Extraer, transformar y analizar

Las empresas pueden centrarse en las ganancias y las ventas, pero **Business Intelligence** (BI) se centra en los datos. Las actividades que dependen de los datos requieren que el analista de negocios los adquiera de diversas fuentes. El término extracto, transformación y carga, comúnmente conocido como ETL (**Extract, Transform and Load**), es un proceso deliberado para obtener, manipular y almacenar datos para satisfacer las necesidades comerciales o analíticas.

ETL es el punto de partida para muchos proyectos analíticos comerciales. Un ETL mal ejecutado puede afectar a un negocio en forma de costo adicional y tiempo perdido para tomar decisiones. Este documento cubre los siguientes cuatro temas clave:

- Comprender Big Data en BI Analytics
- Extraer datos de fuentes
- Transformar datos para adaptarse a las necesidades analíticas
- Carga de datos en sistemas comerciales para el análisis

Este documento presenta cada paso ETL dentro del contexto del entorno computacional R. Cada paso se divide en niveles de detalle más finos e incluye una variedad de situaciones que los analistas de negocios encuentran al ejecutar BI en un mundo de los negocios de Big Data.

Comprender Big Data en BI Analytics

Antes de comenzar a describir el proceso ETL, considera su importancia en la inteligencia de negocios. La revista CIO ofrece una definición popular y útil de BI (Mulcahy, 2007):

"*Business Intelligence, o BI, es un término general que se refiere a una variedad de aplicaciones de software utilizadas para analizar los datos sin procesar de una organización. La BI como disciplina se compone de varias actividades relacionadas, incluida la minería de datos, el procesamiento analítico en línea, la consulta y los informes"*.

La definición de Mulcahy esa la esencia de esta parte del curso, ya que presenta soluciones en R para guiarte a través de los pasos desde las técnicas analíticas de datos para comunicar tus resultados. El propósito de las aplicaciones de BI ha cambiado en la última década, ya que los desafíos de Big Data afectan al mundo de los negocios de manera que se experimenta por primera vez en las ciencias hace décadas.

Puedes encontrar el término Big Data en muchas configuraciones comerciales. Aparece en los anuncios de los campamentos de entrenamiento, atrae a los asistentes a conferencias y a los líderes empresariales perplejos. Podría decirse que el término está mal definido. Una presentación de 1998 dada por John Mashey, entonces científico principal de Silicon Graphics, a menudo se cita como el documento que introdujo el término. El impacto de los grandes datos en los negocios es innegable, a pesar de su significado esquivo. Hay un acuerdo general sobre las siguientes tres características de Big Data, llamado **3VS**:

- **Volumen**: El tamaño de los conjuntos de datos ha crecido de megabytes a petabytes
- **Velocidad**: La velocidad de la llegada de datos ha cambiado a casi tiempo real
- **Variedad**: Las fuentes de datos han crecido desde bases de datos estructuradas a las no estructuradas, como las redes sociales, los sitios web, el audio y el video

Juntas, estas tres características plantean un desafío creciente para la comunidad empresarial. Los datos se almacenan en instalaciones en una vasta red de servidores locales o bases de datos relacionales.

El software virtual accede a las aplicaciones basadas en la nube. Las aplicaciones BI generalmente han incluido paneles (dashboards) estáticos basados en medidas fijas que utilizan datos estructurados. Big Data cambia el negocio al brindar una ventaja competitiva a aquellos que pueden extraer valor de las fuentes grandes y rápidas de diversos datos.

Hoy, la gente pregunta a los analistas de negocios, ¿qué va a pasar? Para responder a este tipo de pregunta, un negocio necesita herramientas y procesos para aprovechar el creciente flujo de datos. A menudo, estos datos no encajarán en las bases de datos existentes sin transformación. La necesidad continua de adquirir datos requiere un enfoque ETL estructurado para disputar la naturaleza no estructurada de los datos modernos. Mientras lees este documento, piensa en cómo las empresas pueden beneficiarse del uso de las técnicas presentadas, incluso cuando son menos complejas que Big Data.

