



AZAHARA CLAVERO MOYA

DESCRIPCIÓN DEL CASO PRÁCTICO



PROPIEDADES DEL DOCUMENTO	
Título	Descripción del caso práctico
Versión	01
Autor	Azahara Clavero
Propietario	Dpto. Consultoría
Fecha de creación	2023/05/17
Código	--

CONTROL DE CAMBIOS		
VERSIÓN	MOTIVO DEL CAMBIO	FECHA
01	Primera edición	2023/05/23



Índice

1. INTRODUCCIÓN	4
2. PREPARACIÓN DEL ENTORNO	4
3. COMIENZO DEL DESARROLLO	5
3.1. CAPA BRONCE.....	5
3.2. CAPA PLATA.....	6
3.3. CAPA ORO	7
3.3.1. Tabla SIPSCO	8
3.3.2. Tabla SIPS.....	8
3.3.3. Tabla CONSUMO_ACTUAL.....	8
4. CONSULTAS DESTACABLES.....	8
5. PROBLEMAS IDENTIFICADOS Y POSIBLES SOLUCIONES	9
ANEXO I -CÓDIGO	10



1. INTRODUCCIÓN

El presente documento tiene como finalidad explicar el procedimiento para resolver el caso práctico propuesto por PUE. Se ha redactado de manera impersonal para seguir los pasos necesarios en la resolución del proyecto.

El proceso se divide en los siguientes pasos:

- Preparación del entorno
- Comienzo del desarrollo
- Consultas destacables
- Problemas identificados y posibles soluciones
- ANEXO I

Con el fin de agilizar la evaluación, se incluyen en el ANEXO I capturas de pantalla del código desarrollado en Zeppelin, permitiendo una rápida evaluación sin la necesidad de descargar el formato JSON, y cargarlo en Zeppelin.

2. PREPARACIÓN DEL ENTORNO

Con el objetivo de cumplir con las limitaciones y requisitos técnicos establecidos en el documento del caso práctico, se comienza instalando Visual Studio Code en la máquina virtual siguiendo las instrucciones proporcionadas en esta página:

[URL de la página de instalación de VS Code en CentOS 7](#)

Sin embargo, durante el proceso de instalación, se encuentra un error que indicaba que la versión de Git es la 1.8 y se requiere la versión 2.0. Para solucionar el problema, se siguen las instrucciones proporcionadas en esta página:

[URL de la página de instalación de Git 2.0 en CentOS 7](#)

A pesar de haber realizado estos pasos, todavía se experimentan dificultades y VS Code no funciona de manera fluida en la máquina virtual. Además, en general, la máquina virtual presenta errores y trabajar en ella resulta lento y provoca bastantes errores de refresco y problemas con los atajos de teclado. Ante esta situación, se decide montar un entorno Cloudera local utilizando Docker usando la imagen de quickstart de Cloudera, llegando a levantar todo el entorno pero sin conseguir realizar la conexión del IDE con Hadoop. Si bien dispongo del documento detallado de este proceso, es extenso y considero que no aporta mucho a este documento. Si es necesario, se puede proporcionar esta información bajo demanda.

En local se instala Metals en VS Code para trabajar en el entorno Docker con el IDE. La idea de trabajar con un IDE, a pesar de no haber sido abordada durante el curso, radica en la posibilidad de cargar directamente los cambios en la cuenta de GitHub, usando Git. Además, al trabajar con Scala, se puede utilizar el archivo “.sbt” de manera similar a como se utilizaría Maven, por lo que he podido averiguar investigando un poco.

Como resultado de los problemas encontrados en el entorno Docker, se opta por trabajar en la máquina virtual sin utilizar IDE.



3. COMIENZO DEL DESARROLLO

Antes de comenzar a trabajar, el equipo de trabajo acordó realizar una reunión el día 15 aprovechando la festividad, con el propósito de discutir el progreso en la lectura de la documentación y plantear cualquier duda relevante para la reunión con PUE programada para el martes 16. Tomo la iniciativa de organizar la reunión y tomar las minutas para que queden registradas.

Posteriormente, comparto la información de la reunión a través de Google Drive para que el resto de los compañeros, que no pudieron asistir, puedan acceder a ella. Esto garantiza que todos los miembros del equipo estén informados y puedan estar al tanto de los puntos tratados y las decisiones tomadas durante la reunión.

Se ha propuesto la realización de reuniones diarias con el objetivo de abordar dudas y brindarnos ayuda mutua en el proyecto.

El propósito de estas reuniones es fomentar la colaboración y facilitar la resolución de cualquier problema o pregunta que surja durante el desarrollo del proyecto. Al ser voluntarias se respeta la flexibilidad y disponibilidad de los miembros permitiendo que participen los que puedan hacerlo en cada ocasión.

Siguiendo las recomendaciones de designar a alguien como Scrum Master, hemos decidido dedicar 15 minutos diarios cada mañana para revisar el progreso de cada compañero y verificar el estado de avance de las tareas. Esta sesión nos permite tener una visión general del progreso individual y del equipo y nos ayudará a identificar problemas y necesidades de apoyo.

Esta información, guardada en un documento Excel, es enviada diariamente a PUE como seguimiento del proyecto y para que nos aporten de soluciones en los obstáculos encontrados.

