### Resumen

En este Proyecto de Fin de Grado, el objetivo final es crear un flujo de transformación de datos ligado a un *Dashboard* interactivo para extraer conocimiento y estadísticas acerca del Mundial de Motociclismo de una forma clara y accesible a todos.

Comenzaremos por listar las herramientas que se han utilizado en este proyecto, tales como Pentaho, PostgreSQL o Superset, y la razón detrás de su elección.

Tras ello explicaremos cómo he obtenido los datos a través de las tablas de la Wikipedia, como se estructuran, limpian y completan estos datos. Todo esto para que exista un formato estándar entre todas las categorías y temporadas para poder ser analizadas por las herramientas ETL.

Definiremos la arquitectura de la solución que planteamos, el diagrama del DataWarehouse y la lógica de negocio tras ella.

Por último hablaremos del análisis realizado sobre los datos, la información relevante y los KPI/Filtros necesarios

### Abstract

**lo mismo pero en ingles**

### **Diccionario:** Dashboard

ETL

DataWarehouse

*webscraping*

URL

reporting

dashboards

CRM

## 1.Introducción:

El Campeonato Mundial de Motociclismo tiene sus orígenes en el año 1907 cuando se organiza el primer Tourist Trophy de la Isla de Man. Este, se convirtió en una de las pruebas deportivas más conocidas y temidas de la competición. Cuarenta años después, en 1949, la "Fédération Internationale de Motocyclisme", más conocida como FIM, crea el Campeonato Mundial de Motociclismo. Este, a lo largo de sus más de 70 años y más de 1.000 grandes premios, nos ha dado algunas de las carreras más emocionantes del motor y varias personalidades cuyo nombre es mas grande que el propio campeonato.



Figura 1 – Logo de MotoGP

En el año 2002, tras el paso a los motores de cuatro tiempos, pasa a llamarse como todos lo conocemos hoy en día: MotoGP. Su alta competitividad, igualdad, velocidad y personalidades, lo han llevado a ser no solo el segundo deporte de motor más visto del planeta (solo por detrás del mundial de Formula1), sino también una de las competiciones con más renombre a nivel mundial.

Tantas carreras y resultados hacen difícil recordar todos los logros y estadísticas que han acontecido ante nuestras pantallas. Es por ello que nace la necesidad de poder ver la historia del campeonato de una forma clara y sencilla, donde podamos conocer más sobre el deporte que a tantos apasiona.

Es por ello que se decidió crear este proyecto. El objetivo es extraer conocimiento a través de un Cuadro de Mandos para mostrar décadas de datos y usando herramientas de código abierto. El uso de herramientas de código abierto hace posible que todo el que quiera, ya sea un aficionado más o un profesional relacionado con el mundial, pueda extraer estadísticas y averiguar más sin los límites que el software propietario impone.

Las herramientas en cuestión son:

·\*Pentaho Data Integration\*, para el procesamiento de datos.

·\*PostgreSQL\*, para el alojamiento del Data Warehouse.

·\*Streamlit+Plotly\*, para la visualización de datos

### 1.1 Contexto del Problema

Motogp está experimentando a día de hoy un alto crecimiento a nivel mundial. Solo en está última temporada de 2023, las audiencias a nivel mundial han aumentado un 20%. El número de fans en los circuitos ha llegado a aumentar en un 80% en algunos circuitos, con la mayoría de los circuitos aumentando en más de un 30% en el número de aficionados que asisten al fin de semana de carreras. El mayo pasado, el legendario circuito de Le Mans en Francia, batió el récord histórico, con más de 278.000 aficionados disfrutando del fin de semana de carreras.

Hoy en día la presencia en redes sociales es imprescindible para cualquier competición deportiva y empresa. Las redes sociales del campeonato suman más de 50 millones de aficionados entre todas las plataformas y redes sociales, incluyendo la recientemente creada Threads del gigante tecnológico Meta.

Pero estas plataformas no permiten a los fans tener una imagen completa de los resultados y estadísticas históricas. Siempre es necesario tener una página web. En los últimos meses, la página web oficial del mundial ha comenzado un proceso de modernización que tiene como objetivo mejorar la imágen del mundial y de proveer resultados, noticias y datos de una forma más clara.

A pesar de estos esfuerzos, a día de hoy la página sigue en construcción. Faltan todavía imágenes e incluso secciones completas, y muchos de los botones no llevan a sus destinos. Sin la posibilidad de utilizar la versión antigua, muchos fans nos hemos visto sin acceso a los recursos que utilizabamos para saber más sobre la herencia histórica del mundial y sus noticias más recientes.

Es por esto que nace este proyecto para el aficionado medio, para ayudarle a entender mucho mejor la trayectoria deportiva de un piloto en particular o de su equipo favorito. La capacidad de poder ver todos los datos disponibles en un solo lugar de forma clara, no solo hará que los aficionados actuales entiendan mejor su campeonato favorito, sino que incluso los profesionales del mundial, entre ellos los periodistas, podrán mejorar sus artículos lo que potencialmente puede atraer a más personas al mundial.

## 1.2. Solución planteada

Crearemos un cuadro de mandos interactivo para poder mostrar de forma clara y concisa los datos y resultados que existen hoy en día en Wikipedia.

Wikipedia es un excelente recurso ya que muestra todos los datos en tablas, y gracias a las herramientas que existen en Excel o en PowerBI es muy sencillo extraer estas tablas.



Figura 2 – Tablas de resultados de equipos en Wikipedia

Aunque el este formato estructurado muestra los datos de una forma aparentemente clara, estos siguen estando en bruto y debemos transformarlos primero para darle una forma que pueda ser analizada.

La creación de los dashboards evita que los usuarios tengan que ir temporada por temporada recordando o apuntando resultados para posteriormente sacar conclusiones. El que el usuario final pueda filtrar los datos a través de una interfaz gráfica no solo agiliza el proceso, sino que permite contextualizar mejor las conclusiones que se obtengan.



Figura 3 – Dashboard o Cuadro de Mandos de ejemplo.

#### 1.2.1 Objetivos principales

El objetivo principal de este proyecto es proveer a cualquier persona interesada en el mundial la capacidad de tener una visión más completa del campeonato, su historia y como trascendieron algunos de las temporadas más relevantes del mismo

#### 1.2.2 Objetivos Secundarios

La creación de proyecto de análisis de datos tiene como objetivos los que se exponen a continuación:

* La creación de un proceso de ingeniería de datos: incluye una ETL de código abierto para procesar los datos obtenidos, así como el diseño de un Data Warehouse que permita no solo almacenar los datos sino también analizarlos. Todo esto con el fin de crear un cuadro de mandos que permita una vista gráfica de los datos
* Evaluar las trayectorias de los pilotos: El análisis de todas las temporadas del mundial de los pilotos nos permitirá tener una vista completa del recorrido de los mismos en el mundial, permitiendo a los usuarios saber más sobre los éxitos y dificultades que ha afrontado cada piloto a lo largo de su carrera deportiva.
* Comparar los éxitos de los equipos: Este mismo análisis permite poner en contexto la dominación de ciertas motos y fabricantes en el mundial. Las mejoras de los constructores se pueden producir no solo de temporada en temporada, sino también dentro de una misma temporada. La posibilidad de ver la evolución de los resultados ayuda a poner en contexto estos cambios en rendimiento.
* Contextualizar la globalidad del campeonato: Los datos geográficos de los pilotos y los grandes premios ayudarán a entender mejor el alcance global del campeonato y a identificar qué países e incluso regiones son las que más pueblan las parrillas de salida.
* Aumentar el interés de las personas por el mundial: Se pretende que este fácil acceso a los datos haga el campeonato más atractivo al aumentar la información a la que los aficionados y periodistas tienen acceso .

En resumen, los objetivos de este proyecto de análisis de datos del mundial de MotoGP no solo tienen como objetivo la creación de una arquitectura de datos sólida y escalable, sino también hacer los datos accesibles a más personas independientemente de sus conocimientos técnicos.

### 1.3 Contenido de la memoria

En primer lugar se hablará sobre el origen de datos y el formato en el que se encuentran los mismos. Tras ello se propondrá una posible solución al modelo analítico que deberá tener el Data Warehouse para posteriormente poder crear un cuadro de mandos. En este, podremos volcar todos los datos procesados para que todos los aficionados que lo requieran puedan visualizarlos de forma clara y legible.

## 2. Datos en bruto

En está sección hablaremos sobre el origen de datos, por qué ha sido seleccionado y cómo se pueden extraer los datos. Acto seguido, se describirá el estado en el que se encuentran los datos, y los procesos de normalización a los que deberemos someterlos para darles la forma necesaria para analizarlos. Este proceso se hará para todas las categorías del mundial, es decir:

* Motogp
* 250cc/Moto2
* 125cc/Moto3
* MotoE

Como origen de los datos se ha elegido la Wikipedia inglesa. Se ha optado por ella ya que las páginas de cada temporada son más completas que la versión española. Además que Excel ofrece una extracción eficiente de tablas desde orígenes web.

En el caso de la Wikipedia, las páginas de cada temporada agrupan los datos de todas las categorías hasta el año 2011 incluido. Tras esto, cada categoría tiene su propia página. Esto solo supone un ligero cambio en el proceso de extracción de datos en Excel, ya que tendremos que usar URLs diferentes y seleccionaremos unas tablas en vez de otras.

### 

### 2.2 Extracción de datos usando Excel

Tenerlos en Excel nos permite normalizar y completar los datos de forma rápida y sencilla.

Excel tiene la capacidad de extraer datos automáticamente de una página web usando su interfaz gráfica. Esto permite que cualquier persona sin experiencia en *web scraping* ni experiencia en programación en general, pueda extraer datos de una página web en formato tabular.

En la Figura 3 se puede apreciar el proceso que se sigue para colocar las tablas en Excel. Se va a la pestaña de ‘Datos’ y se selecciona la opción ‘De la web’. En el menú que nos aparece podemos introducir la URL de la página que nos interesa.

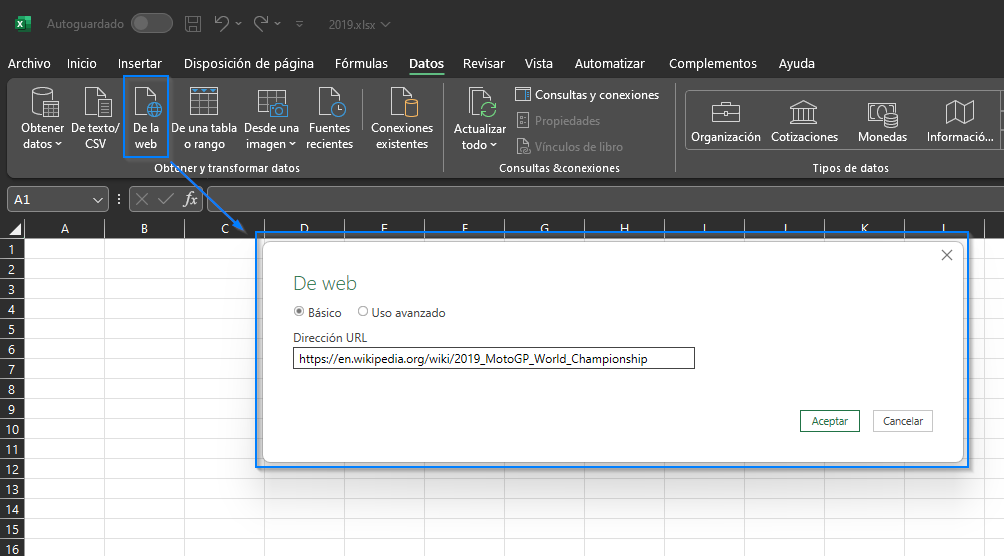


Figura 3 – Extraccion de datos Web desde Excel

Una vez se introduce la URL, nos aparece un menú donde podemos seleccionar múltiples tablas en una sola vez, que serán puestas de forma individual en hojas del Excel.

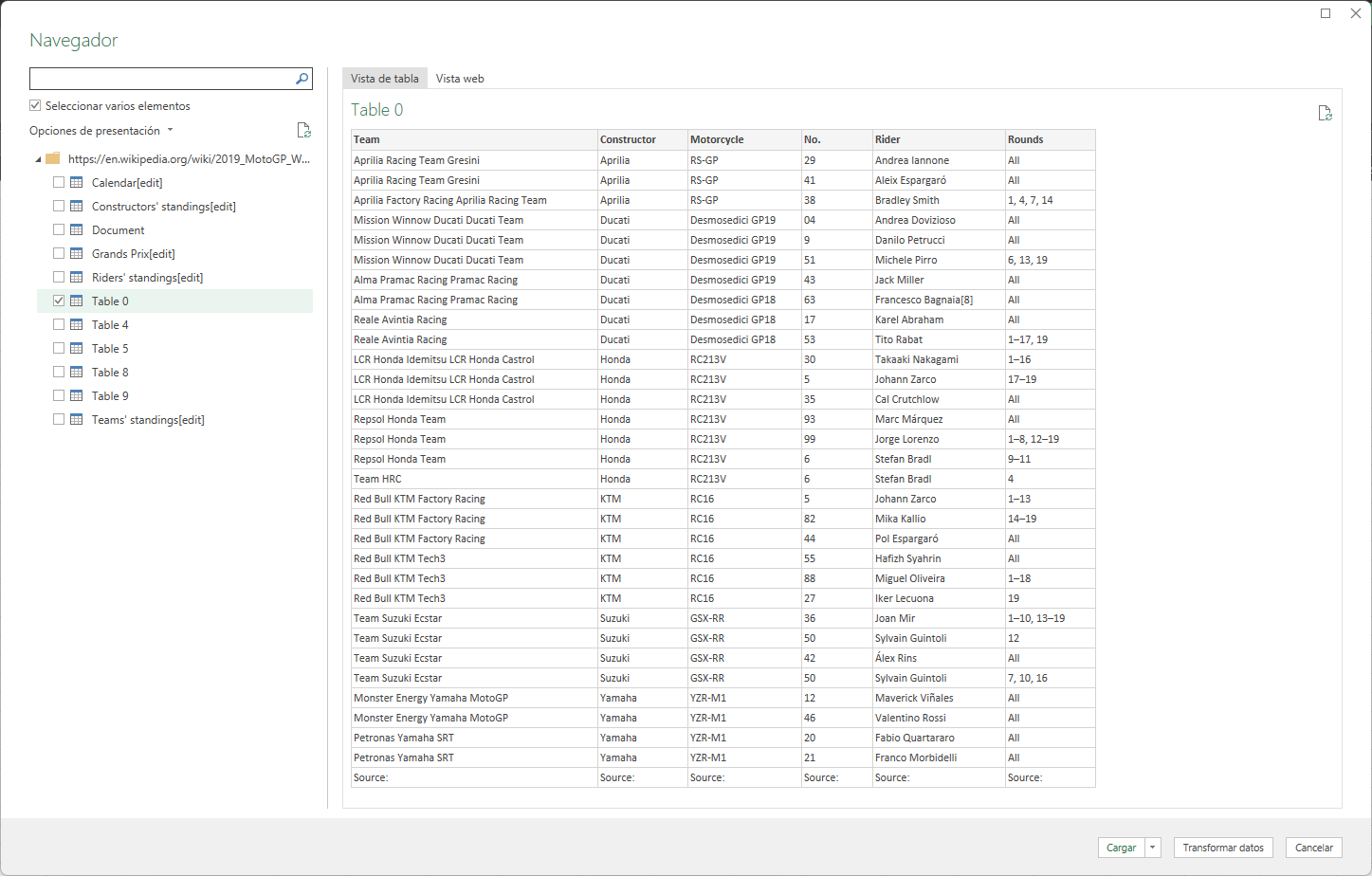


Figura 4 – Menú de selección de tablas

### 2.2 Conjunto de Datos de Equipos y Pilotos

Este tipo de tabla nos permitirá obtener los datos necesarios para relacionar a los pilotos, los equipos y los constructores en todas las categorías. Además de datos como las rondas en las que participó un piloto, el nombre de la moto que pilotaron y el número de cada piloto.