Caso de uso: Bike Sharing, LLC

Comenzarás tu exploración de BI y análisis a través de la lente de un negocio ficticio llamado Bike Sharing, LLC. La compañía opera y mantiene una flota de bicicletas de alquiler públicamente en el área metropolitana de Washington D.C. Sus clientes son típicamente del área urbana, incluidas personas de negocios, gobierno y universidades. Los clientes disfrutan de la comodidad de encontrar bicicletas fácilmente dentro de una red de estaciones de bike-sharing en toda la ciudad. Los inquilinos pueden alquilar una bicicleta en un lugar y dejarla en otra estación.

Bike Sharing, LLC comenzó las operaciones en 2011, y ha disfrutado de un crecimiento continuo. Rápidamente establecieron un grupo BI para realizar un seguimiento de los datos recopilados sobre transacciones, clientes y factores relacionados con alquileres, como el clima, las vacaciones y las horas del día. En 2014, comenzaron a comprender cómo podrían usar conjuntos de datos de código abierto para guiar las decisiones sobre ventas, operaciones y publicidad. En 2015, ampliaron su grupo de talentos de BI con analistas de negocios experimentados con R y métodos estadísticos que podrían usar datos compartidos de bicicletas de nuevas maneras.

Te uniste a Bike Sharing hace solo unos meses. Tienes una comprensión básica de R de los muchos cursos y tutoriales que utilizas para expandir tus habilidades. Estás trabajando con un buen grupo que tiene un conjunto de habilidades diversos, incluida la programación, las bases de datos y el conocimiento empresarial. Los primeros datos que se les ha dado son los datos de alquiler de bicicletas que cubren los dos años. Período del 1 de enero de 2011 al 31 de diciembre de 2012. Puedes usar el archivo adjunto bike_sharing_data.csv.

Las fuentes de datos a menudo incluyen un diccionario de datos para ayudar a los nuevos usuarios a comprender el contenido y la codificación de los datos.

Diccionario de datos para los datos de Bike Sharing:

- datetime: Fecha por hora + marca de tiempo
- season: 1 = primavera, 2 = verano, 3 = otoño, 4 = invierno
- holiday: Si el día se considera unas vacaciones
- workingday: Si el día no es un fin de semana ni unas vacaciones
- weather
    - 1: Claro, Pocas nubes, Parcialmente nubladas, Parcialmente nubladas
    - 2: Mist + nublado, Neblina + Nubes rotas, Niebla + Pocas nubes, Niebla
    - 3: Nieve ligera, Lluvia ligera + Tormenta eléctrica + Nubes dispersas, Lluvia ligera + Nubes dispersas
    - 4: Lluvia pesada + Gránulos de hielo + Tormenta eléctrica + Neblina, Nieve + Niebla
- temp: Temperatura en Celsius
- Atemp: Se siente como temperatura en Celsius
- humidity: Humedad relativa
- windspeed: Velocidad del viento
- casual: Número de alquileres de usuarios no registrados iniciados
- registered: Número de alquileres de usuarios registrados iniciados
- count: Número de alquileres totales

Uno de tus objetivos es fortalecer tus habilidades ETL. En este caso de uso, aprenderás habilidades comunes de extracción, transformación y carga para almacenar un conjunto de datos en un archivo para el análisis. Bienvenido al equipo Bike Sharing

Extraer datos de fuentes

Ahora nos ponemos a trabajar. El objetivo de este curso es diseñar, desarrollar y entregar un producto de datos de inteligencia de negocios, un producto de datos. En esta sección, exploraremos dos métodos de extracción para importar datos de diferentes tipos de fuentes:

- Importación de CSV y otros formatos de archivo
- Importar datos de bases de datos relacionales

Dependiendo de tu formación, puedes estar más o menos familiarizado con ambos tipos de métodos de extracción. Aquí, aprenderás o refrescarás tu conocimiento de ambos para tener una comprensión más completa de ETL.