A título personal he creado un proyecto en Jira para llevar un seguimiento del proyecto, creando tareas y dead lines, pero al tener que realizar todas las mismas tareas deja de tener sentido. Quizás otra metodología tipo kanban se ajuste más a este proyecto. Como sólo disponemos de 6 días no puedo dedicar mucho tiempo a estas tareas.

Una vez comentada la parte de gestión, comenzamos con la resolución del ejercicio.

Al comienzo del proyecto comencé a realizar pruebas en los cuadernos de Zeppelin y a medida que he ido entendiendo las necesidades he ido ordenando y estructurando los cuadernos.

Inicialmente me creé una carpeta de GitHub para subir diariamente los cambios, pero al renombrar finalmente los archivos, he decidido crear un nuevo proyecto público en GitHub con la exportación de todos los cuadernos de Zeppelin. <https://github.com/aclaverom/ProyectoPUE.git>

3.1. CAPA BRONCE

Para realizar esta etapa se deben seguir los siguientes pasos:

1. Descargar los archivos proporcionados en Google Drive.



2. Descomprimir los archivos para examinar la estructura de carpetas y su contenido.
3. En la máquina virtual, usando un magic command para trabajar en consola, crear el directorio especificado. En mi caso `/user/azahara/raw`.
4. Al intentar introducir los datos, se produce un error debido a que no se es el propietario del documento. Para solucionarlo, se puede agregar el usuario como superusuario, pero aun así continúan los problemas para el tratamiento y acceso a la documentación.
5. Para solventar el paso anterior se puede acceder directamente a HDFS (Hadoop Distributed File System) y utilizar la interfaz de usuario para generar la ruta requerida según la especificación técnica.
6. Para volcar la documentación descargada en el punto primero, se puede hacer directamente desde la consola o desde HDFS.

Archivos generados en esta etapa en cuaderno Zeppelin: *HDFS files*. Ver código en Anexo I

3.2. CAPA PLATA

Para realizar esta capa, denominada ETL (Extract, Transform, Load), se deben seguir los siguientes pasos:

- Generar la ruta donde se guardarán los archivos generados en esta etapa: `/user/azahara/refined` utilizando el sistema mencionado en la capa bronce.
- Realizar la ingesta de los archivos ubicados en las diferentes carpetas y que tiene diferentes formatos. Enfocarse en los archivos csv de la carpeta CNMC y los contratos, ya que son necesarios para los siguientes pasos.
- Durante esta etapa, leer los archivos, imprimir el esquema y verificarlos con la información de la documentación de cliente. Modificar el esquema para indicar los nombre y tipos de datos de las tablas, considerando las diferentes nomenclaturas utilizadas.

Ha existido una gran labor colaborativa a la hora de identificar los valores de los campos de cada tabla con los valores indicados en el documento del caso práctico, se ha generado un documento conjunto para poder renombrarlos correctamente y ha facilitado y se han resuelto dudas al respecto.

Existe la posibilidad de cambiar los datos en el esquema o durante la creación del DataFrame. En este caso realizamos la mayoría de los cambios en el esquema ya que a priori esta modificación es más flexible.

Comenzar con el esquema de la tabla SIPSCO. Una vez modificado, se crea un DataFrame cargando el esquema modificado.

Ahora implementamos un par de modificaciones relevantes con `“.withColumn”`, que son:

- Realizar el cálculo del número de días (fecha final- fecha inicial).



- Centro de medidas y diagnóstico (consumo total / número de días).

Estas modificaciones la realizamos en un nuevo DataFrame basado en el anterior.

Se realizan los mismos pasos con la información de SIPS. Se encuentran errores debido a que muchos campos en el csv están vacíos y algunos tipos de datos no permiten campos vacíos. Encontrar estos errores es bastante complicado ya que Zeppelin no dispone de un depurador que indique la ubicación del problema, lo que implica realizar pruebas manuales por secciones. Al encontrar un valor vacío en un tipo de dato, todos los datos de transforman en nulos.

Finalmente, para solucionar este problema, se cambian todos los tipos de datos que aparentemente dan problemas, como han sido los Booleanos, los DateType y los Int, por StringType y se añade comentario para cambiarlo posteriormente. Quizás en este caso hubiese sido más óptimo dejar que spark infiriera el esquema para luego modificar los tipos y hacer la conversión en el DataFrame.

Una vez realizado el cambio, convertir el DataFrame a formato parquet.

Archivos generados en esta etapa en cuaderno Zeppelin: *ETL CNMC_Consumption* y *ETL CNMC_SupplyPoint*. Ver código en Anexo I

El siguiente paso es crear las tablas *cnmc_sipsco* y *cnmc_sips* usando el parquet generado anteriormente.

Archivos generados en esta etapa en cuaderno Zeppelin: *table cnmc_sips* y *table cnmc_sipsco*

Se realiza la misma operativa con la tabla de contratos. En este caso se cambia la lectura del archivo al tratarse de un fichero json en lugar de csv.

Se busca en la última tabla CONSUMO_ACTUAL del documento los campos que se necesitan de esta tabla y hacemos un renombrado con un “.withColumn” y un “.withColumnRename”.

Se convierte el DataFrame a parquet para proceder al siguiente paso.