Vamos a poner como ejemplo la tabla de la temporada 2012 ya que contiene muchos de los problemas que debemos manejar antes de llegar a la ETL:

<https://en.wikipedia.org/wiki/2012_MotoGP_World_Championship#Teams_and_riders>

Figura 5 – Tabla de Pilotos y Equipos del Año 2012

Siguiendo el proceso anteriormente mencionado, podemos importar esta tabla a Excel. En los datos en bruto se nos presentan ciertos problemas que debemos solventar a mano ya que, o no podemos automatizarlos en la ETL, o es más práctico no dejar esta lógica en las manos del proceso ETL.

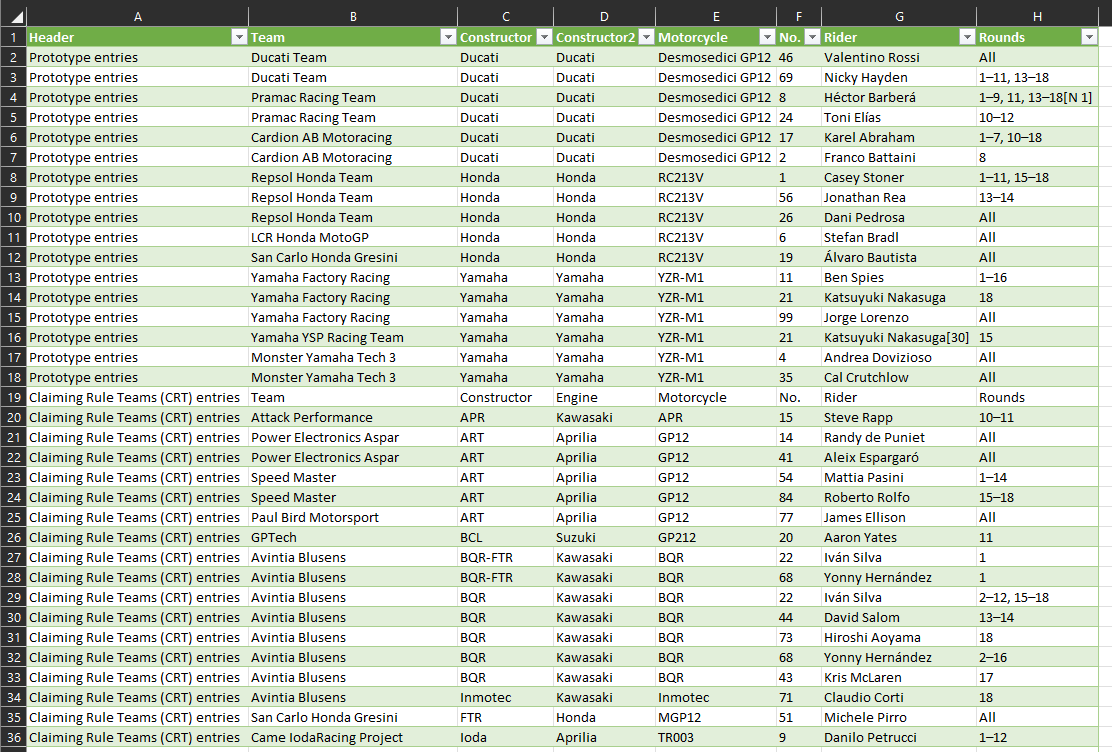


Figura 5 – Tabla de Pilotos y Equipos del Año 2012 en Excel

En la columna A, vemos que se distingue entre los equipos ‘Prototype’ (Prototipo) y los ‘CRT’. Esto se debe a que entre las temporadas 2012 y 2015 existían dos categorías dentro de MotoGP. ‘CRT’ pasó a llamarse ‘Open’ en el año 2014. Deberemos mantener esta diferenciación también en los datos, y colocar el tipo de moto siempre en el mismo lugar. En el resto de temporadas sin ‘CRT’ ni ‘Open’, la ETL automáticamente asigna el tipo de moto a ‘Prototype’. Esto se verá más adelante en la memoria.

En la columna B, hay equipos que compiten usando más de un nombre a lo largo de una temporada. Esto puede deberse a que en algunos países ciertos patrocinios son ilegales, o bien se alcanzan acuerdos con los patrocinadores para cambiar el nombre del equipo. Pero al tratarse siempre del mismo equipo, independientemente de los patrocinadores, debemos rectificar esto para que cada equipo tenga un único nombre en una temporada.

En las columnas B, E y H, hay pilotos, equipos y rondas que tienen alguna cita en la Wikipedia junto a su nombre (G-16). Esto se puede eliminar a mano en el Excel o se pueden usar Expresiones Regulares en el proceso ETL.

Entre los años 2012 y 2015, en las tablas de orígen, se distinguen dos constructores: El fabricante del chasis y el fabricante de los motores. En el caso de los ‘prototype’ este es siempre el mismo constructor. Pero en el caso de las ‘CRT’ y las ‘Open’ son diferentes. Por lo que se decide incluirlos en una sola columna a ambos fabricantes ya que es información relevante.

Además, deberemos añadir ciertas columnas que no se incluyen en las tablas originales ya que se dan por supuestos por el contexto del artículo de la Wikipedia. En concreto, se deben añadir:

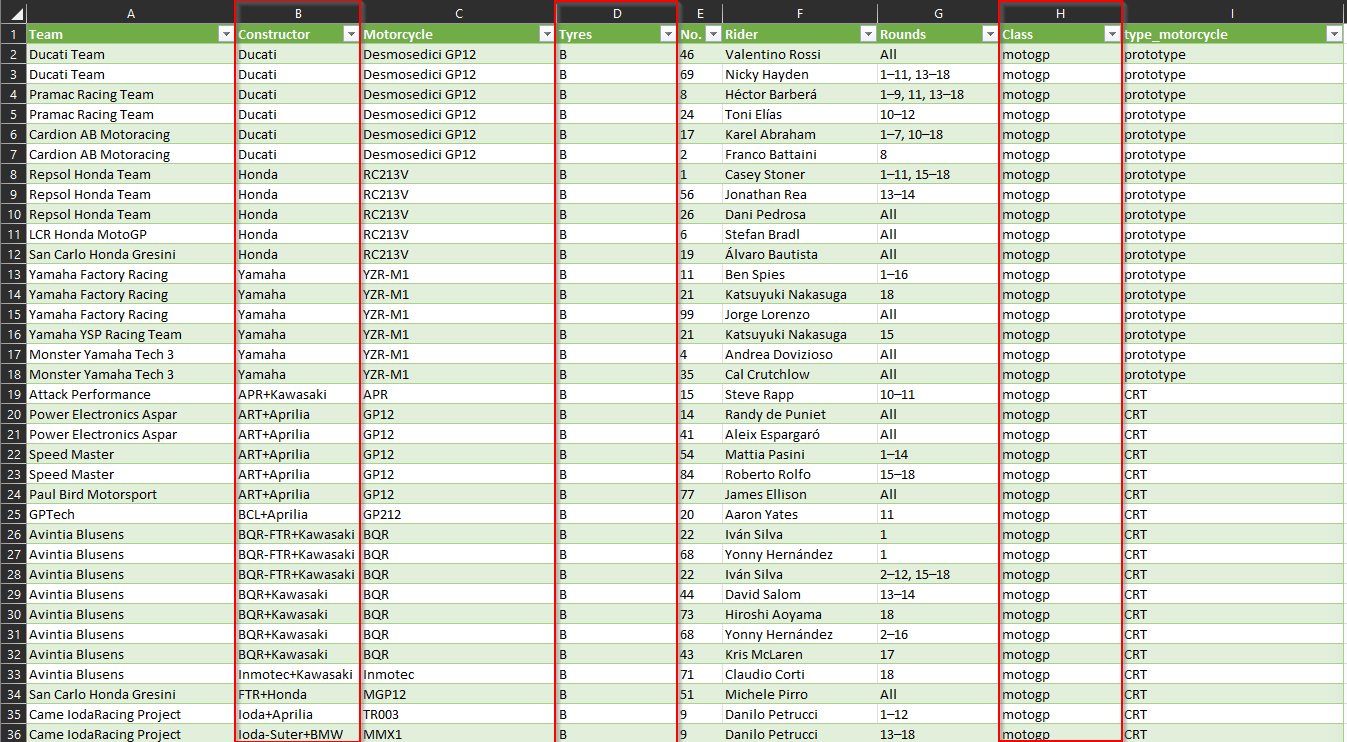
* Los neumáticos (Tyres - Columna D) usados por cada equipo: En las tablas hasta el año 2008 se incluye este dato ya que había múltiples proveedores de neumáticos. Entre las temporadas 2009-2015 todos los equipos fueron suministrados por Bridgestone, y a partir de 2016 el proveedor único es Michelín. Se añaden las iniciales de los proveedores:
  + B: Bridgestone
  + M: Michelin
  + D: Dunlop
* La categoría (Class - Columna H) de los equipos y pilotos: Como deberá haber hojas Excel de este estilo para cada una de las 4 categorías que manejaremos, es importante contextualizar estos datos con la categoría, ya que un piloto puede competir en más de una categoría en una temporada. Un ejemplo de esto es Anthony West en el año 2007.

Figura 6 – Tabla de Pilotos y Equipos del Año 2012, con las columnas mencionadas.

Las columnas anteriormente mencionadas cuentan con los siguientes tipos de datos:

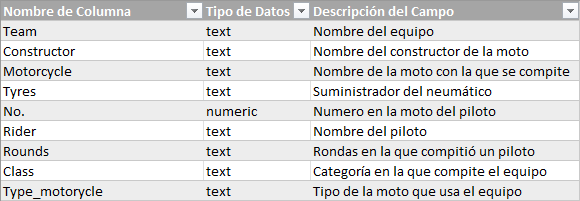


Figura 7 – Tipos de datos en la Hoja de Pilotos y Equipos

### 2.3 Calendario

Esta tabla nos proporciona los datos relativos a cada gran premio, como por ejemplo el orden de los mismos dentro de una temporada, la fecha en la que se corrió la carrera, el nombre del gran premio, y el circuito donde se celebró el fin de semana de carreras.



Figura 8 – Tabla de Calendario en Wikipedia

Este conjunto de datos es más limpio que el anterior. Lo único que debemos añadir es una nueva columna ‘Location’ para conocer la ubicación del gran premio. Esto se consigue dividiendo la columna ‘Circuit’ por el separador que en este caso es la coma. En el caso de que no incluya ese dato en el nombre, se añadirá usando *Web Scraping*. Este último proceso se verá más adelante en la memoria.

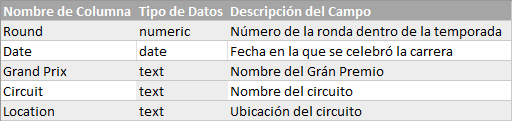


Figura 9 – Tipos de datos en la hoja de Calendario

### 2.4 Resultados por carrera

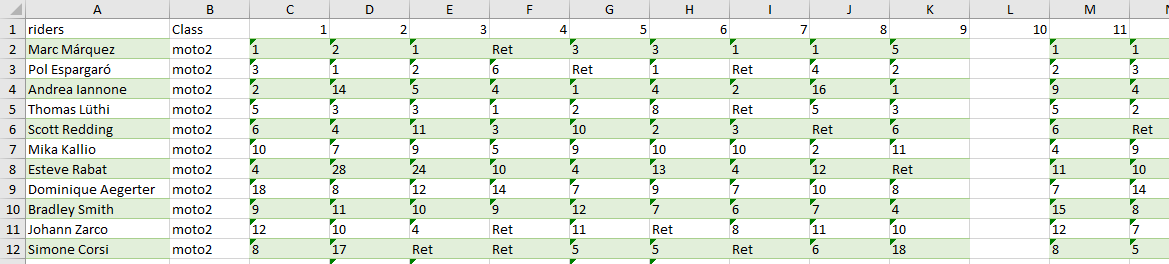
Está tabla llamada ‘Riders standings’, nos permite saber los resultados de cada piloto en cada carrera de la temporada. Estos resultados pueden ser la posición en la que terminó la carrera, o la razón por la que no lo hizo. Estas razones pueden ser Ret (Retirada), DNS (No comenzó la carrera) o DNQ (No se clasificó para la carrera) entre otros.



Figura 10 – Tabla de Resultados en Wikipedia

De está tabla nos interesa extraer los resultados por ronda de cada piloto. Para ello deberemos cambiar el nombre acortado del gran premio (QAT, SPA, …) por el número de ronda de ese gran premio. Esto se hace para poder relacionar cada resultado con el gran premio al que se corresponde. Al estar estos grandes premios en orden, es tan sencillo como crear una numeración ascendente hasta el final.

El dato de la moto no es necesario, al igual que el tipo de moto. Este dato se puede extraer directamente de la tabla de equipos. Hay otro arreglo que se deberá hacer, y este es exclusivo a las hojas de resultados de las categorías de Moto2/250cc y 125cc. Entre los años 2005 y 2013, el Grán Premio celebrado en Laguna Seca era exclusivo de la categoría MotoGP, así que para esa ronda, la columna de estas dos categorías más pequeñas deberá ser añadida a mano y quedar completamente vacía. Esto es vital para poder relacionar correctamente el resultado con la ronda.

Figura 11 – Ejemplo de hoja de resultados con la ronda de Laguna Seca vacía (Moto2)

## 3. Herramientas Software

Una vez visto el conjunto de datos, procedemos a describir y explicar las herramientas usadas en este proyecto. Listaremos la justificación tras su elección, sus ventajas, capacidades y modo de uso.

### 3.1 Pentaho

Pentaho es una familia de herramientas Open Source de Inteligencia de Negocio propiedad de Hitachi Vantara. Ofrece herramientas de integración de datos, reporting y dashboards entre otras. Ofrece dos tipos de producto: la Community Edition que es de carácter gratuito, y la Enterprise Edition, de pago pensada para empresas.

En nuestro caso usaremos la Community Edition de la herramienta Pentaho Data Integration.

PDI es un programa de Extracción, Transformación y Carga de datos de escritorio, es decir, que no está alojado en la nube como por ejemplo Azure Data Factory. Esto, sumado a su naturaleza open source, ofrece a cualquier tipo de usuario una puerta abierta para crear soluciones de Business Intelligence personalizadas.

El anteriormente mencionado proceso de Extracción, Transformación y Carga de datos (ETL) es uno de los pilares de la ingeniería de datos. Este proceso permite a los ingenieros de datos cargar datos en bruto desde sus orígenes. Tras esto estos datos se procesan y se les da la forma deseada. Está transformación incluye operaciones como la normalización de campos, cambio de tipos de datos, reordenación de campos, unión de datos de diferentes orígenes, entre otros. Finalmente, este proceso concluye con los datos procesados siendo almacenados en un sistema analítico como puede ser un DataWarehouse o un Lakehouse. En este proyecto usaremos el primer tipo. Los orígenes anteriormente mencionados pueden ser estructurados (por ejemplo tablas) o no estructurados (por ejemplo documentos).

A continuación, listamos más en detalle las numerosas ventajas que proporciona de uso de Pentaho Data Integration en los proyectos de ingeniería de datos:

* Naturaleza Abierta: Como hemos mencionado anteriormente, Pentaho, más en concreto PDI, es una solución ETL completamente Open Source, distribuida bajo la licencia Apache License Version 2.0. Esta licencia permite modificar y distribuir el software modificado no sólo para uso privado, sino también para uso comercial. Con las limitaciones de que no se puede responsabilizar de daños a los creadores originales

Esta licencia, sumada a su programación en Java, ha permitido a innumerables organizaciones personalizar la herramienta base para cumplir sus requerimientos. Una de las más notables es Apache Hop, que ofrece una actualización notable en el uso de la herramienta junto a la retrocompatibilidad con los proyectos ya creados en PDI.

* Ecosistema Pentaho: Como mencionamos anteriormente, aunque solo vayamos a usar PDI, Pentaho se compone de múltiples herramientas creadas con el propósito de extraer la máxima cantidad de información y conocimiento de los datos. Entre ellos cabe destacar:
  + Pentaho Reporting: permite visualizar informes a partir de modelos creados desde una fuente de datos.
  + Pentaho Dashboard: como su nombre indica, permite la creación de cuadros de mando, informes y gráficos a partir de un conjunto de datos.
  + Conectores: además de herramientas completas, Pentaho ofrece también la capacidad de operar con otras herramientas y sistemas externos. Estos Plugins se pueden adquirir desde el propio Marketplace de Pentaho. Algunas de estas permiten por ejemplo en envío de notificaciones Push a dispositivos móviles desde PDI o comunicarse directamente con Apache ZooKeeper para administrar sistemas distribuidos
* Flexibilidad y Versatilidad: PDI ofrece una gran variedad de orígenes de datos. Estos pueden ser ficheros como XML, CSV, JSON o Excel; correos electrónicos, cubos OLAP e incluso conexiones con CRMs como por ejemplo Salesforce.

