys_group_user_map sgum ON sgum.sys_user_id = su.sys_user_id
JOIN sys_group sg ON sg.sys_group_id = sgum.sys_group_id
JOIN sys_group_permission_map sgpm ON sgpm.sys_group_id = sg.sys_group_id
JOIN sys_permission sp ON sp.sys_permission_id = sgpm.sys_permission_id
LEFT JOIN sys_table st ON st.sys_table_id = sp.sys_table_id
JOIN cooperator c ON c.cooperator_id = sp.created_by
WHERE sp.owned_by != 48
  AND user_name LIKE 'brian%'
```

Determine What Sites have the MANAGE_SITE group

```
SELECT group_tag
FROM sys_group
WHERE owned_by = 48
AND group_tag  LIKE  'MANAGE_SITE%'
```

Determine what staff are included in the MANAGE_SITE group

```
SELECT su.user_name, sg.group_tag, sp.permission_tag, st.table_name, CONCAT(c.first_name, ' ',
c.last_name) AS owner
FROM sys_user su
JOIN sys_group_user_map sgum ON sgum.sys_user_id = su.sys_user_id
JOIN sys_group sg ON sg.sys_group_id = sgum.sys_group_id
JOIN sys_group_permission_map sgpm ON sgpm.sys_group_id = sg.sys_group_id
JOIN sys_permission sp ON sp.sys_permission_id = sgpm.sys_permission_id
LEFT JOIN sys_table st ON st.sys_table_id = sp.sys_table_id
JOIN cooperator c ON c.cooperator_id = su.cooperator_id
WHERE
sg.group_tag  LIKE  'MANAGE_SITE%'
AND c.site_id = 3
/* omit line above to display all sites/cooperators */
ORDER BY sp.permission_tag, c.last_name
```

Statistics

Determine which Web Orders have had multiple Inventory items requested

(for the same accession)

Beginning in Server Release 2.1.0, it is possible to make requests by inventory (rather than by accession). Hence a requestor can now request multiple items for an accession that has more than one distributable form.

```
SELECT DISTINCT web_order_request_id, CONVERT(date, created_date) AS date
FROM web_order_request_item
WHERE created_date > '2021-06-21'
AND accession_id IN (SELECT accession_id
FROM accession a
WHERE (SELECT count(*) FROM inventory WHERE accession_id = a.accession_id and is_distributable =
'Y' AND is_available = 'Y') > 1)
```

COUNT of species in database

Counts the number of species in the database. Counts any record (i.e. subspecific taxa such as cultivars, subspecies, botanical varieties etc.), as long as the record has a unique taxonomy species ID. Only counts active accessions in the database.

```
SELECT COUNT(distinct ts.taxonomy_species_id) AS species
FROM taxonomy_species ts
WHERE ts.taxonomy_species_id IN
  (select taxonomy_species_id FROM accession WHERE status_code = 'ACTIVE')
```

COUNT of accessions by family

```
SELECT count(*) AS a_count, family_name
FROM taxonomy_family tf
JOIN taxonomy_genus tg ON tg.taxonomy_family_id = tf.taxonomy_family_id
JOIN taxonomy_species ts ON ts.taxonomy_genus_id = tg.taxonomy_genus_id
JOIN accession a ON a.taxonomy_species_id = ts.taxonomy_species_id
GROUP BY family_name
ORDER BY family_name
```

COUNT of accessions by genus name

```
SELECT count(*) AS accessions_count, genus_name
FROM taxonomy_genus tg
JOIN taxonomy_species ts ON ts.taxonomy_genus_id = tg.taxonomy_genus_id
JOIN accession a ON a.taxonomy_species_id = ts.taxonomy_species_id
```

```
GROUP BY genus_name
ORDER BY genus_name
```

COUNT unique records in a table

```
SELECT COUNT ( DISTINCT accession_source_id )
AS "Number of source records"
FROM accession_source
```

COUNT of items shipped by state

```
SELECT
  g.adm1 AS State,
  COUNT(*) AS Count

FROM order_request_item ori
JOIN order_request o
       ON ori.order_request_id = o.order_request_id
JOIN cooperator own
       ON ori.owned_by = own.cooperator_id
JOIN site
       ON own.site_id = site.site_id
JOIN cooperator c
        ON o.final_recipient_cooperator_id = c.cooperator_id
JOIN geography g
        ON c.geography_id = g.geography_id

