GRIN-Global SQL Library



Revision Date

January 22, 2025

|  |  |
| --- | --- |
|  | **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.** |
|  | The GRIN-Global documentation site contains a page with links to various SQL resources at <http://grin-global.org/sql_examples.htm>.  |

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

Contents

[Basic Queries 6](#_Toc171957696)

[Display all fields for table *xyz* 6](#_Toc171957697)

[Site or Accession Data 7](#_Toc171957698)

[Curators and their Crops & Species 7](#_Toc171957699)

[Curators and their Genera 8](#_Toc171957700)

[Crops with Observations for a Specified Site 8](#_Toc171957701)

[List Your Site’s Top Accession Names 9](#_Toc171957702)

[Display Inventory Fields for a Specific Accession 9](#_Toc171957703)

[Count of accessions with more than one available, distributable inventory 10](#_Toc171957704)

[List of Accessions with …Multiple Available Inventory Records 10](#_Toc171957705)

[List Unavailable Accessions 10](#_Toc171957706)

[Determine Accessions for a Taxon w/out Distributable Inventory (“Non-Exists”) 10](#_Toc171957707)

[Determine Accessions at a Site Needing Backup 11](#_Toc171957708)

[Determine counts (genera, species, accessions) for a specified Inventory Maintenance Policy 12](#_Toc171957709)

[Crop-Related 13](#_Toc171957710)

[Calculating Most Recent Viability 13](#_Toc171957711)

[List Traits for a Crop 13](#_Toc171957712)

[List Accessions That Have Observations for a Specified Trait 14](#_Toc171957713)

[Obtains Counts of Accessions Observed per Methods 14](#_Toc171957714)

[Cooperator & Order-Related 15](#_Toc171957715)

[Display List of Repeat Requestors from the Recent Quarter 15](#_Toc171957716)

[Determine Web Orders which have had multiple Inventory items requested 18](#_Toc171957717)

[List Web Orders for a Requestor (by requestor’s email address or name) 18](#_Toc171957718)

[Related Web Orders and Orders 19](#_Toc171957719)

[Determine the Highest Web Order Number\* 20](#_Toc171957720)

[List New Web Orders 20](#_Toc171957721)

[Listing Web Orders Using SQL (…MIXED) 21](#_Toc171957722)

[Web Order Request NRR Statistics 22](#_Toc171957723)

[For a Web Order, List the Taxon, Curators, and Crops 23](#_Toc171957724)

[List the Number of Orders Sent to a Location by Genus 24](#_Toc171957725)

[Finding Counts of Shipments by Genus for a Specified Time Range 24](#_Toc171957726)

[Retrieve Order Information to Streamline Inventory Loading Activities 25](#_Toc171957727)

[Web User (PW Profile Exist?) 26](#_Toc171957728)

[Display Basic Web Cooperator Info when eMail Address is Known 26](#_Toc171957729)

[Determine Shipping Address(es) by the Web Cooperator’s eMail Address 26](#_Toc171957730)

[Find Related Orders When the WebOrder # is Known 27](#_Toc171957731)

[List History of Weborders & Orders by the Web Cooperator’s email Address 27](#_Toc171957732)

[or WebOrder Number 27](#_Toc171957733)

[Display Web Orders and Related Orders 28](#_Toc171957734)

[(by date and site – or by requestor email address, or by web order #) 28](#_Toc171957735)

[Display the Weborder & Order (with Item Counts, Genus, and Shipping Status) 30](#_Toc171957736)

[Determine Curators Responsible For Web Order Items Across All Sites 31](#_Toc171957737)

[Duplicate Cooperators (Finding) 32](#_Toc171957738)

[GRIN-Global System Information 34](#_Toc171957739)

[Display Current Data Dictionary 34](#_Toc171957740)

[Display Table and Field Name Information 34](#_Toc171957741)

[Display Table and Name In formation with Field Sizes & Types 34](#_Toc171957742)

[Locate Lowercase Data 35](#_Toc171957743)

[Display *Table* Field Names, Data Type, Lengths, and Column Headings (Schema) 36](#_Toc171957744)

[Owner parent relationships in GG 37](#_Toc171957745)

[Display *Table* Field Names (and their Dataview Areas) 38](#_Toc171957746)

[Display *Dataview* Field Names 38](#_Toc171957747)

[Determine Date when Fields, Dataviews, & Tables were Introduced 39](#_Toc171957748)

[Dataview SQL (display a dataview’s SQL) 39](#_Toc171957749)

[Determine a Dataview’s Name 39](#_Toc171957750)

[Display Codes and Code Groups 40](#_Toc171957751)

[Count of Codes Used by the Action Records 40](#_Toc171957752)

[Find which Code Groups are used in which Fields: 40](#_Toc171957753)

[Default Values for the Coded Fields 42](#_Toc171957754)

[Display Source Habitat Descriptors Codes and Code Groups 43](#_Toc171957755)

