

SSIS EXERCISES



Microsoft®
SQL Server®
Integration Services



TABLE OF CONTENTS

SOURCES, DESTINATIONS, AND SQL EXECUTE TASK	4
1. Export a CSV flat file to a SQL Server table	4
2. Import an Excel workbook of songs about food to SQL Server	5
3. Import a table of mentors from SQL Server to Excel	6
DATA CONVERSION	8
4. Import an Excel songs list to SQL Server, converting data	8
5. Import Excel data, redirecting data conversion errors	9
6. Export a table of Grand Prix events to a new flat file	10
MULTICAST, SORT, AND SAMPLING	12
7. Use Data Viewers to Show Sampled / Unsampled Data	12
8. Create a random playlist from an Excel workbook of songs	13
CONDITIONAL SPLIT AND ROW COUNT	14
9. Divide the Disney princesses into feminist role models/others	14
10. Use a conditional split to count two types of row in a table	14
LOOKUP	16
11. Get a list of all the dinosaur time periods which don't match	16
12. Find the id number for each X Factor mentor	17
UNION ALL, MERGE, AND MERGE JOIN	18
13. Find the differences between the 2015 and 2016 premier leagues	18
DERIVED COLUMN	19
14. Assign rollercoasters a scariness rating using a conditional split	19
AGGREGATE	20
15. Create aggregate shopping statistics depending on store type	20
PIVOT AND UNPIVOT	21
16. Collapse a two-dimensional array into a relational database	21

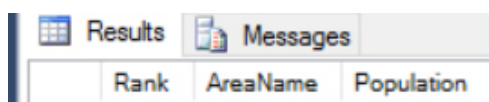
CACHE TRANSFORM	22
17. Find the differences between two lists of shopping centres	22
YOU'RE ON YOUR OWN!	23
18. Export X Factor Series Data to SQL Server Table	23
19. Import contestants, linking to separate category table	24
20. Show a summary of Brexit voting figures by status	25
21. Import multiple CSV files of Big Bang data into SQL Server	25
22. Add new genres to a lookup table, then import a list of films	26
HELP!	27
1. Exercise 8: Create a random playlist from an Excel workbook of songs	27
2. Exercise 9: Divide the Disney princesses into feminist role models/others	27
3. Exercise 12: Find the id number for each X Factor mentor	28
4. Ex. 13: Find the differences between the 2015 and 2016 premier leagues	29
5. Ex. 14: Assign rollercoasters a scariness rating using a conditional split	30
6. Ex. 15: Create aggregate shopping statistics depending on store type	31
7. Ex. 16: Collapse a two-dimensional array into a relational database	31
8. Ex. 17: Find the differences between two lists of shopping centres	32
9. Ex. 19: Import contestants, linking to separate category table	33
10. Ex. 20: Show a summary of Brexit voting figures by status	34

(Source: Wise Owl)

SOURCES, DESTINATIONS, AND SQL EXECUTE TASK

1. Export a CSV flat file to a SQL Server table

In SQL Server Management Studio, open the script called Generate table of built-up areas in the above folder and execute it to show an empty table:



Rank	AreaName	Population
------	----------	------------

The folder above also contains a CSV file listing the top 20 UK built-up areas by population (the data comes from Wikipedia, and is from the 2011 census):

```
Rank,Built-up area,Population,SquareKm
1,Greater London area,"9,787,426","1,737.90"
2,Greater Manchester area,"2,553,379",630.3
3,West Midlands area,"2,440,986",598.9
4,West Yorkshire area,"1,777,934",487.8
5,Greater Glasgow area,"1,209,143",368.5
6,Liverpool area,"864,122",199.6
```

Create a package called **Go Manchester** which uses an **Execute SQL** task to delete any old rows from the **UKBuiltUpAreas** table, then uses a data flow task to import the areas from the CSV file into this table. Here's what the control flow and data flow could look like:



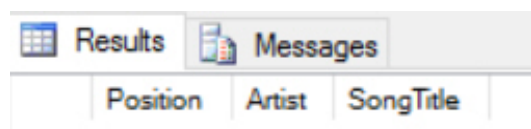
Run your package a couple of times to check that it gives you the top 20 built-up areas in the UK (as of the 2011 census):

Rank	AreaName	Population
1	Greater London area	9787426
2	Greater Manchester area	2553379
3	West Midlands area	2440986
4	West Yorkshire area	1777934

The first few of the 20 areas.

2. Import an Excel workbook of songs about food to SQL Server

In SQL Server Management Studio, open the script in the above folder and execute it to get an empty table called **FoodMusic**:



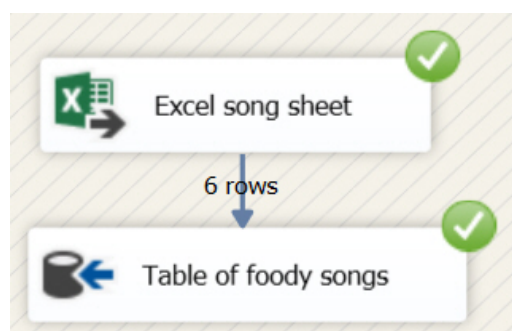
Position	Artist	SongTitle
----------	--------	-----------

Open the Excel workbook in the above folder to have a look at its contents:

	A	B	C
1	Position	Artist	Song
2	1	Don McLean	American Pie
3	2	Booker T and the MG's	Green Onions
4	3	The Beatles	Strawberry Fields Forever
5	4	The Rolling Stones	Brown Sugar
6	5	Fats Domino	Blueberry Hill
7	6	Abba	Honey, Honey

You might be beginning to guess where this is going ...

Close the Excel workbook, and create a package called **If music be the food** which imports the Excel workbook's data into the SQL Server table:



Your data flow task could look something like this.

Run this package - you should get a table of food songs in the **FoodMusic** table in SQL Server:

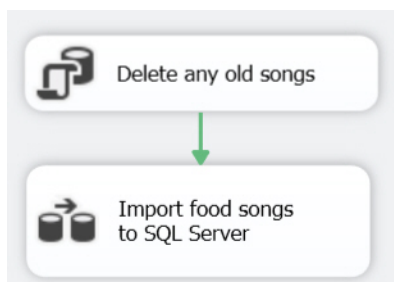
Position	Artist	SongTitle
1	Don McLean	American Pie
2	Booker T and the MG's	Green Onions
3	The Beatles	Strawberry Fields Forever
4	The Rolling Stones	Brown Sugar
5	Fats Domino	Blueberry Hill
6	Abba	Honey, Honey

The table is populated when you run this package.

Run the package a second time - you should find you get another set of 6 songs. Add an **Execute SQL** task to the control flow of your package to run this command:

```
TRUNCATE TABLE FoodMusic
```

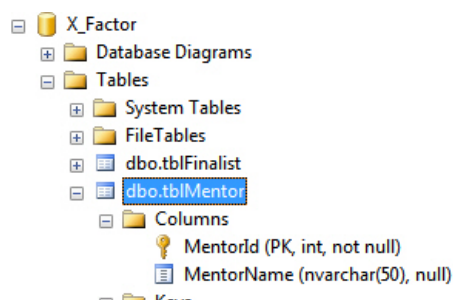
Here's what your final control flow could look like:



Run the package to check that you only get 6 songs in the SQL Server table.

3. Import a table of mentors from SQL Server to Excel

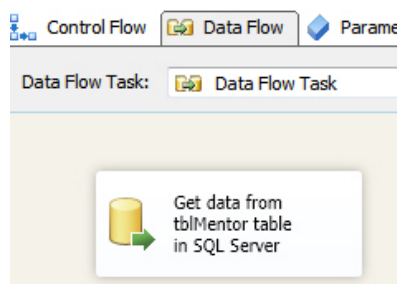
In SQL Server Management Studio, open the script called **Create mentors table.sql** in the above folder and execute it:



You should now have a table of mentors ready for use.