/*  Edit */
WHERE ori.status_date >= '1/1/2013' AND ori.status_date < '1/1/2014'
  AND site.site_short_name = 'W6'
  AND o.order_type_code = 'DI'
  AND ori.status_code = 'Shipped'
  AND g.country_code = 'USA'

GROUP BY g.adm1
ORDER BY g.adm1
```

COUNT of Items Shipped by Countries for a Date Range

```
SELECT cvl.title AS country, COUNT(*) AS Count

FROM order_request_item ori
JOIN order_request o
    ON ori.order_request_id = o.order_request_id
JOIN cooperator own
    ON ori.owned_by = own.cooperator_id
JOIN site
    ON own.site_id = site.site_id
JOIN cooperator c
    ON o.final_recipient_cooperator_id = c.cooperator_id
JOIN geography g
    ON c.geography_id = g.geography_id
LEFT JOIN code_value cv
    ON g.country_code = cv.value AND cv.group_name = 'GEOGRAPHY_COUNTRY_CODE'
LEFT JOIN code_value_lang cvl
    ON cv.code_value_id = cvl.code_value_id AND cvl.sys_lang_id = 1  -- English

/*  Edit */
WHERE ori.status_date >= '1/1/2015' AND ori.status_date < '1/1/2016'
AND site.site_short_name = 'W6'
AND o.order_type_code = 'DI'
And ori.status_code = 'Shipped'
GROUP BY cvl.title
ORDER BY cvl.title
```

Determine Order Statistics: Items Shipped, International, Categories, etc.

Orders

```
SELECT * FROM
(
  SELECT ord.site, cat, ISNULL(COUNT(DISTINCT oi.orno),0) XCOUNT

FROM ord,oi,coop

WHERE ord.orno=oi.orno AND ord.final=coop.cno AND ord.ortype IN ('DI','RP') AND
    oi.status IN ('INSPECT','PSHIP','SHIPPED') AND CAST(oi.acted as date)

/* edit dates and SITE code*/
BETWEEN  '10-01-2014' and '9-30-2015'
AND ord.site='DAV'  GROUP BY ord.site,caT
) src
PIVOT
(
 SUM(xcount)
 FOR cat IN ([UARS], [UFED], [STA], [UCOM], [UPRU] , [UIND], [UAID], [INT], [FGEN], [FCOM], [FPRU],
[FIND])

) piv;
```

_____

Order Items

```
SELECT * FROM
(
  SELECT ord.site, cat, ISNULL(COUNT(*),0) XCOUNT
    FROM ord,oi,coop WHERE ord.orno=oi.orno AND ord.final=coop.cno AND ord.ortype IN ('DI','RP')
AND
    oi.status IN ('INSPECT','PSHIP','SHIPPED') AND CAST(oi.acted as date)

/* edit dates and SITE code*/
BETWEEN '10-01-2014' AND '9-30-2015'  AND ord.site='DAV'
GROUP BY ord.site,cat
) src
PIVOT
(
 SUM(xcount)
 FOR cat IN ([UARS], [UFED], [STA], [UCOM], [UPRU] , [UIND], [UAID], [INT], [FGEN], [FCOM], [FPRU],
[FIND])

