# ANALYTICS TOOLS FOR HADOOP

Alex Campos | Big Data Professional

https://www.linkedin.com/in/campossalex/
@campossalex

# Agenda

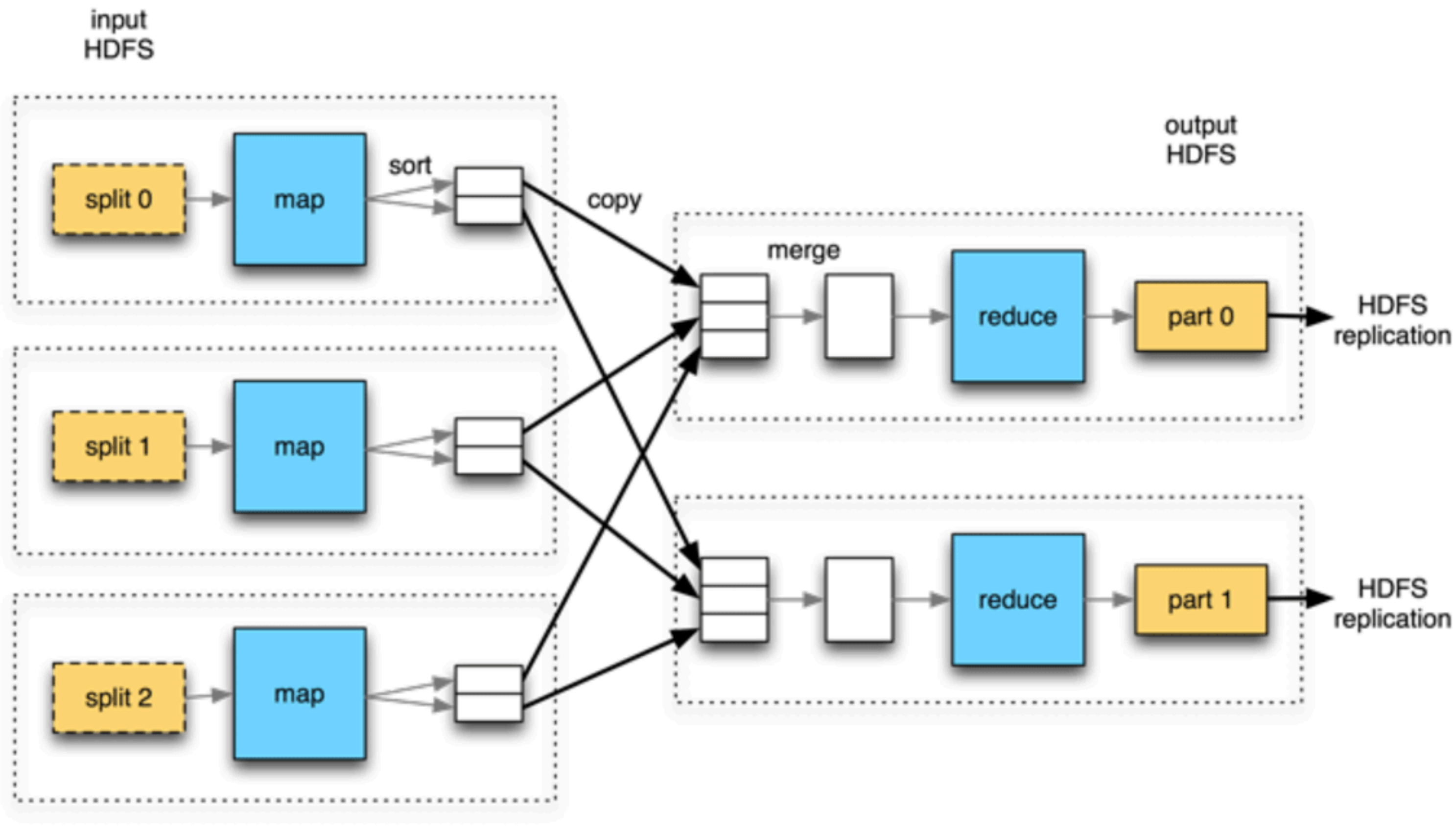
- MapReduce
- Analytics Tools
- Choosing the Right Engine
- Data Layers
- Dos and don’ts

