Build Data Warehouse with Hadoop, Hive, Spark

Group 8

Bùi Đoàn Gia Phong
Nguyễn Đắc Hoàng Phú

Topic Covered

- **Problem Statement**
- **Technology**
- **Setup**
  - Data
  - Script
  - Flow
- **Live Demo**
- **Reference**

Problem Statement

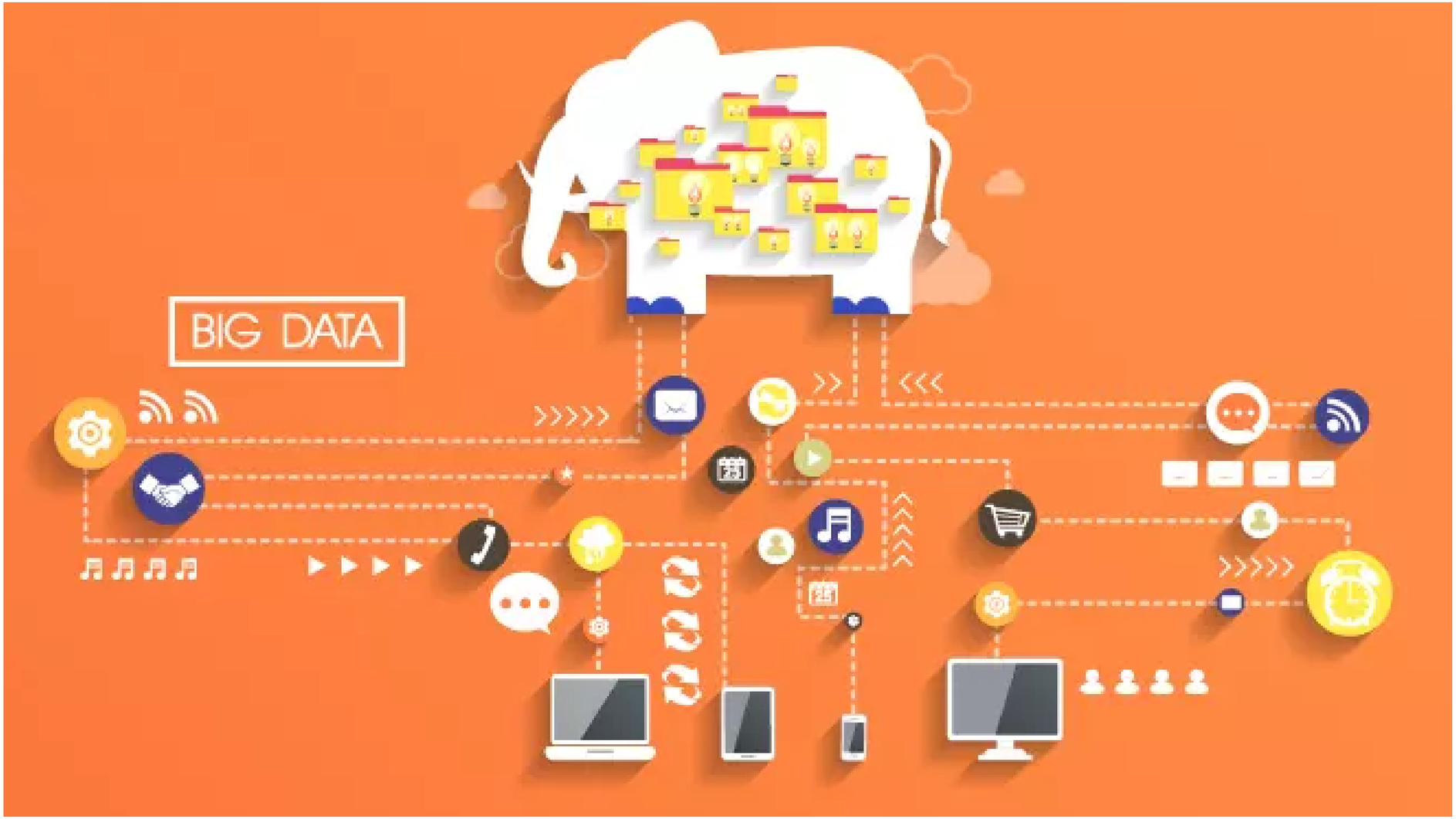
- The car sales business is growing, currently the database only contains information about the number of orders per day and related products and buyers.

