Contents

The project, “creation of a marine zooplankton database”, required the following steps:

1. data wrangling (power query, python),
2. relational database model design and implementation of it (MSSQL)
3. user-friendly queries for data mining in R programming language

Here follows the procedure of all the steps mentioned above:

Appendix-Power Query guide

MACROS used

MACROS with name ‘trX’ used:

```
Sub trX()


' TeachExcel.com


' Make the macro run faster on large data sets.
Application.ScreenUpdating = False


' Do something with the user-selected cells/range.
xRow = Selection.Rows.Count
xCol = Selection.Column


' The row that the Transposed data will be put into.
nextRow = 1


' Hard-code Column header example
'Range("C1").Value = "Column Header"


' How many rows to Transpose.
```

```
stepValue = InputBox("How many rows should be grouped together?")

' Loop through the user-selected data using a step value.
For i = 1 To xRow Step stepValue

    ' Copy the data, using the step value to determine the size of
    ' the copied range.
    Cells(i, xCol).Resize(stepValue).Copy

    'Transpose the data.
    Cells(1, xCol).Offset(nextRow, 3).PasteSpecial Paste:=xlPasteAll, Transpose:=True

    ' Increment the nextRow value so the copied data goes onto
    ' a new line.
    nextRow = nextRow + 1

Next

' Remove the "copy lines" from the Transposed data.
Application.CutCopyMode = False
' Make Excel function as expected after the macro is finished.
Application.ScreenUpdating = True

End Sub
```

Walkthrough

Abbreviations:

DA: down arrow, RA: right arrow

Produce Excel files from ".dat" files

Dat files:

Produce excel file with the results values:

Open Excel → Data tab → Get Data→ From File → From Folder → Choose DANA0697 → choose Dat → OK → OK → on the pop-up Data preview window click Transform Data (Figure 1) → We are in the Power Query Editor(Figure 2) → right click on Content column → Chose Remove Other Columns → click Combine files (Figure 3) → on Combine files preview screen choose "Tab" as delimiter (Figure 4)→ click OK → click the arrow next to column1 → choose "text Filters (Figure 5)→ Does not contain "," (Figure 6)→ Now logging data is gone (Figure 7)→ At Filter tab , under "Text filter" enter "136" and untick it (Figure 8)→ Close&Load from Home tab tool bar → Data is transferred to Excel worksheet → Right click on column → choose Table → Convert to range (Figure 9)→ OK → Delete first row → CTRL+SHIFT+DA to choose all working cells of first column → Alt+F8 to open Macros tab → Run Macro → in the pop-up window at the question "How many rows" fill "136" (Figure 10)→ Click OK

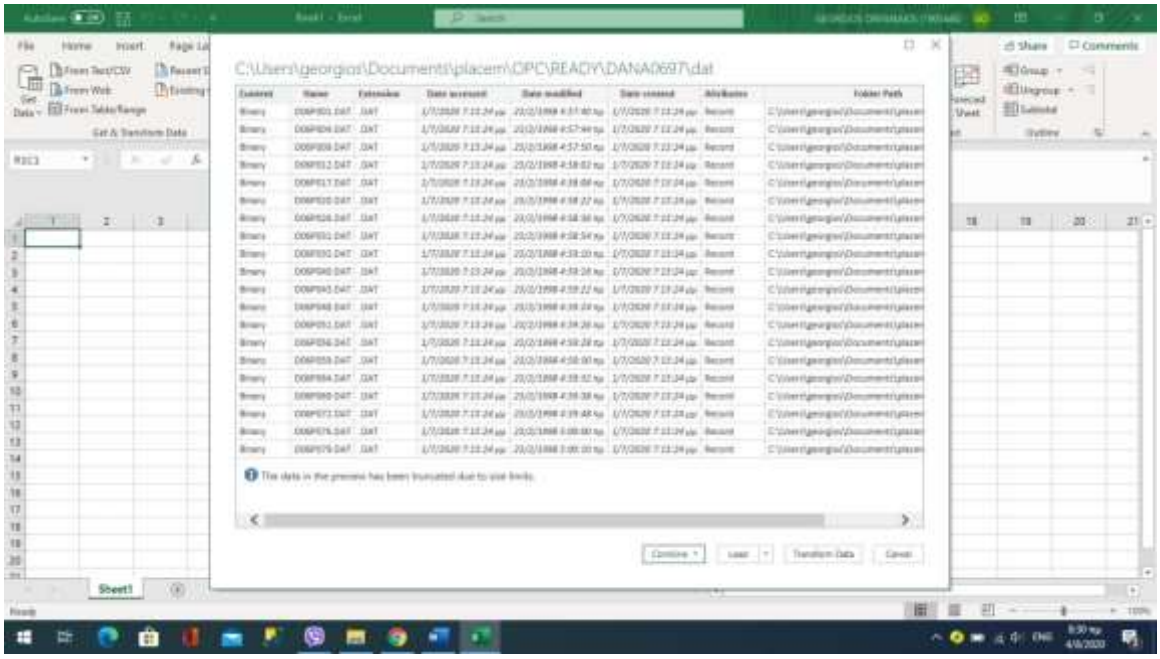


Figure 1

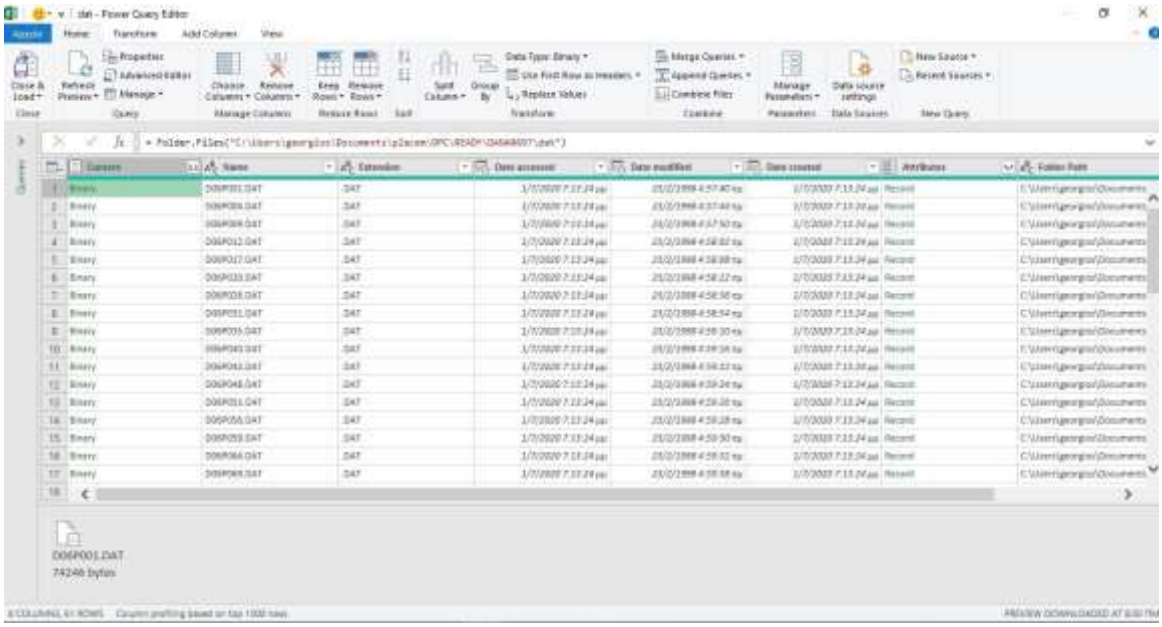


Figure 2

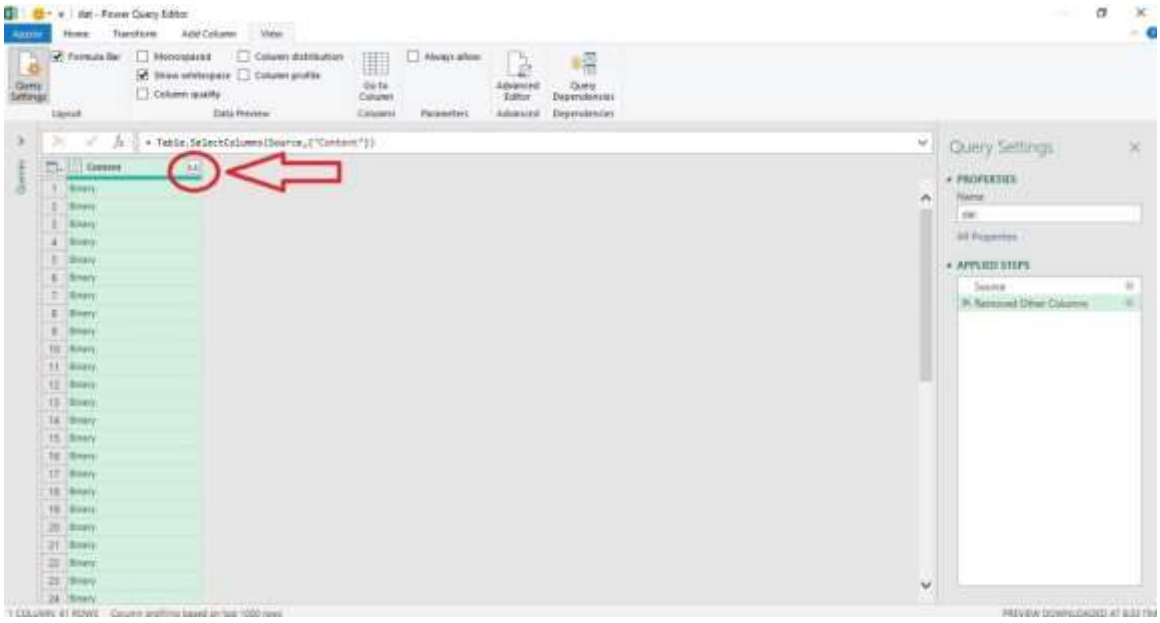


Figure 3

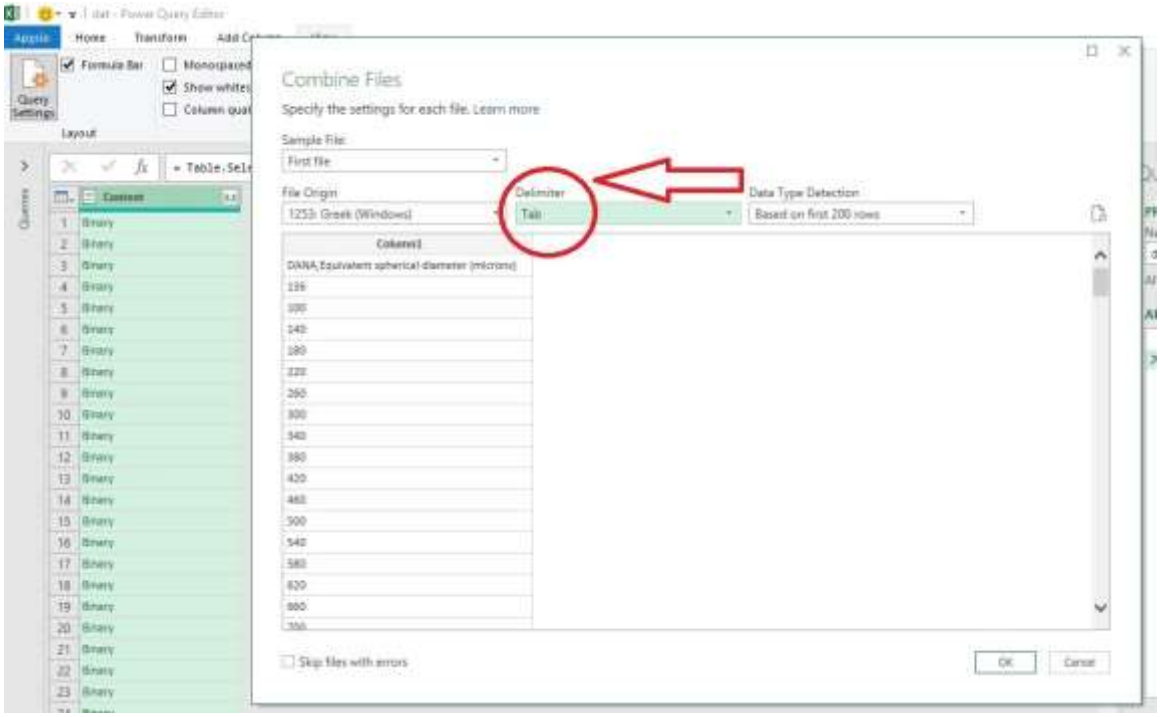


Figure 4

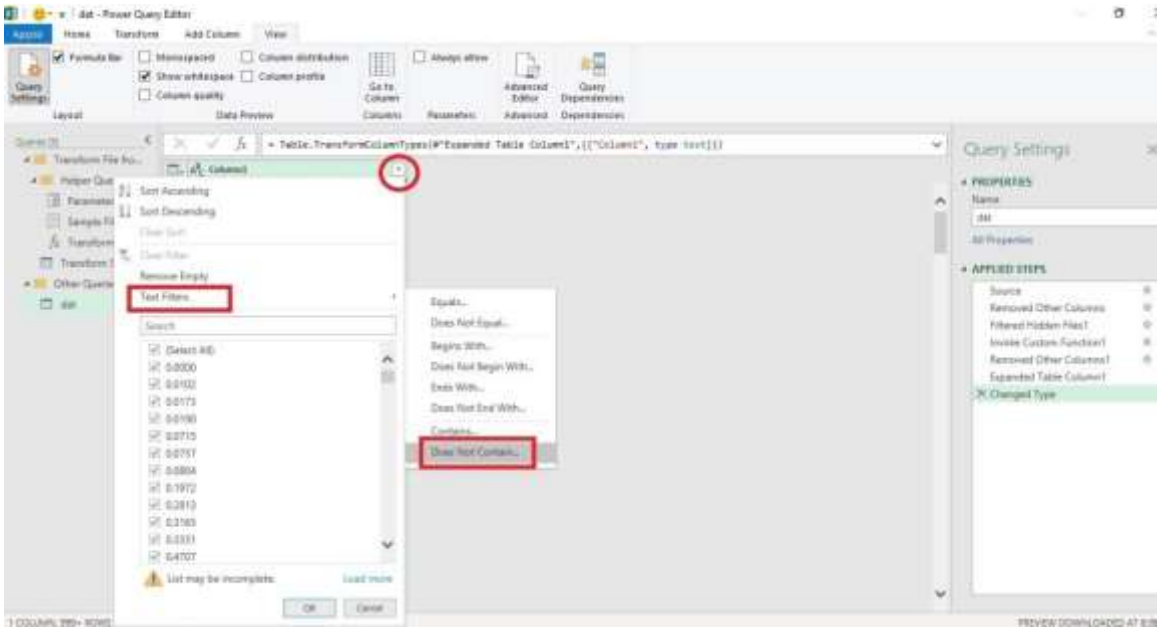


Figure 5

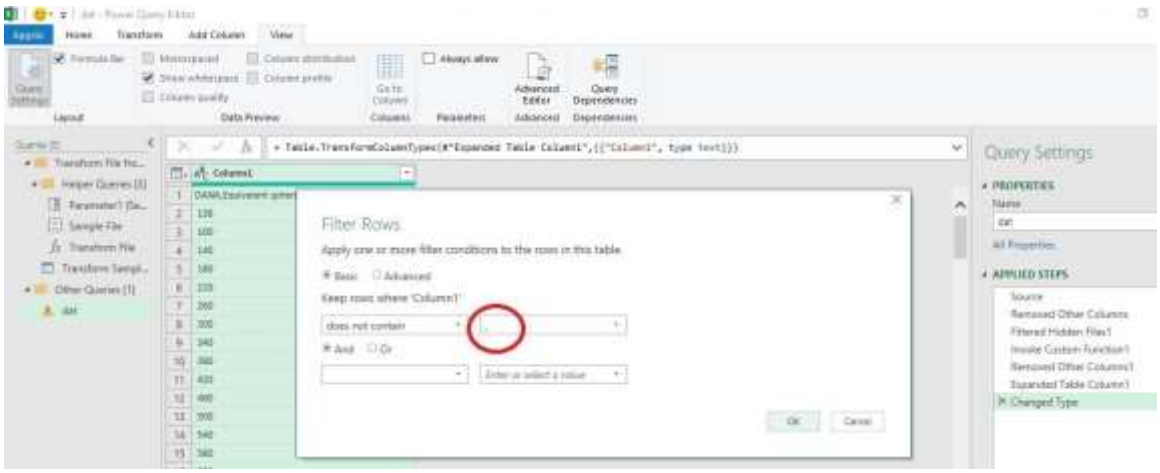


Figure 6

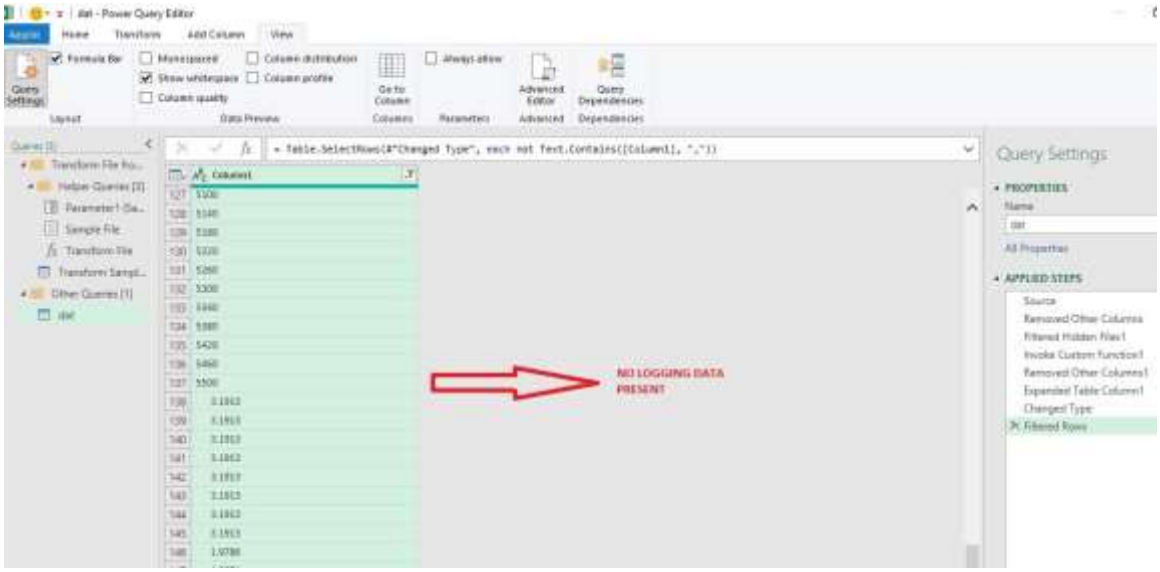


Figure 7

Figure 8

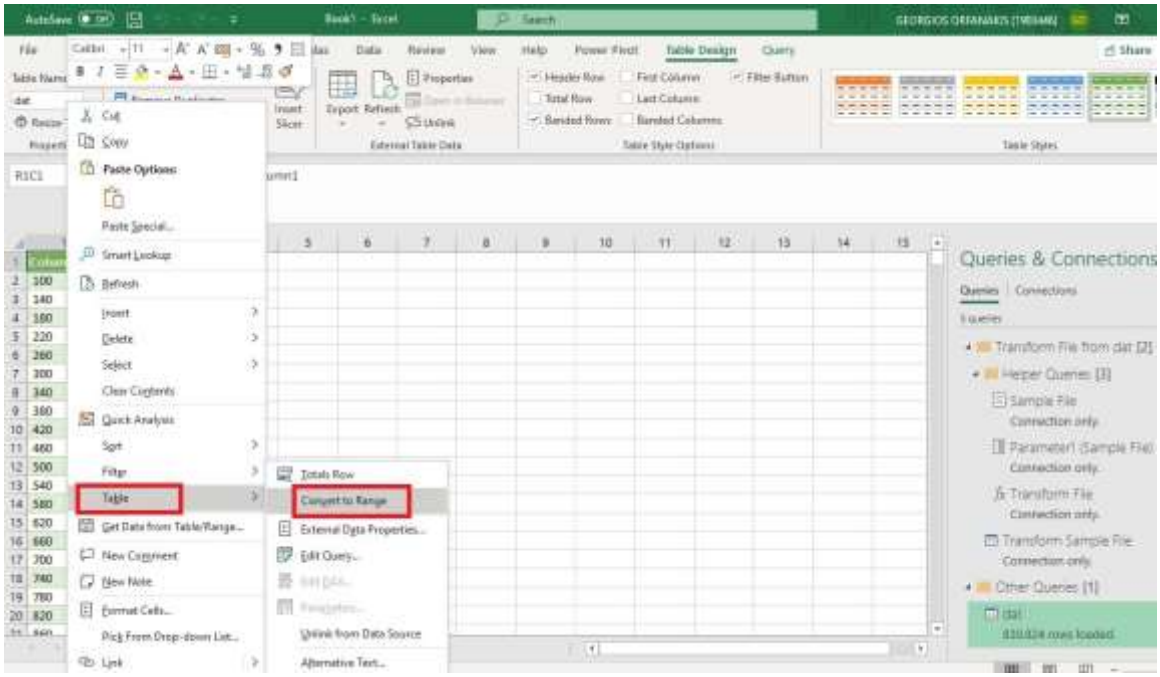


Figure 9

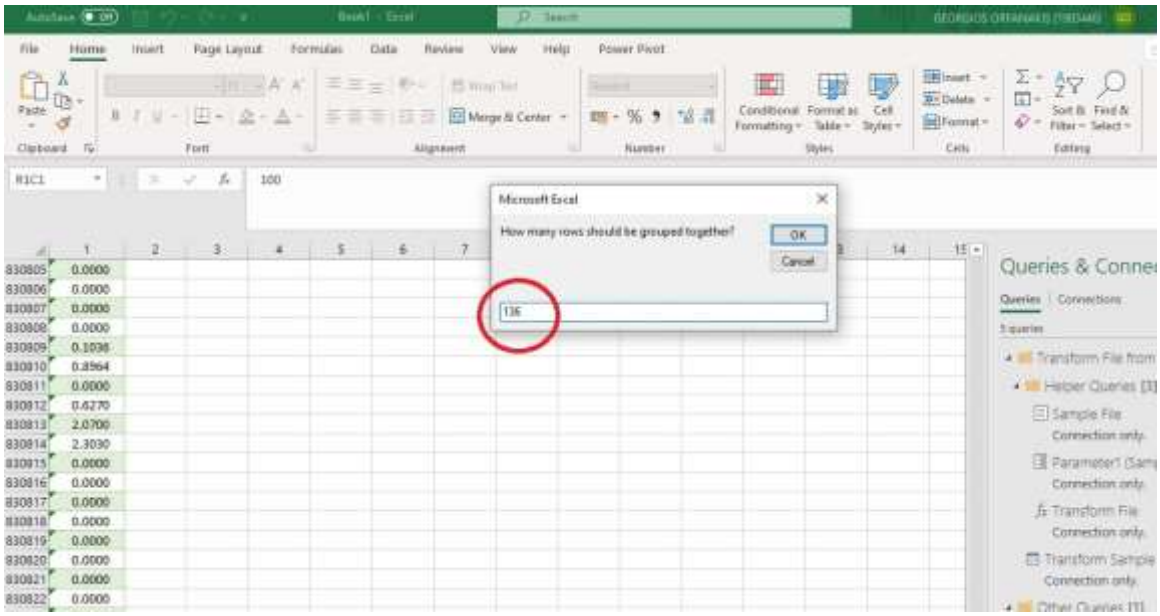


Figure 10

The initial 830824 rows were transformed into 6110 rows. The categories row though is repeated (Figure 11) so it needs to be removed.