# MapReduce

# MapReduce

- Programming model.
- Mostly written in Java (also Python)
- Two phases: map y reduce (shuffle and sort).
- Each map process one input split (block).
- Reducer aggregate mappers results.
- Key -> value

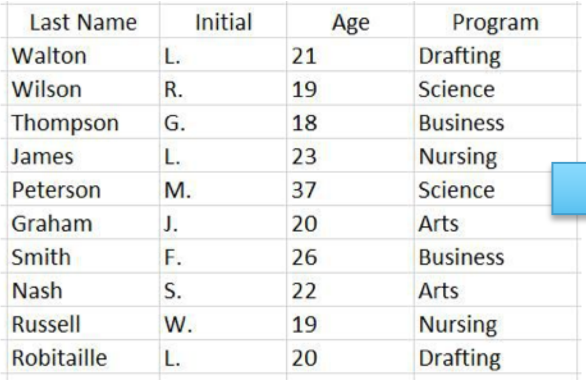
# MapReduce - Phases



# MapReduce - Phases

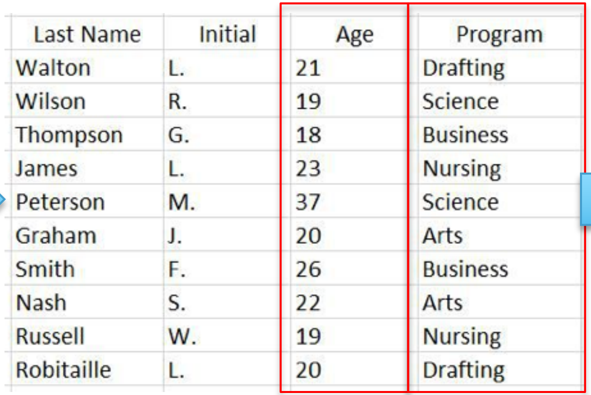
## Calculate program age average?

Input



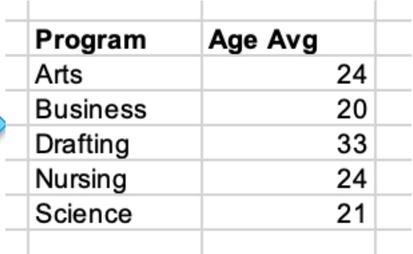
| Last Name | Initial | Age | Program |
|---|---|---|---|
| Walton | L. | 21 | Drafting |
| Wilson | R. | 19 | Science |
| Thompson | G. | 18 | Business |
| James | L. | 23 | Nursing |
| Peterson | M. | 37 | Science |
| Graham | J. | 20 | Arts |
| Smith | F. | 26 | Business |
| Nash | S. | 22 | Arts |
| Russell | W. | 19 | Nursing |
| Robitaille | L. | 20 | Drafting |

Map



| Last Name | Initial | Age | Program |
|---|---|---|---|
| Walton | L. | 21 | Drafting |
| Wilson | R. | 19 | Science |
| Thompson | G. | 18 | Business |
| James | L. | 23 | Nursing |
| Peterson | M. | 37 | Science |
| Graham | J. | 20 | Arts |
| Smith | F. | 26 | Business |
| Nash | S. | 22 | Arts |
| Russell | W. | 19 | Nursing |
| Robitaille | L. | 20 | Drafting |

Reduce



| **Program** | **Age Avg** |
|---|---|
| Arts | 24 |
| Business | 20 |
| Drafting | 33 |
| Nursing | 24 |
| Science | 21 |

Shuffle
and
sort

# Analytics Tools

# Motivation to Hive/Impala

- Limitation of MR
  - Have to use M/R model
  - Not Reusable
  - Error prone
  - For complex jobs:
    - Multiple stage of Map/Reduce functions
    - Just like ask dev to write specify physical execution plan in the database
- Provide higher-level language to facilitate large-data processing
- Higher-level language “compiles down” to Hadoop jobs