Archivos generados en esta etapa en cuaderno Zeppelin: *ETL contracts*

El siguiente paso es crear la tabla *contracts* usando el parquet generado anteriormente.

Archivos generados en esta etapa en cuaderno Zeppelin: *table contracts*. Ver código en Anexo I

3.3. CAPA ORO

Para realizar esta última capa del proceso, se deben seguir los siguientes pasos:

- Generar la ruta donde se guardarán los archivos generados en esta etapa:
/user/azahara/trusted utilizando el sistema mencionado en la capa bronce.
- Generar tablas SIPS y SIPSICO con el formato indicado en la documentación técnica.



3.3.1. Tabla SIPSCO

Se crea la tabla SIPSCO desde la tabla cnmc_sipsco y la convertimos a parquet.

Archivos generados en esta etapa en cuaderno Zeppelin: *table SIPSCO*. Ver código en Anexo I

3.3.2. Tabla SIPS

Calcular primero en una vista temporal la potencia máxima procedente de la tabla SIPSCO.

Generar la tabla SIPS con los campos especificados en la documentación de cliente y realizar un JOIN a la tabla temporal de la potencia máxima.

Convertimos a formato parquet.

Archivos generados en esta etapa en cuaderno Zeppelin: *table SIPS*. Ver código en Anexo I

3.3.3. Tabla CONSUMO_ACTUAL

Para generar esta tabla, se vuelve a trabajar con DataFrames (DF).

1. Leer los DF de sips, sipsco y contratos para ir generando los requisitos de la tabla.
2. Generar un nuevo DF ordenándolo mediante groupBy de los cups y agregando por última fecha de finalización.
3. Crear un nuevo DF realizando un JOIN entre el DF sipsco y el DF generado anteriormente.
4. Generar un nuevo DF que combine el anterior con el DF sips.
5. Generar un nuevo DF que combine el anterior con el DF de contratos.
6. Generar el último DF que contendrá los campos indicados en la documentación técnica correspondientes a CONSUMO_ACTUAL. En esta fase, el campo tarifa genera un error y se deben añadir las columnas, especificando el DF del cual procede tarifa.
7. Una vez finalizado, generar el formato parquet.

Archivos generados en esta etapa en cuaderno Zeppelin: *table currentConsumption*. Ver código en Anexo I.

Finalmente, generar la tabla CONSUMO_ACTUAL usando el parquet, al igual que se ha realizado con el resto de tablas.

Archivos generados en esta etapa en cuaderno Zeppelin: *CONSUMO_ACTUAL*. Ver código en Anexo I.

4. CONSULTAS DESTACABLES

Como consultas destacables proponemos las siguientes, que vienen detalladas en el siguiente cuaderno Zeppelin: *Disponible queries from currentConsumption*. Ver código en Anexo I

Potencia máxima contratada por código postal y distribuidora.



Numero máximo de días que una distribuidora ha estado dada de alta.

Consumo promedio por cada tarifa.

Número total de contratos por distribuidora.

Código postal con mayor número de contratos.

5. PROBLEMAS IDENTIFICADOS Y POSIBLES SOLUCIONES

Durante la ejecución del proyecto, he buscado soluciones en diversas fuentes, incluyendo reuniones con mis compañeros, búsquedas en Google y consultas a Chat GPT. Sin embargo, reconozco que aún me falta asimilar algunos conceptos clave. Durante el desarrollo del temario, hemos abordado una amplia gama de documentación, y es posible que se puedan aplicar algunos conceptos para optimizar el código.

PROBLEMA	SOLUCIÓN
Interpretación de la documentación. Muchas veces no tenemos claro qué nos están pidiendo.	Acudir a compañeros, cliente, etc. para solicitar aclaración.
Carga de documentación en HDFS. He tenido problemas con los permisos de los ficheros.	Generar la estructura de carpetas y cargar la documentación directamente de la interfaz de usuario de HDFS.
Error durante el cambio de los tipos de datos. Campos vacío o nulos.	Investigando me he dado cuenta de que hay tipos de datos que no pueden ser nulos o vacíos. Volver a cambiar los datos que dan problemas a StringType.
Depuración de errores.	En los esquemas, y debido al tipo de datos he tenido errores que he tenido que depurar a mano, viendo la ejecución de línea a línea o varias líneas.



ANEXO I -CÓDIGO

CAPA BRONCE

Archivo: HDFS files

BRONZE LAYER/HDFS files

Create HDFS directory

```
%sh
hdfs dfs -mkdir /user/azahara/raw
hdfs dfs -mkdir /user/azahara/refined
hdfs dfs -mkdir /user/azahara/trusted
```




Put data in HDFS

```
%sh
hdfs dfs -put -f /home/casoPractico/datos/CNMC /user/azahara/raw/CNMC
hdfs dfs -put -f /home/casoPractico/datos/contratos /user/azahara/raw/contracts
hdfs dfs -put -f /home/casoPractico/datos/EDP /user/azahara/raw/EDP
hdfs dfs -put -f /home/casoPractico/datos/FENOSA /user/azahara/raw/FENOSA
hdfs dfs -put -f /home/casoPractico/datos/IBERDROLA /user/azahara/raw/IBERDROLA
hdfs dfs -put -f /home/casoPractico/datos/VIESGO /user/azahara/raw/VIESGO
```

Browse Directory





/user/azahara/

Go!