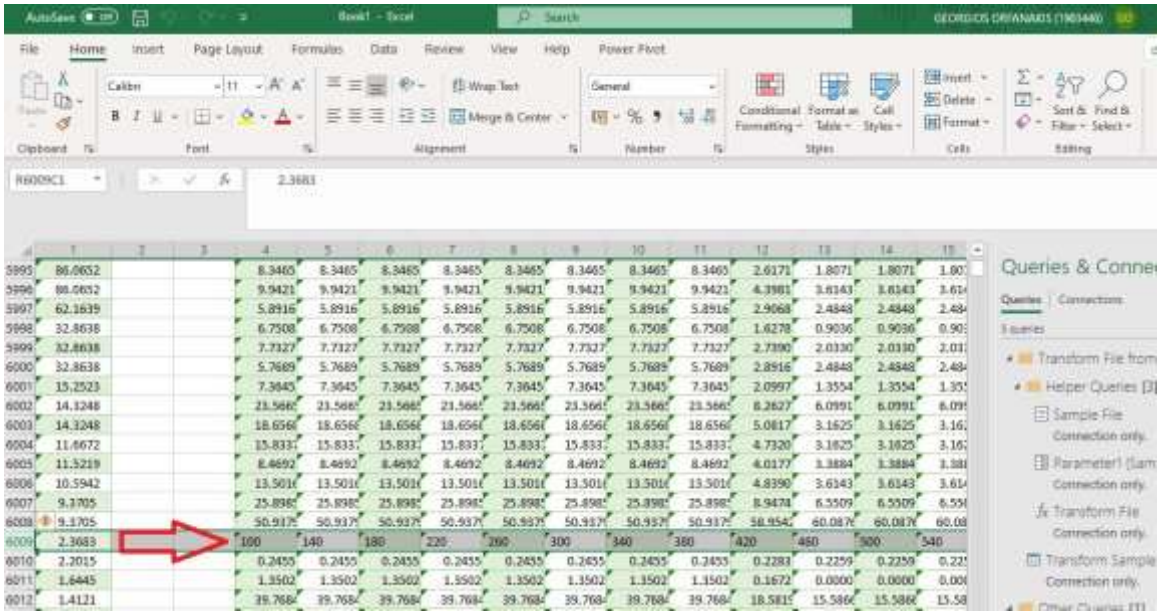


Figure 11

Right click on top of first column→ Delete → OK → Click on top of first left working cell with value ”100” → CTRL+SHIFT+RA then CTRL+SHIFT+DA → All working cells are chosen → CTRL+F → in “Find what” enter 1140, which is one of the categories → click “Find all” –Make sure no value other than the category is in → Click CTRAL+A → hold CTRL and click on the first row in the pop-up window and on values that are not categories to exclude them (Figure 12)→Close window → On Home tool bar click Delete → Delete Sheet Rows (Figure 13). All in between rows that have categories values are omitted.

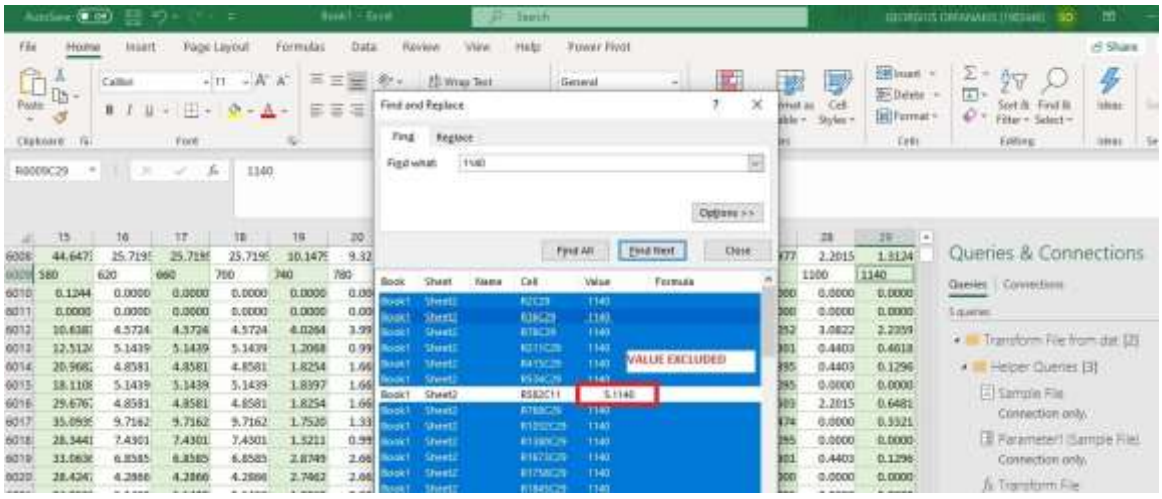


Figure 12

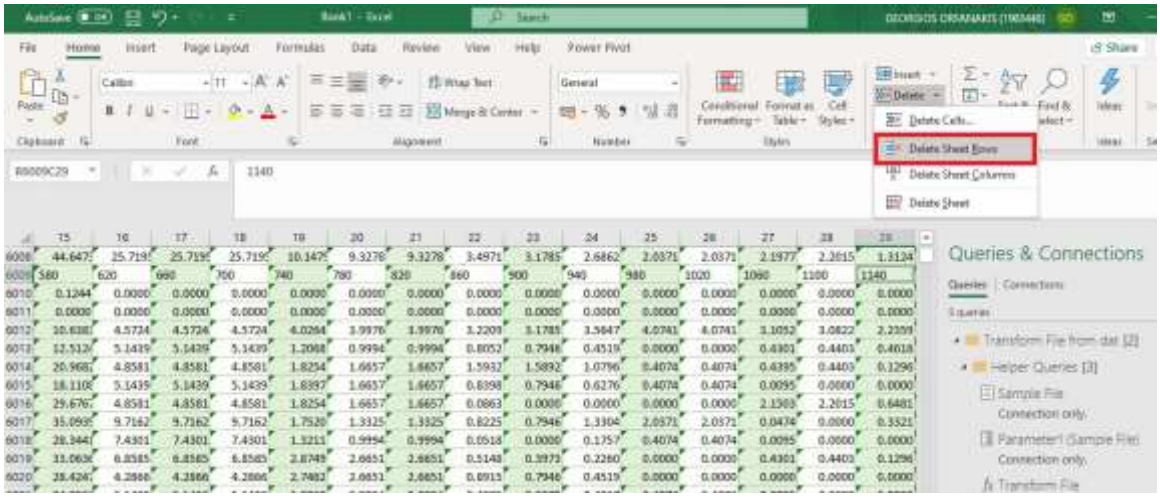


Figure 13

Choose another sheet → at the Data tab click Data → and select the same file following the steps as above. → click the arrow next to column1 → choose “text Filters” (Figure 14) → “Contains” and input “,” and “Does not contain” and input “/” (Figure 15)→ The same with “Does not contain” and enter “D” (Figure 16)→ Close&Load for the column containing the haul number and volume (Figure 17)→ Right click on column1 → Table → Convert to range → OK → CTRL+SHIFT+DA to choose all working cells of first column → On Data tab click “Text to columns” → click the “Delimited” option → Next → choose “comma” option → Next → Choose “Text option” for both columns → Finish → click on first cell of first column → CTRL+SHIFT+DA to choose all working cells of first column → copy paste it as first column on the previous sheet where all the values are (Figure 18)→ Enter column name as “HAUL_NO”. (Figure 19)

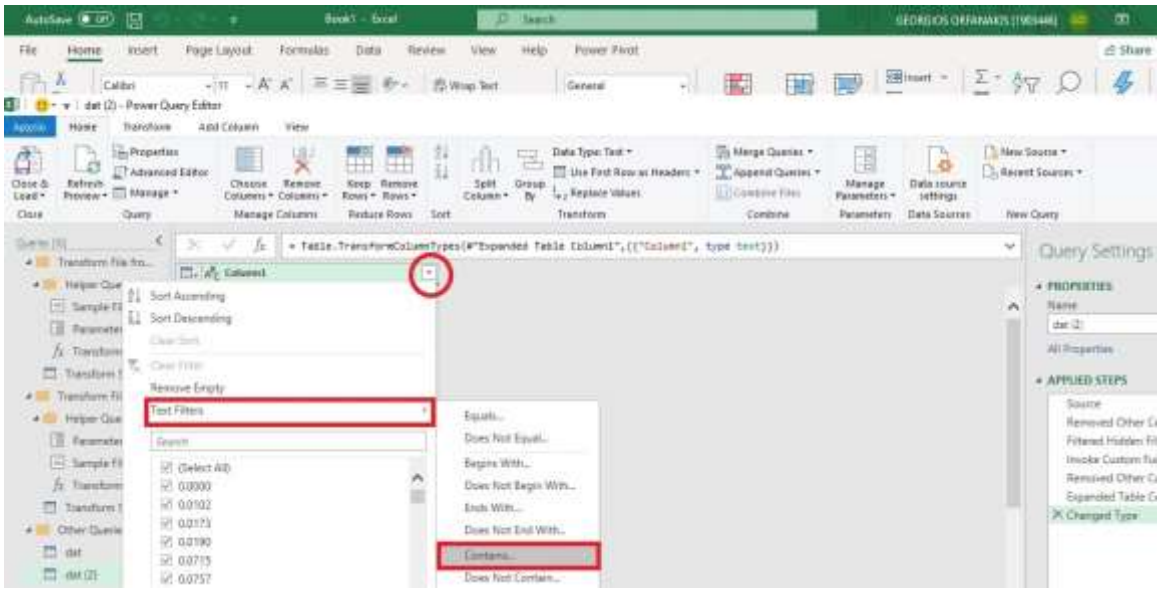


Figure 14

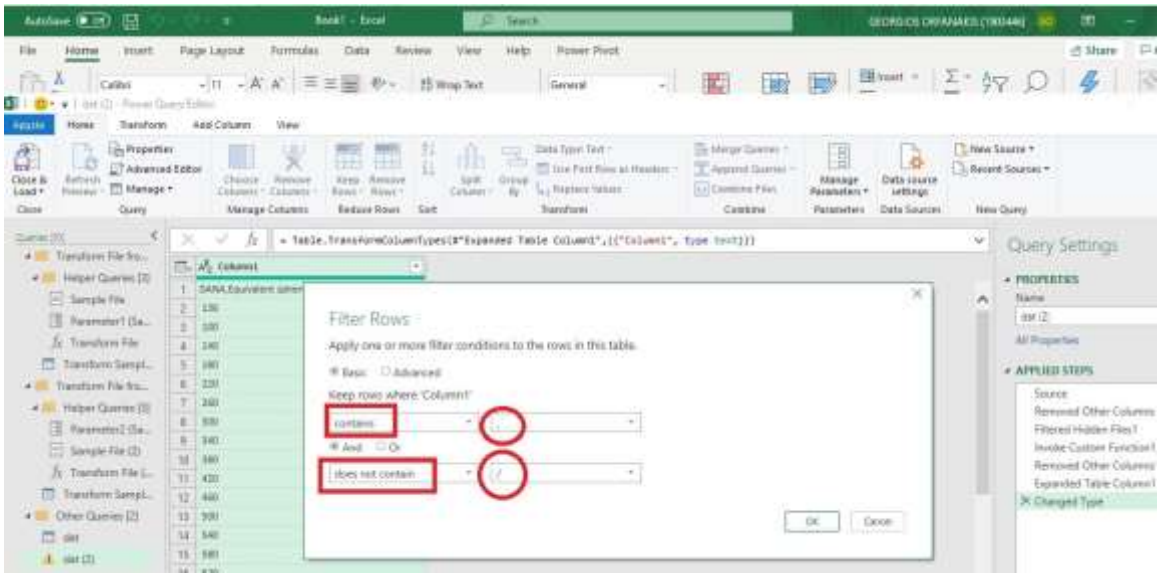


Figure 15

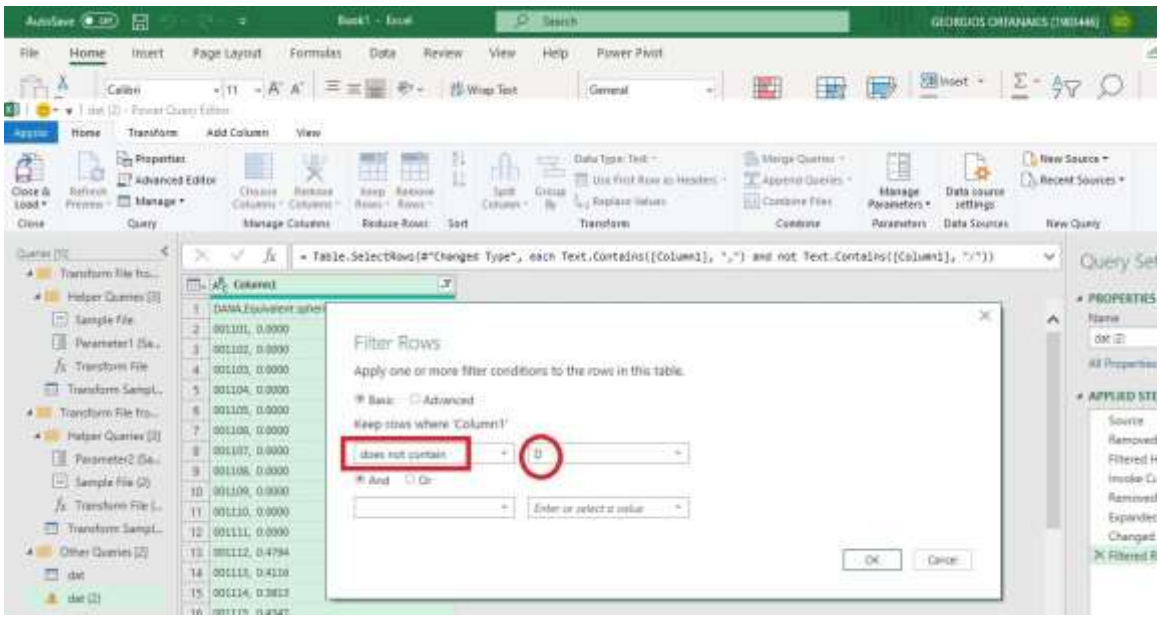


Figure 16

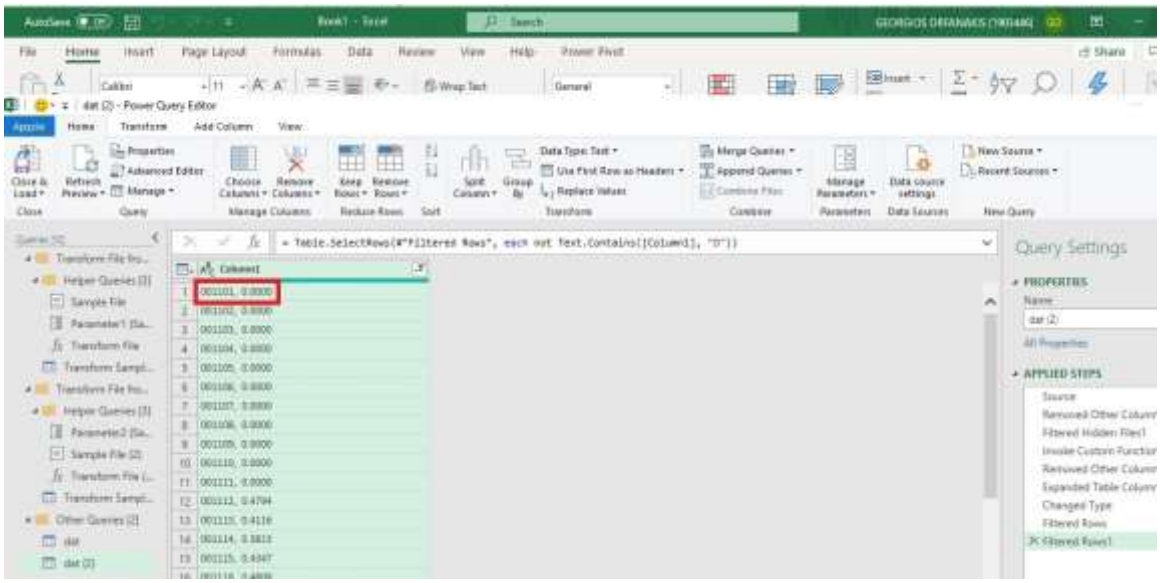


Figure 17

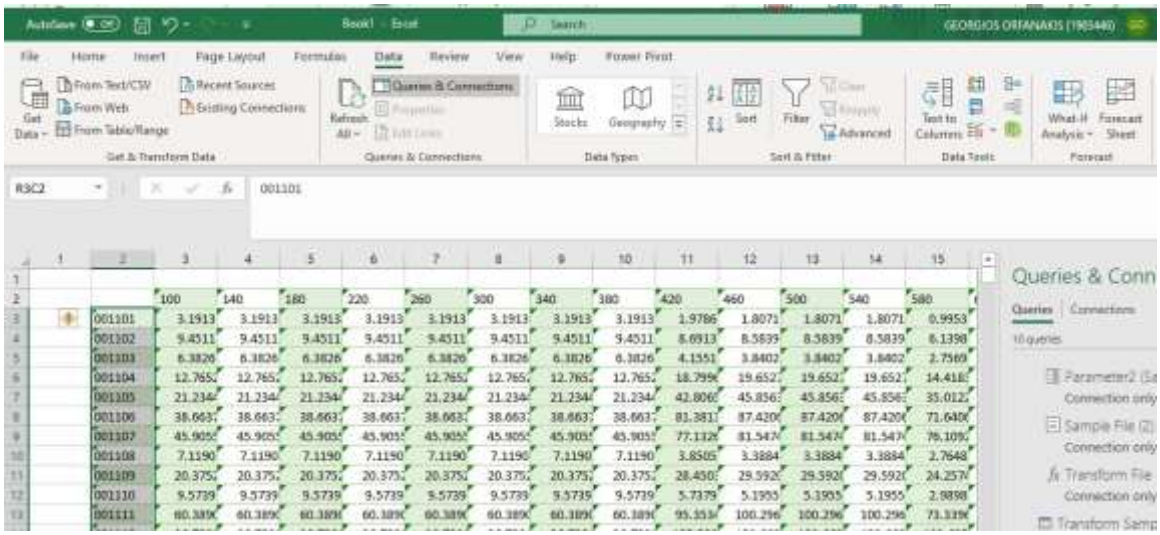


Figure 18

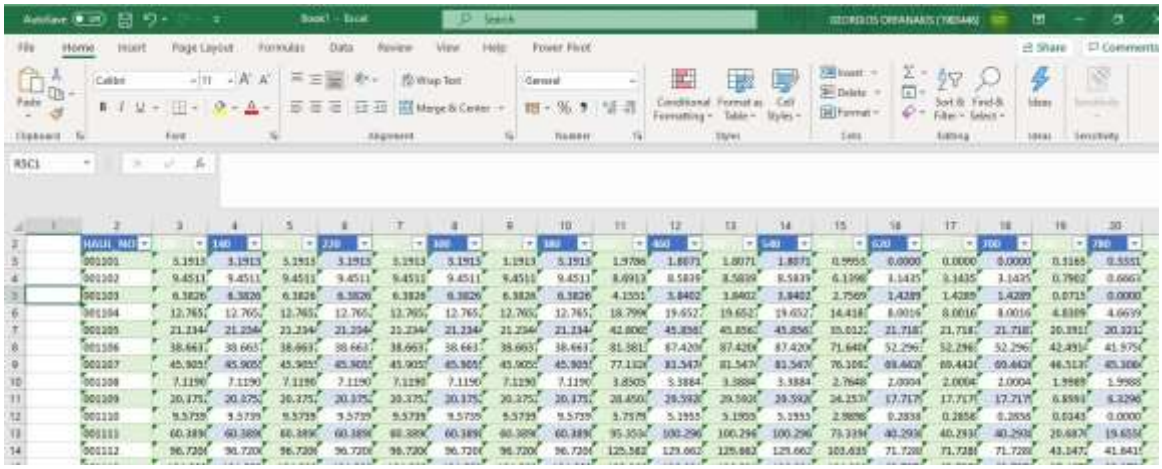


Figure 19

A way to test we are right in this process is at this step to go at the very last working cell of the first column and check if it ends where the values end. (Figure 20) The first column contains the haul numbers so there can be no value without a haul number.

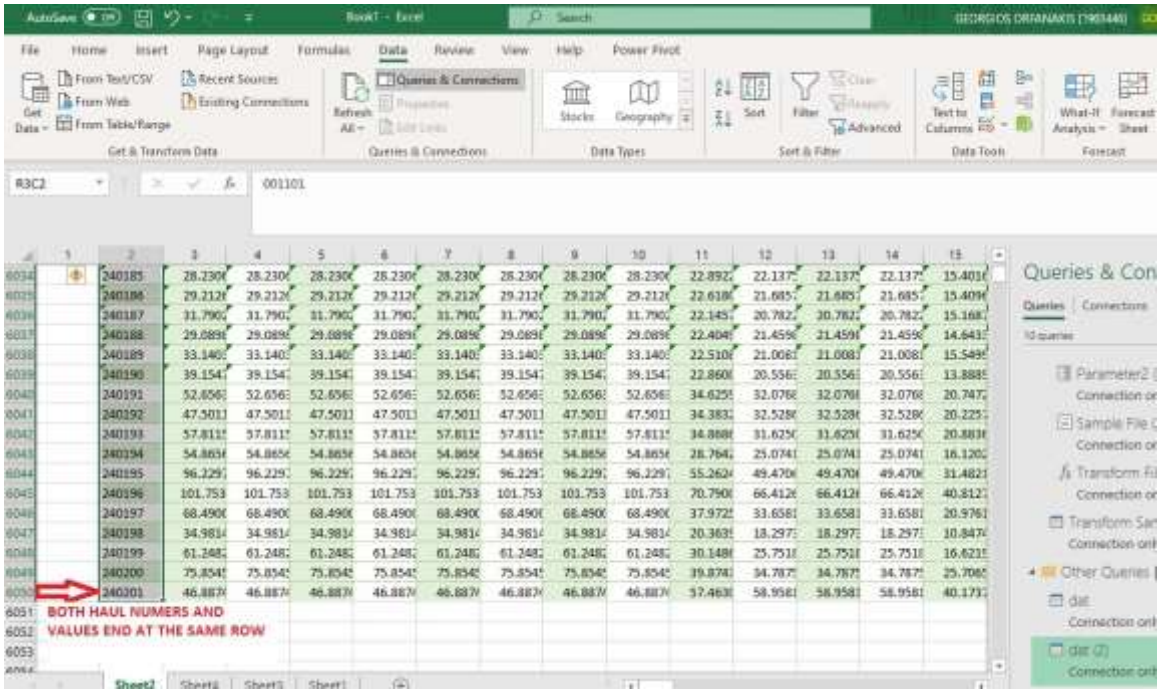


Figure 20

Now, CTRL+SHIFT+RA then CTRL+SHIFT+DA → All working cells are chosen → CTRL+T to turn the working cells into a table with headers → At Data tab click From Table/Range to enter the Power Query → CTRL+A → From Home tab choose Data Type → Text (Figure 21) → Replace current → Right click on HAUL_NO column → Unpivot other columns (Figure 22) → Rename the columns as “CLASS” and “VALUE”(Figure 23) → Close&Load.