# Hive

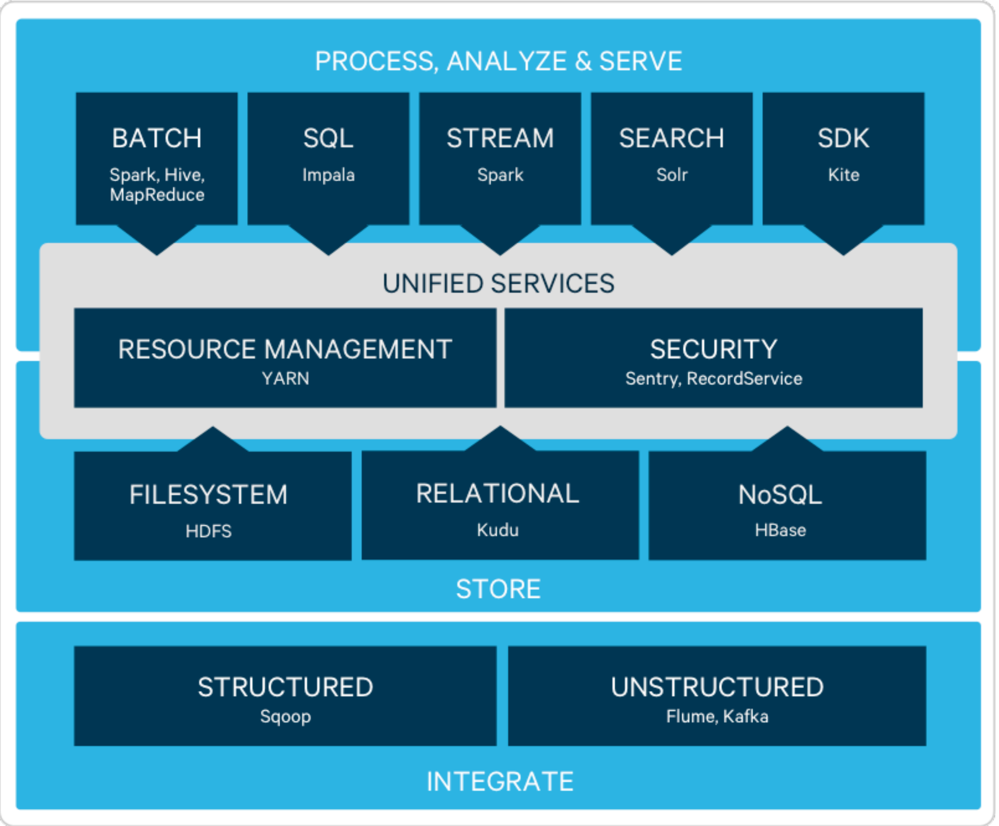
- Developed in Facebook
- “Relational database” built on Hadoop
  - Maintains list of table schemas
  - SQL-like query language (HiveQL)
  - Can call Hadoop Streaming scripts from HiveQL
  - Supports table partitioning, clustering, complex data types, some optimizations

# Hive

Hive uses a SQL-like language called HiveQL

```
SELECT zipcode, SUM(cost) AS total
FROM customers
JOIN orders
ON (customers.cust_id = orders.cust_id)
WHERE zipcode LIKE '63%'
GROUP BY zipcode
ORDER BY total DESC;
```

# Hive


PROCESS, ANALYZE & SERVE
BATCH
Spark, Hive, MapReduce
SQL
Impala
STREAM
Spark
SEARCH
Solr
SDK
Kite
UNIFIED SERVICES
RESOURCE MANAGEMENT
YARN
SECURITY
Sentry, RecordService
FILESYSTEM
HDFS
RELATIONAL
Kudu
NoSQL
HBase
STORE
STRUCTURED
Sqoop
UNSTRUCTURED
Flume, Kafka
INTEGRATE

# Hive – Data Storage

- Tables are logical data units; table metadata associates the data in the table to hdfs directories.
- Hdfs namespace: tables (hdfs directory), partition (hdfs subdirectory), buckets (subdirectories within partition)
- /user/hive/warehouse/test_table is a hdfs directory

# Hive – Architecture

- Metastore: stores system catalog
- Query compiler: Compiles HiveQL into a directed in map/reduce tasks
- Client components: CLI, web interface, jdbc/odbc inteface
- Extensibility interface include SerDe, User Defined Functions and User Defined Aggregate Function.

