Building a Fault-Tolerant ETL Pipeline for Claims CAFé

Internship Presentation - Summer 2018

Kim Hammar

Data Analytics Engineer

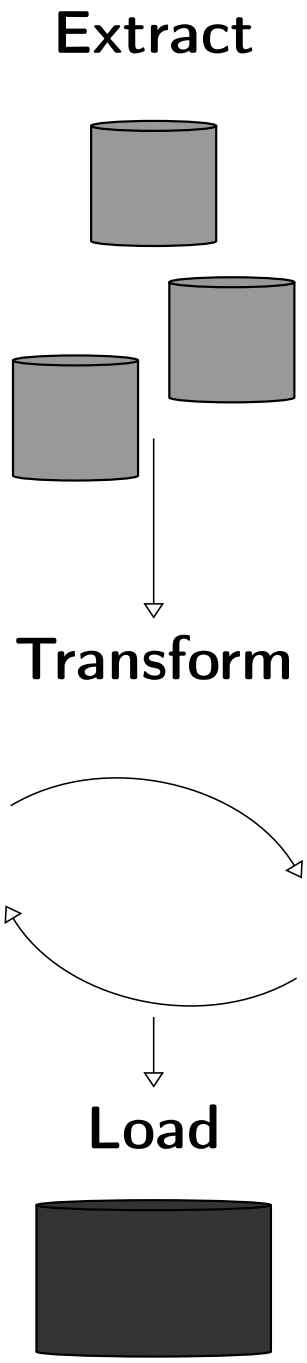
khamq@allstate.com ***or*** *kimham@kth.se*

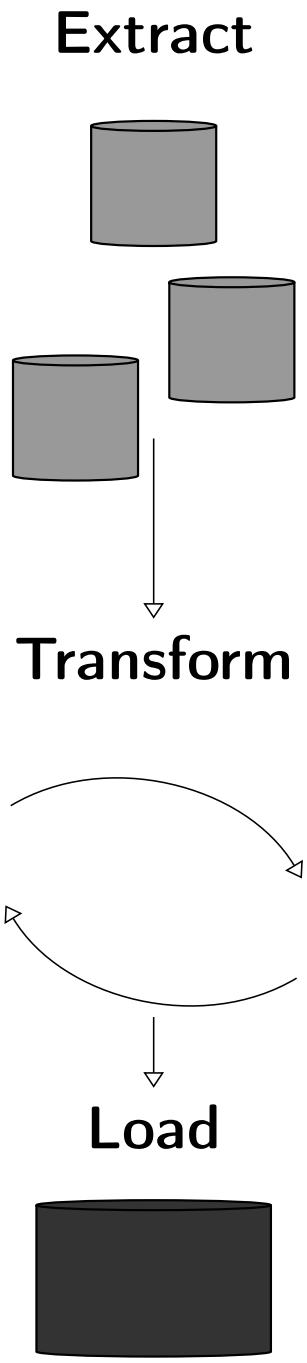
August 30, 2018

Allstate®

You're in good hands.

D³:
DATA
DISCOVERY
DECISION SCIENCE


Extract
Transform
Load



Corrupt data that cannot be parsed
Inconsistent Schema
Wrong File paths

Unexpected null values
Duplicates
Code bugs
Missing values
Inconsistent data types

Target database unavailable
Permission error
Scalability problems

Outline

What is Claims CAFé?

