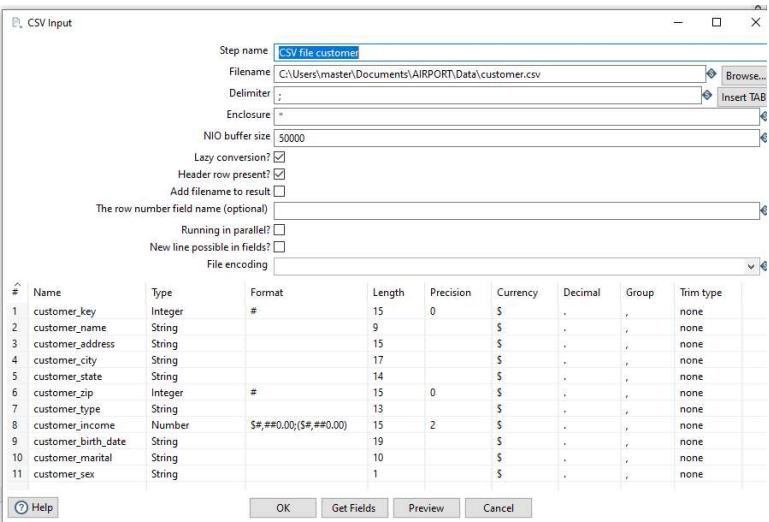


Tarea 4: A complete set of data transformationsAssignment

Una vez hecha la tarea 3, se procede a realizar la carga de datos de las demás dimensiones. Se mostrarán los pasos de una de ellas, ya que las restantes se hacen de la misma forma.

Carga de datos de la dimensión CUSTOMER:

1. Se utiliza el *CSV Input* para cargar los datos en el flujo

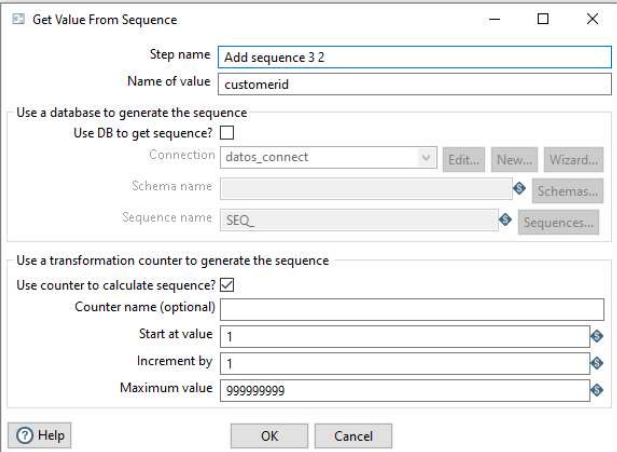


The screenshot shows the 'CSV Input' dialog box with the following details:

- Step name: CSV file customer
- Filename: C:\Users\master\Documents\AIRPORT\Data\customer.csv
- Delimiter: ;
- Enclosure: *
- NIO buffer size: 50000
- Lazy conversion?: ☒
- Header row present?: ☒
- Add filename to result: ☐
- The row number field name (optional):
- Running in parallel?: ☐
- New line possible in fields?: ☐
- File encoding:

#	Name	Type	Format	Length	Precision	Currency	Decimal	Group	Trim type
1	customer_key	Integer		15	0	\$.		none
2	customer_name	String	#	9		\$.		none
3	customer_address	String		15		\$.		none
4	customer_city	String		17		\$.		none
5	customer_state	String		14		\$.		none
6	customer_zip	Integer	#	15	0	\$.		none
7	customer_type	String		13		\$.		none
8	customer_income	Number	\$.##0.00;(\$\$.##0.00)	15	2	\$.		none
9	customer_birth_date	String		19		\$.		none
10	customer_marital	String		10		\$.		none
11	customer_sex	String		1		\$.		none

2. Se crea el ID de esta dimensión con la herramienta *ADD Sequence*



The screenshot shows the 'Get Value From Sequence' dialog box with the following details:

- Step name: Add sequence 3 2
- Name of value: customerid
- Use a database to generate the sequence: ☐
- Connection: datos_connect
- Schema name:
- Sequence name: SEQ_
- Use a transformation counter to generate the sequence: ☒
- Counter name (optional):
- Start at value: 1
- Increment by: 1
- Maximum value: 999999999