Por supuesto también puede conectarse a numerosos tipos de bases de datos tanto relacionales como SQL Server, MySQL y Oracle; además de otras no relacionales como MongoDB y Cassandra.

Se pueden introducir scripts de JavaScript en las transformaciones para poder realizar operaciones más complejas que no ofrezcan PDI por defecto. Gracias a la posibilidad de ejecutar scripts en la Shell del sistema, lo que no solo posibilita el ejecutar múltiples acciones en el propio equipo sino que también permite la ejecución de scripts externamente en cualquier lenguaje de programación.

A pesar de todas las ventajas que ofrece Pentaho, su uso nos plantea ciertas dificultades y restricciones:

* Rendimiento: El hecho de ejecutarlo en una máquina local condiciona su rendimiento al hardware en el que se ejecute. Además de ofrecer posibilidades limitadas para coordinar operaciones concurrentes, lo que puede causar interbloqueos y condiciones de carrera en ciertos escenarios.
* Limitaciones técnicas: el uso de PDI a través de su interfaz gráfica permite a los usuarios abstraerse al realizar operaciones complejas en los datos, pero el usuario está limitado a la implementación de esos elementos. Esto se puede suplir usando los scripts, pero puede no ser suficiente en proyectos muy complejos.
* Soporte Limitado: La versión Community de Pentaho tiene una comunidad de usuarios detrás que puede no ser de utilidad para problemas más complejos. En cambio la versión Enterprise de Pentaho si ofrece un soporte técnico por parte de Hitachi a cambio de una suscripción anual al servicio.

### 3.2 PostgreSQL

PostgreSQL es una de las bases de datos relacionales más utilizadas y reputadas del mundo. Fue creada en el año 1996 por el profesor universitario Michael Stonebraker en la universidad californiana de Berkeley.

Siguiendo en la línea de este proyecto, PostgreSQL es un sistema de gestión de bases de datos de Código Abierto de tipo relacional orientado a objetos. Esto último permite operar con los datos como Objetos usando conceptos de Programación Orientada a Objetos como la herencia. Pero esta característica no la usaremos en este proyecto.

Algunas de sus ventajas son:

* Funciones multi lenguaje: PostgreSQL permite escribir funciones en la base de datos no solo en SQL, sino también en Perl, Python, R e incluso Shell. Esta flexibilidad permite a usuarios con diferente experiencia en programación sacar el máximo partido de la base de datos usando los puntos más fuertes de cada lenguaje de programación.
* Variedad de tipos de datos: permite usar no solamente los tipos de datos clásicos como numeric, string o boolean, sino también direcciones de red IPv4 e IPv6, elementos geométricos como puntos, líneas y polígonos; objetos JSON y arrays multidimensionales.
* Seguridad: ofrece el poder gestionar permisos, control de acceso y autenticación. Permitiendo así mantener la seguridad de los datos frente a ataques terceros maliciosos, y restringir la vista de datos a los diversos usuarios autenticados en función de su nivel de permisos.
* Precio: al ser open source es totalmente gratuito tanto para usuarios particulares como para empresas, siendo el único coste el alojamiento de los datos y los recursos de cómputo del sistema.

### 3.3 Streamlit

Streamlit es un *Framework* de Python para la creación de aplicaciones web, originalmente ideada para la Ingeniería de Datos y el Machine Learning. Permite la creación de *webapps* rápidamente usando sus componentes y elementos de organización.

Los elementos de organización tales como los menús laterales, las pestañas y los contenedores, permiten alojar y organizar los componentes. Está organización es esencial para la estética y usabilidad de la aplicación web.

Los componentes, son elementos preprogramados que permiten al desarrollador abstraerse y no tener que programar cada una de las funcionalidades básicas de una página web. Algunos de los ejemplos de estos componentes son los botones, los selectores de fecha por calendario, los dataframes entre otros.

Y por supuesto, es open source y distribuido a través de la licencia Apache 2.0 . Ha sido usado por organizaciones tales como Uber, Spotify, NASA y la OMS desde el año 2018. Entre sus ventajas está la Streamlit Community Cloud que permite alojar de forma gratuita las webapps que se crean.

### 3.4 Scrapy y GeoPy

Como no puede ser de otra manera, la librería que usaremos para hacer *web scrapping* será también open source. En concreto usaremos Scrapy. Está es una librería muy capaz y extensa para crear scripts de extracción de datos web. FreeCodeCamp tiene un curso de más de 4 horas sobre esta herramienta:

https://www.youtube.com/watch?v=mBoX\_JCKZTE

GeoPy, es una sencilla librería para la obtención de datos globales a través de su API gratuita. GeoPy obtiene estos datos de fuentes fiables tales como Google Maps, Open StreeMap y ArcGIS.

Ya que wikipedia no contiene ciertos datos biográficos de los pilotos, y en ocasiones carece de datos geográficos de los circuitos, deberemos usar *web scrapping* y GeoPy para obtenerlos.

## 4. Arquitectura del Proyecto

En la siguiente sección comentaremos la estructura y procesos que componen la solución planteada para este proyecto.

### 4.1 Resumen de la solución

Como comentamos anteriormente, los orígenes de datos que hemos seleccionado para esta solución son las páginas de cada temporada de MotoGP en la versión inglesa de la Wikipedia. Esta elección se debe a la sencillez que ofrece Excel para la extracción de datos. A través de Excel cualquier usuario, independientemente de su experiencia técnica, puede extraer todas las tablas que requiera para extraer conocimiento.

A pesar de su sencillez, estos datos deben de ser preprocesados para darles una forma adecuada y consistente entre todas las temporadas para que el proceso ETL pueda trabajar todas las temporadas de forma automática.

Una vez se cree cada dimensión y hecho, se subirán estos datos a una instancia de PostgreSQL. Esta puede ser tanto local como en la nube.

Una vez creado el modelo analítico, este puede ser utilizado en una de las innumerables herramientas de visualización de datos como son PowerBI o Quick Sight. En nuestro caso usaremos Streamlit debido a su carácter open source y gratuito.



Figura 11 – Diagrama del proyecto

### 4.2 Data Warehouse

Un data warehouse permite almacenar los datos en un modelo analítico. Este tipo de modelos es diferente de los transaccionales que usamos en la carrera. Se prioriza el almacenamiento histórico de datos y las lecturas rápidas a través de la desnormalización de datos. Todo esto nos permitirá crear medidas y gráficos para visualizar mejor dichos datos.

Las tablas de dimensiones son el “Qué?” del modelo, y nos permiten conocer los atributos y características de los elementos que forman la tabla de hechos. Las tablas de hechos contienen dos tipos de columnas: las claves que las relacionan a las dimensiones y las que contienen los hechos.

El modelo de tipo copo de nieve que hemos creado en el Data Warehouse y que poblaremos con los datos transformados es el siguiente:

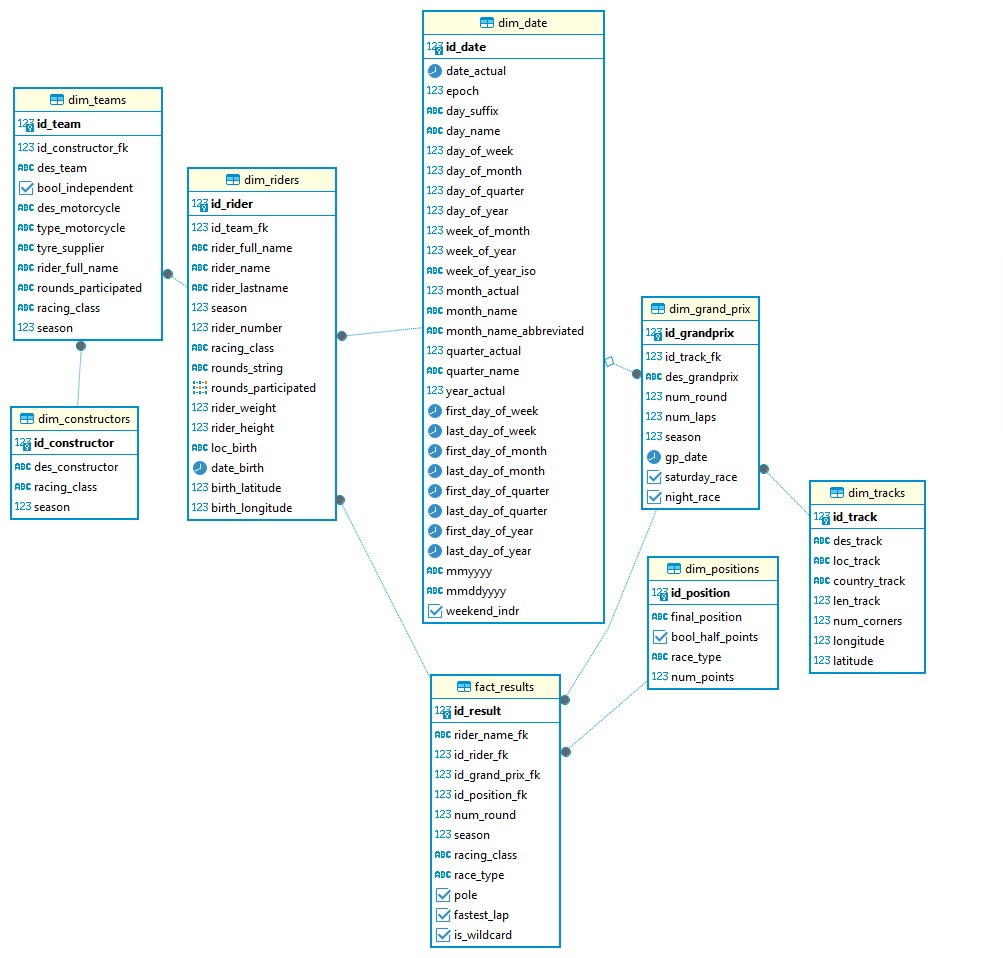


Figura 12 – Diagrama del modelo atalítico

#### 4.2.1 Dimensión Circuitos - ‘dim\_tracks’

Esta dimensión nos permite saber más sobre el circuito donde se celebró el gran premio como su localización o longitud. Esta tabla tiene como clave primaria el campo “id\_track” de tipo Serial.

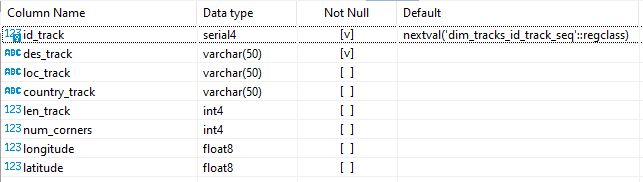
Estos son los campos y sus tipos:  
 

Figura 13 – Columnas y tipos de datos de dim\_tracks

#### 4.2.2 Dimensión Grandes Premios - ‘dim\_grand\_prix’

Esta dimensión nos permite informarnos sobre las características del gran premio. Podremos saber su nombre y por tanto el patrocinador principal, que orden ocupó en una temporada y la temporada en sí. La clave principal de esta tabla es el campo “id\_grandprix”. El campo “id\_track\_fk” sirve para poder relacionar el gran premio con el circuito en el que se celebró

Estos son los campos y sus tipos:

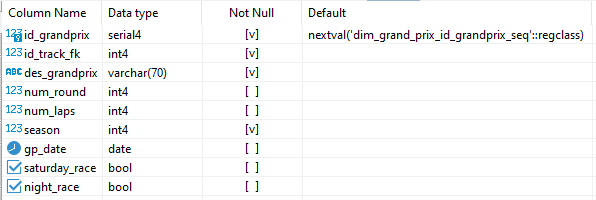


Figura 14 – Columnas y tipos de datos de dim\_grand\_prix

#### 4.2.3 Dimensión Constructores - ‘dim\_constructors’

La dimensión “dim\_constructor” nos permite saber que constructores participaron en cada categoría desde la temporada 2002. La clave principal de esta tabla es “id\_constructor”.

Estos son los campos y sus tipos:

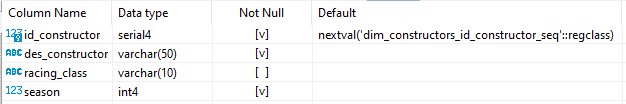


Figura 15 – Columnas y tipos de datos de dim\_constructors

#### 4.2.4 Dimensión Equipos - ‘dim\_teams’

Esta dimensión tiene como objetivo agrupar los equipos que compitieron en cada temporada de los mundiales en todas las categorías. Permite también saber qué constructor les suministraba gracias a la columna “id\_constructor\_fk”. Además de saber que suministrador de neumáticos tenían contratado y con qué moto corrían exactamente.

Estos son los campos y sus tipos:

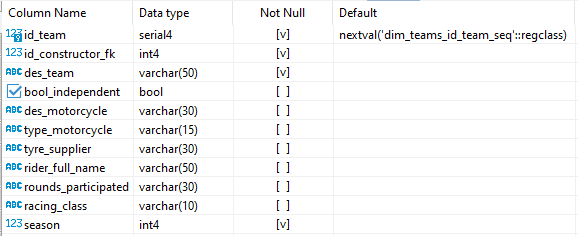


Figura 16 – Columnas y tipos de datos de dim\_constructors

#### 4.2.5 Dimensión Pilotos - ‘dim\_riders’

Esta importante dimensión permite almacenar información sobre cada piloto como su nombre, apellido, su número y en qué equipo estaba en una temporada a través del campo “id\_team\_fk”. Su clave principal es “id\_rider”.

Estos son los campos y sus tipos:

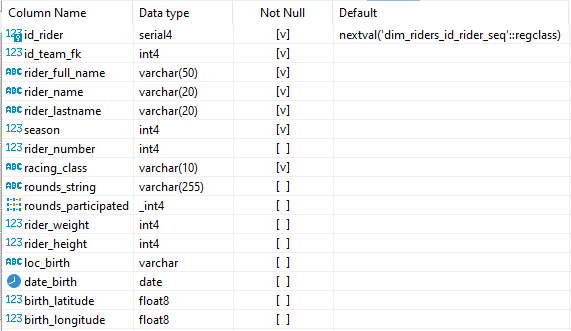


Figura 17 – Columnas y tipos de datos de dim\_riders

#### 4.2.6 Dimensión Posiciones- ‘dim\_positions’

Esta dimensión permite saber cuántos puntos consiguió un piloto tras una carrera, ya sea la carrera principal o una carrera sprint. Su clave principal es “id\_position”.

Estos son los campos y sus tipos:

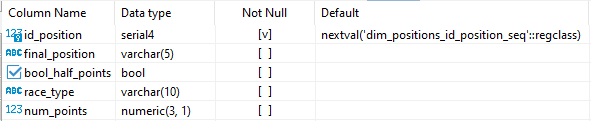


Figura 18 – Columnas y tipos de datos de dim\_positions

#### 4.2.7 Dimensión Fechas- ‘dim\_date’

Este tipo de dimensión permite extraer más información a partir de una fecha, como el trimestre, el día de la semana, la semana del año entre muchas otras.Esto nos permitirá contextualizar mucho mejor las fechas de los grandes premios. Su clave principal es “id\_date”.

Estos son los campos y sus tipos:

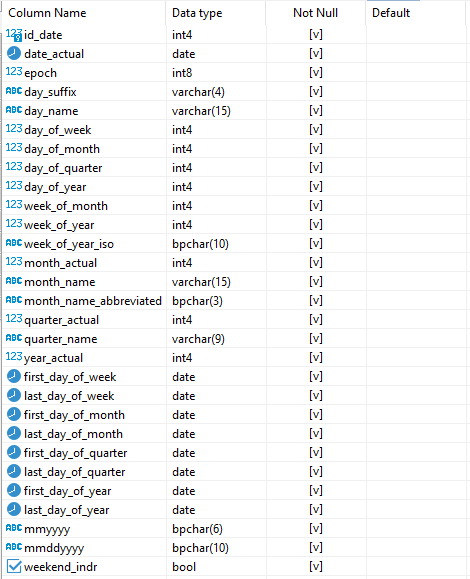


Figura 19 – Columnas y tipos de datos de dim\_positions

#### 4.2.8 Tabla de Hechos Resultados- ‘fact\_results’

Esta tabla contiene los resultados de cada piloto, por categoría, grán premio y temporada; además de si salió en “pole” o si realizó la vuelta más rápida en la carrera principal. Mediante el uso de claves foráneas se enlaza con las tablas de dimensión dim\_riders, dim\_grand\_prix y dim\_positions. Su clave principal es “id\_results”.

