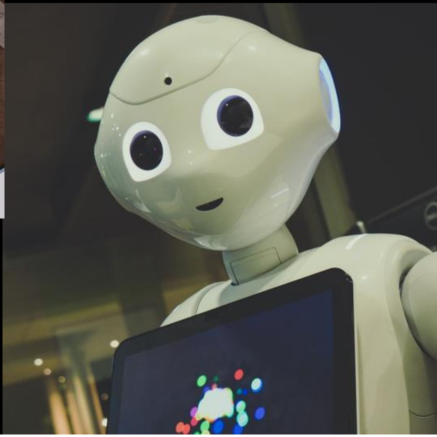
Maltem
Consulting Group

# Data Engineering and Big Data



Strictly private and confidential

# Timetable – Day 4

| Date: Thrusday, 30 September 2021 | | Time: 9:00 am to 05:00am | ITE Campus | | |
|---|---|---|---|---|---|
| Agenda | | Presenter | Time | | Duration (mins) |
| 1 | Catch up | Mahesh | 9:00 AM | 10:00 AM | 60 |
| 2 | Data Integration | Mahesh | 10:00 AM | 10:30 AM | 30 |
| 3 | Break | Mahesh | 10:15 AM | 10:45 AM | 30 |
| 4 | Pentaho | Mahesh | 10:45 AM | 12:15 PM | 90 |
| 5 | Lunch Break | Mahesh | 12:15 PM | 1:15 PM | 60 |
| 6 | Big Data & Hadoop | Ben | 1:15 PM | 2:15 PM | 60 |
| 7 | Play with Spark | Mahesh | 2:15 PM | 2:45 PM | 30 |
| 8 | Break | Mahesh | 2:45 PM | 3:15 PM | 30 |
| 9 | Assessment | Mahesh | 3:15 PM | 3:45 PM | 30 |
| 10 | Feedback | Mahesh | 3:45 PM | 4:45 PM | 60 |

# Recap of Day 3

## Data Modelling

- Key Concept of Relationships
- Introduction to data modelling
- Transaction data modelling
- Analytical data modelling
- Normalization and its types
- Denormalization
- Different types of modelling techniques
- Star schema
- Snowflake schema
- Facts
- Dimension
- SCD

# Course outline – Day 4

## Data Integration

- What is data integration
- Importance of data integration
- List of data integration tools
- ETL vs ELT

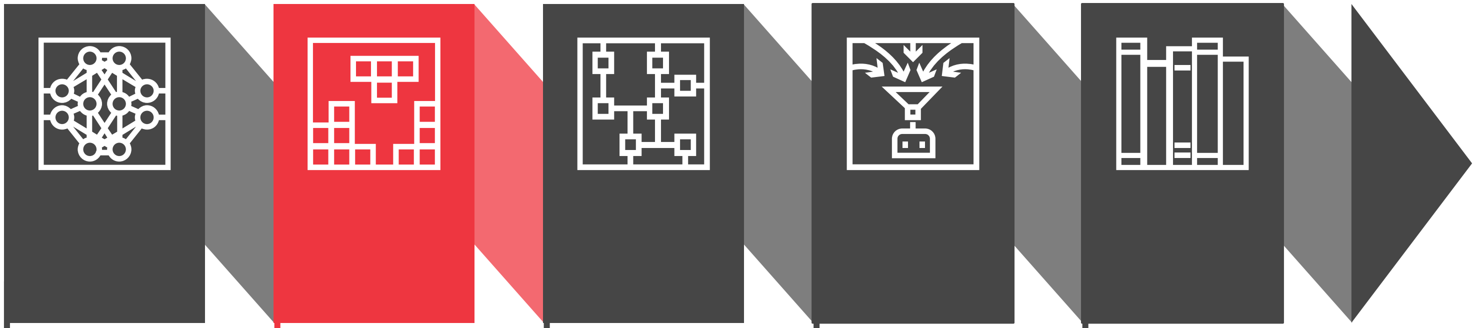
## Data Integration details

- Introduction
- Operation
- Filter, Aggregation, Sort
- Join
- Union
- Business Logic

## Big Data

- Hive
- Kafka
- Spark and demo

# Day Four



Data Modelling

**Data Integration**

- What is data integration
- Importance of data integration
- List of tools
- ETL
- ELT
- ETL vs. ELT
- Different phases of ETL

Introduction to Pentaho

Big Data

Case Study

# Importance of Data integration tools

**Need for data integration tools**

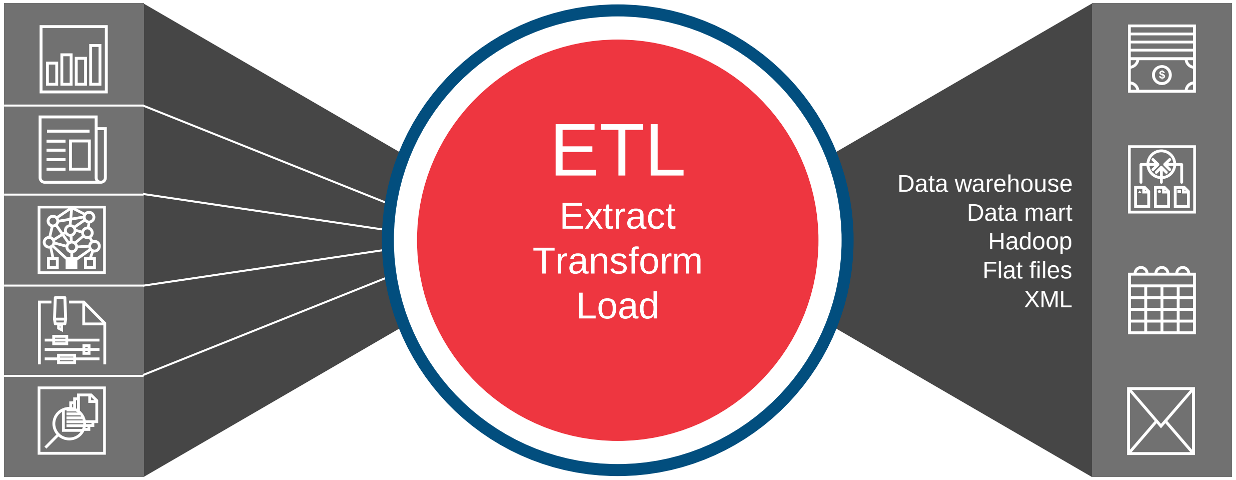
- **Data scattered** across different system
- **Improve collaboration** across departments, system and process
- **Reduce error** with human intervention

**Solutions provided by data integration tools**

- Improve **decision making** with real-time data
- Improves **customer experience** by getting 360 degree view of data by integration **data in silos**
- **Automates** the entire process by increasing the productivity
- **Security and compliance** are key concerns when dealing with data. Data integration tools have these features checked, providing its users the cushion to work on highly confidential data

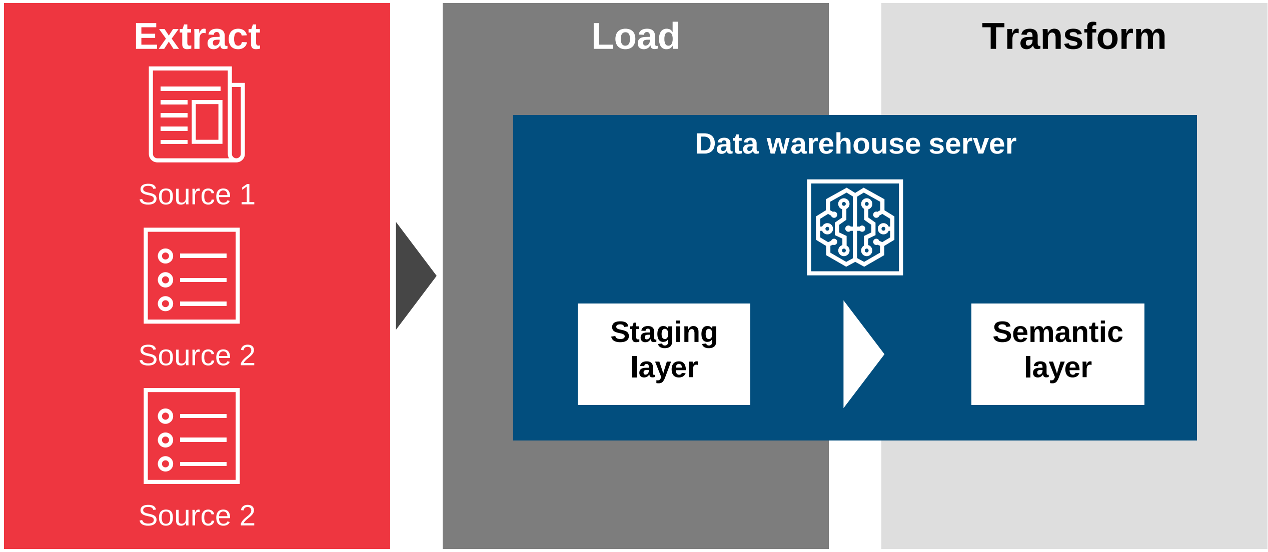
# What is ETL

- **Need:** If you want to build a data warehouse, data lake, data mart or any form of data repository, that brings in data from different sources into it, ETL is the solution
- **ETL** combines the three database functions extract, transform and load into one tool and pulls data out of one or more databases and place into the other
  - **Extract:** Process of reading data from a one or more databases. In this phase, data is collected from different sources such as Excel, CSV etc.
  - **Transform:** In this phase, data is transformed from its source format to the desired target format
  - **Load:** Is the process of writing data into the target files & databases