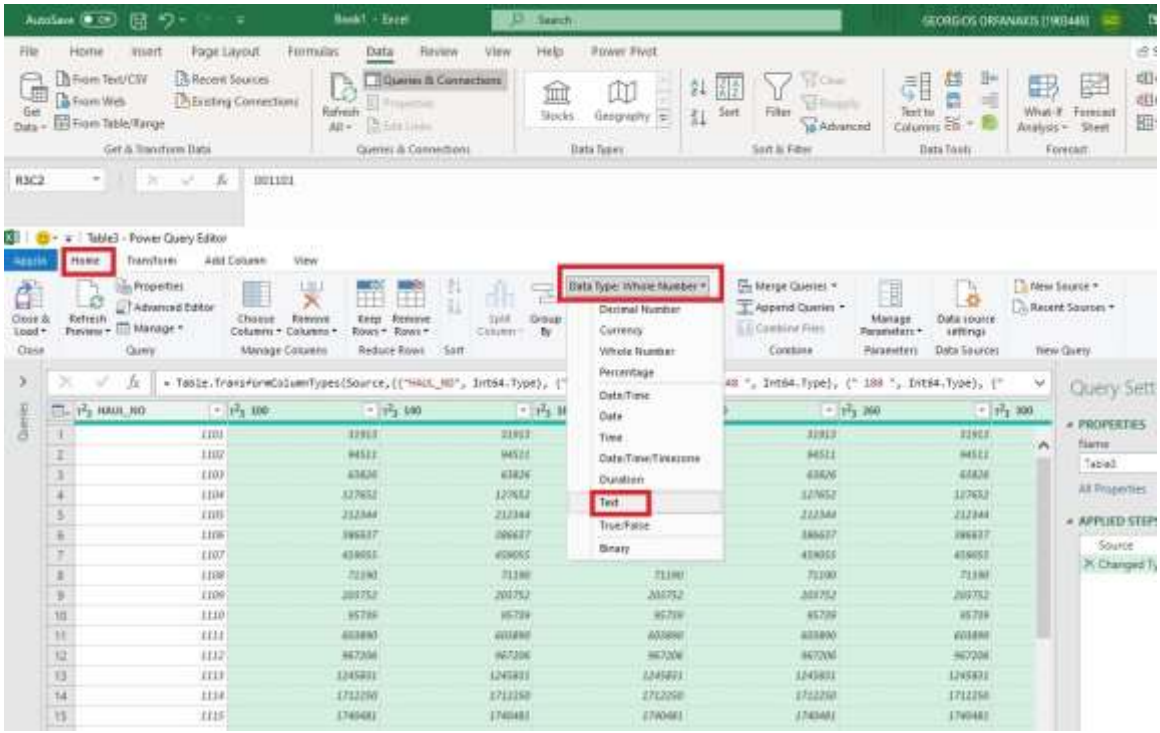


Figure 21

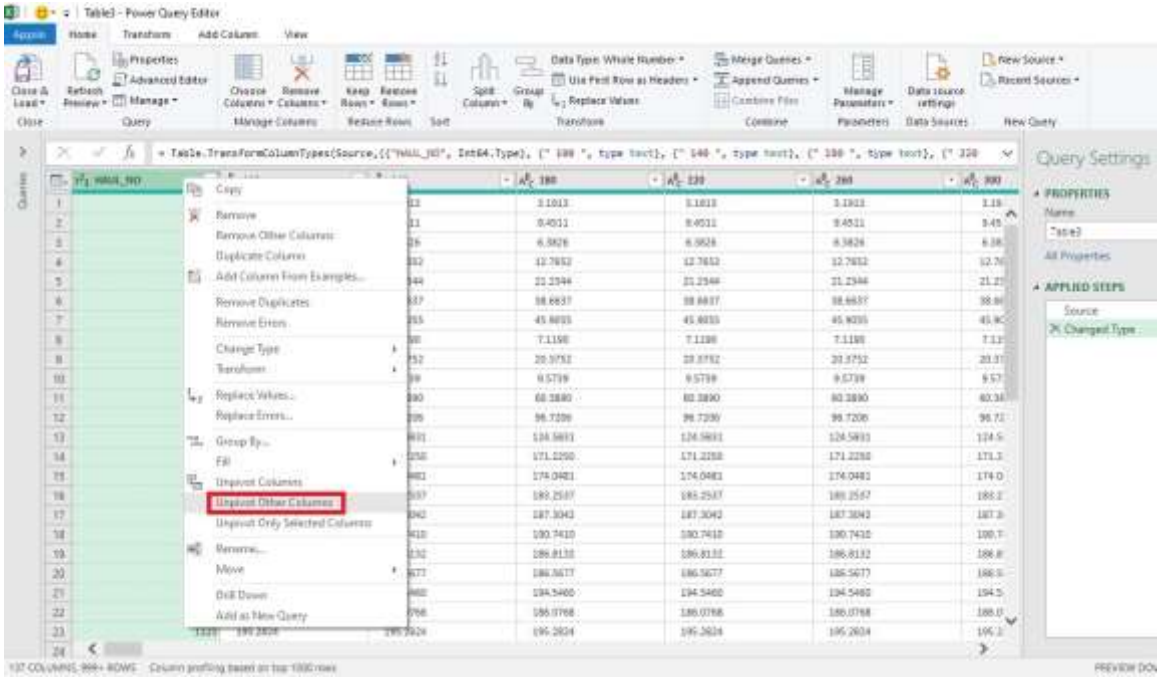


Figure 22

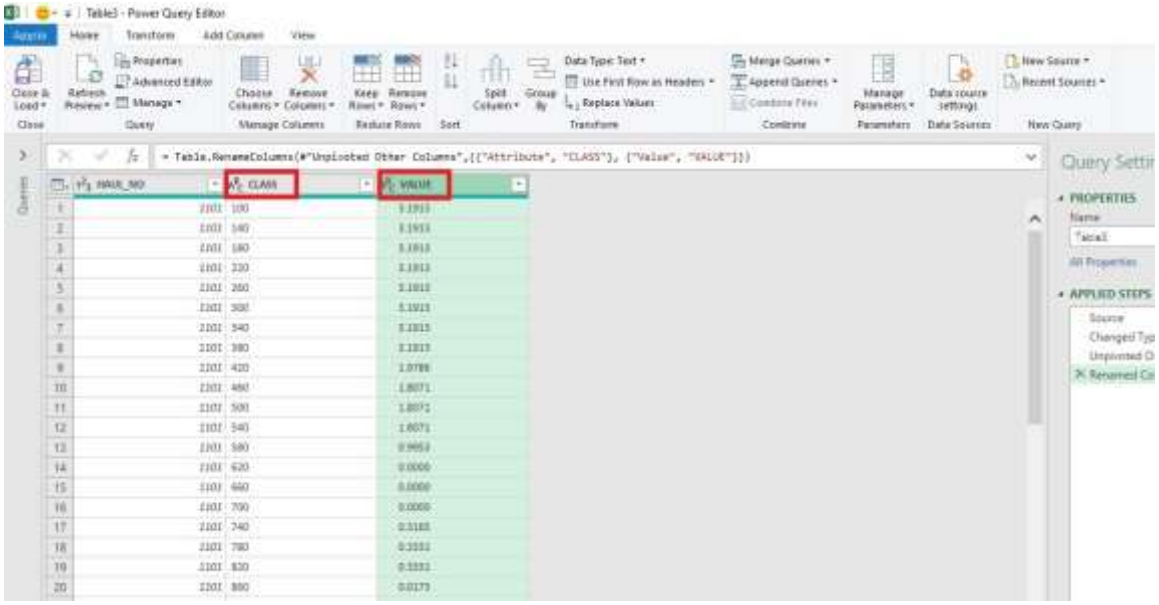


Figure 23

Once the data is loaded in the excel worksheet we insert the columns “TYPE” and “CRUISE_CODE” → fill “TYPE” with “CONCENTRATION” and “CRUISE_CODE” with “D0697” → copy paste “CRUISE_CODE” column as value → Delete all other sheets, but keep HAUL_NO and Volume columns on another excel workbook for later use → Save file with appropriate name eg results_opc_D0697.

<u>Produce excel file with the logging values:</u>

On the new workbook where the HAUL_NO and depth columns where stored on a new worksheet → At Data tab choose “From file” → From folder → OK → Choose DANA0697 → choose Dat → OK → OK → on the pop-up Data preview window click Transform Data → We are in the Power Query Editor → right click on Content column → Chose Remove Other Columns → click Combine files → on Combine files preview screen choose “Tab” as delimiter→ click OK → click the arrow next to column1 → choose “text Filters → “Contains” input “/” (Figure 24) → Close&Load

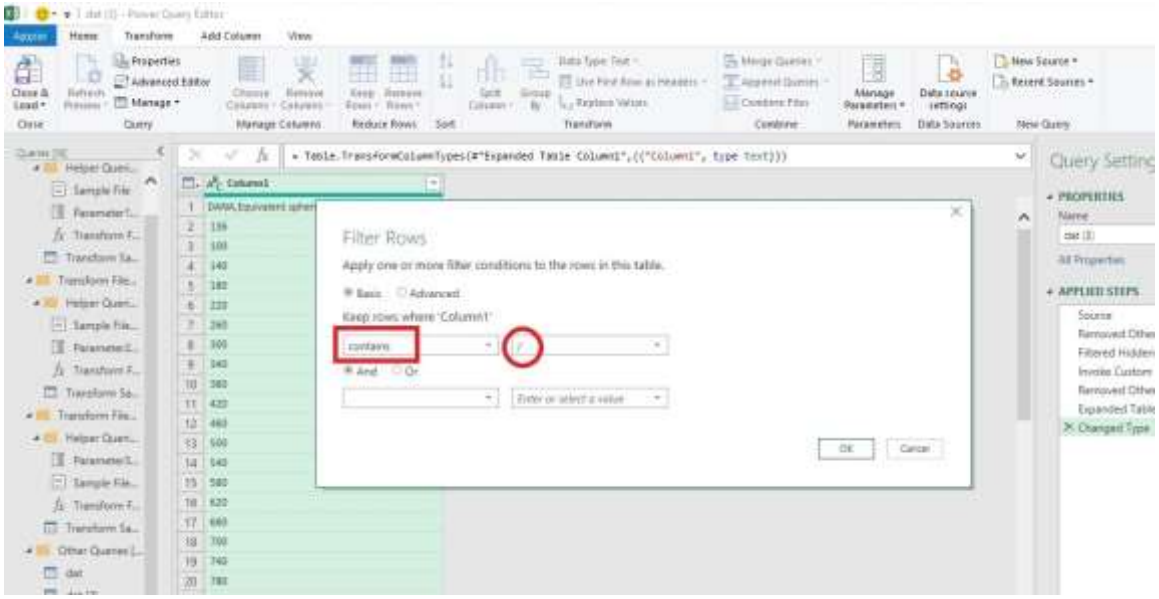


Figure 24

In the Excel sheet where the data is loaded CTRL+SHIFT+DA → All working cells of first column are chosen → At Data tab choose "Text to Data" → Delimited → Next → choose "comma" → Next →choose first column as Date in "DMY" format → Finish.

Now we copy paste at the top row the column names that we have created previously. After this the transformation of each consecutive three rows into 1 begins with Start, Mid and End columns being inserted the values as shown in Figure 25 with subsequent colors. This happens for all contents of Latitude, Longitude, Echo and Sampler_depth.

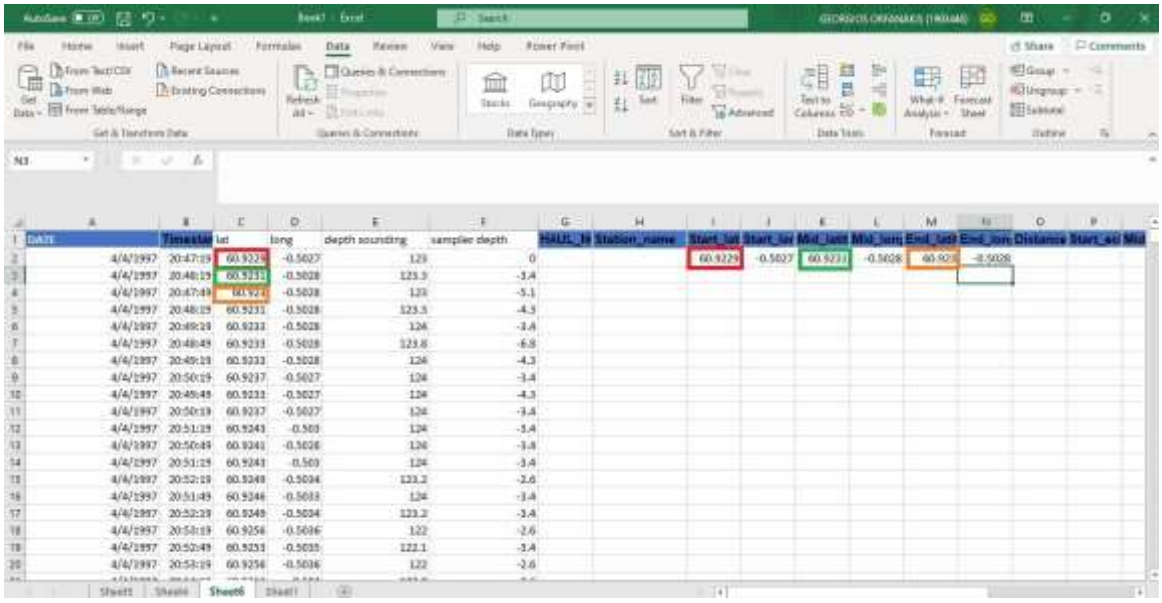


Figure 25

For Duration we calculate the difference between 3rd row and 1st row (Figure 26). We choose the above computed columns → Double click on the down left corner of the last chosen working cell and let excel iterate the calculation → Fill CRUISE_CODE column with D0697 → Timestamp column is chosen with CTRL+SHIFT+DA → right click → format cells → Time → choose "37:50:55" format → Same goes for Duration column →All computed columns are copy-pasted as values → We delete column lat-long → In the column next to

CRUISE_CODE we add on the second and third row the "@" value (Figure 27). We choose the first three rows of the column and double click an let excel iterate the values (Figure 28)→ We choose the first cell of this last column → CTRL+SHIFT+DA → the entire working column is chosen → CTRL+F → enter "@" → CTRL+A (Figure 29)→ at Home tab choose Delete → Delete sheet rows → Every second and third row is gone → the HAUL_NO from the other worksheet is added at the subsequent HAUL_NO column, same goes for depth volume_dec_cubic_m column → The sampler column is filled with "PC" as value → Valid_Y_N column is filled with "Y" as value → All other sheets are deleted → Excel file is saved with appropriate name e.g. records_opc_D0697

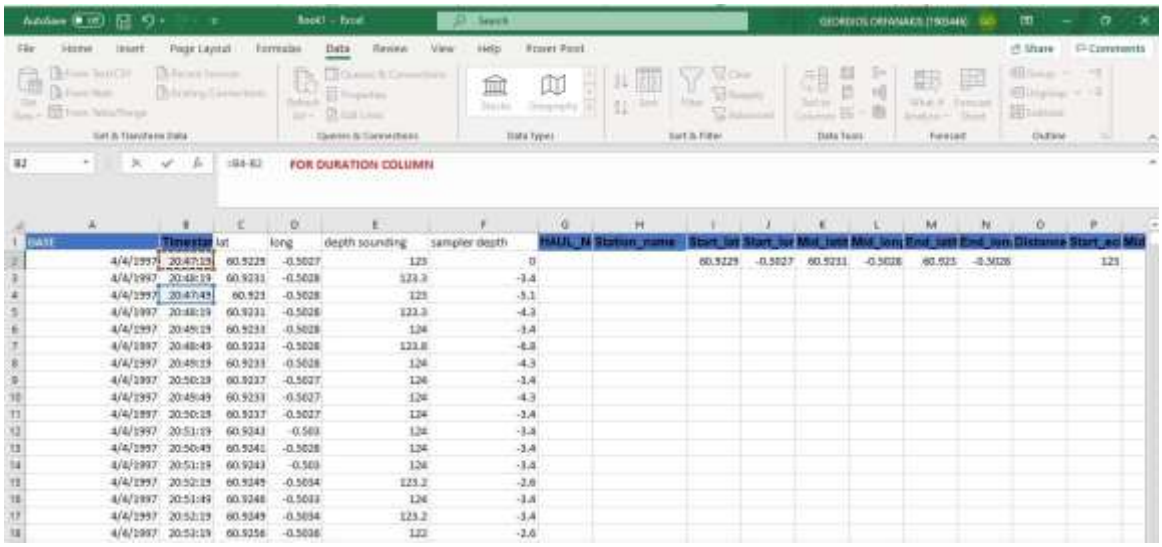


Figure 26

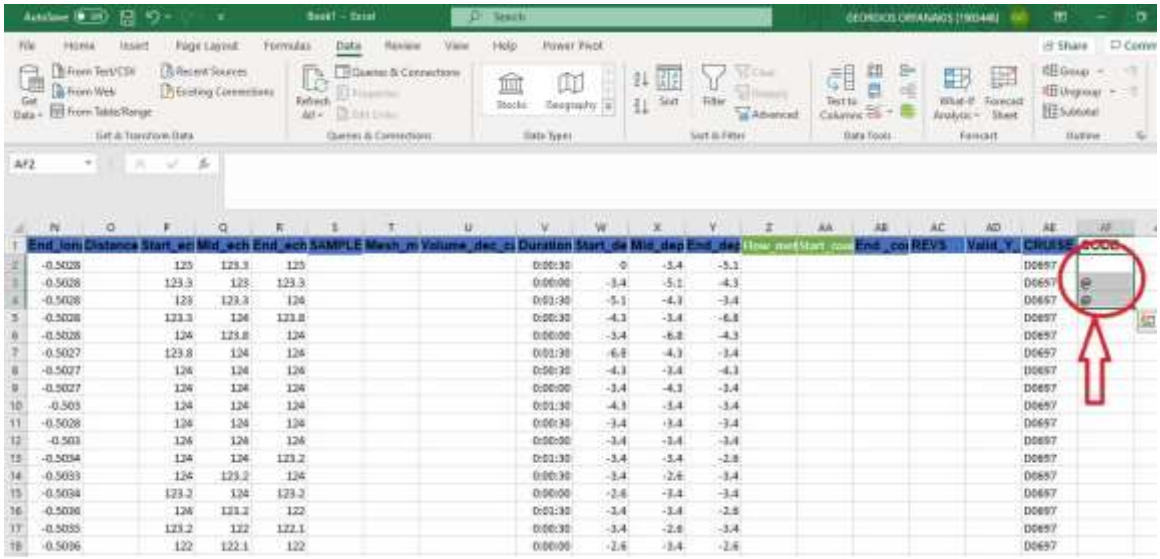


Figure 27

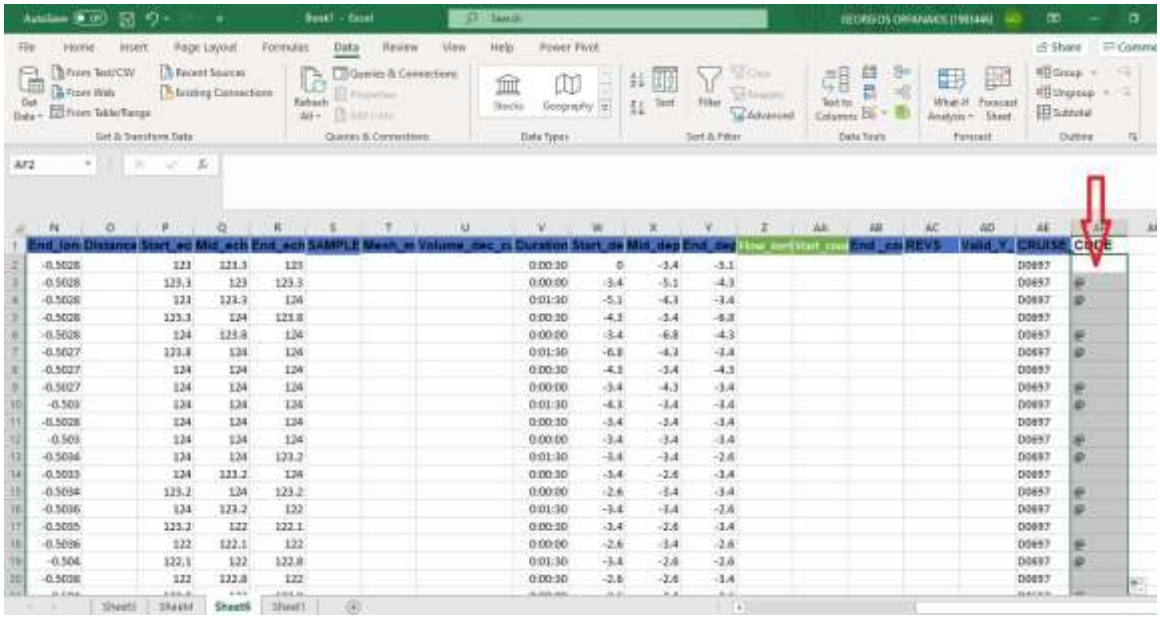


Figure 28

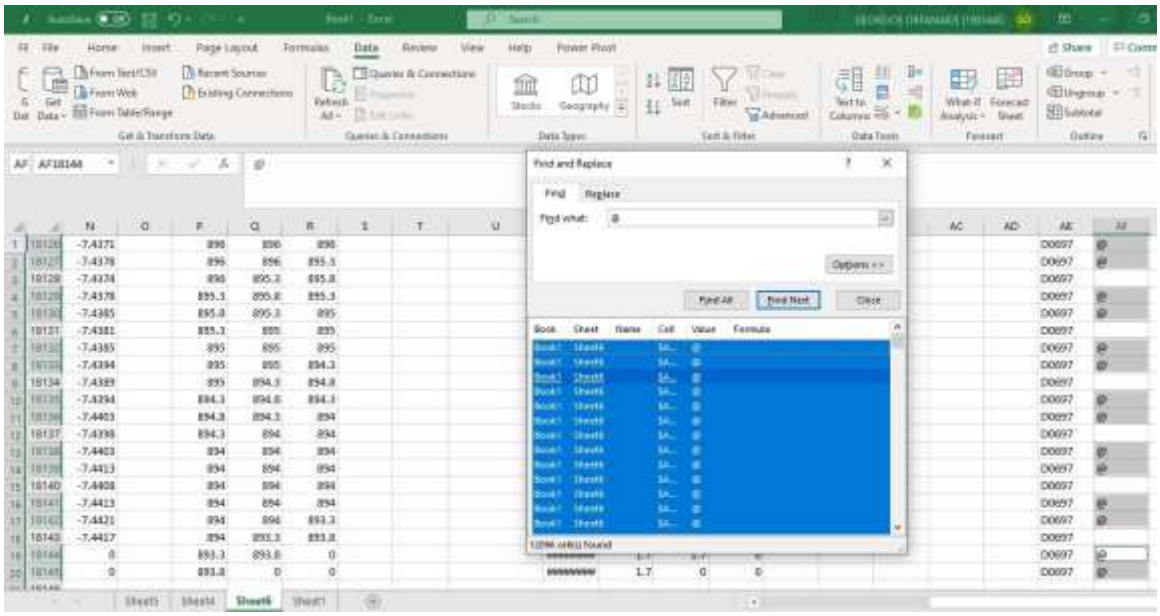


Figure 29

A way to check that the deletion has been achieved is by checking whether the HAUL_NO column's values end at the same row as the rest pre-existing values (Figure 30)

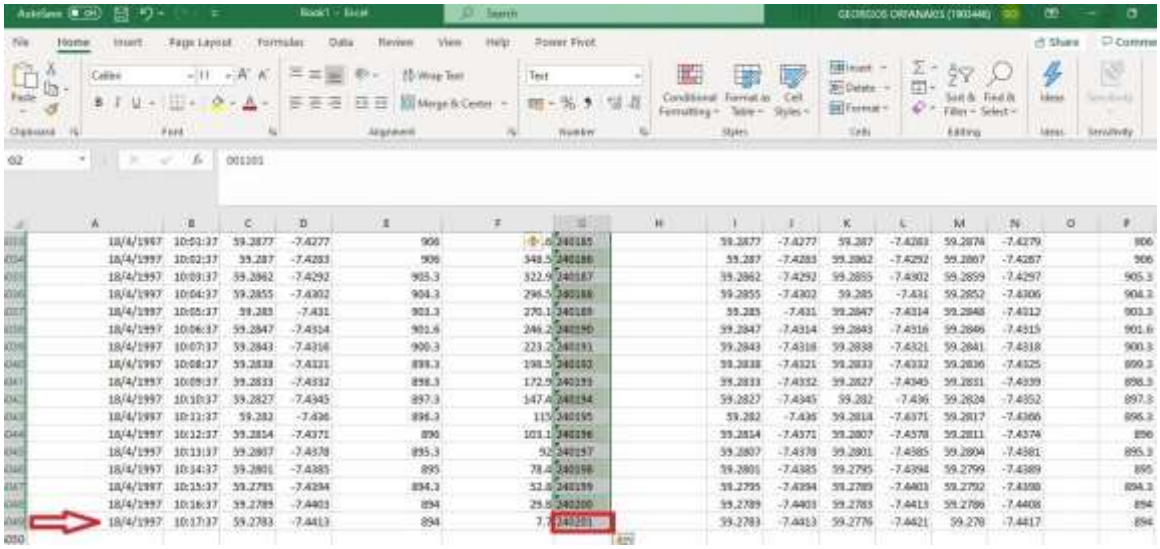


Figure 30

For all the other collections (ARIES, MICD, OCEAN) records and sample data are stored in different files. Records are stored in pla/plw files and sample data is in excel, csv or other text format file from which it is derived. This constitutes the unstructured part of the data and it is not possible to depict each case in this report so two distinct and most common cases are shown here for sample data extraction.

