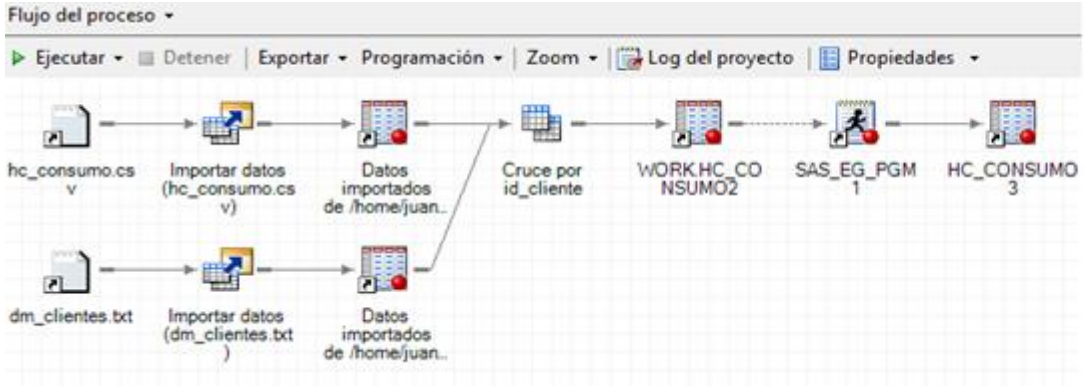



```
data work.hc_consumo3;
 set work.hc_consumo2;
 where id_segmento in (1,2,3);
 id_anyo = int(id_periodo/100);
 id_origen = 'Fichero';
 run;

proc sort data=work.hc_consumo3;
 by id_segmento;
 run;
```

Pasos básicos transformación

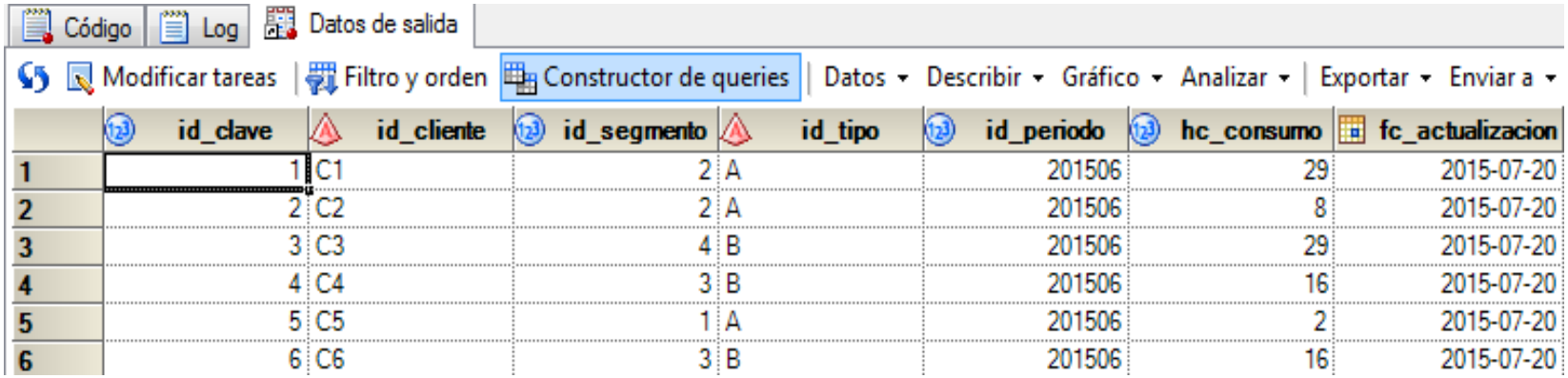
sas

datademy

Trabajar con tablas SAS

Trabajar con tablas SAS:

- Las tablas SAS son ficheros propios de SAS donde podemos almacenar la información en el formato habitual de tabla de BBDD (filas y columnas). Las tablas SAS pueden ser consultadas mediante el constructor de queries o las utilidades de filtro y orden, pueden ser exportadas a otros formatos o pueden ser utilizadas en informes, gráficos o tareas de Enterprise Guide