- You need a better understanding of your total sales and remaining inventory each day to make important decisions

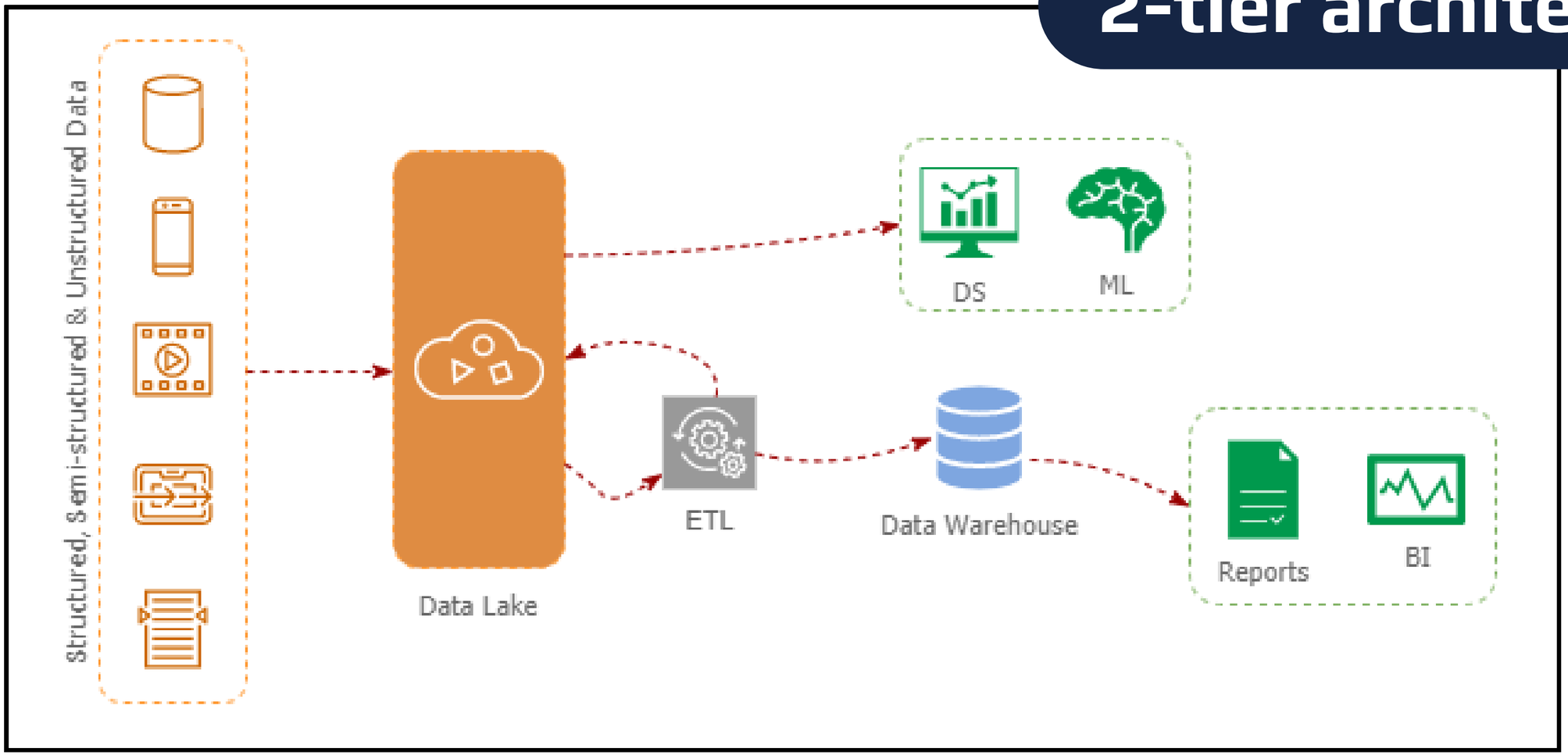
- You need a visual tool that links to the information contained in the database to capture information more easily