Produce Excel files from ".pla/.plw" files

Pla/plw files:

On an Excel worksheet follow phase_1 and all data is imported in Power Query. Click on Filters and choose "Does not contain" with "/". Filter out 0,0 by unticking it. Hit Close&Load. The data is loaded as one column in the Excel worksheet. CTRL+SHIFT+DA all data is chosen. On the Data tab we choose Text to Columns. On the pop-up window we choose Delimited and hit Next. Then choose "comma". The Next. Then for the 1st column we choose Date as DMY. Hit Finish. Then we Copy Paste the columns Names on the first row. CTRL+SHIFT+RA then CTRL+SHIFT+DA to choose all working cells. CTRL+T to turn them into a table with headers. Then we need to transform the Timestamp column. For this, 4 extra columns are created after the Timestamp column, which has been formatted into a text column and its title has been deleted (Figure 31). For the first new column the function TEXT(B2;"0000") is used to make sure all numbers have 4 digits (Figure 31). If a number is missing a digit, then 0 is added in the beginning. The result of this column is copy-pasted as Value on the next column. The third new column is using the function Timestamp (Figure 31)* to turn the previous column cells into actual time. Specifically, the function takes the text 0653 and brings it in the form "06:53", then converts it to actual time. The columns selected, then follows right click, format cells, time, and choose format 37:30:55 (Figure 32). The cells of this column are pasted as copy-pasted as values onto the next column which gets the name Timestamp. The initial column and all the ones created except for the last one are deleted. We then fill the CRUISE_CODE column with S1801. Then we save the Excel workbook with just the working sheet using an appropriate name e.g. pla_AR_records.

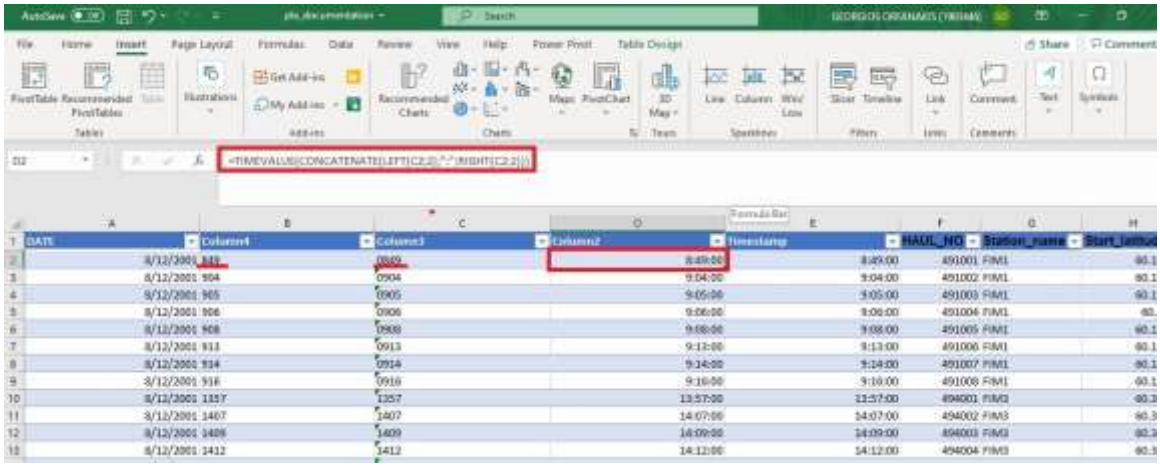


Figure 31

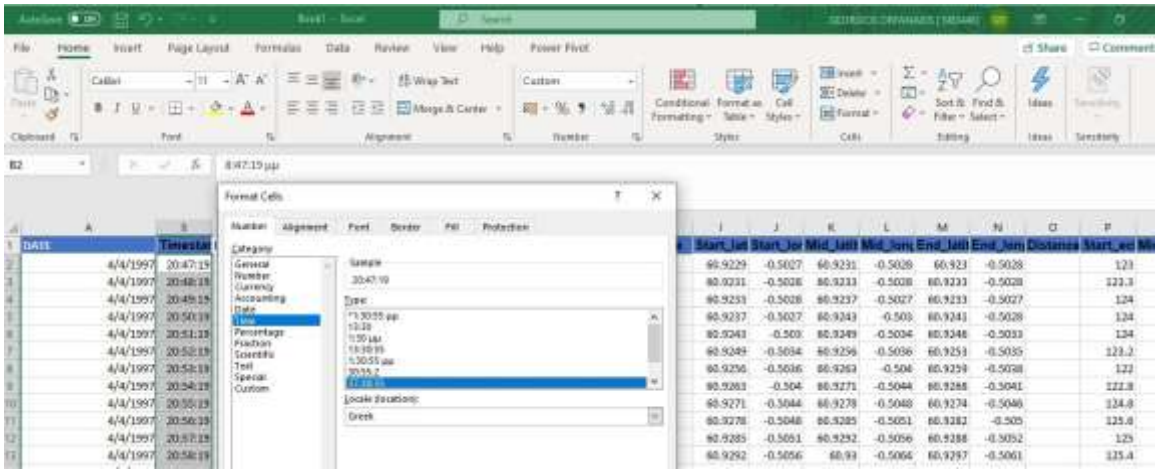


Figure 32

Produce Excel file from unstructured files

The results from an ARIES dataset can be rather unstructured, an example of which we see here(Figure 33). All data Chrons and Concentrations are copy pasted as values on a new workbook. All cells of the Chrons column are selected. At the Home tab we choose Fill&Select, Go To Special, then select blanks option. At the Home tab we choose Delete, Delete Sheet Rows. All rows that were initially empty are deleted. The empty columns are deleted manually. All working cells are selected, CTRL+T to turn the working cells into a table with headers. All columns are selected, then from Data tab we choose from Table/Range to enter the data into Power Query. All column except for the first are selected and from the Data Type option we choose Text. We then choose the 1st column. Right click and Unpivot other columns. Rename 1st column as HAUL_NO, Attribute column into CLASS and Value column into VALUE. Close&Load. In the Excel worksheet we choose the VALUE column and from Home tab we choose Find&Select, Replace, in Find What we enter "," and in "Replace with" we enter ".". Then hit Replace All. Then we select the entire column, right click, format cells, Number and fill decimal places with 6. From the Data tab we choose Text to columns and in the pop-up window we hit Finish. Then we add 2 columns, one is named TYPE and filled with "CONCENTRATION" and one named CRUISE_CODE and filled with S1801. The cruise code is copy pasted as value on the same column. The entire column is selected, right click, format cells,

Text option is chosen. From the Data tab we choose Text to columns and in the pop-up window we hit Finish. Then we save the workbook keeping only the working sheet with the appropriate name e.g. aries_results_S1801.

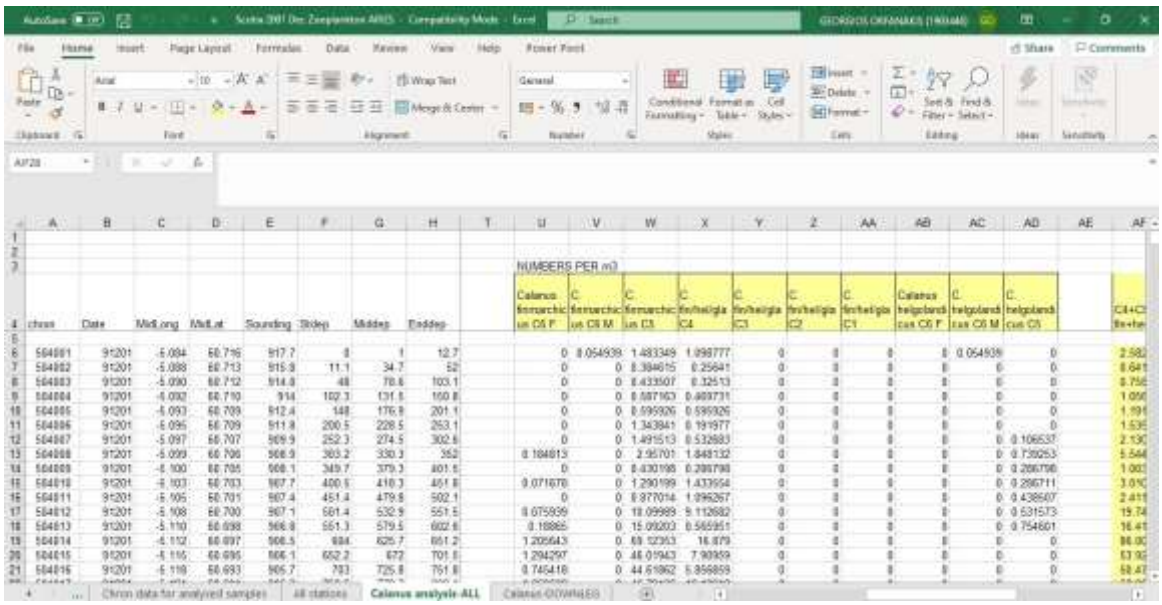


Figure 33

Appendix-Data Wrangling in Python

Import Packages

```
# PACKAGES IMPORT AND CHECK OF DIRECTORY USED
import os
import glob
import pandas as pd
from pathlib import Path
import csv
os.getcwd() #check we are at the right directory, where files are kept
```

Two step process: excel files to csv & merging of csv files

Before the code below runs, we need to copy paste all excel files of a certain collection e.g. ARIES into the folder in Jupyter Notebook that has been created for this procedure. The directory of the files could have been used directly, but due to reasons of data safety this path was chosen to be followed. This makes sure that the excel files created via the data wrangling process are kept intact.

```
# TWO STEP EXCEL FILES TO CSVS & MERGE OF CSVS TO ONE CSV FOR BULK INSERT
```

```
## STEP 1
### massively transform xlsx/xls to csv
for file in glob.glob("*.xlsx"):

    df = pd.read_excel(file)
    #base=os.path.basename(file)
    p = Path(file).stem

    df.to_csv(p+'_csv'+'.csv', index=False, header= True)

    os.remove(file)
```

```
## STEP 2
### combine all csv files into 1 csv file
extension = 'csv'
all_filenames = [i for i in glob.glob('*.{}'.format(extension))]
#combine all files in the list
combined_csv = pd.concat([pd.read_csv(f, header = 0) for f in all_filenames ])
#export to csv
combined_csv.to_csv( "allrec.csv", index=False, encoding='utf-8-sig')
#print merged csv
print(combined_csv.head(1000))
```

Number of decimals changed to six

The number of decimals in columns containing needs to be down to 6 according to Marine Scotland request.

```
# CHANGE NUMBER OF DECIMALS ON FILES
#read in initial file
dataset = pd.read_csv('allrec.csv')
#returns input if failure occurs (e.g. if not a float)
def try_cutoff(x):

   try:
```

```
        return round(float(x), 6)
    except Exception:
        return x
#loop over each column and apply try_cutoff to each element in each column
#IF VALUE COLUMN ONLY REMOVE '#' AND PUT A '#' IN SECOND hlist lines

#hlist = ["VALUE"]
hlist = ['Start_latitude', 'Start_longitude', 'Mid_latitude', 'Mid_longitude', 'End_latitude', 'End_longitude','Start_echo', 'Mid_echo', 'End_echo', 'Volume_dec_cubic_m','Start_depth', 'Mid_depth', 'End_depth']
for field in hlist:

        dataset[field] = dataset[field].map(try_cutoff)

#write new dataset result to CSV file
dataset.to_csv("allrec_file.csv", index = False)
```

The code above runs for a ARIES, OPC, OCEAN and MIKD collections separately. Then the records csv file for each collection is put into the folder and the **##STEP2** and **#CHANGE NUMBER OF DECIMALS ON FILES** is run again creating a csv file that contains records of all collections. The same procedure is followed for sample data too.

Drop column in for csv containing SAMPLES data

At first the student was not sure if the volume column would be kept in the Samples data of the OPC collection and if the volume would need to be imported as column other collections too, so the OPC samples csv file created that had all OPC Sample data in it had this column too. After discussions with the client it was decided that the column would be dropped. For this to come true the following code is run for the OPC sample data csv file.

```
# DROP COLUMN IN FINAL CSV
f=pd.read_csv("samples_filename.csv ")
keep_col = ['HAUL_NO', 'CLASS', 'TYPE', 'VALUE', 'SAMPLER', 'CRUISE_CODE']
new_f = f[keep_col]
new_f.to_csv("newFile.csv", index=False)
```

Create empty column for SAMPLES csv file for future FK population

The csv file containing the sample data of all collections needed the adding of an empty column at the end, where the foreign key would be later on created. So, the following code ran.

```
# CREATE EMPTY COLUM AT THE END OF FILE TO IMPORT TO STAGE_SAMPLES TABLE

##THE EMPTY COLUMN MATCHES THE FK_REC_ID COLUMN OF THE TABLE

import pandas as pd

f = pd.read_csv("samples_filename.csv")

f["FK_REC_ID"] = ""

f.to_csv("samples_filename.csv ", index=False)
```

Create empty column for SAMPLES csv file for alternative database for future incremented PK population

For the case of the alternative database one more step needs to be added at this very point, which is adding another column at the beginning of the dataset as shown in the code below:

```
# INSERT EMPTY COLUMN AT THE BEGINNING OF THE csv FILE

g = pd.read_csv("samples_filename.csv ")

g.insert(0, 'PK_staging_samples', '', allow_duplicates=True)

g.head()

g.to_csv("newStaginSamplesFile.csv", index=False)

g.head()
```

Count number of rows in csv files

Counting the number of rows is important to make sure that all rows are uploaded in the database by comparing the number from python scripts to the number from SQL SERVER counting of rows.

```
# COUNT NUMBER OF ROWS IN CSV FILE-TESTING

with open('allrec_file.csv') as f:

    z = sum(1 for line in f)

    print(z)
```

Appendix-Zooplankton Database Creation

After the steps above two csv files have been created: one for the RECORDS table and one for The SAMPLES table. We open the SSMS18 → right click on databases → create new database → accept all default parameters and enter “Zooplankton” as the name of the database → Right click on the database → choose New Query. In the query tab opened the following code is inserted:

STAGE_RECORDS, bulk insert & RECORDS table

```
/* Entire database creation*/

---Create STAGE_RECORDS table

CREATE TABLE [dbo].[STAGE_RECORDS](
	[DATE] [date] NOT NULL,
	[Timestamp] [time](0) NOT NULL,
	[HAUL_NO] [int] NOT NULL,
	[Station_name] [nvarchar](50) NULL,
	[Start_latitude] [decimal](18, 6) NULL,
	[Start_longitude] [decimal](18, 6) NULL,
	[Mid_latitude] [decimal](18, 6) NULL,
	[Mid_longitude] [decimal](18, 6) NULL,
	[End_latitude] [decimal](18, 6) NULL,
	[End_longitude] [decimal](18, 6) NULL,
	[Distance] [nvarchar](50) NULL,
	[Start_echo] [decimal](18, 6) NULL,
	[Mid_echo] [decimal](18, 6) NULL,
	[End_echo] [decimal](18, 6) NULL,
	[SAMPLER] [nvarchar](50) NOT NULL,
	[Mesh_micro_m] [nvarchar](50) NULL,
	[Volume_dec_cubic_m] [decimal](18, 6) NULL,
	[Duration] [time](0) NULL,
	[Start_depth] [decimal](18, 6) NULL,
	[Mid_depth] [decimal](18, 6) NULL,
	[End_depth] [decimal](18, 6) NULL,
	[Flow_meter_type] [nvarchar](50) NULL,
	[Start_count] [nvarchar](50) NULL,
	[End_count] [nvarchar](50) NULL,
	[REVS] [nvarchar](50) NULL,
	[Valid_Y_N] [nvarchar](50) NULL,
	[CRUISE_CODE] [nvarchar](50) NOT NULL
) ON [PRIMARY]
```

```
GO


---Bulk insertion for STAGE_RECORDS table

BULK INSERT [dbo].[STAGE_RECORDS]

FROM
'C:\Users\georgios\Documents\placem\RESULTS_CSV_COMBINED\work_files\allrec_COMBfile_THE_OFFICIAL
_WORKING.csv'

WITH (FIRSTROW = 2,

   FIELDTERMINATOR = ',',

            ROWTERMINATOR='\n' ,

             MAXERRORS=4000000,

             BATCHSIZE=40000);


---Count rows in STAGE_RECORDS table

SELECT COUNT(*) FROM [dbo].[STAGE_RECORDS];


---Check for duplicates in STAGE_RECORDS table and then delete them

/*First we run the query with the SELECT* end statement. Then comment that statement out and uncomment
the DELETE FROM statement and re-rerun the query with it.*/

WITH CTE ([HAUL_NO],

                        [SAMPLER],

                        [CRUISE_CODE],

                        duplicatecount)

AS (SELECT [HAUL_NO],

                          [SAMPLER],

                          [CRUISE_CODE],

        ROW_NUMBER() OVER(PARTITION BY [HAUL_NO],

                                                    [SAMPLER],

                                                     [CRUISE_CODE]

        ORDER BY [HAUL_NO]) AS DuplicateCount

   FROM [dbo].[STAGE_RECORDS])

SELECT * FROM CTE WHERE DuplicateCount <> 1; /* Some duplicated have been inserted on purpose for this
query to show it is working*/

---DELETE FROM CTE WHERE DuplicateCount <> 1;

---(746 rows affected)
```

```
---COUNT ROWS IN STAGE_RECORDS TABLE TO CHECK THE CHANGE
SELECT COUNT(*) FROM [dbo].[STAGE_RECORDS];
```

It is important to notice that duplicates do exist. This is not due to a problem of the data wrangling process. Some initial Excel files created had the contained data duplicated in order to the test scripts above to run and verify that they are working. This way, when Marine Scotland will use the database for future data entry, they know that they can trust this mechanism to find duplicate records.

```
---Create RECORDS table
CREATE TABLE [dbo].[RECORDS](
        [RECORDS_ID] [int] IDENTITY(1,1) NOT NULL PRIMARY KEY,
        [DATE] [date] NOT NULL,
        [Timestamp] [time](0) NOT NULL,
        [HAUL_NO] [int] NOT NULL,
        [Station_name] [nvarchar](50) NULL,
        [Start_latitude] [decimal](18, 6) NULL,
        [Start_longitude] [decimal](18, 6) NULL,
        [Mid_latitude] [decimal](18, 6) NULL,
        [Mid_longitude] [decimal](18, 6) NULL,
        [End_latitude] [decimal](18, 6) NULL,
        [End_longitude] [decimal](18, 6) NULL,
        [Distance] [nvarchar](50) NULL,
        [Start_echo] [decimal](18, 6) NULL,
        [Mid_echo] [decimal](18, 6) NULL,
        [End_echo] [decimal](18, 6) NULL,
        [SAMPLER] [nvarchar](50) NOT NULL,
        [Mesh_micro_m] [nvarchar](50) NULL,
        [Volume_dec_cubic_m] [decimal](18, 6) NULL,
        [Duration] [time](0) NULL,
        [Start_depth] [decimal](18, 6) NULL,
        [Mid_depth] [decimal](18, 6) NULL,
        [End_depth] [decimal](18, 6) NULL,
        [Flow_meter_type] [nvarchar](50) NULL,
```

```
	[Start_count] [nvarchar](50) NULL,
	[End_count] [nvarchar](50) NULL,
	[REVS] [nvarchar](50) NULL,
	[Valid_Y_N] [nvarchar](50) NULL,
	[CRUISE_CODE] [nvarchar](50) NOT NULL
) ON [PRIMARY]
GO

---Insert data from STAGE_RECORDS table to RECORDS table
INSERT INTO [dbo].[RECORDS] SELECT * FROM [dbo].[STAGE_RECORDS];

---Count rows in STAGE_RECORDS table
SELECT COUNT(*) FROM [dbo].[RECORDS];
```

Creation of CRUISES table

```
---Create new table CRUISES using task import flat file (Figure 34)
```

