Using GRHANITE Hashes as generated by GRHANITE Client

March 2017 V4.0

Table of Contents

Background

A **cryptographic hash function** is a hash function that takes an arbitrary block of data and returns a fixed-size bit string, the *cryptographic hash value*, such that any (accidental or intentional) change to the data will (with very high probability) change the hash value.

- GRHANITE utilises cryptographic hashes for privacy-protecting record linkage.
- A GRHANITE Hash – is a cryptographic hash
- A GRHANITE Hash Type – is a combination of patient identifying data that forms the basis of the information used in generating any particular GRHANITE Hash.

The GRHANITE Data Linkage technology is capable of generating a range of different hash values that can be used in record linkage projects. GRHANITE Client as utilised in GRHANITE GP data extractions utilises a specific subset of hash types. The choice of hash types supported by GRHANITE Client was made after a comprehensive analysis of hashing quality conducted utilising the BioGrid datasets and FERBL dummy datasets.

Different research projects may require different levels of accuracy in record linkage. The exact accuracy cannot be determined because different populations have different characteristics – for example a study looking at differences in twins would have specific challenges in avoiding false-positive linkages where the dates of birth, location, Medicare ID and name of twins are similar. For this reason, the researcher must understand the composition of GRHANITE hashes so that decisions can be made based on your patient population about how to interpret the records to refine your data linkage.

The GRHANITE Client Hashes

GRHANITE Client generates and extracts hash values based on the following information:

- Medicare digits 5-9: **Medicare5-9**
- Date of Birth: **DoB**
- Date of Birth with day/month transpositions permitted: **M(DoB)**
- Sex: **Sex**
- Phonetic encoding of surname and forename where they are permitted to be either way round – e.g. John Smith or Smith John: **Reversal(M(Surname),M(FName))**
- First 3 digits of Forename: **FName13**
- Year of Birth: **YOB**
- Postcode: **PCode**

The following Hash types utilising the information above is exported:

HashTypes

HashType	Identifiers
1020	Medicare5-9_DoB_Sex
2210	Medicare5-9, PCode, FName13, YoB
3012	Reversal(M(Surname), M(FName)), M(DoB)
4112	Reversal(M(Surname), M(FName)), MCare 5-9

Notes

All hash types are sensitive in identifying valid patient linkages but there are the following potential problems in relation to false positive record identification:

Hash types 2210, 3012 and 4112 do not require a sex match. Any data linkage using these hashes where there is a sex mismatch has a high likelihood of being a false positive.

Hash type 3012 is very flexible in allowing surnames and forenames to be reversed and errors to be present in a date of birth match. The other hashes rely on the Medicare ID being present – if no Medicare ID is present then only a 3012 hash is available. If a match exists using hash 3012 ONLY there is a risk that the match is a false positive. If the Medicare ID is normally available in your dataset then matches that exist only on a 3012 have a high likelihood of being false-positives. In many datasets, you have additional information you can use to help verify that two records belong to the same individual. An example is, if you have a 3012 match and the patient has a referral date that is closely aligned with a clinic admission this may help verify the quality of the match.

Strategies for false-positive avoidance

The 3012 example above means that determining if two records should link requires additional thought rather than necessarily trusting the linkages. The GRHANITE stand-alone record linker can be utilised instead of the Databank 'Link Data' button to give more flexibility in understanding and interpreting linkages.

Utilising the GRHANITE stand-alone record linker

Step 1: Database Creation

Create a database (if not already created) to hold the tables used in the record linkage process – for example create a database called GRHANITE_Linkage

```
CREATE DATABASE [GRHANITE_Linkage]
GO
```

Step 2: Creation of a List of Eligible Patients for Record Linkage

Every patient record transmitted has a unique GRHANITE-assigned patient number. This is usually called 'Patient_UUID'. In some cases, GRHANITE has to be reset within a practice, lab or clinic. In these cases, the 'Patient_UUID' assigned to the patient will be re-generated. This means that the GRHANITE_Hashes data that is present may contain hashes that refer to OLD Patient_UUID values referring to OLD data. This can lead to large numbers of Hash records that are no longer current and that should not participate in the record linkage. To resolve this, you must create a list of all the current patients (Patient_UUID's) that represent valid patients you wish to record link.