# What is ELT

- ELT (Extract, Load, Transform) is the variation of ETL (Extract, Transform, Load)
- ELT allows raw data to be **directly loaded** into the **target** and **then to be transformed** from there
- Used mostly to **process the large datasets** required for Big data analytics and business intelligence
- **Reduced load time**, as compared to ETL, since it need not transform the source data before loading
- Requires sufficient processing within the data processing engine to perform transformation on demand and return results to the customer within time limit



# ETL vs ELT

Compared based on the below criteria:

**Load and Transformation Time:**

**ETL:** Uses staging area and system, extra time to load data.

**ELT:** All in one system. Load only once and time not dependent on data size.

**Maintenance:**

**ETL:** High as data transform and load for enhancement of main data repository takes cost of resources.

**ELT:** Low as all data is always available.

**Implementation Complexity:**

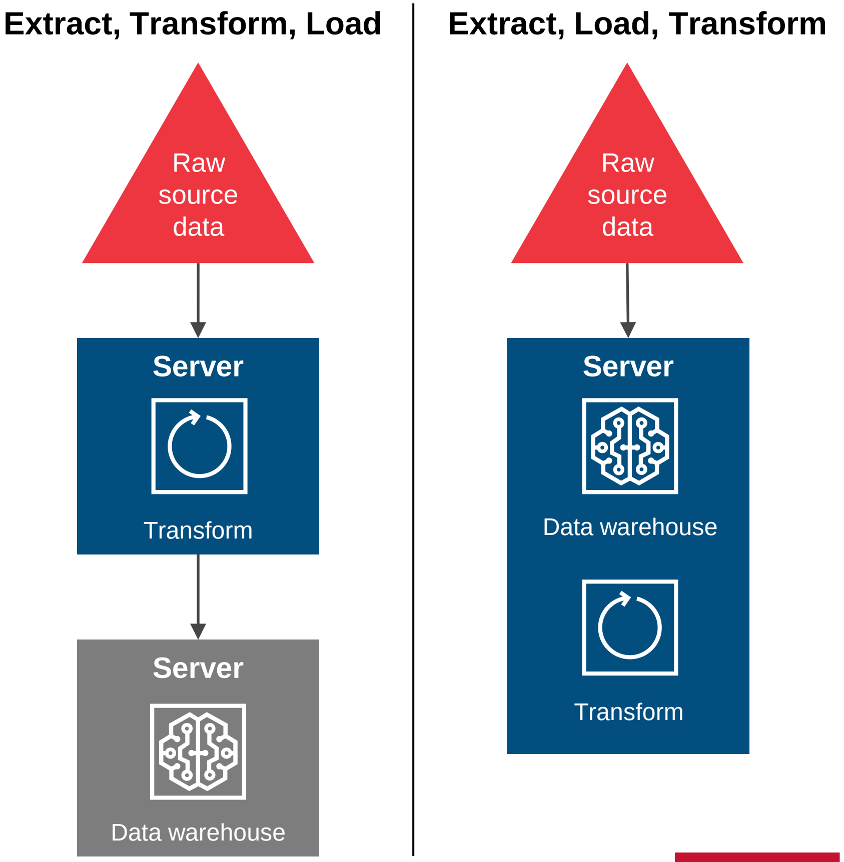
**ETL:** At initial stage, requires less space and result is neat.

**ELT:** Needs in-depth knowledge of tools and large repository to store

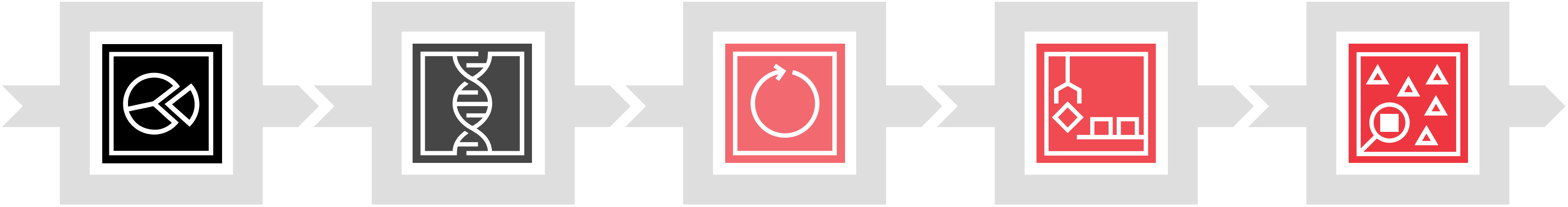
**Data limitation in supply:**

**ETL:** By choosing data a priori.

**ELT:** Data retention policy and no data limitation.



# Components of Data Integration



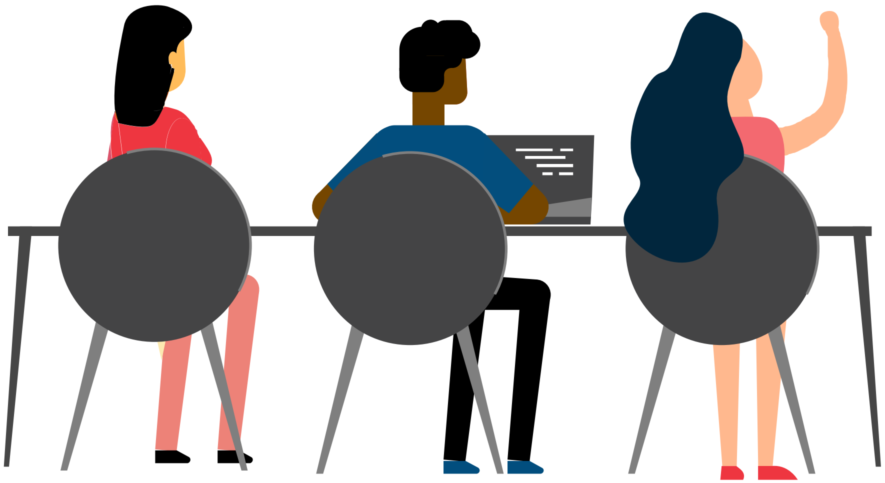
Extraction

Transformation

Loading

Orchestration

Exception/
Error handling

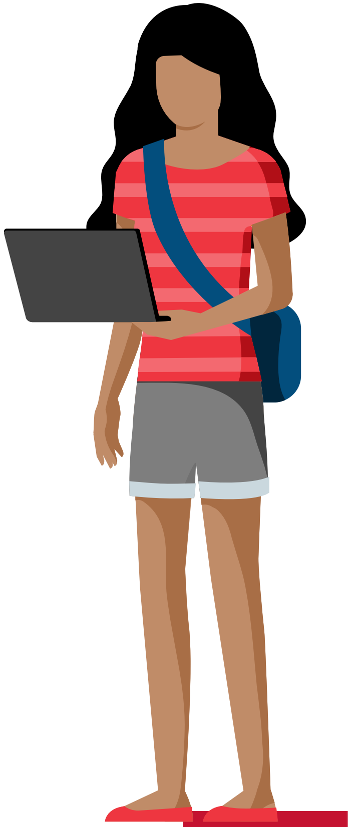


# Extraction

- During extraction phase of the ETL, the desired data sources—the rows , columns and fields to be extracted from them, are **identified**. These sources include transactional databases hosted on-site or in the cloud, business applications etc.
- This stage includes **planning** the **volume of data** from each data source
- Data extraction happens in one of the three phases:
  - **Update notificatio**n: The source system notifies you when a record has been changed
  - **Incremental extraction**: Some systems are unable to notify an update but are able to identify the **modified records** and provide an extract on those records. In subsequent ETL steps, the system needs to identify changes and bring them down. A negative impact is that it may not be able to detect deleted records in source data
  - **Full extraction**: Some systems are **not able to identify** the **modified data**, so
    **re-loading** all the data is the only way to get data out of the system. It requires you to keep a back-up of the previous record, so as to compare and identify new records. As this approach results in high data transfer volumes, it is least recommended

# Transformation

- Refers to **cleansing** and **aggregation** on data to prepare it for analysis
- There are Four ways to perform data transformation:
  - **Multistage data transformation** - Extracted data is moved to a staging area where transformations occur before loading the data into the warehouse
  - **In-warehouse data transformation** - This happens in ELT. The data is extracted and loaded into warehouse and transformations are performed there
- The **complexity** of transformation ranges over Four types:
  - **Basic transformation**: Cleaning->Deduplication->Format revision->Key restructuring
  - **Advanced transformation**: Derivation->Filtering->Joining->Splitting->Data Validation-> Summarization->Aggregation->Integration



# Loading

- The load stage of the ETL process depends largely on the **intent of operation** on the loaded data
- There are Four primary methods to load data into the warehouse:
  - **Full Load**: The entire source data is dumped into the data warehouse at the first go
  - **Incremental Load**: The data is loaded at regular intervals, with every next load bringing in data extracted after the last extract date

| | Full load | Incremental load |
|---|---|---|
| Rows sync | All rows in source data | New and updated records only |
| Time | More time | Less time |
| Difficulty | Low | High. ETL must be checked for new/updated row. Recovery from an issue is harder |