Show25entries

Search:

<input type="checkbox"/>	Permission	Owner	Group	Size	Last Modified	Replication	Block Size	Name	
<input type="checkbox"/>	drwxr-xr-x	dr.who	supergroup	0 B	May 15 16:14	0	0 B	raw	
<input type="checkbox"/>	drwxr-xr-x	dr.who	supergroup	0 B	May 20 22:01	0	0 B	refined	
<input type="checkbox"/>	drwxr-xr-x	dr.who	supergroup	0 B	May 23 08:22	0	0 B	trusted	

Showing 1 to 3 of 3 entries

Previous1Next



CAPA SILVER

Archivo: ETL CNMC_Consumption

SILVER LAYER/ETL CNMC_Consumption



Create new schema as required in the especification

```
%spark
// First, import the Spark SQL types
import org.apache.spark.sql.types._

// Then specify the schema as a `StructType` instance:
val schemaElectricityConsumption = StructType(Array(
  StructField("cups", StringType, nullable = false),
  StructField("finicio", DateType, nullable = false),
  StructField("ffin", DateType, nullable = false),
  StructField("tarifa", StringType, false),
  StructField("consumoactivap1", FloatType, nullable = false),
  StructField("consumoactivap2", FloatType, false),
  StructField("consumoactivap3", FloatType, false),
  StructField("consumoactivap4", FloatType, false),
  StructField("consumoactivap5", FloatType, false),
  StructField("consumoactivap6", FloatType, false),
  StructField("consumoreactivap1", FloatType, false),
  StructField("consumoreactivap2", FloatType, false),
  StructField("consumoreactivap3", FloatType, false),
  StructField("consumoreactivap4", FloatType, false),
  StructField("consumoreactivap5", FloatType, false),
  StructField("consumoreactivap6", FloatType, false),
  StructField("potmaxp1", FloatType, true),
  StructField("potmaxp2", FloatType, true),
  StructField("potmaxp3", FloatType, true),
  StructField("potmaxp4", FloatType, true),
  StructField("potmaxp5", FloatType, true),
  StructField("potmaxp6", FloatType, true),
  StructField("numdias", IntegerType, true),
  StructField("cmd", FloatType, true)
))

val electricityConsumption = (spark.read
  .format("csv")
  .option("sep", ",")
  .option("header", "true")
  .schema(schemaElectricityConsumption)
  .load("/user/azahara/raw/CNMC/201712_SIPS2_CONSUMOS_ELECTRICIDAD_nacional.csv"))

electricityConsumption.show()
```

Create column numdias (ffin-finicio) and cmd (sum(active power) / numdias)

```
%spark
val electricityConsumptionMod = (electricityConsumption
  .withColumn("numdias", datediff(col("ffin"), col("finicio")))
  .withColumn("cmd", (col("consumoactivap1") + col("consumoactivap2") + col("consumoactivap3") + col("consumoactivap4") + col("consumoactivap5") + col("consumoactivap6")) / col("numdias")))
```

Convert to parquet

```
%spark
val SIPS2Path = "/user/azahara/refined/CNMC/201712_SIPS2_CONSUMOS_ELECTRICIDAD_nacional"
electricityConsumptionMod.write.mode("overwrite").parquet(SIPS2Path)
```

Showing modified schema

```
%spark
val dataFrameFromParquet = spark.read.parquet("/user/azahara/refined/CNMC/201712_SIPS2_CONSUMOS_ELECTRICIDAD_nacional")
dataFrameFromParquet.printSchema()
```