=> Build data warehouse

2. TECHNOLOGY



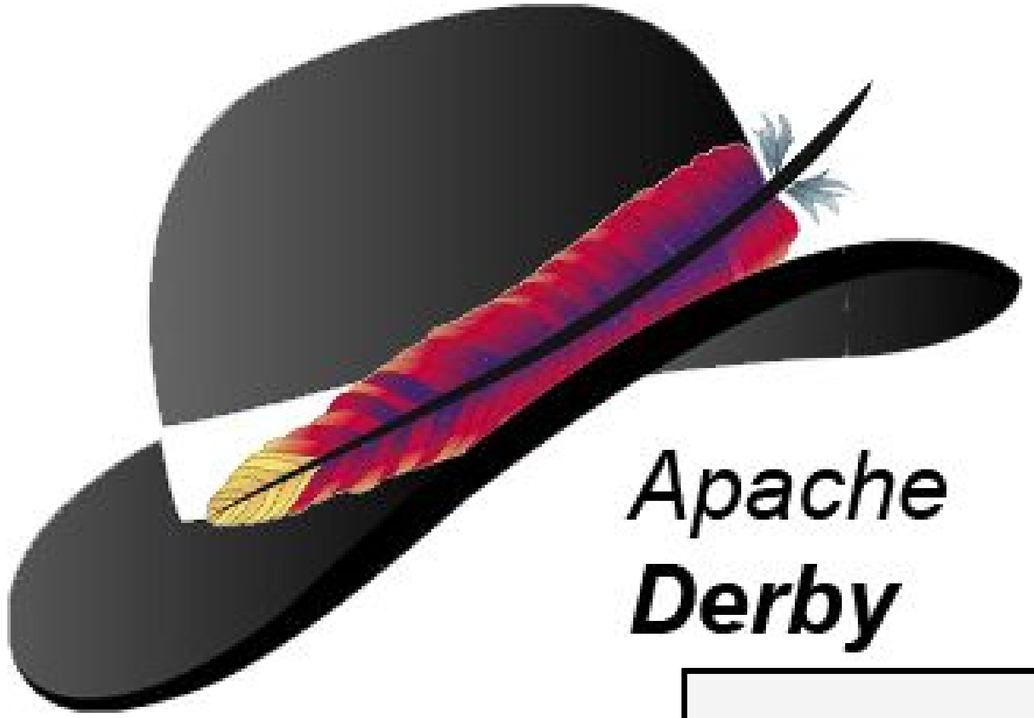
2-tier architecture



- Data lake ingest data from many source
- Thourgh ETL process, data become enrichment and load in data warehouse
- Visualization & BI tools to help find inside about data

