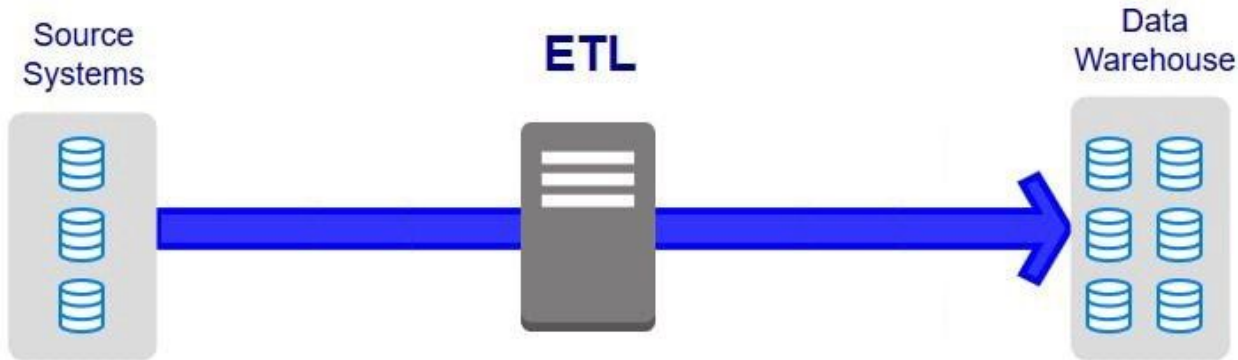




# Data Processing

# Transformación

Es el proceso de mapear la información desde la estructura utilizada en el origen a la estructura utilizada en las etapas de almacenamiento.



# Transformación

**Hay muchos tipos de transformaciones, entre ellas existe:**

- Conversión de tipos de datos: string a integer
- Substitución de datos faltante por un tipo de dato default
- Filtrar datos que rompen reglas de negocio: el valor de un año que sea entre 1990 y 2030
- Eliminar columnas que no son necesarias

# Transformación

**Se puede realizar transformaciones desde diferentes lenguajes:**

- SQL
- Python
- Scala
- Java
- Entorno gráfico

# Ejemplos

## Fechas

```
>>> from datetime import datetime
>>> now = datetime.now()
>>> print(now.strftime("%c"))
Mon Jun 13 08:30:45 2022
>>> print(now.strftime("%x"))
06/13/22
>>> print(now.strftime("%X"))
08:30:45
```

	orderNumber	orderDate	requireddate	shippedDate
▶	10100	2003-01-06	Mon 13th Jan 2003	Friday 10th January 2003
	10101	2003-01-09	Sat 18th Jan 2003	Saturday 11th January 2003
	10102	2003-01-10	Sat 18th Jan 2003	Tuesday 14th January 2003
	10103	2003-01-29	Fri 7th Feb 2003	Sunday 2nd February 2003
	10104	2003-01-31	Sun 9th Feb 2003	Saturday 1st February 2003
	10105	2003-02-11	Fri 21st Feb 2003	Wednesday 12th February 2003
	10106	2003-02-17	Mon 24th Feb 2003	Friday 21st February 2003

# Ejemplos

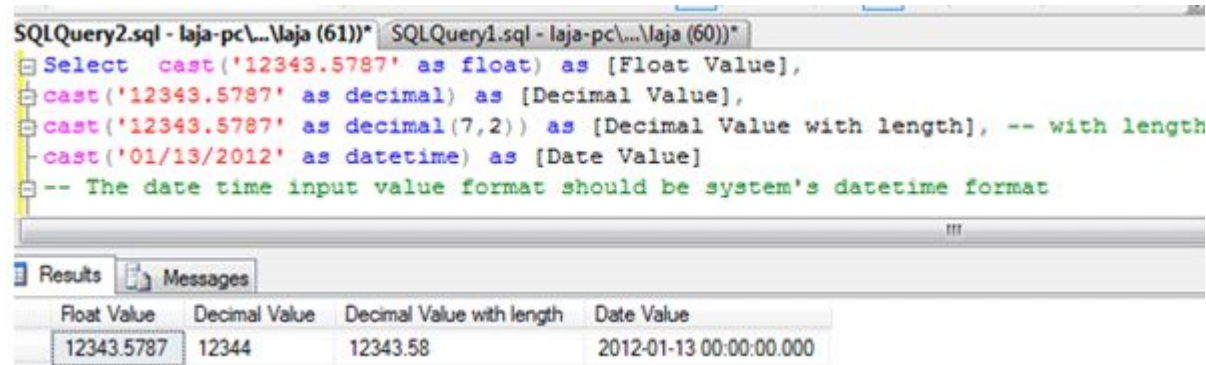
## String -> int

```
INSERT INTO sto_employees  
VALUES (8, 'Jay', CAST ('36' AS int) ,5500, '2017-08-23');
```

```
>>> age = "40"  
>>> print(type(age))  
<class 'str'>  
>>> age = int(age)  
>>> print(type(age))  
<class 'int'>  
>>> █
```