Archivo: ETL CNMC_SupplyPoint

SILVER LAYER/ETL CNMC_SupplyPoint

Create new schema as require in the especification

```
%spark
// First, import the Spark SQL types
import org.apache.spark.sql.types._

// Then specify the schema as a 'StructType' instance:
val schemaElectricitySupplyPoint = StructType(Array(
  StructField("coddist", StringType, nullable = false),
  StructField("cups", StringType, nullable = false),
  StructField("distribuidora", StringType, nullable = false),
  StructField("codpostal", IntegerType, false),
  StructField("municipioPS", StringType, nullable = false),
  StructField("codigoProvinciaPS", StringType, false),
  StructField("falta", DateType, false),
  StructField("varia", StringType, false),
  StructField("tensionps", IntegerType, false),
  StructField("potmaxib", StringType, false),
  StructField("potenciaMaximaAPM", DoubleType, false),
  StructField("codigoClasificacionPS", StringType, false),
  StructField("codigoDisponibilidadCP", StringType, false),
  StructField("perfilree", StringType, false),
  StructField("derext", StringType, false), // Ponemos String en lugar de Double ya que Double da error
  StructField("deraccl", StringType, false), // Ponemos String en lugar de Double ya que Double da error
  StructField("potcontrol", DoubleType, false),
  StructField("potcontrol2", DoubleType, false),
  StructField("potcontrol3", DoubleType, false),
  StructField("potcontrol4", DoubleType, false),
  StructField("potcontrol5", DoubleType, false),
  StructField("potcontrol6", DoubleType, false),
  StructField("fechaUltimoCambiodContrata", DateType, false),
  StructField("fechaUltimoCambiodComercializador", DateType, false),
  StructField("fechaIntercambioRescisos", DateType, false),
  StructField("fechaUltimaLectura", DateType, false),
  StructField("informacionImpagos", StringType, false),
  StructField("importeDepositoGarantiaEuros", StringType, false), // Ponemos String en lugar de Int ya que Int no soporta valores nulos, convertiremos despues si es necesario
  StructField("tipoidTitular", StringType, false),
  StructField("codviviendahabitual", StringType, false),
  StructField("codigoComercializadora", StringType, false),
  StructField("codigoTelegest", StringType, false), // Ponemos String en lugar de Int ya que Int no soporta valores nulos, convertiremos despues si es necesario
  StructField("fase", StringType, false), // Leemos como String ya que el valor que aparece es M o T, no tiene mucho sentido convertirlo a entero
  StructField("autoconsumo", StringType, false), // Leemos como String para luego realizar la conversin a Boolean posteriormente
  StructField("codigoTipoContrato", StringType, false),
  StructField("codigoPeriodicidadFacturacion", StringType, false), // Ponemos String en lugar de Int ya que Int no soporta valores nulos, convertiremos despues si es necesario
  StructField("codigoBIE", StringType, false),
  StructField("fechaEmissionBIE", StringType, true), // Ponemos String en lugar de Date, por algun motivo esta fallando al convertir cuando hay valores nulos
  StructField("fechaCaducidadBIE", StringType, true), // Ponemos String en lugar de Date, por algun motivo esta fallando al convertir cuando hay valores nulos
  StructField("fechaEmissionAPM", StringType, false), // Ponemos String en lugar de Date, por algun motivo esta fallando al convertir cuando hay valores nulos
  StructField("fechaCaducidadAPM", StringType, false), // Ponemos String en lugar de Date, por algun motivo esta fallando al convertir cuando hay valores nulos
  StructField("CNAE", StringType, false), // Ponemos String en lugar de Int ya que Int no soporta valores nulos, convertiremos despues si es necesario
  StructField("codigoModoControlPotencia", StringType, false), // Ponemos String en lugar de Int ya que Int no soporta valores nulos, convertiremos despues si es necesario
  StructField("potenciaCOP", StringType, false),
  StructField("codigoEquipobobmedida", StringType, false),
  StructField("codigoPSContratable", StringType, false), // Ponemos String en lugar de Int ya que Int no soporta valores nulos, convertiremos despues si es necesario
  StructField("motivoEstadoNoContratable", StringType, false), // Ponemos String en lugar de Int ya que Int no soporta valores nulos, convertiremos despues si es necesario
  StructField("codigoTensionMedida", StringType, false), // Ponemos String en lugar de Int ya que Int no soporta valores nulos, convertiremos despues si es necesario
  StructField("codigoClaseExpediente", StringType, false),
  StructField("codigoMotivoExpediente", StringType, false), // Ponemos String en lugar de Int ya que Int no soporta valores nulos, convertiremos despues si es necesario
  StructField("aplicacionBonoSocial", StringType, false), // Ponemos String en lugar de Int ya que Int no soporta valores nulos, convertiremos despues si es necesario
  StructField("aplicacionBonoSocial", StringType, false) // Ponemos String en lugar de Int ya que Int no soporta valores nulos, convertiremos despues si es necesario
))

//electricitySupplyPoint.show()
electricitySupplyPoint.show(2)
```

Schema modified

```
%spark
electricitySupplyPoint.printSchema()
```

Changing columns types

```
%spark
val electricitySupplyPointMod = {electricitySupplyPoint
  .withColumn("derext", when(col("derext").isNull, lit(0.0)).otherwise(col("derext").cast(DoubleType)))
  .withColumn("deraccl", when(col("deraccl").isNull, lit(0.0)).otherwise(col("deraccl").cast(DoubleType)))
  .withColumn("importeDepositoGarantiaEuros", when(col("importeDepositoGarantiaEuros").isNull, lit(0)).otherwise(col("importeDepositoGarantiaEuros").cast(IntegerType)))
  .withColumn("codigoTelegest", when(col("codigoTelegest").isNull, lit(0)).otherwise(col("codigoTelegest").cast(IntegerType)))
  .withColumn("autoconsumo", when(col("autoconsumo").isNull, false).otherwise(true))
  .withColumn("codigoPeriodicidadFacturacion", when(col("codigoPeriodicidadFacturacion").isNull, lit(0)).otherwise(col("codigoPeriodicidadFacturacion").cast(IntegerType)))
  .withColumn("fechaEmissionBIE", when(col("fechaEmissionBIE").isNull, lit(null)).otherwise(col("fechaEmissionBIE").cast(DateType)))
  .withColumn("fechaCaducidadBIE", when(col("fechaCaducidadBIE").isNull, lit(null)).otherwise(col("fechaCaducidadBIE").cast(DateType)))
  .withColumn("fechaEmissionAPM", when(col("fechaEmissionAPM").isNull, lit(null)).otherwise(col("fechaEmissionAPM").cast(DateType)))
  .withColumn("fechaCaducidadAPM", when(col("fechaCaducidadAPM").isNull, lit(null)).otherwise(col("fechaCaducidadAPM").cast(DateType)))
  .withColumn("CNAE", when(col("CNAE").isNull, lit(0)).otherwise(col("CNAE").cast(IntegerType)))
  .withColumn("codigoModoControlPotencia", when(col("codigoModoControlPotencia").isNull, lit(0)).otherwise(col("codigoModoControlPotencia").cast(IntegerType)))
  .withColumn("potenciaCOP", when(col("potenciaCOP").isNull, lit(0)).otherwise(col("potenciaCOP").cast(IntegerType)))
  .withColumn("motivoEstadoNoContratable", when(col("motivoEstadoNoContratable").isNull, lit(0)).otherwise(col("motivoEstadoNoContratable").cast(IntegerType)))
  .withColumn("codigoTensionMedida", when(col("codigoTensionMedida").isNull, lit(0)).otherwise(col("codigoTensionMedida").cast(IntegerType)))
  .withColumn("codigoClaseExpediente", when(col("codigoClaseExpediente").isNull, lit(0)).otherwise(col("codigoClaseExpediente").cast(IntegerType)))
  .withColumn("aplicacionBonoSocial", when(col("aplicacionBonoSocial").isNull, lit(0)).otherwise(col("aplicacionBonoSocial").cast(IntegerType)))
}
```

```
%spark
electricitySupplyPointMod.select("fechaEmissionAPM")
```