# Orchestration

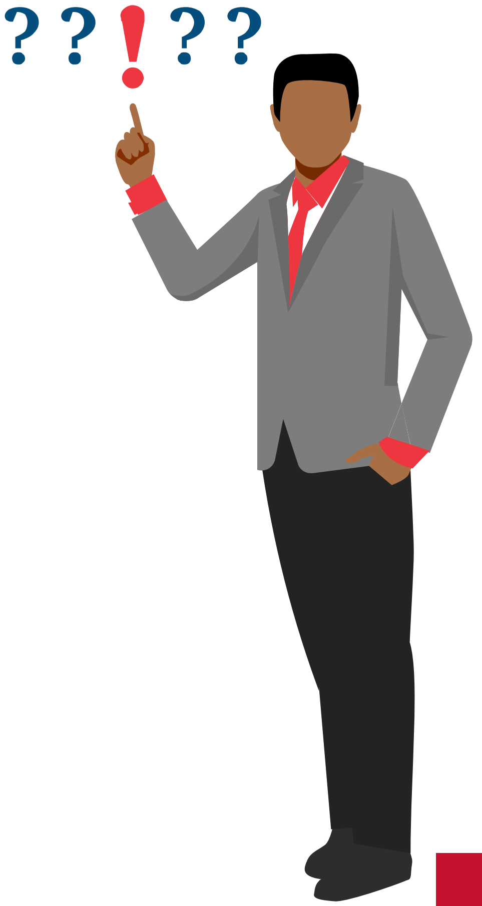
- **Process of integrating** Four or more tasks together to automate a workflow
- Simplifies complex workflows, avoids automation silos, connects different processes and more importantly **maintains** and **manages** the workflow at **single point**
- End to end integration and automation ensures we are **meeting the SLA** and delivering the insights to the business on time
- Some of the benefits
  - Improves quality, speed and reduces time to market
  - Reduces complexity when dealing with environments such as on-premises, cloud or hybrid
  - Risk of failure is reduced which increases scalability and reliability
  - Better integration with upstream/downstream applications
  - Better exception/error handling

# Exception/Error handling

- Exceptions/Errors can occur due to any **abnormal termination, incorrect data** and undesirable events which can stop the ETL/ELT process
- Handling these cases helps in stopping data from getting **corrupted** and making sure the process **terminates gracefully**
- All exceptions and errors normally contain time taken to execution, success or failure and error description
- Exception can be caused due to **data** or **infrastructure** problems like incorrect data format, incomplete data, operation system, hardware, database etc.
- Some of the best practices to analyse and fix the issue quickly
  - Alert mechanism – send error message as an email to business users and IT support team
  - Stop the process if it has a direct impact on business logic, if not ignore them but do save it for reference
  - Log all exception and errors in file or table

# Change Data Capture (CDC)

CDC is used to **determine** and **track changed data** instead of dumping the entire database.



## Why do we need CDC?

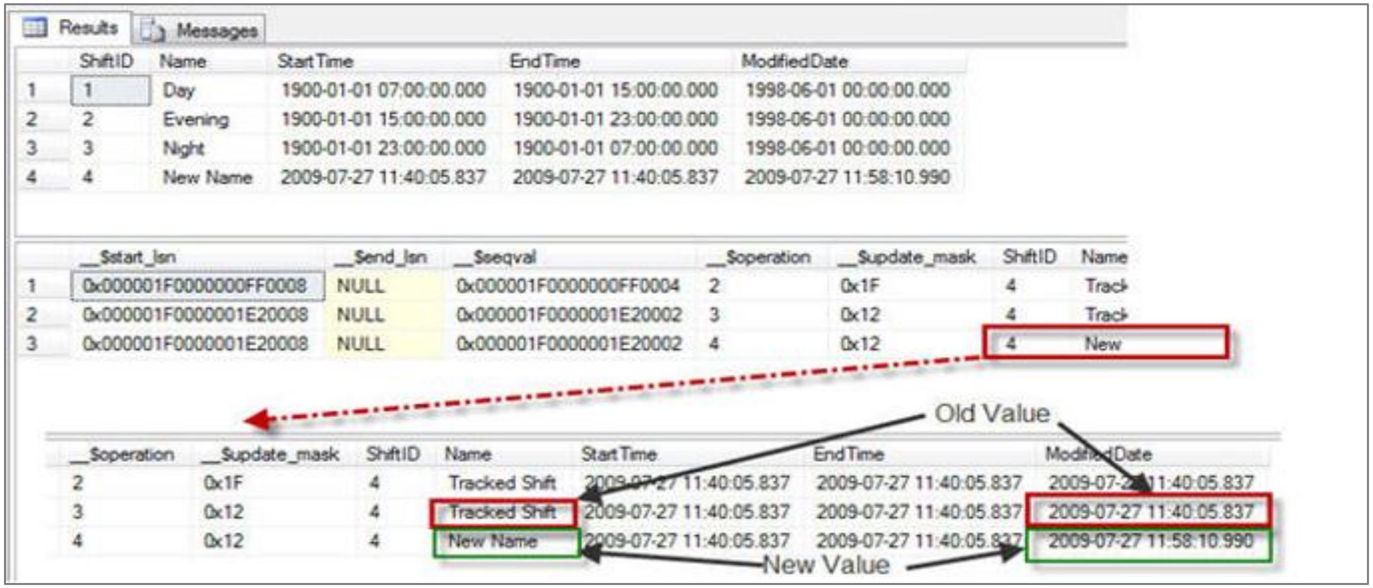
- CDC is much more **scalable** because it only deals with data changes
- The replication could be done much **faster**, often in real-time

## Four Common Methods to perform CDC

- Date Modified
- Diff
- Triggers
- Log-Based Change Data Capture

# CDC – Date modification with example

- Keep track of changes and filter **Date_Modified** column to retrieve modified data
- It works well for traditional data warehouse applications that are populated using Extract, Transform and Load (ETL) jobs, when the source tables do not process deletes
- Challenges of date modification
  - – Data deleted cannot be extracted, as no information is available for deleted data
  - – Date_Modified must be available on all tables and must be reliably set



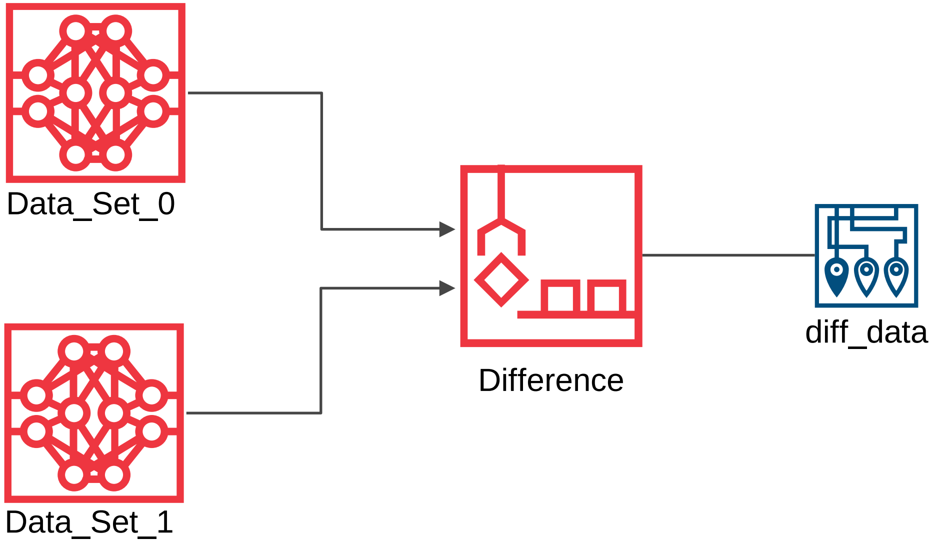
| | ShiftID | Name | StartTime | EndTime | ModifiedDate |
|---|---|---|---|---|---|
| 1 | 1 | Day | 1900-01-01 07:00:00.000 | 1900-01-01 15:00:00.000 | 1998-06-01 00:00:00.000 |
| 2 | 2 | Evening | 1900-01-01 15:00:00.000 | 1900-01-01 23:00:00.000 | 1998-06-01 00:00:00.000 |
| 3 | 3 | Night | 1900-01-01 23:00:00.000 | 1900-01-01 07:00:00.000 | 1998-06-01 00:00:00.000 |
| 4 | 4 | New Name | 2009-07-27 11:40:05.837 | 2009-07-27 11:40:05.837 | 2009-07-27 11:58:10.990 |

| | __$start_lsn | __$end_lsn | __$seqval | __$operation | __$update_mask | ShiftID | Name |
|---|---|---|---|---|---|---|---|
| 1 | 0x000001F0000000FF0008 | NULL | 0x000001F0000000FF0004 | 2 | 0x1F | 4 | Track |
| 2 | 0x000001F0000001E20008 | NULL | 0x000001F0000001E20002 | 3 | 0x12 | 4 | Track |
| 3 | 0x000001F0000001E20008 | NULL | 0x000001F0000001E20002 | 4 | 0x12 | 4 | New |

| __$operation | __$update_mask | ShiftID | Name | StartTime | EndTime | ModifiedDate |
|---|---|---|---|---|---|---|
| 2 | 0x1F | 4 | Tracked Shift | 2009-07-27 11:40:05.837 | 2009-07-27 11:40:05.837 | 2009-07-27 11:40:05.837 |
| 3 | 0x12 | 4 | Tracked Shift | 2009-07-27 11:40:05.837 | 2009-07-27 11:40:05.837 | 2009-07-27 11:40:05.837 |
| 4 | 0x12 | 4 | New Name | 2009-07-27 11:40:05.837 | 2009-07-27 11:40:05.837 | 2009-07-27 11:58:10.990 |