# Ejemplos

String -> float



The screenshot shows a SQL query window with the following text:

```
SQLQuery2.sql - laja-pc\...\laja (61))* SQLQuery1.sql - laja-pc\...\laja (60))*
Select cast('12343.5787' as float) as [Float Value],
cast('12343.5787' as decimal) as [Decimal Value],
cast('12343.5787' as decimal(7,2)) as [Decimal Value with length], -- with length
cast('01/13/2012' as datetime) as [Date Value]
-- The date time input value format should be system's datetime format
```

Below the query, the 'Results' tab is active, displaying a table with the following data:

Float Value	Decimal Value	Decimal Value with length	Date Value
12343.5787	12344	12343.58	2012-01-13 00:00:00.000

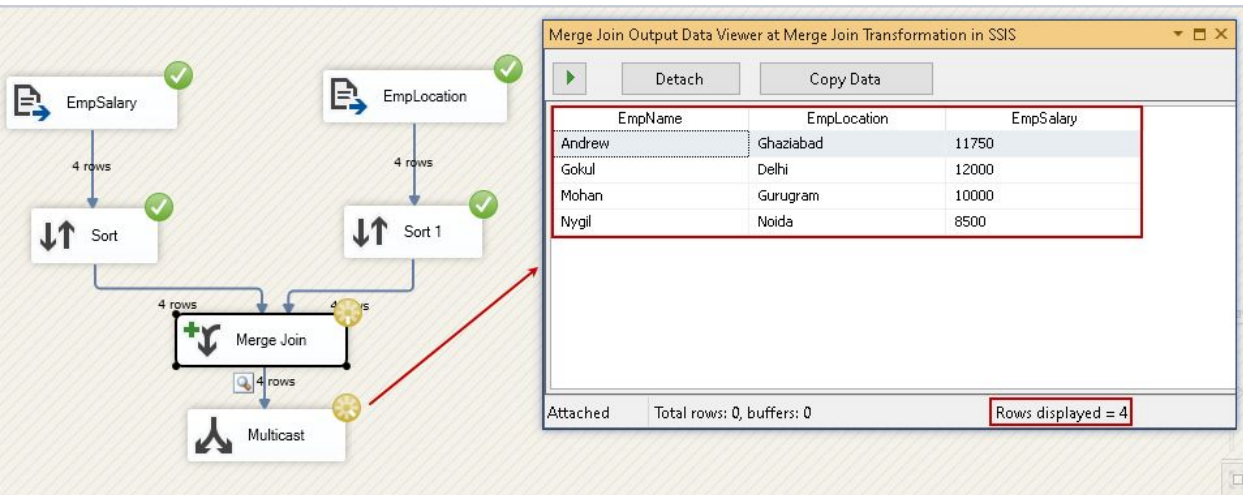
```
>>> value = "3.14159"
>>> print(type(value))
<class 'str'>
>>> value = float(value)
>>> print(type(value))
<class 'float'>
>>> print(value)
3.14159
>>>
```