Estos son los campos y sus tipos:

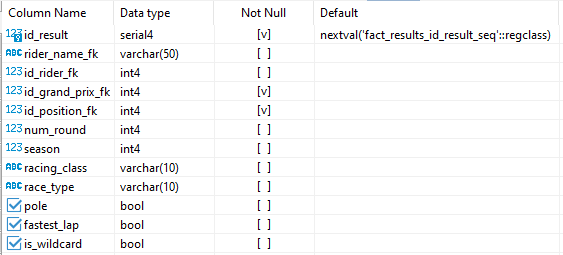


Figura 20 – Columnas y tipos de datos de fact\_results

#### 4.2.9 Script de creación del Datawarehouse

-- public.dim\_date definition

-- Drop table

-- DROP TABLE public.dim\_date;

CREATE TABLE public.dim\_date (

id\_date int4 NOT NULL,

date\_actual date NOT NULL,

epoch int8 NOT NULL,

day\_suffix varchar(4) NOT NULL,

day\_name varchar(15) NOT NULL,

day\_of\_week int4 NOT NULL,

day\_of\_month int4 NOT NULL,

day\_of\_quarter int4 NOT NULL,

day\_of\_year int4 NOT NULL,

week\_of\_month int4 NOT NULL,

week\_of\_year int4 NOT NULL,

week\_of\_year\_iso bpchar(10) NOT NULL,

month\_actual int4 NOT NULL,

month\_name varchar(15) NOT NULL,

month\_name\_abbreviated bpchar(3) NOT NULL,

quarter\_actual int4 NOT NULL,

quarter\_name varchar(9) NOT NULL,

year\_actual int4 NOT NULL,

first\_day\_of\_week date NOT NULL,

last\_day\_of\_week date NOT NULL,

first\_day\_of\_month date NOT NULL,

last\_day\_of\_month date NOT NULL,

first\_day\_of\_quarter date NOT NULL,

last\_day\_of\_quarter date NOT NULL,

first\_day\_of\_year date NOT NULL,

last\_day\_of\_year date NOT NULL,

mmyyyy bpchar(6) NOT NULL,

mmddyyyy bpchar(10) NOT NULL,

weekend\_indr bool NOT NULL,

CONSTRAINT dim\_date\_id\_date\_pk PRIMARY KEY (id\_date)

);

CREATE INDEX dim\_date\_date\_actual\_idx ON public.dim\_date USING btree (date\_actual);

DROP TABLE IF EXISTS dim\_constructors CASCADE;

CREATE TABLE IF NOT EXISTS dim\_constructors (

id\_constructor SERIAL PRIMARY KEY,

des\_constructor VARCHAR(50) NOT NULL,

racing\_class varchar(10),

season integer NOT NULL

);

/\*------------------------------------------------------------------------\*/

DROP TABLE IF EXISTS dim\_teams CASCADE;

create table IF NOT EXISTS dim\_teams (

id\_team SERIAL PRIMARY KEY,

id\_constructor\_fk integer not null,

des\_team varchar(50) not null,

bool\_independent boolean,

des\_motorcycle varchar(30),

type\_motorcycle varchar(15),

tyre\_supplier varchar(30),

rider\_full\_name varchar(50),

rounds\_participated varchar(30),

racing\_class varchar(10),

season integer not null,

constraint fk\_id\_constructor

foreign key(id\_constructor\_fk)

references dim\_constructors(id\_constructor)

ON DELETE SET NULL

);

/\*------------------------------------------------------------------------\*/

DROP TABLE IF EXISTS dim\_tracks CASCADE;

CREATE TABLE if not exists dim\_tracks (

id\_track serial primary key,

des\_track varchar(50) NOT NULL,

loc\_track varchar(50) NULL,

country\_track varchar(50) NULL,

len\_track int4 NULL,

num\_corners int4 NULL,

longitude float8 NULL,

latitude float8 NULL

);

/\*------------------------------------------------------------------------\*/

DROP TABLE IF EXISTS dim\_grand\_prix CASCADE;

create table IF NOT EXISTS dim\_grand\_prix(

id\_grandprix serial primary key,

id\_track\_fk integer not null,

des\_grandprix varchar(70) not null,

num\_round integer,

num\_laps integer,

season integer not null,

gp\_date date,

saturday\_race boolean,

night\_race boolean,

constraint fk\_id\_track

foreign key(id\_track\_fk)

references dim\_tracks(id\_track)

ON DELETE SET NULL

);

/\*------------------------------------------------------------------------\*/

DROP TABLE IF EXISTS dim\_riders CASCADE;

CREATE table IF NOT EXISTS public.dim\_riders (

id\_rider serial primary key,

id\_team\_fk int4 NOT NULL,

rider\_full\_name varchar(50) NOT NULL,

rider\_name varchar(20) NOT NULL,

rider\_lastname varchar(20) NOT NULL,

season int4 NOT NULL,

rider\_number int4 NULL,

racing\_class varchar(10) NOT NULL,

rounds\_string varchar(255),

rounds\_participated integer[] NULL,

rider\_weight int4,

rider\_height int4,

loc\_birth varchar NULL,

date\_birth date NULL,

birth\_latitude float8,

birth\_longitude float8

);

/\*------------------------------------------------------------------------\*/

DROP TABLE IF EXISTS fact\_results CASCADE;

create table IF NOT EXISTS fact\_results(

id\_result serial primary key,

rider\_name\_fk varchar(50),

id\_rider\_fk integer ,

id\_grand\_prix\_fk integer not null,

id\_position\_fk integer not null,

num\_round integer,

season integer,

racing\_class varchar(10),

race\_type varchar(10),

pole bool,

fastest\_lap bool,

is\_wildcard bool,

constraint fk\_id\_grand\_prix

foreign key(id\_grand\_prix\_fk)

references dim\_grand\_prix(id\_grandprix)

ON DELETE SET null,

constraint fk\_id\_position

foreign key(id\_position\_fk)

references dim\_positions(id\_position)

ON DELETE SET NULL

);

/\*------------------------------------------------------------------------\*/

**truncate** **table** dim\_positions

**insert** **into** dim\_positions(final\_position,race\_type,bool\_half\_points,num\_points)

**values** (**'1'**,**'main'**,**FALSE**,25),(**'2'**,**'main'**,**FALSE**,20),(**'3'**,**'main'**,**FALSE**,16),(**'4'**,**'main'**,**FALSE**,13),(**'5'**,**'main'**,**FALSE**,11),(**'6'**,**'main'**,**FALSE**,10),

(**'7'**,**'main'**,**FALSE**,9),(**'8'**,**'main'**,**FALSE**,8),(**'9'**,**'main'**,**FALSE**,7),

(**'10'**,**'main'**,**FALSE**,6),(**'11'**,**'main'**,**FALSE**,5),(**'12'**,**'main'**,**FALSE**,4),(**'13'**,**'main'**,**FALSE**,3),(**'14'**,**'main'**,**FALSE**,2),(**'15'**,**'main'**,**FALSE**,1),

(**'16'**,**'main'**,**FALSE**,0),(**'17'**,**'main'**,**FALSE**,0),(**'18'**,**'main'**,**FALSE**,0),(**'19'**,**'main'**,**FALSE**,0),(**'20'**,**'main'**,**FALSE**,0),(**'21'**,**'main'**,**FALSE**,0),

(**'22'**,**'main'**,**FALSE**,0),(**'23'**,**'main'**,**FALSE**,0),(**'24'**,**'main'**,**FALSE**,0),(**'25'**,**'main'**,**FALSE**,0),(**'26'**,**'main'**,**FALSE**,0),(**'27'**,**'main'**,**FALSE**,0),

(**'DNF'**,**'main'**,**FALSE**,0), (**'DNS'**,**'main'**,**FALSE**,0), (**'DSQ'**,**'main'**,**FALSE**,0),

(**'WD'**,**'main'**,**FALSE**,0), (**'NC'**,**'main'**,**FALSE**,0), (**'DNPQ'**,**'main'**,**FALSE**,0),

(**'DNP'**,**'main'**,**FALSE**,0), (**'DNA'**,**'main'**,**FALSE**,0), (**'EX'**,**'main'**,**FALSE**,0), (**'Ret'**,**'main'**,**FALSE**,0),(**'null'**,**'main'**,**FALSE**,0), (**'C'**,**'main'**,**FALSE**,0),

(**'WLD'**,**'main'**,**FALSE**,0),(**'WLD'**,**'sprint'**,**FALSE**,0),(**'WLD'**,**'main'**,**TRUE**,0),(**'WLD'**,**'sprint'**,**TRUE**,0),

(**'1'**,**'sprint'**,**FALSE**,12),(**'2'**,**'sprint'**,**FALSE**,9),(**'3'**,**'sprint'**,**FALSE**,7),(**'4'**,**'sprint'**,**FALSE**,6),(**'5'**,**'sprint'**,**FALSE**,5),(**'6'**,**'sprint'**,**FALSE**,4),

(**'7'**,**'sprint'**,**FALSE**,3),(**'8'**,**'sprint'**,**FALSE**,2),(**'9'**,**'sprint'**,**FALSE**,1),

(**'10'**,**'sprint'**,**FALSE**,0),(**'11'**,**'sprint'**,**FALSE**,0),(**'12'**,**'sprint'**,**FALSE**,0),(**'13'**,**'sprint'**,**FALSE**,0),(**'14'**,**'sprint'**,**FALSE**,0),(**'15'**,**'sprint'**,**FALSE**,0),

(**'16'**,**'sprint'**,**FALSE**,0),(**'17'**,**'sprint'**,**FALSE**,0),(**'18'**,**'sprint'**,**FALSE**,0),(**'19'**,**'sprint'**,**FALSE**,0),(**'20'**,**'sprint'**,**FALSE**,0),(**'21'**,**'sprint'**,**FALSE**,0),

(**'22'**,**'sprint'**,**FALSE**,0),(**'23'**,**'sprint'**,**FALSE**,0),(**'24'**,**'sprint'**,**FALSE**,0),(**'25'**,**'sprint'**,**FALSE**,0),(**'26'**,**'sprint'**,**FALSE**,0),(**'27'**,**'sprint'**,**FALSE**,0),

(**'DNF'**,**'sprint'**,**FALSE**,0), (**'DNS'**,**'sprint'**,**FALSE**,0), (**'DSQ'**,**'sprint'**,**FALSE**,0),

(**'WD'**,**'sprint'**,**FALSE**,0), (**'NC'**,**'sprint'**,**FALSE**,0), (**'DNPQ'**,**'sprint'**,**FALSE**,0),

(**'DNP'**,**'sprint'**,**FALSE**,0), (**'DNA'**,**'sprint'**,**FALSE**,0), (**'EX'**,**'sprint'**,**FALSE**,0), (**'Ret'**,**'sprint'**,**FALSE**,0),(**'null'**,**'sprint'**,**FALSE**,0), (**'C'**,**'sprint'**,**FALSE**,0),

(**'1'**,**'main'**,**TRUE**,12.5),(**'2'**,**'main'**,**TRUE**,10),(**'3'**,**'main'**,**TRUE**,8),(**'4'**,**'main'**,**TRUE**,6.5),(**'5'**,**'main'**,**TRUE**,5.5),(**'6'**,**'main'**,**TRUE**,5),

(**'7'**,**'main'**,**TRUE**,4.5),(**'8'**,**'main'**,**TRUE**,4),(**'9'**,**'main'**,**TRUE**,3.5),

(**'10'**,**'main'**,**TRUE**,3),(**'11'**,**'main'**,**TRUE**,2.5),(**'12'**,**'main'**,**TRUE**,2),(**'13'**,**'main'**,**TRUE**,1.5),(**'14'**,**'main'**,**TRUE**,1),(**'15'**,**'main'**,**TRUE**,0.5),

(**'16'**,**'main'**,**TRUE**,0),(**'17'**,**'main'**,**TRUE**,0),(**'18'**,**'main'**,**TRUE**,0),(**'19'**,**'main'**,**TRUE**,0),(**'20'**,**'main'**,**TRUE**,0),(**'21'**,**'main'**,**TRUE**,0),

(**'22'**,**'main'**,**TRUE**,0),(**'23'**,**'main'**,**TRUE**,0),(**'24'**,**'main'**,**TRUE**,0),(**'25'**,**'main'**,**TRUE**,0),(**'26'**,**'main'**,**TRUE**,0),(**'27'**,**'main'**,**TRUE**,0),

(**'DNF'**,**'main'**,**TRUE**,0), (**'DNS'**,**'main'**,**TRUE**,0), (**'DSQ'**,**'main'**,**TRUE**,0),

(**'WD'**,**'main'**,**TRUE**,0), (**'NC'**,**'main'**,**TRUE**,0), (**'DNPQ'**,**'main'**,**TRUE**,0),

(**'DNP'**,**'main'**,**TRUE**,0), (**'DNA'**,**'main'**,**TRUE**,0), (**'EX'**,**'main'**,**TRUE**,0), (**'Ret'**,**'main'**,**TRUE**,0),(**'null'**,**'main'**,**TRUE**,0), (**'C'**,**'main'**,**TRUE**,0),

(**'1'**,**'sprint'**,**TRUE**,6),(**'2'**,**'sprint'**,**TRUE**,4.5),(**'3'**,**'sprint'**,**TRUE**,3.5),(**'4'**,**'sprint'**,**TRUE**,3),(**'5'**,**'sprint'**,**TRUE**,2.5),(**'6'**,**'sprint'**,**TRUE**,2),

(**'7'**,**'sprint'**,**TRUE**,1.5),(**'8'**,**'sprint'**,**TRUE**,1),(**'9'**,**'sprint'**,**TRUE**,0.5),

(**'10'**,**'sprint'**,**TRUE**,0),(**'11'**,**'sprint'**,**TRUE**,0),(**'12'**,**'sprint'**,**TRUE**,0),(**'13'**,**'sprint'**,**TRUE**,0),(**'14'**,**'sprint'**,**TRUE**,0),(**'15'**,**'sprint'**,**TRUE**,0),

(**'16'**,**'sprint'**,**TRUE**,0),(**'17'**,**'sprint'**,**TRUE**,0),(**'18'**,**'sprint'**,**TRUE**,0),(**'19'**,**'sprint'**,**TRUE**,0),(**'20'**,**'sprint'**,**TRUE**,0),(**'21'**,**'sprint'**,**TRUE**,0),

(**'22'**,**'sprint'**,**TRUE**,0),(**'23'**,**'sprint'**,**TRUE**,0),(**'24'**,**'sprint'**,**TRUE**,0),(**'25'**,**'sprint'**,**TRUE**,0),(**'26'**,**'sprint'**,**TRUE**,0),(**'27'**,**'sprint'**,**TRUE**,0),

(**'DNF'**,**'sprint'**,**TRUE**,0), (**'DNS'**,**'sprint'**,**TRUE**,0), (**'DSQ'**,**'sprint'**,**TRUE**,0),

(**'WD'**,**'sprint'**,**TRUE**,0), (**'NC'**,**'sprint'**,**TRUE**,0), (**'DNPQ'**,**'sprint'**,**TRUE**,0),

(**'DNP'**,**'sprint'**,**TRUE**,0), (**'DNA'**,**'sprint'**,**TRUE**,0), (**'EX'**,**'sprint'**,**TRUE**,0), (**'Ret'**,**'sprint'**,**TRUE**,0),(**'null'**,**'sprint'**,**TRUE**,0), (**'C'**,**'sprint'**,**TRUE**,0),

(**'WLD'**,**'main'**,**FALSE**,0),(**'WLD'**,**'sprint'**,**FALSE**,0),(**'WLD'**,**'main'**,**TRUE**,0),(**'WLD'**,**'sprint'**,**TRUE**,0);

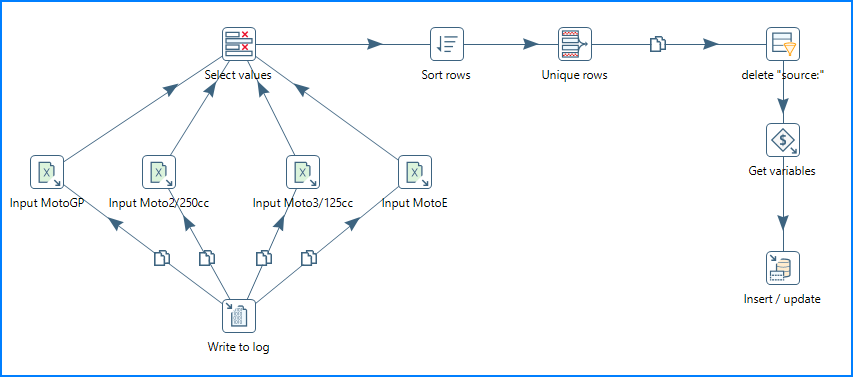
### 4.3 Implementacion de la ETL en Pentaho

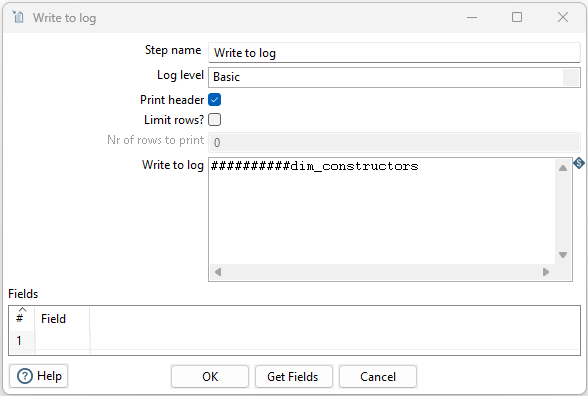
En este apartado mostraremos en detalle el funcionamiento y la lógica detrás de cada uno de los procesos creados en Pentaho para poder transformar el conjunto de datos de forma automática.