Convert to parquet

```
%spark
val SIPSPath = "/user/azahara/refined/CNMC/201712_SIPS2_PS_ELECTRICIDAD_nacional"
electricitySupplyPoint.write.mode("overwrite").parquet(SIPSPath)
```



Ejemplo archivo parquet guardado.

Browse Directory

/user/azahara/refined/CNMC

Go!

Show

25

entries

Search:

<input type="checkbox"/>	Permission	Owner	Group	Size	Last Modified	Replication	Block Size	Name	
<input type="checkbox"/>	drwxr-xr-x	livy	supergroup	0 B	May 20 09:39	0	0 B	201712_SIPS2_CONSUMOS_ELECTRICIDAD_nacional	
<input type="checkbox"/>	drwxr-xr-x	livy	supergroup	0 B	May 20 16:55	0	0 B	201712_SIPS2_PS_ELECTRICIDAD_nacional	

Browse Directory

/user/azahara/refined/CNMC/201712_SIPS2_CONSUMOS_ELECTRICIDAD_nacional

Go!



Show 25 entries

Search:

<input type="checkbox"/>	Permission	Owner	Group	Size	Last Modified	Replication	Block Size	Name	
<input type="checkbox"/>	-rw-r--r--	livy	supergroup	0 B	May 20 09:39	3	128 MB	_SUCCESS	
<input type="checkbox"/>	-rw-r--r--	livy	supergroup	4.79 MB	May 20 09:39	3	128 MB	part-00000-586bf2f8-a409-45b5-ab8c-6cf94fc0efeb-c000.snappy.parquet	
<input type="checkbox"/>	-rw-r--r--	livy	supergroup	5.49 MB	May 20 09:39	3	128 MB	part-00001-586bf2f8-a409-45b5-ab8c-6cf94fc0efeb-c000.snappy.parquet	

Showing 1 to 3 of 3 entries

Previous

1

Next

Archivo: ETL Contracts

SILVER LAYER/ETL contracts

Read a JSON file

```
%spark
val contracts = spark.read.json("/user/azahara/raw/contracts/contracts.json")
```

Contracts Schema

```
%spark
contracts.printSchema()
```

Rename columns

```
%spark
val renamedContracts = (contracts
  .withColumn("codpostal", col("fiscal_address.zip_code"))
  .withColumn("email", col("personal_data.email"))
  .withColumnRenamed("contract_number", "numcontrato")
  .withColumnRenamed("contract_date", "fcontrato")
  .withColumnRenamed("id", "codproducto") ) // asumimos que codproducto es el id

renamedContracts.show
```

Convert to parquet

```
%spark
val contractsPath = "/user/azahara/refined/contracts/contracts.parquet"
renamedContracts.write.mode("overwrite").format("parquet").save(contractsPath)
```



Archivo: table cnmc_sipsco

SILVER LAYER/table cnmc_sipsco

`%md`
Working exclusively with CNMC information

Drop table

```
%sql
DROP TABLE IF EXISTS cnmc_sipsco
```

Create table cnmc_sipsco

```
%sql
CREATE TABLE cnmc_sipsco USING PARQUET OPTIONS (PATH '/user/azahara/refined/CNMC/201712_SIPS2_CONSUMOS_ELECTRICIDAD_nacional')
```

showing table

```
%sql
select * from cnmc_sipsco limit 5
```

Archivo: table cnmc_sips

SILVER LAYER/table cnmc_sips

`%md`
Working exclusively with CNMC information

Drop table

```
%sql
DROP TABLE IF EXISTS cnmc_sips
```

Create table cnmc_sips

```
%sql
CREATE TABLE cnmc_sips
USING PARQUET
OPTIONS (PATH '/user/azahara/refined/CNMC/201712_SIPS2_PS_ELECTRICIDAD_nacional')
```

