

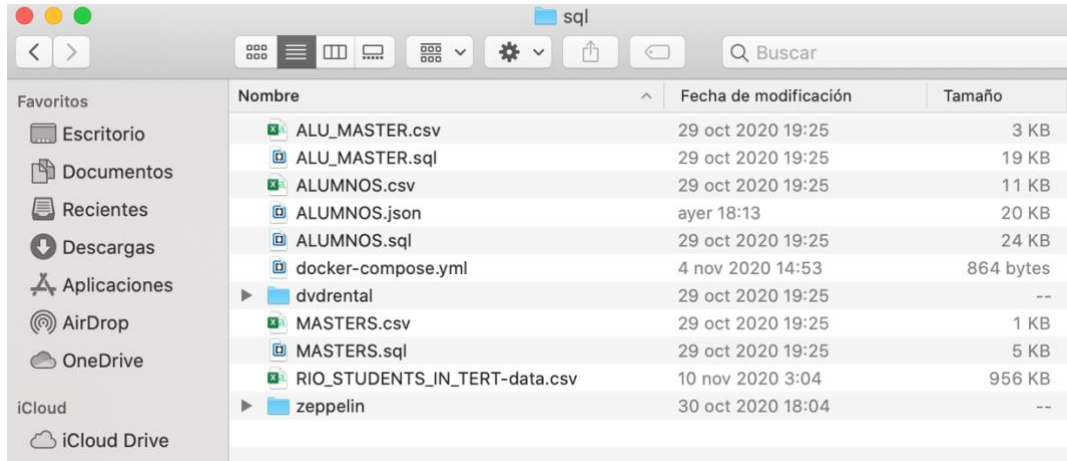
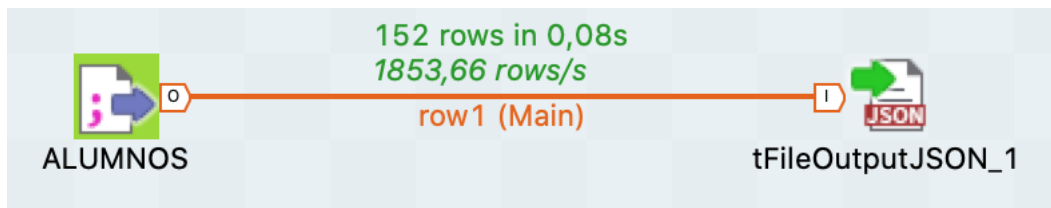


# EJERCICIOS ETL

Máster Data Analytics para la Empresa- EDEM

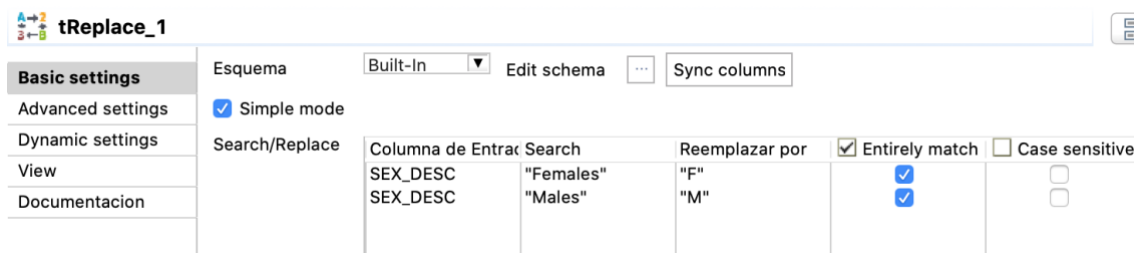
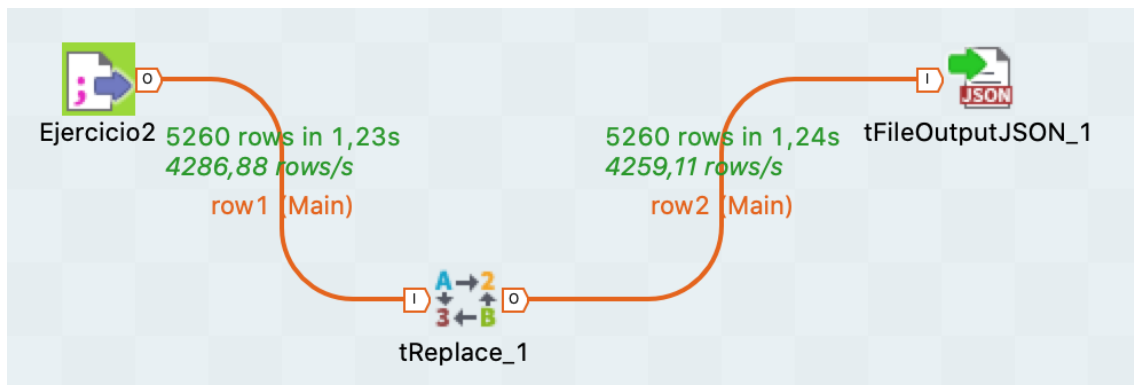
**Carlos Donoso Cabero**