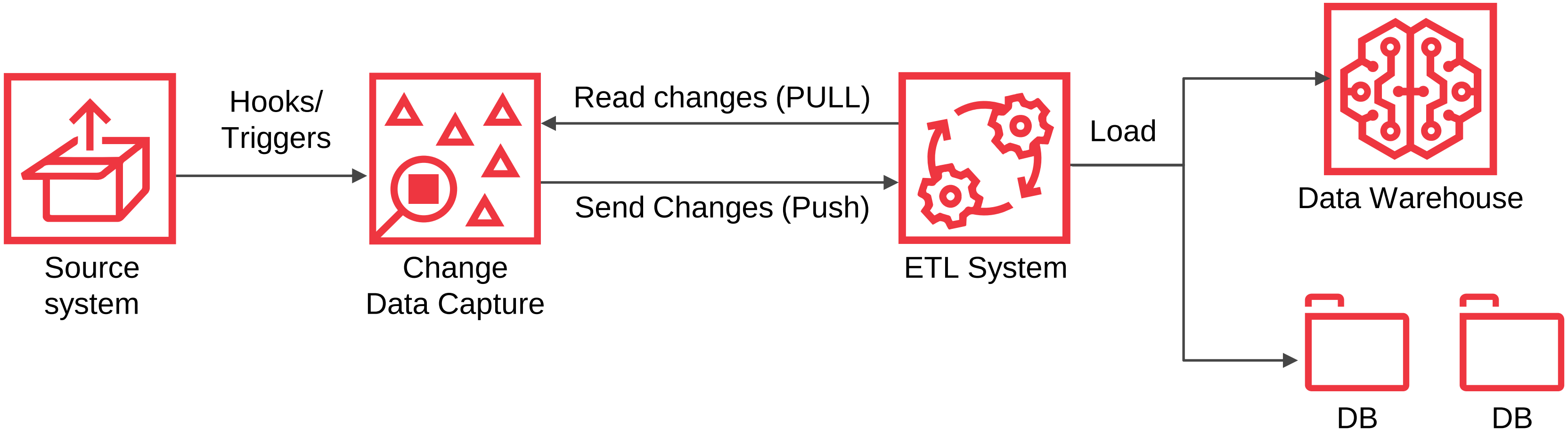
# CDC — Diff with example

- Diff **compares** the **current state of data** and **previous state of data** to identify changes
- This method works well for **small data volume**
- Challenges of Diff
  - It consumes lots of resources to compute the difference between original data and changed data
  - It cannot be done in real-time



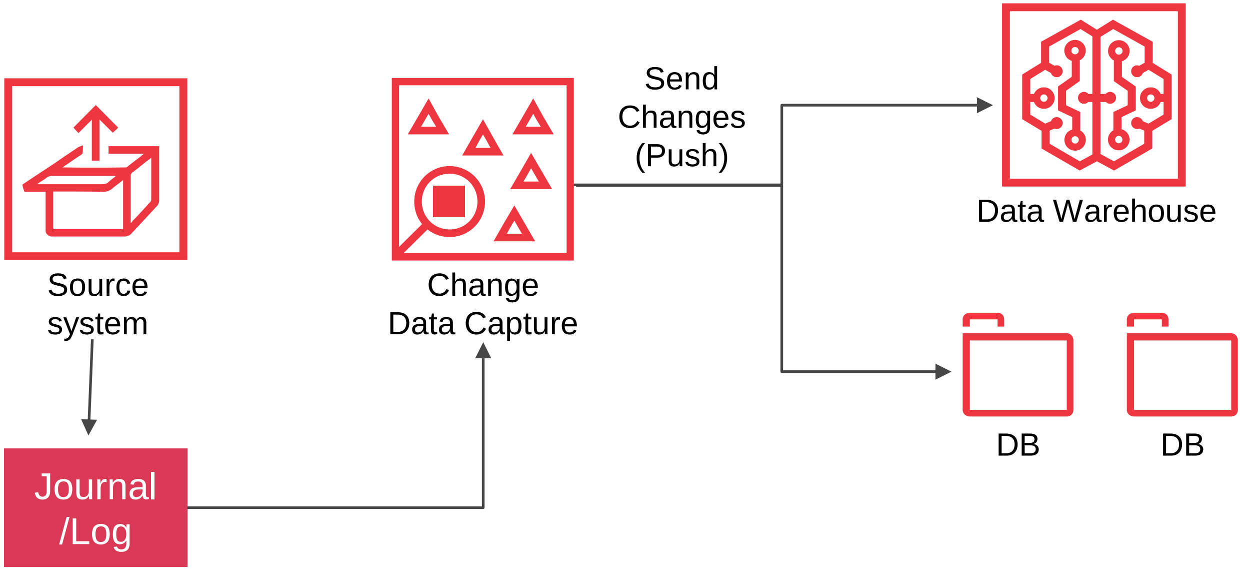
# CDC — Triggers with example

- Triggers could be used to perform CDC in **shadow tables**. Shadow tables could store the entire row to keep track of each single column change
- Trigger is very **easy to set up** and has little risk to implement
- Challenges of Triggers
  - Triggers lower the overhead cost to extract changes but increases the overhead cost to record the changes



# CDC — Log-based with example

- Log-based takes advantage of transactional database store from the **journals/commit log**
- Log captures and tracks data that has changed. It is **configured per table**
- Challenges of Log-Based CDC:
  - – Data vendors may not provide an interface to logs



# Quick Quiz

Please list three advantages of ELT

Please describe Four ways for data transformation

# List of Data integration tools available in the market

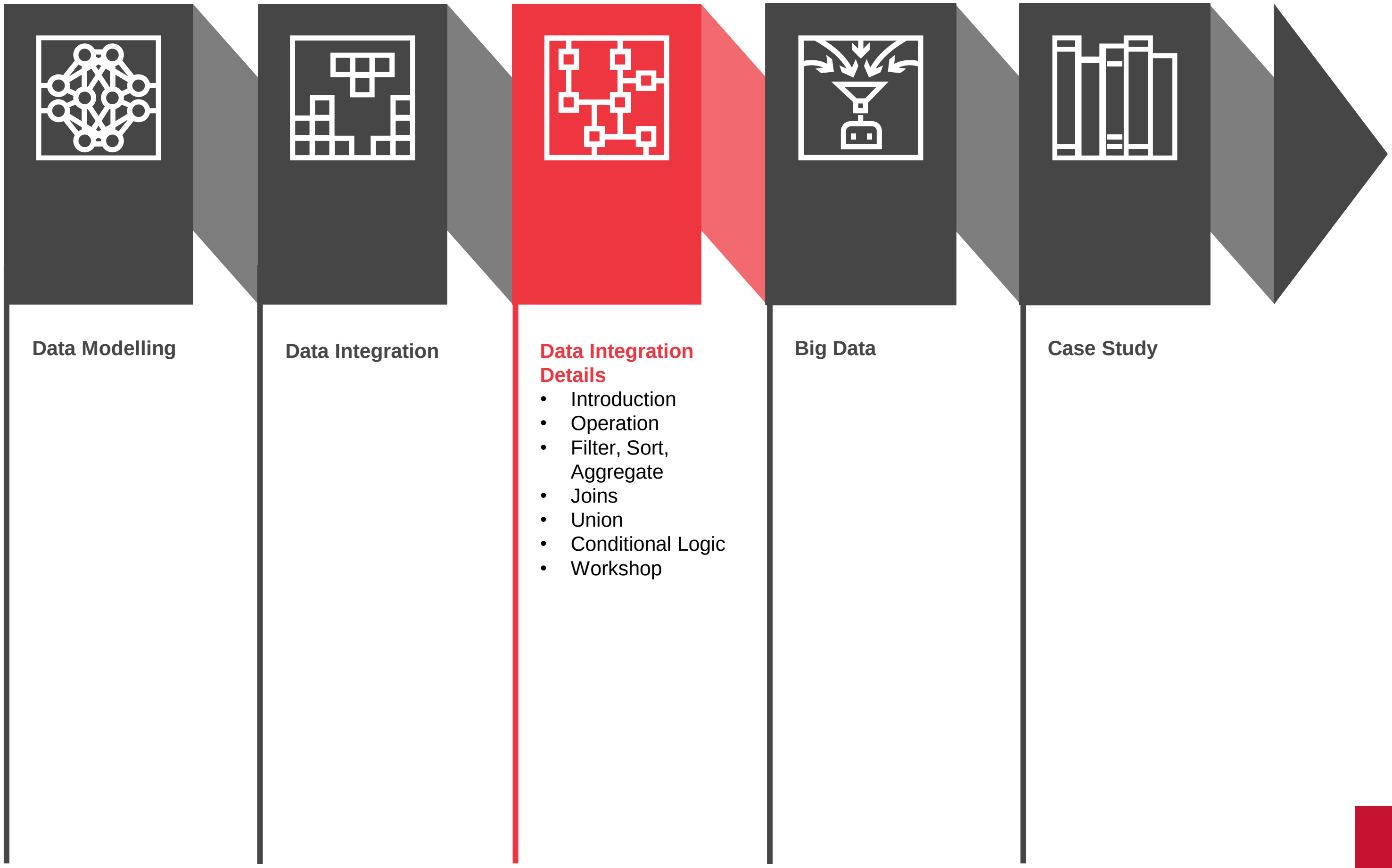
**On-premise** data integration tools:

- **Pentaho**
- IBM InfoSphere
- Informatica PowerCenter
- Microsoft SQL
- Oracle Data Service Integrator
- Talend Data Integration
- webMethods
- Centerprise Data Integrator

**Cloud-based** data integration services and tools:

- Alooma
- Dell Boomi AtomSphere
- Informatica Cloud Data Integration
- Jitterbit
- MuleSoft Anypoint Platform
- Oracle Integration Cloud Service
- Salesforce Platform: Salesforce Connect
- SnapLogic
- Talend Cloud Integration

# Day Four



Data Modelling

Data Integration

**Data Integration Details**

- Introduction
- Operation
- Filter, Sort, Aggregate
- Joins
- Union
- Conditional Logic
- Workshop

Big Data

Case Study

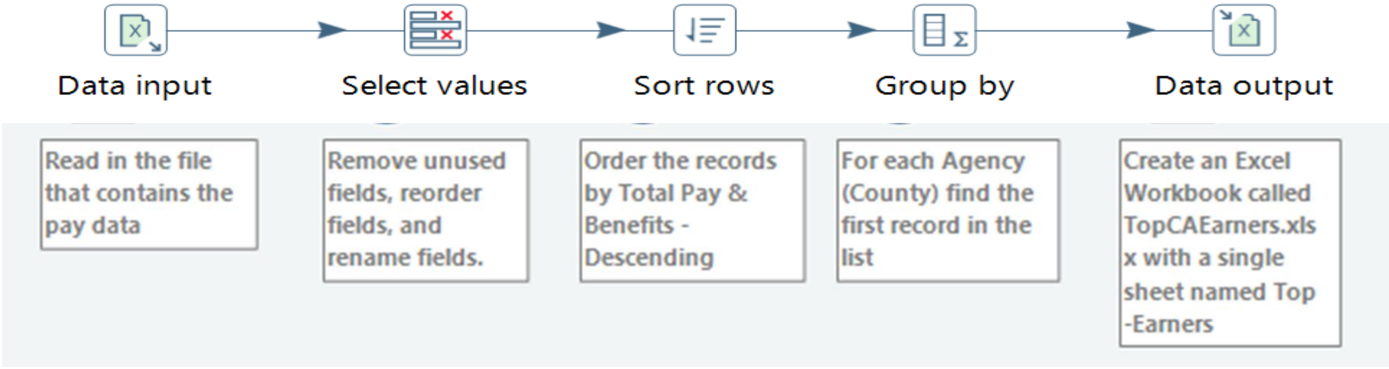
# Introduction to Pentaho tool

## What is Pentaho?

- A **graphical**, **workflow-based** tool designed to **empower business users** in the evolving world of data everywhere
- A tool that allows users to **blend**, **manipulate** and report on data quickly and easily in a repeatable **drag-and-drop** workflow **without** needing any special tools or **programming skills**

## Components of a workflow

- Tool – step in the process
- Anchors – input and/or output of a tool
- Connections– controls the flow of data from one tool to the next



Data input

Read in the file that contains the pay data

Select values

Remove unused fields, reorder fields, and rename fields.

Sort rows

Order the records by Total Pay & Benefits - Descending

Group by

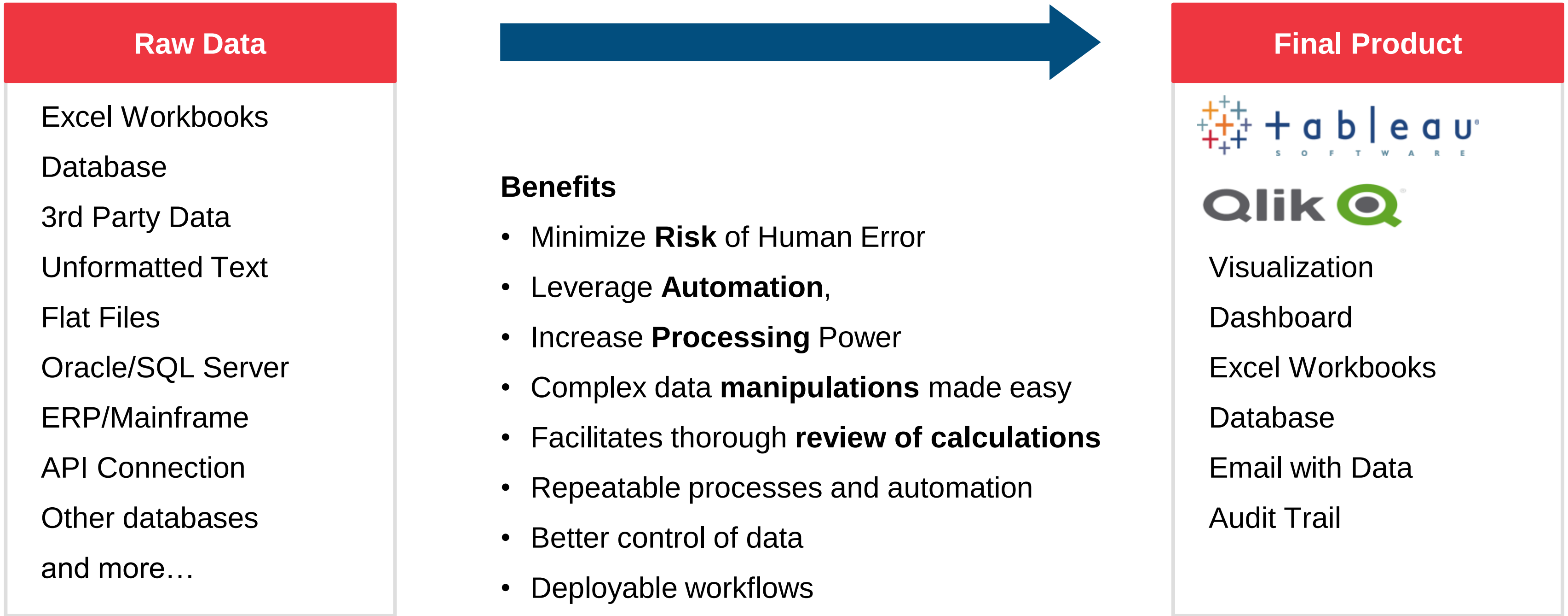
For each Agency (County) find the first record in the list

Data output

Create an Excel Workbook called TopCAEarners.xlsx with a single sheet named Top-Earners

# Basic operations in Pentaho tool

**Why do we need Pentaho?**



**Raw Data**

Excel Workbooks
Database
3rd Party Data
Unformatted Text
Flat Files
Oracle/SQL Server
ERP/Mainframe
API Connection
Other databases
and more…

**Benefits**

- Minimize **Risk** of Human Error
- Leverage **Automation**,
- Increase **Processing** Power
- Complex data **manipulations** made easy
- Facilitates thorough **review of calculations**
- Repeatable processes and automation
- Better control of data
- Deployable workflows
- Increased efficiency
- and more… (ETL, advance analytics, data mining, geospatial, data cleansing)

**Final Product**

Visualization
Dashboard
Excel Workbooks
Database
Email with Data
Audit Trail

# Filter, Sort and Aggregate

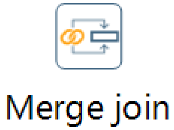
**Filter:** The filter tool **splits** a data stream into Four streams based on the conditional expression. Records that satisfy the condition flow put of the True anchor, and the rest come out the False anchor

**Sort:** The sort tool **arranges records** in a table in an ascending or descending order based on the value of one or more specified data fields

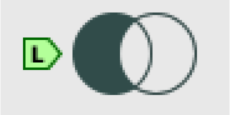
**Aggregate:** The aggregate tool moves values held in multiple horizontal fields into a single column

# Joining Data

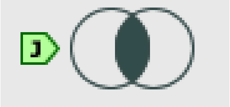
Joins are essentially like **V-lookup** functions in Excel, used when **combining fields (Columns)** from multiple tables **based upon** a **common** field that they share.



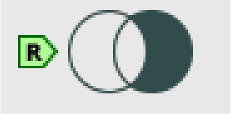
- **Left joined** – Shows all information from the left table which did not find a matching value in the right data set



- **Inner Join** – Only returns values for which a shared field between the right and left datasets has matching values

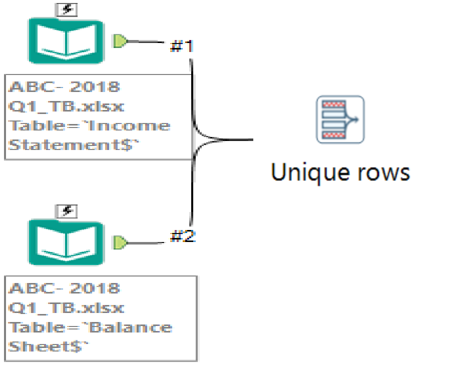


- **Right joined** – Shows all information from the right table which did not find a matching value in the left data set



# Union Data

Union helps to add data sets i.e. it is similar to **copying** contents from different tables/dataset and **pasting** it into a new table.