Showing table

```
%sql
SELECT * FROM cnmc_sips LIMIT 10
```



Archivo: table contracts

SILVER LAYER/table contracts



Drop table

```
%sql  
DROP TABLE IF EXISTS contracts
```

Create table contracts

```
%sql  
CREATE TABLE contracts USING PARQUET OPTIONS (PATH '/user/azahara/refined/contracts/contracts.parquet')
```

Showing table

```
%sql  
select * from contracts LIMIT 5
```



CAPA GOLD

Archivo: table SIPSCO

GOLD LAYER/table SIPSCO



Create table SIPSCO

```
%sql
CREATE TABLE SIPSCO AS
SELECT *
FROM edp_sipSCO
```

Show table SIPSCO

```
%sql
select * from SIPSCO
```

Convert to parquet

```
%spark
// read SQL and load information in DF
val dfSIPSCO = spark.sql("SELECT * FROM SIPSCO ")
// Path or parquet file
val SIPSCOPath = "/user/azahara/trusted/SIPSCO"
// writting parquet file
dfSIPSCO.write.mode("overwrite").parquet(SIPSCOPath)
```

Archivo: table SIPS

GOLD LAYER/table SIPS



Dropping table

```
%sql
DROP TABLE IF EXISTS SIPS
```

Create temporary view from SIPSCO table to obtain max power

```
%sql
CREATE TEMPORARY VIEW max_values AS
SELECT
  MAX(potmaxp1 ) AS potmaxp1,
  MAX(potmaxp2 ) AS potmaxp2,
  MAX(potmaxp3 ) AS potmaxp3,
  MAX(potmaxp4 ) AS potmaxp4,
  MAX(potmaxp5 ) AS potmaxp5,
  MAX(potmaxp6 ) AS potmaxp6
FROM SIPSCO
```




Create table SIPS joining max power from table sipsco

```
%sql
2
3 CREATE TABLE SIPS AS
4 SELECT
5     cups,
6     distribuidora,
7     coddist,
8     falta,
9     tarifa,
10    tensionps,
11    fases,
12    potmaxbie,
13    derext,
14    deraccll,
15    perfilree,
16    potcontrp1,
17    potcontrp2,
18    potcontrp3,
19    potcontrp4,
20    potcontrp5,
21    potcontrp6,
22    autoconsumo
23 FROM edp_sips
24
25 CROSS JOIN max_values
26
```

Show table SIPS

```
%sql
select * from SIPS
```

Convert to parquet

```
%spark

// read SQL and load information in DF
val dfSIPS = spark.sql("SELECT * FROM SIPS ")

// Path or parquet file
val SIPSPath = "/user/azahara/trusted/SIPS"

// writting parquet file
dfSIPS.write.mode("overwrite").parquet(SIPSPath)
```



Archivo: table currentConsumption

GOLD LAYER/table currentConsumption

Create DF groups by cups from SIPSCO

```
%spark
// First, import the Spark SQL types
import org.apache.spark.sql.DataFrame
import org.apache.spark.sql.functions._

// Read tables SIPS, SIPSCO y contracts
val sips: DataFrame = spark.table("SIPS")
val sipsCo: DataFrame = spark.table("SIPSCO")
val contracts: DataFrame = spark.table("contracts")

// Obtain the latest consumption by CUPS in SIPSCO
val lastConsumptionWithFfin = sipsCo.groupBy("cups").agg(max("ffin") as "ffin")
```

Adding rest of SIPSCO's fields

```
%spark
// Obtain the rest of the fields of sipsCo for the rows resulting from the aggregation
val lastConsumption = sipsCo.join(lastConsumptionWithFfin, Seq("cups", "ffin"), "inner")
lastConsumption.show(5)
```

Adding SIPS to the DF

FINISHED ▶ 🔍 ⌕

```
%spark
// Combine with the complete data of the last consumptions with the table of supplies
val lastConsumptionWithSips = sips.join(lastConsumption, Seq("cups", "ffin"), "inner")
lastConsumptionWithSips.show(5)
```

Adding contracts to the DF

FINISHED ▶ 🔍 ⌕

```
%spark
// unifiy with contracts
val latestConsumptionWithSipsWithContracts = contracts.join(lastConsumptionWithSips, Seq("CUPS"), "inner")
```

Selecting columns for final DF-DA ERROR-

FINISHED ▶ 🔍 ⌕

```
%spark
// Select the necessary columns for the CURRENT_CONSUMPTION table
val currentConsumption = latestConsumptionWithSipsWithContracts.select("CUPS", "numcontrato", "fcontrato", "codproducto", "email", "codpostal", "distribuidora", "coddist", "falta", "tarifa", "tensionps", "fases", "potmaxbie", "perfilree")
```