**Ejercicio 1:** Leer un fichero CSV y escribirlo a fichero JSON en la misma carpeta:

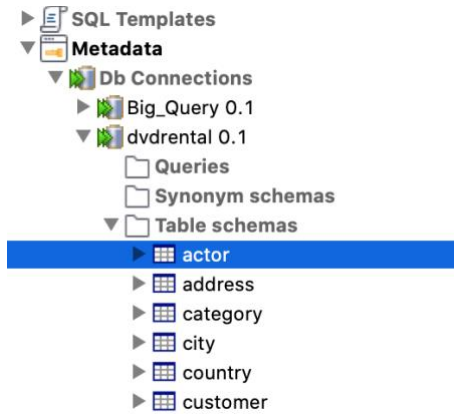
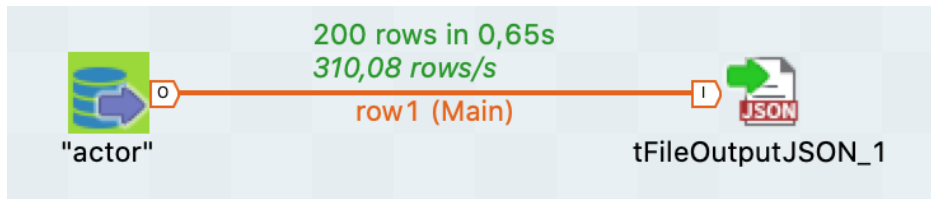