3. Por último, se exporta a la tabla de *MYSQL*

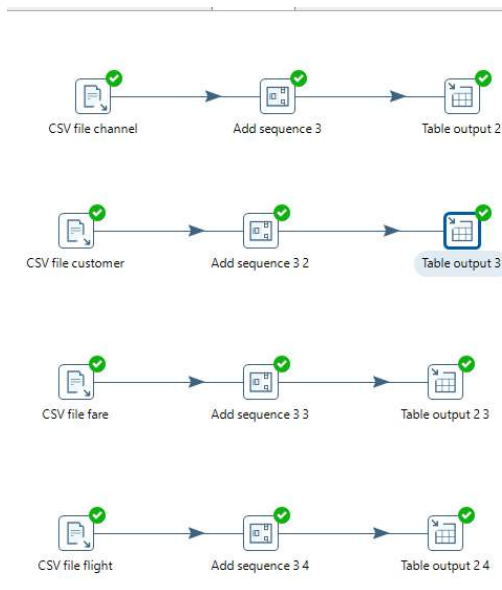


Table output

Step name

Connection

Target schema

Target table

Commit size

Truncate table ☒

Ignore insert errors ☐

Specify database fields ☒

Main options **Database fields**

Partition data over tables ☐

Partitioning field

Partition data per month ☒

Partition data per day ☐

Use batch update for inserts ☒

Is the name of the table defined in a field? ☐

Field that contains name of table:

Store the tablename field ☒

Return auto-generated key ☐

Name of auto-generated key field

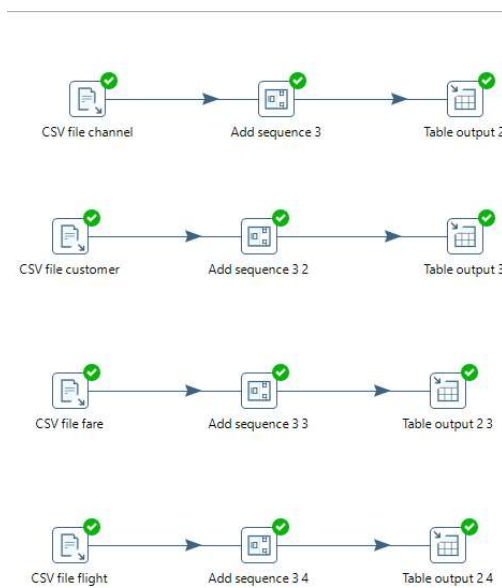


Table output

Step name: Table output 3

Connection: datos_connect

Target schema:

Target table: customer

Commit size: 1000

Truncate table: ☒

Ignore insert errors: ☐

Specify database fields: ☒

Main options: Database fields

Fields to insert:

#	Table field	Stream field
1	customer_key	customer_key
2	customer_name	customer_name
3	customer_address	customer_address
4	customer_city	customer_city
5	customer_state	customer_state
6	customer_zip	customer_zip
7	customer_type	customer_type
8	customer_income	customer_income
9	customer_birth_date	customer_birth_date
10	customer_marital	customer_marital
11	customer_sex	customer_sex
12	customerID	customerid