Generate a table containing a unique list of Patient_UUID's for patients you wish to record-link and populate it.

Example Table Creation:

```
USE [GRHANITE_Linkage]
GO

CREATE TABLE [GRHANITE_Linkage].[dbo].[All_Patients_For_Linkage](
	[Patient_UUID] [varchar](50) COLLATE SQL_Latin1_General_CP1_CI_AS NULL,
) ON [PRIMARY]
```

Example insertion of unique patient ID's:

```
INSERT INTO [GRHANITE_Linkage].[dbo].[ All_Patients_For_Linkage]
(
	[Patient_UUID]
)
SELECT DISTINCT [Patient_UUID] FROM [GRHANITE_PROJECT_<PROJECTNAME>].[dbo]…
```

Step 3: Creation of the GRHANITE_Hashes_Working Table

GRHANITE Hashes arrive in a GRHANITE_PROJECT_<projectname> database. The following SQL code creates a table in your working record linkage database to hold the hashes you wish to record link. Note that **<PROJECTNAME>** must be replaced with the name of your GRHANITE project:

```
-- Cleanup GRHANITE_Linkaged database - assumes the GRHANITE_Linkage database has been created
first
USE [GRHANITE_Linkage]
GO
IF  EXISTS (SELECT * FROM sys.objects WHERE object_id =
OBJECT_ID(N'[dbo].[GRHANITE_Hashes_Working]') AND type in (N'U'))
BEGIN
	TRUNCATE TABLE [dbo].[GRHANITE_Hashes_Working]
	DROP TABLE [dbo].[GRHANITE_Hashes_Working]
END
GO

SET ANSI_NULLS ON
GO
SET QUOTED_IDENTIFIER ON
```

```
GO
SET ANSI_PADDING ON
GO
CREATE TABLE [dbo].[GRHANITE_Hashes_Working](
        [GRHANITE_Site] [varchar](50) COLLATE SQL_Latin1_General_CP1_CI_AS NOT NULL,
        [GRHANITE_Clinic] [varchar](50) COLLATE SQL_Latin1_General_CP1_CI_AS NOT NULL,
        [GRHANITE_DBVersion] [varchar](30) COLLATE SQL_Latin1_General_CP1_CI_AS NOT NULL,
        [GRHANITE_Import_Time] [char](18) COLLATE SQL_Latin1_General_CP1_CI_AS NOT NULL,
        [GRHANITE_Import_Status] [char](1) COLLATE SQL_Latin1_General_CP1_CI_AS NOT NULL,
        [GRHANITE_CHECKSUM] [bigint] NULL,
        [Patient_UUID] [varchar](50) COLLATE SQL_Latin1_General_CP1_CI_AS NULL,
        [HashType] [smallint] NULL,
        [Hash] [char](144) COLLATE SQL_Latin1_General_CP1_CI_AS NULL,
        [AgrWeight] [real] NULL,
        [GRHANITE_GUID] [varchar](200) COLLATE SQL_Latin1_General_CP1_CI_AS NOT NULL
) ON [PRIMARY]

GO
SET ANSI_PADDING OFF

GO
```