# Hive – Components

- Shell Interface: Like the MySQL shell
- Driver:
  - Session handles, fetch, execution
- Complier:
  - Parse, plan, optimzie
- Execution Engine:
  - Run map or reduce

# Hive – Application

- Log processing
  - Daily Report
  - User Activity Measurement
- Data/Text mining
  - Machine learning (Training Data)
- Business intelligence

# Hive – Example

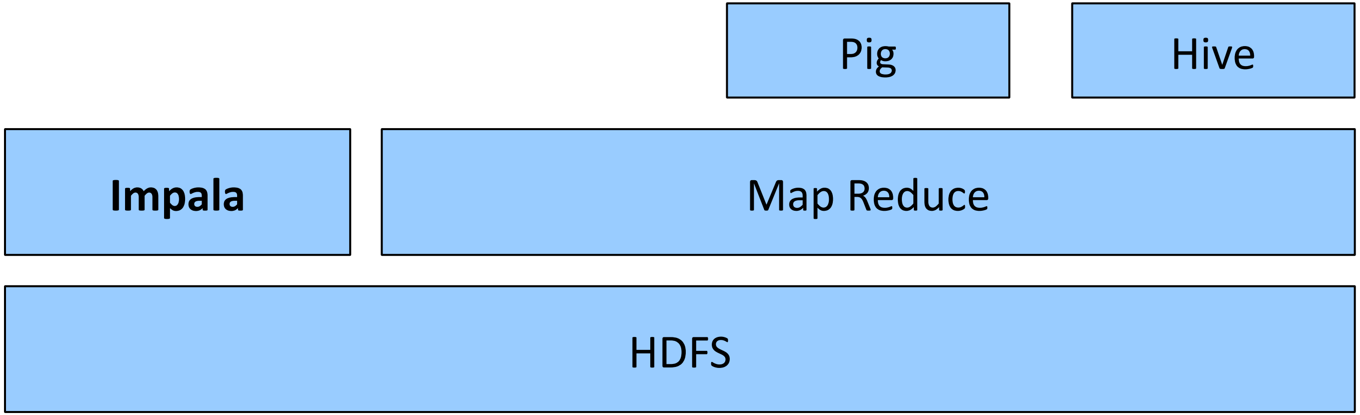
- Find all page views coming from xyz.com on March 31st:

```
SELECT page_views.*
FROM page_views
WHERE page_views.date >= '2008-03-01'
AND page_views.date <= '2008-03-31'
AND page_views.referrer_url like '%xyz.com';
```

- Hive only reads partition `2008-03-01,*` instead of scanning entire table

# Impala

- Massive parralel processing (MPP) database engine, developed by Cloudera.
- Integrated into Hadoop stack on the same level MapReduce, and not above it (as Hive and Pig)