#1
ABC- 2018 Q1_TB.xlsx Table=`Income Statement$`
Unique rows
#2
ABC- 2018 Q1_TB.xlsx Table=`Balance Sheet$`

## Example of Union Data

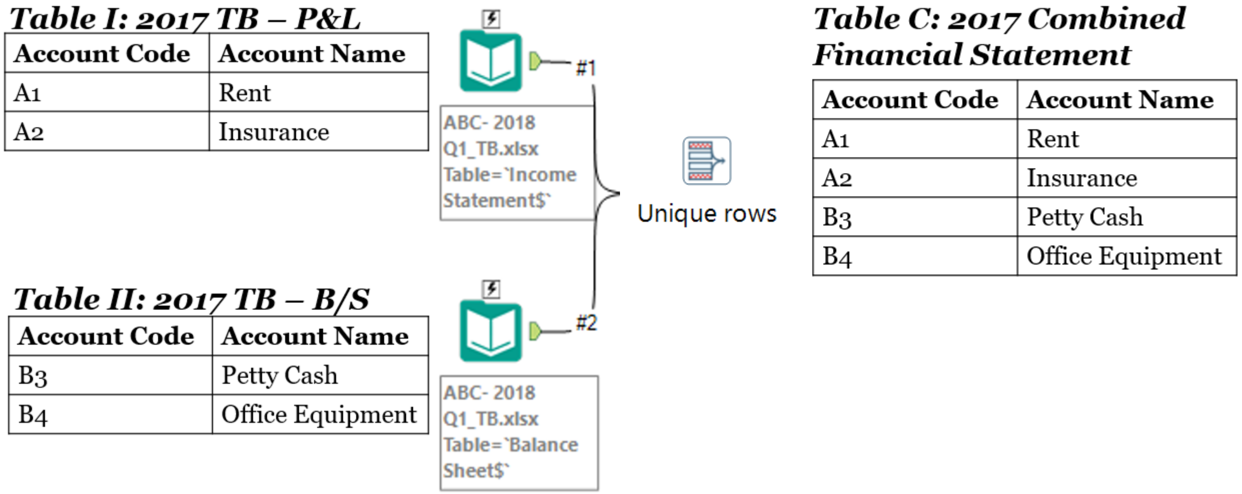


*Table I: 2017 TB – P&L*

| Account Code | Account Name |
|---|---|
| A1 | Rent |
| A2 | Insurance |

*Table II: 2017 TB – B/S*

| Account Code | Account Name |
|---|---|
| B3 | Petty Cash |
| B4 | Office Equipment |

#1
ABC- 2018 Q1_TB.xlsx Table=`Income Statement$`
Unique rows
#2
ABC- 2018 Q1_TB.xlsx Table=`Balance Sheet$`

*Table C: 2017 Combined Financial Statement*

| Account Code | Account Name |
|---|---|
| A1 | Rent |
| A2 | Insurance |
| B3 | Petty Cash |
| B4 | Office Equipment |

# Business Logic

Business Logic describes all of the **transformations** and **filtering** that needs to be done to data in order to **make it fit** for **analysis**. This typically takes the form of functions for Conditional Logic, Operations, or Formatting.

- **Conditional Logic** – The application of if-then statements to apply business rules based on specific criteria (conditions)
  Example: If (Apartment.Bedrooms = 1) Then “One-Bedroom Unit” Else “Multi-Bedroom Unit”

- **Operations** – The use of various procedural functions to modify data content, such as arithmetic addition,
  or string truncation
  Example: Transaction.Profit = Transaction.Revenue – Transaction.Cost

- **Formatting** – The use of functions to change the way data is displayed
  Example: FormatDate(‘Apr 3, 2018’, YYYYMMDD)

# Pentaho Case Study

## Case Study

PatientCare Solutions is a globally recognised consulting company who provide services to insurance industries. You work as a Data Engineer to help clients in analysing clients data and provide solution to problems

A medium sized manufacturing company “Pack&Take” whose offices are based in Singapore, India, China  and US approaches “PatientCare” to conduct a survey and analyse their data gathered to recommend relevant Insurance policies catering to their needs.

## Problem Statement

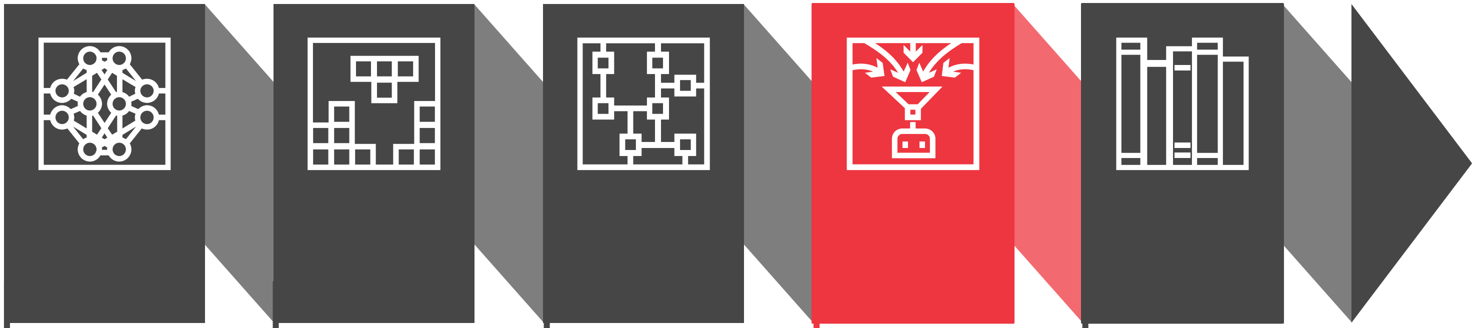
Your team contacts the client and conducts a survey and collects data like employee’s identity number, blood group, age etc. Data is received from all the four offices. Team has a list of its policies, survey data and list of Insurance which are shared with you in CSV file

Your manager recommend to use Pentaho as an ETL tool for analysis as its intuitive and fast development

## Solution

The data derived from survey is pulled into an ETL tool where the data goes through transformation. Transformed data are grouped based on their age, salary and habits. Different groups are then suggested different insurance policies according to their needs

# Day Four



**Data Modelling**

**Data Integration**

**Data Integration Details**

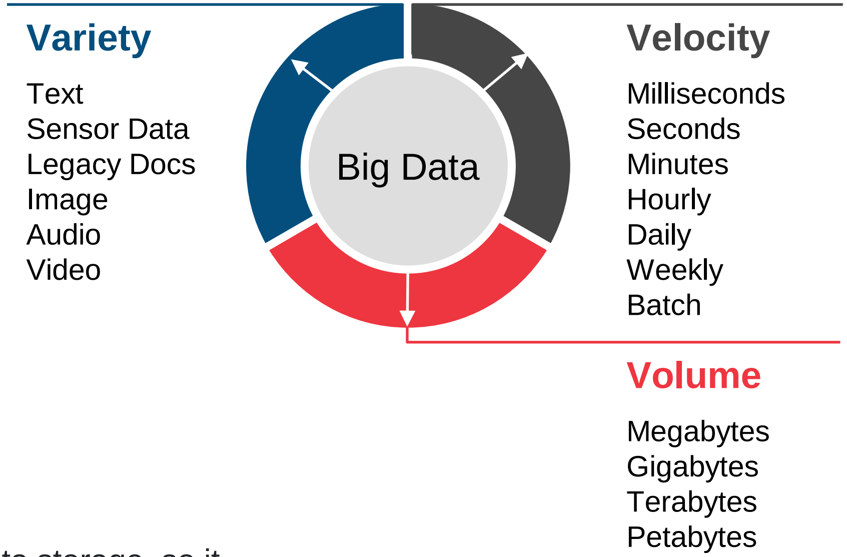
**Big Data**
- Introduction to Big Data
- Data Lake
- Hadoop ecosystem
- Big data components for data pipeline

**Case Study**

# Big Data Recap

Big Data is **high-volume, high-velocity** and/ or **high-variety** information assets that demand **cost-effective, innovative** forms of information processing that enable enhanced insight, decision making, and process automation.

**Gartner**
Source



**Variety**

Text
Sensor Data
Legacy Docs
Image
Audio
Video

Big Data

**Velocity**

Milliseconds
Seconds
Minutes
Hourly
Daily
Weekly
Batch

**Volume**

Megabytes
Gigabytes
Terabytes
Petabytes

## Is a 100GB data set “Big Data”?

- 100GB of structured data is easily managed by traditional data storage, so it would not be considered a **Big Data Volume**.  When we start talking about 100TB or PB, it becomes Big Data
- Processing 100GB of data per minute (**Velocity**), or analysing 100GB of unstructured (social media, image, etc.)  data (**Variety**) would be Big Data as well
- **Cost-effective solution** for the problem created by a **combination of 3Vs**
- Requires **innovative thinking** and **use of innovative tools and techniques**.