Importación de CSV y otros formatos de archivo

Puedes cargar el archivo de datos de Bike Sharing en el entorno R utilizando la función read.csv(). Es una forma fácil y comúnmente utilizada de cargar archivos CSV:

```
> bike<-read.csv("bike_sharing_data.csv")
```

Llamar a esta función leerá el archivo siempre que se encuentre en tu directorio de trabajo R (en la carpeta Documentos). El directorio de trabajo es el espacio R que usas, muy parecido a un directorio de inicio. Hay dos comandos que puedes usar para verificar y cambiar tu directorio de trabajo:

- getwd(): Esto devolverá el directorio de trabajo actual como una ruta en la consola R.
- setwd(<path>): Esto se usa en la consola para cambiar el directorio de trabajo a <pathh> que pasas en la función.

Si intentas leer un archivo que no se encuentra en tu directorio de trabajo, verás un mensaje de error.
Algunos analistas administran los datos en un directorio de datos independiente, un nivel por debajo de su directorio de trabajo. En este caso, puedes agregar una ruta delante del nombre del archivo. El siguiente ejemplo muestra cómo se ubica el archivo de datos un nivel por

debajo en un directorio de datos. Agregar la cadena ./data/ al principio del nombre del archivo permitirá que R acceda a los datos:

```
> bike<-read.csv("./data/bike_sharing_data.csv")
> str(bike)
```

La función str() no es necesaria para importar los datos, pero sí proporciona una confirmación de que los datos se leyeron en el entorno de R. También te proporciona una vista rápida de la estructura del conjunto de datos, sus dimensiones y los nombres de las variables:

```
'data.frame':   17379 obs. of  12 variables:
 $ datetime  : chr  "1/1/2011 0:00" "1/1/2011 1:00" "1/1/2011 2:00" "1/1/2011 3:00" ...
 $ season    : int  1 1 1 1 1 1 1 1 1 1 ...
 $ holiday   : int  0 0 0 0 0 0 0 0 0 0 ...
 $ workingday: int  0 0 0 0 0 0 0 0 0 0 ...
 $ weather   : int  1 1 1 1 1 2 1 1 1 1 ...
 $ temp      : num  9.84 9.02 9.02 9.84 9.84 ...
 $ atemp     : num  14.4 13.6 13.6 14.4 14.4 ...
 $ humidity  : int  81 80 80 75 75 75 80 86 75 76 ...
 $ windspeed : num  0 0 0 0 0 ...
 $ casual    : int  3 8 5 3 0 0 2 1 1 8 ...
 $ registered: int  13 32 27 10 1 1 0 2 7 6 ...
 $ count     : int  16 40 32 13 1 1 2 3 8 14 ...
```

La función read.csv() utiliza parámetros predeterminados que son consistentes con los archivos CSV, como el uso de una coma (,) como separador. El entorno R tiene una opción de lectura más flexible para los casos en los que puede tener datos delimitados por tabulaciones o un archivo TXT. Esta opción es la función read.table():

```
> bike<-read.table("bike_sharing_data.csv", sep = ",", header = TRUE)
```

Esta función funciona de manera idéntica a la función read.csv(). De hecho, read.csv() se basa en read.table() y utiliza sep = "," y header = TRUE como parámetros predeterminados.

Importación de datos desde bases de datos relacionales

También puedes utilizar R para acceder a los datos almacenados en muchas bases de datos relacionales a través de la interfaz de programación de aplicaciones Open Database Connectivity (ODBC). En R, esto se hace mediante el paquete RODBC. Este paquete te proporciona una ventaja para los datos heredados. Al establecer una conexión R con las bases de datos relacionales en Bike Sharing, LLC, puedes reutilizar las consultas de tipo SQL existentes que ya puedan existir en el grupo de BI.