Pig

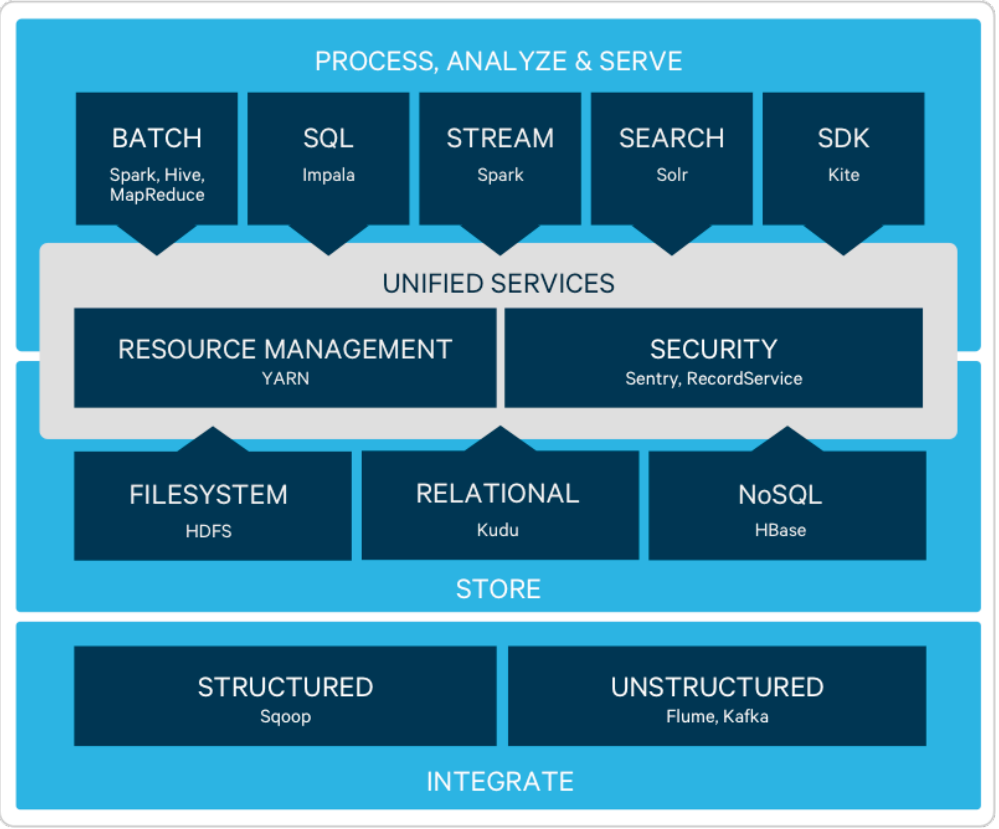
Hive

**Impala**

Map Reduce

HDFS

# Impala


PROCESS, ANALYZE & SERVE
BATCH
Spark, Hive, MapReduce
SQL
Impala
STREAM
Spark
SEARCH
Solr
SDK
Kite
UNIFIED SERVICES
RESOURCE MANAGEMENT
YARN
SECURITY
Sentry, RecordService
FILESYSTEM
HDFS
RELATIONAL
Kudu
NoSQL
HBase
STORE
STRUCTURED
Sqoop
UNSTRUCTURED
Flume, Kafka
INTEGRATE

# Impala - Why

- Reuse the data and metadata of Hive
- In the same time – MapReduce is not must
- Impala process data in Hadoop cluster **without** using MapReduce

# Impala - More

- Bypass MapReduce latency
- Caching hdfs file blocks location
- Simple query engine. It actually doing things which can be done in memory.
- Support UDF
- Fine tuning options
  - Caching some

# Impala - Example

```
select
        l_returnflag,
        l_linestatus,
        sum(l_quantity),
        sum(l_extendedprice),
        sum(l_extendedprice * (1 - l_discount)),
        sum(l_extendedprice * (1 - l_discount) * (1 + l_tax)),
        avg(l_quantity),
        avg(l_extendedprice),
        avg(l_discount),
        count(1)
from
        lineitem
where
        l_shipdate<='1998-09-02'
group by
        l_returnflag,
        l_linestatus
```

# Impala – Input Formats

- There are scanners for the following types:
- RCFile
- Parquet (native dremel format)
- CSV
- AVRO
- Sequence File

# Choosing the Right Engine

# Choosing the Right Engine

HIVE

MapReduce

OR

Spark

Batch Processing

Impala

BI and SQL Analytics

Spark SQL

Procedural Development

# Choosing the Right Engine

- **Let's first look at *similarities* between Hive, Pig, and Impala**
  - Queries expressed in high-level languages
  - Alternatives to writing MapReduce code
  - Used to analyze data stored on Hadoop clusters
- **Impala shares the metastore with Hive**
  - Tables created in Hive are visible in Impala (and vice versa)

# Choosing the Right Engine

- **Hive and Pig answer queries by running MapReduce jobs**
  - MapReduce is a general-purpose computation framework
  - Not optimized for executing interactive SQL queries
- **MapReduce overhead results in high latency**
  - Even a trivial query takes 10 seconds or more
- **Impala does not use MapReduce**
  - Uses a custom execution engine built specifically for Impala
  - Queries can complete in a fraction of a second

# Choosing the Right Engine

- **Hive, Pig, and Impala also support**
  - Execute queries via interactive shell or command line
  - Grouping, joining, and filtering data
  - Read and write data in multiple formats
- **Impala currently lacks some Hive and Pig features**
  - More details later in this chapter
- **Hive and Pig are best suited to long-running batch processes**
  - Particularly data transformation tasks
- **Impala is best for interactive/ad hoc queries**

# Choosing the Right Engine

- **Custom extensions not currently supported in Impala**
  - User-defined functions (UDFs) and external transformations
  - File and row format support (SerDes)

# Choosing the Right Engine

- **Impala is a high-performance SQL engine**
  - Runs on Hadoop clusters
  - Reads and writes data in HDFS or HBase tables
- **Queries are expressed in SQL dialect similar to HiveQL**
- **Primary difference compared to Hive/Pig is speed**
  - Impala avoids MapReduce latency and overhead
- **Impala is best suited to ad hoc/interactive queries**
  - Hive and Pig are better for long-running batch processes
  - Impala does not currently support all features of Hive

# Data Layers

# Data Layers

Data Lake or Data Swamp?