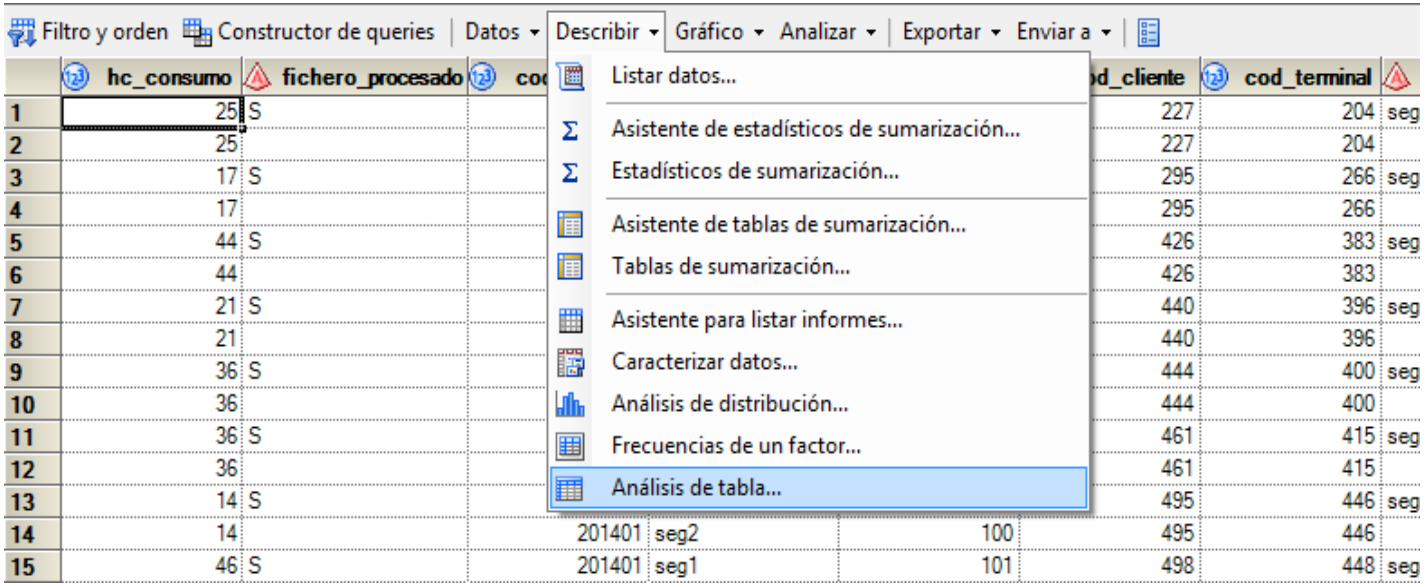


	id_clave	id_cliente	id_segmento	id_tipo	id_periodo	hc_consumo	fc_actualizacion
1	1	C1	2	A	201506	29	2015-07-20
2	2	C2	2	A	201506	8	2015-07-20
3	3	C3	4	B	201506	29	2015-07-20
4	4	C4	3	B	201506	16	2015-07-20
5	5	C5	1	A	201506	2	2015-07-20
6	6	C6	3	B	201506	16	2015-07-20

Trabajar con tablas SAS

Trabajar con tablas SAS:

- Desde una tabla SAS podemos lanzar las tareas de Enterprise Guide

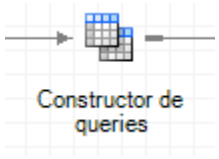


	hc_consumo	fichero_procesado				od_cliente	cod_terminal	
1	25	S				227	204	seg
2	25					227	204	
3	17	S				295	266	seg
4	17					295	266	
5	44	S				426	383	seg
6	44					426	383	
7	21	S				440	396	seg
8	21					440	396	
9	36	S				444	400	seg
10	36					444	400	
11	36	S				461	415	seg
12	36					461	415	
13	14	S				495	446	seg
14	14		201401	seg2	100	495	446	
15	46	S	201401	seg1	101	498	448	seg

Pasos básicos de transformación

Diferentes formas de implementar pasos de transformación:

- Para validar y elaborar el dato partiendo de la información en bruto ya importada en SAS son necesarios una serie de pasos de transformación que podremos implementar utilizando pasos data de SAS Base o bien utilizar una tarea E. Guide. Un tarea muy común es el constructor de queries genera código SQL. Es habitual que un paso data de SAS tenga su equivalente en SQL.



