# CIS 9440 - Data Warehousing and Analytics

Class #13

## Class note: Final Exam

The Final Exam will be hosted on 12/21/22 via Blackboard at 6:00pm - 8:00pm.

Like the Midterm Exam, I will leave a Zoom chat open for questions.

## Class note: Final Project Presentations (Milestone #5, due 12/6/22)

- Milestone #5 due on Blackboard on 12/6/22
- All Final Projects will be presented on 12/7/22
    - Sign up link here: link
    - You may present in-person or via Zoom
    - Maximum length of presentation is **9** minutes
    - Each presentation will have 2 minutes for questions
    - Your presentation may start earlier than the sign up time
    - **Attendance required** for entire class

## Airflow as an ETL Scheduler

- **Poll**
- **(Optional)** Enable Google Cloud Composer

## Week 13 Class Overview:

1. Dashboard filters in Tableau
2. Hadoop, Hive, and Spark
3. NoSQL Databases
4. Final Exam review
5. Scheduling an ETL Pipeline with Airflow

## Week 13 Class Overview:

1. **Dashboard filters in Tableau**
2. Hadoop, Hive, and Spark
3. NoSQL Databases
4. Final Exam review
5. Scheduling an ETL Pipeline with Airflow

## How to add user actions to a Tableau Dashboard?

1. Download the data: link

2. Create the following Worksheets:
   a. Average Volume by Sector
   b. Average Close by Date
   c. Sum of Volume by Company

3. Create a Dashboard

4. Select “Dashboard” -> “Actions”
   a. Create both a Select Filter and a Hover Highlight

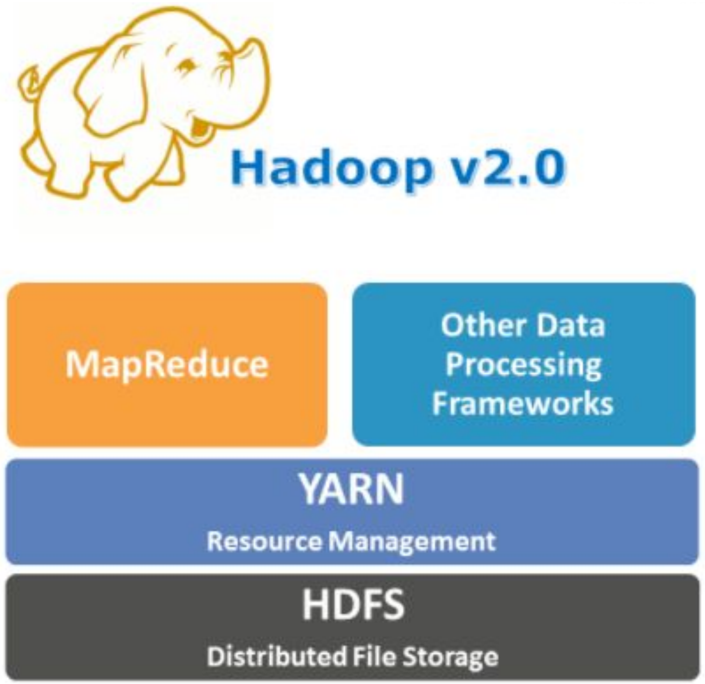
## Week 13 Class Overview:

1. Dashboard filters in Tableau
2. **Hadoop, Hive, and Spark**
3. NoSQL Databases
4. Final Exam review
5. Scheduling an ETL Pipeline with Airflow

## What is Hadoop?

Hadoop is a collection of *software* and services that support highly scalable distributed processing. The software and services include:

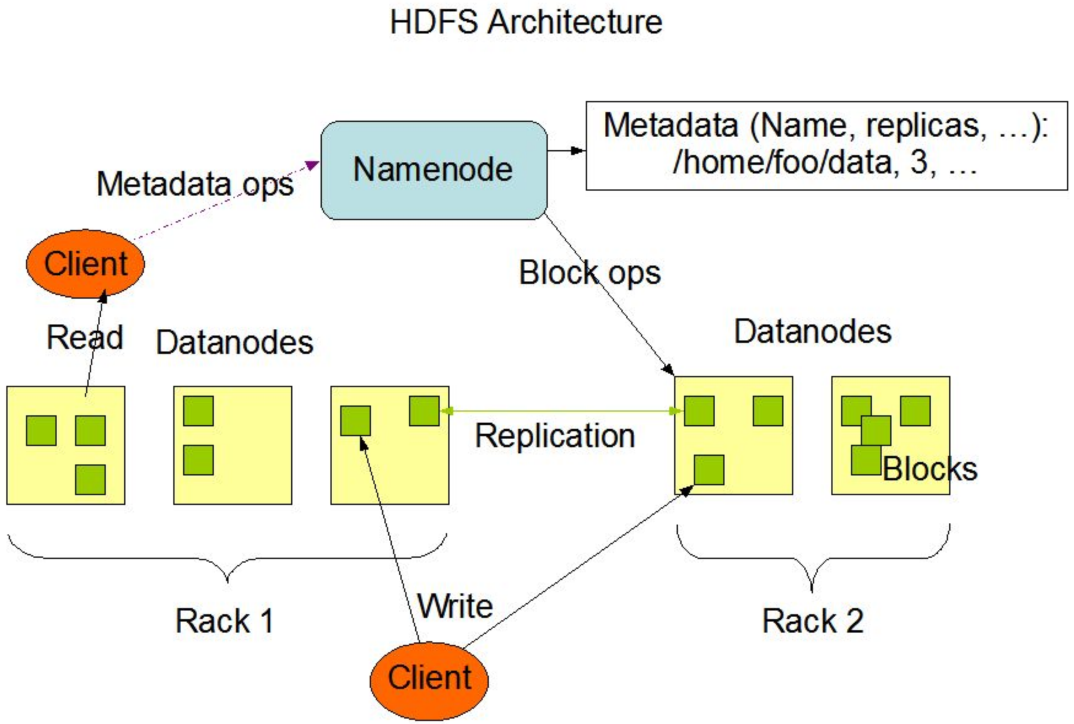
- **Storage Layer**: Hadoop Distributed File System (HDFS)
- **Scheduling Layer**: Hadoop YARN (Yet Another Resource Negotiator)
- **Execution Layer**: Hadoop MapReduce (or Spark)
- **Applications**: Hive (SQL), Flink, Storm