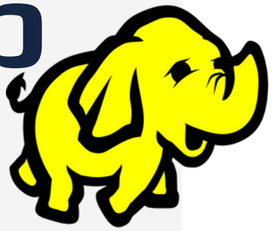
Tool - source

Store raw data

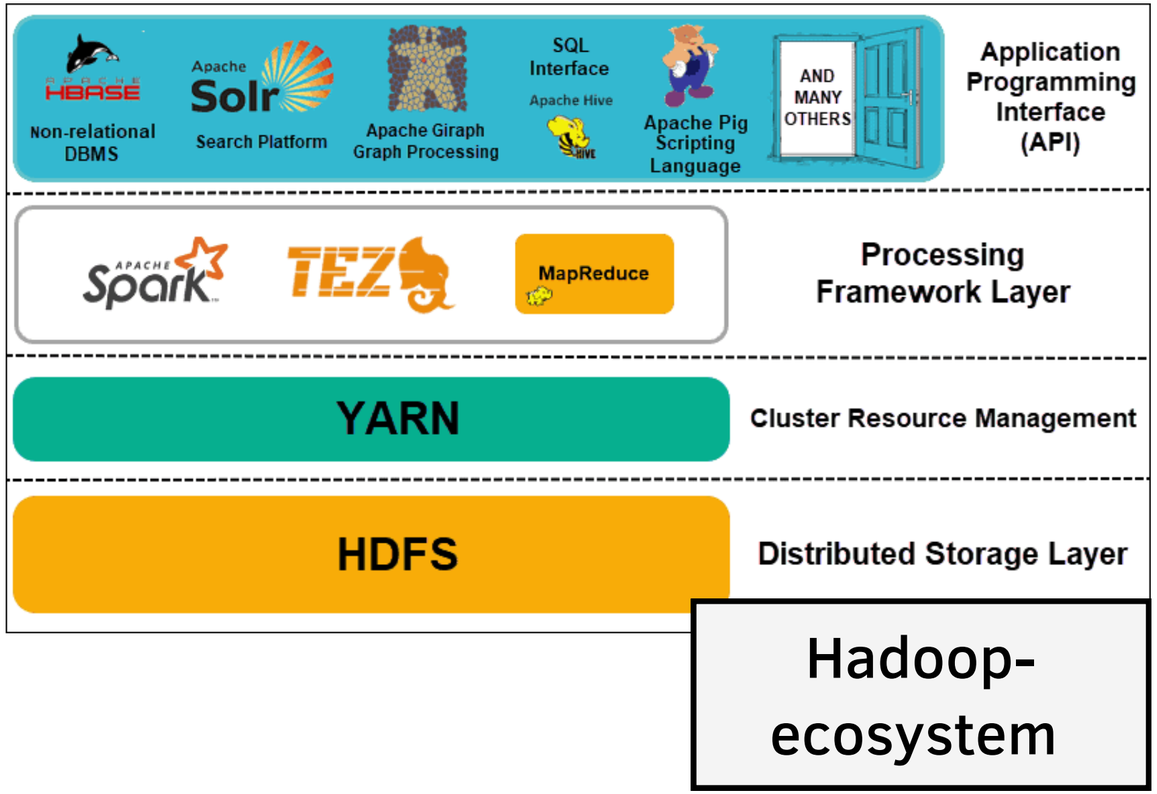


MetaStore

Apache Hadoop



- Framework that allows for the distributed processing of large data sets across clusters of computers
- Provides a parallel environment to execute Map-Reduce tasks.
- Support to integrate many other tools to create a data processing and management ecosystem