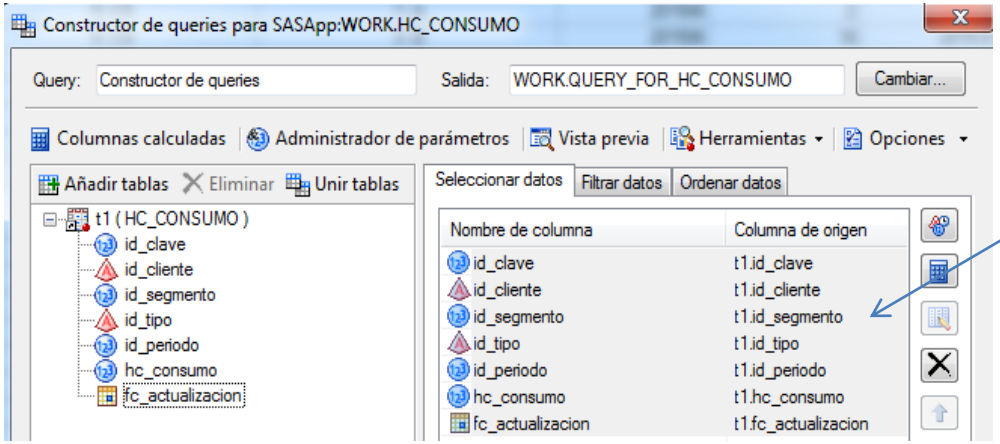

```
DATA <tabla_destino>;
SET <tabla_origen>;
/* transformaciones */
RUN;
```

```
PROC SQL;
CREATE TABLE <tabla_destino> AS
(SELECT *  /* transformaciones */
FROM <tabla_origen>);
QUIT;
```

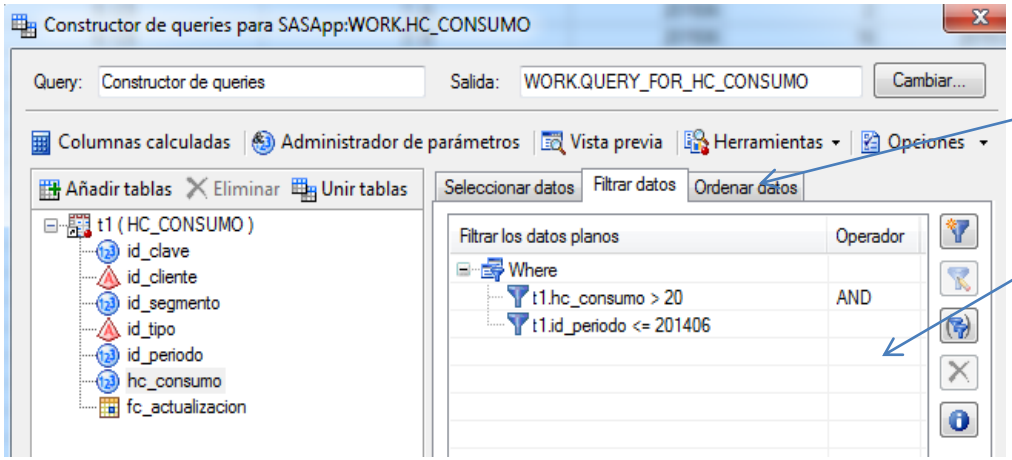
Pasos básicos de transformación/ Constructor de queries

Constructor de queries:

- El **constructor de queries** nos permite realizar consultas concretas sobre los datos. Indicando los campos que seleccionamos, el filtro que aplicamos y el orden de salida.