Step 4: Populating GRHANITE_Hashes_Working

You now need to populate the GRHANITE_Hashes_Working table with the Hashes and patients you wish to record-link. Note the section in Yellow below that ensures the hashes utilised are for patients that are in your unique list of eligible patients created in step 2 above:

```
-- Copy data for related sites only
INSERT INTO [GRHANITE_Linkage].[dbo].[GRHANITE_Hashes_Working]
(
        [GRHANITE_Site],[GRHANITE_Clinic],[GRHANITE_DBVersion],
        [GRHANITE_Import_Time],[GRHANITE_Import_Status],
        [GRHANITE_CHECKSUM],[Patient_UUID],[HashType],[Hash],
        [AgrWeight],
    [GRHANITE_GUID]
)
SELECT
        [GRHANITE_Site],[GRHANITE_Clinic],[GRHANITE_DBVersion],
        [GRHANITE_Import_Time],[GRHANITE_Import_Status],
        [GRHANITE_CHECKSUM],[Patient_UUID],[HashType],[Hash],
        0, -- [AgrWeight]
    [GRHANITE_GUID]
FROM [GRHANITE_PROJECT_<PROJECTNAME>].[dbo].[GRHANITE_Hashes] h
INNER JOIN [GRHANITE_Linkage].[dbo].[ All_Patients_For_Linkage] ap
ON ap.Patient_UUID = h.Patient_UUID
```

Step 5: Cleaning your data and removing duplicates

For each generated hash type, your data should have no duplicate records in Patient_UUID from a single data source before running the record linker.

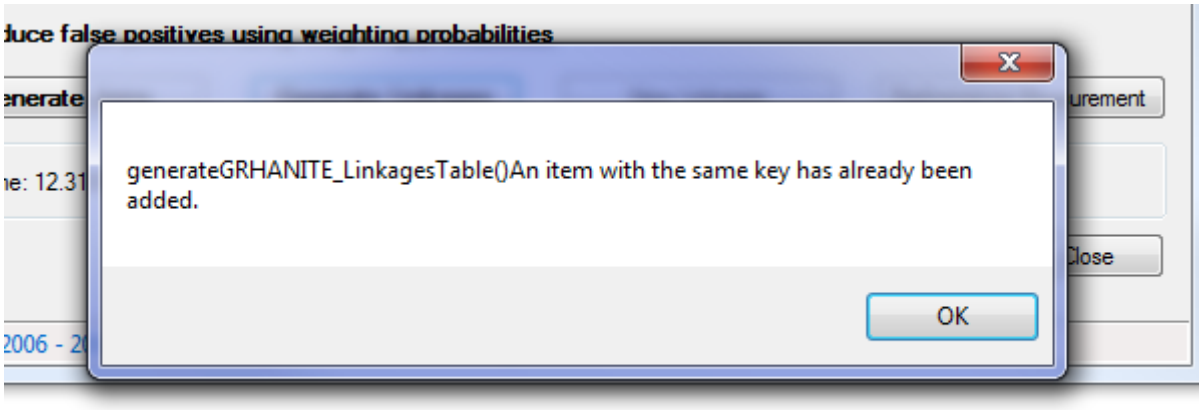
If you have a problem with your underlying data the linker will produce an error message. Although the linkage process will still complete and the linked files will be produced, your linked results will be incomplete. The query below should return no rows. Any rows returned indicate that you have duplicate records and you need to review how you have merged and prepared the data for the linker:

```
SELECT GRHANITE_Clinic + '_' + Patient_UUID + '_'+CONVERT(varchar(4),HashType),
COUNT(GRHANITE_Clinic + '_' + Patient_UUID + '_'+CONVERT(varchar(4),HashType))
FROM GRHANITE_Hashes_Working
GROUP BY GRHANITE_Clinic + '_' + Patient_UUID + '_'+CONVERT(varchar(4),HashType)
HAVING COUNT(GRHANITE_Clinic + '_' + Patient_UUID + '_'+CONVERT(varchar(4),HashType)) > 1
```

If you have multiple data extracts from different clinics at one site then the Patient_UUID must be different from each clinic. The combination highlighted below is not permitted.