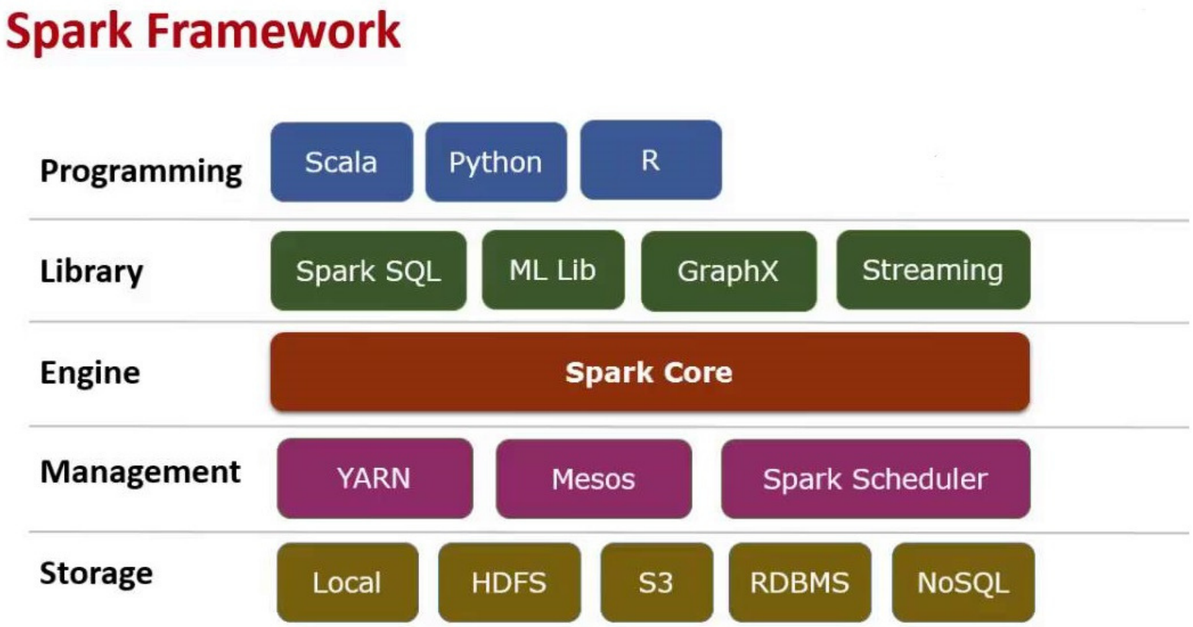


Hadoop-ecosystem

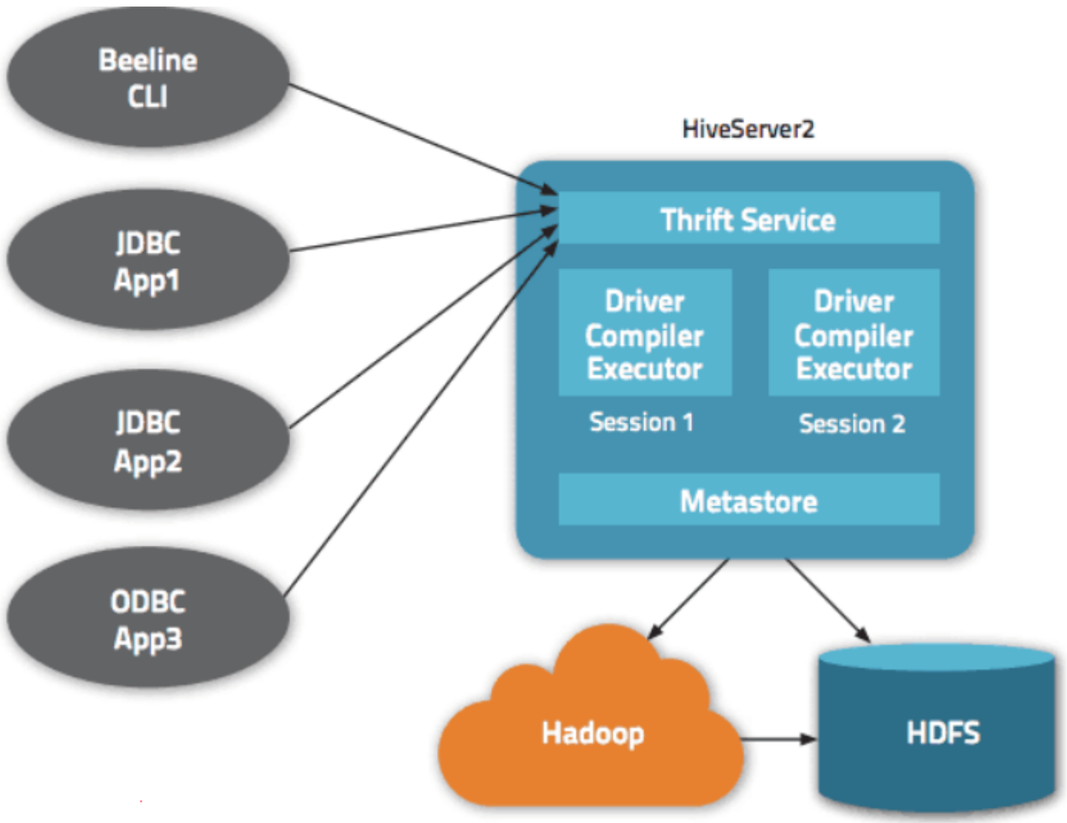
Tool - data lake

Apache Spark

- Is an open-source data processing framework
- Use in-memory computing to perform quickly processing tasks
- Includes libraries for various different tasks: MLLib, SparkSQL,..
- Well compatible with hadoop, kafka and query engines like hive, drill