Además, servirá para poder entender mejor el modo de uso de la herramienta Pentaho Data Integration, cómo usar los trabajos y las transformaciones, y qué pasos (steps) existen en el programa.

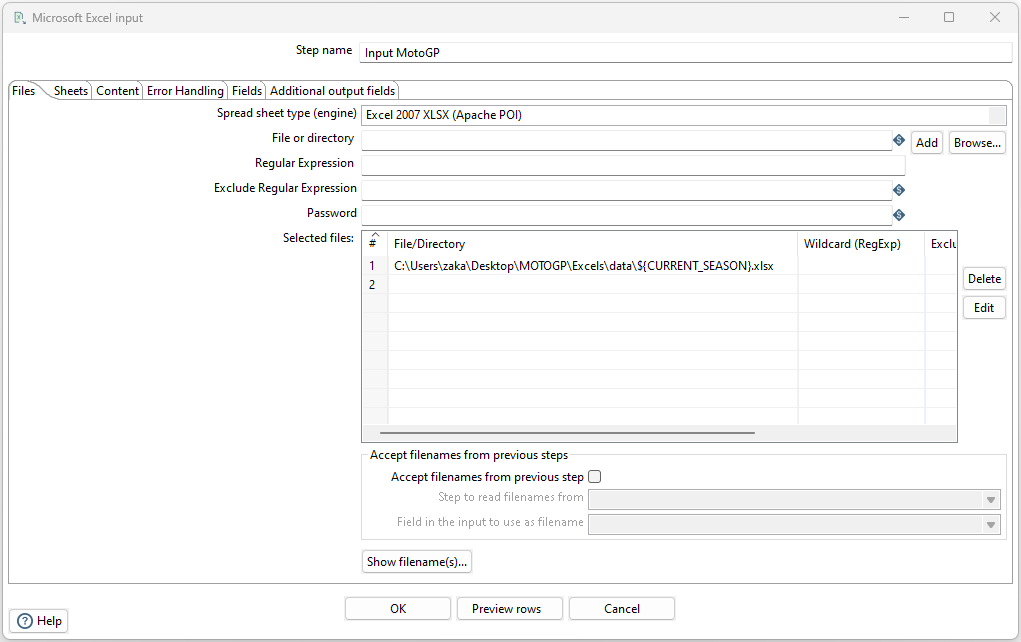
#### 4.3.1 Dimensión Constructor

La transformación para cargar los datos de los constructores comienza por escribir en los logs de PDI que esta transformación comienza. De esta forma, es más sencillo hacer un seguimiento de la ejecución del programa e identificar errores.

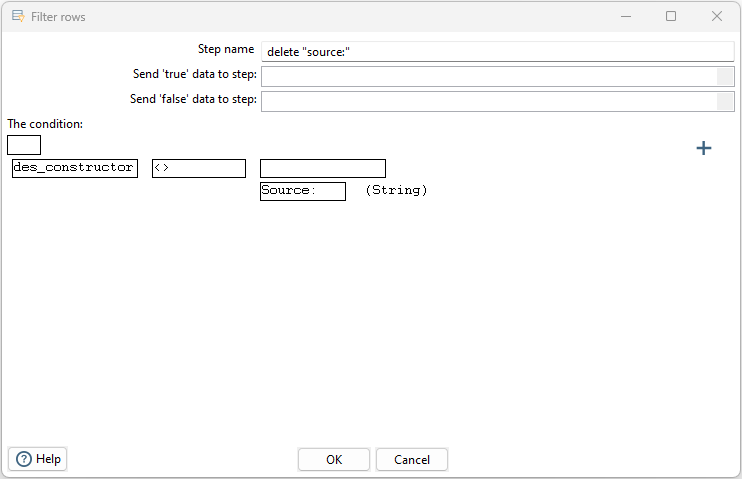




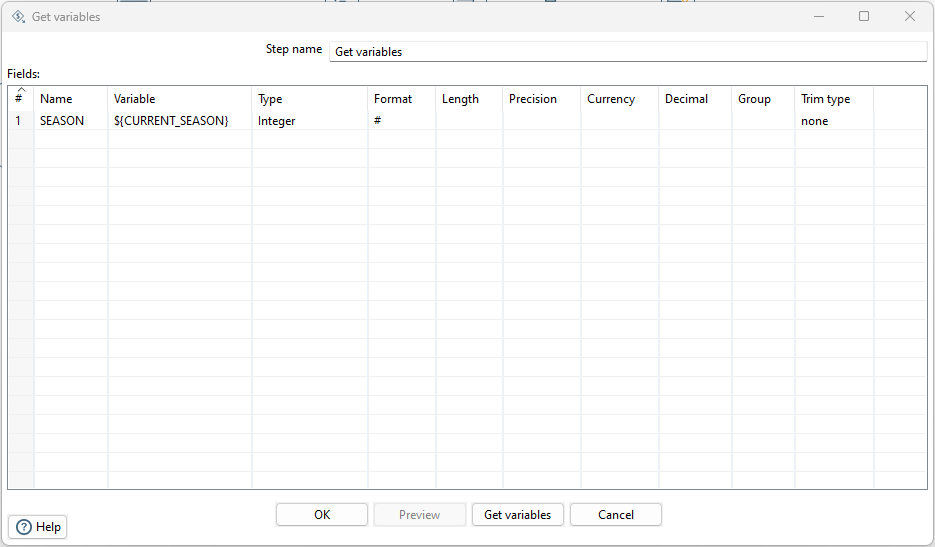
Tras ello, se cargan las hojas de excel para cada categoría y temporada usando la variable “${CURRENT\_SEASON}”, esto se explicará más adelante en el Trabajo Principal. Una vez se tienen todos los datos, se conservan únicamente los relativos al nombre del equipo y su categoría asegurando que no haya duplicados ni datos erróneos.



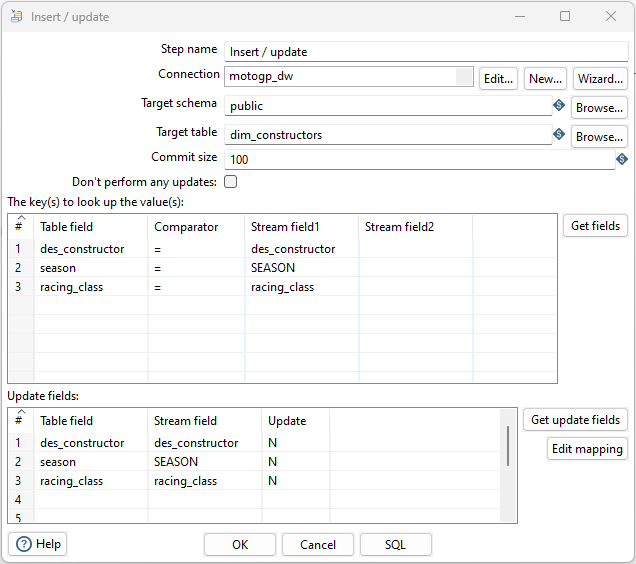
En este caso, una de las filas que se comprueba que no haya son las que contienen “source:”. Estas filas surgen como consecuencia de la importación de las tablas desde Wikipedia. Se eliminan a mano desde Excel, pero en el caso de que no se hayan eliminado, PDI se ocupa de eliminarlos.



En el siguiente paso, se recupera la variable anteriormente mencionada “${CURRENT\_SEASON}” que contiene el valor de la temporada que se está procesando. Este es el último dato que se necesita para poder subir los datos al Data Warehouse.

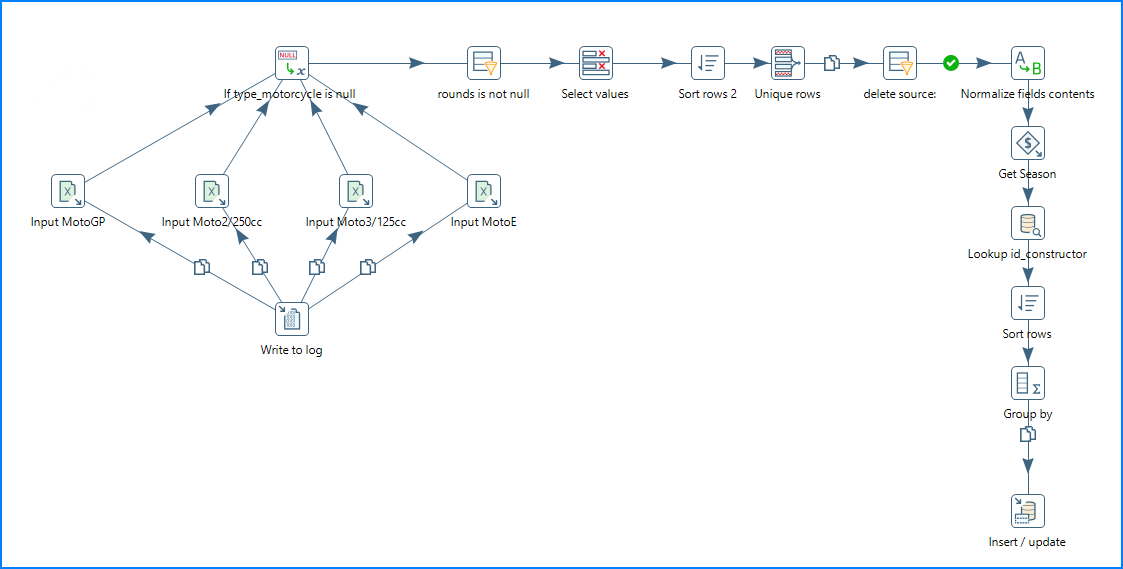


Usando un paso del tipo “Insert / Update”, subimos los datos de los constructores al Data Warehouse en la dimensión “dim\_constructors”. Hacemos esto sin actualizar ninguno de los datos que haya en la dimensión, ya que la combinación de nombre, categoría y temporada es siempre única e irrepetible.



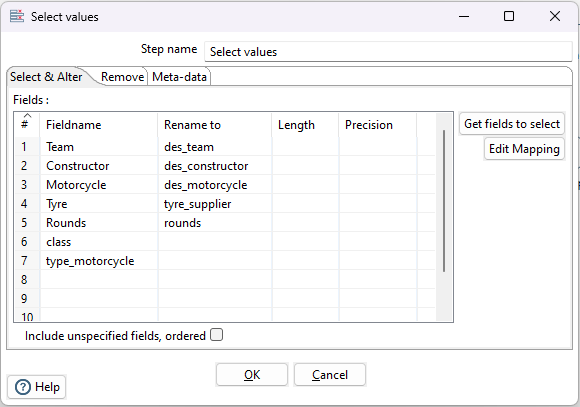
#### 4.3.2 Dimensión Equipos

La transformación para cargar los datos de los equipos comienza por escribir en los logs de PDI que esta transformación comienza. De esta forma, es más sencillo hacer un seguimiento de la ejecución del programa e identificar errores. Tras ello se añaden las hojas relativas a los equipos por cada categoría y temporada.

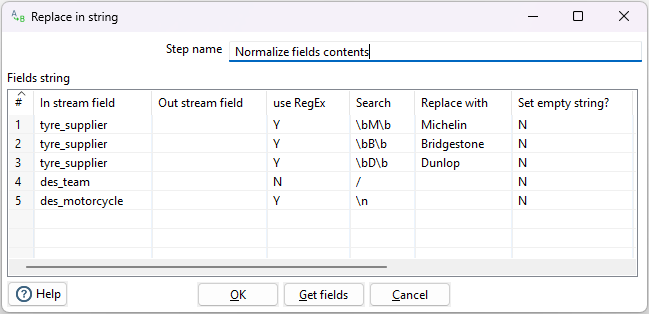


Una vez se tienen todos los datos, se comprueba si estos tienen poblada la columna “type\_motorcycle” (tipo de moto), en el caso de que esté vacía, se les da el valor “Prototype” por defecto. Esta lógica está aquí ya que en la categoría de MotoGP entre los años 2012 y 2015 el campo “type\_motorcycle” podría ser “Prototype”, “CRT” u “Open”. En el resto de temporadas y categorías esta columna no existe en los datos ya que siempre son “Prototype”.

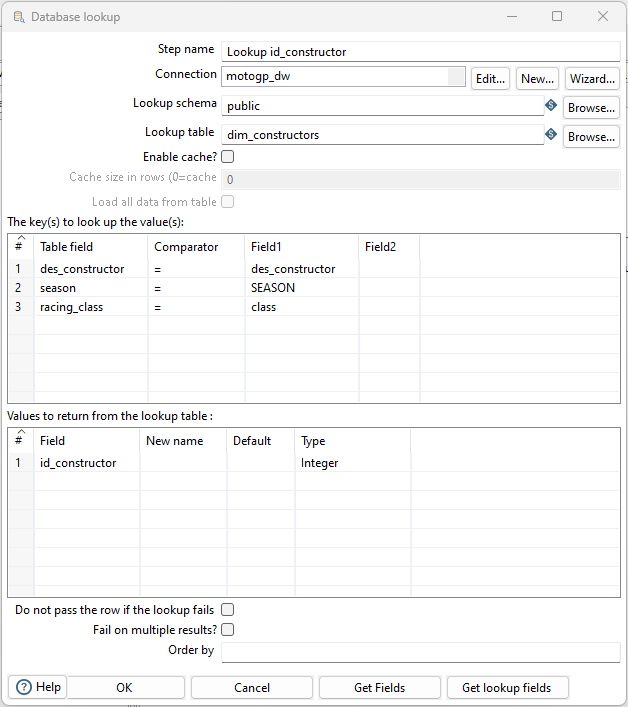
Se comprueba que el número de rondas no esté vacío ya que esto no tendría sentido y el dato sería incorrecto. Procedemos a seleccionar y renombrar únicamente los campos que nos interesan en el paso “Select Values”. Una vez hecho esto, ordenamos para eliminar las filas duplicadas en los datos. En equipos también podemos encontrar las filas que contienen “source:” como en constructores, por lo tanto las eliminamos también.