	GRHANITE_Site	GRHANITE_Clinic	Patient_UUID
1	QUEENSLAND_LABS	LAB1	1
2	QUEENSLAND_LABS	LAB2	1
3	QUEENSLAND_LABS	LAB1	2
4	QUEENSLAND_LABS	LAB1	3
5	QUEENSLAND_LABS	LAB1	4
6	QUEENSLAND_LABS	LAB1	5

A combination like this will generate the error below:

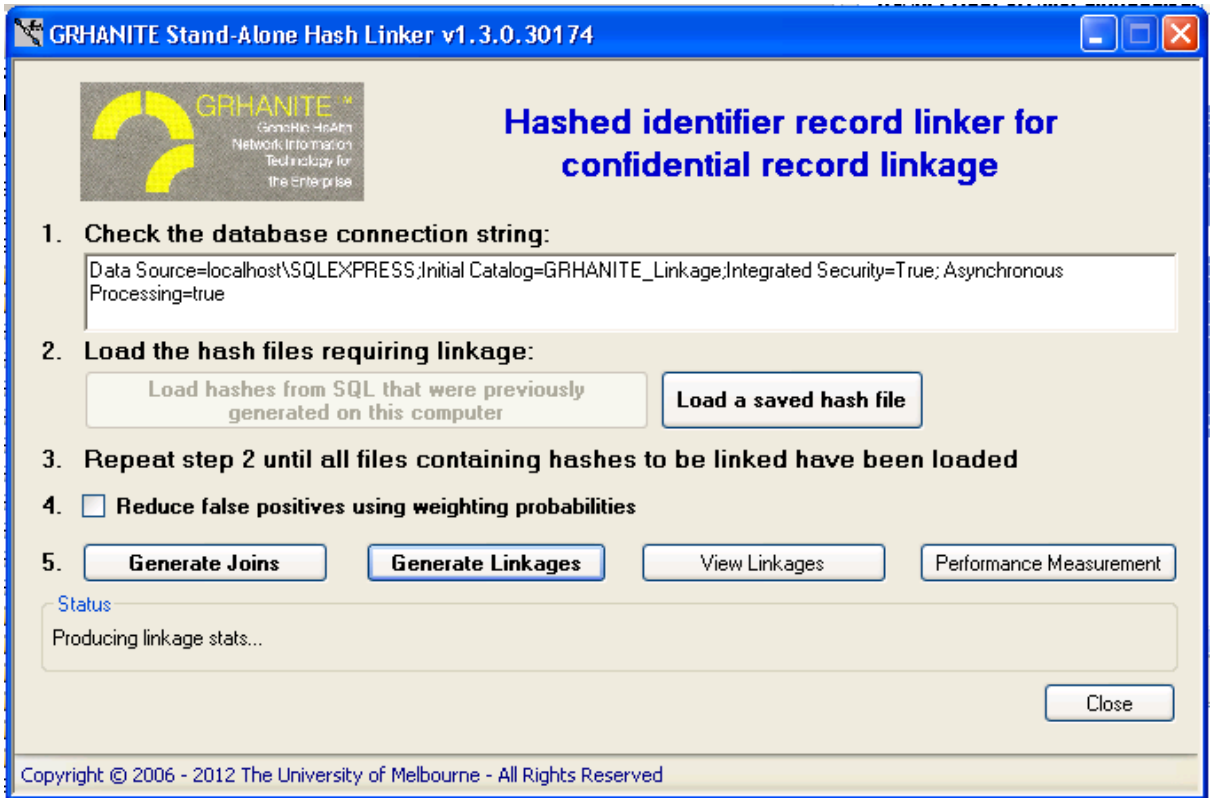


].[dbo].[DB_SVNIBAC_1_0_GRHANITE_Hashes_V1]

A new version of the linker that supports this combination will be available in 2017.

Step 6: Running the Record Linker

Run the GRHANITE stand-alone record linker v1.3.0.30174 or above:



- Check the database connection string points to the GRHANITE_Linkage database and has appropriate security rights
- Click on 'Load hashes from SQL' – this will load the hashes that Step 1 has placed in the GRHANITE_Linkages database
- Click on 'Generate Joins'
    - If you want to analyse the join results before linking, please see Appendix A for a detailed dicussion of interpreting and utilising joins. This can be done before genrating the final record linkages
- Click 'Generate Linkages'

The GRHANITE record linkages will now be present in **`[GRHANITE_Linkage].[dbo].[GRHANITE_Linkages]`**