## Hadoop's Storage Layer

Hadoop Storage Layer is called "**Hadoop Distributed File System (HDFS)**". Important notes about HDFS:

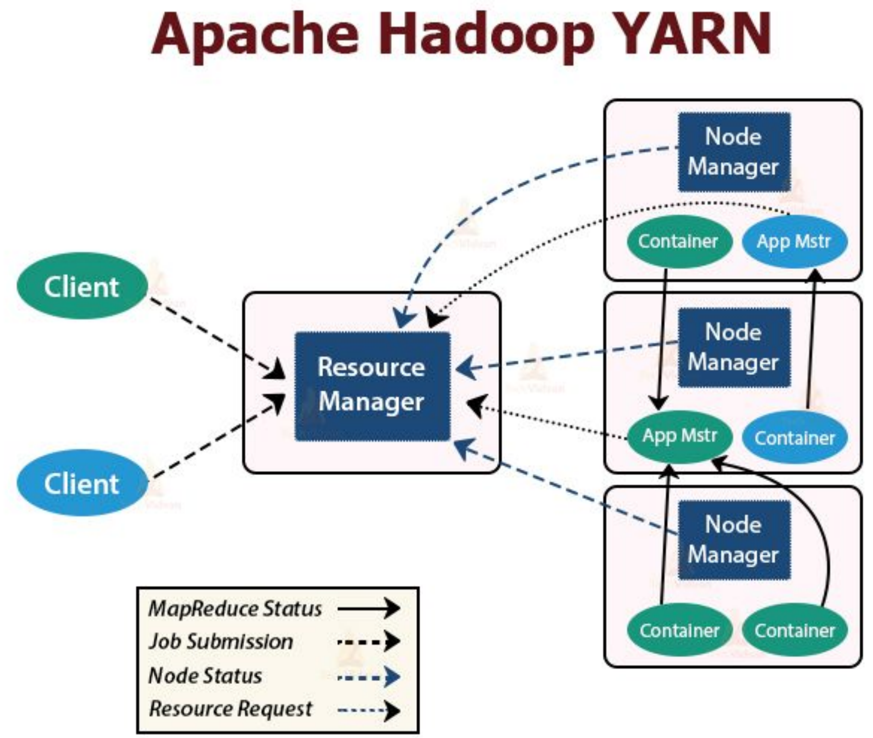
- The Replication Factor is default set to 3, that means 3 copies of each block of data
- Namenode manages the the file system overall



## Hadoop's Scheduling Layer

Hadoop Scheduling Layer is called "**Hadoop YARN (Yet Another Resource Negotiator)**". Important notes about YARN:

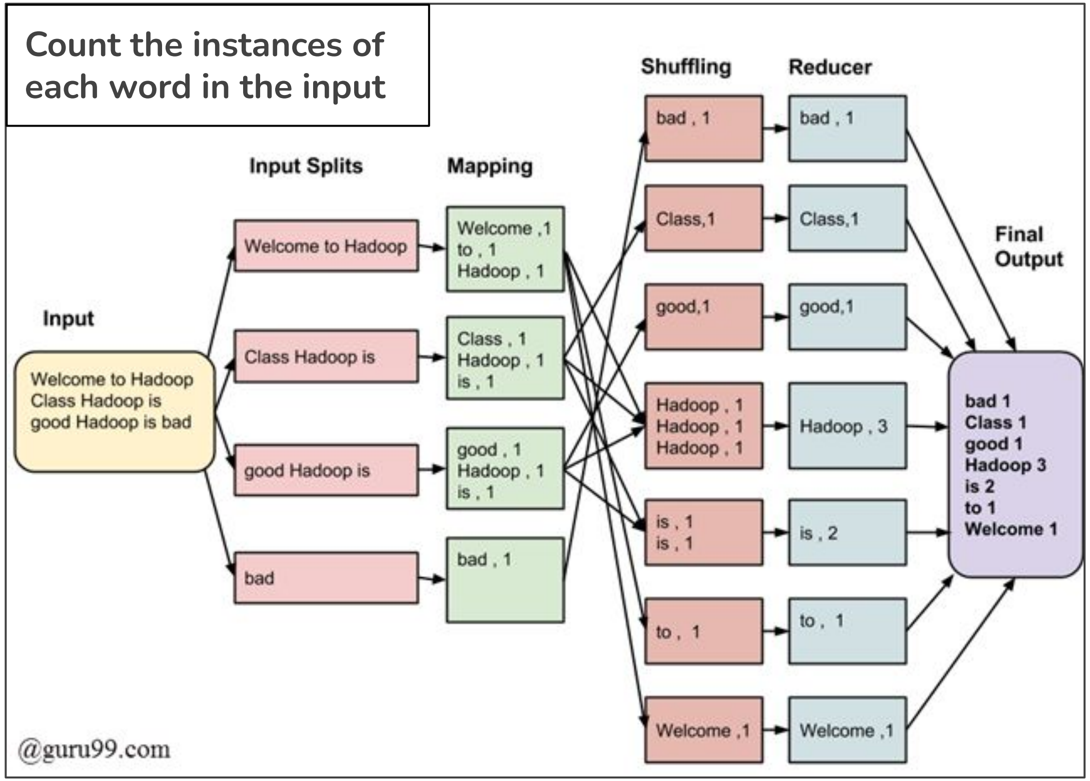
- Container: resources to process job
- Application Master: requests container from Node Manager
- Node Manager: handle resources within a node
- Resource Manager: assigns resources