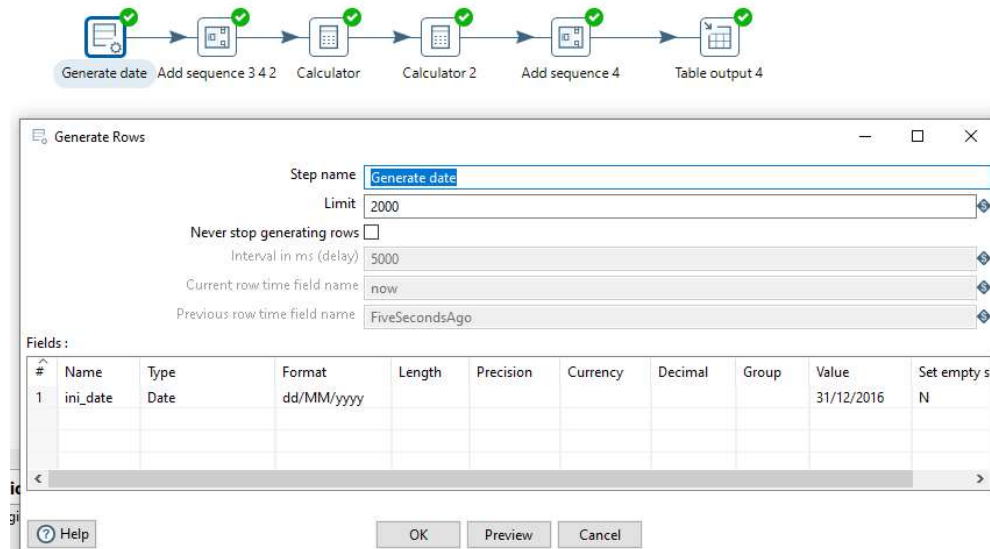
Get fields

Enter field mapping

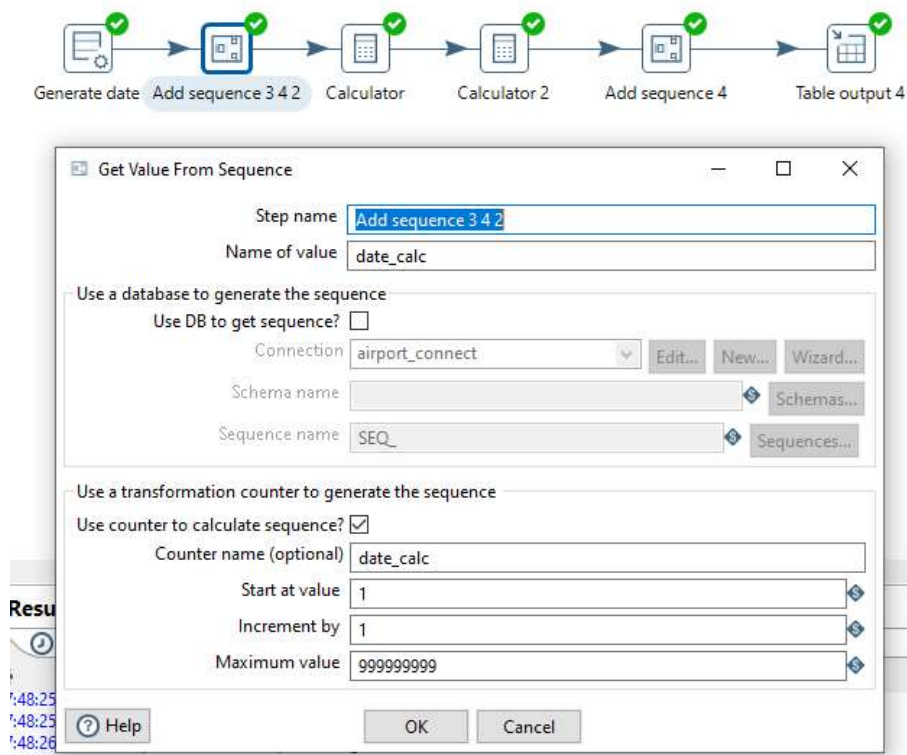
Tabla DATE

Una vez realizado esto con las dimensiones *channel*, *fare* y *flight* se procede a crear las fechas en la que se producirán los vuelos:

1. Se utiliza el generador de fechas (*Generate Date*)



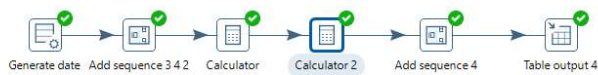
2. Se crea una variable que se irá incrementando a 1 por cada vez que se utilice, de esta forma se podrá calcular los días siguientes a la fecha de inicio



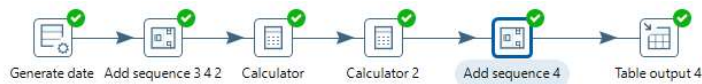
3. En este paso se calcula y se utiliza el siguiente día, hasta un total de 1000 días después del día de inicio

[illegible]

4. En el calculator 2, se crean las variables day, month y year. De tal forma que se podrían realizar búsquedas generalizadas en vez de realizar búsquedas específicas de una fecha en específica.

[illegible]

5. Se genera la ID de las fechas

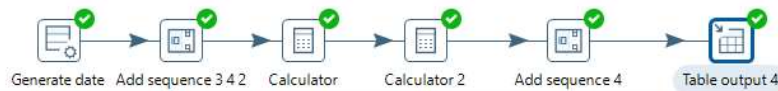


Ventana de configuración 'Get Value From Sequence':

- Step name: **Add sequence 4**
- Name of value: **dateid**
- Use a database to generate the sequence:
 - Use DB to get sequence? ☐
 - Connection: **airport_connect** (Botones: Edit..., New..., Wizard...)
 - Schema name: [Campo vacío] (Botón: Schemas...)
 - Sequence name: **SEQ_** (Botón: Sequences...)
- Use a transformation counter to generate the sequence:
 - Use counter to calculate sequence? ☒
 - Counter name (optional): [Campo vacío]
 - Start at value: **1**
 - Increment by: **1**
 - Maximum value: **999999999**

Botones: Help, OK, Cancel.