Selección de campos



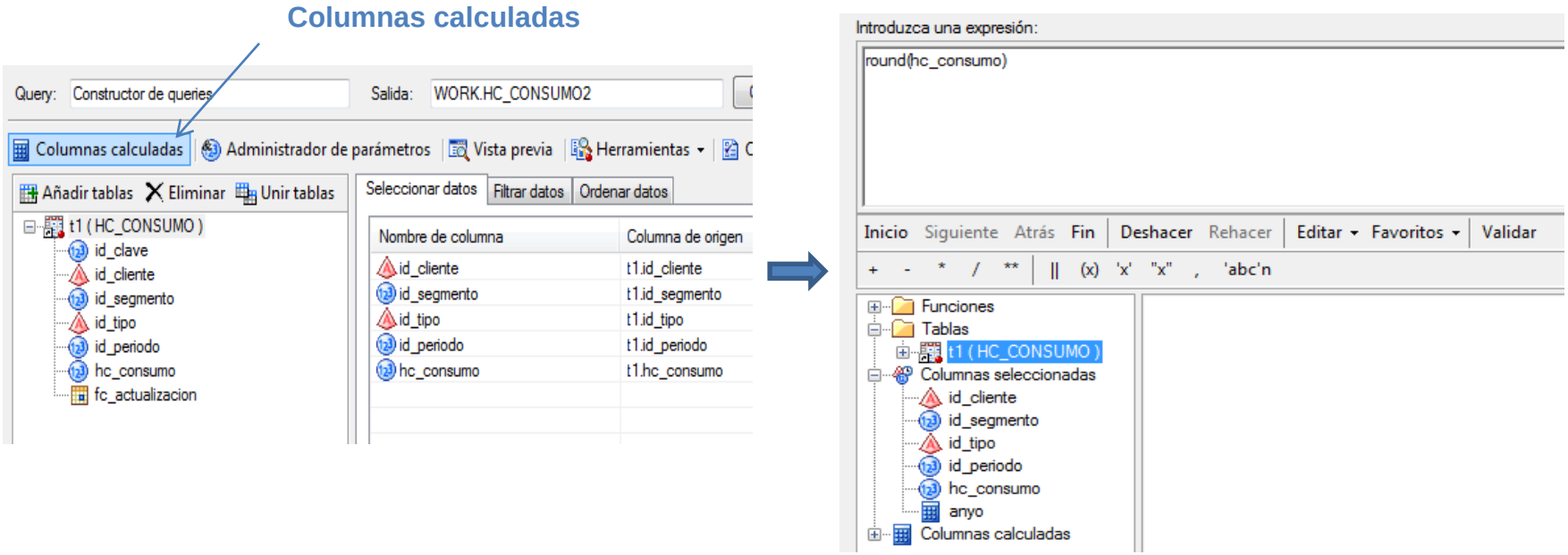
Campos ordenación

Condición de filtrado

Pasos básicos de transformación/ Constructor de queries

Constructor de queries:

- En el constructor de queries podemos introducir algunos pasos de transformación, como la creación de nuevos campos en base a los existentes. El editor de expresiones permite introducir de forma guiada el campo nuevo a calcular. Podemos obtener: recodificación de campos, columnas agregadas o expresiones avanzadas.



Pasos básicos de transformación/ Leer conjuntos de datos SAS y crear variables

Paso DATA SET:

```
data consumo_1;
set tabsas.lectura_excel;
run;

data tabsas.consumo_1;
length hc_consumo 8. fichero_procesado $1.;
set tabsas.lectura_excel;
hc_consumo = round(50*ranuni(1));
fichero_procesado = 'S'
run;
```

La instrucción **SET** asigna al dataset del paso **DATA** los mismos datos, variables y propiedades que el dataset de la instrucción SET.

Sintaxis:

DATA <tabla_destino>;
SET <tabla_origen>;
RUN;

Se pueden crear nuevas variables en la tabla destino, asignando el valor que corresponda.

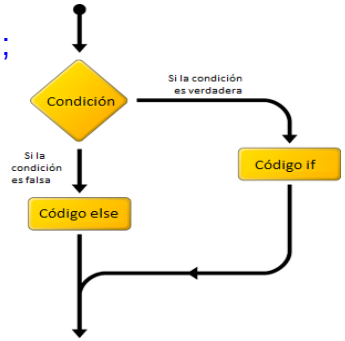
Empleando la clausula **LENGTH** definimos la longitud y el tipo de las variables.

Pasos básicos de transformación/ Procesos condicionales

Paso DATA Condicional:

```
data tabsas.consumo_2;
set tabsas.consumo_1;
if hc_consumo >= 20 then nivel_consumo = 2;
else nivel_consumo = 1;
run;
```

```
data tabsas.consumo_2;
set tabsas.consumo_1;
if hc_consumo >= 20 then do;
                nivel_consumo = 2;
end;
else do;
               nivel_consumo = 1;
end;
run;
```



Sentencia **condicional**:

Se pueden condicionar la ejecución de determinadas sentencias a el cumplimiento de determinada condición. Existiendo como en las estructuras condicionales estándar dos posibles ramas : **IF - ELSE**

```
IF condicion THEN acción;
  ELSE  acción;
```

La estructura **DO – END** permite la inclusión de un bloque de instrucciones:

```
IF condicion THEN DO;
    acción1;
    acción2;
END;
ELSE DO;
    acción1;
    acción2;
END;
```

Pasos básicos de transformación/ Procesos condicionales

Paso DATA Condicional. Múltiples ramas y borrado:

```
data consumo_2;
set tabsas.consumo_2;
if hc_consumo < 10 then n_consumo = 1;
else if hc_consumo < 30 then n_consumo = 2;
else if hc_consumo < 50 then n_consumo = 3;
else n_consumo = 4;
run;

data consumo_3;
set tabsas.consumo_2;
if hc_consumo < 20 then delete;
run;

data consumo_6;
set tabsas.consumo_5;
if (cod_tarifa = 100) and (cod_segmento = 'seg2');
run;
```