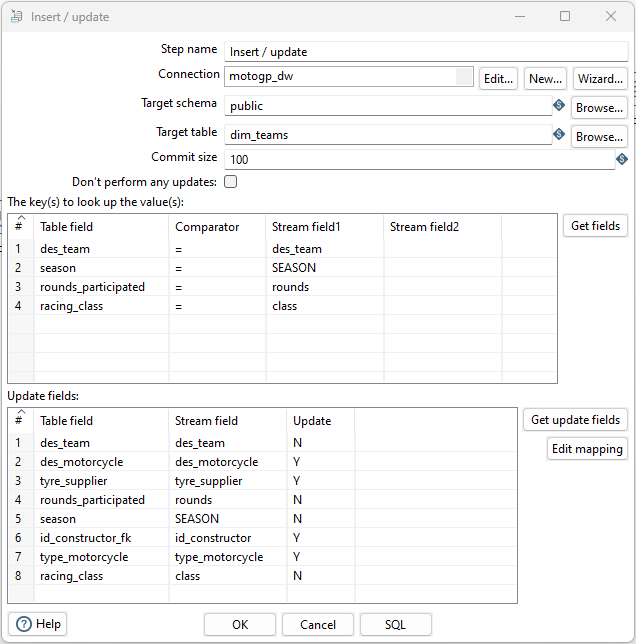
Una vez hemos seleccionado únicamente las filas que nos interesan, procedemos a limpiar los datos de los campos que tenemos. En los casos 1,2 y 3 lo que hacemos es sustituir la inicial del suministrador de neumáticos por su nombre completo. En el paso 4 eliminamos los “/” de los nombres de los equipos. Esto puede producirse ya que hay equipos con más de un país, así que en las tablas de la Wikipedia se separan las banderas de esos países usando “/”, y cuando se pasan a Excel estas “/” pueden mantenerse en el nombre. Por último se eliminan los posibles saltos de línea que pueda haber en los nombres de las motos que emplean los equipos.



Tras esto, recuperamos la variable que indica la temporada de los datos que estamos procesando igual que en constructores. Debemos también hacer un lookup a la tabla “dim\_construcotres” para recuperar el “id\_constructor” del constructor que suministra a cada equipo en cada temporada. Esto lo hacemos con el paso “Database Lookup”.

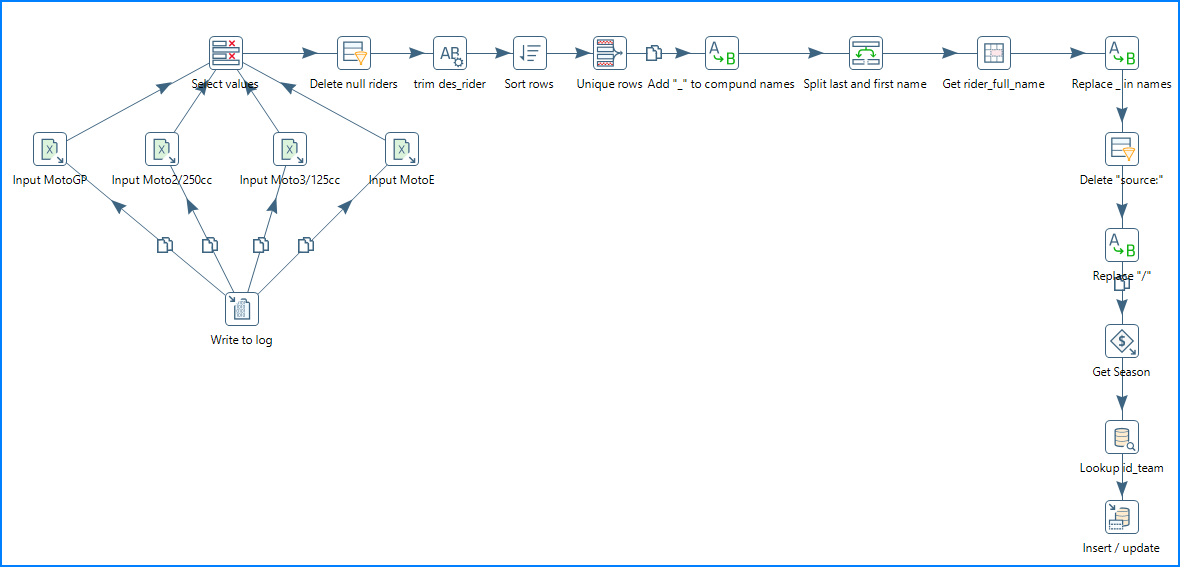


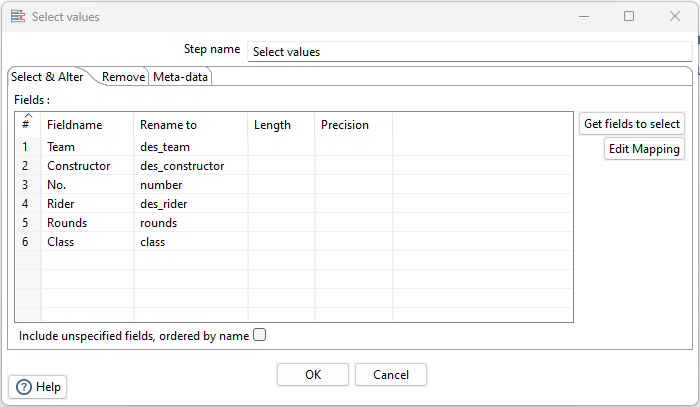
Acto seguido, agrupamos los datos por nombre de equipo e identificador de constructor. Para ello debemos ordenar primero los datos para poder realizar el agrupado de forma correcta. Finalmente cargamos los datos usando un paso del tipo “Insert Update”. En este podremos establecer los elementos que hacen a un registro único evitando que haya dos equipos con el mismo nombre, categoría, temporada y rondas competidas; pudiendo actualizar el resto de campos.



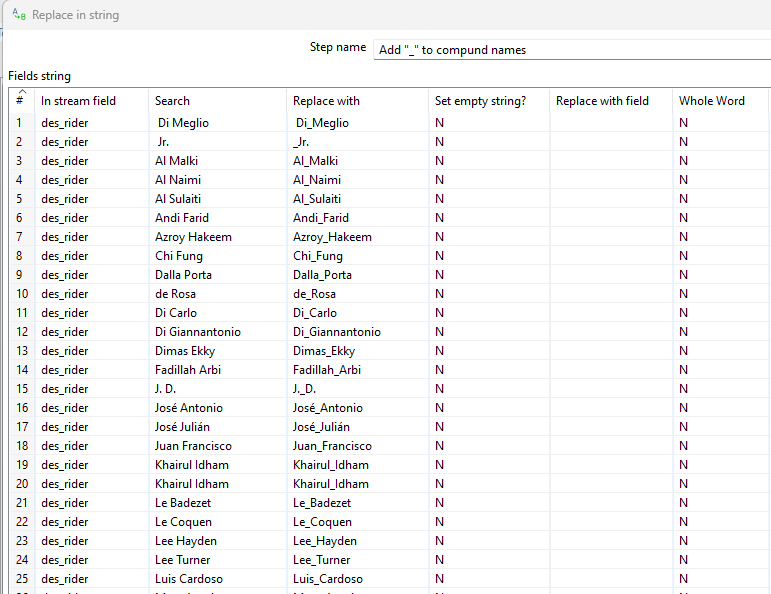
#### 4.3.3 Dimensión Pilotos

La transformación para cargar los datos de los pilotos comienza como las anteriores escribiendo en los logs de PDI que esta transformación comienza. Tras ello se añaden las hojas relativas a los equipos por cada categoría y temporada. Una vez tenemos los datos, seleccionamos los campos que nos interesa conservar y renombramos algunos de ellos. Acto seguido se eliminan los campos con los pilotos sin nombre si los hubiera, y se eliminan los espacios que tengan esos nombres al principio o al final. Como es costumbre, se ordena para eliminar las posibles filas duplicadas.



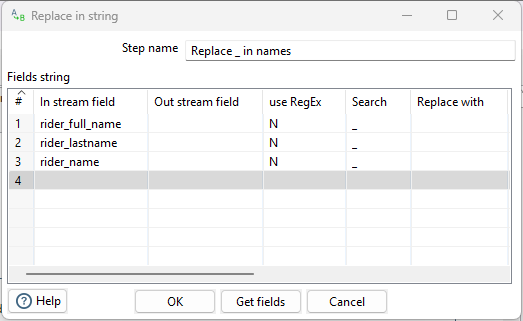


En este conjunto de datos existe una particularidad que no se ve en los otros datos. Al operar con nombres de personas, muchas veces estos son compuestos o se componen de más de una palabra. Por ejemplo el nombre “José Antonio” o el apellido “Di Giannantonio”. Debido a que no existe forma de identificar automáticamente que dos o más palabras forman parte de un nombre o de un apellido, estas separaciones deben hacerse a mano. Los siguientes son algunos de los nombres que se identifican y se les añade una barra baja para ayudar a la separación de nombres y apellidos:

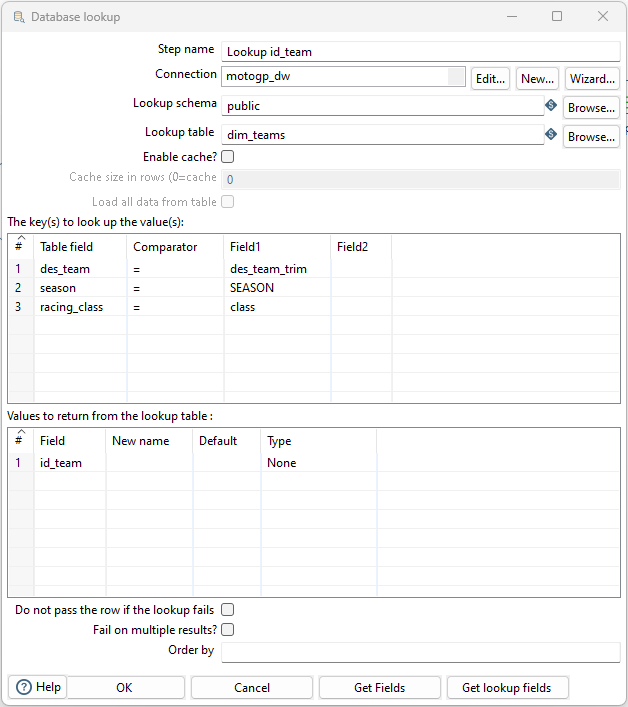


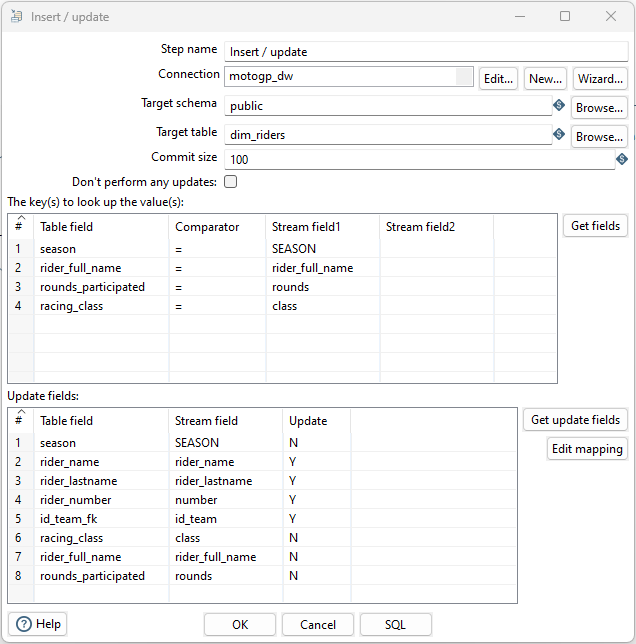
Una vez los todos los nombres y/o apellidos que tenga un piloto estén separados por otro caracter diferente al espacio, se procede a dividir el campo por el primer espacio que se encuentre. De esta forma obtenemos un campo para los nombres y otro para los apellidos. Tras esta operación, debemos volver a crear el campo nombre completo ya que en la división se elimina automáticamente.

Una vez separados estos campos, procedemos a reemplazar la barra baja por un espacio en esos campos, de esta forma tenemos los nombres correctamente otra vez.



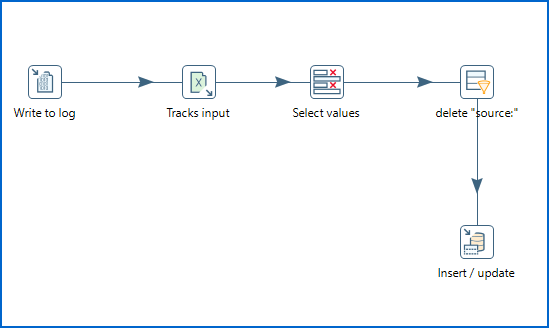
Como en los anteriores, se eliminan los registros que contienen “source:” si los hubiera, se elimina “/” del nombre de los equipos y se recupera la temporada que se está procesando. Este último lo usaremos para no solo hacer el lookup a la dimensión equipo, sino que también lo subiremos a la dimensión de pilotos.



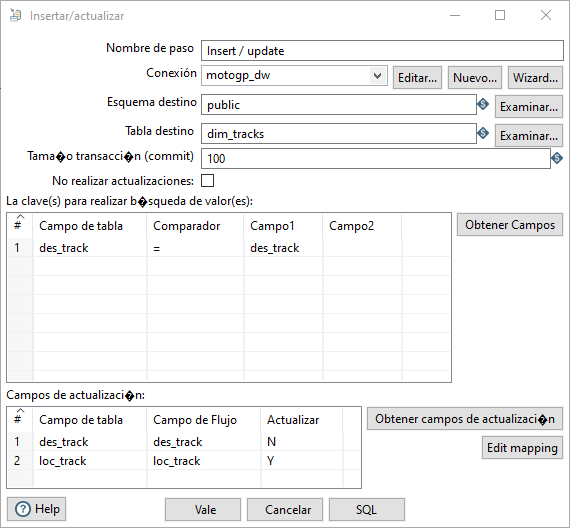


#### 4.3.4 Dimensión Circuitos

La transformación para cargar los datos de los circuitos en los que se compite. Está transformacion es relativamente simple. Se cargan los datos de la hoja de Excel “Calendar” de la temporada que se está procesando, se seleccionan los campos relativos al circuito, se eliminan las filas que contengan “source:” si las hubiera y se cargan los datos en “dim\_tracks”.