Engine Processing



Apache Hive

- Structured data processing in Hadoop.
- It sits on top of Hadoop to summarize Big Data and make querying and analysis easy
- It stores the schema in the database and processes the data into HDFS, OLAP and SQL tool

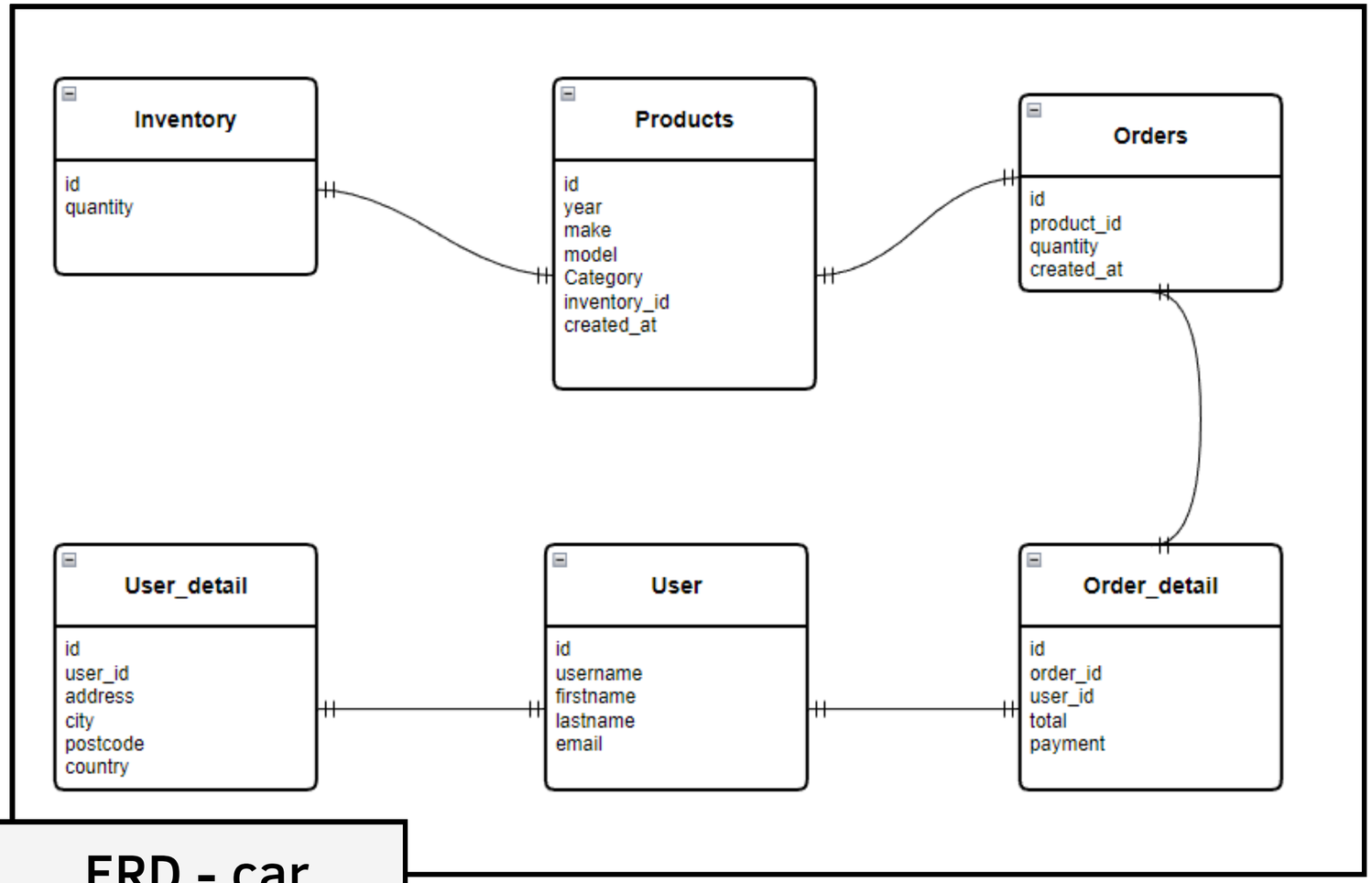
Tool - data warehouse

Tool - visualization

Apache Superset™

- Is an open-source data visualization tool, written on Python

- Support to quickly build dashboard, connect multiple data sources: mysql, hive, mongodb, drill, ..