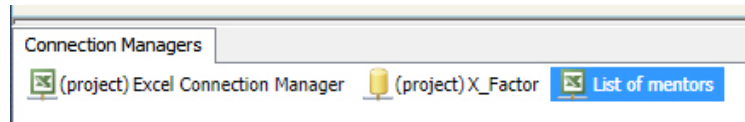
Create a new package, and rename it as **Mentors to Excel.dtsx**.

Create a data flow task inside this package, and within this create a source which takes its data from the **tblMentor** table in the **X_Factor** database:



You can use the existing OLEDB connection manager for this OLEDB source.

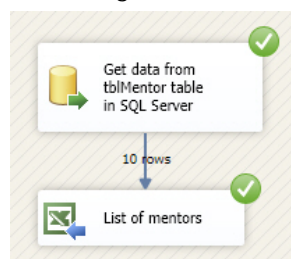
Create a new connection within this package which will link to the Excel file called **List of mentors.xlsx** in the above folder:



Your new Excel connection should link to the worksheet called **List of mentors** in the file called **List of mentors.xlsx**!

Create an Excel destination within your data flow task based on this connection.

Connect the source to the destination and map the columns. When you run this package, you should see something like this:



Ten rows have been transferred into the Excel worksheet.

If you open the workbook called **List of mentors.xlsx**, you should see this:

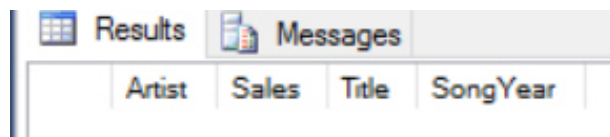
	A	B
1	Mentor number	Name of mentor
2	1	Cheryl Cole
3	2	Dannii Minogue
4	3	Gary Barlow
5	4	Kelly Rowland
6	5	Louis Walsh
7	6	Nicole Scherzinger
8	7	Sharon Osbourne
9	8	Simon Cowell
10	9	Unknown
11	10	David Wakefield
12		

We could have done with converting the mentor id to a number before exporting it, but softly softly, catchy monkey ...

DATA CONVERSION

4. Import an Excel songs list to SQL Server, converting data

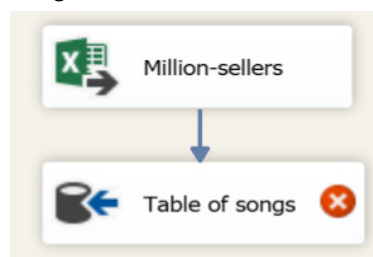
First go into SQL Server Management Studio. Open the SQL file in the above folder and execute it, to get an empty table called **MillionSellers**:



Artist	Sales	Title	SongYear
--------	-------	-------	----------

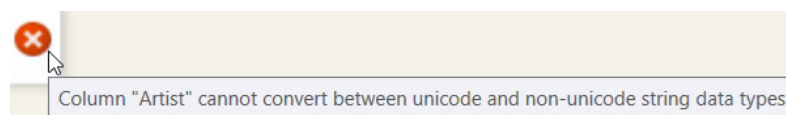
Hmmm ... I wonder what data type those silly owls have used for the **Artist** and **Title** columns?

Create a package called **Song converter**, and use this to try to import the Excel workbook of million-selling songs in the above folder into the SQL Server table that you've just created:



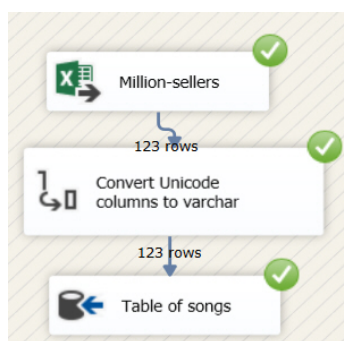
You should now have an error - if you like, you can miss out this stage and go straight to creating the final answer.

Let your mouse hover over the red error icon and you'll see the nature of your problem:



It's those pesky Unicode text strings!

Add a data conversion transform into your data flow so that you can successfully export the Excel workbook of songs to the SQL Server table:

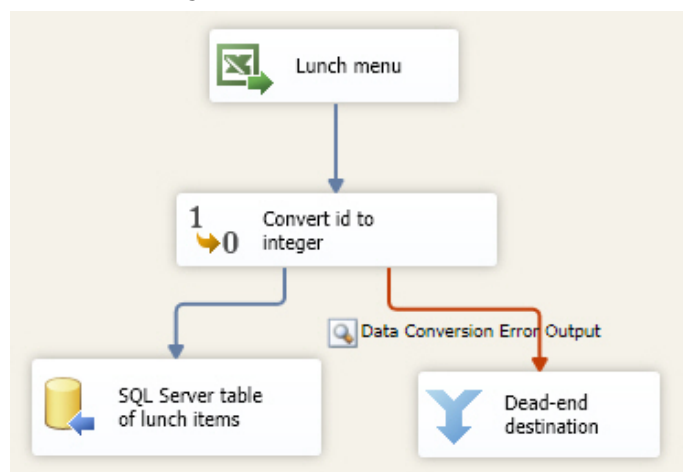


What your final data flow task could look like.

5. Import Excel data, redirecting data conversion errors

Run the script in the above folder in SQL Server Management Studio to create a table of 4 Bushtucker Trial menu items.

Create a package which tries to import additional items from the Excel workbook above into this table:



The data flow task consists of an Excel source, a data conversion task, an OLEDB destination and a Union All destination. You should add a data viewer to see redirected errors.

You should configure the error output for the data conversion task to redirect errors.

When you run this package, you should see two errors in the data viewer:

Data Conversion Error Output Data Viewer at Import lunch				
<input type="button" value="Detach"/> <input type="button" value="Copy Data"/>				
Item id	Item name	Erro...	Erro...	ErrorCode - Description
Seven	Celery	-1...	6	The data value cannot be converted f...
Nine	Foam bananas	-1...	6	The data value cannot be converted f...

The two data conversion errors - SSIS can't convert **Seven** and **Nine** into integers.

Test that your package has added some (but not all) rows to the **tblBushTuckerItem** table, then close it down.

Check that your package has added 123 songs into the SQL Server table, then close it down.

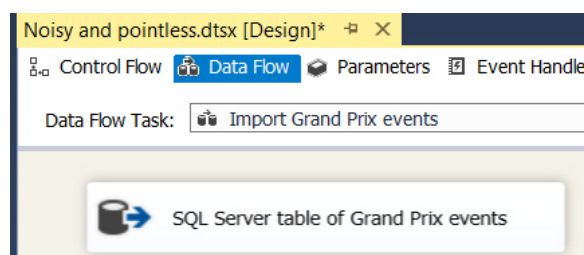
6. Export a table of Grand Prix events to a new flat file

To do this exercise, first open the script in the above folder in Management Studio and execute it to get a table called **GrandPrixEvents**:

GrandPrixId	Round	VenueName	Season
28	8	European Grand Prix	2016
29	9	Austrian Grand Prix	2016
30	10	British Grand Prix	2016
31	11	Hungarian Grand Prix	2016

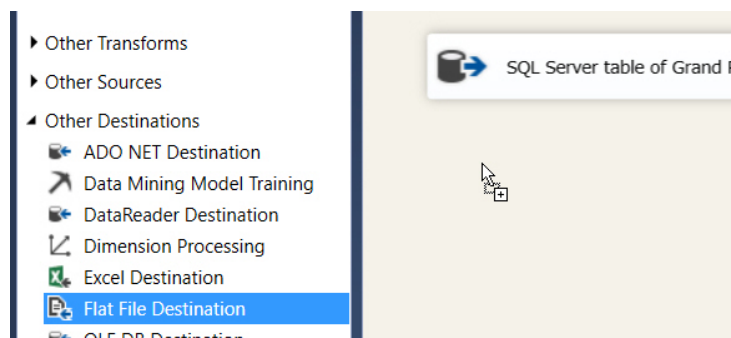
The first few rows generated by the above script.