# Hadoop's Execution Layer

Hadoop Execution Layer is called "**Hadoop MapReduce**". Important notes about MapReduce:

- MapReduce is a program that does 2 things:
  - First, allocates work (maps) to each node
  - Second, organizes (reduces) the work of each node into a single answer


Count the instances of each word in the input
Input
Welcome to Hadoop Class Hadoop is good Hadoop is bad
Input Splits
Welcome to Hadoop
Class Hadoop is
good Hadoop is
bad
Mapping
Welcome ,1 to , 1 Hadoop , 1
Class , 1 Hadoop , 1 is , 1
good , 1 Hadoop , 1 is , 1
bad , 1
Shuffling
bad , 1
Class,1
good,1
Hadoop , 1 Hadoop , 1 Hadoop , 1
is , 1 is , 1
to , 1
Welcome ,1
Reducer
bad , 1
Class,1
good,1
Hadoop , 3
is , 2
to , 1
Welcome ,1
Final Output
bad 1 Class 1 good 1 Hadoop 3 is 2 to 1 Welcome 1
@guru99.com

## Hadoop’s Applications

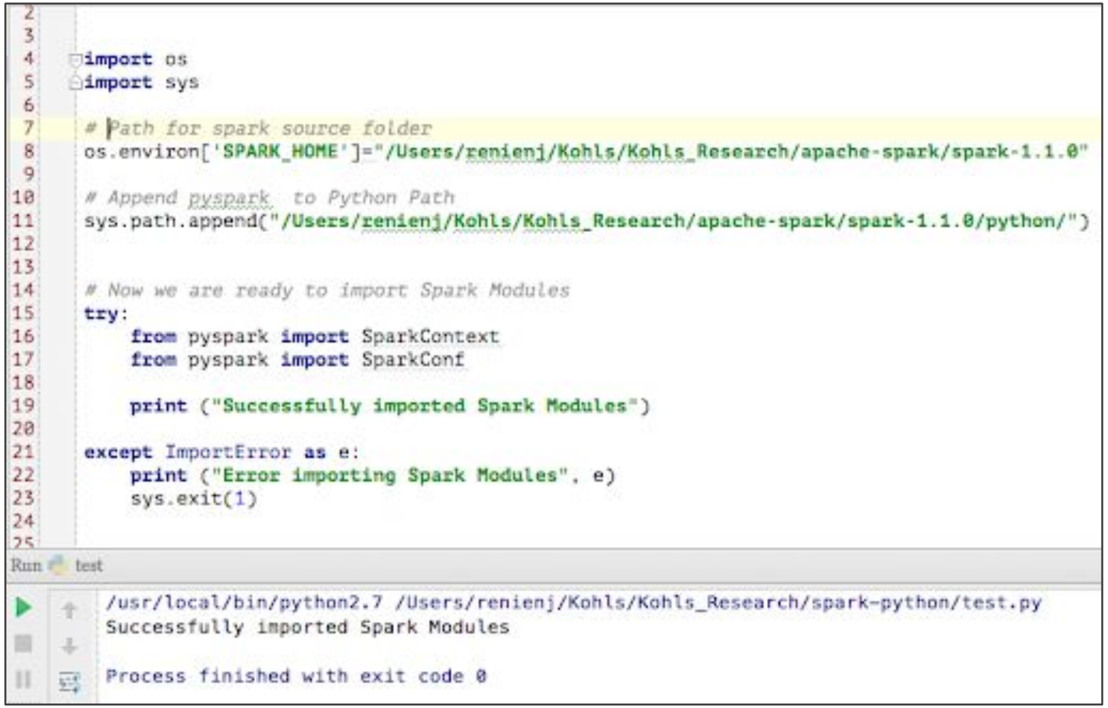
Hadoop has many applications built onto it’s platform, some are listed below:

- **Apache Hive**: a data warehouse software project built on top of Apache Hadoop for providing data query and analysis. Hive gives an **SQL-like interface** to query data stored in various databases and file systems that integrate with Hadoop
  - Practice Hive: https://demo.gethue.com/
  - Differences from SQL to Hive: link
- **Apache Pig**: a high-level platform for creating programs that run on Apache Hadoop. The language for this platform is called Pig Latin. Pig can execute its Hadoop jobs in MapReduce, Apache Tez, or Apache Spark
- **Apache Zookeeper**: an open-source server for highly reliable distributed coordination of cloud applications. It is a project of the Apache Software Foundation

# What is Apache Spark?

**Apache Spark**: an open-source unified analytics engine for large-scale data processing. Spark provides an interface for programming entire clusters with implicit data parallelism and fault tolerance.

- **More simply**, you can write applications in Java, Scala, Python, R, or SQL right on top of the Hadoop HDFS



```
import os
import sys

# Path for spark source folder
os.environ['SPARK_HOME']="/Users/renienj/Kohls/Kohls_Research/apache-spark/spark-1.1.0"

# Append pyspark  to Python Path
sys.path.append("/Users/renienj/Kohls/Kohls_Research/apache-spark/spark-1.1.0/python/")


# Now we are ready to import Spark Modules
try:
    from pyspark import SparkContext
    from pyspark import SparkConf

    print ("Successfully imported Spark Modules")

except ImportError as e:
    print ("Error importing Spark Modules", e)
    sys.exit(1)
```

Run test

```
/usr/local/bin/python2.7 /Users/renienj/Kohls/Kohls_Research/spark-python/test.py
Successfully imported Spark Modules

Process finished with exit code 0
```

## Week 13 Class Overview:

1. Dashboard filters in Tableau
2. Hadoop, Hive, and Spark
3. **NoSQL Databases**
4. Final Exam review
5. Scheduling an ETL Pipeline with Airflow

## What are NoSQL Databases?

- In the 1990’s, we had the traditional database vendors (DBMS) - Oracle, Microsoft, IBM, etc.
    - We support **transaction processing**, we’re good at it, we use SQL
- At the same time, the internet was evolving and users were exploring many new ideas, one of which “can we store data without SQL?”