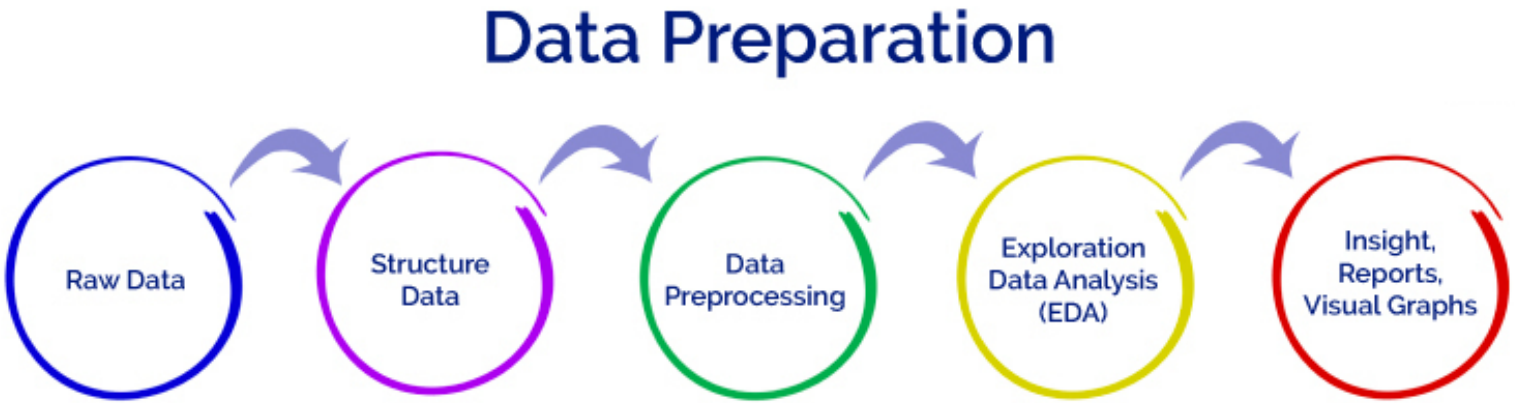
# Data Layers

or Data Recycle?

# Data Layers

- Optimize data consumption, improving analytics experience
- Create specialized datasets
- Govern data from ingestion to analytics (lineage, quality, metadata, security)
- Improve data reusage – Data as a Service

# Data Wrangling



Data Preparation

Raw Data

Structure Data

Data Preprocessing

Exploration Data Analysis (EDA)

Insight, Reports, Visual Graphs

# Data Layers

RAW

- Mainly from ingestion pipeline
- Minimum transformations or cleansing
- Not optimized for final consumption
- Vary in formats and structure
- Keep granularity

DIMENSIONAL

- Output from data preparation and analytics processes
- Combination and enrichment of more data sources
- KPI and metrics generation
- Attend specific requirements
- Build “common” usage datasets

# Data Layers

## USER

- View or materialized datasets
- Attend departments or areas
- Visible by visualization applications, human beings, business processes, or services
- Designed for better performance

## EXPERIMENTAL

- Aka Sandbox
- Area of exploration and freedom for new development
- Mash-up and produce new datasets
- Rapid and agile prototyping
- Usually DS Teams are the primary users

# Dos and don’ts

# Dos and don’ts

- Leverage compression
  - Parquet is your best friends, saving storage space
  - Optimized for analytics tools (Hive, Impala, Spark, etc)
  - Better for network traffic
- Partition your data
  - Avoid full scan
  - Think about common data access
  - Find the better granularity

# Dos and don’ts

- Use data layers
    - Determine where users query data
    - Track small files
- Hive and Impala are not optimized for “pick one” queries
    - Leverage other frameworks for “specific” data access

# Dos and don’ts

- Minimize the overhead of transmitting results back to client
  - Aggregate data
  - Save results back to data storage layer
  - Filter and sample data for verification
- Compute table stats

# THANKS!