If your question fits into this definition, you have a **big data problem**

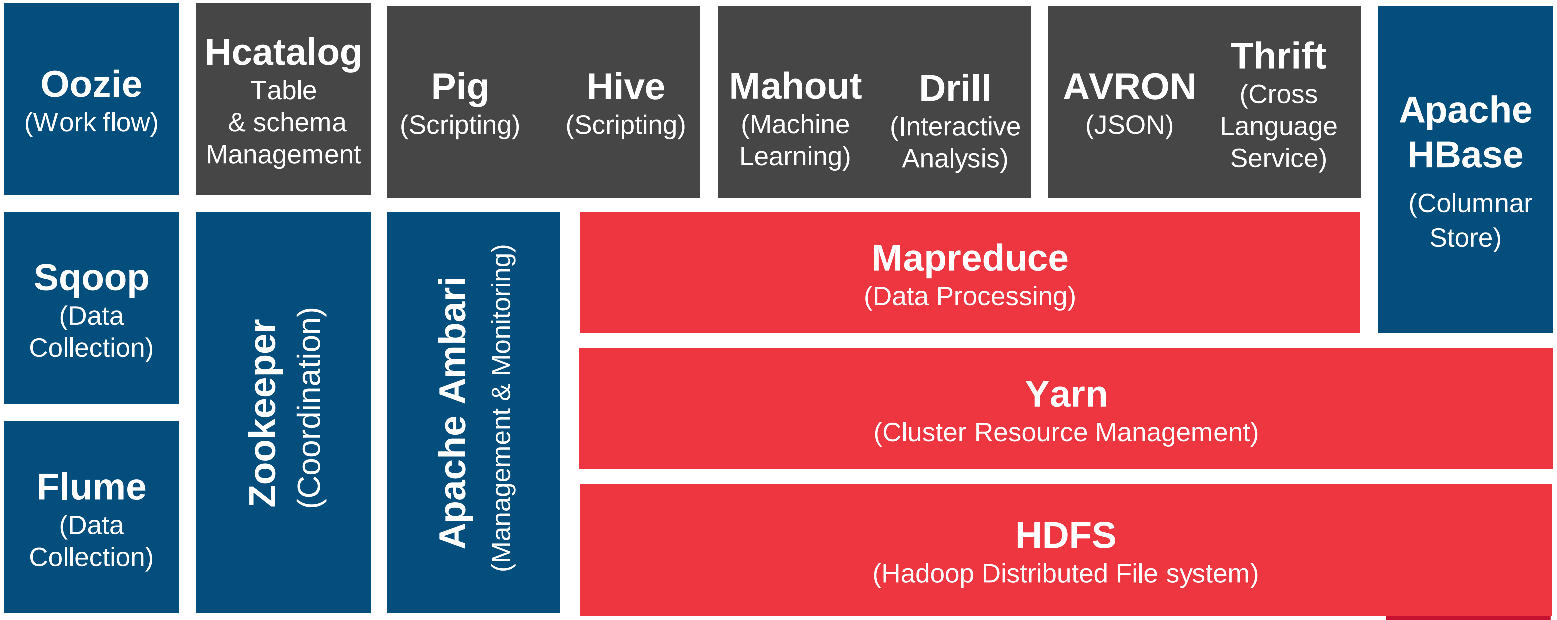
# Brief overview of Hadoop ecosystem

In Hadoop we have Four core components.

- A distributed file system **(HDFS)** – Hadoop distributed file system.
- A distributed programming framework **(Map Reduce)**

A set of tools that work on top of Hadoop core or around Hadoop core. We call it has Hadoop ecosystem.

- **Hive**
- Pig
- HBase
- **Spark**
- Sqoop
- **Kafka**
- Oozie



**Oozie** (Work flow)

**Hcatalog** Table & schema Management

**Pig** (Scripting)

**Hive** (Scripting)

**Mahout** (Machine Learning)

**Drill** (Interactive Analysis)

**AVRON** (JSON)

**Thrift** (Cross Language Service)

**Apache HBase** (Columnar Store)

**Sqoop** (Data Collection)

**Flume** (Data Collection)

**Zookeeper** (Coordination)

**Apache Ambari** (Management & Monitoring)

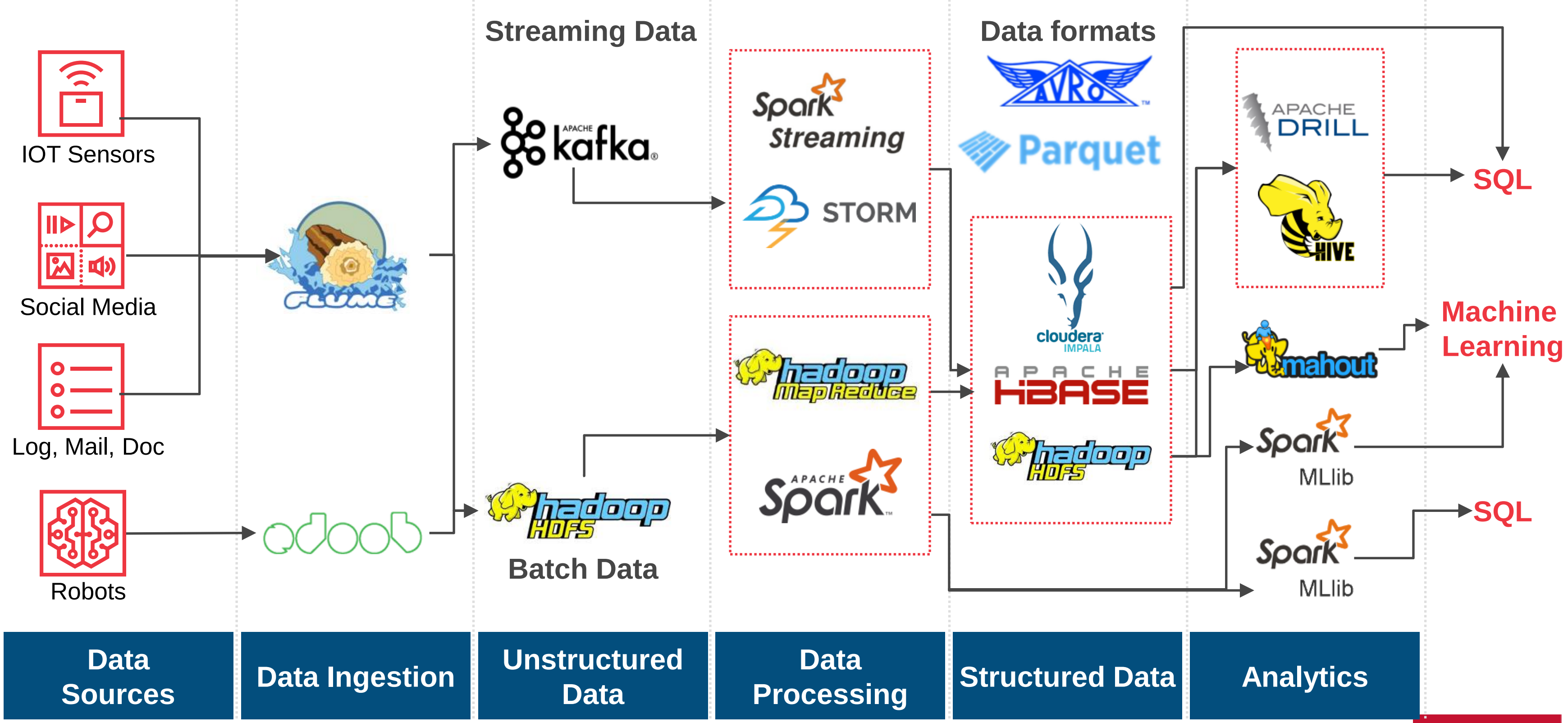
**Mapreduce** (Data Processing)

**Yarn** (Cluster Resource Management)

**HDFS** (Hadoop Distributed File system)

# Components of a Big Data pipeline

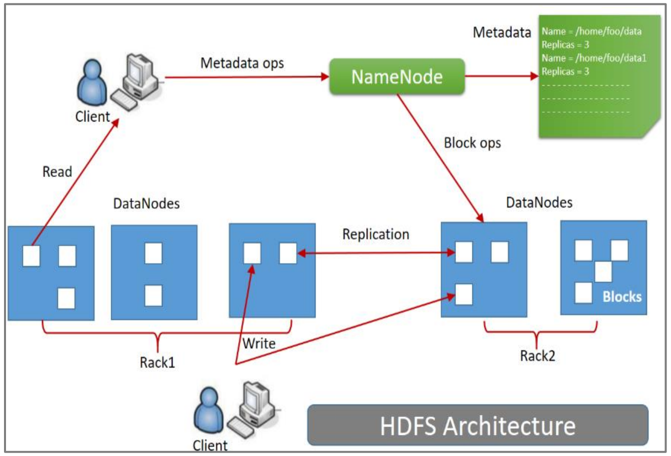
Real-time streaming data processing and batch data processing workflow



# Apache HDFS & MapReduce

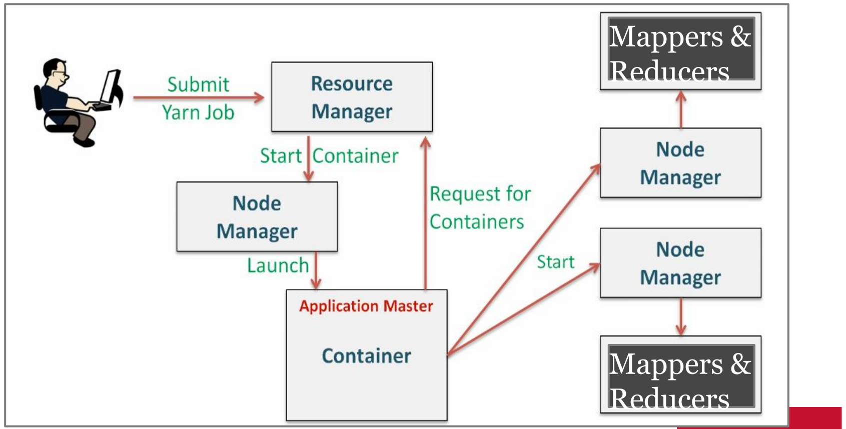
## HDFS

- Distributed File system
- Partitions the data and stores across cluster
- Master/Slave architecture
- Runs on commodity hardware
- Fault tolerant
- Support large data set
- Scales horizontally