Podemos crear varias ramas con la estructura **elseif , filtrar los registros** de la salida y también borrar registros añadiendo la sentencia **DELETE**:

- Sintaxis:

a) Varias ramas:

IF condición THEN acción;

ELSEIF condición THEN acción;

b) Borrado registros:

IF condición THEN DELETE;

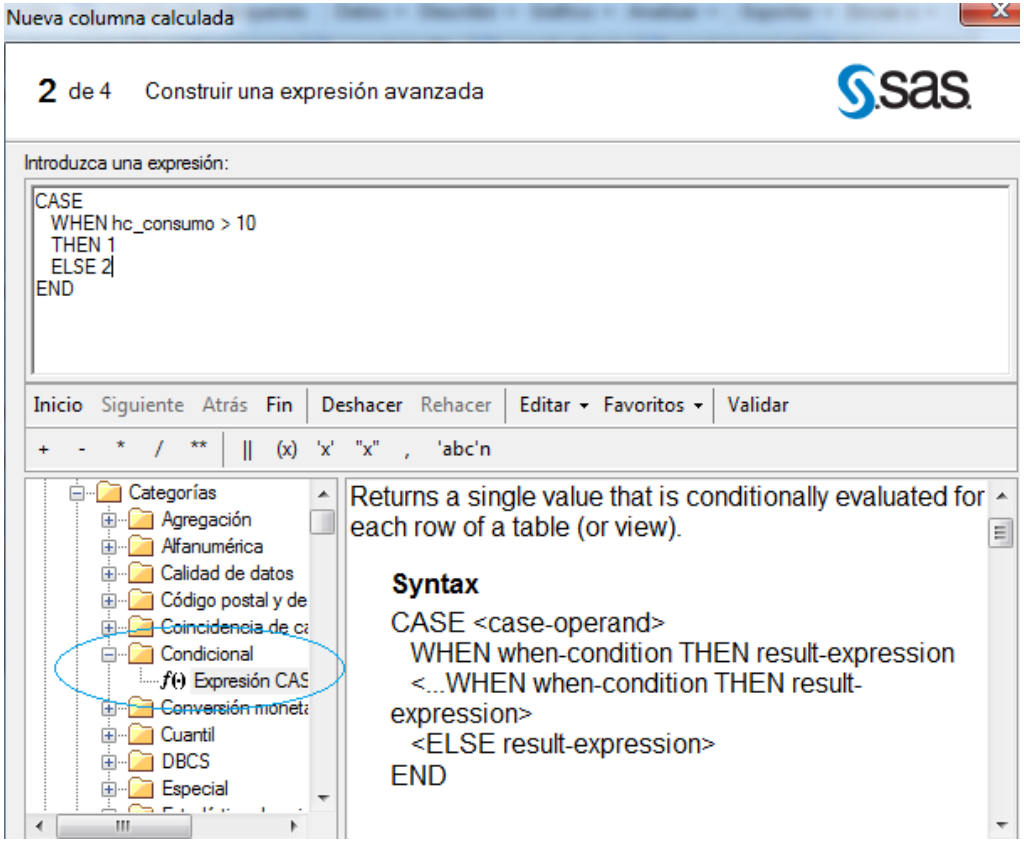
c) Filtrado registros a la salida:

IF condición;

Pasos básicos de transformación/ Procesos condicionales

Estructuras condicionales con el constructor de queries:

- Para introducir estructuras condicionales utilizamos el constructor de queries. Añadimos una columna calculada que se obtiene empleando las función condicional **CASE/WHEN**.



Pasos básicos de transformación/ Gestionar variables

Gestión de variables (rename, keep y drop):

```
data tabsas.consumo_4 (drop = cod_secuencia    nivel_consumo);
set tabsas.consumo_2;
run;

data tabsas.consumo_4 (keep = cod_cliente cod_periodo
cod_segmento cod_tarifa
cod_terminal hc_consumo);
set tabsas.consumo_2;
run;

data tabsas.consumo_5 (rename = (nivel_consumo=n_consumo));
set tabsas.consumo_2;
run;
```

- **KEEP:** La instrucción **KEEP** permite guardar en el dataset creado sólo las variables allí mencionadas. Las demás variables son eliminadas del dataset resultante.