6. Se exporta el CSV modificado a la tabla de MYSQL



Ventana de configuración 'Table output':

- Step name: **Table output 4**
- Connection: **airport_connect** (Botones: Edit..., New..., Wizard...)
- Target schema: [Campo vacío] (Botón: Browse...)
- Target table: **date** (Botón: Browse...)
- Commit size: **1000**
- Truncate table: ☒
- Ignore insert errors: ☐
- Specify database fields: ☒

pestañas: Main options | Database fields

- Partition data over tables: ☐
 - Partitioning field: [Campo vacío]
 - Partition data per month: ☒
 - Partition data per day: ☐
- Use batch update for inserts: ☒
- Is the name of the table defined in a field? ☐
 - Field that contains name of table: [Campo vacío]
 - Store the tablename field: ☒
- Return auto-generated key: ☐
 - Name of auto-generated key field: [Campo vacío]

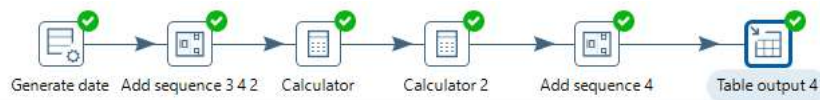


Table output

Step name:

Connection:

Target schema:

Target table:

Commit size:

Truncate table: ☒

Ignore insert errors: ☐

Specify database fields: ☒

Main options | Database fields

Fields to insert:

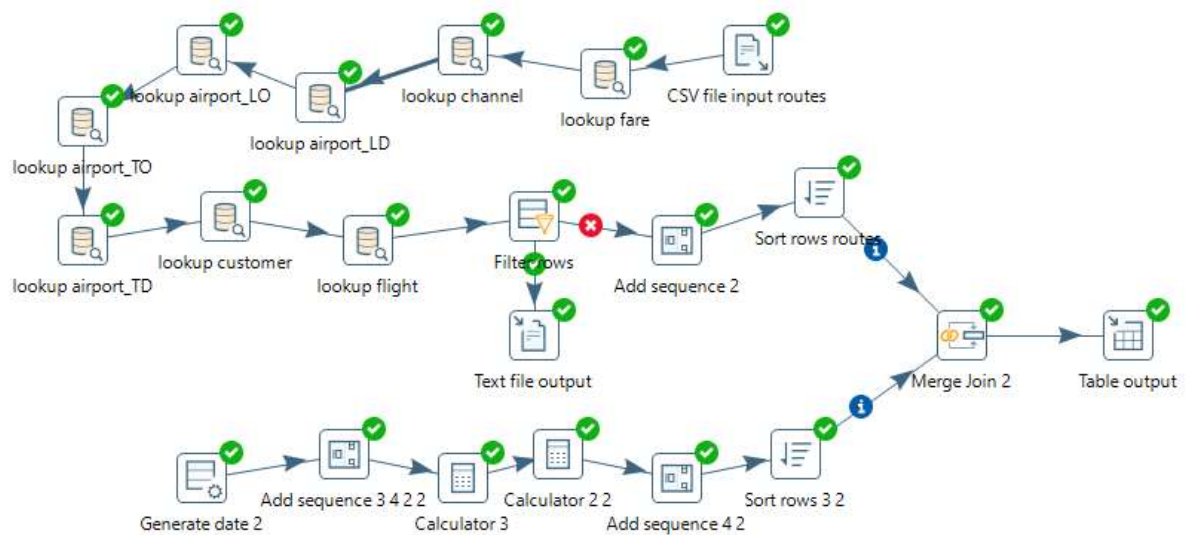
#	Table field	Stream field	
1	day	day_date	
2	month	month_date	
3	year	year_date	
4	dateid	dateid	

Tabla FACT

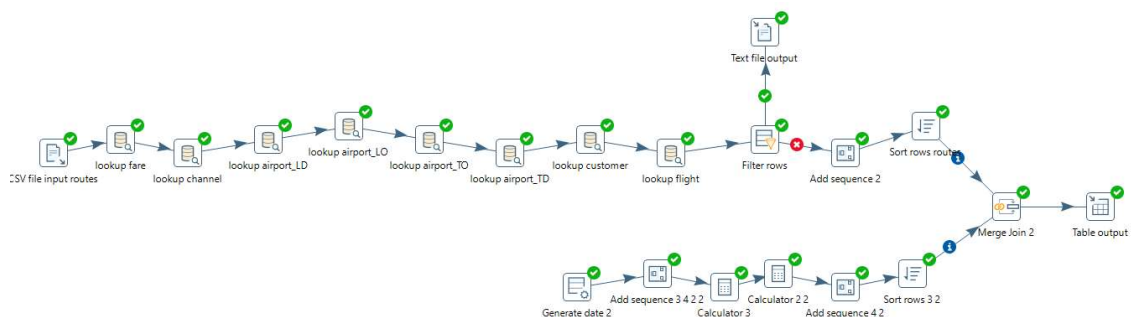
En este paso se procede a la creación de la tabla de hechos haciendo uso de las herramientas:

- Lookup
- Filter
- Merge
- Add sequence
- Input y Output

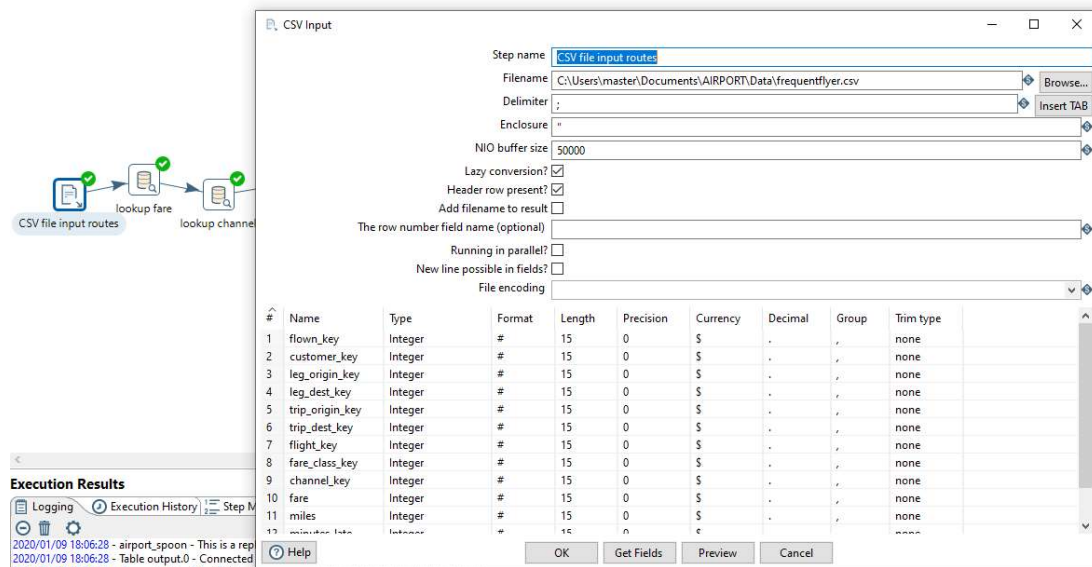
En la siguiente imagen se mostrará el resultado de nuestra tabla de hechos, la cual se irá comentando uno a uno los elementos de la que se compone.



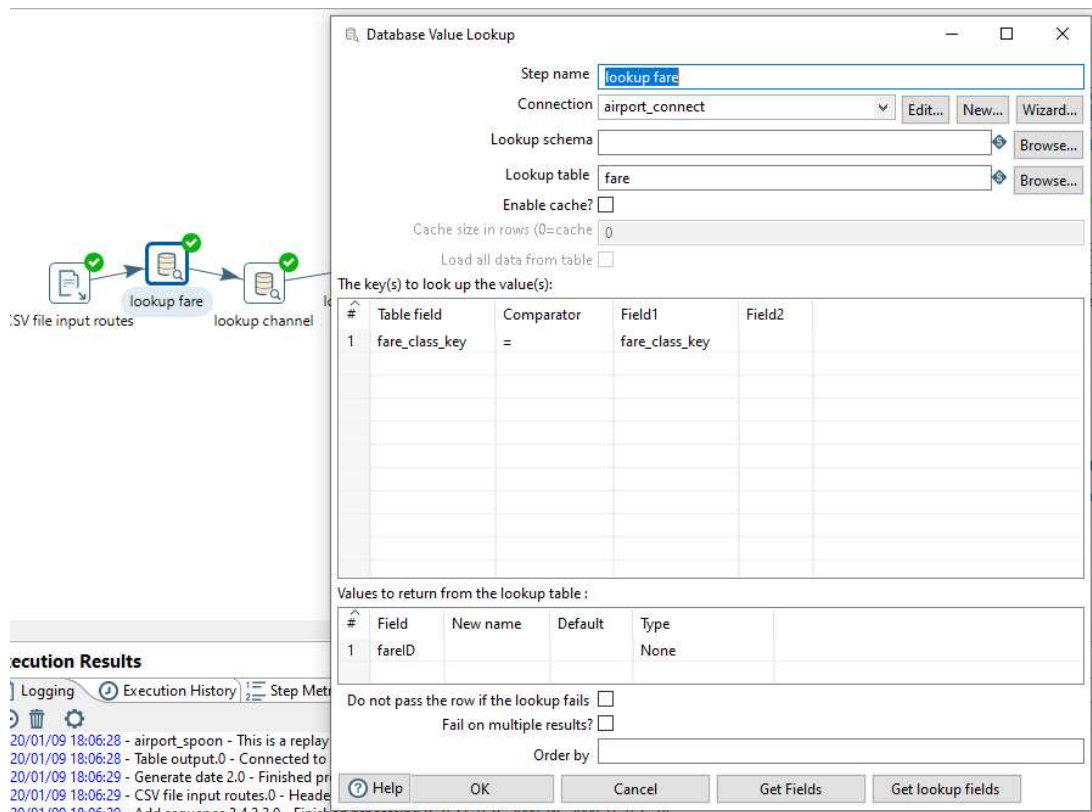
Para explicar los elementos de la tabla *FACT* utilizaremos el siguiente orden, ya que la imagen de arriba solo está para que se pueda ampliar más la imagen.