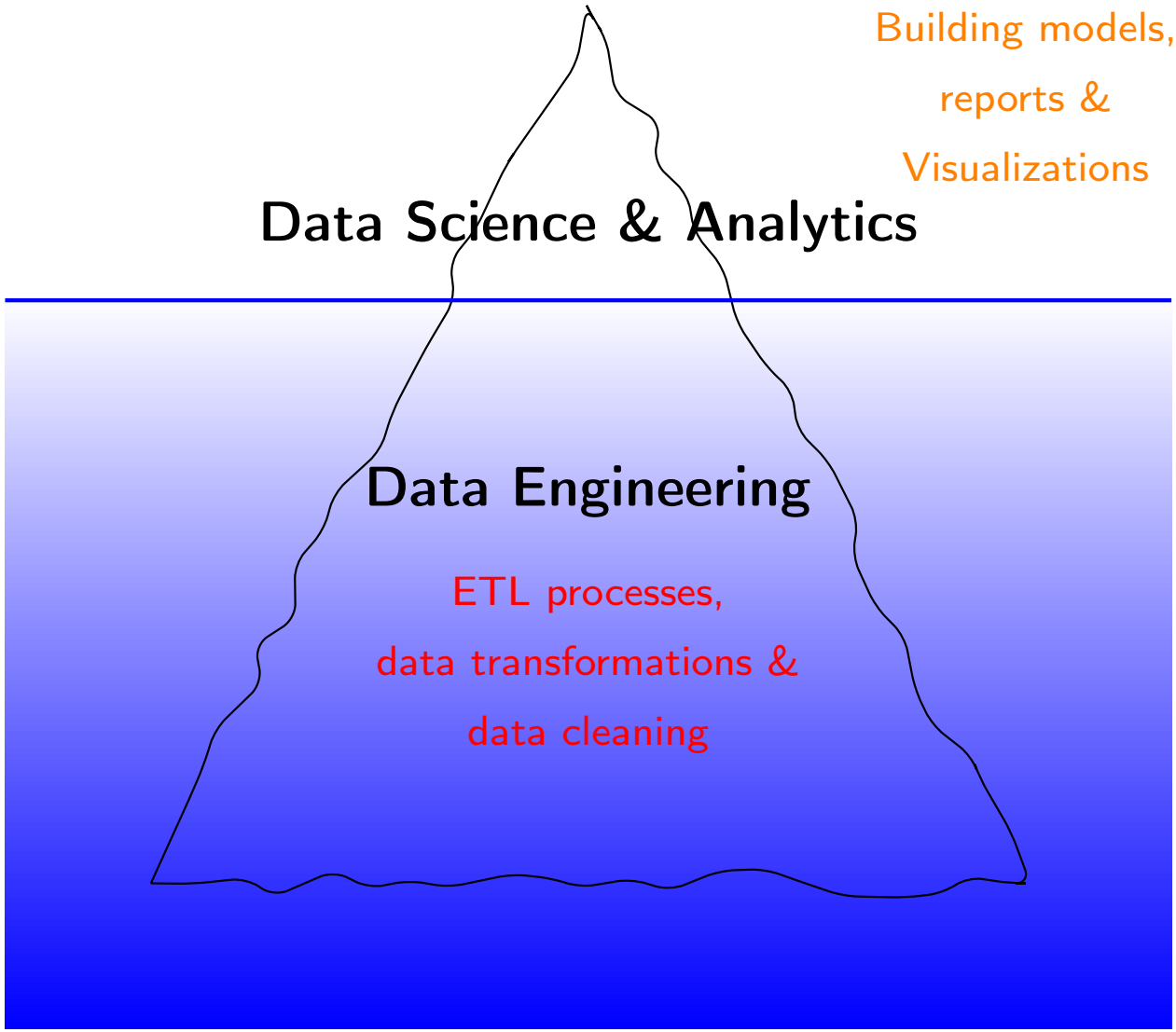
- **Claims CAFé: Claims (C)entral (A)nalytical (F)il(e)**

Claims

claimId	policy	participant	persVeh	...
B005	auto	John Doe	URK389 Audi	...
B007	home	A.Svensson	PTO291 Ford	...
B003	life	C.Strömbäck	RNU999 Volvo	...
B004	property	G.Åsbrink	WEM650 Benz	...
B002	health	L.Löfven	KQO209 Tesla	...