Imagínate que el conjunto de datos de Bike Sharing no fuera un archivo CSV, sino que estuviera almacenado en una base de datos relacional. Puedes acceder a los datos cargando el paquete RODBC y estableciendo una conexión con la base de datos mediante los siguientes parámetros: nombre de la fuente de datos (data source), ID de usuario y contraseña. En este ejemplo teórico, el usuario Paul desea acceder a la fuente de datos ourDB utilizando su contraseña R4BI:

```
> install.packages("RODBC")
> library(RODBC)
> connection <- odbcConnect(dsn = "ourDB", uid = "Paul", pwd = "R4BI")
```

Una vez que hayas establecido una conexión con tu base de datos, puedes usar R para escribir una consulta SQL, pasar esta consulta a la función sqlQuery() y leer los datos en un framework de datos de bicicletas. Aprendiste que hay una tabla en la base de datos llamada marketing:

```
> query <- "SELECT * FROM marketing"
> bike <- sqlQuery(connection, query)
> close(connection)
```

La función close() cierra la conexión entre R y la base de datos. Al hacer esto, se liberará la memoria consumida por la conexión abierta. Es una buena práctica a seguir.

Consejo de R: Una cosa que puedes hacer al establecer conexiones es agregar un registro para registrar tus procesos ETL. En caso de que algo salga mal, el registro capturará el problema y te ayudará a encontrarlo rápidamente. El paquete log4r está escrito específicamente para esta aplicación.

También hay bases de datos no relacionales en el lugar de trabajo. Las bases de datos no relacionales almacenan documentos, datos de valores llave y datos no estructurados. A diferencia del enfoque de base de datos relacional, contienen diferentes estructuras de datos agrupadas en nodos, clústeres u otros esquemas. La comunidad R ha creado varios paquetes para conectarse con bases de datos no relacionales y sistemas de archivo distribuidos. Puedes conectarte a MongoDB mediante el paquete rmongodb y a la computación distribuida Spark mediante el paquete SparkR.

Consejo de BI: Al observar el enfoque de base de datos utilizado dentro de una organización y conceptualizar los datos, un analista de negocios puede comprender mejor los procesos comerciales para mejorar la calidad del análisis y satisfacer las necesidades de la organización.

Transformación de datos para satisfacer las necesidades analíticas

En la sección anterior, aprendiste a extraer datos e importarlos a R desde varias fuentes. Ahora puedes transformarlos para crear subconjuntos de los datos. Esto es útil para proporcionar a otros miembros del equipo una parte de los datos que pueden usar en su trabajo sin necesidad de tener el conjunto de datos completo. En esta sección, aprenderás las siguientes cuatro actividades clave asociadas con la transformación:

- Filtrar filas de datos
- Seleccionar columnas de datos
- Agregar una columna calculada a partir de datos existentes
- Agregar datos en grupos

Aprenderás a usar funciones del paquete dplyr para realizar la manipulación de datos. Si estás familiarizado con SQL, entonces dplyr es similar en la forma en que filtra, selecciona, ordena y agrupa los datos. Si no estás familiarizado con SQL, no te preocupes. Esta sección te presentará el paquete dplyr. Obtén más información sobre el paquete dplyr en el sitio web de R, https://cran.r-project.org/web/packages/dplyr/index.html.

Solicitud de marketing: Marketing desea un extracto de los datos con los ingresos durante las temporadas de primavera y verano de los días en los que solo los usuarios ocasionales alquilan bicicletas. Quieren un pequeño archivo CSV con solo la cantidad de usuarios ocasionales y los ingresos agrupados por temporada.

Filtrado de filas de datos

Filtrarás filas con dplyr utilizando la función filter() para extraer un subconjunto de filas que cumplan con los criterios definidos con operadores lógicos como los que se muestran en la siguiente tabla. Puedes obtener más información al respecto escribiendo ?Comparison o ?base::Logic en la consola R:

Logical Operators in the R Environment			
<	Less than	>	Greater than
<=	Less than or equal to	>=	Greater than or equal to
==	Equal to	!=	Not equal to
%in%	Group membership	&, |, !, xor, any, and all	Boolean operators
is.na	Is NA	!is.na	Is not NA

Puedes utilizar estos operadores para pasar un criterio, o muchos criterios, a tu filtro. A Marketing le gustaría saber cuántas veces durante la primavera o el verano solo los usuarios ocasionales alquilaron bicicletas. Puedes comenzar creando un subconjunto de los datos utilizando la función filter() junto con el operador == y el operador *or* booleano (|). Coloca los resultados en un frame de datos temporal de filas extraídas (extracted_rows):

```
> library(dplyr)
> extracted_rows <- filter(bike, registered == 0,
+   season == 1 | season == 2)
```

Obtenemos el siguiente resultado:

```
> dim(extracted_rows)
[1] 10 12
```

La función dim() muestra que solo 10 observaciones cumplen con los criterios de filtro. Esto demuestra el poder de filtrar conjuntos de datos más grandes.