1. CSV Input para crear el flujo de datos



2. Se crea un lookup por cada dimensión de la que se ha hecho uso



Lookup de customer:

Step name: **lookup customer**

Connection: **airport_connect**

Lookup schema: **customer**

Lookup table: **customer**

Cache size in rows (0=cache): **0**

Load all data from table: ☐

The key(s) to look up the value(s):

#	Table field	Comparator	Field1	Field2
1	customer_key	=	customer_key	

Values to return from the lookup table:

#	Field	New name	Default	Type
1	customerID			None

Do not pass the row if the lookup fails: ☐

Fail on multiple results?: ☐

Order by:

Buttons: **Help** **OK** **Cancel** **Get Fields** **Get lookup fields**

Para la dimensión airport, como se ha hecho uso 4 veces (Leg_Orig, Trip_Orig, Leg_Dest, Trip_Dest) se crea un lookup para cada uno de ellos (si se crease los 4 en un mismo lookup da error).

Step name: **lookup airport_LO**

Connection: **airport_connect**

Lookup schema: **airport**

Lookup table: **airport**

Cache size in rows (0=cache): **0**

Load all data from table: ☐

The key(s) to look up the value(s):

#	Table field	Comparator	Field1	Field2
1	leg_key	=	leg_origin_key	

Values to return from the lookup table:

#	Field	New name	Default	Type
1	airportID	airportID_LO		None

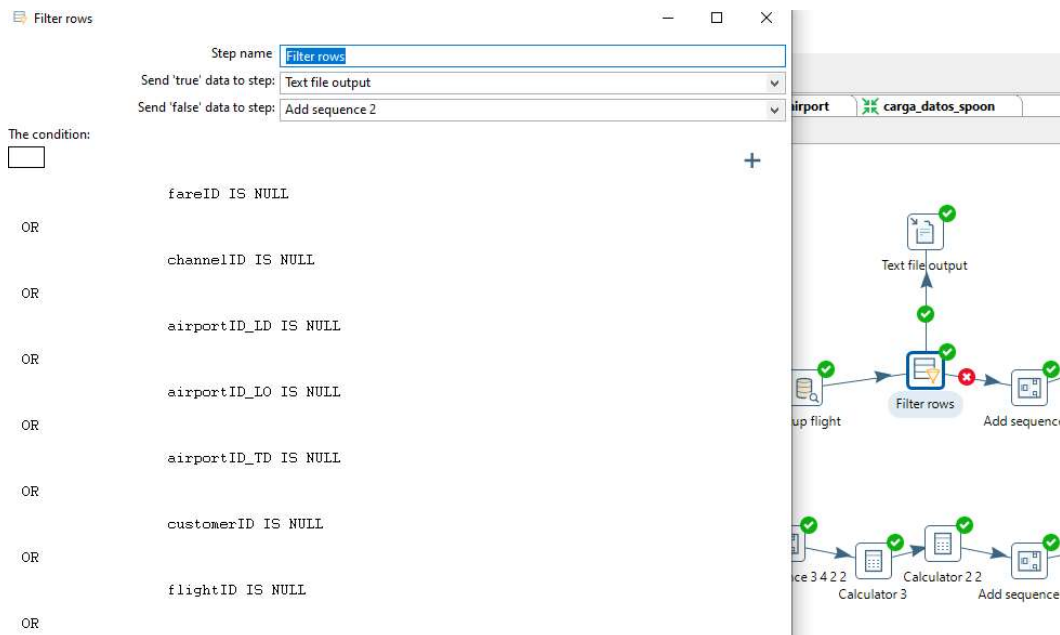
Do not pass the row if the lookup fails: ☐