Now create a package called **Noisy and pointless**, and within this create a data flow task using the above table as a source:



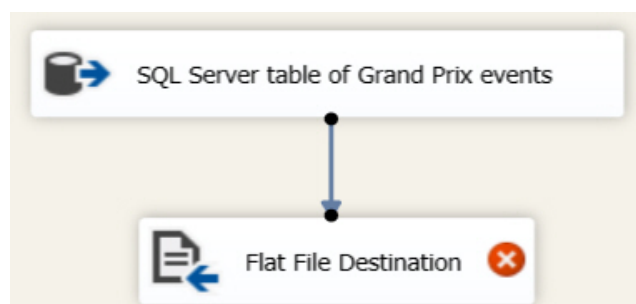
Your new package should take data from the table you've just generated in SQL Server.

Add a flat file destination like this:



Click on the **Flat File** destination, and drag it into your data flow task.

Now connect the source to the destination:



Feed the output from the source into the destination.

You should now be able to configure the destination to create a new flat file called **Grand Prix venues**:

```
GrandPrixId,Round,VenueName,Season
1,1,Australian Grand Prix,2017
2,2,Chinese Grand Prix,2017
3,3,Bahrain Grand Prix,2017
4,4,Russian Grand Prix,2017
5,5,Spanish Grand Prix,2017
6,6,Morocco Grand Prix,2017
```

The start of the text file generated when you run your package.

You'll need to create a new flat file connection as part of configuring the flat file destination.

When your package is working, check that it's generated a suitable text file, then close it down.

MULTICAST, SORT, AND SAMPLING

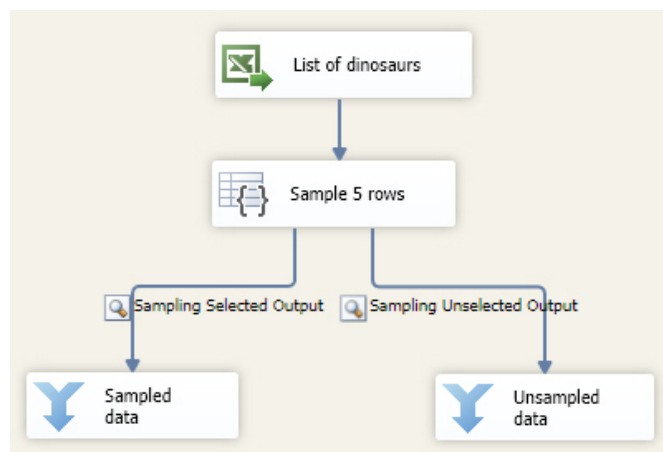
7. Use Data Viewers to Show Sampled / Unsampled Data

Create a package called **Prehistoric Enumeration**, and within this a source linked to the dinosaurs workbook in the above folder.

	A	B	C	D	E	F
1	Name	Meaning of Name	Pronunciation	Period	Diet	Length (i
2	Albertosaurus	Alberta lizard	al-BERT-oh-saw-russ	Cretaceous	carnivore	9
3	Allosaurus	different reptile	AL-oh-saw-russ	Jurassic	carnivore	12
4	Baryonyx	heavy claw	bah-ree-ON-icks	Cretaceous	carnivore	10
5	Coelophysis	hollow form	see-LOH-fiss-iss	Triassic	carnivore	3

The start of the list of dinosaurs in the Excel workbook.

Add a **Row Sampling** transformation and two **Union All** destinations to create this package:



We send the 5 sampled rows down the left pipe, and all of the others down the right pipe. Only include the dinosaur name, its period and its diet.

Add data viewers to each pipe so that you can see what's happening:

Sampling Selected Output Data Viewer at Data Flow Task			
Detach Copy Data			
Name	Period	Diet	
Parasaurolophus	Cretaceous	herbivore	
Chasmosaurus	Cretaceous	herbivore	
Centrosaurus	Cretaceous	herbivore	
Deinonychus	Cretaceous	carnivore	
Pachycephalosaurus	Cretaceous	herbivore	
Attached	Total rows: 0, buffers: 0		Rows displayed :

This data viewer, for example, is showing the 5 randomly selected rows (don't forget that yours will select different rows).

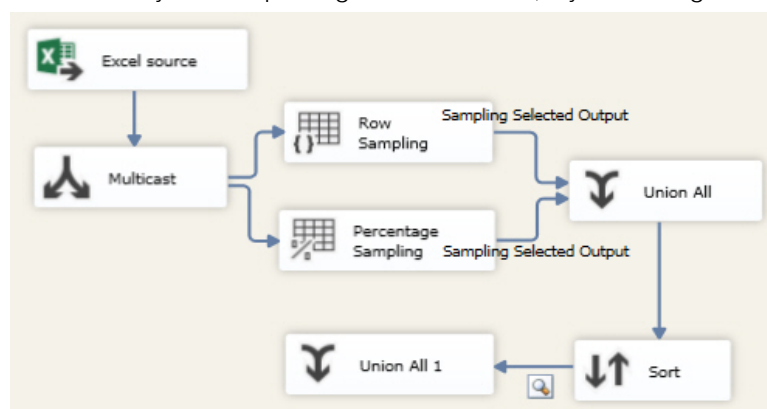
8. Create a random playlist from an Excel workbook of songs

The aim of this exercise is to create a playlist of about 5 songs, although by a fairly roundabout route! First create a package called **Playlist**, and within this a data flow task which includes the following:

What	What it does
Excel source	Import the 123 songs from the Excel workbook in the folder above.
Multicast transform	Duplicate this list of songs.
Row sampling transform	Sample 3 rows from the left-hand list.
Percentage sampling transform	Sample 1% of the rows from the right-hand list
Union all transform	Combine the two paths into one
Sort transform	Sort the songs so the one with the highest sales comes first
Union all transform	Add this, and put a data viewer on the data pipe leading to it so you can see which songs you've chosen!

All of this is much easier to understand with a flow chart - for which read on!

Here's what your final package could look like, if you didn't give any of the transforms nice names:



A possible final flow chart for your package

Here's what your playlist should look like (your songs will obviously be different):

Artist	Rank	Sales	Title	Year
BONEY M	6	2	RIVERS OF BABYLON/BROW...	1978
HARRY BELAFONTE	65	1.18	MARY'S BOY CHILD	1957
LEONA LEWIS	105	1.04	BLEEDING LOVE	2007
BING CROSBY WITH THE KEN...	116	1.01	WHITE CHRISTMAS	1942

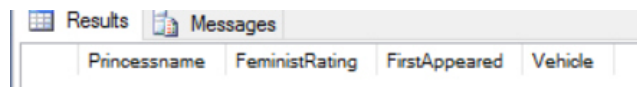
A mix of styles here ...

Generate a couple of playlists, then close your package down.

CONDITIONAL SPLIT AND ROW COUNT

9. Divide the Disney princesses into feminist role models/others

Open the SQL script in the above folder in Management Studio and execute it. You should be looking at an empty table:



Princessname	FeministRating	FirstAppeared	Vehicle
--------------	----------------	---------------	---------

We'll aim to fill the table with the feisty princesses only.

Create a new package which gets its data from the Excel workbook in the above folder, then use a conditional split to divide the data into 3 streams:

Condition	Send to
Feminist rating column equals Feisty	The SQL Server table called Princesses
Feminist rating column is not Drippy	A new flat file called Not drippy.txt
Everyone else	Ignored (as indeed they should be)

Here's what the **Princesses** table should hold:

Princessname	FeministRating	FirstAppeared	Vehicle
Belle	Feisty	1991	Beauty and the Beast
Jasmine	Feisty	1992	Aladdin

Perhaps Pocahontas should have been here too?

Test your package generates a text file of non-drippy, non-feisty princesses, then close it down.

10. Use a conditional split to count two types of row in a table

The above folder contains a CSV file of sightings of whales and porpoises by the Hebridean Whale and Dolphin Trust. Create a package called No real porpoise to import this data.