ERD - car dealership

Database

- Use dgscli tool to generate fake data with csv format, write some sql-file to create database
- The database is designed to be simple with the main purpose of focusing on building a data processing pipeline
- 1000 record inventory, 1000 record products, 1000 record user and 10000 record orders

Script - ingest

```python
parser = argparse.ArgumentParser()
parser.add_argument("--table_name")
parser.add_argument("--exe_date")
args = parser.parse_args()
```

Param Processing

```python
tblLocation = f"hdfs://master:9000/datalake/{table_name}"
tblQuery = ""
if exists:
    df = spark.read.parquet(tblLocation)
    record_id = df.agg(max("id")).head().getLong(0)
    tblQuery = f"(SELECT * FROM {table_name} WHERE id > {record_id}) AS tmp"
else:
    tblQuery = f"(SELECT * FROM {table_name}) AS tmp"
```

Incremental Loading

```python
spark = SparkSession.builder \
        .master('local[*]') \
        .appName("ETL from MYSQL to HIVE") \
        .getOrCreate
```

Create SparkSession

```python
jdbcDF = spark.read.format("jdbc") \
    .option("url","jdbc:mysql://localhost:3306/test_demo") \
    .option("user","root") \
    .option("password","071202") \
```

Get data from MYSQL

```python
outputDF = jdbcDF.withColumn("year",lit(year)) \
                 .withColumn("month",lit(month)) \
                 .withColumn("day",lit(day))
```

Save partition to data Lake

```
spark = SparkSession.builder \
        .master('local[*]') \
        .appName("Daily Report2") \
        .config('hive.metastore.urls','thrift://localhost:9083') \
        .config('hive.exec.dynamic.partition','true') \
        .config('hive.exec.dynamic.partition.mode',"nonstrict") \
        .enableHiveSupport() \
        .getOrCreate()
```

Create spark-session

```
mapDf = preDF.groupBy("Make","Model","Category","product_id","inv_quantit
                .agg(sum("quantity").alias("Sales"),sum("total").alias("R

resultDf = mapDf.withColumn("LeftOver",col("inv_quantity") - col("Sales")
                .withColumn("year",lit(year)) \
                .withColumn("month",lit(month)) \
                .withColumn("day",lit(day)) \
                .select("Make","Model","Ca
```

GroupBy and Select

```

preDF = ordersDf.filter(ordersDf["created_at"] == exe_date) \
        .join(orderDetailDf, ordersDf["product_id"] == orderDetailDf["order_id"],
        .join(productsDf,ordersDf["product_id"] == productsDf["id"],"inner") \
        .join(inventoryDf.select(col("quantity"). \
        alias("inv_quantity"),col("id")),productsDf["inventory_id"] \
        == inventoryDf["id"],"inner")