# Why store data without SQL?

- **Why?** We want/have:
  - More Unstructured data
    - SQL is very structured, the internet is flexible and unstructured/unknown
  - A different performance characteristic
    - We must go faster, internet is in petabytes
  - Less Joins (NoSQL does not always support joins)
    - Data may be flatter from the internet
  - A different storage model
    - We want to search lists of objects with different data types, rather than rows and columns
  - A different use case
    - Nodes and edges, relationships are most important

## What does NoSQL look like today?

- After years of evolution there are many types of NoSQL databases (link), some actually still respond to SQL while others do not use only SQL (**NoSQL = “not only SQL”**). The types of databases are:

  - **Key-Value pair databases** (Oracle NoSQL, Amazon DynamoDB, Azure Cosmos DB)
  - **Document Store databases** (e.g., MongoDB, CouchDB, MarkLogic)
  - **Column-store databases** (e.g., HBase, Sybase IQ, Google BigQuery, AWS Redshift, etc.)
  - **Graph Databases** (e.g., SparkleDB, Neo4j, AWS Neptune, SparkSee (DEX), IBM DB2/RDF GraphStore)
  - **Streaming / Complex Event Processing databases** (e.g., Aleri, Streambase, Apama, SQL Stream, Amazon Kinesis, Oracle Streaming Analytics (CEP))

## Which NoSQL Database to learn?

- Well, there are 100’s of NoSQL databases and all are very different
- Thus, we will discuss the characteristics of NoSQL databases so you have some exposure, then we will move onto a popular NoSQL database “MongoDB”

## NoSQL Database Characteristics

Characteristics (and good questions to ask in an interview):

- **Model**: Relational vs. Non-relational
    - Relational: tabular data, with relationships among tables through Primary Keys, Foreign Keys, and Joins
    - Non-relational: not strictly tables with columns, keys, and relationships among tables

## NoSQL Database Characteristics

- **Schema**: Fixed schema vs. dynamic (non-fixed or flexible) schema
    - Fixed schema: you must first create a table with columns, then you can insert data into the table, finally query the data
    - Dynamic schema: first store the data, then query the data, which dynamically forms a schema on command

## NoSQL Database Characteristics

- **DDL/DML**: **How do you program it?** Programming Language API vs. SQL vs. Other query language
  - SQL: query the database with SQL
  - API: some databases only respond to programming language API's. You can embed this code inside your (python) code.
  - Other query languages: totally proprietary language

## NoSQL Database Characteristics

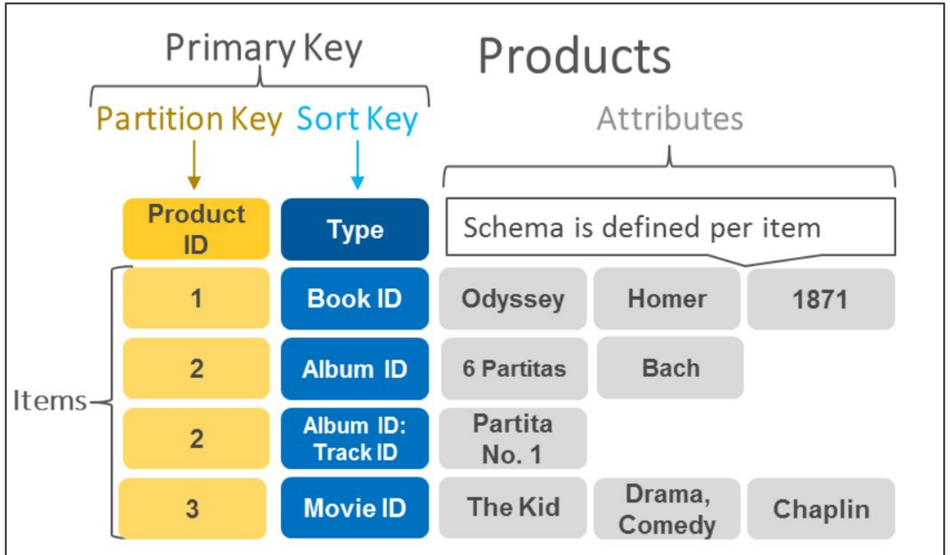
- **Hosting**: Local vs. Cloud-native
  - Local: download a local copy of the database to write queries (MS SQL Server)
  - Cloud-native: nothing to download, all hosted in the cloud (not the data, the database management system)

## NoSQL Database Characteristics

- **Use cases**: Transaction oriented vs. Analytic oriented
  - Transaction oriented: this database is designed for storing data with transaction and following ACID properties
    - Atomicity, Consistency, Isolation, Durability
  - Analytic oriented: database is designed to run analytics without analytical engine attached (like BI), probably follows BASE properties
    - Basic Availability, Soft State, Eventual consistency

## Types of NoSQL Databases

- **Key-Value Pair databases**:
  - Riak, Amazon DynamoDB
  - Very simple databases, like dictionaries in python