[Display the Current Client Dataviews (with all of their fields) 44](#_Toc171957756)

[COUNT Number of records in a table 45](#_Toc171957757)

[Number of traits per crop 45](#_Toc171957758)

[Germination Results for the Distribution Lots of a Given Inventory Maintenance Policy 45](#_Toc171957759)

[List the Unique Indexes and Their Fields 46](#_Toc171957760)

[Full Text Indexing – Indicate if it is On or Off 46](#_Toc171957761)

[Display Indexed Fields (Fields with Full Text Search Indexes) 46](#_Toc171957762)

[List the “Autofields” Used in the Search Box 46](#_Toc171957763)

[Site & User Information 47](#_Toc171957764)

[List the Site’s CT User Accounts 47](#_Toc171957765)

[Display CT User Accounts 47](#_Toc171957766)

[Determine the Coop Record Related to the CT User Account 47](#_Toc171957767)

[Determine if the PW Account is Enabled 47](#_Toc171957768)

[Determine Which Permission Policies a User Has Been Included 48](#_Toc171957769)

[Determine What Sites have the MANAGE\_SITE group 48](#_Toc171957770)

[Determine what staff are included in the MANAGE\_SITE group 48](#_Toc171957771)

[Determine what staff are included in the MANAGE\_COOPERATOR group 49](#_Toc171957772)

[Statistics 50](#_Toc171957773)

[Determine which Web Orders have had multiple Inventory items requested 50](#_Toc171957774)

[(for the same accession) 50](#_Toc171957775)

[COUNT of species in database 50](#_Toc171957776)

[COUNT of accessions by family 50](#_Toc171957777)

[COUNT of accessions by genus name 50](#_Toc171957778)

[COUNT unique records in a table 51](#_Toc171957779)

[COUNT of items shipped by state 51](#_Toc171957780)

[COUNT of Items Shipped by Countries for a Date Range 52](#_Toc171957781)

[Determine Order Statistics: Items Shipped, International, Categories, etc. 53](#_Toc171957782)

[Orders 53](#_Toc171957783)

[Order Items 53](#_Toc171957784)

[Distinct Items 54](#_Toc171957785)

[Distinct Orders 56](#_Toc171957786)

[Selecting Material for Svalbard Backup Consideration 58](#_Toc171957787)

This is an online document stored at:

The direct URL for this document is: <https://www.grin-global.org/sql_examples.htm>

|  |  |
| --- | --- |
|  | To learn the basics of SQL, 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 **\*/** 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,…,](#display_field_names)

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

  )

|  |  |
| --- | --- |
| image2449.png | 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 of Accessions with …Multiple Available Inventory Records

SELECT a.accession\_id, a.accession\_number\_part1, a.accession\_number\_part2, a.accession\_number\_part3, s.site\_short\_name

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

|  |  |
| --- | --- |
| A blue pencil in a rectangular box  Description automatically generated | You can use almost the same WHERE clause in the ST, only needing to change the a.accession\_id in the sub-select to accession.accession\_id:WHERE (SELECT count(\*) FROM inventory WHERE accession\_id = accession.accession\_id AND is\_distributable = 'Y' AND is\_available = 'Y') > 1This works in the Search Tool’s accession dataview. |

#### List Unavailable Accessions

This SQL is explained in the [online SQL Guide](http://www.grin-global.org/docs/gg_sql_quick_guide.docx), 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 counts (genera, species, accessions) for a specified Inventory Maintenance Policy

This SQL is a good example of many joins. These are necessary to connect an **Inventory Maintenance Policy** to **Taxonomy\_Family**.

SELECT family\_name

  , count(distinct tg.genus\_name) AS genera

  , count(distinct ts.name) AS species

  , count (distinct a.accession\_id) AS accessions

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

JOIN inventory i ON i.accession\_id = a.accession\_id

JOIN inventory\_maint\_policy imp ON imp.inventory\_maint\_policy\_id = i.inventory\_maint\_policy\_id

WHERE imp.maintenance\_name = 'w6\_native'

GROUP by family\_name

ORDER BY accessions DESC

###### Sample Output

|  |  |  |  |
| --- | --- | --- | --- |
| **family\_name** | **genera** | **species** | **accessions** |
| Asteraceae | 192 | 595 | 2881 |
| Poaceae | 69 | 219 | 1506 |
| Cyperaceae | 17 | 239 | 901 |
| Chenopodiaceae | 13 | 40 | 473 |
| Juncaceae | 2 | 61 | 339 |
| Rosaceae | 27 | 80 | 295 |
| Brassicaceae | 28 | 79 | 254 |
| Apiaceae | 27 | 79 | 226 |
| Fabaceae | 38 | 108 | 214 |
| Malvaceae | 7 | 23 | 192 |
| Lamiaceae | 21 | 69 | 176 |
| Boraginaceae | 9 | 61 | 161 |
| Hydrophyllaceae | 4 | 43 | 150 |
| Onagraceae | 12 | 48 | 147 |
| Polygonaceae | 9 | 48 | 132 |
| Plantaginaceae | 17 | 57 | 109 |
| Zygophyllaceae | 2 | 2 | 89 |
| Solanaceae | 6 | 18 | 79 |
| Asparagaceae | 10 | 32 | 72 |
| Orobanchaceae | 8 | 34 | 68 |

### 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

#### Obtains Counts of Accessions Observed per Methods

The following SQL provides counts of accessions evaluated per method for a crop, in a grid (matrix). You need to supply the crop ID and the method names.