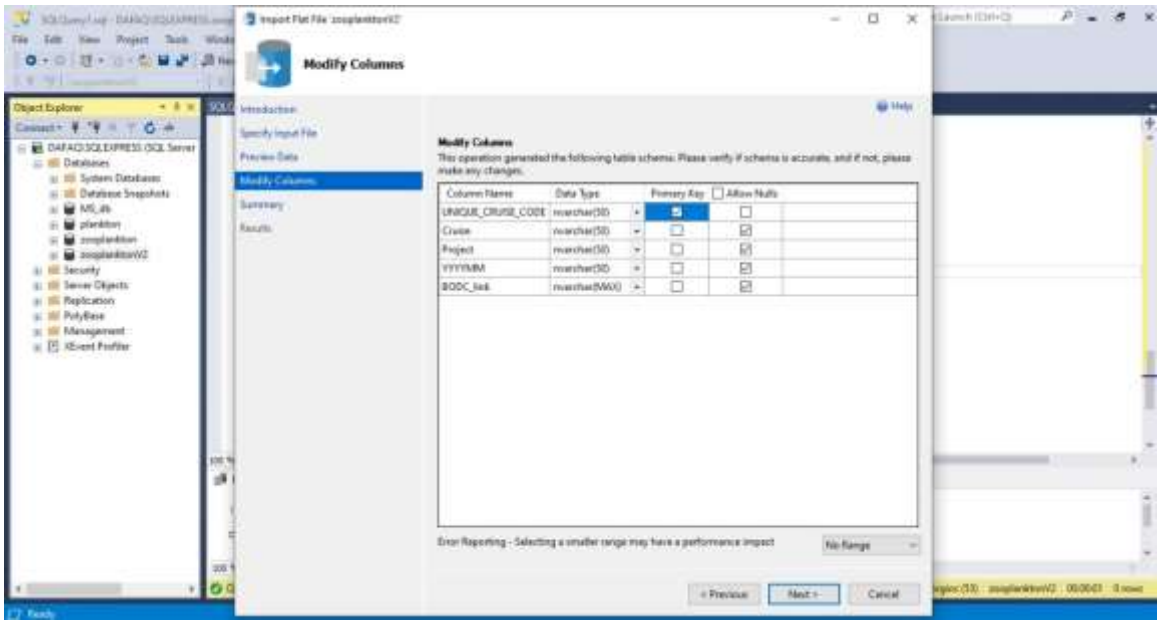


Figure 34 Cruises table

STAGE_SAMPLES & bulk insert

```
---Create STAGE_SAMPLES table
```

```
CREATE TABLE [dbo].[STAGE_SAMPLES](
		[HAUL_NO] [int] NOT NULL,
		[CLASS] [nvarchar](100) NULL,
		[TYPE] [nvarchar](50) NULL,
		[VALUE] [decimal](18, 6) NULL,
		[SAMPLER] [nvarchar](50) NOT NULL,
		[CRUISE_CODE] [nvarchar](50) NOT NULL,
		[FK_REC_ID] [int] NULL
		) ON [PRIMARY]
GO


---Bulk insertion for STAGE_SAMPLES table
BULK INSERT [dbo].[STAGE_SAMPLES]
FROM
'C:\Users\georgios\Documents\placem\RESULTS_CSV_COMBINED\work_files\allSAMPLE_22_07_OFFICIAL.csv'
WITH (FIRSTROW = 2,
   FIELDTERMINATOR = ',',
		ROWTERMINATOR='\n' ,
		 MAXERRORS=4000000,
		 BATCHSIZE=40000);
```

Foreign key relationship creation between CRUISES and RECORDS tables

```
---after populating CRUISES and RECORDS table create FK between them
---FK relationship between RECORDS and CRUISES tables
ALTER TABLE [dbo].[RECORDS]  WITH CHECK ADD  CONSTRAINT [FK_RECORDS_CRUISES] FOREIGN
KEY([CRUISE_CODE])
REFERENCES [dbo].[CRUISES] ([UNIQUE_CRUISE_CODE])
GO


ALTER TABLE [dbo].[RECORDS] CHECK CONSTRAINT [FK_RECORDS_CRUISES]
GO
```

```
---Problem arises/ we check problem
```

```
select Distinct rec.[CRUISE_CODE] from [dbo].[RECORDS] as rec
WHERE rec.[CRUISE_CODE] NOT IN
(SELECT ct.[UNIQUE_CRUISE_CODE] from [dbo].[CRUISES] as ct);
SELECT DISTINCT [CRUISE_CODE] from [dbo].[RECORDS] where [CRUISE_CODE] like 'LIN%';
```

```
---Make appropriate changes in RECORDS table, then run FK creation command again
UPDATE
   [dbo].[RECORDS]
SET
   [CRUISE_CODE] = REPLACE([CRUISE_CODE], 'LINNHE9', 'LINNHE91')
WHERE
    [CRUISE_CODE] LIKE  'LINNHE9';

---rerun FK creation between RECORDS nad CRYUISES table
ALTER TABLE [dbo].[RECORDS]  WITH CHECK ADD  CONSTRAINT [FK_RECORDS_CRUISES] FOREIGN KEY([CRUISE_CODE])
REFERENCES [dbo].[CRUISES] ([UNIQUE_CRUISE_CODE])
GO

ALTER TABLE [dbo].[RECORDS] CHECK CONSTRAINT [FK_RECORDS_CRUISES]
GO
```

Fixes in STAGE_SAMPLES table, creation of FK, creation of SAMPLES table & insert of data

```
---Now check STAGE_SAMPLES FOR LINNHE91
SELECT DISTINCT [CRUISE_CODE] from [dbo].[STAGE_SAMPLES] where [CRUISE_CODE] like 'LIN%';
---Make appropriate change
UPDATE
   [dbo].[STAGE_SAMPLES]
SET
   [CRUISE_CODE] = REPLACE([CRUISE_CODE], 'LINHE91', 'LINNHE91')
WHERE
    [CRUISE_CODE]  LIKE 'LINHE91';
```

```
---Create FK in STAGE_SAMPLES table based on constraints
UPDATE [dbo].[STAGE_SAMPLES]
SET  [dbo].[STAGE_SAMPLES].[FK_REC_ID] = sr.[RECORDS_ID]
FROM [dbo].[RECORDS] AS sr, [dbo].[STAGE_SAMPLES] AS da
WHERE sr.HAUL_NO = da.HAUL_NO
AND sr.CRUISE_CODE = da.CRUISE_CODE
AND       sr.[SAMPLER] = da.SAMPLER;

---Create SAMPLES Table
CREATE TABLE [dbo].[SAMPLES](
            [SAMPLE_ID] [int] IDENTITY(1,1) NOT NULL PRIMARY KEY,
            [HAUL_NO] [int] NOT NULL,
            [CLASS] [nvarchar](100) NOT NULL,
            [TYPE] [nvarchar](50) NULL,
            [VALUE] [decimal](18, 6) NULL,
            [SAMPLER] [nvarchar](50) NOT NULL,
            [CRUISE_CODE] [nvarchar](50) NULL,
            [FK_REC_ID] [int] NULL
            ) ON [PRIMARY]
GO

---Insert data from STAGE_SAMPLES table to SAMPLES table
INSERT INTO [dbo].[SAMPLES] SELECT * FROM [dbo].[STAGE_SAMPLES];
```

Creation of Species_info table

```
---Import Species_info table with flat file import wizard (Figure 35)
```

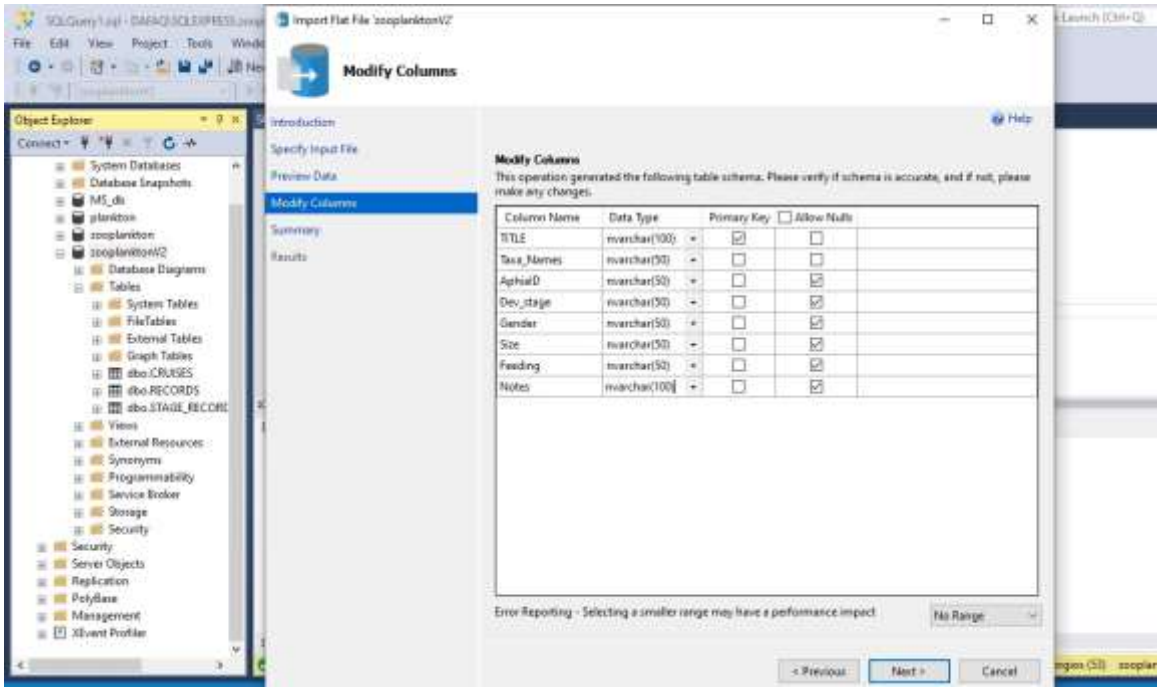


Figure 35 Species_info table

Changes on tables before Foreign Key relationships creation

```
---Perform left rimming on class from Samples and TITLE&Taxa_Names from Species_info table

UPDATE [dbo].[SAMPLES] SET CLASS = LTRIM(CLASS);

UPDATE [dbo].[Species_info] SET [Taxa_Names] = LTRIM([Taxa_Names]);

UPDATE [dbo].[Species_info] SET [TITLE] = LTRIM([TITLE]);


---Check not matching elements between tables for problem in FK creation

SELECT Distinct [CLASS] AS dCLASS FROM [dbo].[SAMPLES]

WHERE [CLASS] NOT IN

(SELECT [TITLE] from [dbo].[Species_info]) ORDER BY dCLASS;


SELECT [TITLE] FROM [dbo].[Species_info];
```

```
---Check for wrongs

select distinct CLASS from [dbo].[SAMPLES] WHERE [CLASS] LIKE '%All%';
```

Last updates on tables

Now that not matching elements have been found with the script above some changes need to be done like the one following. The “gooseberry” was identified from the initial excel file due to its name uniqueness and it was intentionally left in a wrong format to test the script above and create a script that would easily implement name changes, like the one below. Such script is important, because names of species may change occasionally due to name updates scientifically decided or due to mistakes that where not noticed in the initial files entered in the database.

```
---Update/replace gooseberry name

UPDATE

   [dbo].[Species_info]

SET

   [TITLE] = REPLACE([TITLE], '"gooseberry"""', 'gooseberry')

WHERE

    [TITLE] = '"gooseberry"""';


SELECT [TITLE] FROM [dbo].[Species_info];


---Check not matching elements between tables for problem in FK creation

select Distinct sa.[CRUISE_CODE] from [dbo].[SAMPLES] as sa

WHERE sa.[CRUISE_CODE] NOT IN

(SELECT ct.[UNIQUE_CRUISE_CODE] from [dbo].[CRUISES] as ct);


---Check and make other changes in SAMPLES table

SELECT DISTINCT [TYPE] AS dTYPE FROM [dbo].[SAMPLES] WHERE [TYPE] LIKE 'CO%';


UPDATE

   [dbo].[SAMPLES]

SET

   [TYPE] = REPLACE([TYPE], 'COUNTS', 'COUNT')

WHERE

    [TYPE] = 'COUNTS';


SELECT DISTINCT [TYPE] AS dTYPE FROM [dbo].[SAMPLES] WHERE [TYPE] LIKE 'CO%';
```

Create remaining Foreign Key Relationships

```
---Create FK relationship between Records and Samples table

ALTER TABLE [dbo].[SAMPLES] WITH CHECK ADD CONSTRAINT [FK_SAMPLES_RECORDS] FOREIGN KEY([FK_REC_ID])

REFERENCES [dbo].[RECORDS] ([RECORDS_ID])

GO

ALTER TABLE [dbo].[SAMPLES] CHECK CONSTRAINT [FK_SAMPLES_RECORDS]

GO


---FK relationship between SAMPLES and CRUISES tables

ALTER TABLE [dbo].[SAMPLES] WITH CHECK ADD CONSTRAINT [FK_SAMPLES_CRUISES] FOREIGN KEY([CRUISE_CODE])

REFERENCES [dbo].[CRUISES] ([UNIQUE_CRUISE_CODE])

GO

ALTER TABLE [dbo].[SAMPLES] CHECK CONSTRAINT [FK_SAMPLES_CRUISES]

GO


---Add FOREIGN KEY relationship with NO CHECK between SAMPLES and Species_info table

ALTER TABLE [dbo].[SAMPLES]

WITH NOCHECK ADD CONSTRAINT [FK_SAMPLES_SpInfo]

        FOREIGN KEY (CLASS) REFERENCES [dbo].[Species_info]([TITLE])

GO
```

Merge and Update on future data

As mentioned before new data may come into play at a later point in time and the client should be able to update the working RECORDS and SAMPLES tables. So once data has been uploaded in the staging tables using the bulk insertion mechanism already described, the following query would make sure that all new data is uploaded in the RECORDS table

```
---how to update existing and insert new

MERGE INTO

 [dbo].[RECORDS] as rec

USING

 [dbo].[STAGE_RECORDS] as sr
```

```
ON
 rec.[HAUL_NO] =sr.[HAUL_NO]
        and rec.[SAMPLER] =sr.[SAMPLER] and rec.[CRUISE_CODE] =sr.[CRUISE_CODE]
WHEN MATCHED THEN
 UPDATE SET
  rec.[DATE] =sr.[DATE] , rec.[Timestamp] =sr.[Timestamp] , rec.[HAUL_NO] =sr.[HAUL_NO] ,
        rec.[Station_name] =sr.[Station_name] ,
        rec.[Start_latitude] =sr.[Start_latitude] , rec.[Start_longitude] =sr.[Start_longitude] ,
        rec.[Mid_latitude] =sr.[Mid_latitude] ,
        rec.[Mid_longitude] =sr.[Mid_longitude] , rec.[End_latitude] =sr.[End_latitude] ,
        rec.[End_longitude] =sr.[End_longitude] ,
        rec.[Distance] =sr.[Distance] , rec.[Start_echo] =sr.[Start_echo] , rec.[Mid_echo] =sr.[Mid_echo] ,
        rec.[End_echo] =sr.[End_echo] ,
        rec.[SAMPLER] =sr.[SAMPLER] , rec.[Mesh_micro_m] =sr.[Mesh_micro_m] ,
        rec.[Volume_dec_cubic_m] =sr.[Volume_dec_cubic_m] ,
        rec.[Duration] =sr.[Duration] , rec.[Start_depth] =sr.[Start_depth] , rec.[Mid_depth]
=sr.[Mid_depth] , rec.[End_depth] =sr.[End_depth] ,
        rec.[Flow_meter_type] =sr.[Flow_meter_type] , rec.[Start_count] =sr.[Start_count] ,
        rec.[End_count] =sr.[End_count] ,
        rec.[REVS] =sr.[REVS] , rec.[Valid_Y_N] =sr.[Valid_Y_N] , rec.[CRUISE_CODE] =sr.[CRUISE_CODE]
WHEN NOT MATCHED THEN
 INSERT
  ([DATE] , [Timestamp] , [HAUL_NO] , [Station_name] , [Start_latitude] , [Start_longitude] ,
        [Mid_latitude] ,
        [Mid_longitude] , [End_latitude] , [End_longitude] , [Distance] , [Start_echo] , [Mid_echo] ,
        [End_echo] ,
        [SAMPLER] , [Mesh_micro_m] , [Volume_dec_cubic_m] , [Duration] , [Start_depth]
, [Mid_depth] , [End_depth] ,
        [Flow_meter_type] , [Start_count] , [End_count] , [REVS] , [Valid_Y_N] ,
        [CRUISE_CODE]
)
 VALUES
  (sr.[DATE] ,sr.[Timestamp] ,sr.[HAUL_NO] ,sr.[Station_name] , sr.[Start_latitude] , sr.[Start_longitude] ,
sr.[Mid_latitude] , sr.[Mid_longitude] ,
sr.[End_latitude] , sr.[End_longitude] , sr.[Distance] , sr.[Start_echo] , sr.[Mid_echo] , sr.[End_echo] ,
sr.[SAMPLER] , sr.[Mesh_micro_m] ,
sr.[Volume_dec_cubic_m] , sr.[Duration] , sr.[Start_depth] , sr.[Mid_depth] , sr.[End_depth] ,
sr.[Flow_meter_type] , sr.[Start_count] , sr.[End_count] ,
```

```
sr.[REVS] , sr.[Valid_Y_N] , sr.[CRUISE_CODE]);
```

and the following query would make sure that all new data is uploaded in the SAMPLES table

```
MERGE INTO
 [dbo].[SAMPLES] AS sa
USING
 [dbo].[STAGE_SAMPLES] AS ss
ON
 sa.[HAUL_NO] =ss.[HAUL_NO]
            and sa.[SAMPLER] =ss.[SAMPLER] and sa.[CRUISE_CODE] =ss.[CRUISE_CODE]
WHEN MATCHED THEN
 UPDATE SET
  sa.[HAUL_NO] =ss.[HAUL_NO] ,   sa.[CLASS] =ss.[CLASS] , sa.[TYPE] =ss.[TYPE] ,          sa.[VALUE]
=ss.[VALUE] ,
            sa.[SAMPLER] =ss.[SAMPLER] ,       sa.[CRUISE_CODE] =ss.[CRUISE_CODE] ,       sa.[FK_REC_ID]
=ss.[FK_REC_ID]

WHEN NOT MATCHED THEN
 INSERT
  ([HAUL_NO] ,   [CLASS], [TYPE] ,  [VALUE],[SAMPLER],         [CRUISE_CODE],   [FK_REC_ID]
)
 VALUES
  (ss.[HAUL_NO] ,ss.[CLASS] ,ss.[TYPE] ,ss.[VALUE] , ss.[SAMPLER] , ss.[CRUISE_CODE] , ss.[FK_REC_ID]);
```

Appendix-Alternative Database Implementation

The process in the SSMS and code used to create it is very similar to the originally created database with the exceptions documented here with blue:

STAGE_RECORDS, bulk insert & RECORDS table

```
/*final zooplankton_db_alternative creation*/

CREATE TABLE [dbo].[STAGE_RECORDS](

            [DATE] [date] NOT NULL,

            [Timestamp] [time](0) NOT NULL,

            [HAUL_NO] [int] NOT NULL,

            [Station_name] [nvarchar](50) NULL,

            [Start_latitude] [decimal](18, 6) NULL,

            [Start_longitude] [decimal](18, 6) NULL,

            [Mid_latitude] [decimal](18, 6) NULL,
```

```
	[Mid_longitude] [decimal](18, 6) NULL,
	[End_latitude] [decimal](18, 6) NULL,
	[End_longitude] [decimal](18, 6) NULL,
	[Distance] [nvarchar](50) NULL,
	[Start_echo] [decimal](18, 6) NULL,
	[Mid_echo] [decimal](18, 6) NULL,
	[End_echo] [decimal](18, 6) NULL,
	[SAMPLER] [nvarchar](50) NOT NULL,
	[Mesh_micro_m] [nvarchar](50) NULL,
	[Volume_dec_cubic_m] [decimal](18, 6) NULL,
	[Duration] [time](0) NULL,
	[Start_depth] [decimal](18, 6) NULL,
	[Mid_depth] [decimal](18, 6) NULL,
	[End_depth] [decimal](18, 6) NULL,
	[Flow_meter_type] [nvarchar](50) NULL,
	[Start_count] [nvarchar](50) NULL,
	[End_count] [nvarchar](50) NULL,
	[REVS] [nvarchar](50) NULL,
	[Valid_Y_N] [nvarchar](50) NULL,
	[CRUISE_CODE] [nvarchar](50) NOT NULL
) ON [PRIMARY]
GO


---Bulk insertion for STAGE_RECORDS table
BULK INSERT [dbo].[STAGE_RECORDS]
FROM
'C:\Users\georgios\Documents\placem\RESULTS_CSV_COMBINED\work_files\allrec_COMBfile_THE_OFFICIAL
_WORKING.csv'
WITH (FIRSTROW = 2,
   FIELDTERMINATOR = ',',
	ROWTERMINATOR='\n' ,
	 MAXERRORS=4000000,
	 BATCHSIZE=40000);
```

```
---COUNT ROWS IN STAGE_RECORDS TABLE

SELECT COUNT(*) FROM [dbo].[STAGE_RECORDS];


---Check for duplicates in STAGE_RECORDS table and then delete them

WITH CTE ([HAUL_NO],

                    [SAMPLER],

                    [CRUISE_CODE],

                    duplicatecount)

AS (SELECT [HAUL_NO],

                      [SAMPLER],

                      [CRUISE_CODE],

      ROW_NUMBER() OVER(PARTITION BY [HAUL_NO],
                                                  [SAMPLER],
                                  [CRUISE_CODE]

      ORDER BY [HAUL_NO]) AS DuplicateCount

  FROM [dbo].[STAGE_RECORDS])

SELECT * FROM CTE WHERE DuplicateCount <> 1; /* Some duplicated have been inserted on purpose for this query to show it is working*/

---DELETE FROM CTE WHERE DuplicateCount <> 1;

---(746 rows affected)


---COUNT ROWS IN STAGE_RECORDS TABLE TO CHECK THE CHANGE

SELECT COUNT(*) FROM [dbo].[STAGE_RECORDS];


---Create RECORDS table

CREATE TABLE [dbo].[RECORDS](

          [RECORDS_ID] [int] IDENTITY(1,1) NOT NULL PRIMARY KEY,

          [DATE] [date] NOT NULL,

          [Timestamp] [time](0) NOT NULL,

          [HAUL_NO] [int] NOT NULL,

          [Station_name] [nvarchar](50) NULL,

          [Start_latitude] [decimal](18, 6) NULL,

          [Start_longitude] [decimal](18, 6) NULL,

          [Mid_latitude] [decimal](18, 6) NULL,
```

```
	[Mid_longitude] [decimal](18, 6) NULL,
	[End_latitude] [decimal](18, 6) NULL,
	[End_longitude] [decimal](18, 6) NULL,
	[Distance] [nvarchar](50) NULL,
	[Start_echo] [decimal](18, 6) NULL,
	[Mid_echo] [decimal](18, 6) NULL,
	[End_echo] [decimal](18, 6) NULL,
	[SAMPLER] [nvarchar](50) NOT NULL,
	[Mesh_micro_m] [nvarchar](50) NULL,
	[Volume_dec_cubic_m] [decimal](18, 6) NULL,
	[Duration] [time](0) NULL,
	[Start_depth] [decimal](18, 6) NULL,
	[Mid_depth] [decimal](18, 6) NULL,
	[End_depth] [decimal](18, 6) NULL,
	[Flow_meter_type] [nvarchar](50) NULL,
	[Start_count] [nvarchar](50) NULL,
	[End_count] [nvarchar](50) NULL,
	[REVS] [nvarchar](50) NULL,
	[Valid_Y_N] [nvarchar](50) NULL,
	[CRUISE_CODE] [nvarchar](50) NOT NULL
) ON [PRIMARY]
GO

---Insert data from STAGE_RECORDS table to RECORDS table
INSERT INTO [dbo].[RECORDS] SELECT * FROM [dbo].[STAGE_RECORDS];

---COUNT ROWS IN STAGE_RECORDS TABLE
SELECT COUNT(*) FROM [dbo].[RECORDS];
```