## Types of NoSQL Databases

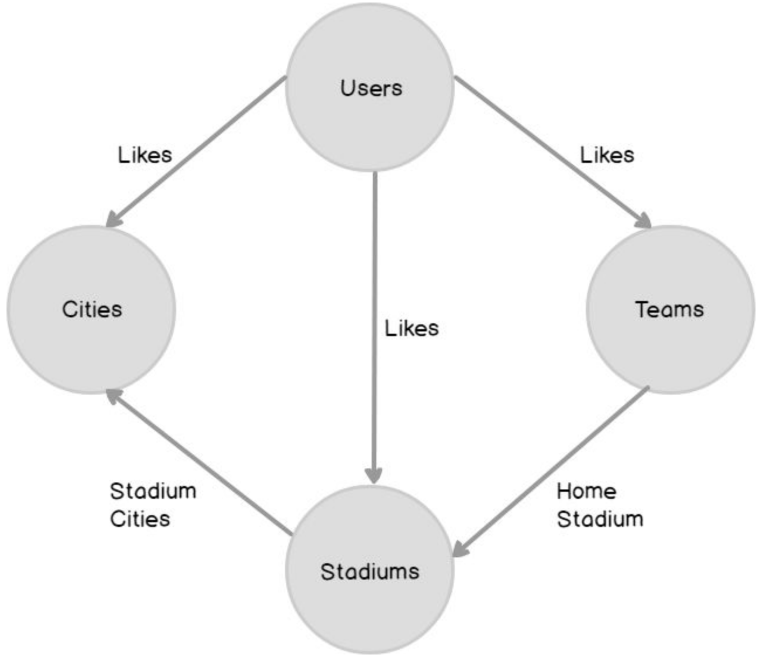
- **Document Store databases**: store an entire document as JSON or XML, essentially store anything
  - MongoDB
  - Documents can all be different formats
  - Data type is not important
  - Schema is not important
  - Relationships are not important

## Types of NoSQL Databases

- **Column-store databases**: rather than store one row, then the next. Column-store databases store each column of a table as a whole.
    - Google BigQuery
    - Very fast performance
    - Slower updates and schema changes

## Types of NoSQL Databases

- **Graph databases**: totally different models, objects are nodes in a graph and relationships are arcs between them
  - AWS Neptune
  - Very specific use-cases that focus on nodes and edges
  - **Relationships** are as important as the data itself

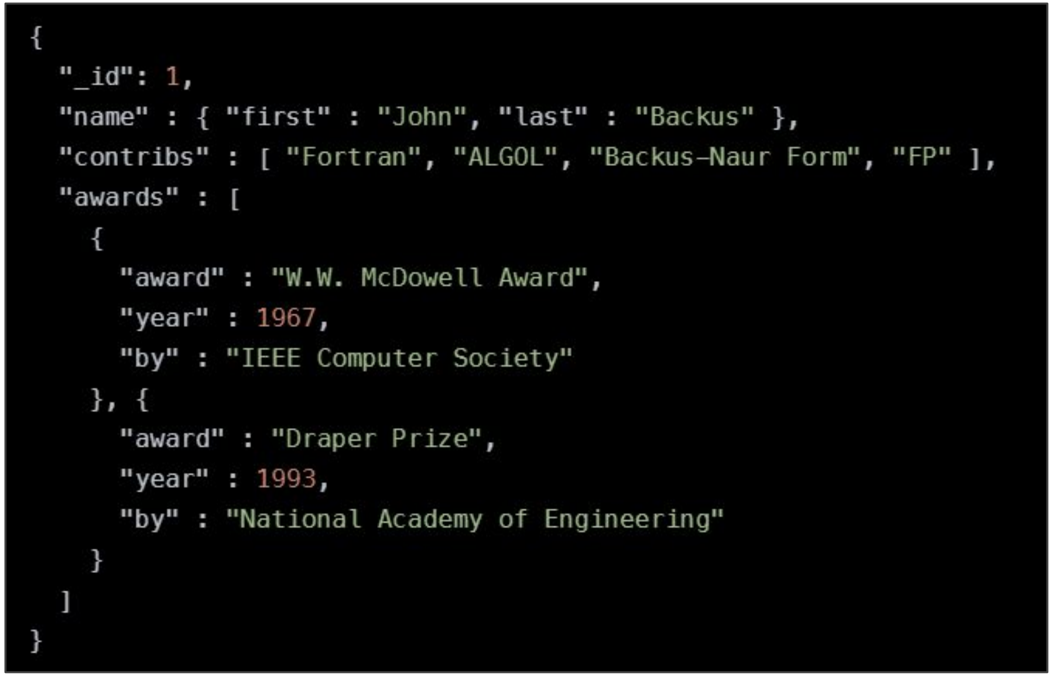


## Types of NoSQL Databases

- **Steaming/Complex Event Processing System**: tracking and analyzing streams of information
  - Amazon Kinesis
  - Query the data first
  - As the data arrives, it is matched to the query
  - If it fits statement of the query, it is diverted to your database
  - Often used for real-time data

## What is MongoDB?

- MongoDB is the most popular NoSQL database
- It is a Document Store Database
    - Key-value pairs, with values that can be full documents
    - You feed JSON documents into MongoDB