# Ejemplos

## Join tablas

REAL \* IA PARA UN MUNDO

```
SELECT
    C.FirstName AS FName,
    C.LastName AS LName,
    I.InvoiceDate,
    I.Total,
    IL.Quantity,
    IL.UnitPrice
FROM CUSTOMER AS C
INNER JOIN INVOICE AS I ON C.CUSTOMERID = I.CUSTOMERID
INNER JOIN INVOICELINE AS IL ON I.INVOICEID = IL.INVOICEID;
```







# Apache Spark

# Apache Spark

- ¿Qué es?
- ¿En qué lenguaje se puede desarrollar?
- Open Source vs Cloud
- Ejemplo Transformación

# Apache Spark

## ¿Qué es?

- Es un sistema de procesamiento distribuido open-source, usado para workloads de big data.
- Utiliza caching en memoria, lo que permite ejecutar queries optimizadas para analytics
- Permite pipelines en batch o streaming data

# Apache Spark

**¿En qué lenguaje se puede desarrollar?**

Provee desarrollo de APIs en:

# Apache Spark

**¿En qué lenguaje se puede desarrollar?**

Provee desarrollo de APIs en:

- Java

# Apache Spark

**¿En qué lenguaje se puede desarrollar?**

Provee desarrollo de APIs en:

- Java
- Scala
- Python
- R

# Apache Spark

## Open Source vs Cloud

- Dataproc - GCP
- Amazon Elastic MapReduce (EMR) - AWS
- HDInsight - Azure
- Databricks

# Apache Spark

## Ejemplo Transformación

```
val parquetFileDF = spark.read.parquet("employee.parquet")
parquetFileDF.createOrReplaceTempView("parquetFile")
val namesDF = spark.sql("SELECT name FROM parquetFile WHERE age BETWEEN 18 AND 30")
namesDF.map(attributes => "&&&> Name: " + attributes(0)).show()
```

```
//Filter multiple condition
df.filter( (df.state == "OH") & (df.gender == "M") ) \
    .show(truncate=False)
```

```
+-----+-----+-----+
|name          |languages          |state|gender|
+-----+-----+-----+
|[James, , Smith] |[Java, Scala, C++] |OH   |M     |
|[Mike, Mary, Williams] |[Python, VB]       |OH   |M     |
+-----+-----+-----+
```





# Apache Beam/Flink

# Apache Beam

- ¿Qué es?
- ¿En qué lenguaje se puede desarrollar?
- Open Source vs Cloud
- Ejemplo

# Apache Beam

## ¿Qué es?

- Apache Flink and Apache Beam son frameworks open-source para procesamiento distribuido en paralelo a gran escala
- Beam no viene con un motor de ejecución propio, se conecta a otros motores de ejecución (Apache Flink, Apache Spark o Google Cloud Dataflow)
- Permite pipelines en batch, preferentemente streaming data

# Apache Beam

**¿En qué lenguaje se puede desarrollar?**

Provee desarrollo de APIs en:

# Apache Beam

**¿En qué lenguaje se puede desarrollar?**

Provee desarrollo de APIs en:

- Java
- Python
- Go
- SQL

# Apache Beam

## Open Source vs Cloud

- Dataflow - GCP



# Apache Beam

## Open Source vs Cloud

- Dataflow - GCP
- Amazon Kinesis - AWS
- Azure Stream Analytics - Azure

# Apache Beam

## Ejemplo

```
import apache_beam as beam
from apache_beam.options.pipeline_options import PipelineOptions

class LongitudPalabraFn(beam.DoFn):
    def process(self, element):
        return [len(element)]

palabras = ['Hola', 'Mundo']

with beam.Pipeline(options=PipelineOptions()) as p:
    longs = p | beam.Create(palabras) \
              | beam.ParDo(lambda palabra: [len(palabra)]) \
              | beam.Map(print)
```

4

5





CDAP

# CDAP

- ¿Qué es?
- ¿En qué lenguaje se puede desarrollar?
- Open Source vs Cloud
- Ejemplo Transformación