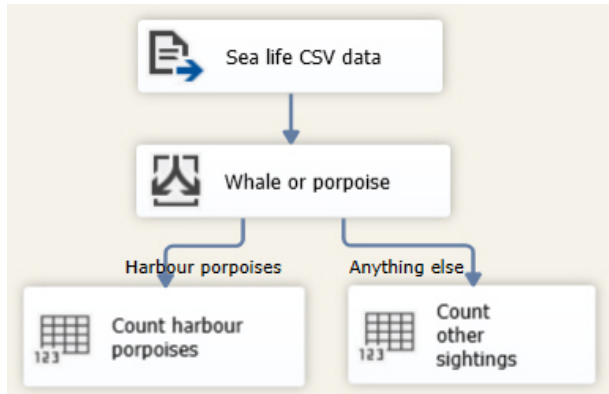
Create two variables as follows:

Variable name	What it will be used for
NumberHarbourPorpoises	Count the number of harbour porpoises seen
NumberOtherSightings	Count the number of sightings of other animals

The aim of this exercise is to create a conditional split with the following rules:

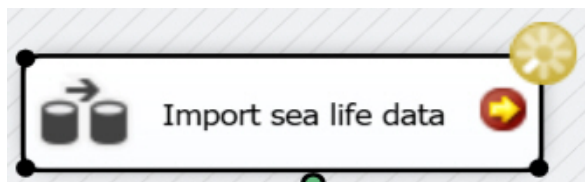
Rule	Named path
The Species column is Harbour Porpoise	Harbour porpoises
All other sightings	Anything else

Create a data flow task which takes the sea life sightings, and splits them into two using the rules above:



Send harbour porpoise sightings down one path, and all other sightings down the other, then use **Row Count** transforms to count the results.

Add a breakpoint to the **PostExecute** event of your control flow task:



Package execution should pause when this task has finished execution.

Use your **Locals** window to check the value of your two variables has been set correctly:

Locals		
Name		Value
System::LocaleID		{2057}
System::MachineName		{HUMPH}
User::NumberHarbourPorpoises		{618}
User::NumberOtherSightings		{710}
System::OfflineMode		{False}

The values that you should see for each variable.

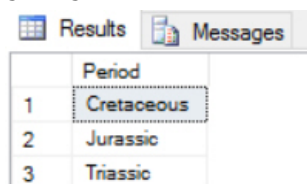
The harbour porpoise (**Phocoena Phocoena**) is one of six species of porpoise. It is one of the smallest marine mammals - and also one of the most common, if the figures above are anything to go by!

Stop debugging your package, and close it down.

LOOKUP

11. Get a list of all the dinosaur time periods which don't match

In SQL Server Management Studio, open the SQL script in the above folder and execute it to get a table of geological periods:

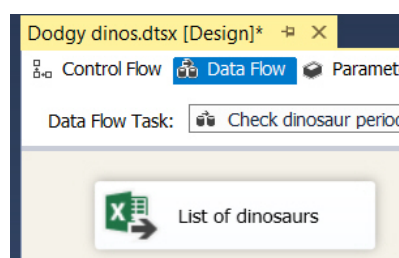


The screenshot shows the 'Results' tab in SQL Server Management Studio. It displays a table with two columns: 'Period' and an unnamed column with values 1, 2, and 3. The 'Period' column contains the values 'Cretaceous', 'Jurassic', and 'Triassic'.

	Period
1	Cretaceous
2	Jurassic
3	Triassic

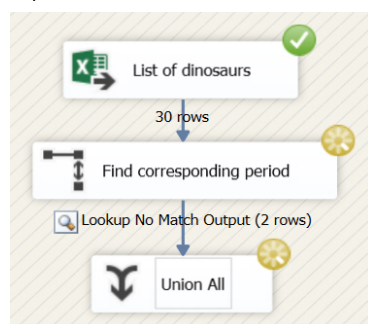
The geological periods to which dinosaurs can belong - we hope.

Create a package called Dodgy Dinos, and within this create a data flow task which imports the list of dinosaurs from the Excel workbook in the above folder:



Untick all of the columns apart from the first four.

Now add a lookup transform to look up the period for each dinosaur in the SQL Server table Periods, and report the errors:



Send the non-matching rows down to a Union All transform, and use a data viewer to see the periods which don't match.

We won't spoil the surprise by revealing the anomalous data, but the two offending dinosaurs should show up in your data viewer.

When this is all working, close the package down.

12. Find the id number for each X Factor mentor

In SQL Server Management Studio open the SQL script in the above folder and execute it to show a table of mentors for X Factor contestants:

MentorId	MentorName
1	Cheryl Cole
2	Dannii Minogue
3	Gary Barlow
4	Kelly Rowland
5	Louis Walsh
6	Nicole Scherzinger
7	Sharon Osbourne
8	Simon Cowell
9	Unknown

In the UK version of the X Factor (a reality TV show) each contestant is assigned a mentor to guide them through the procedures. That's all you need to know about the show!

The idea behind this exercise is to find the contestants whose mentors don't appear in the list above, and report them as having mentor number 9 (**Unknown**) instead. To start, create a package called **Mentors**, and within this create a variable to make it easier to refer to the unknown mentor id number:

Variables

Name	Scope	Data type	Value	Ex
UnknownMentorId	Mentors	Int32	9	

Create an integer variable, and set it to equal 9.

Create a data flow task to import the finalists from Excel, and look up for each the mentor id from the **Mentor** table.

Check that in the **Finalist** table that each contestant has a mentor id (six of these will be 9, for **Unknown**), then close the package down.

UNION ALL, MERGE, AND MERGE JOIN

13. Find the differences between the 2015 and 2016 premier leagues

Have a look at the two worksheets in the workbook in the above folder. You should see that they contain the Premier League final tables for the 2015 and 2016 seasons:

	A	B	C	D	E	F
1	Team	Played	Won	Drawn	Lost	Points
2	Chelsea	38	26	9	3	87
3	Manchester City	38	24	7	7	79
4	Arsenal	38	22	9	7	75

The start of one of the worksheets. Your task is to work out which teams changed!

Create a package called **Moon or parrot** which shows all the discrepancies between the two tables:

Team	Team (1)
NULL	AFC Bournemouth
Burnley	NULL
Hull City	NULL
Leicester City	NULL
NULL	Leicester City
NULL	Norwich City
Queens Park Rangers	NULL
NULL	Watford

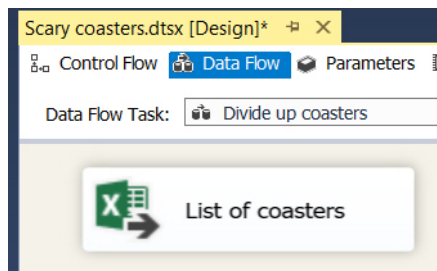
The relegations and promotions between the two seasons.

If you need help, read on to see a flow diagram of what you need to do!

DERIVED COLUMN

14. Assign rollercoasters a scariness rating using a conditional split

Create a new package called **Scary coasters**, and within this a data flow task. This should have an Excel source, which gets a list of rollercoasters from the workbook in the above folder:



So far, so good!

Now add a conditional split which divides coasters up as follows:

Rule	Category
The Type is Wood and the speed is more than 40 mph	Very scary
The coaster is old (opened before 1990) or the Design is not Sit Down	Quite scary
All other coasters failing these two conditions	Not scary

Run your package to check that you get 2 scary coasters, 15 quite scary ones and 33 not scary ones, and a text file like this:

```
Roller Coaster,Amusement Park,Type,Design,Status,Opened,Speed ( mph ),Scar
Air,,Steel,Flying,,2002,46.60000000000001,Medium
Boomerang,,Steel,Sit Down,,1993,47,Low
Cobra,,Steel,Sit Down,,2006,31.10000000000001,Low
Colossus,,Steel,Sit Down,,2002,45,Low
Corkscrew,,Steel,Sit Down,,1980,40,Medium
Corkscrew,,Steel,Sit Down,,1983,40,Medium
Crazy Mouse,,Steel,Sit Down,,1998,29.10000000000001,Low
Crazy Mouse,,Steel,Sit Down,,2000,29.10000000000001,Low
Enigma,,Steel,Sit Down,,1995,34,Low
Express,,Steel,Sit Down,,2006,28,Low
Fantasy Mouse,,Steel,Sit Down,,2000,29.10000000000001,Low
G Force,,Steel,Sit Down,,2005,43.5,Low
Grand National,,Wood,Sit Down,,1935,40,Medium
Infusion,,Steel,Inverted,,2007,49.70000000000003,Medium
Irr-Bru Revolution,,Steel,Sit Down,,1979,45,Medium
Jubilee Odyssey,,Steel,Inverted,,2002,63,Medium
Jungle Coaster,,Steel,Sit Down,,2004,35,Low
Knightmare,,Steel,Sit Down,,2007,43.5,Low
Kumali,,Steel,Inverted,,2006,54.89999999999999,Medium
Magic Mouse,,Steel,Sit Down,,2007,29.10000000000001,Low
Megafobia,,Wood,Sit Down,,1996,48,High
Millennium Roller Coaster,,Steel,Sit Down,,1999,55.89999999999999,Low
Nemesis,,Steel,Inverted,,1994,50,Medium
```

Megafobia is the first coaster in alphabetical order with **Scariness** factor **High**.