Hay varias formas de transformar los datos. Puedes crear un conjunto de datos idéntico utilizando el operador %in%. Este operador analiza cada fila (observación) y determina si es miembro del grupo según los criterios que especifiques. El primer parámetro es el nombre del frame de datos, el segundo y los parámetros sucesivos son expresiones de filtrado:

```
> using_membership <- filter(bike, registered == 0, season %in% c(1, 2))
> identical(extracted_rows, using_membership)
```

Obtenemos el resultado de la siguiente manera:

```
[1] TRUE
```

La función identical() compara dos objetos R y devuelve TRUE si son idénticos y FALSE en caso contrario. Creaste un subconjunto de datos filtrando filas y guardándolo en un frame de datos separado. Ahora puedes seleccionar columnas de ese subconjunto.

Selección de columnas de datos

La función select() extrae las columnas que deseas conservar en tu conjunto de datos final. El equipo de marketing indicó que solo estaban interesados en la temporada (season) y los inquilinos ocasionales (casual renters). No expresaron interés en las condiciones ambientales

o las vacaciones. Proporcionar a los miembros del equipo productos de datos que cumplan con sus especificaciones es una forma importante de mantener las relaciones.

Puedes extraer las columnas requeridas de las filas extraídas y guardarlas en otro frame de datos temporal de columnas extraídas. Pasa la función select(), el frame de datos y los nombres de las columnas que deseas extraer, season y casual:

```
> extracted_columns <- select(extracted_rows, season, casual)
```

La siguiente tabla ofrece una vista de las dos primeras observaciones del frame de datos del subconjunto que generaste. Observa que falta algo. El departamento de marketing desea conocer la cantidad de inquilinos ocasionales y los ingresos por temporada. No hay ninguna variable de ingresos en los datos que estás utilizando. ¿Qué puedes hacer al respecto? Puedes agregar una columna al frame de datos, como se describe en la siguiente sección:

	season	Casual
1	1	2
2	1	1

Agregar una columna calculada a partir de datos existentes

Para su situación particular, agregarás una columna calculada. Preguntas al departamento de marketing sobre la estructura de los costos de alquiler y se enteró de que los inquilinos ocasionales pagan cinco dólares por un pase diario. Descubrió que todo lo que tiene que hacer es multiplicar la cantidad de inquilinos ocasionales por cinco para obtener los ingresos por cada día en su frame de datos.

La función mutate() calculará y agregará una o más columnas, según los parámetros. Sus parámetros incluyen el frame de datos y una expresión que indica el nombre de la nueva columna y el cálculo para crear los ingresos:

```
> add_revenue <- mutate(extracted_columns, revenue = casual * 5)
```

El resultado será el siguiente:

```
> add_revenue
```

Puedes ver la imagen en la siguiente página.
¡Perfecto! Ya casi has terminado. Lo único que te queda por hacer es agrupar y resumir los datos en un frame de datos final.

	season	casual	revenue
1	1	2	10
2	1	1	5
3	1	4	20
4	1	1	5
5	1	1	5
6	1	1	5
7	1	1	5
8	1	3	15
9	2	3	15
10	2	1	5

Agregar datos en grupos

El paquete dplyr te proporciona las funciones group_by() y summarise() para ayudarte a agregar datos. A menudo verás que estas dos funciones se usan juntas. La función group_by() toma el frame de datos y la variable en la que te gustaría agrupar los datos como parámetros, en tu caso, season:

```
> grouped <- group_by(add_revenue, season)
```