# CDAP

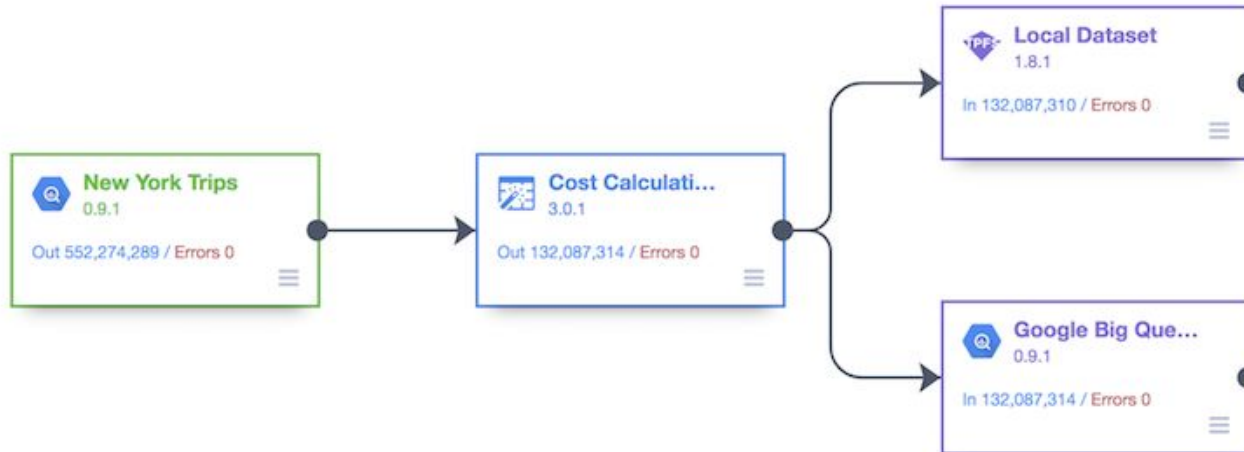
## ¿Qué es?

- Es un framework de datos para crear y administrar análisis de datos en entornos híbridos y de múltiples clouds integrada con el ecosistema Hadoop
- Permite crear pipelines, limpieza y preparación de datos, crear reglas de transformaciones y machine learning de una forma gráfica
- Permite pipelines en batch y streaming data

# CDAP

## ¿En qué lenguaje se puede desarrollar?

- Se crean los pipelines en modo gráfico



# CDAP

## Open Source vs Cloud

- Datafusion - GCP
- Aws Glue - AWS
- Cask CDAP on Azure HDInsight - Azure

## Ejemplo

CDAP Control Center Preparation Pipelines **Analytics** Rules Metadata

Hub Namespace default Local Sandbox

Create a new experiment

File System sales.tsv

Update Model More

	Double price	String city	String zip	String type	String beds	String baths	Double sqft	Double lot_sqft	String stories	Integer year_built	
1	Parse	94041	Single-Family Home	3			1176.0	6000.0	1	1920	
2	Set character encoding	94306	Single-Family Home	2			1160.0	5250.0	1	1923	
3	Change data type	94062	Single-Family Home	4	3		1930.0	10710.0	1	1928	
4	Format	94301	Condo	2	2.5		1471.0	840.0	2	1982	
5	Calculate	95054	Single-Family Home	3	2		1341.0	5985.0	1	1976	
6	Custom transform										
7	Filter				4	2	2181.0	6000.0	1	1950	
8	Send to error										
9	Find and replace				Home	4	2	2500.0	0.25	2	2001
10	Fill null or empty cells				Home	3	2	1699.0	7749.0	1	1965
11	Copy column				Home	3	2	1575.0	8092.0	1	1955
12	Delete column				Home	4	2.5	2285.0	3960.0	2	1973
13	Keep column				Home	3		1070.0	5000.0	1	1953
14	Join two columns										
15	Swap two column names	94085	Single-Family Home	3			1056.0	6344.0	1	1953	
16	Extract fields	94063	Single-Family Home	3			970.0	3750.0		1959	
17	Explode	94303	Single-Family Home	3			1010.0	6050.0		1952	
18	Define variable										

Columns (10) Directives (6)