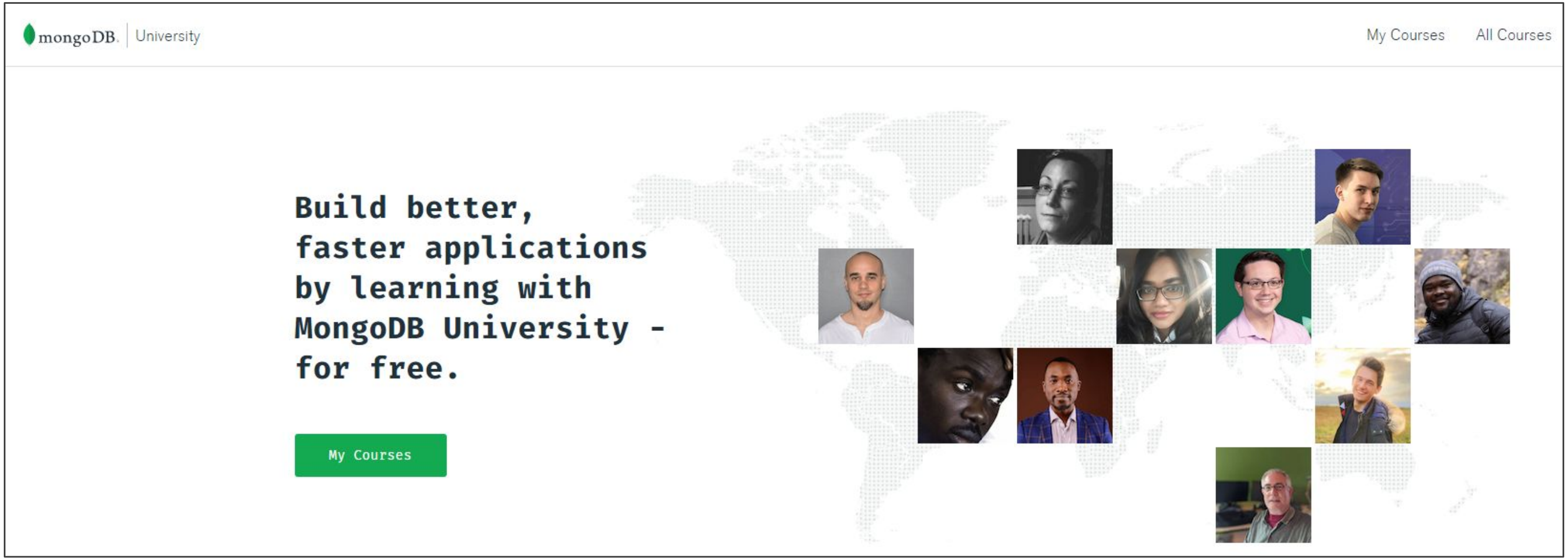
```
{
  "_id": 1,
  "name" : { "first" : "John", "last" : "Backus" },
  "contribs" : [ "Fortran", "ALGOL", "Backus-Naur Form", "FP" ],
  "awards" : [
    {
      "award" : "W.W. McDowell Award",
      "year" : 1967,
      "by" : "IEEE Computer Society"
    }, {
      "award" : "Draper Prize",
      "year" : 1993,
      "by" : "National Academy of Engineering"
    }
  ]
}
```

## MongoDB, why is it so popular?

- Speed
  - Horizontal scaling
- Speed to scale
  - No need to re-design database, the schema evolves on its own
- Flexibility
  - Regardless of previous documents, store more documents with more, less, or different fields. No schema.
- Complex
  - “Query” database with programming type statements

## MongoDB, how to get started?

https://university.mongodb.com/



Build better, faster applications by learning with MongoDB University - for free.

## MongoDB, how to get started?

- If you want to jump right in and use MongoDB Atlas, start here:
  - https://www.mongodb.com/nosql-explained/nosql-vs-sql
- At the bottom of the page, you can click on the link to get started:
  - https://www.mongodb.com/cloud/atlas?tck=NoSQLvsSQL

## Interacting with MongoDB

- MongoDB Shell
- Python API

## MongoDB example together

Open your web browser to:
https://docs.mongodb.com/manual/tutorial/getting-started/

## MongoDB, example

1. Open MongoDB GUI - https://docs.mongodb.com/manual/tutorial/getting-started/
   a. db
   b. show collections
   c. db.cis9440.insert({class: “data warehousing”, size: 15, status: “week 12”})
   d. db.cis9440.insert({class: “data warehousing”, semester: “Spring 2022”, size: 40, status: “not started”})
   e. show collections
   f. db.cis9440.find()
   g. db.cis9440.find({size: 15})

10 Minute Break

## Week 13 Class Overview:

1. Dashboard filters in Tableau
2. Hadoop, Hive, and Spark
3. NoSQL Databases
4. **Final Exam topics**
5. Scheduling an ETL Pipeline with Airflow

## Final Exam Format

- Final Exam will available on Blackboard at 6:00pm on 12/21/22
    - Folder: **Course Documents -> HWs, Projects, Exams -> Exams**
- You will have until 8:00pm to complete the Final Exam
- If you have questions, I will be available via Zoom Chat or email to respond
- To prepare, use Slides from previous classes and your Business Intelligence Guidebook textbook. The Final Exam is open book.
- The exam is worth 20% of your Final Grade

# List of Final Exam Topics

The Final Exam will test your knowledge of the material we learned after the Midterm Exam (from Week 9 - Week 13)

Business Intelligence Guidebook chapters 13, 14, 15, and 16

(Week 9) BI Application Design

(Week 9) BI Application Development

(Week 9) The Kimball Lifecycle

(Week 10) Analytics

(Week 10) Business Intelligence Best Practice and Analyses by Data Type

(Week 11) Business Intelligence Applications and Users

(Week 12) Data Warehouse Technical Architecture

(Week 12) Distributed Data Processing Architecture

(Week 13) NoSQL databases

## DW Technical Architecture track

According to the Kimball Lifecycle best practices, the design process is robust (since this is for both small and large companies). The two main components of the DW Technical Architecture track are:

1. **Technical Architecture Design** - (*Technical Requirements*)

2. Product Selection and Installation - (*Specific Products and Details*)

## What’s the process for Technical Architecture? Details continued

2. Product Selection and Installation
   a. Market research and Develop a product evaluation matrix
      i. https://library.educause.edu/-/media/files/library/2013/1/acti1302-xls.xls
   b. Evaluation a short list of options
   c. Select product, install on trial, negotiate
      i. *no price, contact sales (https://www.getdbt.com/pricing/)

## Which BI Application do I use?

1. When to use a Report?

   A Report is best used for broad, historic, and static data. Reports are often generated on a recurring basis by the business

2. When to use a Dashboard?

   Dashboards are best for data that may be interacted with, drilled upon, or manipulated by the users. Dashboards are often near-live, and used on a daily basis for BI analysis.