When you're happy your package is working successfully (it's hard sometimes to write exercises without *double entendres*), close it down.

AGGREGATE

15. Create aggregate shopping statistics depending on store type

In Management Studio, open and execute the SQL script in the above folder to generate an empty table called **WeirdStats**. Your mission is to fill this from the Excel workbook of shopping purchases to get this:

StoreName	Statistic	NumberPurchases	Amount
Tesco	Normal	134	2.44
Sainsbury	Normal	77	2.60
Co-op	Normal	26	2.50
Asda	Normal	48	1.99
Independent	Normal	145	2.91
Morrison	Normal	52	2.33
Aldi	Smallest	149	0.39
Waitrose	Biggest	63	9.21
Marks and Spencer	Biggest	103	15.63

For posh shops (**Marks and Spencer** and **Waitrose**), show the single most expensive purchase; for cheap shops (**Aldi**) show the single cheapest purchase; otherwise show the average purchase value.

The table should also show which statistic you've chosen to display, as well as the number of purchases made in that store.

If you're feeling confident, try doing this without reading on to see a suggested flow diagram.

When you've got this working, close down the package.

PIVOT AND UNPIVOT

16. Collapse a two-dimensional array into a relational database

Someone has given you the following pivot table to import:

Sum of Spending					
	2014	2015	2016	2017	
Aldi	69.54	114.6	20.92	64.59	
Asda	32.75	22.66	31.95	8.4	
Co-op	17.22	26.29	9.18	12.49	
Independent	281.77		141.3		
Marks and Spencer	117.62	36.36	80.74	10.32	
Morrison	73.51	24.29	16.51	11.62	
Sainsbury	101.15	46.68	37.05	15.69	
Tesco	184	34.91	72.75	36.47	
Waitrose	90.87		67.89		

Your mission is to turn it into this:

Year	Spending	Store
2014	69.54	Aldi
2015	114.6	Aldi
2016	20.92	Aldi
2017	64.59	Aldi
2014	32.75	Asda
2015	22.66	Asda
2016	31.95	Asda
2017	8.4	Asda
2014	17.22	Co-op
2015	26.29	Co-op
2016	9.18	Co-op
2017	12.49	Co-op
2014	281.77	Independent
2016	141.3	Independent
2014	117.62	Marks and Spencer
2015	36.36	Marks and Spencer
2016	80.74	Marks and Spencer

CACHE TRANSFORM

17. Find the differences between two lists of shopping centres

Open the Excel workbook in the above folder, and you should see that it contains two worksheets:

D	E	F
reMetres	NumberUnits	Details
15143	62	1 Pavilion Shopping Centre EN8 7BZ 15,143 62
15143	71	2 Times Square Shopping Centre SM1 1LF 15,143 71
15143	9	3 North Quay Retail Park NR32 2ED 15,143 9
15228	11 4	4 Norman Park CB3 8EL 15,228 11

The **Details** column combines all of the other columns into a single value.

Your task is to list out the five rows in the original list which have been changed in the revised list, and see what they share in common. To do this, first create a package called **Beyond the Pale**.

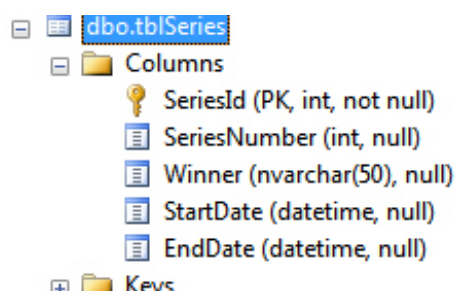
When you run the package, it should give you five centres whose details have been in some way changed. What do they all have in common?

Close the package, reflecting on the fact that its name now makes a bit more sense ...

YOU'RE ON YOUR OWN!

18. Export X Factor Series Data to SQL Server Table

In SQL Server Management Studio, open and run the script called **Create table for series.sql** in the above folder:



You should now have a skeleton table of series, ready to be filled in with data.

Create a new package called **Serious series**, and within this create two control flow tasks:



The first task should execute the SQL statement **TRUNCATE TABLE tblSeries** for the X Factor OLEDB connection, while the second one should be a data flow task which we'll configure in a moment. Ignore the yellow triangle!

Now configure the data flow task so that it:

- Takes data from the Excel workbook in the above folder (you'll need to create an Excel connection manager for this); and
- Exports it to the **tblSeries** table in the **X_Factor** database (using the existing OLEDB connection).

Run your package twice. You should get the following table of data:

SeriesId	SeriesNumber	Winner	StartDate	EndDate
1	1	Steve Brookstein	2004-09-04 00:00:00.000	2004-12-11 00:00:00.000
2	2	Shayne Ward	2005-08-20 00:00:00.000	2005-12-17 00:00:00.000
3	3	Leona Lewis	2006-08-19 00:00:00.000	2006-12-16 00:00:00.000
4	4	Leon Jackson	2007-08-18 00:00:00.000	2007-12-15 00:00:00.000
5	5	Alexandra Burke	2008-08-16 00:00:00.000	2008-12-13 00:00:00.000
6	6	Joe McElderry	2009-08-22 00:00:00.000	2009-12-13 00:00:00.000
7	7	Matt Cardle	2010-08-21 00:00:00.000	2010-12-12 00:00:00.000
8	8	Little Mix (Rhythmic)	2011-08-20 00:00:00.000	2011-12-11 00:00:00.000
9	9	James Arthur	2012-08-18 00:00:00.000	2012-12-09 00:00:00.000

Each time you run the package you get rid of old data in the table before adding new rows.
Close this package down.

19. Import contestants, linking to separate category table

The aim of this exercise is to import the X Factor contestants (again!), storing for each their category number. Here's what the final **tblFinalist** table should look like:

FinalistId	FinalistName	FinishingPosition	SeriesNumber	CategoryId	ImportNote
109	Wally Owl	15	10	5	Birds not found
93	Wagner	6	7	5	Over 28s not found
34	Voices with Soul	6	1	4	Imported without problem
97	Verity Keays	8	1	5	Over 25s not found
27	Union J	4	9	4	Imported without problem
59	Treyc Cohen	10	7	3	Imported without problem
70	The X Factor	12	2	4	Imported without problem

The final table of finalists (shown here in reverse alphabetical order).

Begin by deleting any existing tables called **tblCategory** or **tblFinalist** from your **X_Factor** database, then run the SQL script in the above folder.

CategoryId	CategoryName
1	16-24s
2	Boys
3	Girls
4	Groups
5	Not found

What you should now see in the **tblCategory** table. All contestants with unknown categories will be assigned the value 5.

Create a variable called **UnknownCategoryId** to hold the number 5 above (it's better practice to store this in a variable, so we can change it easily).

Optionally create a task to truncate the table of finalists (so that you can rerun the package several times), then create a data flow task.

