

# Curso Data Engineer: Creando un pipeline de datos

MODULO C - Clase 6

# Agenda



- Transformaciones en Pyspark
- DataWarehouse
- Hive
- Creación de Tablas en Hive
- Load en Hive



# Transformaciones en Pyspark

# Transformaciones



## SQL- Creación de Vistas

```
df.createOrReplaceTempView("vtripdata")
```

# Transformaciones



## SQL- Filtros

```
df_transform = spark.sql("select * from vtripdata where fare_amount > 10")
```

# Transformaciones



## SQL- Cast

`cast(tpep_pickup_datetime as timestamp) -> Timestamp`

`cast(tpep_pickup_datetime as date) -> Date`

`cast(passenger_count as integer) → Integer`

# Transformaciones



## PYSPARK - SELECT

```
df_2 = df.select(df.forename, df.surname, df.nationality, df.points.cast("int"))
```

# Transformaciones



## PYSPARK - JOIN

```
dfjoin = df.join(df2, df1.userId == df2.userId, "inner")
```

```
dfjoin = df.join(df2, df1.userId == df2.userId, "left")
```

```
dfjoin = df.join(df2, df1.userId == df2.userId, "right")
```



# Transformaciones



## PYSPARK - Group By

**IMPORTANTE:** antes del group by debemos importar algunas funciones:

```
from pyspark.sql.functions import sum, asc, desc
```

```
dfgroup = df.groupBy(df.forename, df.surname).agg(sum(df.points).alias("points")).sort(desc("points"))
```

# Transformaciones



## PYSPARK - FILTER

```
dffilter = df.filter( (df.airport_fee > 0) & (df.payment_type == 2) )
```

# Transformaciones



## PYSPARK - UNION

```
dfunion = df1.union(df2)
```

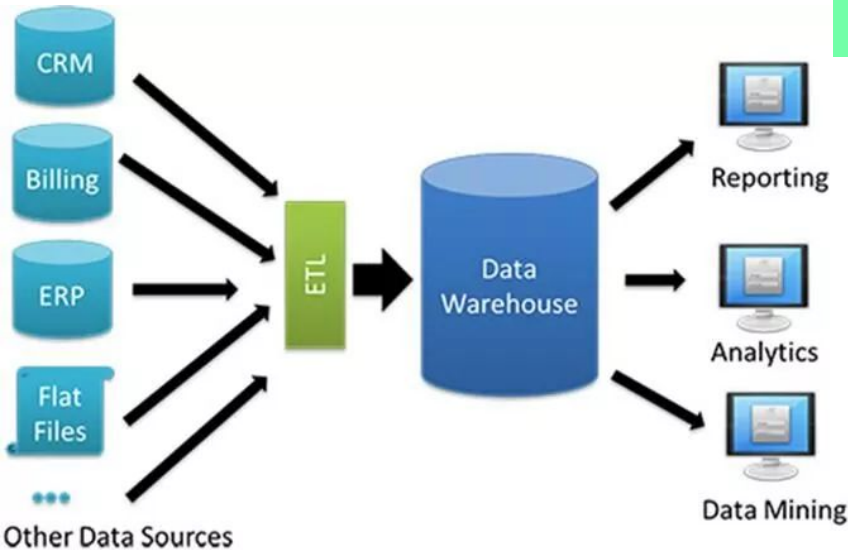


# Data Warehouse



# Data warehouse

Es un depósito central de información que se puede analizar para tomar decisiones más informadas. Los datos fluyen hacia un almacén de datos desde sistemas transaccionales, bases de datos relacionales y otras fuentes, normalmente con una cadencia regular.



# Data warehouse

- **Orientado a un tema:** pueden analizar datos sobre un tema o área funcional en particular (como marketing).
- **Integrado:** los almacenes de datos crean consistencia entre diferentes tipos de datos de fuentes dispares.
- **No volátil:** una vez que los datos están en un almacén de datos, son estables y no cambian.
- **Variable en el tiempo:** El análisis del almacén de datos analiza los cambios a lo largo del tiempo.



# Hive

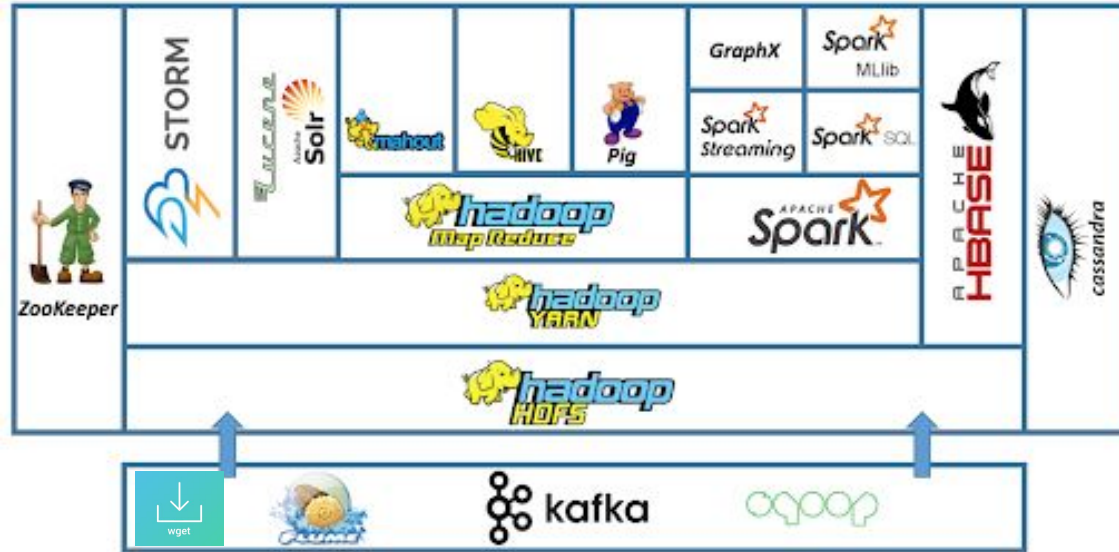
# Hive



- Es un data warehouse distribuido, tolerante a fallas que permite análisis a gran escala.
- Permite a los usuarios leer, escribir y administrar petabytes de datos mediante SQL.
- Se monta sobre Apache Hadoop.