STAGE_SAMPLES & bulk insert

```
---Create STAGE_SAMPLES table
CREATE TABLE [dbo].[STAGE_SAMPLES](
	[PK_staging_samples] [int] IDENTITY(1,1) NOT NULL PRIMARY KEY,
```

```
	[HAUL_NO] [int] NOT NULL,
	[CLASS] [nvarchar](100) NULL,
	[TYPE] [nvarchar](50) NULL,
	[VALUE] [decimal](18, 6) NULL,
	[SAMPLER] [nvarchar](50) NOT NULL,
	[CRUISE_CODE] [nvarchar](50) NOT NULL,
	[FK_REC_ID] [int] NULL
	) ON [PRIMARY]
GO
---Bulk insertion for STAGE_SAMPLES table
BULK INSERT [dbo].[STAGE_SAMPLES]
FROM
'C:\Users\georgios\Documents\placem\RESULTS_CSV_COMBINED\work_files\for_alternativedb\newStaginSamplesFile.csv'
WITH (FIRSTROW = 2,
  FIELDTERMINATOR = ',',
	ROWTERMINATOR='\n' ,
	 MAXERRORS=4000000,
	 BATCHSIZE=40000);
```

Creation of CRUISES table and FK relationship with RECORDS table

```
---Create new table CRUISES using task import flat file (Figure 34)


---after populating CRUISES and RECORDS table create FK between them
---FK relationship between RECORDS and CRUISES tables
ALTER TABLE [dbo].[RECORDS] WITH CHECK ADD CONSTRAINT [FK_RECORDS_CRUISES] FOREIGN KEY([CRUISE_CODE])
REFERENCES [dbo].[CRUISES] ([UNIQUE_CRUISE_CODE])
GO


ALTER TABLE [dbo].[RECORDS] CHECK CONSTRAINT [FK_RECORDS_CRUISES]
GO
```

```
---Problem arises/ we check problem

SELECT DISTINCT rec.[CRUISE_CODE] from [dbo].[RECORDS] as rec

WHERE rec.[CRUISE_CODE] NOT IN

(SELECT ct.[UNIQUE_CRUISE_CODE] FROM [dbo].[CRUISES] AS ct);

SELECT DISTINCT [CRUISE_CODE] FROM [dbo].[RECORDS] WHERE [CRUISE_CODE] like 'LIN%';


---Make appropriate changes in RECORDS table, then run FK creation command again

UPDATE

   [dbo].[RECORDS]

SET

   [CRUISE_CODE] = REPLACE([CRUISE_CODE], 'LINNHE9', 'LINNHE91')

WHERE

    [CRUISE_CODE] LIKE  'LINNHE9';


---rerun FK creation between RECORDS and CRYUISES table

ALTER TABLE [dbo].[RECORDS]  WITH CHECK ADD  CONSTRAINT [FK_RECORDS_CRUISES] FOREIGN KEY([CRUISE_CODE])

REFERENCES [dbo].[CRUISES] ([UNIQUE_CRUISE_CODE])

GO


ALTER TABLE [dbo].[RECORDS] CHECK CONSTRAINT [FK_RECORDS_CRUISES]

GO
```

Fixes in STAGE_SAMPLES table, creation of FK, creation of SAMPLES table & insert of data

```
---Now check STAGE_SAMPLES FOR LINNHE91

SELECT DISTINCT [CRUISE_CODE] from [dbo].[STAGE_SAMPLES] where [CRUISE_CODE] like 'LIN%';


---Make appropriate change

UPDATE

   [dbo].[STAGE_SAMPLES]

SET

   [CRUISE_CODE] = REPLACE([CRUISE_CODE], 'LINHE91', 'LINNHE91')
```

```
WHERE
    [CRUISE_CODE]  LIKE 'LINHE91';


---populate fk_rec column based on RECORDS tables constraints
UPDATE [dbo].[STAGE_SAMPLES]
SET  [dbo].[STAGE_SAMPLES].[FK_REC_ID] = sr.[RECORDS_ID]
FROM [dbo].[RECORDS] AS sr, [dbo].[STAGE_SAMPLES] AS da
WHERE sr.HAUL_NO = da.HAUL_NO
AND sr.CRUISE_CODE = da.CRUISE_CODE
AND       sr.[SAMPLER] = da.SAMPLER;


---Create SAMPLES Table
CREATE TABLE [dbo].[SAMPLES](
            [SAMPLE_ID] [int] NOT NULL PRIMARY KEY,
            [CLASS] [nvarchar](100) NULL,
            [TYPE] [nvarchar](50) NULL,
            [VALUE] [decimal](18, 6) NULL,
            [Taxa_Names] [nvarchar](50) NULL,
            [AphiaID] [nvarchar](50) NULL,
            [Dev_stage] [nvarchar](50) NULL,
            [Gender] [nvarchar](50) NULL,
            [Size] [nvarchar](50) NULL,
            [Feeding] [nvarchar](50) NULL,
            [Notes] [nvarchar](100) NULL,
            [FK_REC_ID] [int] NULL
            ) ON [PRIMARY]
GO


---Insert data from STAGE_SAMPLES table SAMPLES table
INSERT INTO [dbo].[SAMPLES]([SAMPLE_ID], [CLASS], [TYPE], [VALUE], [FK_REC_ID])
SELECT [PK_staging_samples], [CLASS], [TYPE], [VALUE], [FK_REC_ID]
FROM [dbo].[STAGE_SAMPLES];
```

```
---Make approprite changes to SAMPLES table
UPDATE [dbo].[SAMPLES] SET CLASS = LTRIM(CLASS);


UPDATE
   [dbo].[SAMPLES]
SET
   [TYPE] = REPLACE([TYPE], 'COUNTS', 'COUNT')
WHERE
    [TYPE] = 'COUNTS';
```

Creation of Species_info table and make fixes in it

```
---Import species_info table with flat file import wizard (Figure 35)


---Perform left rimming on class from Samples and TITLE&Taxa_Names from Species_info table
UPDATE [dbo].[Species_info] SET TITLE = LTRIM(TITLE);


---Update/replace gooseberry name in Species_info table
UPDATE
   [dbo].[Species_info]
SET
   [TITLE] = REPLACE([TITLE], '"gooseberry"""', 'gooseberry')
WHERE
    [TITLE] = '"gooseberry"""';


SELECT [TITLE] FROM [dbo].[Species_info];
```

Insert data from Species_info table to SAMPLES table & create FK relationship with RECORDS table

```
---UPDATE SAMPLES table from Species_info table
UPDATE [dbo].[SAMPLES]
  SET [SAMPLES].[Taxa_Names] = sp.[Taxa_Names],
  [SAMPLES].[AphiaID] = sp.[AphiaID],
  [SAMPLES].[Dev_stage] = sp.[Dev_stage],
  [SAMPLES].[Gender] = sp.[Gender],
```

```
  [SAMPLES].[Size] = sp.[Size],
  [SAMPLES].[Feeding] = sp.[Feeding],
  [SAMPLES].[Notes] = sp.[Notes]
  FROM [dbo].[SAMPLES] AS sa  INNER JOIN  [dbo].[Species_info] as sp ON sa.[CLASS] = sp.[TITLE]


---Create FK relationship between Records and Samples table
ALTER TABLE [dbo].[SAMPLES]  WITH CHECK ADD  CONSTRAINT [FK_SAMPLES_RECORDS] FOREIGN KEY([FK_REC_ID])
REFERENCES [dbo].[RECORDS] ([RECORDS_ID])
GO
```

Merge and Update on future data

```
---UPDATE SAMPLES TABLE
---STEP1
MERGE INTO
 [dbo].[SAMPLES] AS sa
USING
 [dbo].[STAGE_SAMPLES] AS ss
ON
 sa.[SAMPLE_ID] = ss.[PK_staging_samples]
WHEN MATCHED THEN
 UPDATE SET
  sa.[HAUL_NO] =ss.[HAUL_NO] ,  sa.[CLASS] =ss.[CLASS] , sa.[TYPE] =ss.[TYPE] , sa.[VALUE] =ss.[VALUE] ,
          sa.[FK_REC_ID] =ss.[FK_REC_ID]


WHEN NOT MATCHED THEN
 INSERT
  ([SAMPLE_ID], [CLASS], [TYPE] ,  [VALUE], [FK_REC_ID]
)
 VALUES
  (ss.[PK_staging_samples] ,ss.[CLASS] ,ss.[TYPE] ,ss.[VALUE] , , ss.[FK_REC_ID]);


---STEP 2
MERGE INTO
```

```
 [dbo].[SAMPLES] AS sa
USING
 [dbo].[Species_info] AS sp
ON
 sa.[CLASS] = sp.[TITLE]
WHEN MATCHED THEN
 UPDATE SET
  sa.[Taxa_Names] =sp.[Taxa_Names] ,  sa.[AphiaID] =sp.[AphiaID] , sa.[Dev_stage] =sp.[Dev_stage] ,
      sa.[Gender] =sp.[Gender],
      sa.[Size] =sp.[Size] , sa.[Feeding] =sp.[Feeding], sa.[Notes] =sp.[Notes]

WHEN NOT MATCHED THEN
 INSERT
  ([Taxa_Names],[AphiaID],  [Dev_stage], [Gender], [Size], [Feeding], [Notes]
)
 VALUES
  (sp.[Taxa_Names],sp.[AphiaID], sp.[Dev_stage], sp.[Gender], sp.[Size], sp.[Feeding], sp.[Notes]);
```

Appendix-Data Mining in R

SCENARIO_1

First the function that will handle the scripting that follows needs to be created.

```
# function to run scripts

g <- function(sql){

 sqlQuery(dbhandle,sql)

}
```

The researcher first needs to verify existing records that are available in both tables according to cruise codes Figure 36

```
### SCENARIO_1 ###

## SAMPLE TABLE ON GENERIC SEARCH BASED ON CRUISE CODE ##

sql <- 'SELECT DISTINCT rec.CRUISE_CODE AS records_Cruise, sa.CRUISE_CODE as samples_Cruise FROM [dbo].[RECORDS] AS rec

FULL JOIN [dbo].[SAMPLES] as sa ON sa.FK_REC_ID = rec.RECORDS_ID;'

df <- g(sql)

df
```

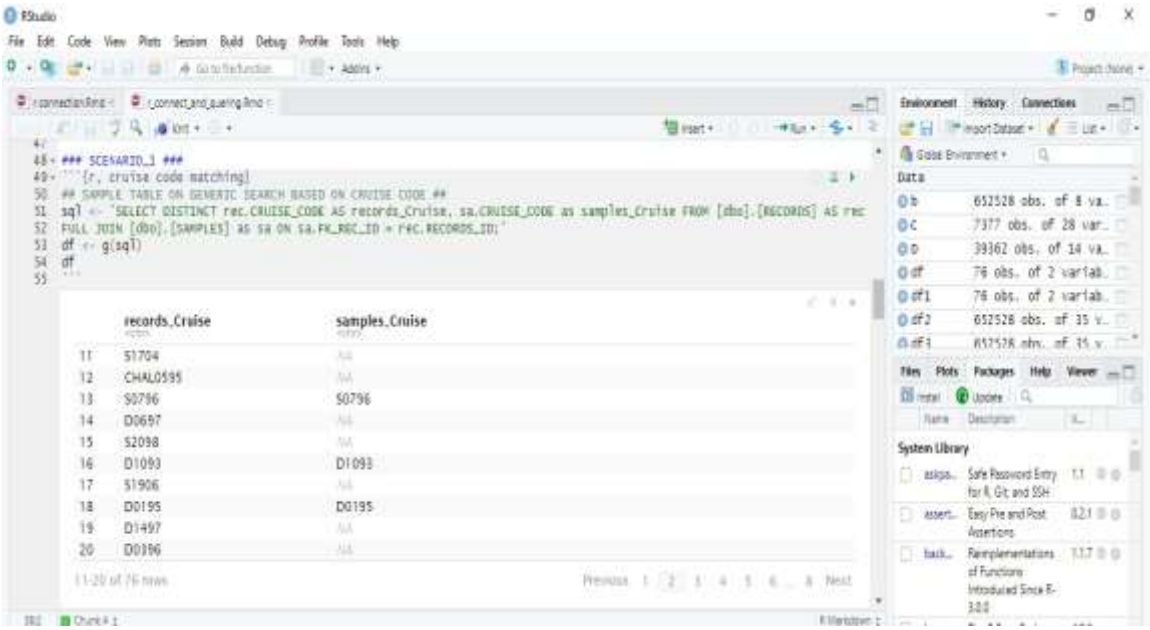


Figure 36 Available cruiseson both tables

The researcher is then asked to provide in the dialogue box the cruise code that they are interested in. Here 'S1704'. Figure 37

```
# search on cruise code
```

```
cruise <- readline(prompt = "ENTER CRUISE CODE IN ' ': ")

message1 <- paste("SELECT * FROM [dbo].[SAMPLES] WHERE CRUISE_CODE = ", cruise, " ;")

b <- g(message1)

b
```

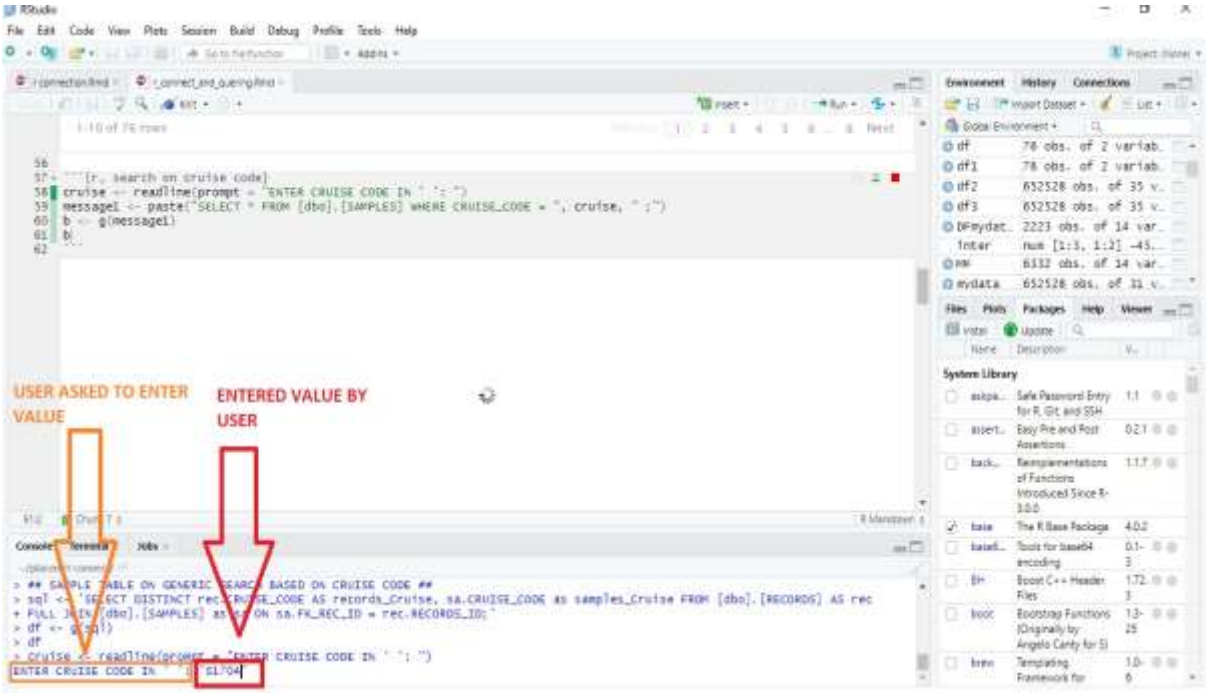
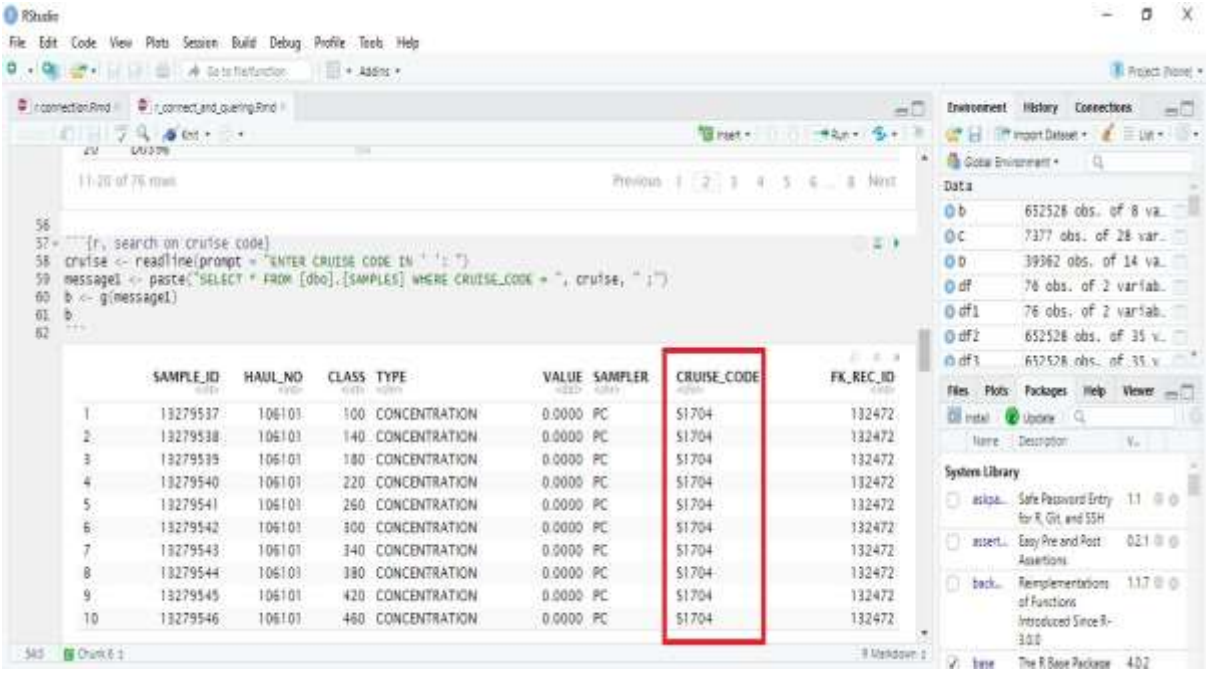


Figure 37 User asked to provide cruise code



	SAMPLE_ID	HAUL_NO	CLASS	TYPE	VALUE	SAMPLER	CRUISE_CODE	FK_REC_ID
1	13279537	106101	100	CONCENTRATION	0.0000	PC	S1704	132472
2	13279538	106101	140	CONCENTRATION	0.0000	PC	S1704	132472
3	13279539	106101	180	CONCENTRATION	0.0000	PC	S1704	132472
4	13279540	106101	220	CONCENTRATION	0.0000	PC	S1704	132472
5	13279541	106101	260	CONCENTRATION	0.0000	PC	S1704	132472
6	13279542	106101	300	CONCENTRATION	0.0000	PC	S1704	132472
7	13279543	106101	340	CONCENTRATION	0.0000	PC	S1704	132472
8	13279544	106101	380	CONCENTRATION	0.0000	PC	S1704	132472
9	13279545	106101	420	CONCENTRATION	0.0000	PC	S1704	132472
10	13279546	106101	460	CONCENTRATION	0.0000	PC	S1704	132472

Figure 38 Retrieved data of interest

Researcher sees something interesting on retrieves data (Figure 38) decides to work on it. This implies combination with logging records. This translates into joining RECORDS, SAMPLES and Species_info tables. (Figure 39) The cruise code previously entered is automatically imported here.

```
# combine tables of interest

message2 <- paste("SELECT rec.*, sa.[CLASS], sa.[TYPE], sa.[VALUE], sp.[Taxa_Names], sp.[AphiaID], ct.[Project], ct.[BODC_link]

FROM [dbo].[CRUISES] AS ct JOIN [dbo].[RECORDS] AS rec ON ct.[UNIQUE_CRUISE_CODE] = rec.[CRUISE_CODE]

JOIN [dbo].[SAMPLES] AS sa ON rec.[RECORDS_ID] = sa.[FK_REC_ID]

FULL JOIN [dbo].[Species_info] AS sp ON sp.[TITLE] = sa.[CLASS] WHERE rec.CRUISE_CODE = ", cruise, ";")

df2 <- g(message2)

df2
```

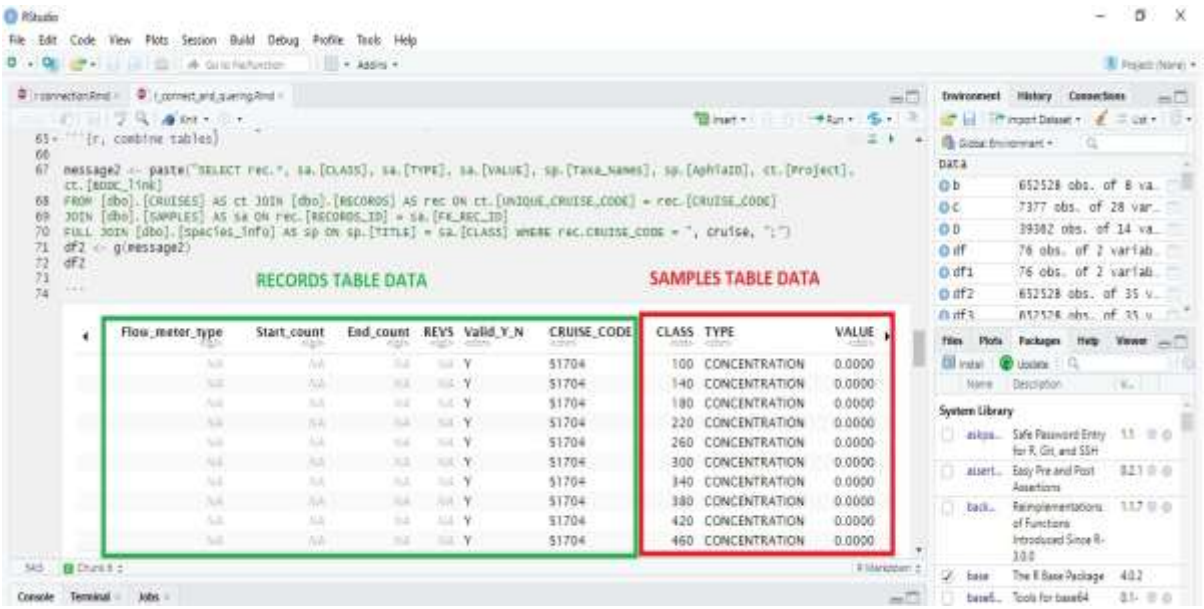


Figure 39 Combined tables

The researcher decides that adequate information exists for further research. The retrieved information above is kept intact and copied as a data frame in R for further data mining. At this point connection with the database is not used which will speed up the search and data mining process. In the code box below excluded columns are for the researcher to be chosen as well as the filtering out of 0s.(Figure 40)

```
# the retrieved data is now and R data frame

df3 <- data.frame(df2)

# exclude columns of no use

mydata = select(df3, -c(RECORDS_ID, Station_name, Taxa_Names, AphiaID))

mydata

# filter out 0s

mydata2 <- filter(mydata, VALUE != 0.0)
```

```
mydata2
```