Selecting columns for final DF

```
%spark
import org.apache.spark.sql.functions.col

// Select the necessary columns for the CURRENT_CONSUMPTION table
val currentConsumption = latestConsumptionWithSipsWithContracts.select(
  col("default.sips.tarifa"), // Specify the table name for the 'tarifa' column
  col("CUPS"),
  col("numcontrato"),
  col("fcontrato"),
  col("codproducto"),
  col("email"),
  col("codpostal"),
  col("distribuidora"),
  col("coddist"),
  col("falta"),
  col("tensionps"),
  col("fases"),
  col("potmaxbie"),
  col("perfilree"),
  col("potcontrp1"),
  col("potcontrp2"),
  col("potcontrp3"),
  col("potcontrp4"),
  col("potcontrp5"),
  col("potcontrp6"),
  col("potmaxp1"),
  col("potmaxp2"),
  col("potmaxp3"),
  col("potmaxp4"),
  col("potmaxp5"),
  col("potmaxp6"),
  col("finicio"), |
  col("ffin"),
  col("consumoactivap1"),
  col("consumoactivap2"),
  col("consumoactivap3"),
  col("consumoactivap4"),
  col("consumoactivap5"),
  col("consumoactivap6"),
  col("numdias"),
  col("cmd")
)

currentConsumption.show(5)
```

Archivo: CONSUMO_ACTUAL

GOLD LAYER/CONSUMO_ACTUAL



Drop table

```
%sql
DROP TABLE IF EXISTS currentConsumption
```

Create table currentConsumption

```
%sql
CREATE TABLE currentConsumption USING PARQUET OPTIONS (PATH '/user/azahara/trusted/consumoActual')
```

Showing table

```
%sql
select * from currentConsumption LIMIT 5
```



Archivo: Disposable queries from currentConsumption

Maxima potencia contratada por codigo postal y distribuidora

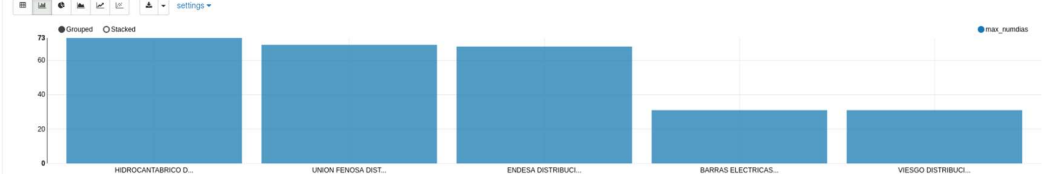
```
%sql
SELECT codpostal, distribuidora, MAX(potmaxbie) AS max_power
FROM currentConsumption
GROUP BY codpostal, distribuidora
ORDER BY max_power DESC
```

-- En el codigo postal 17013 hay una potencia de 9200000 , entendemos que es una errata, aun asi deberia aparecer como los primeros valores al estar ordenado descendientemente.

codpostal	distribuidora	max_power
93476	HIDROCANTABRICO D...	9900
75384	HIDROCANTABRICO D...	9900
42068	HIDROCANTABRICO D...	9900
92751	HIDROCANTABRICO D...	9900
63233	HIDROCANTABRICO D...	9900
03643	HIDROCANTABRICO D...	9900
02734	HIDROCANTABRICO D...	9900
26617	HIDROCANTABRICO D...	9900

Numero maximo de dias que una distribuidora ha estado de alta

```
%sql
SELECT distribuidora, MAX(numdias) AS max_numdias
FROM currentConsumption
GROUP BY distribuidora
ORDER BY max_numdias DESC
```



Consumo promedio por cada tarifa

```
%sql
SELECT tarifa, AVG(consumoactivap1 + consumoactivap2 + consumoactivap3 + consumoactivap4 + consumoactivap5 + consumoactivap6) AS consumo_promedio
FROM currentConsumption
GROUP BY tarifa
ORDER BY consumo_promedio DESC
```

tarifa	consumo_promedio
U11	74269000
003	4628439.024390244
006	769333.3333333334
004	468348.83720930235
005	429933.3333333333
001	166437.68996960486

Numero total de contratos por distribuidora

```
%sql
SELECT distribuidora, COUNT(DISTINCT numcontrato) AS total_contratos
FROM currentConsumption
GROUP BY distribuidora
ORDER BY total_contratos DESC
```

distribuidora	total_contratos
UNION FENOSA DIST...	1847
HIDROCANTABRICO D...	760
ENDESA DISTRIBUCI...	126
BARRAS ELECTRICAS...	9
VIESGO DISTRIBUCI...	4



Numero total de contratos por distribuidora

```
%sql
SELECT distribuidora, COUNT(DISTINCT numcontrato) AS total_contratos
FROM currentConsumption
GROUP BY distribuidora
ORDER BY total_contratos DESC
```

      settings ▼

distribuidora	total_contratos
UNION FENOSA DIST...	1847
HIDROCANTABRICO D...	760
ENDESA DISTRIBUCI...	126
BARRAS ELECTRICAS...	9
VIESGO DISTRIBUCI...	4

Codigo postal con mayor numero de contratos

```
%sql
SELECT codpostal,distribuidora, COUNT(DISTINCT numcontrato) AS total_contratos
FROM currentConsumption
WHERE finicio <= '2016-01-11'
GROUP BY codpostal, distribuidora
ORDER BY total_contratos DESC
LIMIT 5
-- Esta consulta no aporta mucho valor... solo me sale 1 contrato por distribuidora y cod portal . es raro
```

      settings ▼

codpostal	distribuidora	total_contratos
68236	UNION FENOSA DIST...	1
57898	UNION FENOSA DIST...	1
01291	UNION FENOSA DIST...	1
48567	UNION FENOSA DIST...	1
41990	UNION FENOSA DIST...	1