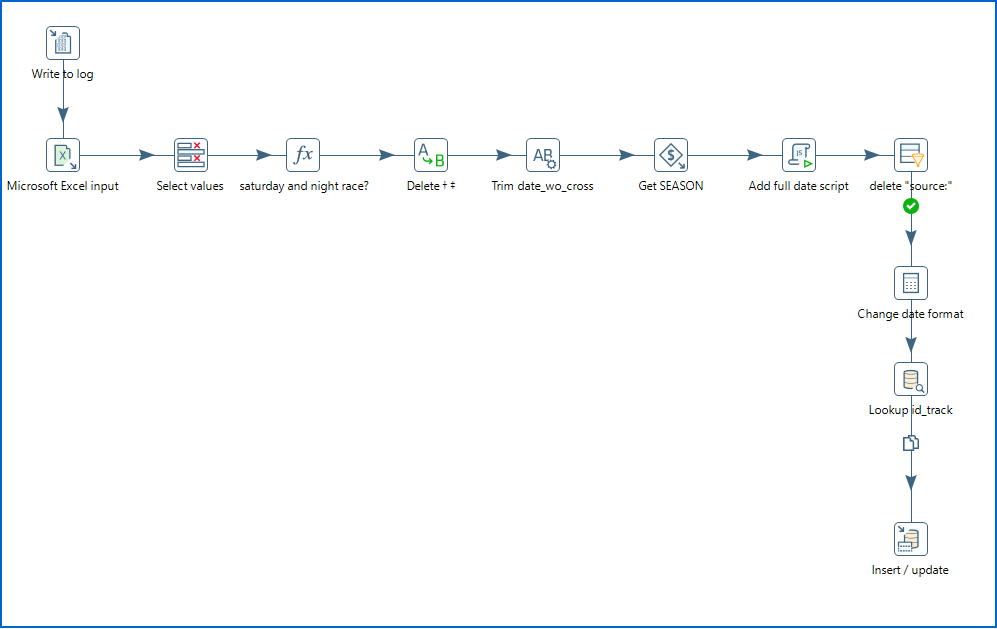


En la carga en la dimensión, el único campo que se puede actualizar es la ubicación del circuito. Esto es porque en las primeras temporadas los datos en la Wikipedia no contienen la ubicación del circuito. Entonces, cuando se procesen temporadas que contengan ese campo, en vez de crear un duplicado, se actualiza ese valor

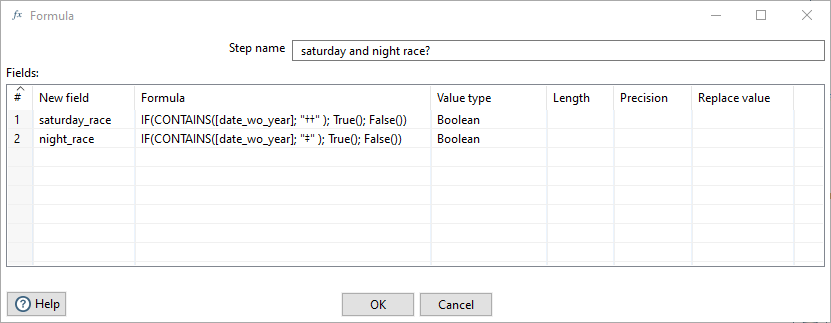


#### 4.3.5 Dimensión Grandes Premios

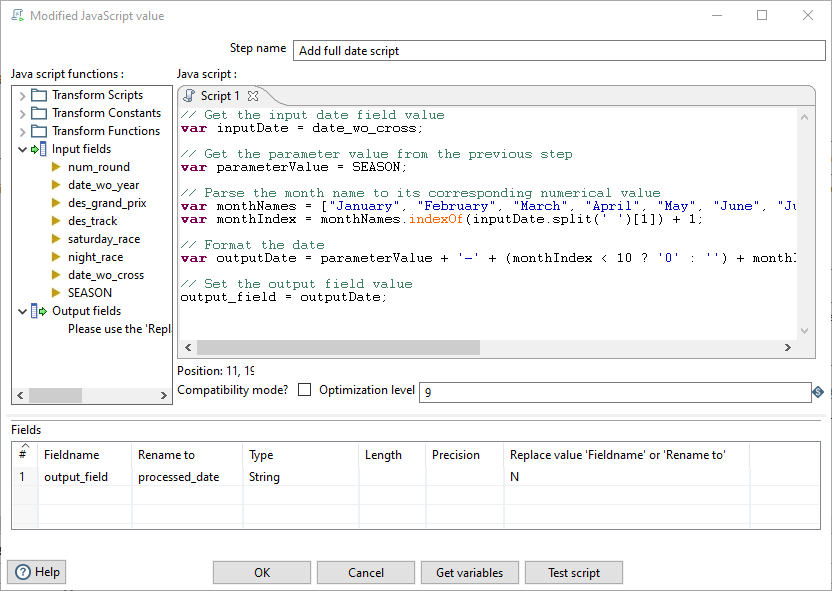
La transformación para cargar los datos de los grandes premios celebrados en una temporada. Está transformación es relativamente simple. Se cargan los datos de la hoja de Excel “Calendar” de la temporada que se está procesando, se seleccionan los campos relativos al circuito.



Tras esto se procesan también los símbolos “††” y “‡” que significan que una carrera se celebró un sábado y que una carrera fué nocturna. Para esto se usa el paso “Formula” donde podemos establecer filtros avanzados para crear nuevos campos en base a ellos. En este caso, si la fecha contiene este símbolo los nuevos campos “saturday\_race” y “night\_race” tienen valor booleano “True”, en caso contrario serán “False”.



Tras ello sustituimos “††” y “‡” por el carácter vacío para eliminarlos puesto que ya están procesados y eliminamos los espacios que pueda a ver al final del valor de la fecha. Una vez obtenida la temporada que se está procesando, usaremos un script en JavaScript para pasar las fechas del formato en texto “8 March” a un formato estándar “YYYY-03-08”. Una vez realizada la operación procedemos a crear el campo “processed\_date” de tipo String.



El código utilizado es el siguiente:

// Get the input date field value

var inputDate = date\_wo\_cross;

// Get the parameter value from the previous step

var parameterValue = SEASON;

// Parse the month name to its corresponding numerical value

var monthNames = ["January", "February", "March", "April", "May", "June", "July", "August", "September", "October", "November", "December"];

var monthIndex = monthNames.indexOf(inputDate.split(' ')[1]) + 1;

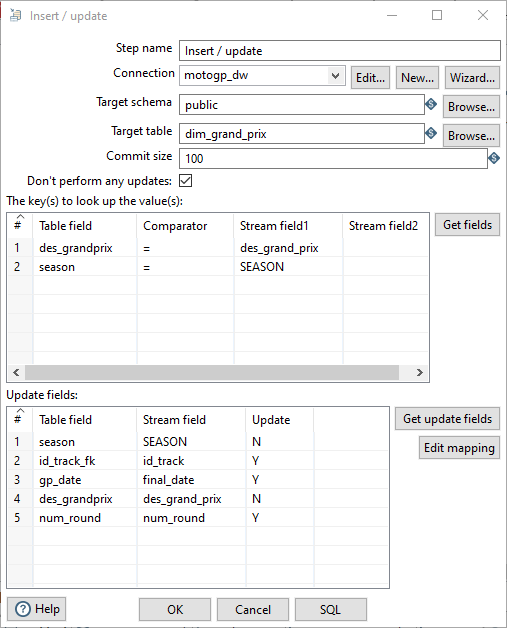
// Format the date

var outputDate = parameterValue + '-' + (monthIndex < 10 ? '0' : '') + monthIndex + '-' + (inputDate.split(' ')[0] < 10 ? '0' : '') + inputDate.split(' ')[0];

// Set the output field value

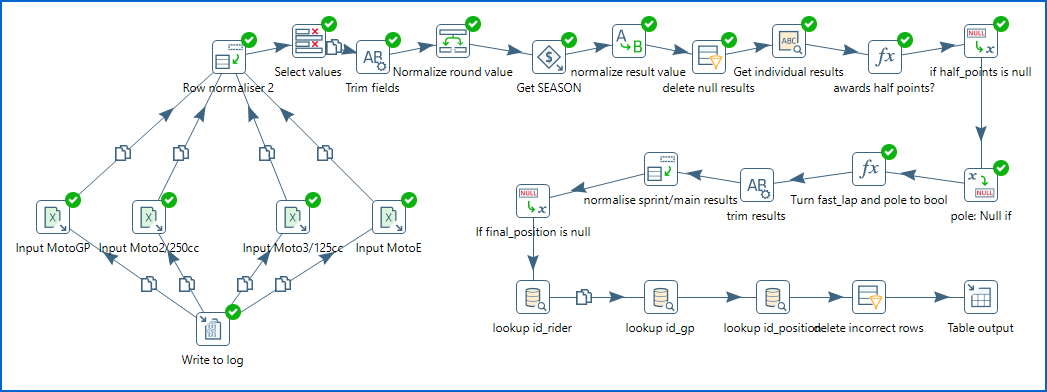
output\_field = outputDate;

Tras eliminar las filas con “source:” si las hubiera, procedemos a cambiar el tipo de dato de “processed\_date” a Date. Acto seguido recuperamos el identificador del circuito donde se celebró el gran premio para poder relacionar esta dimensión con la dimensión de circuitos. Para terminar la transformación, cargamos los datos a “dim\_grand\_prix”, donde los identificadores que hacen un registro único serán el nombre del gran premio y la temporada, pudiendo actualizar cualquiera de los otros datos.

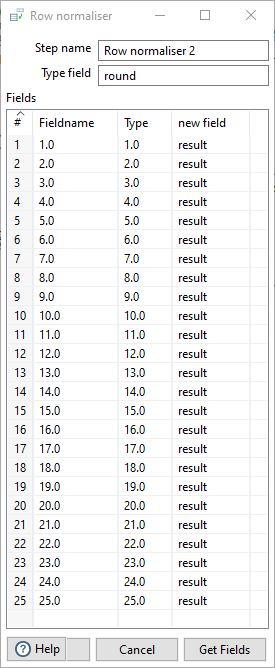


#### 4.3.6 Tabla de hechos Resultados

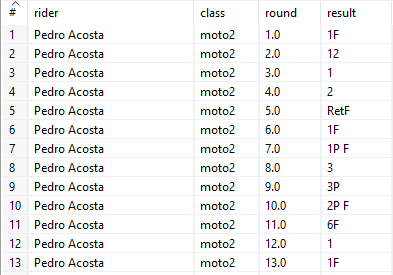
Este proceso poblará la única tabla de hechos que tenemos en nuestro modelo que almacena los resultados de cada piloto por carrera.



Comenzamos de igual forma que las dimensiones, con la diferencia de que normalizamos las filas. Esto es ya que en los datos cada fila es un piloto y cada columna es el resultado por carrera. En cambio lo que nosotros requerimos es que cada fila sea el resultado de cada piloto por cada carrera, teniendo así cada piloto el mismo número de filas que tenga la temporada. Para eso usamos el paso “Row Normaliser”. Así, cada resultado de cada ronda (del 1 al 25) será un valor de results



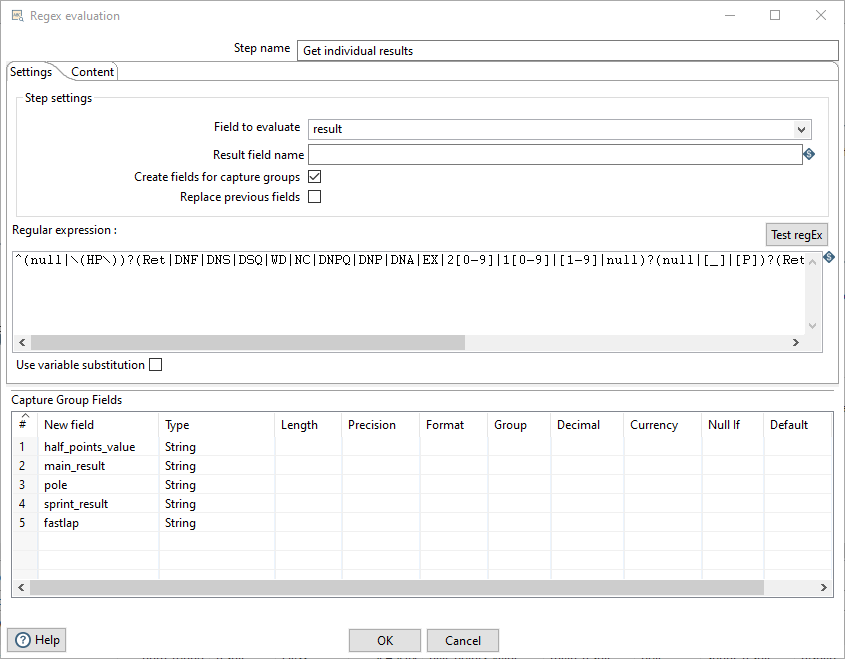
De está forma, el resultado de este paso queda tal que así:



Una vez hecho esto, los siguientes pasos son similares a los ya usados en las anteriores transformaciones.Con la excepción de que debemos ajustar los valores de la ronda y resultado para eliminar los decimales que crea Pentaho al procesar los valores.

En el paso “Gest individual results” usaremos expresiones regulares para procesar los valores del campo “results”. Extraemos para trabajar los siguientes campos de “results”:

* half\_points\_value: nos ayuda a saber si se otorgaron la mitad de puntos para esa carrera.
* main\_result: resultado de la carrera principal.
* pole: nos informa si el piloto salió desde la pole en las carreras de ese gran premio
* sprint\_result: resultado de la carrera sprint
* fastlap: nos informa si el piloto obtuvo la vuelta más rápida en la carrera principal de ese gran premio.



La expresión regular es la siguiente:

^(null|\(HP\))?

(Ret|DNF|DNS|DSQ|WD|NC|DNPQ|DNP|DNA|EX|2[0-9]|1[0-9]|[1-9]|null)?

(null|[\_]|[P])?

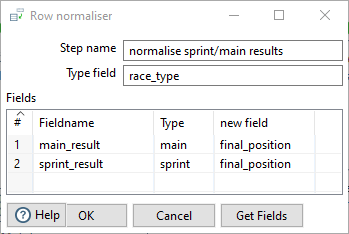
(Ret|DNF|DNS|DSQ|WD|NC|DNPQ|DNP|DNA|EX|2[0-9]|1[0-9]|[1-9]|null)?

(null|[F])?$

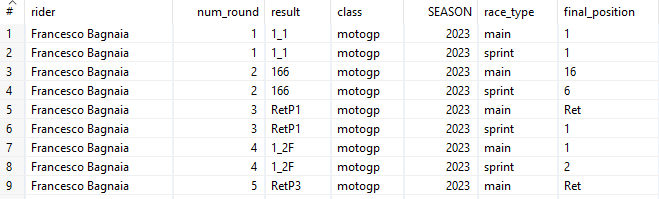
La primera fila nos ayudará a identificar si los resultados de la carrera principal otorgaron la mitad de puntos. La segunda nos permitirá aislar el resultado de la carrera principal. La tercera nos dice si ese piloto consiguió la pole en ese fin de semana. La cuarta nos permitirá aislar el resultado de la carrera sprint. Y la última si un piloto obtuvo la vuelta más rápida en la carrera principal. Los signos de interrogación al final de cada una de estas expresiones nos permiten que en el caso que un piloto tenga uno de estos campos en blanco, este hecho se almacenará correctamente en los campos anteriormente mencionados.

Los siguientes 5 pasos se encargan de asegurar que todos los campos creados sean del tipo adecuado y que estén en el formato que necesitamos.

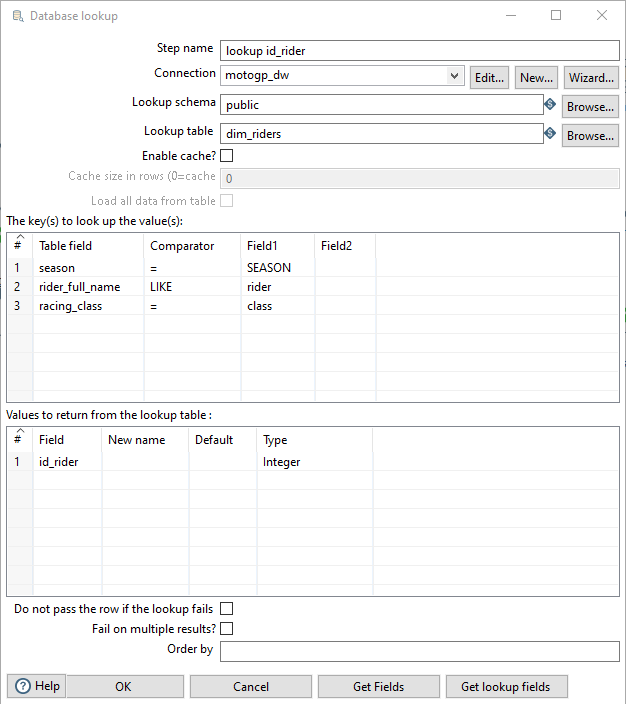
El segundo row normaliser de está transformación se encargará de separar en dos filas las sprints y las carreras principales, pasando el resultado a “final position”. De esta forma podremos asociar mejor los puntos de la posición dependiendo de la carrera.