**Ejercicio 2:** Leer un fichero CSV (nuevo CSV descargado *ad hoc*) y reemplazar:

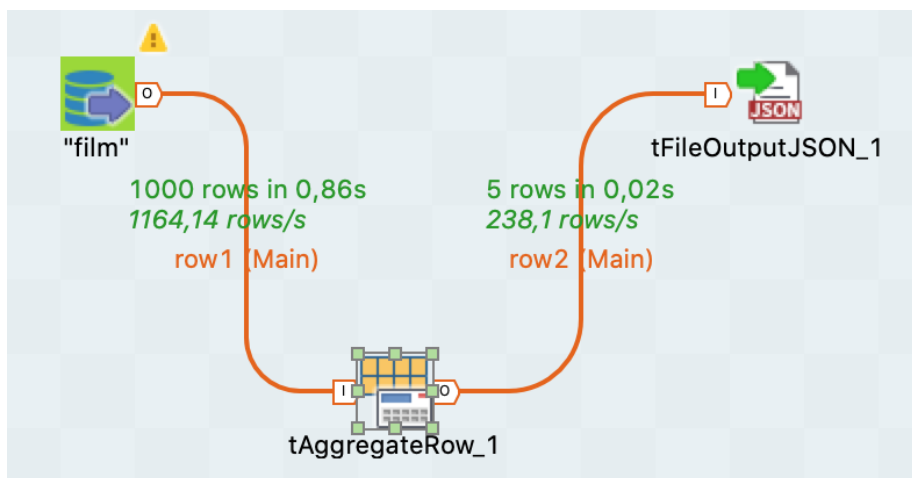
- MALES -> M
- FEMALES -> F



**Ejercicio 3:** Leer tabla de actores (db) y volcarlo a fichero JSON:



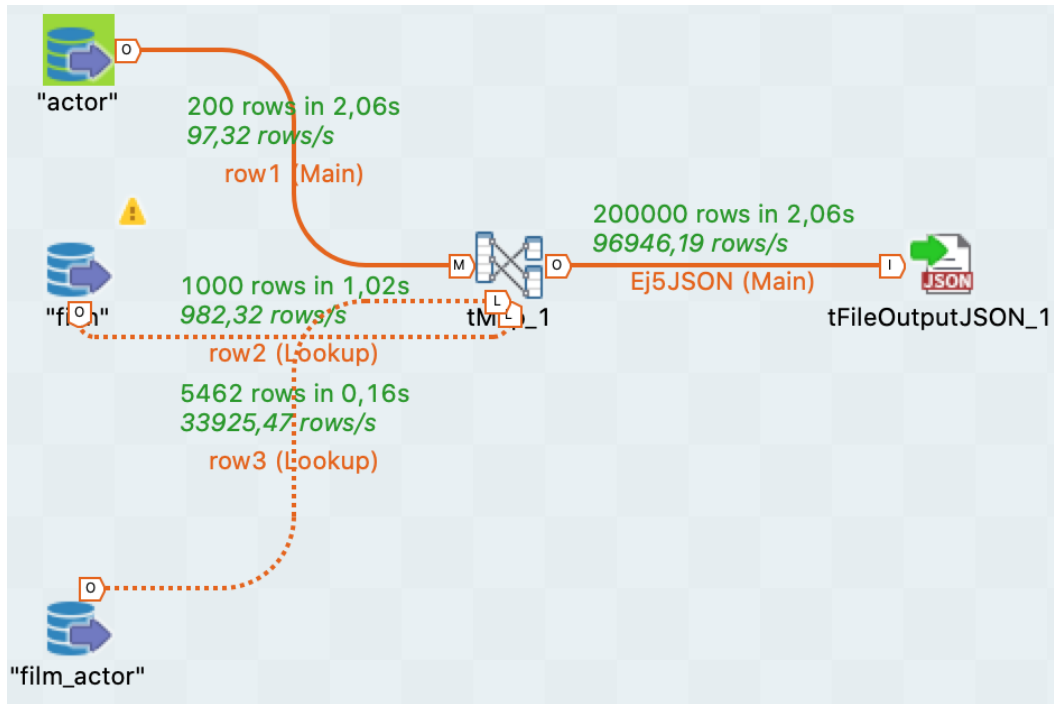
**Ejercicio 4:** Agregar las películas por 'rating' y mostrar un count, volcar a JSON el resultado:



The screenshot shows the configuration window for 'tAggregateRow\_1'. It has a sidebar with 'Basic settings', 'Advanced settings', 'Dynamic settings', 'View', and 'Documentacion'. The main area is divided into two sections: 'Agrupar por' (Group by) and 'Operaciones' (Operations).  
 In the 'Agrupar por' section, the 'Esquema' (Schema) is set to 'Built-In'. The 'Columna de Salida' (Output Column) is 'rating' and the 'Input column position' is 'rating'.  
 In the 'Operaciones' section, the 'Columna de Salida' (Output Column) is 'count', the 'Función' (Function) is 'count', and the 'Input column position' is 'film\_id'. There is an 'Ignore null va' checkbox which is unchecked.

**Ejercicio 5:** Realizar un JOIN entre Actor / Film / Film\_Actor y volcar a JSON un fichero con estos campos:

- Nombre
- Apellido
- Película



Editor de esquemas y Editor de expresiones para el archivo Ej5JSON.

**row1**

Column
actor_id
first_name
last_name
last_update

**row2**

Expr. key	Column
film_id	title
	description
	release_year
	language_id
	rental_duration
	rental_rate
	length
	replacement_cost
	rating
	last_update
	special_features
	fulltext

**row3**

Property	Value
actor_id	Int
film_id	Int
last_update	Date

**Ej5JSON**

Expresión	Column
row1.first_name	first_name
row1.last_name	last_name
row2.title	title

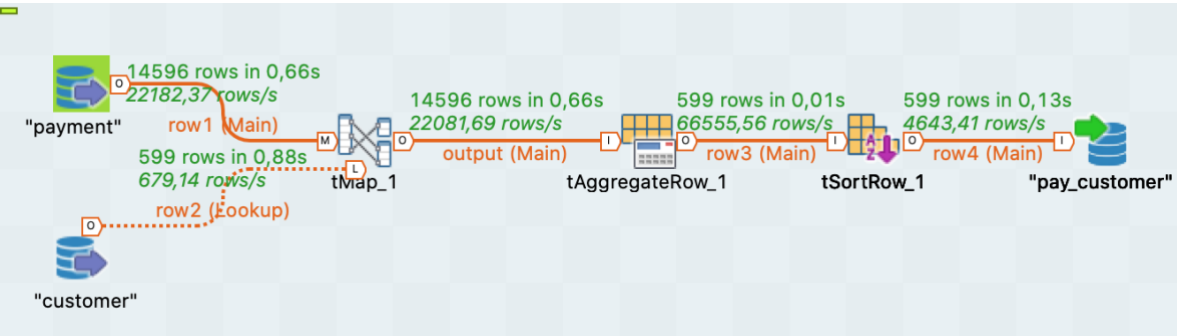
**Editor de esquemas**

Columna	Clave	Tipo	Nullab	Date Pattern	Ctr Longitud	Precision	Default	Comentari
actor_id	Int	Int			5	0		
film_id	Int	Int			5	0		
last_update	Date	Date		"dd-MM-yyyy"	29	6	'now()'	

**Ej5JSON**

Columna	Clave	Tipo	Nullab	Date Pattern	Ctr Longitud	Precision	Default	Comentari
first_name	String	String			45	0		
last_name	String	String			45	0		
title	String	String			255	0		