Fail on multiple results?: ☐

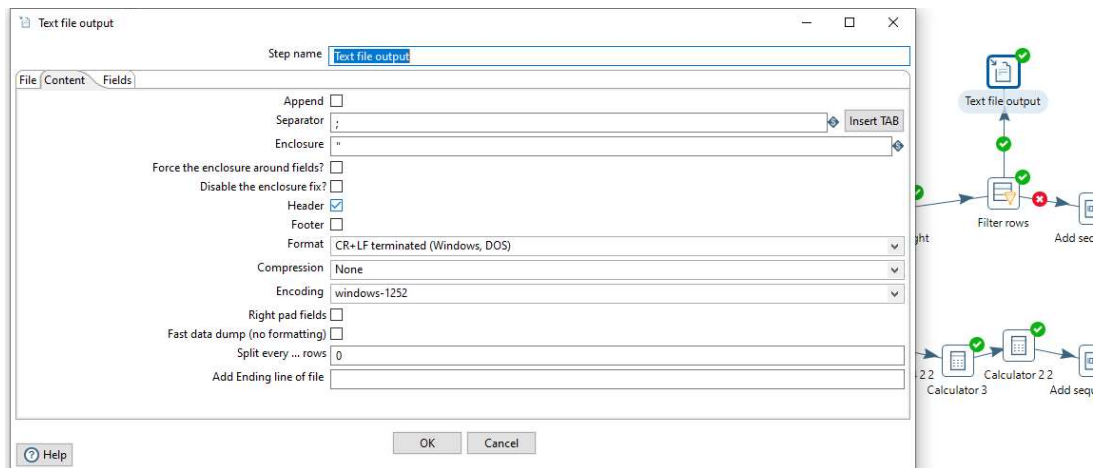
Order by:

Buttons: **Help** **OK** **Cancel** **Get Fields** **Get lookup fields**

3. Una vez hecho todos los *lookups* se debe de filtrar por la clave *ID*, ya que algunas tuplas tienen *nulls* en estas *ID*.