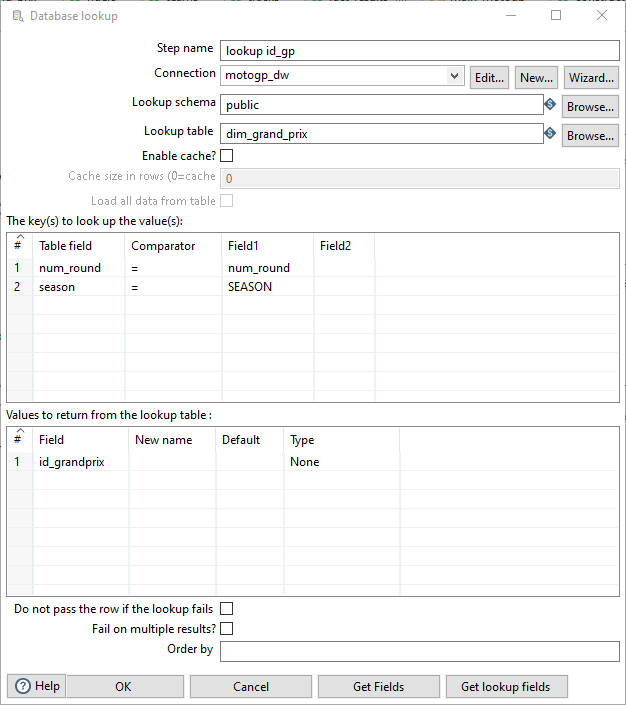


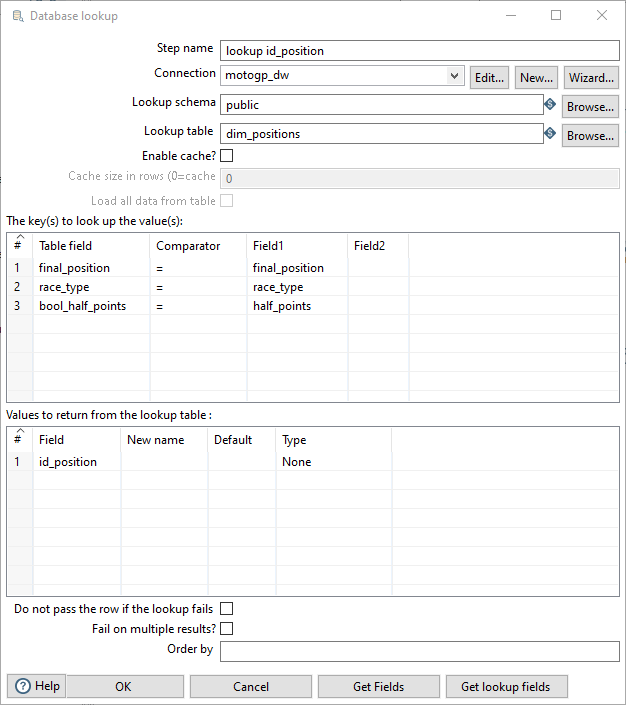
Quedando las filas tal que así:



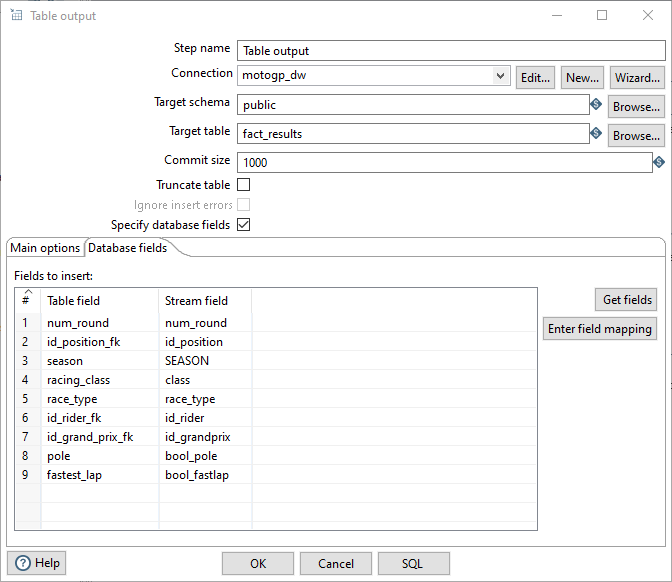
Una vez tenemos todos los datos, se hace una búsqueda a las dimensiones de posición, piloto y gran premio para recuperar los identificadores necesarios para relacionar correctamente esta tabla de hechos con esas dimensiones.





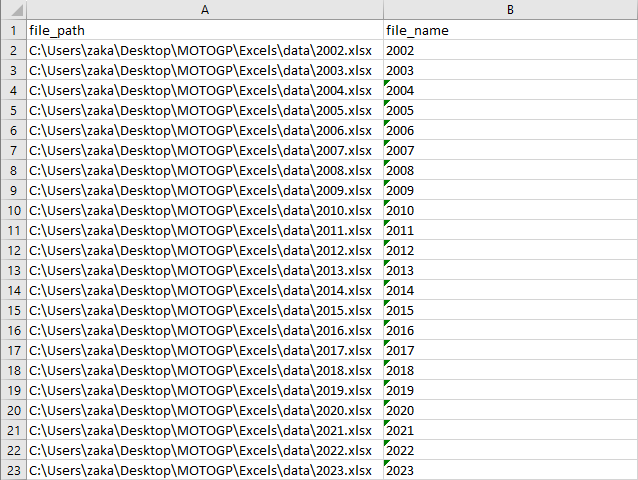


Una vez se tienen los datos se suben a la tabla de hechos. Usamos el tipo de paso “table output” en vez de los “Insert/Update” de las dimensiones ya que no hay campos “clave” para subir los datos.



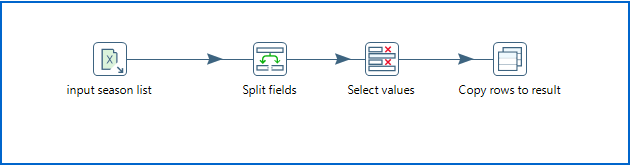
#### 4.3.7 Transformación de carga de temporadas

Esta transformación no está dedicada a procesar los datos de ninguna dimensión ni hecho, se encarga de cargar la lista de temporadas que deseamos procesar desde un excel.



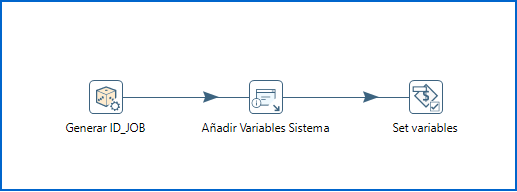
En está solución, lo que hacemos es aislar únicamente la fila “file\_name” ya que al estar todos los excel de datos en la misma ruta no es necesario usar la ruta. De todas formas, este campo se conserva por si pudiera ser de utilidad en futuras evoluciones del proyecto.

Una vez aislado este campo, se usa el paso “copy rows to result” para enviar las filas a la siguiente transformación o trabajo. Este concepto se explicará en el apartado 4.3.10.

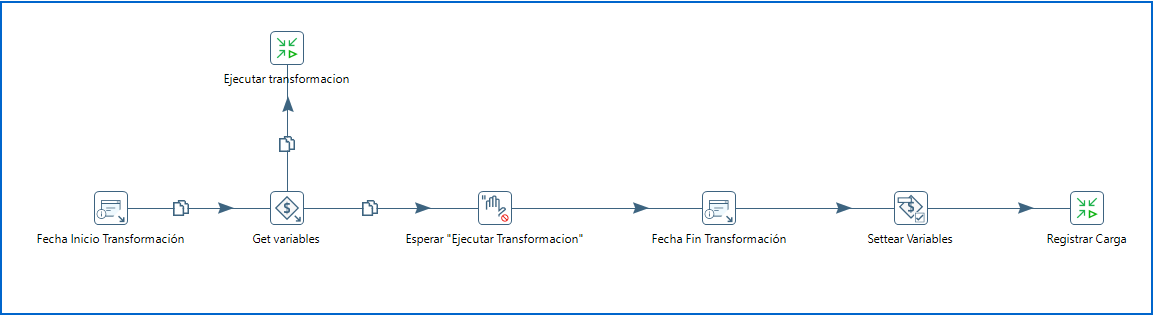


#### 

#### 4.3.8 Transformación de generación de identificador de trabajo (PLACEHOLDER)



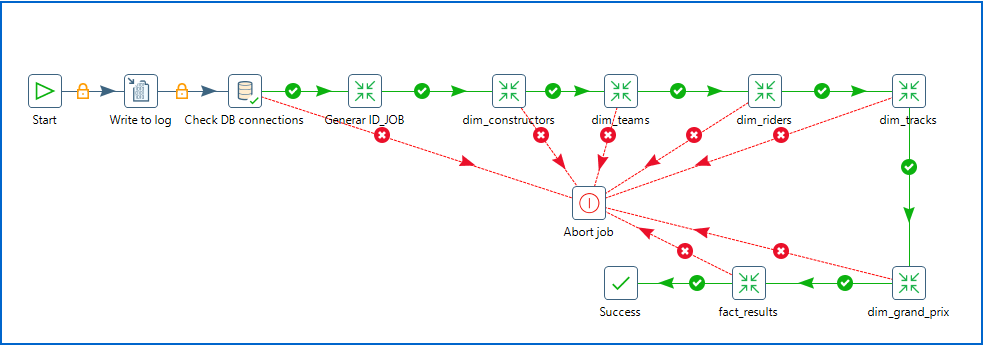
#### 4.3.8 Transformación de Orquestacion(PLACEHOLDER)



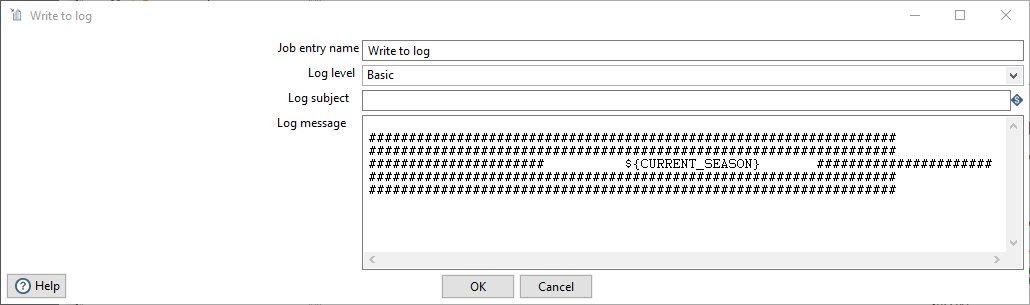
#### 

#### 

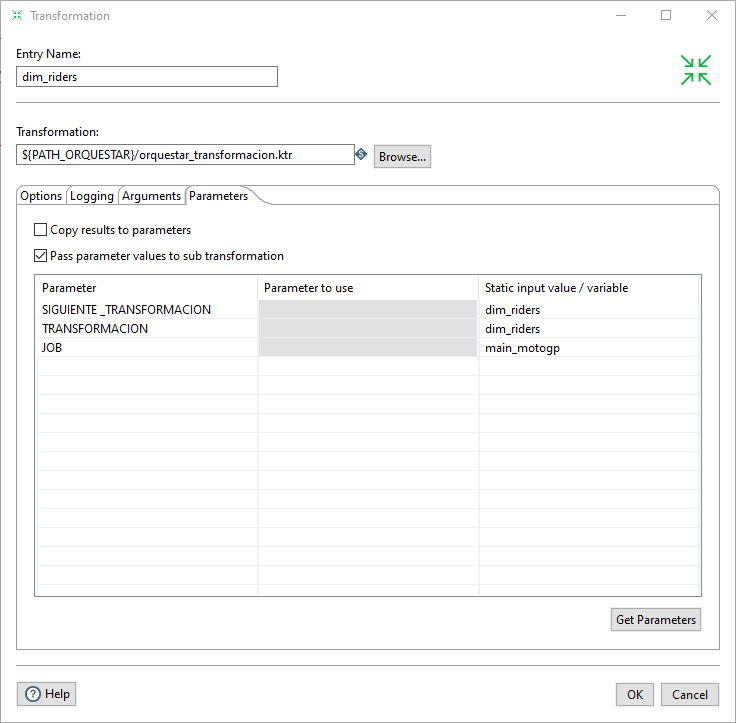
#### 4.3.9 Trabajo de carga de dimensiones y hechos



Este trabajo de Pentaho orquesta la carga de todas las dimensiones y hechos en orden detectando los errores que puedan surgir. Comenzamos por escribir en los logs la temporada que estamos procesando usando el parámetro “${CURRENT\_SEASON}” para poder seguir mejor el proceso una vez se ejecute.

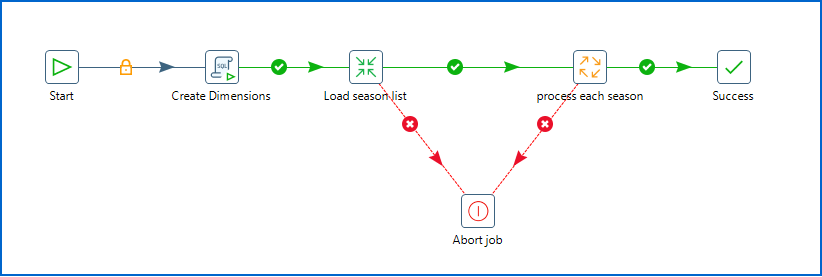


Tras ello detectamos primero si la conexión a la base de datos es correcta y generamos el identificador del job (PLACEHOLDER). Acto seguido se ejecuta cada una de las transformaciones relativas a cada dimensión y hecho para la temporada que se esta procesando a través de la transformación de Orquestación (PLACEHOLDER). En el caso de que estas transformaciones fallen, se maneja ese fallo usando el “Abort Job” usando las flechas rojas

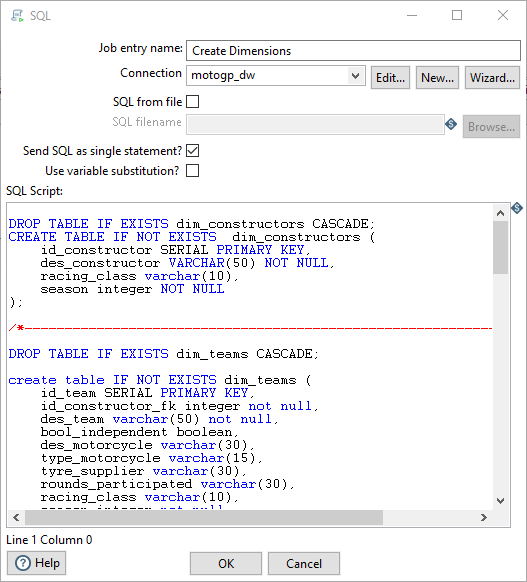


#### 4.3.10 Trabajo principal

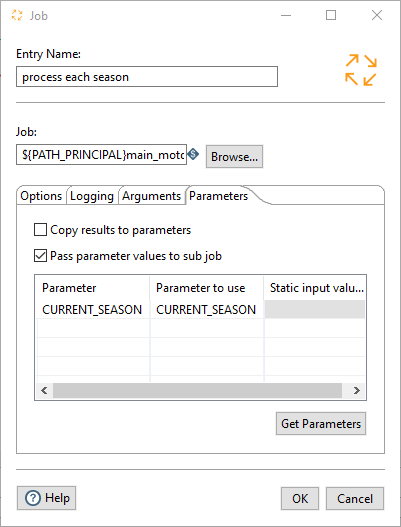
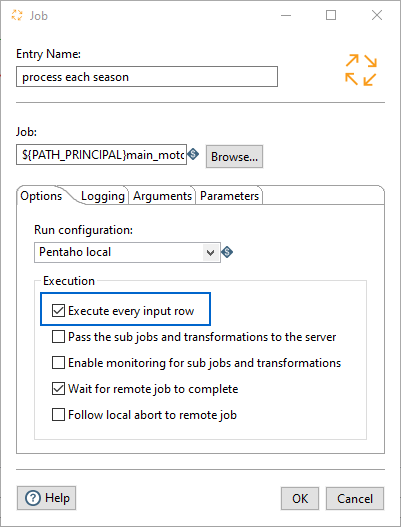
Este es el trabajo de Pentaho que ejecutamos inicialmente para poner en marcha todo el proceso. Comienza por ejecutar un script SQL que limpia todas las tablas de hechos y dimensiones que se pueblan en este proceso, es decir todas menos la dimensión de fecha y la dimensión de posiciones.



Esta limpieza se realiza para evitar siempre la duplicidad de datos. Gracias a la rapidez de procesamiento de los datos y la baja frecuencia de ejecución esto no supone un problema a nivel de rendimiento. Este script se puede cambiar más adelante si cambiara la naturaleza del proyecto, pero para la situación actual es una solución óptima.



Una vez hecho esto, se ejecuta la transformación 4.3.7 Transformación de carga de temporadas que detallamos anteriormente, que gracias al paso “Copy rows to results” permite pasar los resultados a la llamada al trabajo 4.3.9 de carga de dimensiones y hechos. En la llamada a este trabajo, se le configura para ejecutar este trabajo para cada valor que se le pasa, y a parametrizar dicho valor como “CURRENT\_SEASON”.



Una vez se ejecuta el trabajo 4.3.9 para cada temporada que tenemos en el excel, damos por concluida la carga de todos los datos en el Data Warehouse.

### 4.4 Cuadro de Mandos

https://ray3168.medium.com/abrief-history-of-streamlit-8d916c454429