) piv;
```

Distinct Items

Display number of items by category ordered in a time period. Edit the dates – in the example, it is for calendar year 2016.

site_short_name	UARS	UFED	STA	UCOM	UPRU	UIND	UAID	INT	FGEN	FCOM	FPRU	FIND	Total
COR	537	5	1599	519	53	1300	0	87	92	108	35	28	4363
DAV	50	10	380	589	116	945	0	0	212	10	211	1	2524
GEN	1079	0	428	583	276	3511	0	0	16	0	45	38	5976
GSOR	2400	0	5207	54	449	13	0	0	0	2	927	6	9058
GSPI	0	0	86	1	0	0	0	0	0	0	2	0	89
HILO	100	0	54	12	7	22	0	0	0	10	41	0	246
MAY	48	28	109	132	13	350	0	0	0	0	9	13	702
MIA	5	0	46	36	165	21	0	0	9	148	3	0	433
NA	72	2	69	51	69	145	0	0	0	0	17	0	425
NC7	5009	213	10296	4092	594	759	0	27	1398	5170	10678	236	38472
NE9	551	0	1081	294	787	180	3	0	105	2488	2736	0	8225
NR6	1730	0	1504	1955	128	523	0	0	0	56	4377	0	10273
NSGC	14412	160	7218	3869	617	1411	0	3603	369	795	17037	146	49637
NSSL	2991	44	181	18	38	2	0	0	0	2	64	0	3340
OPGC	4	0	177	91	33	58	0	0	0	47	19	104	533
PARL	11	0	4	3	69	135	0	0	4	56	10	0	292
PVPO	40	0	0	0	0	0	0	0	0	0	0	0	40
RIV	8	0	0	0	0	0	0	0	0	0	0	0	8
S9	4036	335	10663	6304	9513	804	0	140	353	4401	6908	30	43487
SOY	1336	432	11210	3807	125	194	0	222	142	1156	4407	0	23031
TOB	9	0	236	59	44	8	0	0	0	0	0	0	356
W6	2710	116	13879	3668	1415	3187	0	31	117	1835	6007	300	33265

```
WITH piv2 AS (
SELECT * FROM (
  SELECT s.site_short_name, c.category_code, COUNT(distinct ori.order_request_item_id) ItemCnt
    FROM order_request o
    JOIN order_request_item ori ON ori.order_request_id = o.order_request_id
    JOIN cooperator c ON c.cooperator_id = o.final_recipient_cooperator_id
    JOIN cooperator own ON own.cooperator_id = o.owned_by
    JOIN site s ON s.site_id = own.site_id
   WHERE o.order_type_code in ('DI','RP')
   AND ori.status_code in ('INSPECT','PSHIP','SHIPPED')
   AND ori.status_date BETWEEN '01-01-2016' and '12-31-2016'
  GROUP BY s.site_short_name, c.category_code
) src
PIVOT
(
SUM(ItemCnt)
for category_code in ([UARS], [UFED], [STA], [UCOM], [UPRU] , [UIND], [UAID], [INT], [FGEN], [FCOM],
[FPRU], [FIND])
) piv
```

```
)
SELECT site_short_name,
COALESCE(UARS,0) UARS,
COALESCE(UFED,0) UFED,
COALESCE(STA,0) STA,
COALESCE(UCOM,0) UCOM,
COALESCE(UPRU,0) UPRU,
COALESCE(UIND,0) UIND,
COALESCE(UAID,0) UAID,
COALESCE(INT,0) INT,
COALESCE(FGEN,0) FGEN,
COALESCE(FCOM,0) FCOM,
COALESCE(FPRU,0) FPRU,
COALESCE(FIND,0) FIND,
COALESCE(UARS,0)+COALESCE(UFED,0)+COALESCE(STA,0)+COALESCE(UCOM,0)+COALESCE(UPRU,0)+CO
ALESCE(UIND,0)+COALESCE(UAID,0)
+COALESCE(INT,0)+COALESCE(FGEN,0)+COALESCE(FCOM,0)+COALESCE(FPRU,0)+COALESCE(FIND,0) AS
Total
FROM piv2
```

Distinct Orders

Display number of orders by category ordered in a time period. Edit the dates – in the example, it is for calendar year 2016.

site_short_name	UARS	UFED	STA	UCOM	UPRU	UIND	UAID	INT	FGEN	FCOM	FPRU	FIND	Total
COR	20	2	108	72	15	365	0	3	8	6	10	8	617
DAV	9	3	57	39	11	95	0	0	3	1	10	1	229
GEN	19	0	34	39	16	263	0	0	1	0	4	8	384
GSOR	65	0	68	14	6	9	0	0	0	1	18	1	182
GSPI	0	0	3	1	0	0	0	0	0	0	1	0	5
HILO	5	0	10	3	3	8	0	0	0	3	4	0	36
MAY	12	3	30	52	12	116	0	0	0	0	6	2	233
MIA	4	0	19	12	13	7	0	0	1	2	3	0	61
NA	9	1	24	20	12	56	0	0	0	0	3	0	125
NC7	62	8	435	242	51	92	0	1	9	107	203	8	1218
NE9	10	0	80	34	16	33	1	0	3	21	42	0	240
NR6	46	0	83	30	6	51	0	0	0	5	9	0	230
NSGC	76	4	233	77	26	106	0	4	14	23	155	13	731
NSSL	34	3	13	4	2	1	0	0	0	1	2	0	60
OPGC	1	0	23	7	4	9	0	0	0	3	5	2	54
PARL	2	0	4	2	4	16	0	0	1	3	3	0	35
PVPO	1	0	0	0	0	0	0	0	0	0	0	0	1
RIV	1	0	0	0	0	0	0	0	0	0	0	0	1
S9	69	6	295	102	54	255	0	1	6	64	108	7	967
SOY	50	1	269	113	5	43	0	1	11	10	38	0	541
TOB	4	0	31	15	3	3	0	0	0	0	0	0	56
W6	63	10	339	157	61	510	0	1	8	51	147	17	1364