- **DROP:** La instrucción **DROP** permite ELIMINAR del dataset las variables allí mencionadas. Las demás variables continúan en el dataset resultante.

- **RENAME**: Permite renombrar una variable del.

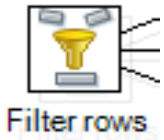
	A	B	C	D	E	F	G	H
1	cod_tarifa	cod_segmen	cod_cliente	cod_periodo	cod_termina	hc_consumo	desc_segmento	desc_tarifa
2	100	seg1	1	201310	1	9	Segmento 1	Tarifa 1
3	102	seg3	1	201311	1	37	Segmento 3	Tarifa 3
4	104	seg1	1	201305	1	49	Segmento 1	Tarifa 5
5	104	seg1	1	201312	1	45	Segmento 1	Tarifa 5
6	105	seg4	1	201304	1	19	Segmento 4	Tarifa 6
7	104	seg1	1	201405	1	49	Segmento 1	Tarifa 5
8	105	seg4	1	201404	1	19	Segmento 4	Tarifa 6
9	100	seg4	2	201309	2	39	Segmento 4	Tarifa 1
10	103	seg3	2	201308	2	30	Segmento 3	Tarifa 4
11	101	seg2	3	201404	3	48	Segmento 2	Tarifa 2
12	101	seg2	3	201304	3	48	Segmento 2	Tarifa 2
13	100	seg3	3	201308	3	40	Segmento 3	Tarifa 1
14	104	seg3	3	201301	3	12	Segmento 3	Tarifa 5
15	104	seg3	3	201401	3	12	Segmento 3	Tarifa 5
16	102	seg2	4	201406	4	31	Segmento 2	Tarifa 3
17	102	seg2	4	201306	4	31	Segmento 2	Tarifa 3
18	104	seg3	4	201306	4	12	Segmento 3	Tarifa 5

Pasos básicos de transformación/ Filtrar observaciones

Paso DATA con filtros:

```
data consumo_6;
set tabsas.consumo_5;
where cod_tarifa = 100 and cod_segmento = 'seg2';
run;

data consumo_6;
set tabsas.consumo_5 (where = (cod_tarifa = 100
and cod_segmento = 'seg2'));
run;
```



- **WHERE:** La instrucción **WHERE** permite filtrar las observaciones que se deseen mediante la inclusión de condiciones.

- Podemos aplicar el filtro en la tabla de entrada del paso data o en la de la salida:

  - Filtro tabla salida:

    ```
    DATA <tabla_salida>;
    SET <tabla_entrada>;
    WHERE ….
    Run:
    ```

  - Filtro tabla entrada:

    ```
    DATA <tabla_salida>;
    SET <tabla_entrada (WHERE …)>;
    Run:
    ```

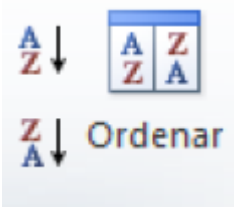
Pasos básicos de transformación/ Ordenar conjuntos de datos

Paso DATA ordenación:

```
proc sort data=tabsas.consumo_5;
by cod_cliente cod_tarifa;
run;


proc sort data=tabsas.consumo_5;
by descending cod_cliente cod_tarifa;
run;

proc sort data=tabsas.consumo_5;
by _ALL_;
Run;
```



• **SORT:** La instrucción **SORT** ordena el conjunto de datos que se le indique.

proc sort DATA= <tabla_entrada> OUT=<tabla_salida>;
by <lista de campos>;
run;

• La ordenación es por defecto en orden ascendente, para que la haga en orden descendente hay que indicarlo: by descending <campo>

• La clausula OUT es opcional, si se omite realiza la ordenación sobre el mismo conjunto de datos.

•Para ordenar por todos los campos, emplear la cláusula _ALL_.

Pasos básicos de transformación/ Ordenar conjuntos de datos

Opciones paso DATA ordenación:

```
proc sort data=total1 out=total2 nodupkey dupout=duplis;
by cod_periodo cod_segmento cod_tarifa cod_cliente
cod_terminal cod_secuencia n_consumo  hc_consumo;
run;

proc sort data=total1 out=total2 nodupkeys dupout=duplis;
by _ALL_;
run;

proc print data=duplis;
title 'Registros duplicados: ';
run;
```

cod_periodo	cod_cliente	cod_tarifa	cod_segmento	hc_consumo
201401	100058	100	S1	20
201401	100060	101	S2	12
201401	100060	101	S2	12
201401	100061	105	S5	35

- Opciones más habituales **SORT:**

  - **NODUPKEYS:** elimina los registros duplicados que haya en base a la clave indicada. Para eliminar duplicados por todos los campos emplear clausula **NODUPLICATES.**

  - **DUPOUT**: indica la tabla en la que guardamos los registros duplicados.