20. Show a summary of Brexit voting figures by status

The aim of this exercise is to produce a text file showing the Brexit voting figures, with the largest differences between leave and remain votes coming first:

```
Area,Remain,Leave,Status,Absolute difference
City of Edinburgh,187796,64498,Remain,123298
Northern Ireland,440707,349442,Remain,91265
Glasgow City,168335,84474,Remain,83861
Lambeth,111584,30340,Remain,81244
Wandsworth,118463,39421,Remain,79042
Dudley,56780,118446,Leave,61666
Hackney,83398,22868,Remain,60530
Southwark,94293,35209,Remain,59084
Doncaster,46922,104260,Leave,57338
Wakefield,58877,116165,Leave,57288
Havering,42201,96885,Leave,54684
Brighton and Hove,100648,46027,Remain,54621
Haringey,79991,25855,Remain,54136
```

Dudley (shown highlighted) was the leave area with the highest absolute difference in votes between leave and remain, but it was only 6th in order of absolute difference in votes cast.

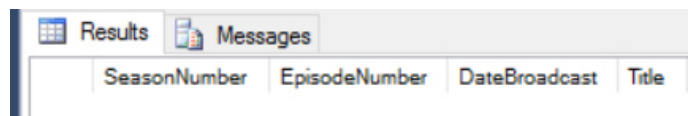
To do this, you'll need to derive the **Status** column shown above, and also calculate the absolute difference and sort by this.

If you're feeling brave, you could continue without turning the page for help!

When everything's working, close down the package and (if a remainder) shed a little tear.

21. Import multiple CSV files of Big Bang data into SQL Server

Run the SQL script in the above folder to generate a table of Big Bang Theory episodes:



SeasonNumber	EpisodeNumber	DateBroadcast	Title
--------------	---------------	---------------	-------

If you've never seen the Big Bang Theory ... do so!

Create a new package called **Big Bang Theories**, and use a **MultiFlatFile** connection to import all four CSV files in the above folder into the SQL Server table you've just generated:

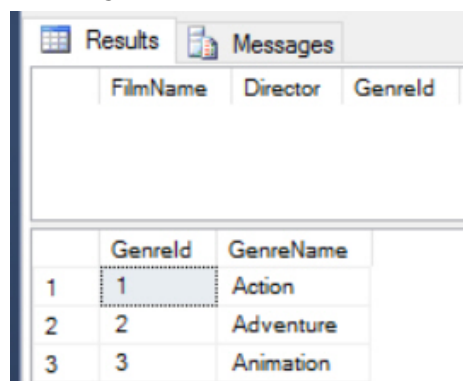
SeasonNumber	EpisodeNumber	DateBroadcast	Title
10	208	2016-09-19	The Conjugal Conjecture
10	209	2016-09-26	The Military Miniaturization
10	210	2016-10-03	The Dependence Transcendence
10	211	2016-10-10	The Cohabitation Experimentation
10	212	2016-10-17	The Hot Tub Contamination

The final table should contain all 91 episodes from the four CSV files (a few of which are shown here).

Check your package runs every time you execute it (you'll need to add an **Execute SQL** task to truncate the table each time), then close it down.

22. Add new genres to a lookup table, then import a list of films

In Management Studio, open the SQL script in the above folder and execute it. This should create two tables:



The screenshot shows the 'Results' pane in SQL Server Enterprise Manager. It displays a table with three columns: 'GenreId', 'GenreName', and 'GenreDescription'. The first three rows are visible, showing 'Action', 'Adventure', and 'Animation' genres.

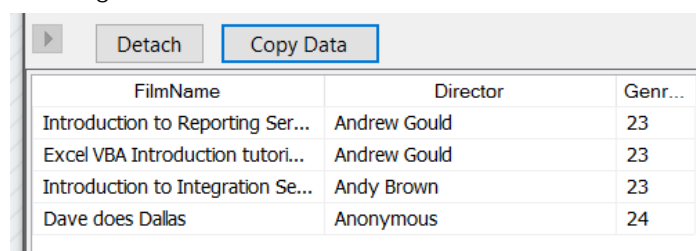
GenreId	GenreName	GenreDescription
1	Action	
2	Adventure	
3	Animation	

The aim of the exercise is to import a list of films into the **FilmsForLookupExercise** table, setting the genre id for each (and adding any new genres which don't exist).

Create a package whose control flow should contain four tasks:

Part	Type	What it should do
1	Execute SQL	Delete any existing rows in the FilmsForLookupExercise table
2	Data flow	Import the films from the workbooks in the above folder, matching the genre to the Genre table in SQL Server and adding any new genres found to this table. The resulting of running this part should be to add two new genres to the table, which will automatically be given the numbers 23 and 24 (the next available numbers)
3	Data flow	Import the films again, matching the genre to the Genre table (which we now know will work), and returning the genre id. Send the results of this to the FilmsForLookupExercise table
4	Data flow	Import the FilmsForLookupExercise table rows, and use a conditional split to show only those where the genre id is more than 22 (ie films for genres just added). Use a data viewer to show these rows.

As well as adding rows to the **Genre** and **FilmsForLookupExercise** tables, the package should produce the following data viewer:



The screenshot shows a data viewer window with a 'Copy Data' button highlighted. The data is presented in a table with three columns: 'FilmName', 'Director', and 'GenreId'. The first four rows are visible, showing films like 'Introduction to Reporting Ser...', 'Excel VBA Introduction tutori...', 'Introduction to Integration Se...', and 'Dave does Dallas'.

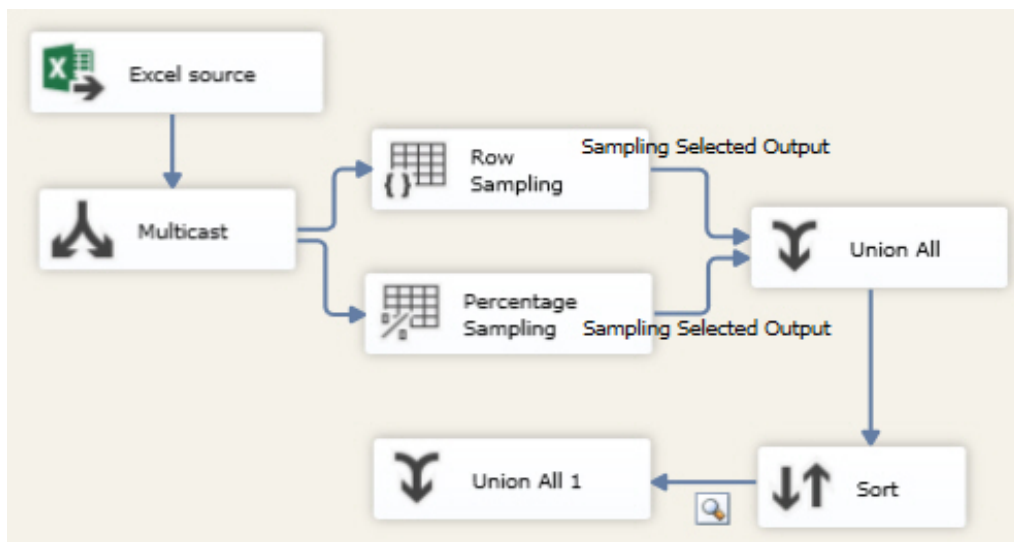
FilmName	Director	GenreId
Introduction to Reporting Ser...	Andrew Gould	23
Excel VBA Introduction tutori...	Andrew Gould	23
Introduction to Integration Se...	Andy Brown	23
Dave does Dallas	Anonymous	24

There are 3 films in the **Software** genre, and one in the **Dark side** genre.

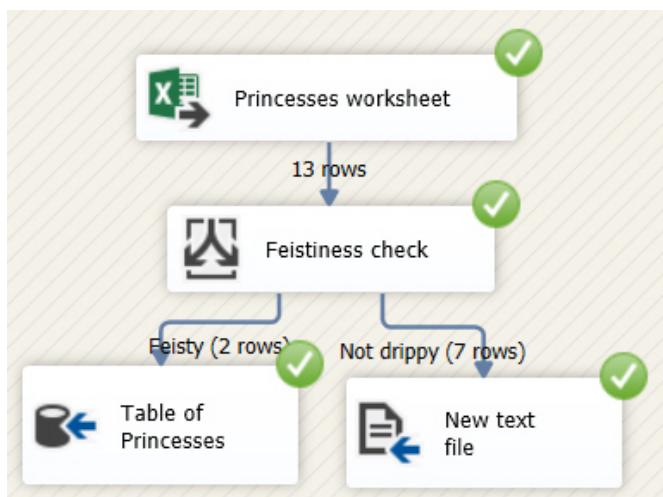
When you've got this package working, close it down!