# Ecosistema Hadoop





# Creación de Tablas en Hive

# Creación de Tablas



Cuando creamos tablas en Hive se pueden crear internas o externas

## Internas

- Las tablas internas predeterminadas se almacenan en el siguiente directorio `"/user/hive/warehouse"`. Puede cambiarlo actualizando la ubicación en el archivo de configuración.
- Al eliminar la tabla, se eliminan los metadatos y los datos del nodo maestro y HDFS, respectivamente.
- La seguridad del archivo de la tabla interna se controla únicamente a través de HIVE.

# Creación de Tablas



## Externas

- La tabla externa almacena archivos en el servidor HDFS, pero las tablas no están completamente vinculadas al archivo de origen.
- Si elimina una tabla externa, el archivo aún permanece en el servidor HDFS.
- Los archivos de tablas externas son accesibles para cualquier persona que tenga acceso a la estructura de archivos HDFS y, por lo tanto, la seguridad debe administrarse a nivel de archivo/carpeta HDFS.
- Los metadatos se mantienen en el nodo maestro, y al eliminar una tabla externa de HIVE solo se eliminan los metadatos, no los datos o el archivo.

# Creación de DBs



Create database <nombre>;

# Creación de Tablas



## Interna

```
CREATE TABLE emp.employee (id int, name string, last_name string, age int )  
COMMENT 'Employee Table'  
ROW FORMAT DELIMITED  
FIELDS TERMINATED BY ';;'
```

# Creación de Tablas



## Externas

```
CREATE EXTERNAL TABLE orders.exports (order_id int, customer_id string, ship_country string, unit_price float, quantity int, total float)  
ROW FORMAT DELIMITED  
FIELDS TERMINATED BY ','  
LOCATION '/tables/external/orders';
```



Load data



# Load



## SQL

```
df.createOrReplaceTempView("nombre_vista")
```

```
spark.sql("insert into db.tabla select * from nombre_vista")
```

# Load



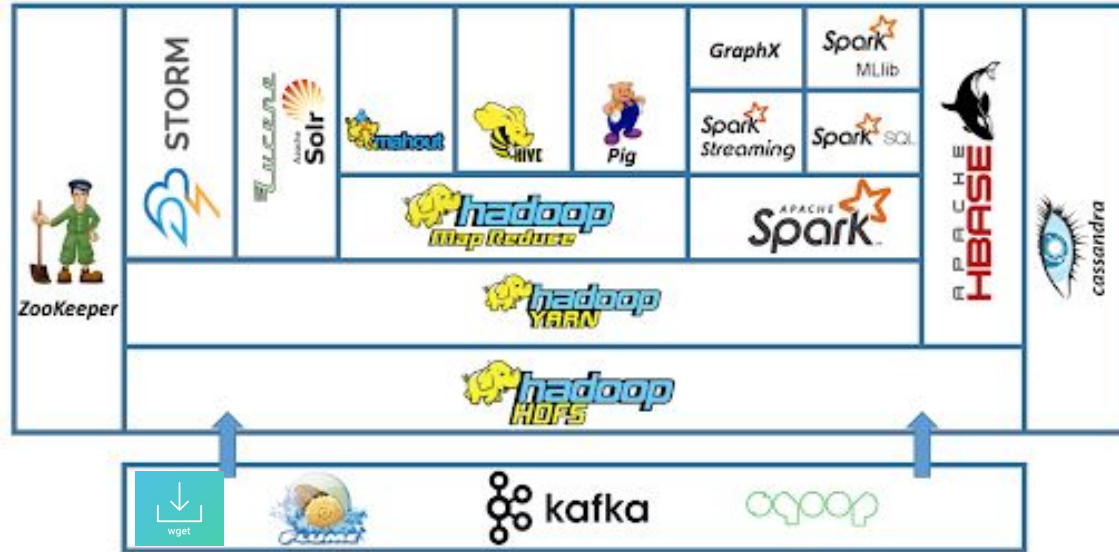
## Pyspark

```
df.write.insertInto("db.table")
```



# Ejercicio

# Ecosistema Hadoop



# Ejercicios



- HIVE
  - Creación de DBs
  - Creación de Tablas internas/externas
  - Describe
  - Uso de SQL

# Load

[https://edvaibucket.blob.core.windows.net/data-engineer-edvai/yellow\\_tripdata\\_2021-01.csv?sp=r&st=2023-11-06T12:52:39Z&se=2025-11-06T20:52:39Z&sv=2022-11-02&sr=c&sig=J4Ddi2c7Ep230hQLPisbYaerlH472iigPwc1%2FkG80EM%3D](https://edvaibucket.blob.core.windows.net/data-engineer-edvai/yellow_tripdata_2021-01.csv?sp=r&st=2023-11-06T12:52:39Z&se=2025-11-06T20:52:39Z&sv=2022-11-02&sr=c&sig=J4Ddi2c7Ep230hQLPisbYaerlH472iigPwc1%2FkG80EM%3D)



**Extract**



**Transform**



**Load**