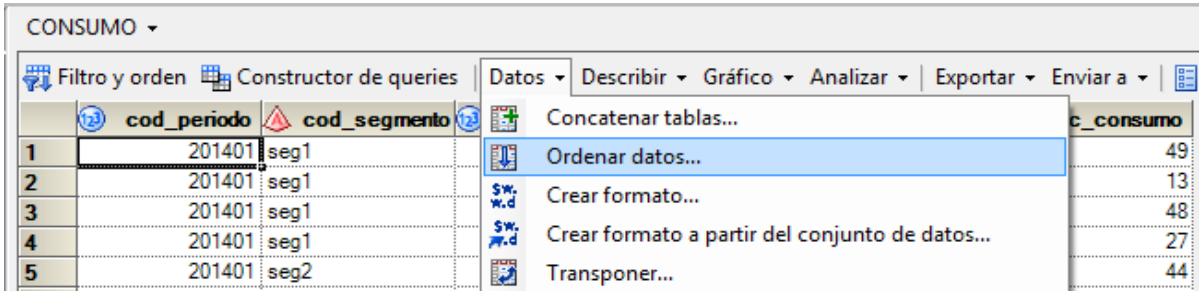
  - **THREADS/NOTHREADS**: permite la ejecución en varios hilos de la ordenación en caso de que por volumen sea aconsejable.

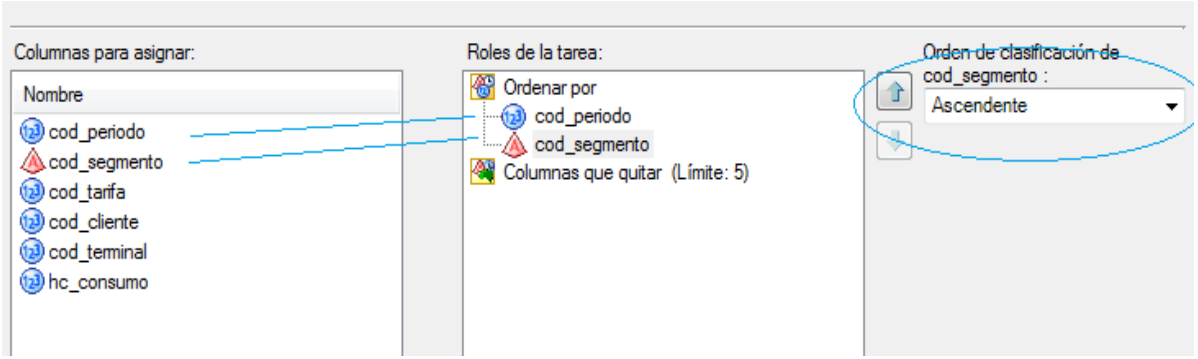
  - **TAGSORT:** Guarda en los ficheros temporales sólo los campos indicados en el BY rediciendo el espacio de memoria utilizado.

Pasos básicos de transformación/ Ordenar conjuntos de datos

Ordenación de datos con tarea E. Guide:

- La tarea '**Ordenar datos**' nos permite ordenar los registros de una tabla SAS en base a una clave de ordenación que definimos. La clave puede estar compuesta de varios campos y la ordenación puede ser ascendente o descendente