```
WITH piv2 AS (
SELECT * FROM (
  SELECT s.site_short_name, c.category_code, COUNT(distinct o.order_request_id) OrdCnt
    FROM order_request o
    JOIN order_request_item ori ON ori.order_request_id = o.order_request_id
    JOIN cooperator c ON c.cooperator_id = o.final_recipient_cooperator_id
    JOIN cooperator own ON own.cooperator_id = o.owned_by
    JOIN site s ON s.site_id = own.site_id
   WHERE o.order_type_code in ('DI','RP')
   AND ori.status_code in ('INSPECT','PSHIP','SHIPPED')
  AND ori.status_date BETWEEN '01-01-2016' and '12-31-2016'
  GROUP BY s.site_short_name, c.category_code
) src
PIVOT
(
SUM(OrdCnt)
for category_code in ([UARS], [UFED], [STA], [UCOM], [UPRU] , [UIND], [UAID], [INT], [FGEN], [FCOM],
[FPRU], [FIND])
```

```
) piv
)
SELECT site_short_name,
COALESCE(UARS,0) UARS,
COALESCE(UFED,0) UFED,
COALESCE(STA,0) STA,
COALESCE(UCOM,0) UCOM,
COALESCE(UPRU,0) UPRU,
COALESCE(UIND,0) UIND,
COALESCE(UAID,0) UAID,
COALESCE(INT,0) INT,
COALESCE(FGEN,0) FGEN,
COALESCE(FCOM,0) FCOM,
COALESCE(FPRU,0) FPRU,
COALESCE(FIND,0) FIND,
COALESCE(UARS,0)+COALESCE(UFED,0)+COALESCE(STA,0)+COALESCE(UCOM,0)+COALESCE(UPRU,0)+CO
ALESCE(UIND,0)+COALESCE(UAID,0)
+COALESCE(INT,0)+COALESCE(FGEN,0)+COALESCE(FCOM,0)+COALESCE(FPRU,0)+COALESCE(FIND,0) AS
Total
FROM piv2

_____
```

Selecting Material for Svalbard Backup Consideration

This SQL example refers to legacy GRIN names and some of the dataviews included here are not using standard GRIN-Global naming conventions. This SQL is very specific to NPGS usage.

This SQL looks at the available distribution lots with good recent germinations and puts them in order based on crop/curator. It looks for accessions that do not have a SVALBARD inventory action or items on a previously created SVALBARD order and that do not originate from a CGIAR genebank. Using this query, you can review the distribution lots for germination tests and on-hand seeds.

```
select  * from accession_view where accession_id in (
-- get all accessions for maint group
select accession_id from inventory i
join inventory_maint_policy m on i.inventory_maint_policy_id=m.inventory_maint_policy_id
 where maintenance_name='NC7-amaranth'
except  -- subtract out Svalbard orders and donated from a coop with cat INT
(select accession_id from inventory i2
join order_request_item oi on oi.inventory_id=i2.inventory_id
join order_request r on oi.order_request_id=r.order_request_id
   where final_recipient_cooperator_id =128348 -- coop id for Svalbard orders
  union
    select acc.acid from acc
join src on acc.acid=src.acid
join smbr on smbr.srcno=src.srcno
join coop on coop.cno=smbr.cno
 where cat='INT' and srctype='DONATED'
 )
   ) order by 2 ,3
```