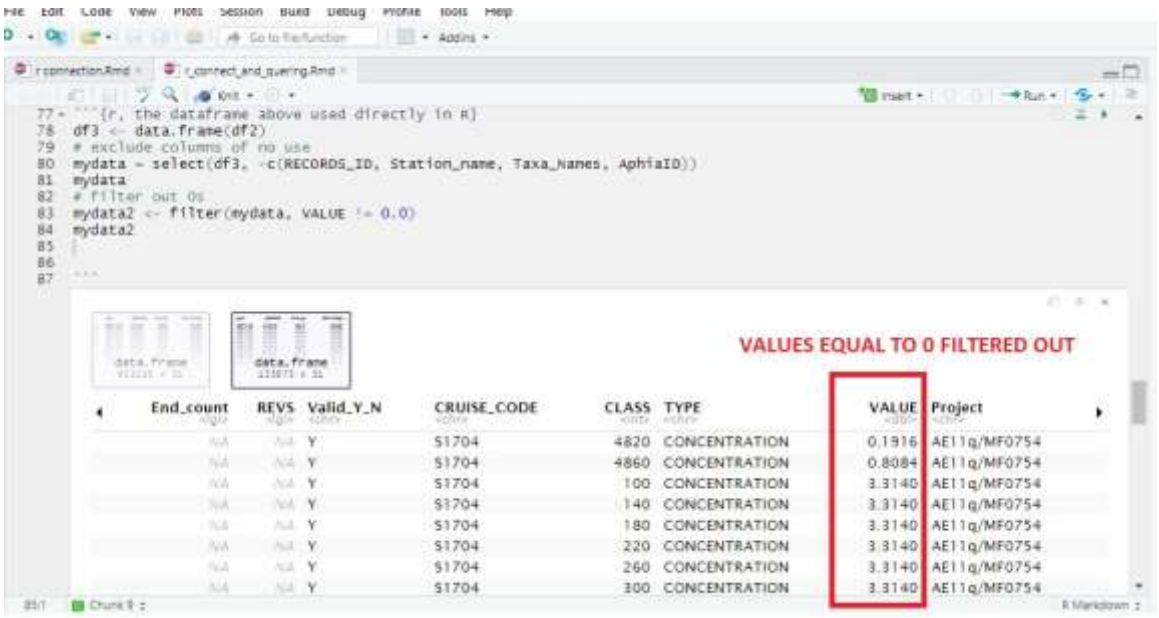


Figure 40 Filtered out data

The researcher needs to narrow down results according to some perimeter. As mentioned by Marine Scotland the echo attribute is of high importance. The script below is run and the researcher is asked to provide numbers for the echo attributes (Figure 41). As can be seen the operators “>” and “<” are fixed and this is correct given the nature of the data stored in them. The result comes up as a data frame (Figure 42)

```
# work on columns based on interest

st_echo <- readline(prompt = "Enter start_echo quantity: ")

en_echo <- readline(prompt = "Enter end_echo quantity: ")

mydata3 <- filter(mydata2, Start_echo > as.numeric(st_echo))

mydata3 <- filter(mydata3, End_echo < as.numeric(en_echo))

mydata3
```

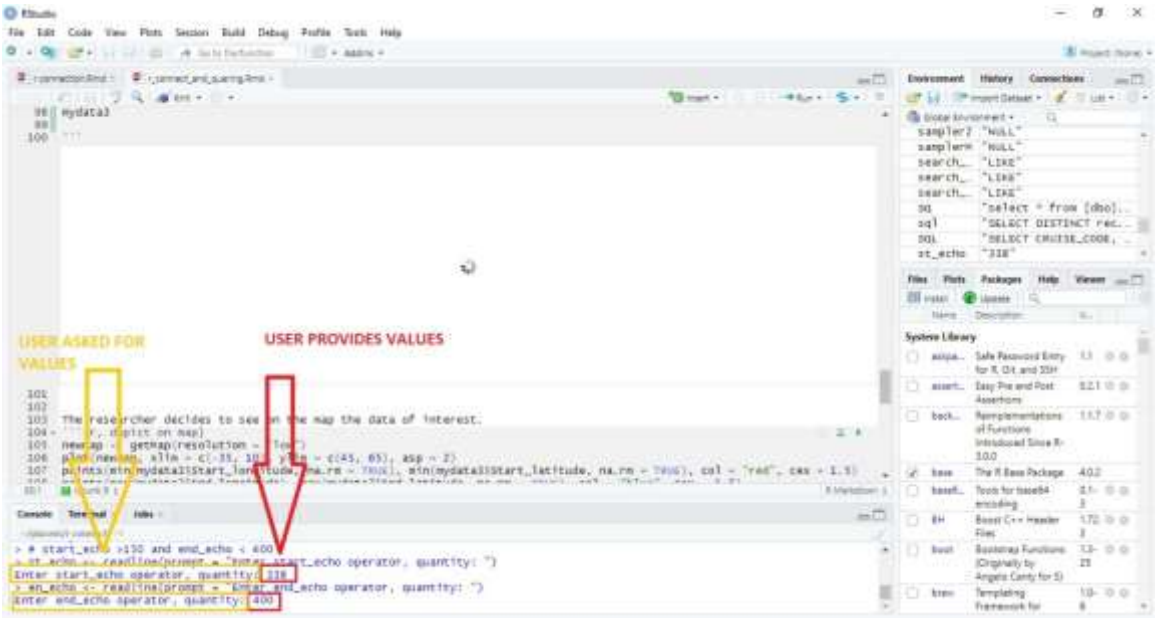


Figure 41 User asked for echo values

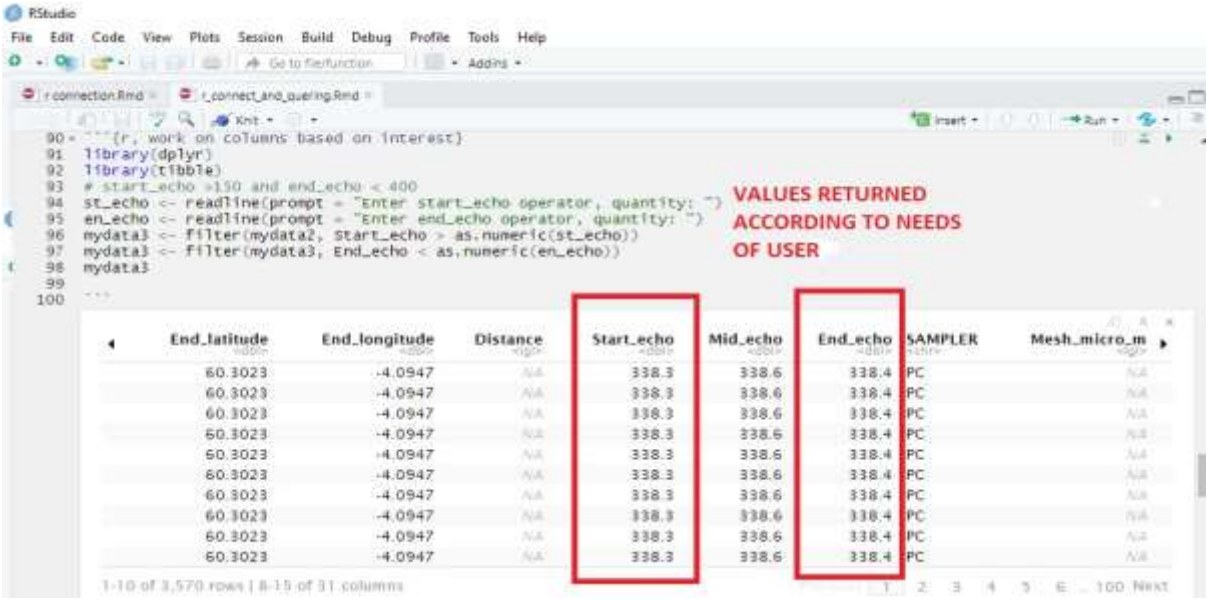


Figure 42 Data frame is filtered according to echo values inserted

The researcher decides to see on the map the data of interest. As mentioned by Marine Scotland, they are highly interested in the coordinates and the depiction of information on maps, because this would show geographical changes of the zooplankton. The map depicted automatically of the general area of interest where all client's measurements have taken place (Figure 43).

```
# depict on map

newmap <- getMap(resolution = "low")

plot(newmap, xlim = c(-35, 10), ylim = c(45, 65), asp = 2)

points(min(mydata3$Start_longitude, na.rm = TRUE), min(mydata3$Start_latitude, na.rm = TRUE), col = "red", cex = 1.5)
```

```
points(max(mydata3$End_longitude), max(mydata3$End_latitude, na.rm = TRUE), col = "blue", cex = 1.5)
points(mydata3$Start_longitude, mydata3$Start_latitude, col = "yellow", cex = .8)
points(mydata3$End_longitude, mydata3$End_latitude, col = "yellow", cex = .8)
# Add a legend
legend("topright", legend=c("Start lon/lat", "End long/lat", "In between points"),
       col=c("red", "blue", "yellow"), lty=1:2, cex=0.8)
```

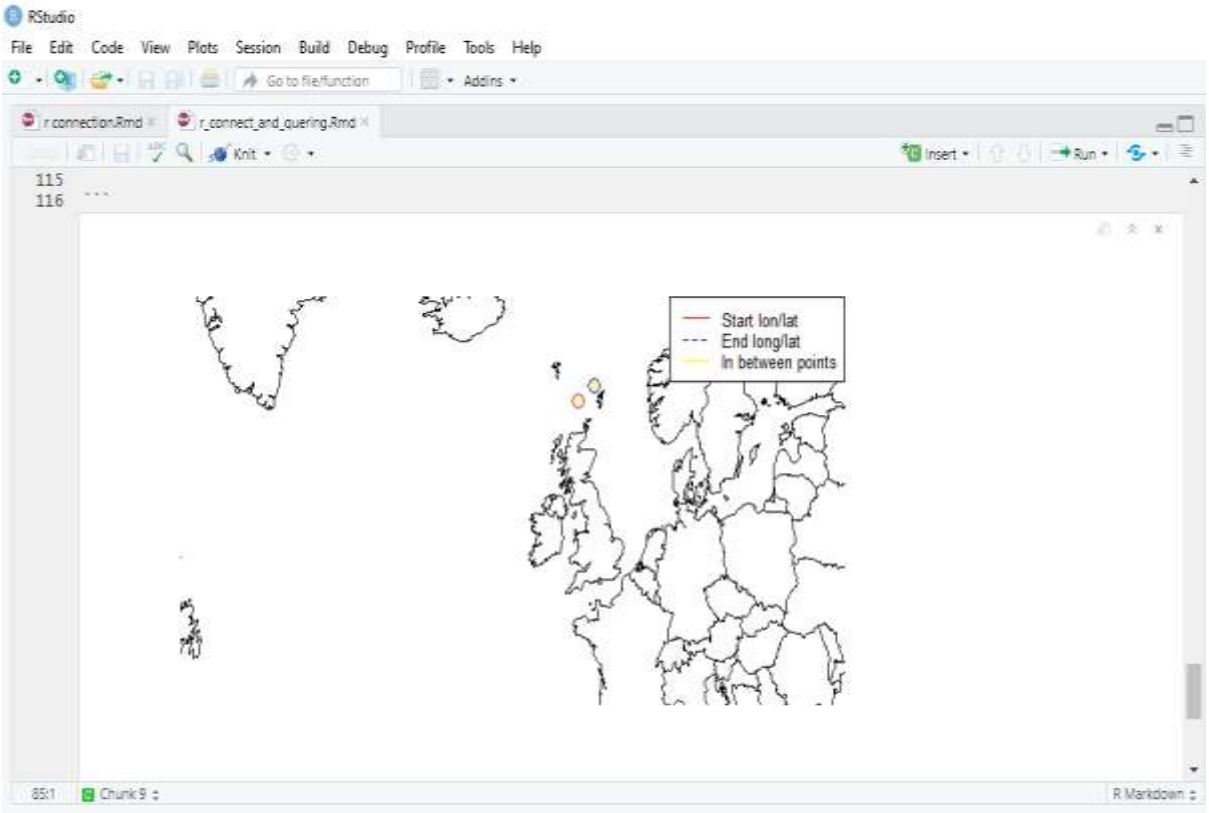


Figure 43 Values depicted on map

According to the interest of the researcher a step may be skipped or rerun if they wish.

SCENARIO_2

First the function that will handle the scripting that follows needs to be created.

```
# function to run scripts
g <- function(sql){
  sqlQuery(dbhandle,sql)
}
```

The researcher first needs to verify existing records that are available in both tables according to taxa names (Figure 44)

```
## SAMPLE TABLE ON GENERIC SEARCH BASED ON CRUISE CODE ##

sql <- 'SELECT DISTINCT sp.[Taxa_Names], sa.Cruise_code AS S_CRUISES, rec.Cruise_code AS REC_CRUISES FROM [dbo].[Species_info] AS sp

JOIN [dbo].[SAMPLES] as sa ON sa.CLASS = sp.[TITLE]

full JOIN [dbo].[RECORDS] AS rec ON sa.FK_REC_ID = rec.RECORDS_ID WHERE sp.[Taxa_Names] IS NOT NULL

ORDER BY sp.[Taxa_Names];'

df <- g(sql)

df
```

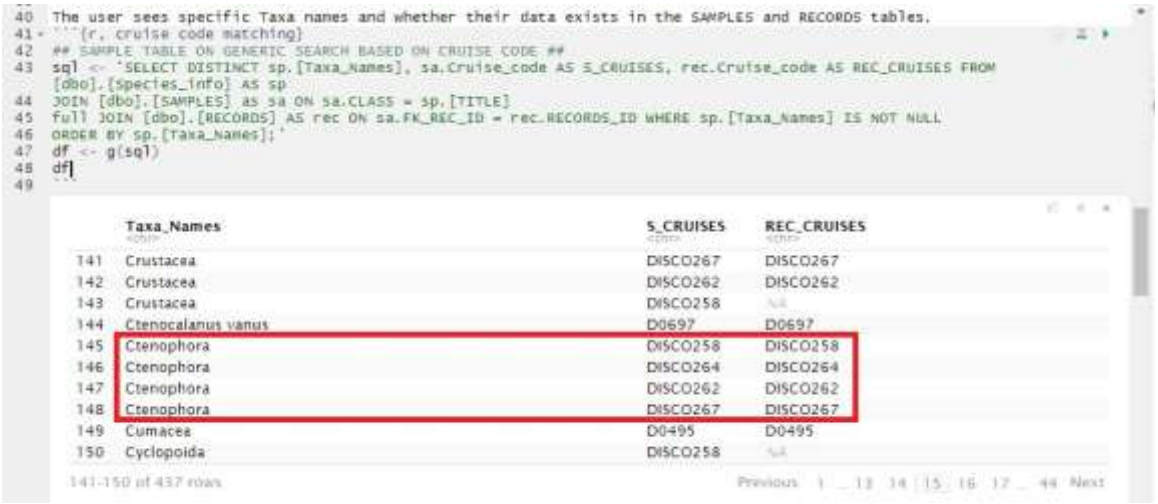


Figure 44 Checking existing records for taxa names

The user decides to search for "Ctenophora" (Figure 46)

```
taxa_name <- readline(prompt = "ENTER TAXA NAME IN ' ': ")

message2 <- paste("SELECT rec.*, sa.[CLASS], sa.[TYPE], sa.[VALUE], sp.[Taxa_Names], sp.[AphiaID], ct.[Project], ct.[BODC_link]

FROM [dbo].[CRUISES] AS ct JOIN [dbo].[RECORDS] AS rec ON ct.[UNIQUE_CRUISE_CODE] = rec.[CRUISE_CODE]

JOIN [dbo].[SAMPLES] AS sa ON rec.[RECORDS_ID] = sa.[FK_REC_ID]

FULL JOIN [dbo].[Species_info] AS sp ON sp.[TITLE] = sa.[CLASS] WHERE sp.[Taxa_Names] = ", taxa_name, ";")

df2 <- g(message2)

df2
```

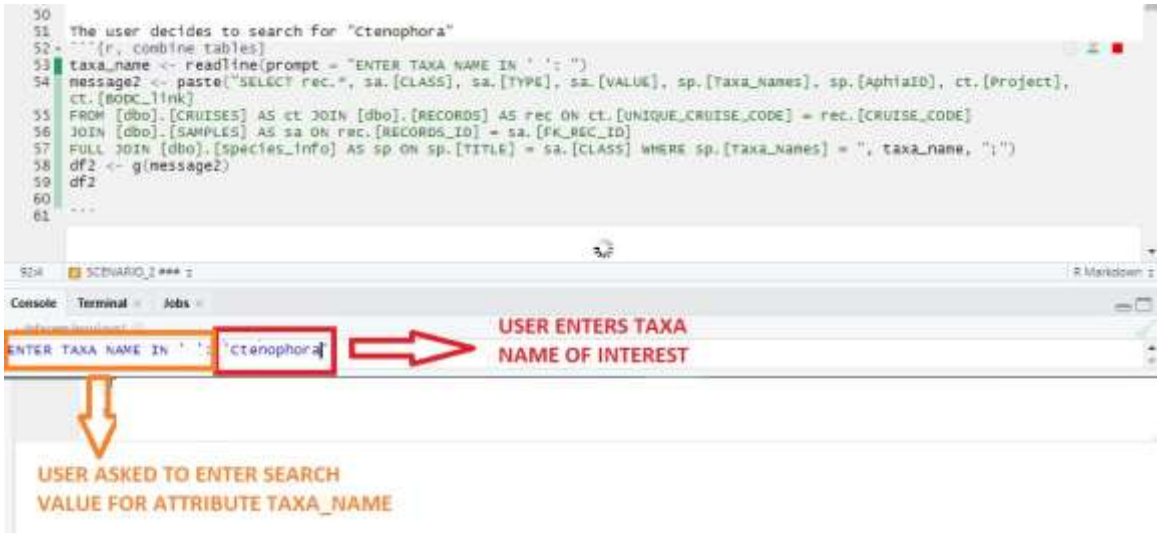


Figure 45 User asked to enter value for attribute 'Taxa_name'

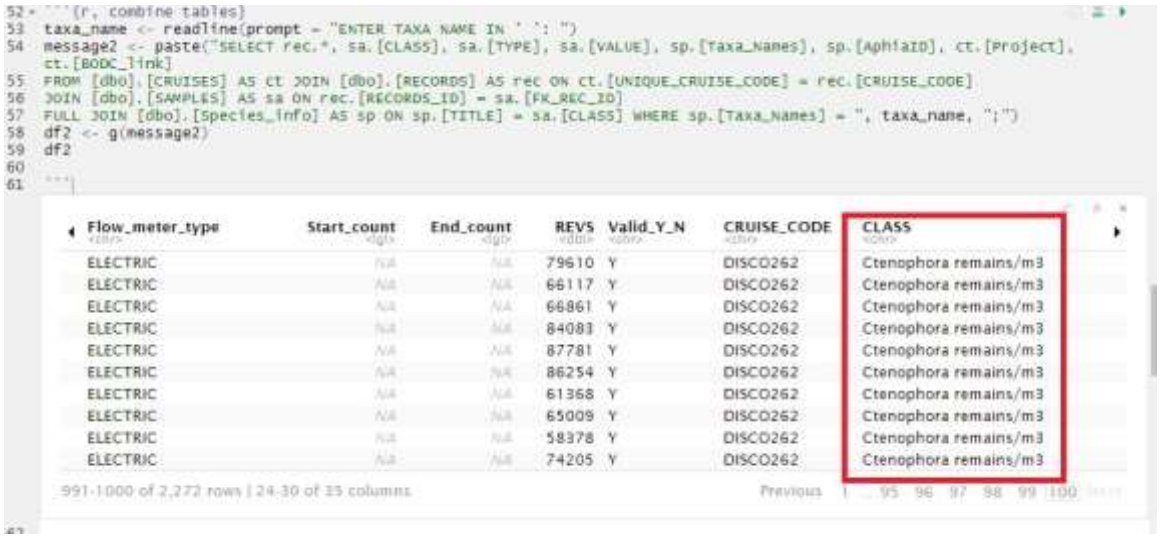


Figure 46 Result of search for 'Ctenophora'

The researcher decides that adequate information exists for further research. The retrieved information above is kept intact and copied as a data frame in R for further data mining (Figure 47).

```
df3 <- data.frame(df2)

# exclude columns of no use

mydata = select(df3, -c(RECORDS_ID, Start_count, End_count))

mydata

# filter out 0s

mydata2 <- filter(mydata, VALUE != 0.0)

mydata2

# filter according to Date
```

```
mydata3 <- tibble::as_tibble(mydata2)

mydata3 <- filter(mydata3, DATE >=  '2002')

mydata3
```

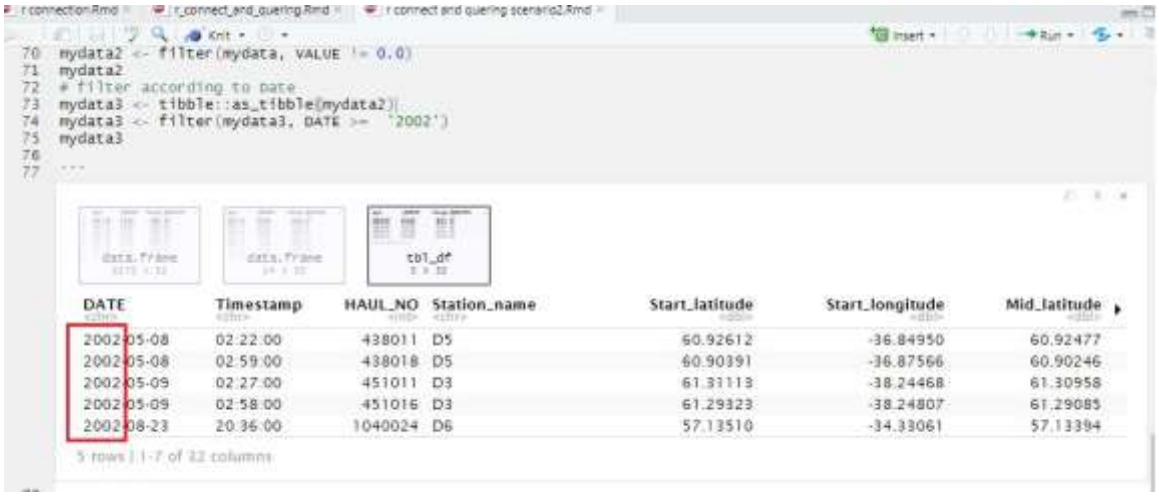


Figure 47 Filter out unwanted columns, 0s and filter for date

The researcher decides to see on the map the data of interest (Figure 48).

```
newmap <- getMap(resolution = "low")

plot(newmap, xlim = c(-35, 10), ylim = c(45, 65), asp = 2)

points(min(mydata3$Start_longitude, na.rm = TRUE), min(mydata3$Start_latitude, na.rm = TRUE), col = "red", cex = 1.5)

points(max(mydata3$End_longitude), max(mydata3$End_latitude, na.rm = TRUE), col = "blue", cex = 1.5)

points(mydata3$Start_longitude, mydata3$Start_latitude, col = "brown", cex = .8)

points(mydata3$End_longitude, mydata3$End_latitude, col = "brown", cex = .8)

# Add a legend

legend("topright", legend=c("Start lon/lat", "End long/lat", "In between points"),

    col=c("red", "blue", "brown"), lty=1:2, cex=0.8)
```

```
plot(newmap, xlim = c(-35, 10), ylim = c(45, 65), asp = 2)
points(min(mydata3$Start_longitude, na.rm = TRUE), min(mydata3$Start_latitude, na.rm = TRUE), col = "red", cex = 1.5)
points(max(mydata3$End_longitude), max(mydata3$End_latitude, na.rm = TRUE), col = "blue", cex = 1.5)
points(mydata3$Start_longitude, mydata3$Start_latitude, col = "brown", cex = .8)
points(mydata3$End_longitude, mydata3$End_latitude, col = "brown", cex = .8)
# Add a legend
legend("topright", legend=c("Start lon/lat", "End long/lat", "In between points"),
       col=c("red", "blue", "brown"), lty=1:2, cex=0.8)


```

Figure 48 Look data of interest on map

Appendix-Power BI

First connectivity between the database and Power BI need to be established. We open Power BI from desktop. At Home tab we choose Get Data→ SQL server (Figure 49)→in the pop-up window we enter the server name

and database name (Figure 50)→ At the new pop-up window we tick the tables we want and click Load (Figure 51)→ once the tables are successfully loaded we see them all on the right of our panel (Figure 52). From this point, one proceeds with the filtering and visualizations they prefer.