Step 7: Finalisation of Your Linked Dataset

Some patients in the `[GRHANITE_Linkage].[dbo].[All_Patients_For_Linkage]` table created in Step 2 will not have valid Hashes for record linkage. This is because some patients will have missing demographic information and hence hashes could not be generated. This means that patients with no hashes will not exist in the GRHANITE_Linkages table. In most cases, you will want to have a LINK_ID associated with all patients whether they were able to generate a hash or not. The queries below generate a record in the GRHANITE_Linkages table for all patients originally supplied who have no GRHANITE Hash available:

```
--Add linkid records for patients with no linkages - so that all records are counted.
--Create a list of all patients with no linkage record
USE GRHANITE_Linkage
GO
SELECT DISTINCT(Patient_UUID) INTO Missing_Patient_UUIDs FROM GRHANITE_Hashes_Working
w LEFT OUTER JOIN GRHANITE_Linkages l ON w.Patient_UUID = l.UID WHERE l.UID IS NULL
GO
SELECT w.* INTO No_Linkages FROM GRHANITE_Hashes_Working w INNER JOIN
Missing_Patient_UUIDs u ON w.Patient_UUID = u.Patient_UUID
GO
--Create a table of all linkage records to be added
DECLARE @maxID AS int
DECLARE @sqlvar As nvarchar(max)
SELECT @maxID = MAX(LINK_ID) FROM GRHANITE_Linkages
SELECT @sqlvar =
'CREATE TABLE ADD_Linkages
(
                LINK_ID int identity(' + CONVERT(varchar,@maxID+1)+',1),
                UID varchar(50) not null,
                [Site] varchar(50) not null,
                SOURCE_CLINIC varchar(50) null
)'
EXEC sp_sqlexec @sqlvar
GO
--Insert the addtional records into the GRHANITE_Linkages table
INSERT INTO ADD_Linkages(UID, [Site], SOURCE_CLINIC) SELECT DISTINCT Patient_UUID,
GRHANITE_Site, GRHANITE_Clinic FROM No_Linkages
INSERT INTO GRHANITE_Linkages SELECT * FROM ADD_Linkages

TRUNCATE TABLE Missing_Patient_UUIDs
TRUNCATE TABLE No_Linkages
TRUNCATE TABLE ADD_Linkages
DROP TABLE Missing_Patient_UUIDs
```

```
DROP TABLE No_Linkages
DROP TABLE ADD_Linkages
```

[GRHANITE_Linkages] now contains the definitive list of record linkages across your patient population as defined in [All_Patients_For_Linkage].

Appendix A – Interpreting and utilising GRHANITE Joins before clicking 'Generate Linkages'

When the 'Generate Joins' step completes, it will have generated two tables in GRHANITE_Linkages:

GRHANITE_Joins – Identifies each occurrence where the GRHANITE-supplied hashes suggest a record linkage. The table identifies the unique ID of the two patient records and allocates a unique JOIN identifier. Additional fields are available for later manipulation.

GRHANITE_JoinDetails – Identifies the Hash Type of the join above i.e. 1020, 2210, 3012 or 4112

To make best use of these tables, GRHANITE_Joins can be updated to record each of the hash types and to hold information such as sex mis-matches. The following code when run at this stage will:

- Update GRHANITE_Joins to record what hash type has occurred for each join
- Create a GENDER table that can be used to determine gender mis-matches
- Copy the GENDER information for each patient from additional tables extracted by GRHANITE – in this case from Medical Director
- Utilises otherwise unused fields in GRHANITE_Joins to clean-up the gender information to identify records with a gender mismatch:
    - AgrWeight_Grp5 = 0 – GENDER MATCH
    - AgrWeight_Grp5 = 1 – GENDER MISMATCH
- The code finally uses the TotalAgrWeight field to record records with a gender mis-match, a match on 3012 only or both.
- The end of the code gives an example where 3012 only records and gender mis-matches are deleted

```
--This code utilises the GRHANITE_Joins and GRHANITE_JoinDetails tables along with Gender data
--to create a remove potential false positive records prior to generating final GRHANITE
Linkages

--The GRHANITE_Joins table is used as an intermediary table to hold information about the
linkages
--It is updated here to record the types of linkages found between patients
UPDATE GRHANITE_Joins SET nGroups = 0
UPDATE GRHANITE_Joins SET AgrWeight_Grp1 = 1020, nGroups = nGroups + 1 FROM GRHANITE_Joins j
INNER JOIN GRHANITE_JoinDetails d
ON j.JoinUID = d.JoinUID AND d.HashType = 1020
UPDATE GRHANITE_Joins SET AgrWeight_Grp2 = 2210, nGroups = nGroups + 1  FROM GRHANITE_Joins j
INNER JOIN GRHANITE_JoinDetails d
ON j.JoinUID = d.JoinUID AND d.HashType = 2210
UPDATE GRHANITE_Joins SET AgrWeight_Grp3 = 3012, nGroups = nGroups + 1  FROM GRHANITE_Joins j
INNER JOIN GRHANITE_JoinDetails d
ON j.JoinUID = d.JoinUID AND d.HashType = 3012
UPDATE GRHANITE_Joins SET AgrWeight_Grp4 = 4112, nGroups = nGroups + 1  FROM GRHANITE_Joins j
INNER JOIN GRHANITE_JoinDetails d
ON j.JoinUID = d.JoinUID AND d.HashType = 4112
GO
--Patient GENDER is required to remove those with a gender mis-match
--The gender is retrieved from the originating database tables
IF EXISTS (SELECT * FROM sysobjects WHERE name = 'GENDER' AND xtype = 'U')
BEGIN
        TRUNCATE TABLE GENDER
        DROP TABLE GENDER
END
CREATE TABLE [dbo].[GENDER](
        [Patient_UUID] [uniqueidentifier] NOT NULL,
        [GENDER_CODE] [varchar](10) NULL
)
INSERT INTO GENDER SELECT Patient_UUID, GENDER_CODE FROM
GRHANITE_PROJECT_ACCEPT.dbo.DB_BestPractice_1711_CHLA_PATIENT
```

```
INSERT INTO GENDER SELECT Patient_UUID, GENDER_CODE FROM
GRHANITE_PROJECT_ACCEPT.dbo.DB_Genie_V7_6_7_ALL_PATIENT
INSERT INTO GENDER SELECT Patient_UUID, GENDER_CODE FROM
GRHANITE_PROJECT_ACCEPT.dbo.DB_Medical_Director_V3_10_5_ALL_PATIENT
INSERT INTO GENDER SELECT Patient_UUID, GENDER_CODE FROM
GRHANITE_PROJECT_ACCEPT.dbo.DB_MEDTECH_32_V7_3_0_CHLA_PATIENT
INSERT INTO GENDER SELECT Patient_UUID, GENDER_CODE FROM
GRHANITE_PROJECT_ACCEPT.dbo.DB_Practix_V1_36_1_8_CHLA_PATIENT
INSERT INTO GENDER SELECT Patient_UUID, GENDER_CODE FROM
GRHANITE_PROJECT_ACCEPT.dbo.DB_QLDHLAB_V1_CHLA_PATIENT
INSERT INTO GENDER SELECT Patient_UUID, GENDER_CODE FROM
GRHANITE_PROJECT_ACCEPT.dbo.DB_Zedmed_V14_ALL_PATIENT