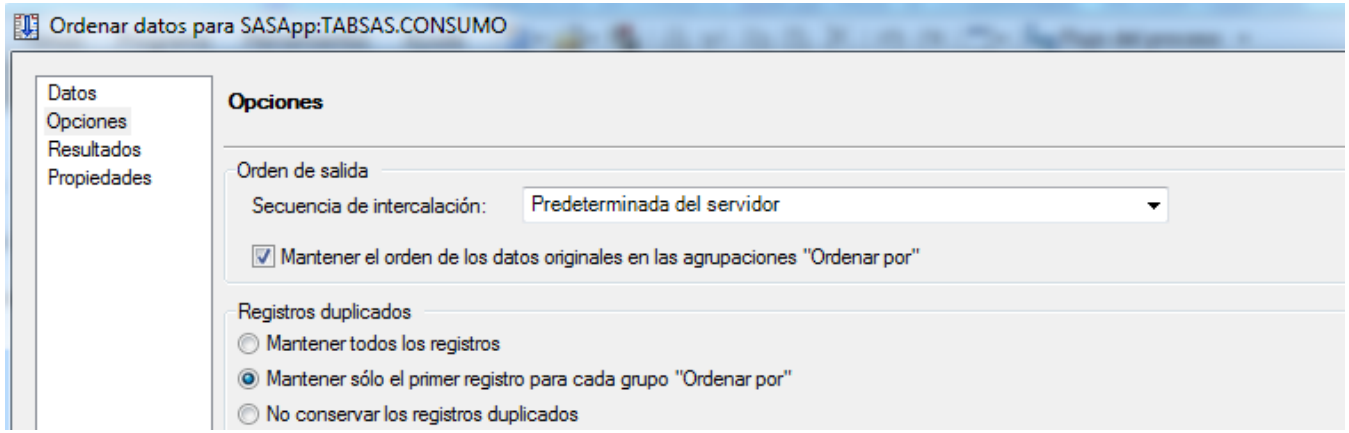


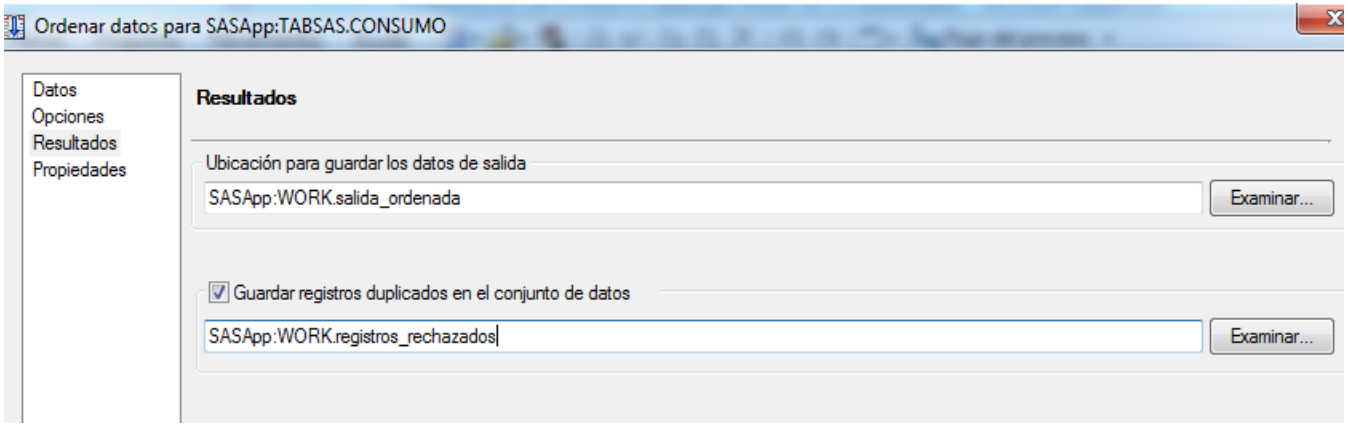


Pasos básicos de transformación/ Ordenar conjuntos de datos

Ordenación de datos con tarea E. Guide:

- La tarea '**Ordenar datos**' nos permite también **eliminar registros duplicados** por la clave de ordenación y especificar tablas de salida para el resultado ordenado y para los registros rechazados





Pasos básicos de transformación/ Aplicación formatos

Aplicación formatos:

- La aplicación de formatos ayuda a visualizar mejor los valores. Ejemplos de formatos**:**
  - **BESTW.D:** Númerico de W posiciones con D decimales. Ejemplo: best5.2 -> 120.79
  - **COMMAW.D:** Numérico de W posiciones con D decimales , el carácter ‘,’ es el separador de miles y el ‘.’ es el separador de decimales. Ejemplo comma8.2: 13,200.75
  - **COMMAXW.D:** Numérico de W posiciones con D decimales , el ‘.’ es el separador de miles y la ‘,’ es el separador de decimales. . Ejemplo comma8.2: 13.200,75
  - **DDMMYY8:** Formato de fecha. Ejemplo: ddmmyy8. 08/02/17
  - **EUROW.D:** Númerico de W posiciones con D decimales y símbolo euro. Ejemplo: euro5. -> E535

imp1	imp2	imp3	imp4	imp5	fecha
120.79	13,200.75	E435	566	13.200,75	08/02/17

- También es posible aplicar **formatos a la entrada (sentencia informat)** para interpretar determinados formatos de entrada