HELP!

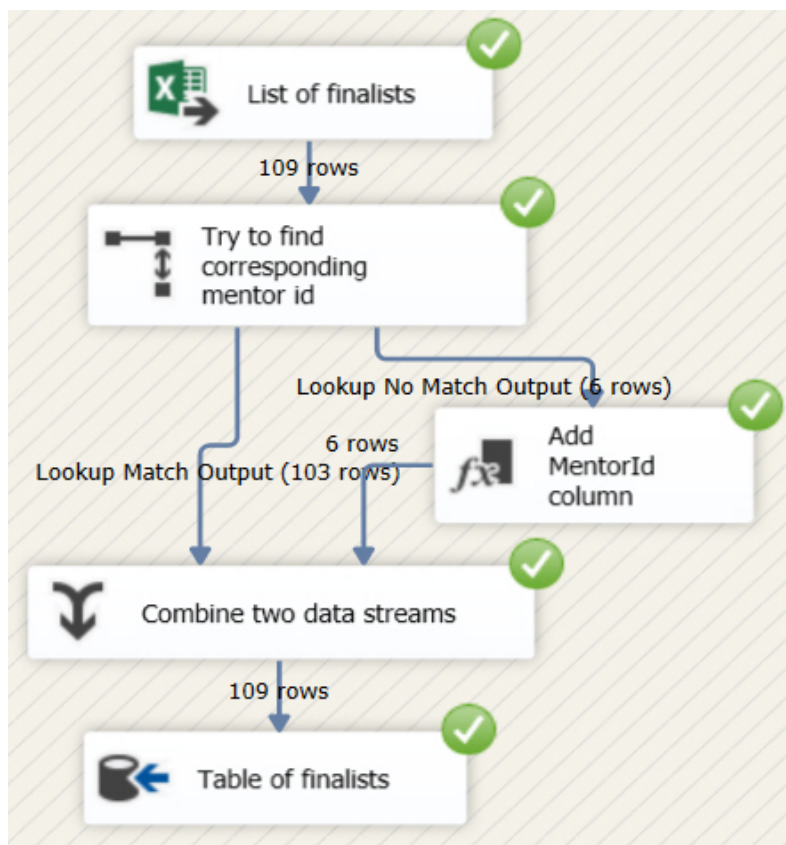
1. Exercise 8: Create a random playlist from an Excel workbook of songs



2. Exercise 9: Divide the Disney princesses into feminist role models/others



3. Exercise 12: Find the id number for each X Factor mentor



4. Ex. 13: Find the differences between the 2015 and 2016 premier leagues

Here's what your data flow could look like:

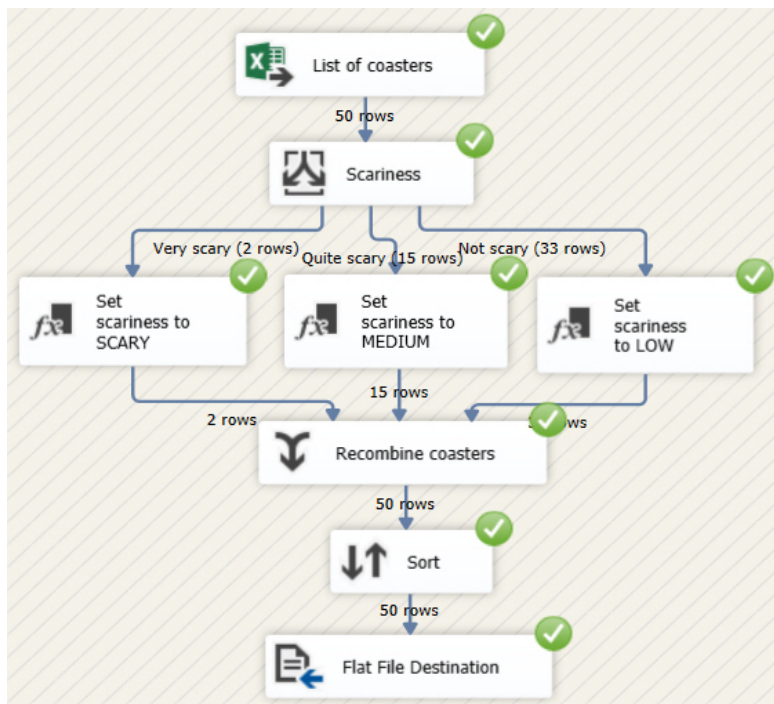


A suggestion for how to solve this problem.

5. Ex. 14: Assign rollercoasters a scariness rating using a conditional split

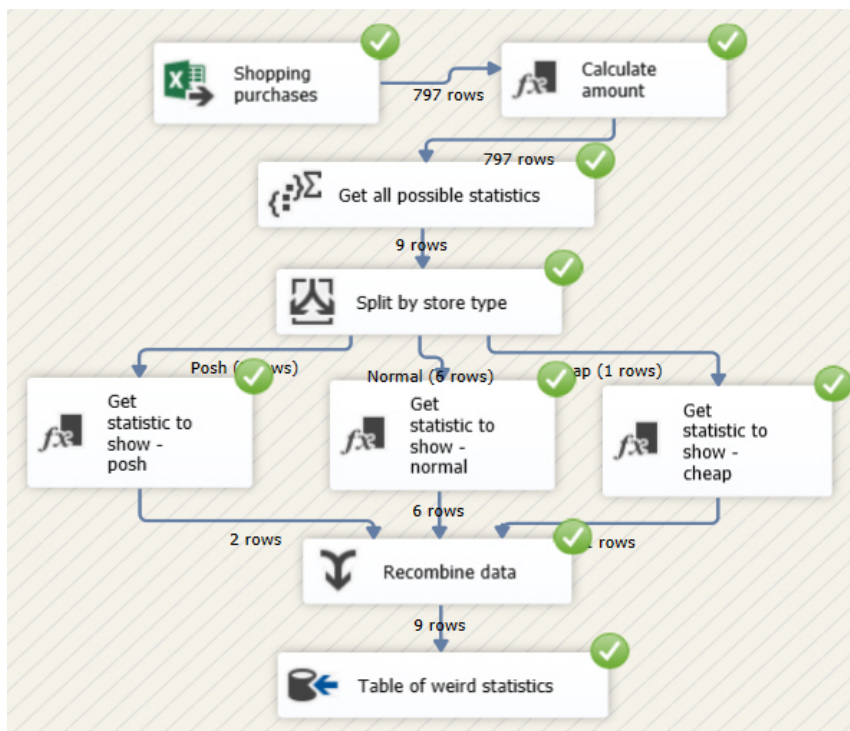
Add three **Derived Column** transforms, each of which sets a new column called **Scariness** to a different value

Also add a **Union All** transform to reunite the rollercoasters at the bottom, a **Sort** transform to sort them by the rollercoaster name and a flat file destination to send the results to a new flat file.



6. Ex. 15: Create aggregate shopping statistics depending on store type

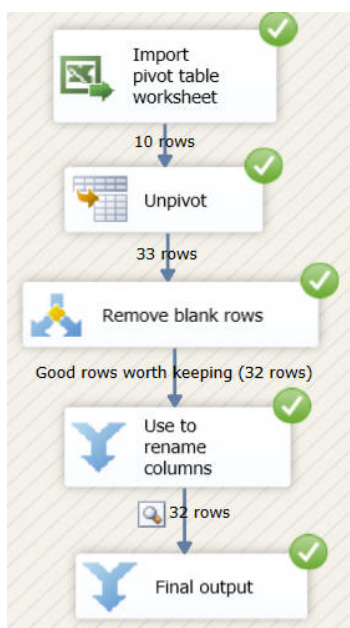
Here's one possible way to solve this:



One possible answer (but not necessarily the best one!).