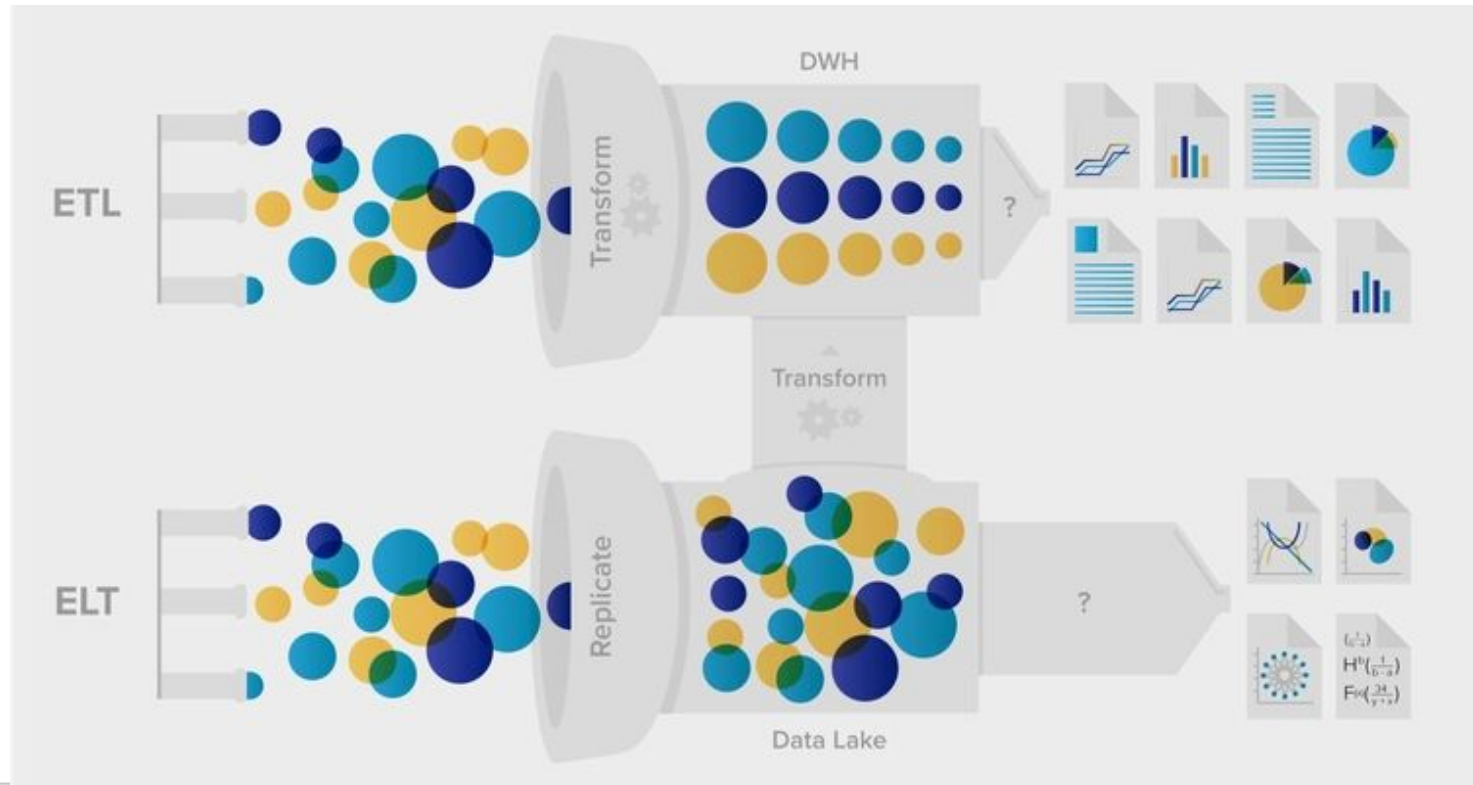
# Directives

- 1 parse-as-csv:body '\t' true
- 2 drop body
- 3 set-type:price double
- 4 set-type:sqft double
- 5 set-type:lot\_sqft double
- 6 set-type:year\_built integer



# ETL vs ELT

# ETL vs ELT





# ETL

## Ventajas

- Compliance: permite enmascarar la información debido a regulaciones
- Costos: debido a que la información es filtrada se reducen los costos en el DW

## Desventajas

- Team: debido al mantenimiento de los pipelines se requiere más Data Engineers
- Errores: debido a la complejidad de los pipelines y fuentes, mayor probabilidad