The example is using crop-Id 78 (strawberry) and then the strawberry crop’s related methods. Remember when building the list of methods, do not use a comma after the last method in the list.

WITH sb AS (SELECT ct.coded\_name, m.name, count(distinct accession\_id) AS accs

FROM crop\_trait ct

JOIN crop\_trait\_observation cto ON cto.crop\_trait\_id = ct.crop\_trait\_id

JOIN method m ON m.method\_id = cto.method\_id

JOIN inventory i ON i.inventory\_id = cto.inventory\_id

WHERE ct.crop\_id = **78**

GROUP BY ct.coded\_name, m.name

)

SELECT \* FROM sb

PIVOT(

    sum(accs)

    FOR name IN (

              **[2003.CROWNROT],**

**[2007.CYTOLOGY],**

**…**

**[Zurn et al.2020 Soilborne Pathogen Evaluation])**

) AS pivot\_table;

### 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:

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](https://docs.google.com/spreadsheet/ccc?key=0AvdWZS-UqEE7dHFaRnRsR1RxOUx0em9KZmhNZTVlRnc&hl=en%23gid=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 matching 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

#### Dataview SQL (display a dataview’s SQL)

-- display SQL for a dataview

-- to learn the dataview’s name, see next SQL example below

SELECT sd.dataview\_name

, REPLACE(sql\_statement, CHAR(13) + CHAR(10), CHAR(60)+'br'+CHAR(62))  AS html\_sql

FROM sys\_dataview sd

JOIN sys\_dataview\_sql sds ON sds.sys\_dataview\_id = sd.sys\_dataview\_id

WHERE sd.dataview\_name =  'web\_order\_request\_item\_list' AND database\_engine\_tag = 'sqlserver'

#### Determine a Dataview’s Name

SELECT dataview\_name, database\_area\_code, title, description

FROM sys\_dataview sd

JOIN sys\_dataview\_lang sdl ON sdl.sys\_dataview\_id = sd.sys\_dataview\_id

WHERE category\_code = 'Site W6'

ORDER BY 2,3

#### 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

#### Count of Codes Used by the Action Records

SELECT action\_name\_code, count(\*) AS use\_count

FROM inventory\_action

-- or accession\_action

GROUP BY action\_name\_code

ORDER BY 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

#### 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

#### 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'**

#### 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

#### 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

### 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 the Coop Record Related to the CT User Account

Select user\_name, sys\_user.modified\_date, c.current\_cooperator\_id

FROM sys\_user

JOIN cooperator c

ON c.cooperator\_id = sys\_user.cooperator\_id

WHERE user\_name

LIKE '%rein%'

#### Determine if PW Accounts are Enabled for CT users

SELECT wu.user\_name AS PW\_user, su.user\_name AS CT\_user
FROM web\_user wu
JOIN cooperator c ON c.web\_cooperator\_id = wu.web\_cooperator\_id
JOIN sys\_user su ON su.cooperator\_id = c.cooperator\_id
ORDER BY PW\_user

*better version – Includes site and other info*

SELECT user\_name, last\_name, first\_name, primary\_phone, site\_short\_name, site\_long\_name, s.site\_id, is\_enabled

FROM sys\_user su

JOIN cooperator c ON su.cooperator\_id = c.cooperator\_id

JOIN site s ON c.site\_id = s.site\_id

-- omit where clause to see all

WHERE site\_short\_name = '**DBMU**'

ORDER BY site\_short\_name, last\_name, first\_name

#### 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

At NPGS, permission groups called MANAGE\_SITE\_*sitename* have been established so that any member in that group, for their site, can change the ownership of *any* records owned by someone in that same site.

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

#### Determine what staff are included in the MANAGE\_COOPERATOR group

Members of this permission group have the ability to update any record in the tables:

* **Cooperator**
* **Cooperator-Group**

|  |
| --- |
| SELECT su.user\_name, sg.group\_tag, sp.permission\_tag, st.table\_name, CONCAT(c.first\_name, ' ', c.last\_name) AS owner, |
| c.site\_id, c.organization\_abbrev |
| 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 |
| JOIN site s ON s.site\_id = c.site\_id |
| WHERE |
| sg.group\_tag LIKE 'MANAGE\_COOPERATOR%' |
| 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.



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)+COALESCE(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.



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)+COALESCE(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

|  |  |
| --- | --- |
| image2449.png | 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