La función summarise() toma el frame de datos y todas las variables que deseas resumir como parámetros. Esto también requiere que especifiques cómo deseas resumirlas. Puedes elegir un promedio, un mínimo, un máximo o una suma. El departamento de marketing desea conocer los alquileres totales y los ingresos por temporada, por lo que utilizarás la función sum():

```
> report <- summarise(grouped, sum(casual), sum(revenue))
```

Obtenemos el resultado de la siguiente manera:

```
> report
# A tibble: 2 × 3
  season `sum(casual)` `sum(revenue)`
   <int>         <int>          <dbl>
1      1            14             70
2      2             4             20
```

Parece que esto es lo que quiere el grupo de marketing. Ahora tienes que entregarlo.

Consejo de R: El paquete dplyr ofrece muchas otras funciones de transformación. Puedes escribir ??dplyr en tu consola R para obtener más información.

Carga de datos en sistemas empresariales para tu análisis

Has importado y transformado datos. Estos se encuentran dentro de tu entorno R como un frame de datos. Ahora tendrás que proporcionárselos al departamento de marketing. Las siguientes son dos formas habituales de exportar datos desde R a un archivo para su uso en otras partes de una organización:

- Escribir datos en un archivo CSV
- Escribir datos en un archivo de texto delimitado por tabulaciones

Estos métodos son similares, pero producen resultados diferentes. Conocerlos y conocer sus diferencias te ayudará a decidir el formato que te gustaría utilizar.

Escritura de datos en un archivo CSV

Los archivos CSV son comunes entre las aplicaciones de datos. Otras aplicaciones de datos, como Excel, pueden leer este tipo de archivos. Los archivos CSV también son útiles porque los sistemas de bases de datos normalmente pueden importarlos a su entorno, de la misma manera que importaste un CSV al entorno de R.
La función write.csv() se utiliza para escribir un frame de datos en un archivo CSV. En este ejemplo, los parámetros de entrada incluyen report y el nombre del archivo de salida, revenue_report.csv:

```
> write.csv(report, "revenue_report.csv", row.names = FALSE)
```

También utilizaste un parámetro row.names = FALSE. Muy a menudo, tu conjunto de datos no contendrá nombres de fila. Este parámetro evita que R agregue una columna de identificadores numéricos al archivo CSV. Hay muchos otros parámetros que puedes usar con write.csv(). Obtén más información sobre ellos escribiendo ?write.csv en la consola de R.

Escritura de datos en un archivo de texto delimitado por tabulaciones

Puede haber ocasiones en las que desees que una aplicación de datos que no importe archivos CSV lea tus datos. Recuerda que en la sección Extracción de datos de las fuentes, read.csv() tenía una contraparte más flexible, read.table(). La función write.table() te brinda mayor flexibilidad sobre cómo se compone el archivo final:

```
> write.table(report, "revenue_report.txt", row.names = FALSE, sep = "\t")
```

La función write.table() utiliza una sintaxis que es muy similar a write.csv(). Verás la adición de sep = "\t". Esto le indica a R que separe los datos con el carácter de tabulación al crear el archivo de texto. Hay muchos otros parámetros que puedes usar con write.table().
Obtén más información sobre ellos escribiendo ?write.table en la consola de R.

Si revisas la carpeta Documentos, podrás constatar que ahí se encuentran tus archivos creados.

Resumen

Completaste el primer paso de tu recorrido y deberías estar satisfecho contigo mismo, ya que la ETL no es trivial ni simple. En este documento, aprendiste a ejecutar un proceso ETL de principio a fin. Esto incluye saber cómo extraer datos de múltiples fuentes, incluidas las bases de datos. Luego, transformaste los datos con el paquete dplyr y exportaste la información a un archivo que se puede cargar en un sistema empresarial.

En el próximo documento, aprenderás a limpiar los datos una vez que se cargan. Verás que, al igual que la ETL, la limpieza de datos es un aspecto amplio y multifacético de la inteligencia y el análisis de negocios.