# ELT

## Ventajas

- Complejidad: sin pipelines ni transformaciones, accedes a la información fácil
- Velocidad: la información es ingestada rápidamente ya que no hay pipelines

## Desventajas

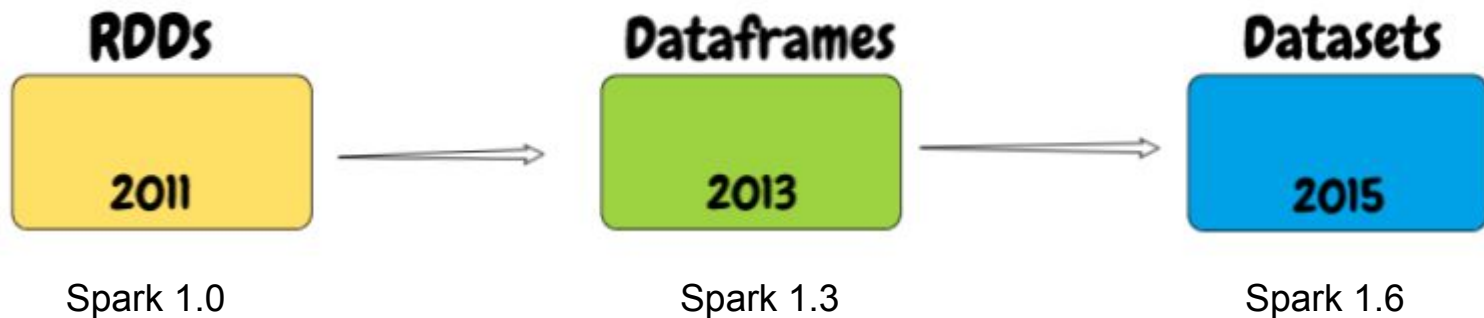
- Riesgo: debido a que está toda la data en el DW, fácilmente la podrían visualizar
- Crecimiento sin control: sin filtrar datos se incrementa el tamaño fácilmente



# Dataset, Dataframe y RDDs

# Resilient Distributed Datasets, Dataframe y Dataset

REAL \* IA PARA  
UN MUNDO





# Resilient Distributed Datasets(RDD)

- Es una colección de datos distribuidos en muchas máquinas en el clúster.
- Puede procesar datos tanto estructurados como no estructurados.
- No infiere el esquema de los datos ingeridos y requiere que el usuario lo especifique. Disponible en Java, Python, Scala, R
- Se puede generar un RDD desde cualquier fuente de datos (archivo de texto, una base de datos a través de JDBC, etc. )

# Dataframe

- Colección distribuida de datos organizados en columnas. Puede procesar datos estructurados y semiestructurados. Disponible en Java, Python, Scala, R

# Dataframe

- Colección distribuida de datos organizados en columnas. Puede procesar datos estructurados y semiestructurados. Disponible en Java, Python, Scala, R
- Los Dataframes permiten que Spark administre el esquema.
- Permite el procesamiento de datos en diferentes formatos (AVRO, CSV, JSON, HDFS, tablas HIVE , MySQL).
- Si generas un DF desde RDD, no es posible recuperar el RDD original

# Dataset

- Combinación de RDD y DataFrame. Solamente disponibles en Java y Scala
- También admite datos de diferentes fuentes.
- Procesa datos estructurados y no estructurados.
- Representa datos en forma de objetos JVM una colección de objetos de filas.



# Cuando usar RDDs

- Data no estructurada
- Las transformaciones son de bajo nivel
- Esquema no es importante
- Tolerancia a fallos. Los RDD son resistentes y pueden volver a calcular las particiones faltantes o dañadas para una recuperación completa si falla un nodo.

# Cuando usar Dataframes/Datasets



- Los datos requieren una estructura: infieren un esquema sobre datos estructurados y semiestructurados.
- Las transformaciones son de alto nivel: requieren un procesamiento de alto nivel y consultas SQL
- Es necesario un alto grado de seguridad de tipo. Dando velocidad de desarrollo y eficiencia



# Data Processing