**Ejercicio 6:** Cargar una tabla con la cantidad de dinero gastada por usuario, nombre y apellido:



dvdrental/postgres@server1

Query Editor

Query History

1 `select * from pay_customer`

Data Output

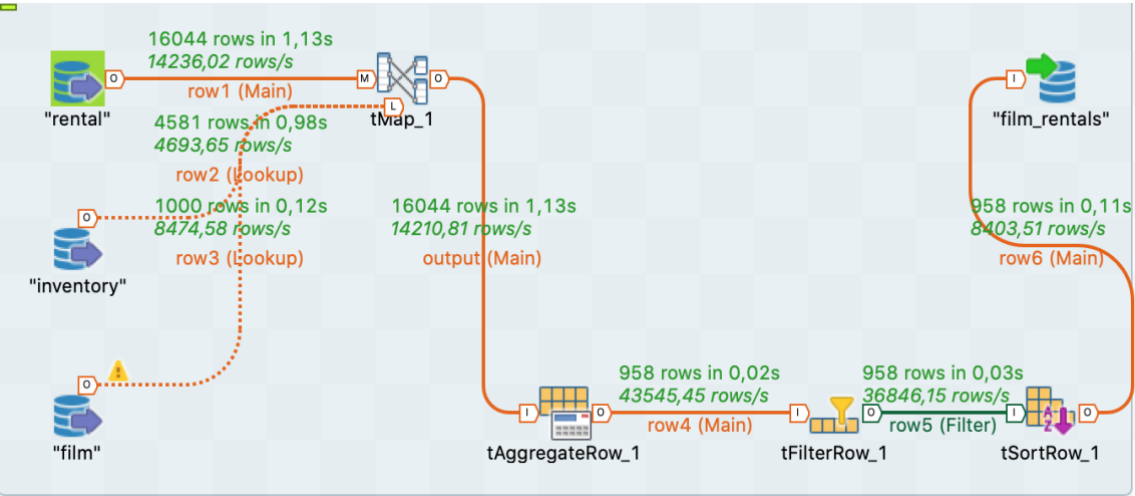
Explain

Messages

Notifications

	customer_id [PK] integer	first_name text	last_name text	total numeric
1	148	Eleanor	Hunt	211.55
2	526	Karl	Seal	208.58
3	178	Marion	Snyder	194.61
4	137	Rhonda	Kennedy	191.62
5	144	Clara	Shaw	189.6

**Ejercicio 7:** Cargar una tabla con el número de veces que se ha alquilado cada película, solo con aquellas que han sido alquiladas al menos una vez:



Query Editor Query History

```
1 select * from film_rentals
```

Data Output Explain Messages Notifications

	film_id [PK] integer	film_title text	total_rentals integer
1	103	Bucket Broth...	34
2	738	Rocketeer M...	33
3	331	Forward Tem...	32
4	382	Grit Clockwork	32
5	489	Juggler Hardly	32

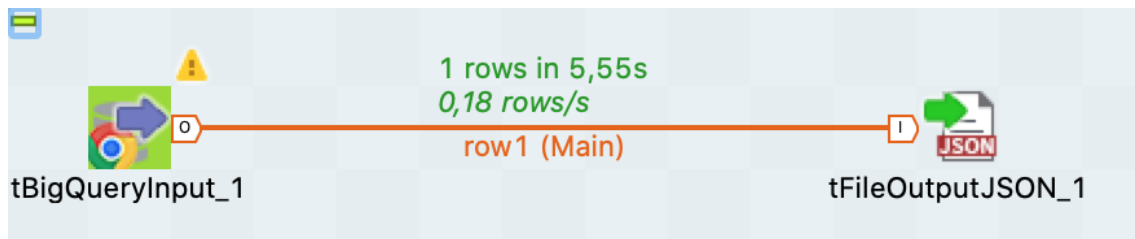
Query Editor Query History

```
1 select * from film_rentals where total_rentals = 0
```

Data Output Explain Messages Notifications

	film_id [PK] integer	film_title text	total_rentals integer
--	-------------------------	--------------------	--------------------------

**Ejercicio 8:** Leer de Big Query de la tabla alumnos filtrando los que contengan un nombre en concreto:



**tBigQueryInput\_1**

Esquema: Repositorio | JDBC (JDBC):Big\_Query - ALUMNOS | Edit schema

Authentication

Authentication mode: Service account

Service account credentials file: "/Users/carlosdonosocabero/Desktop/favorable-logic-289913-3d116c453699.json"

Project Id: "favorable-logic-289913"

☐ Use Legacy SQL

Query: "select \* from edem1.ALUMNOS where nom like '%Célia%'"