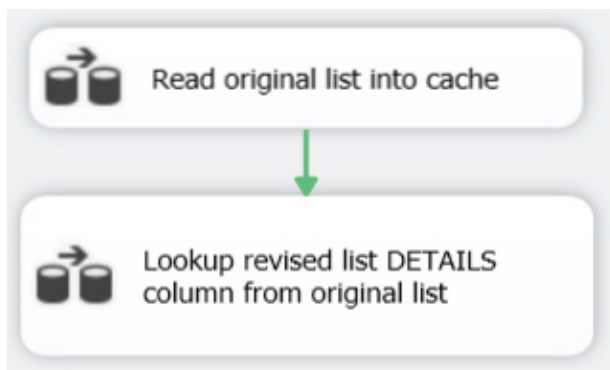
7. Ex. 16: Collapse a two-dimensional array into a relational database

The final diagram looks like this:

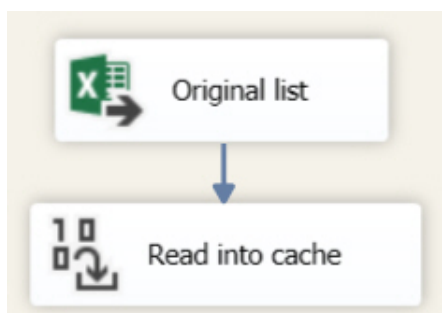


8. Ex. 17: Find the differences between two lists of shopping centres

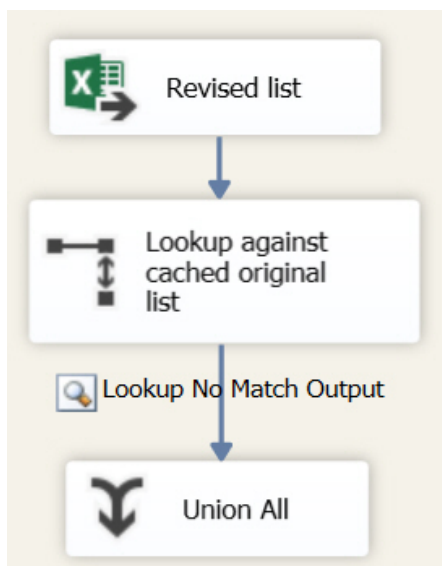
Here's what this package's control flow should do:



Here's what the two data flow tasks could look like:

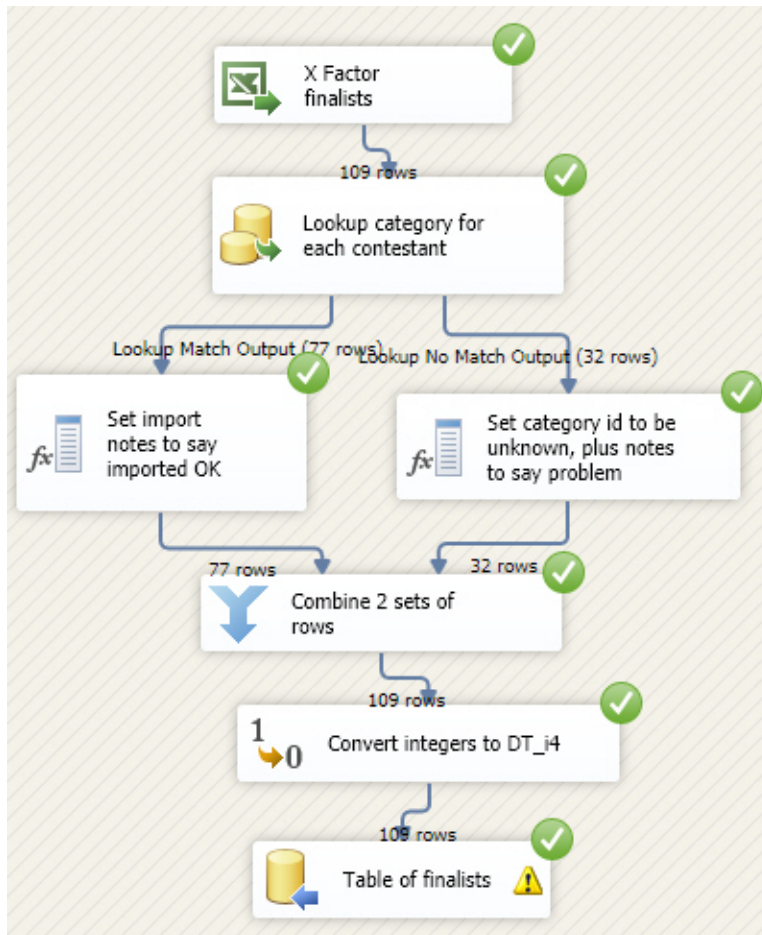


First data flow task



Second data flow task

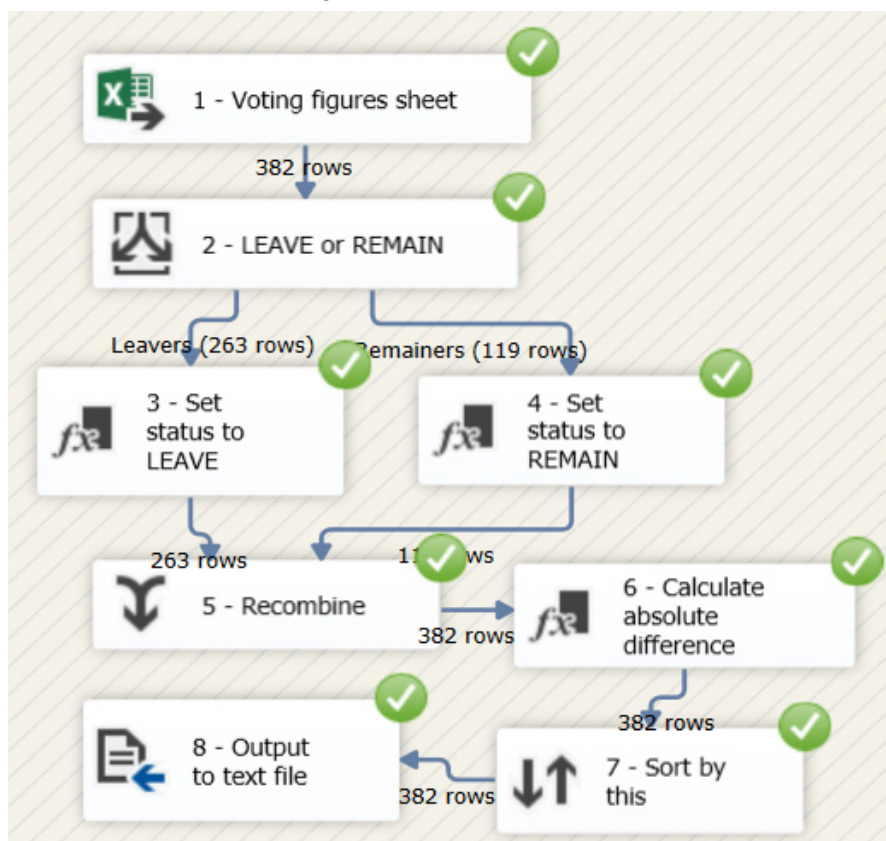
9. Ex. 19: Import contestants, linking to separate category table



As this shows, there are 32 contestants with no corresponding category name.

10. Ex. 20: Show a summary of Brexit voting figures by status

Here's what the final package could look like:



A suggestion for how to solve this problem - a guide to each step follows.

Here's what each step could do:

Step	What it could do
1	Import the Excel workbook contained in the folder at the top of this exercise.
2	A conditional split to determine if the leave votes exceeded the remain ones.
3	A derived column transform to create a new field called Status, and set it to Leave.
4	A derived column transform to create a new field called Status, and set it to Remain.
5	A Union All transform to unite leavers and remainers (if only it were that simple).
6	Use the ABS function in a derived column transform to work out the difference between leave and remain votes, ignoring any sign.
7	Sort by this newly calculated column.
8	Send the output to a new flat file.