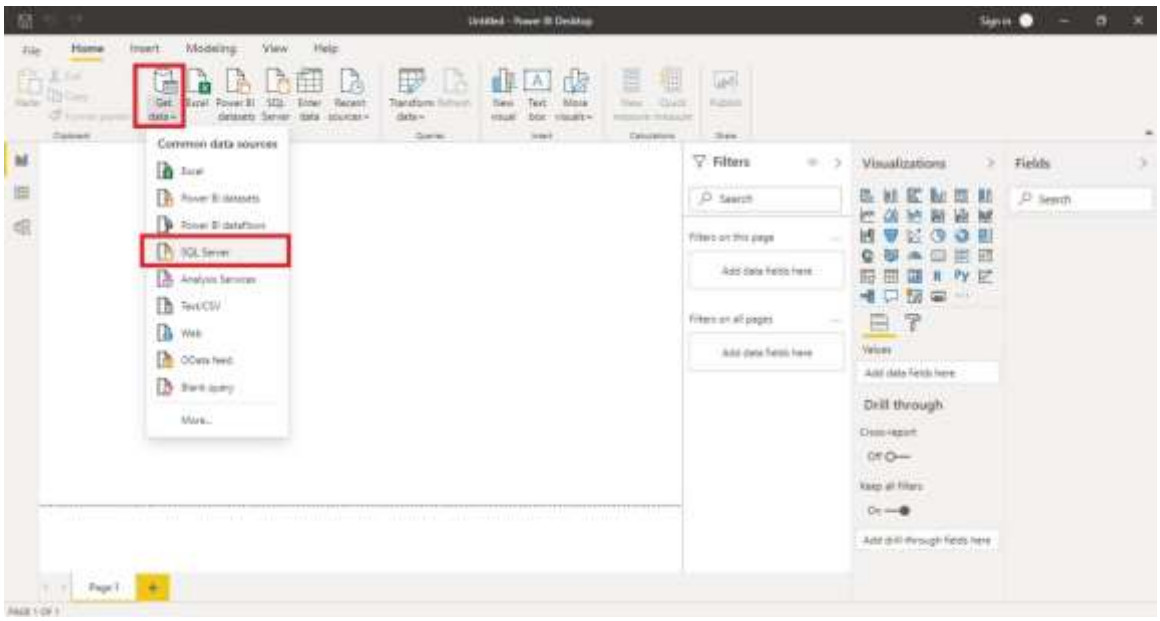


Figure 49 Import from SQL server

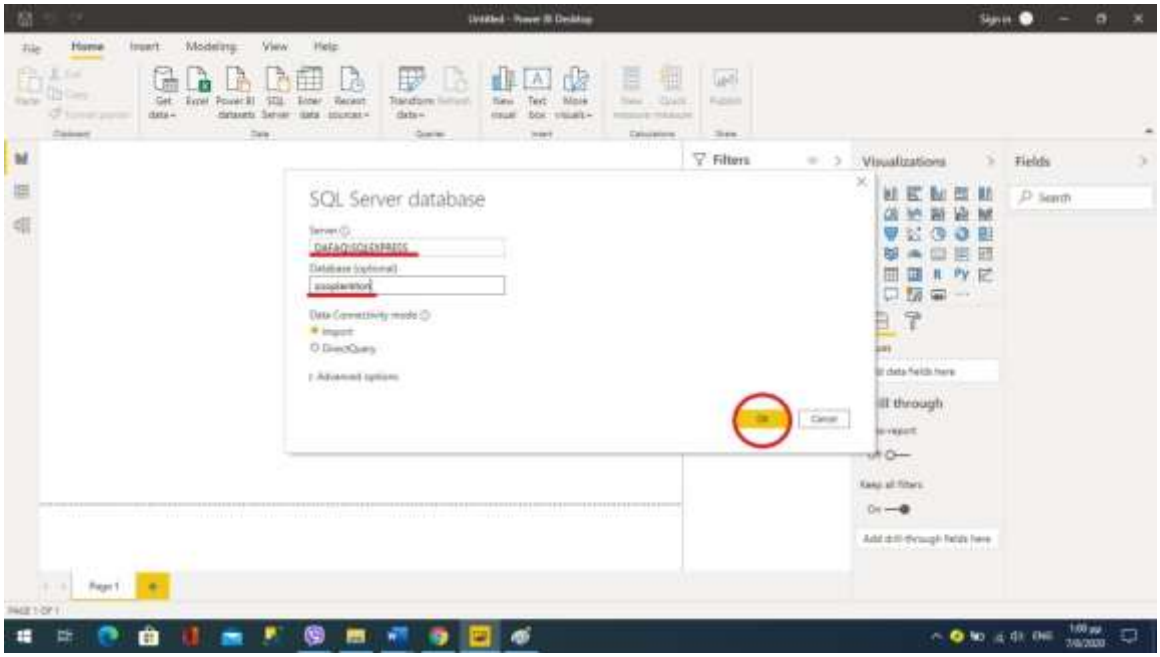


Figure 50 Insert credentials

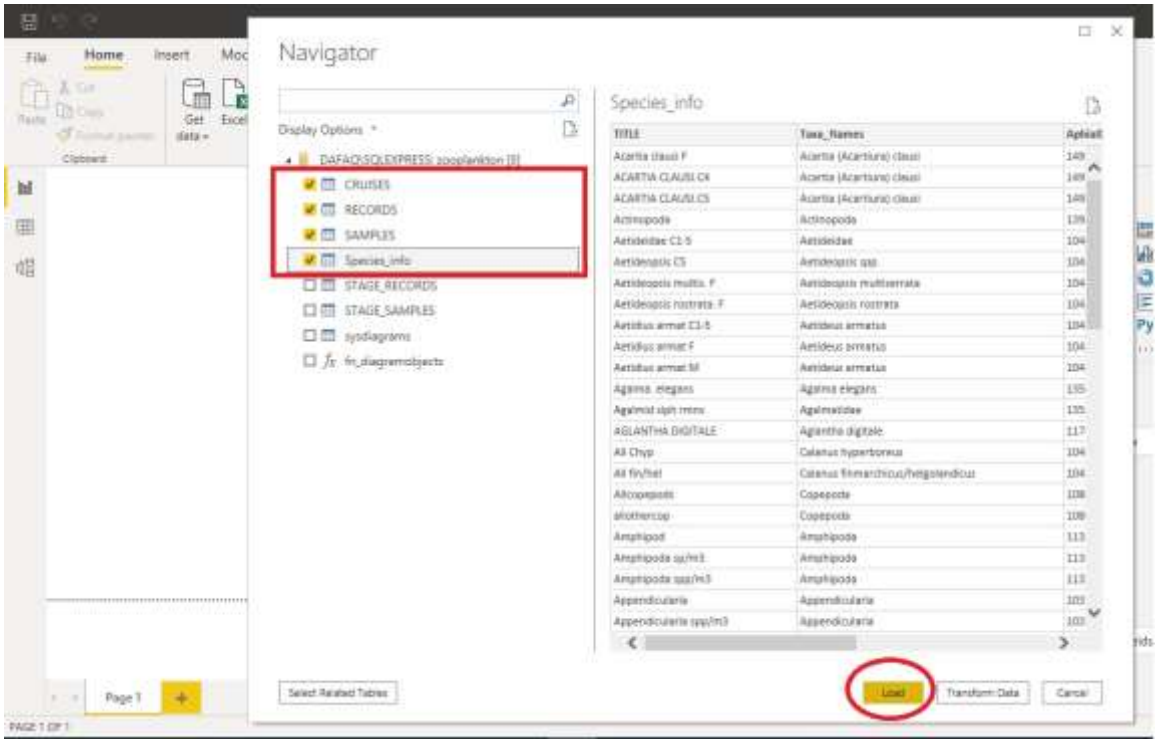


Figure 51 Load tables

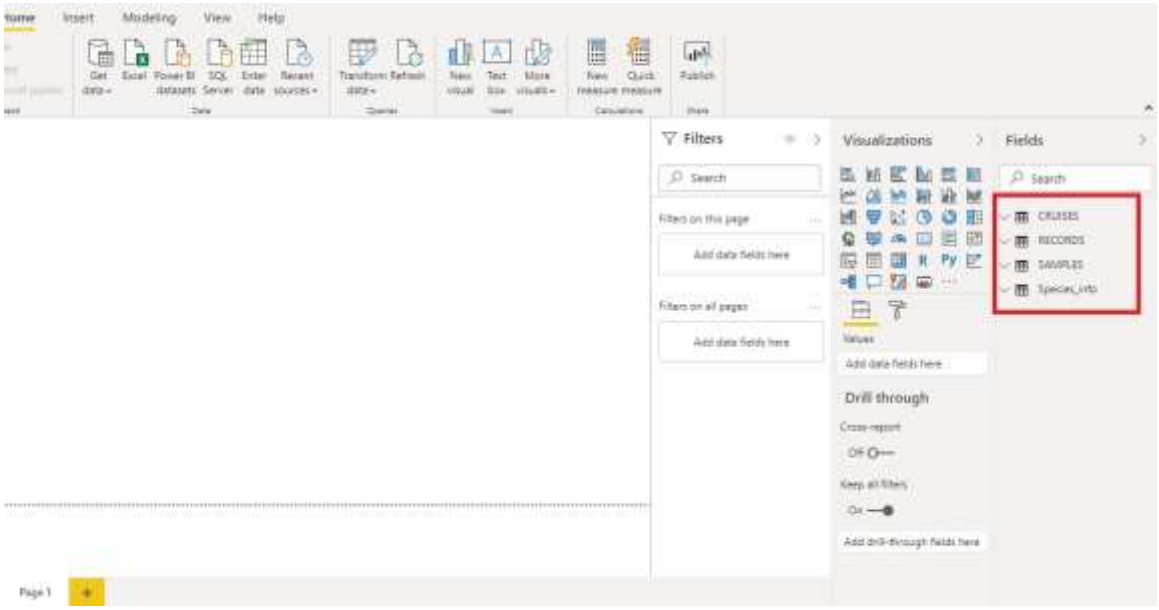


Figure 52 Successful import of tables

When we try to combine the data of two tables and for this example put stigmas on a an map we get the following response from Power BI (Figure 53). We click on “Fix this”.

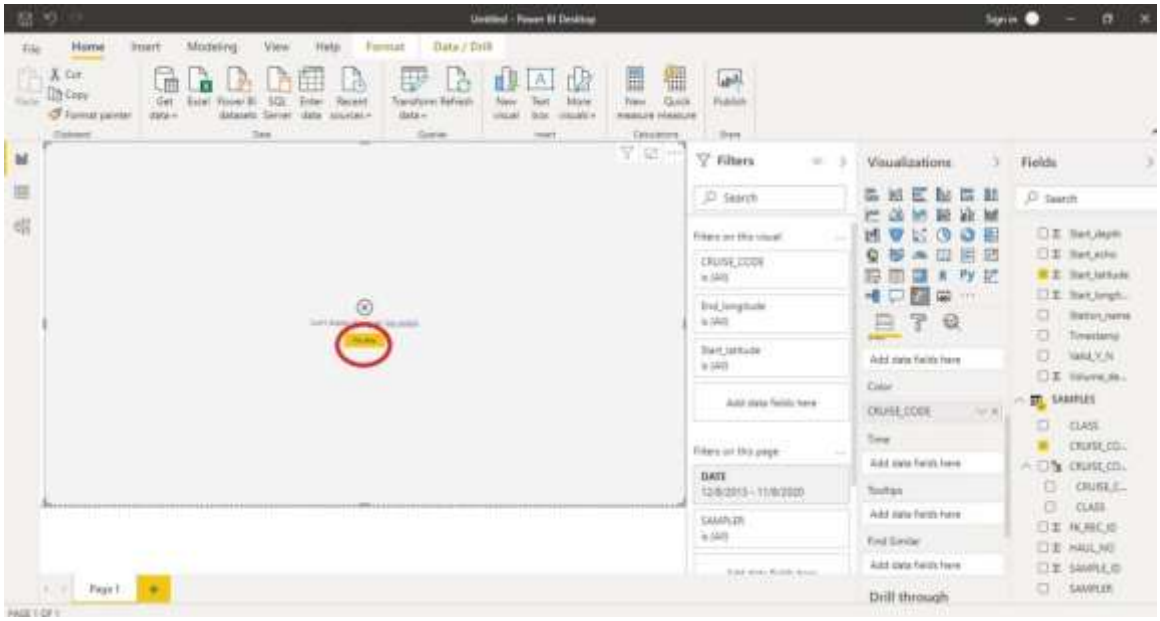


Figure 53 Map tables

We then click on “Autodetect” (Figure 54). Yet Power BI is incapable of auto-detecting relationships so we choose “Manage relationships” (Figure 55).

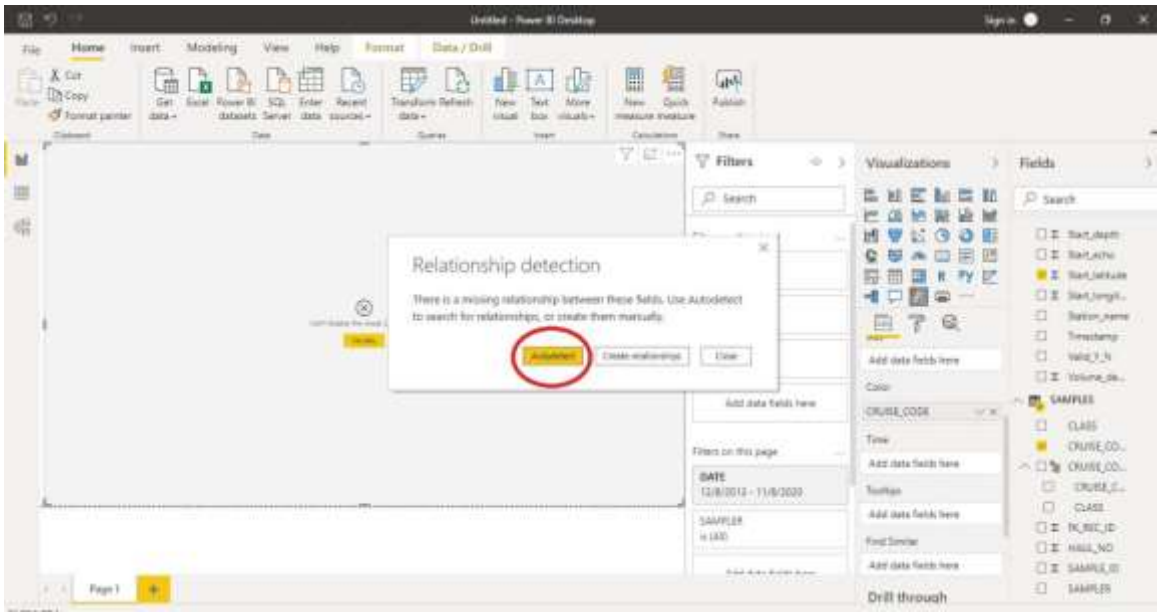


Figure 54 Relationship auto-detection

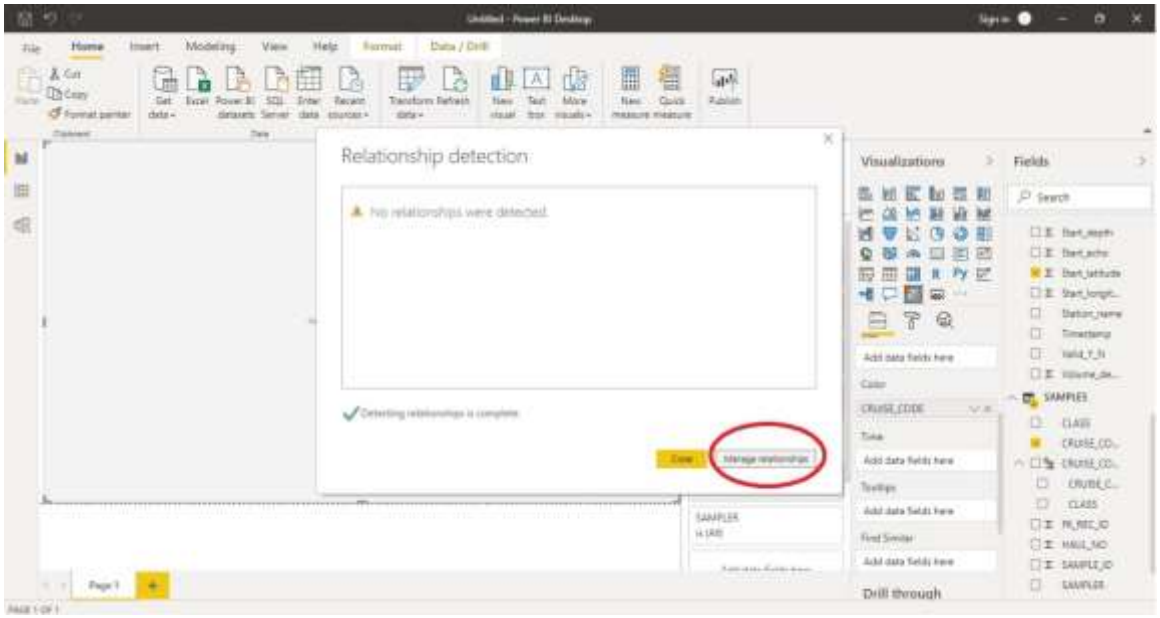


Figure 55 Manage relationship option

In the "Manage relationships" pop-up window we choose the relationship that is ticked, which is the one we have created in SSMS for this database, the RECORDS_ID to FK_REC_ID relationship (Figure 56).

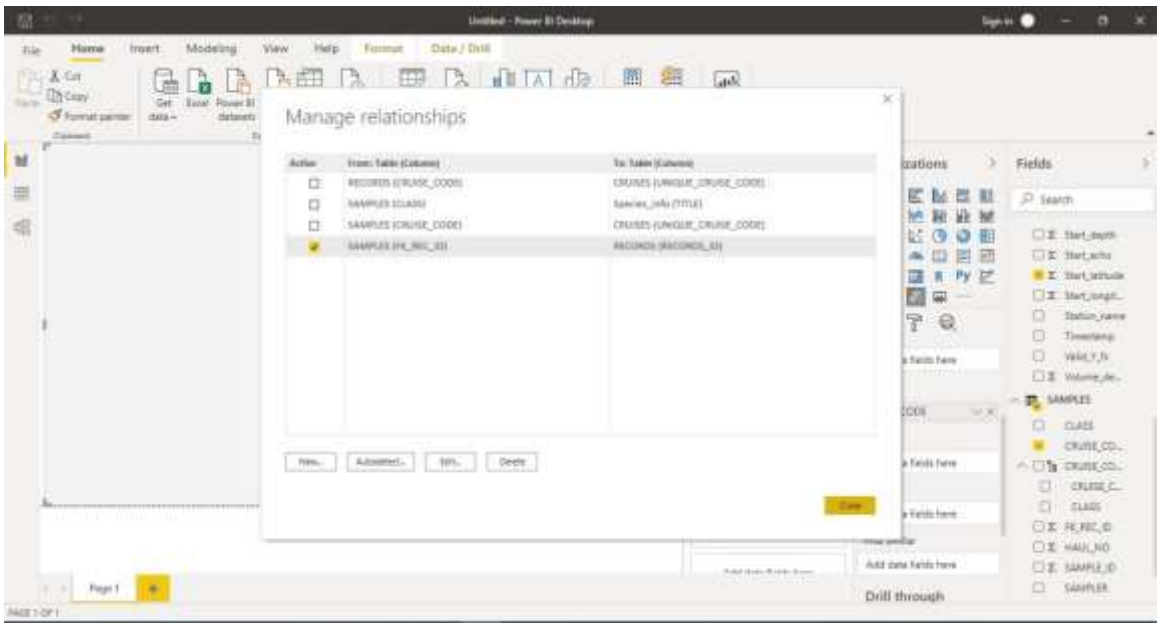


Figure 56 Choosing the correct relationship

Then we filter according to Sampler and Date and depict Cruise_code on the map (Figure 57).

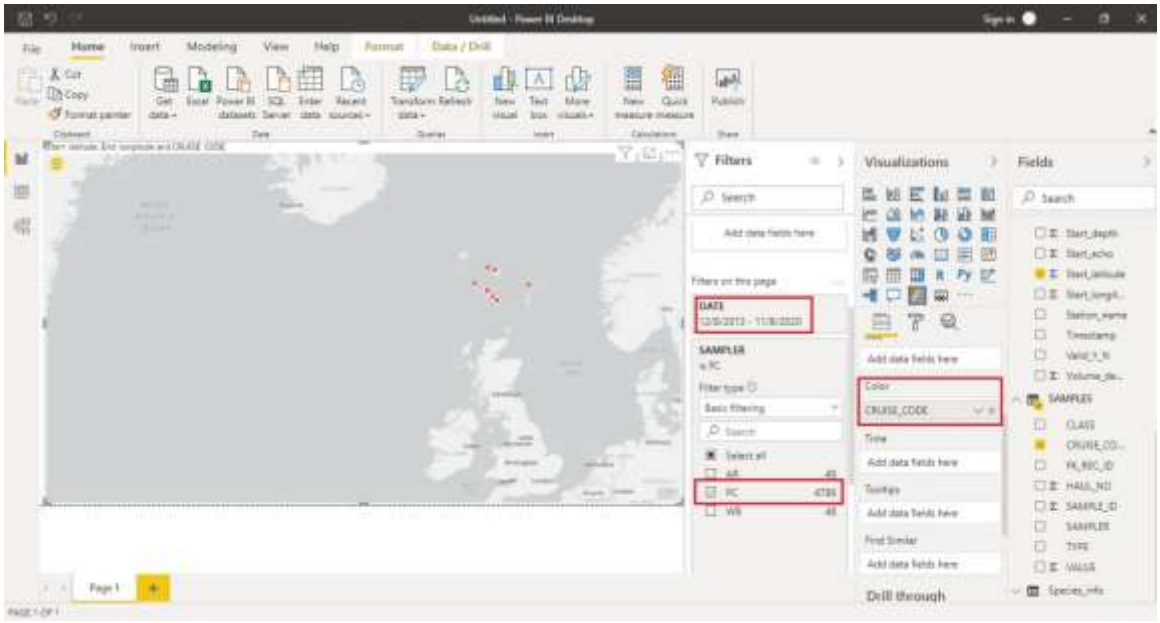


Figure 57 Multiple filtering

This is an example of how Power BI can easily provide initial information that can lead to insights for further data mining and analysis in RStudio.