## Which BI Application do I use?
**(continued)**

3. When to use a Scorecard?

   Scorecards are optimal for measuring performance against benchmarks. Scorecards are more specific than reports, with static views of very specific KPI’s. Think about Quarterly Sales vs Goals as a typical scorecard,

4. When to use a Data Visualization?

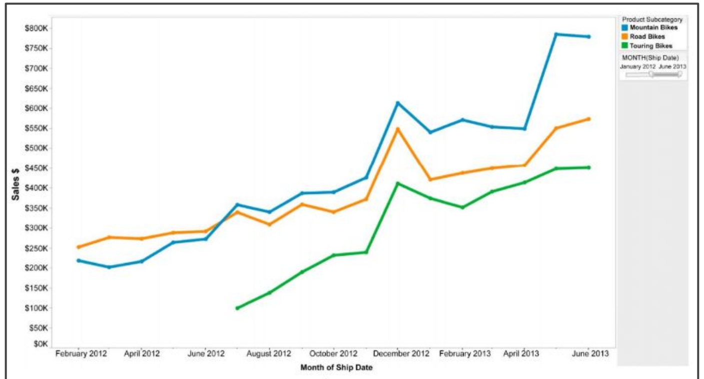
   Data Visualizations are generated ad hoc, for specific presentations, emails, or research. One place you may find a Data Visualization is a Team presentation to the company.

## Different BI Users

- Casual Consumers
  - Account Executives, Marketing team, Store Teams, HR associates, Graphic Designers, etc.
- Analysts
  - Financial Analyst, Sales Analyst, Marketing Analyst, Operations Analyst
- Power Users
  - Data Analyst, Business Information Team, Business Insights Team, Marketing Analytics Team
- Data Scientists