## MapReduce/YARN

- Distributed data processing framework
- Splits data in to chunks and apply mapper and reducers functions.
- Parallel processing
- Resilient
- Scalability – process petabytes of data
- Minimal data movements



# Apache Hive

- Data warehouse build on top of Hadoop
- Provides SQL like interface to analyse data
- High throughput and medium latency
- Custom user defined MapReduce programs
- Mainly holds structure data
- Hide MapReduce complexity from end user
- ACID transactions
- In memory caching
- JDBC & ODBC support to connect to  major BI tools
- Store data in partitions and buckets
- Open source

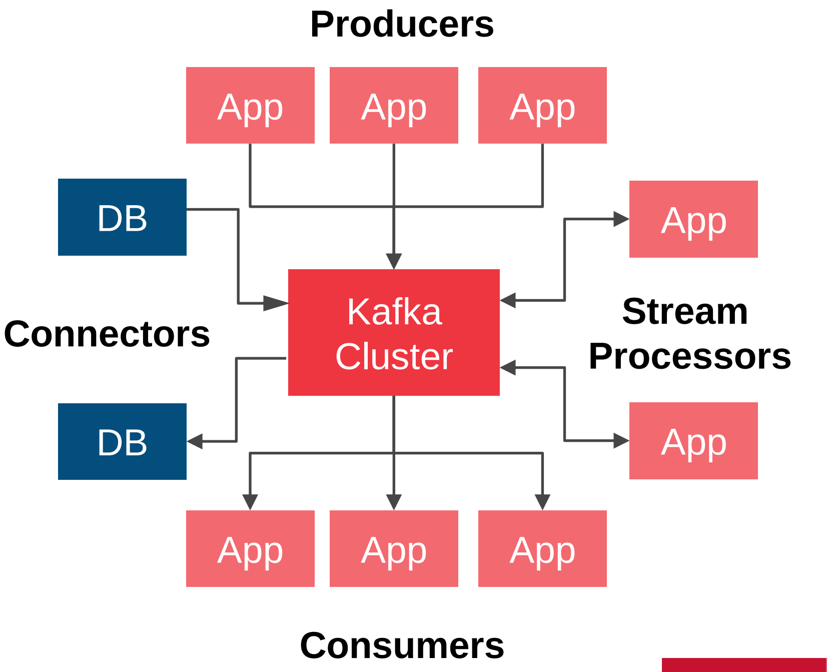
# Apache Kafka

Apache Kafka is a distributed streaming platform.

- **Publish** and **subscribe** streams of records, similar to a message queue or **enterprise messaging system**
- Store streams of records in a **fault-tolerant** durable way
- Process streams of records as they occur
- Scalability – easy scale with out downtime
- Durability – Persist data on disk and provides intra-cluster replication
- Reliability - Automatic balances consumers
- Performance – High throughput
- Design de-coupled architecture

APACHE kafka

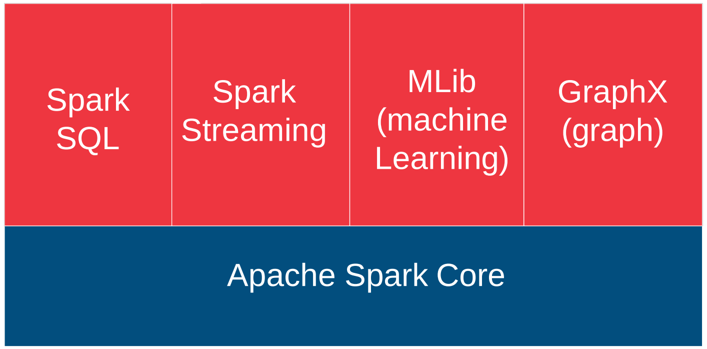
A distributed streaming platform



# Apache Spark

An open-source distributed general-purpose cluster-computing framework

- Speed - Spark's **in-memory data processing** engine means that it can perform tasks up to 100 times faster than MapReduce
- Developer-friendly Spark API - Spark **hides** much of the **complexity** of a distributed processing engine behind simple method calls



Spark SQL

Spark Streaming

MLib (machine Learning)

GraphX (graph)

Apache Spark Core

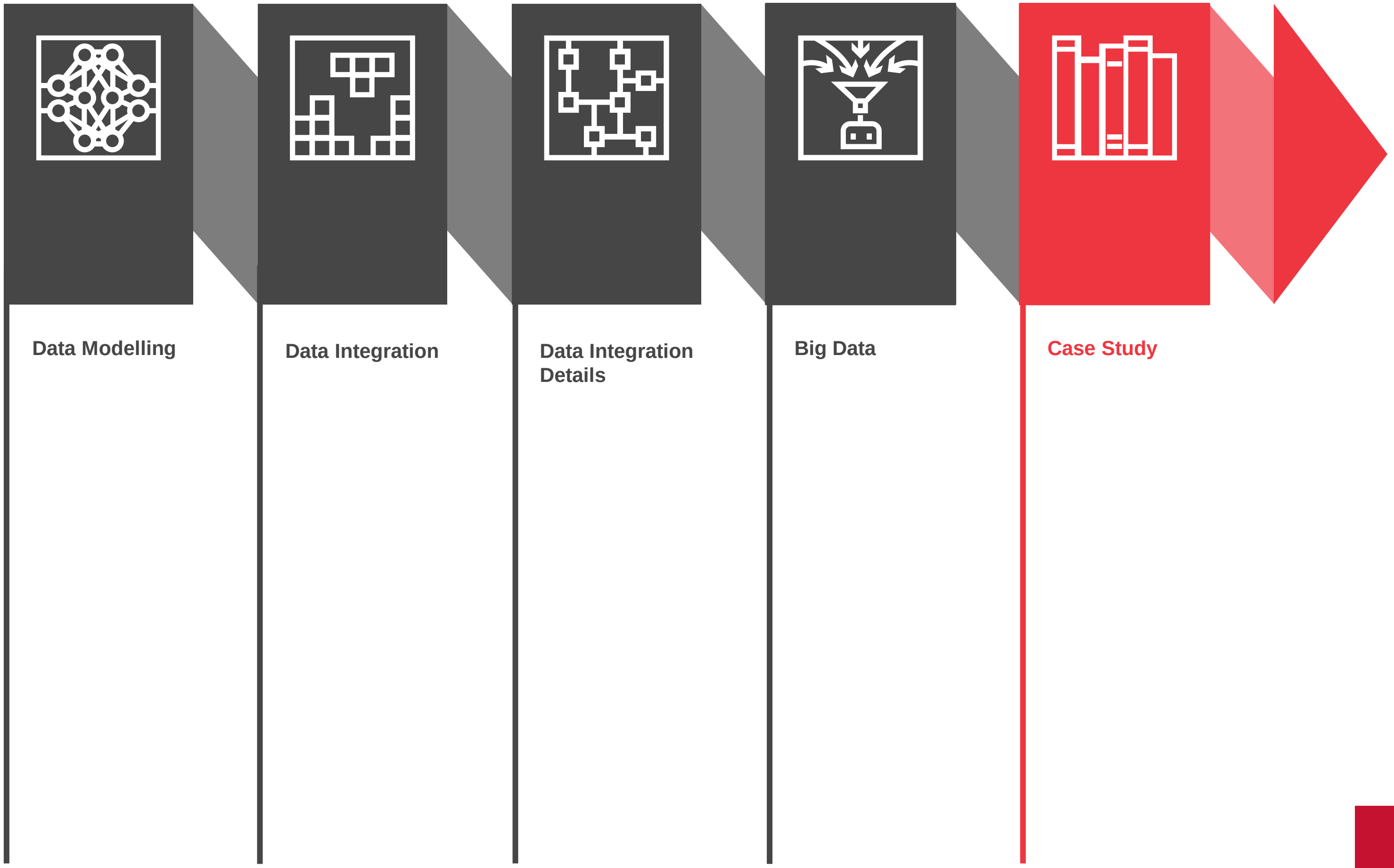
## Spark vs. Hadoop MapReduce

| Apache Spark | Apache Hadoop |
|---|---|
| Easy to program and no required abstraction | Difficult to program and requires abstraction |
| Streaming, batch processing, machine learnings, all in the same cluster | Helps to generate reports which helps find answers to historical queries |
| In-built interactive mode | No in-built interactive mode |
| Faster than Hadoop MapReduce | Does not leverage the memory of Hadoop cluster to the maximum |

# Demo on Spark

Run a demo to show the strength of in-memory processing by performing a few transformations like Filter, Group By using RDD

# Day Four



Data Modelling

Data Integration

Data Integration Details

Big Data

Case Study

# Retrospective Presentation

# Assessment

Assessment contains 30 multiple choice questions and you will be given 30 minutes to complete it.

Questions will cover hands-on concepts and the theory you learnt in the last Four days

https://docs.google.com/forms/d/e/1FAIpQLSe0Io1bazWxCSKtRiBTsyqUyboDVUersZ5jpduonODHQPkLZQ/viewform

# End of Day 4