UPDATE GENDER SET GENDER_CODE = 'U' WHERE GENDER_CODE NOT IN ('M','F','U','I')
GO
--The following code sets a flag temporarily recording GENDER from UID1
UPDATE GRHANITE_Joins SET TotalAgrWeight = 0
UPDATE GRHANITE_Joins SET TotalAgrWeight = 1 FROM GRHANITE_Joins j INNER JOIN GENDER g
ON j.UID1 = g.Patient_UUID AND GENDER_CODE = 'M'
UPDATE GRHANITE_Joins SET TotalAgrWeight = 2 FROM GRHANITE_Joins j INNER JOIN GENDER g
ON j.UID1 = g.Patient_UUID AND GENDER_CODE = 'F'
UPDATE GRHANITE_Joins SET TotalAgrWeight = 3 FROM GRHANITE_Joins j INNER JOIN GENDER g
ON j.UID1 = g.Patient_UUID AND GENDER_CODE = 'U'
UPDATE GRHANITE_Joins SET TotalAgrWeight = 4 FROM GRHANITE_Joins j INNER JOIN GENDER g
ON j.UID1 = g.Patient_UUID AND GENDER_CODE = 'I'
UPDATE GRHANITE_Joins SET TotalAgrWeight = 3 WHERE TotalAgrWeight = 0
--The following identifies patients who have a gender match and those where gender is unknown
- in which case a matched gender is assumed
UPDATE GRHANITE_Joins SET AgrWeight_Grp5 = 0
UPDATE GRHANITE_Joins SET AgrWeight_Grp5 = 1 FROM GRHANITE_Joins j INNER JOIN GENDER p
ON j.UID2 = p.Patient_UUID WHERE GENDER_CODE = 'M' AND TotalAgrWeight = 1
UPDATE GRHANITE_Joins SET AgrWeight_Grp5 = 1 FROM GRHANITE_Joins j INNER JOIN GENDER p
ON j.UID2 = p.Patient_UUID WHERE GENDER_CODE = 'F' AND TotalAgrWeight = 2
UPDATE GRHANITE_Joins SET AgrWeight_Grp5 = 1 WHERE TotalAgrWeight = 3
UPDATE GRHANITE_Joins SET AgrWeight_Grp5 = 1 WHERE TotalAgrWeight = 4
UPDATE GRHANITE_Joins SET AgrWeight_Grp5 = 1 FROM GRHANITE_Joins j INNER JOIN GENDER p
ON j.UID2 = p.Patient_UUID WHERE GENDER_CODE = 'U' AND TotalAgrWeight = 2
UPDATE GRHANITE_Joins SET AgrWeight_Grp5 = 1 FROM GRHANITE_Joins j INNER JOIN GENDER p
ON j.UID2 = p.Patient_UUID WHERE GENDER_CODE = 'I' AND TotalAgrWeight = 2
--This code uses the TotalAgrWeight to identify patients with a 3012 match only and those with
a sex mis-match
--0 = exact match
--1 = 3012 linkage only
--2 = sex mismatch
--3 = 3012 linkage only and a sex mismatch
UPDATE GRHANITE_Joins SET TotalAgrWeight = 0
UPDATE GRHANITE_Joins SET TotalAgrWeight = 1 WHERE nGroups = 1 AND AgrWeight_Grp3 = 3012
UPDATE GRHANITE_Joins SET TotalAgrWeight = TotalAgrWeight + 2 WHERE AgrWeight_Grp5 = 0
GO

--Run this code to delete 3012 only joins and sex mis-matches
--DELETE FROM GRHANITE_Joins WHERE TotalAgrWeight > 0
```

Once unwanted joins have been removed, click on Genarate Linkages to create the GRHANITE_Linkages table that contains the definitive record of linkage between tables. After generating the GRHANITE_Linkages table, the following cide will updat the linkage information in the GRHANITE_Joins table. This allows you to better visualise the linkages:

```
--After generating the GRHANITE_Linkages, run this so that you can see the LINK_ID in the
GRHANITE_Joins table
UPDATE GRHANITE_Joins SET AgrWeight_Grp5 = 0
UPDATE GRHANITE_Joins SET AgrWeight_Grp5 = LINK_ID FROM GRHANITE_Joins j INNER JOIN
GRHANITE_Linkages l
ON j.UID1 = l.UID
UPDATE GRHANITE_Joins SET AgrWeight_Grp5 = LINK_ID FROM GRHANITE_Joins j INNER JOIN
GRHANITE_Linkages l
ON j.UID2 = l.UID
GO
```