```

Filter and Join data from Hadoop

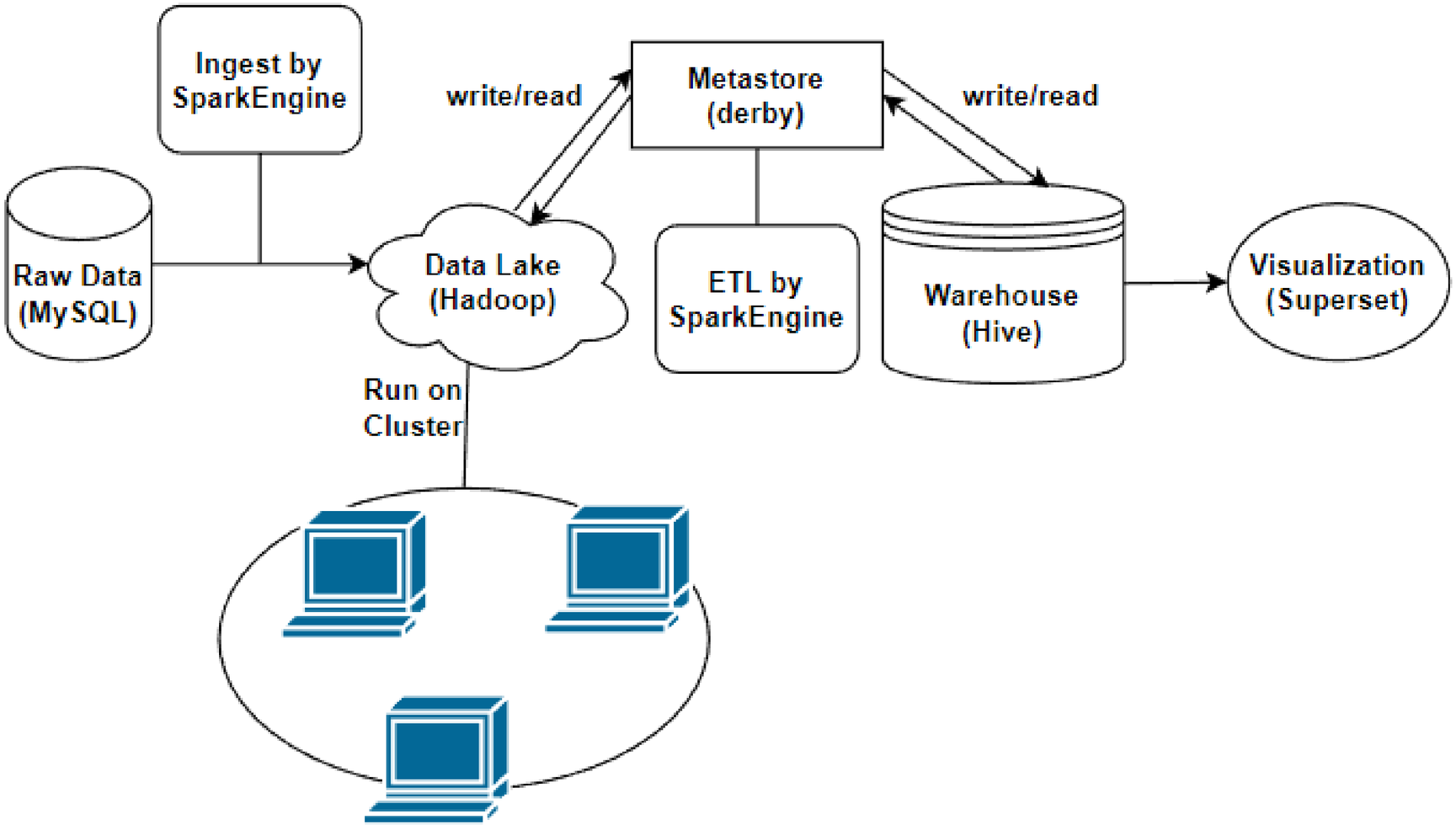
```
spark.sql("CREATE DATABASE IF NOT EXISTS reports")

resultDf.write \
        .format("hive") \
        .partitionBy("year","month","day") \
        .mode("append") \
        .saveAsTable("reports.d
```

Write to Hive

Script - ETL

Flow


Ingest by
SparkEngine
write/read
Metastore
(derby)
write/read
Raw Data
(MySQL)
Data Lake
(Hadoop)
ETL by
SparkEngine
Warehouse
(Hive)
Visualization
(Superset)
Run on
Cluster

4. Demo

5. REFERENCE

- **Source Code:** https://github.com/hoangphu7122002/datawarehouse
- **Docs Build:** https://docs.google.com/document/d/1eOzka1UHOz25-rKAljGpbeQLzjMdWyphB7iVPBYmJq0/edit?usp=sharing
- **Tool generate data:** https://github.com/canhtran/dgscli
- **Build Cluster VM:** https://www.youtube.com/watch?v=y0FSQmlTf5U