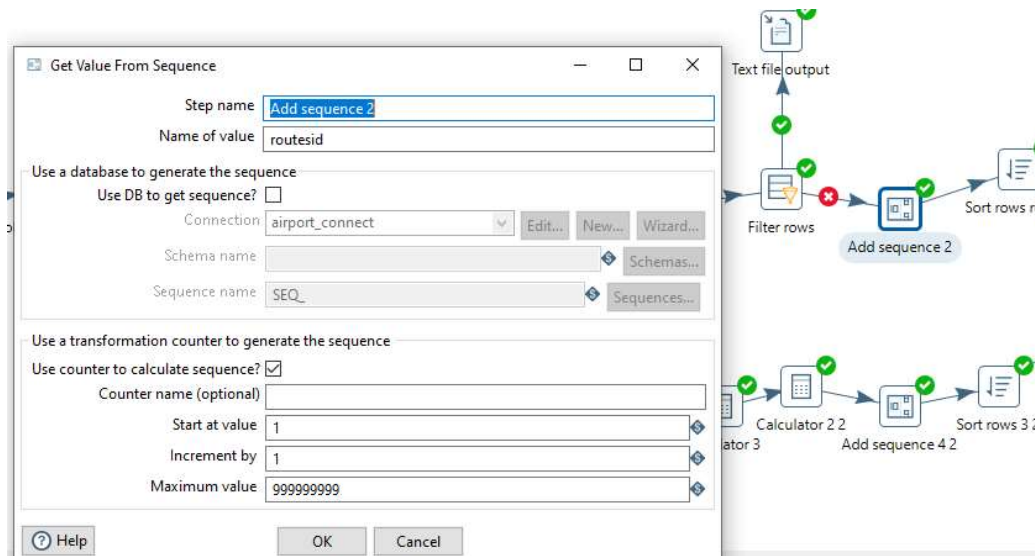


4. Si alguna *ID* resulta ser *null* la condición es *TRUE* y no interesaría esa tupla. Se manda a un archivo de texto.

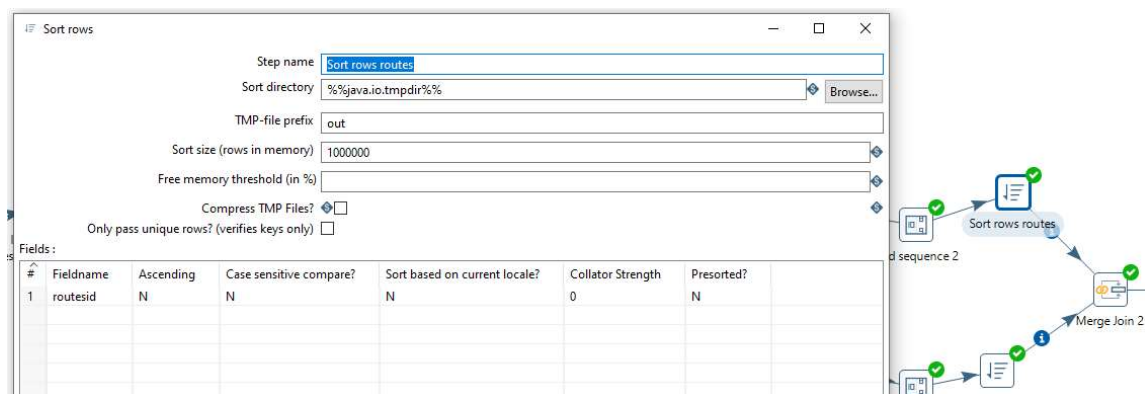


En cambio, si ninguno de los *ID* es *null* la condición es *false* y la tupla si nos interesa.

- Se genera un ID por cada tupla que ha conseguido pasar el filtro

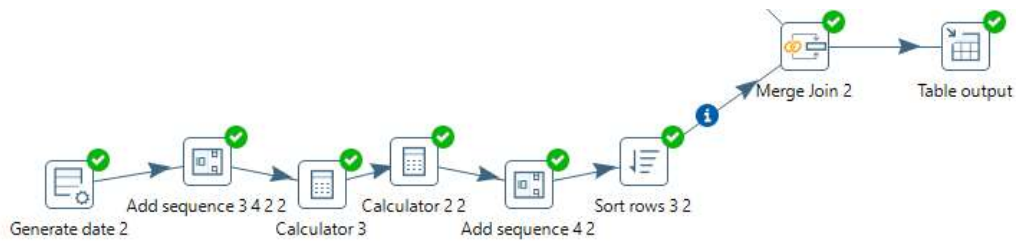


- Se ordenan estas tuplas para posteriormente hacer uso de la herramienta *merge* (que exige este orden)



Antes de mostrar el elemento *merge*, se mostrarán los elementos que harán uso de este *merge*. En este caso la creación de *DATE* para *routes*.

7. Los pasos y elementos son los mismos que en la creación de la tabla DATE hasta llegar al *sort row*



Sort Row:

Step name: Sort rows 3 2

Sort directory: %%java.io.tmpdir%%

TMP-file prefix: out

Sort size (rows in memory): 1000000

Free memory threshold (in %):

Compress TMP Files? ☐

Only pass unique rows? (verifies keys only) ☐

#	Fieldname	Ascending	Case sensitive compare?	Sort based on current locale?	Collator Strength	Presorted?
1	dateid	N	N	N	0	N

8. Se unen ambos flujos de datos utilizando la herramienta *MERGE*

```
graph LR; A[Customer] --> B[lookup flight]; B --> C[Filter rows]; C --> D[Add sequence 2]; D --> E[Sort rows routes]; E --> F[Sort rows 3 2]; F --> G[Merge Join 2]; G --> H[Table output];
```

Step name: Merge Join 2

First Step: Sort rows routes

Second Step: Sort rows 3 2

Join Type: INNER

#	Key field
1	routesid

#	Key field
1	dateid

Get key fields

Get key fields

Help OK Cancel

9. Por último, se exportan los datos a nuestra tabla *routes* en *MYSQL*

Table output

Step name:

Connection: Edit... New... Wizard...

Target schema: Browse...

Target table: Browse...

Commit size:

Truncate table: ☒

Ignore insert errors: ☐

Specify database fields: ☒

Main options Database fields

Partition data over tables: ☐

Partitioning field:

Partition data per month: ☒

Partition data per day: ☐

Use batch update for inserts: ☒

Is the name of the table defined in a field?: ☐

Field that contains name of table:

Store the tablename field: ☒

Return auto-generated key: ☐

Name of auto-generated key field:

Merge Join 2

Table output

Table output

Step name:

Connection: Edit... New... Wizard...

Target schema: Browse...

Target table: Browse...

Commit size:

Truncate table: ☒

Ignore insert errors: ☐

Specify database fields: ☒

Main options Database fields

Fields to insert:

#	Table field	Stream field
1	flown_key	flown_key
2	customer	customerID
3	flight	flightID
4	date	dateid
5	aeropuerto...	airportID_TO
6	aeropuerto...	airportID_TD
7	aeropuerto...	airportID_LD
8	aeropuerto...	airportID_LO
9	channel	channelID
10	ticket_num...	ticket_number
11	miles	miles
12	minutes_late	minutes_late
13	fare	fare
14	fare_class...	fareID

Get fields

Enter field mapping

Merge Join 2

Table output