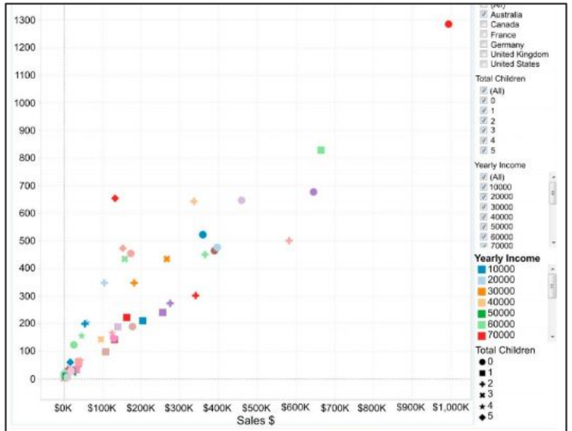
# BI Application Design

**Question**: From left to right, select the correct type of Analysis for each Visualization.



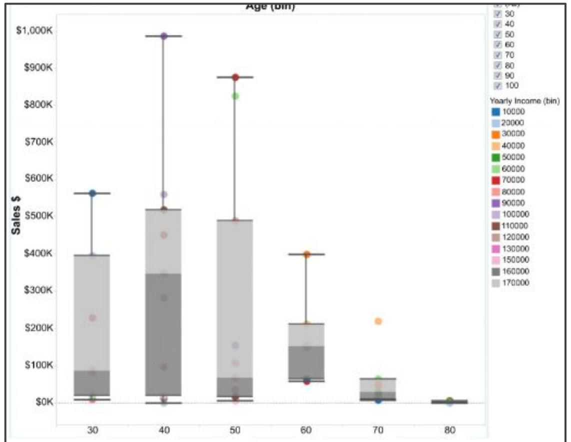
f13-05

1



f13-09

2

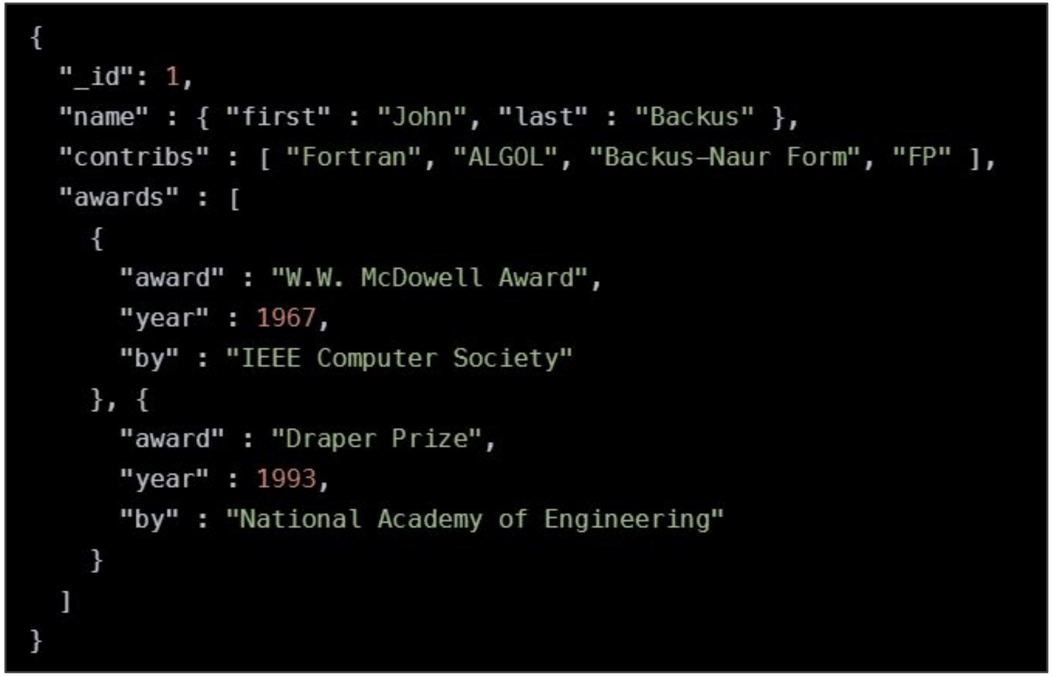


f13-11

3

## What is MongoDB?

- MongoDB is the most popular NoSQL database
- It is a Document Store Database
  - Key-value pairs, with values that can be full documents
  - You feed JSON documents into MongoDB



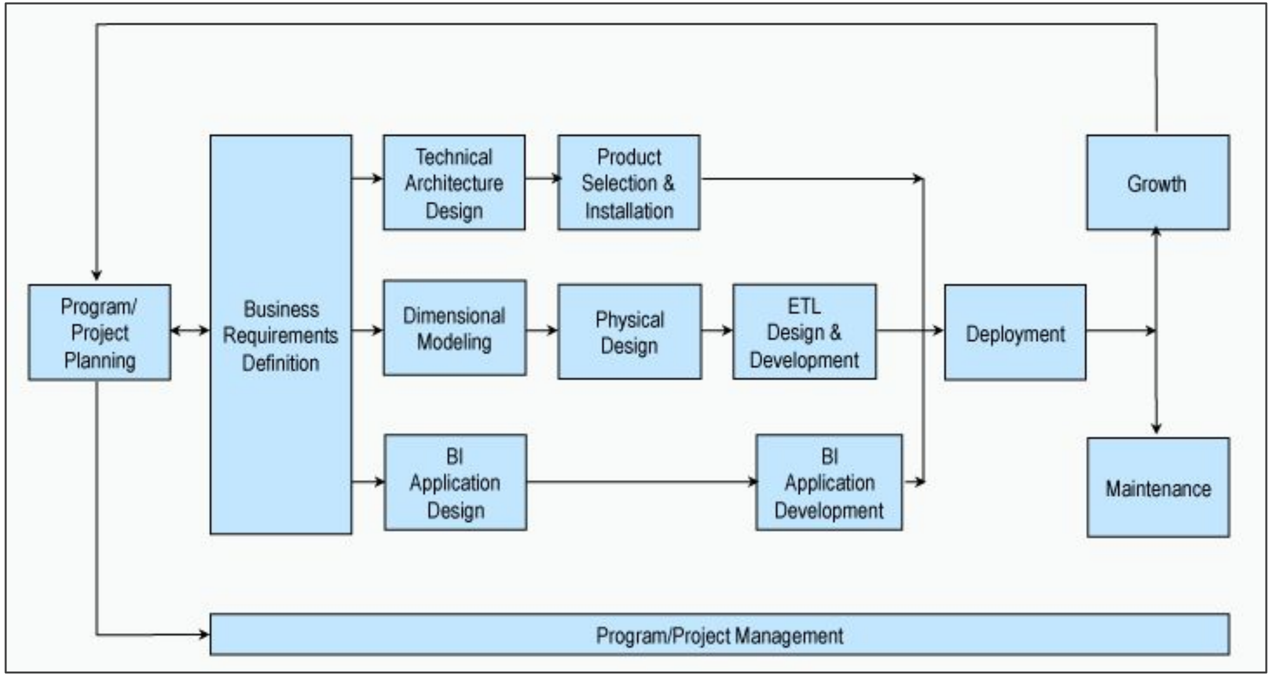
```
{
  "_id": 1,
  "name" : { "first" : "John", "last" : "Backus" },
  "contribs" : [ "Fortran", "ALGOL", "Backus-Naur Form", "FP" ],
  "awards" : [
    {
      "award" : "W.W. McDowell Award",
      "year" : 1967,
      "by" : "IEEE Computer Society"
    }, {
      "award" : "Draper Prize",
      "year" : 1993,
      "by" : "National Academy of Engineering"
    }
  ]
}
```

## Why is Distributed Data Processing important?

- **Scalable** - Far cheaper than Centralized Data Processing for Big Data
    - 10 machines (nodes) each with 8GB RAM is much cheaper than one machine with 80 GB RAM
        - This is exponential as data gets larger
- **Reliable** - More reliable than Centralized Data Processing
    - Each machine has a chance of failure, especially when the machine is running near computing capacity
        - In Distributed Data Processing, one node can fail but the network can carry-on successfully
- **Fault Tolerant** - data is replicated and processing jobs are re-startable

## The Kimball Lifecycle

**Purpose**: A roadmap to effectively complete a Data Warehousing project


Program/ Project Planning
Business Requirements Definition
Technical Architecture Design
Product Selection & Installation
Dimensional Modeling
Physical Design
ETL Design & Development
BI Application Design
BI Application Development
Deployment
Growth
Maintenance
Program/Project Management

## Week 13 Class Overview:

1. Dashboard filters in Tableau
2. Hadoop, Hive, and Spark
3. NoSQL Databases
4. Final Exam topics
5. **Scheduling an ETL Pipeline with Airflow**

## What is Airflow?

- Airflow is an orchestrator - run tasks in set orders at set times
- Benefits of Airflow for ETL pipeline:
  - Dynamic
  - Scalable
  - User Interface
- What is a DAG? What is an operator?
- Let’s look at a DAG, then build this DAG in your own environment

## Homework:

1. Final Project Milestone #5 on Blackboard, due Friday, 12/6/22 at 6:00pm ET