- A one stop shop for claims analytics.
- Optimized for analytical use-cases by saving raw denormalized data in hadoop:
  - Increase scalability
  - Makes data processing easier: No more slow and complex SQL-Joins

What **Actually** Goes Into Building a Model

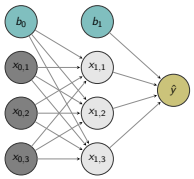


Now:

complex and slow join to pull data

transform & clean data create derived fields

Predict claim automation: Deep FNN model



complex and slow join to pull data

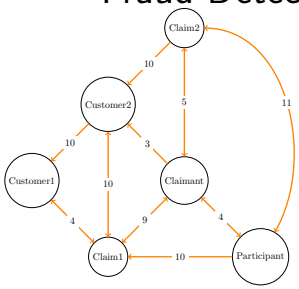
transform & clean data create derived fields

Customer report & dashboards

complex and slow join to pull data

transform & clean data create derived fields

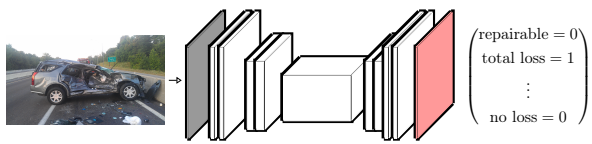
Link Analysis Fraud Detection



complex and slow join to pull data

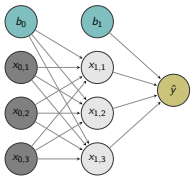
transform & clean data create derived fields

Classify Image Vehicle Loss Deep CNN Model



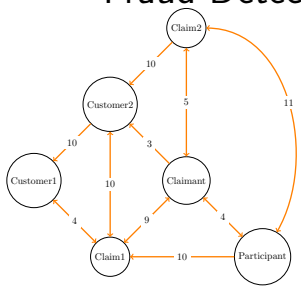
Now:

complex and slow join to pull data → transform & clean data create derived fields → Predict claim automation: Deep FNN model

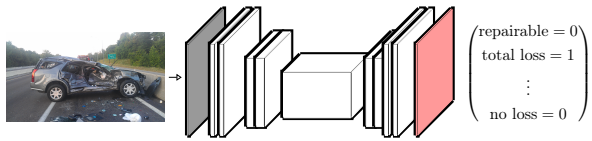


complex and slow join to pull data → transform & clean data create derived fields → Customer report & dashboards

complex and slow join to pull data → transform & clean data create derived fields → Link Analysis Fraud Detection

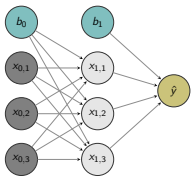


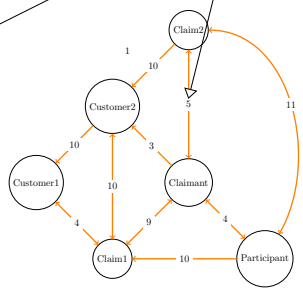
complex and slow join to pull data → transform & clean data create derived fields → Classify Image Vehicle Loss Deep CNN Model

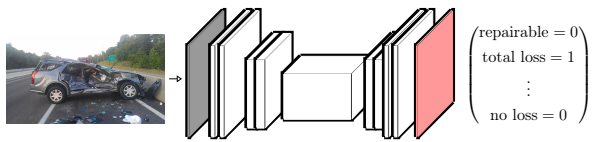


Future:

complex and slow join to pull data → transform & clean data create derived fields → Claims CAFé







What Goes Into Making CAFé

Making Coffee

Coffee Plants

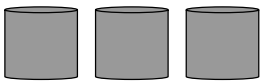
Harvest

Processing:
pulping the cherries,
filtering out bad ones,
sorting&drying

Prepare for consumption:
roasting,
grinding,
brewing

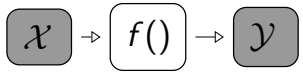
Consume:
drink,
sell

Making Claims CAFé



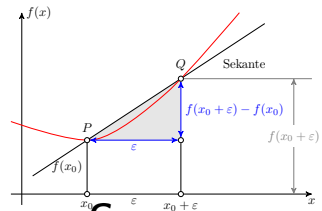
Raw Data Sources

ETL



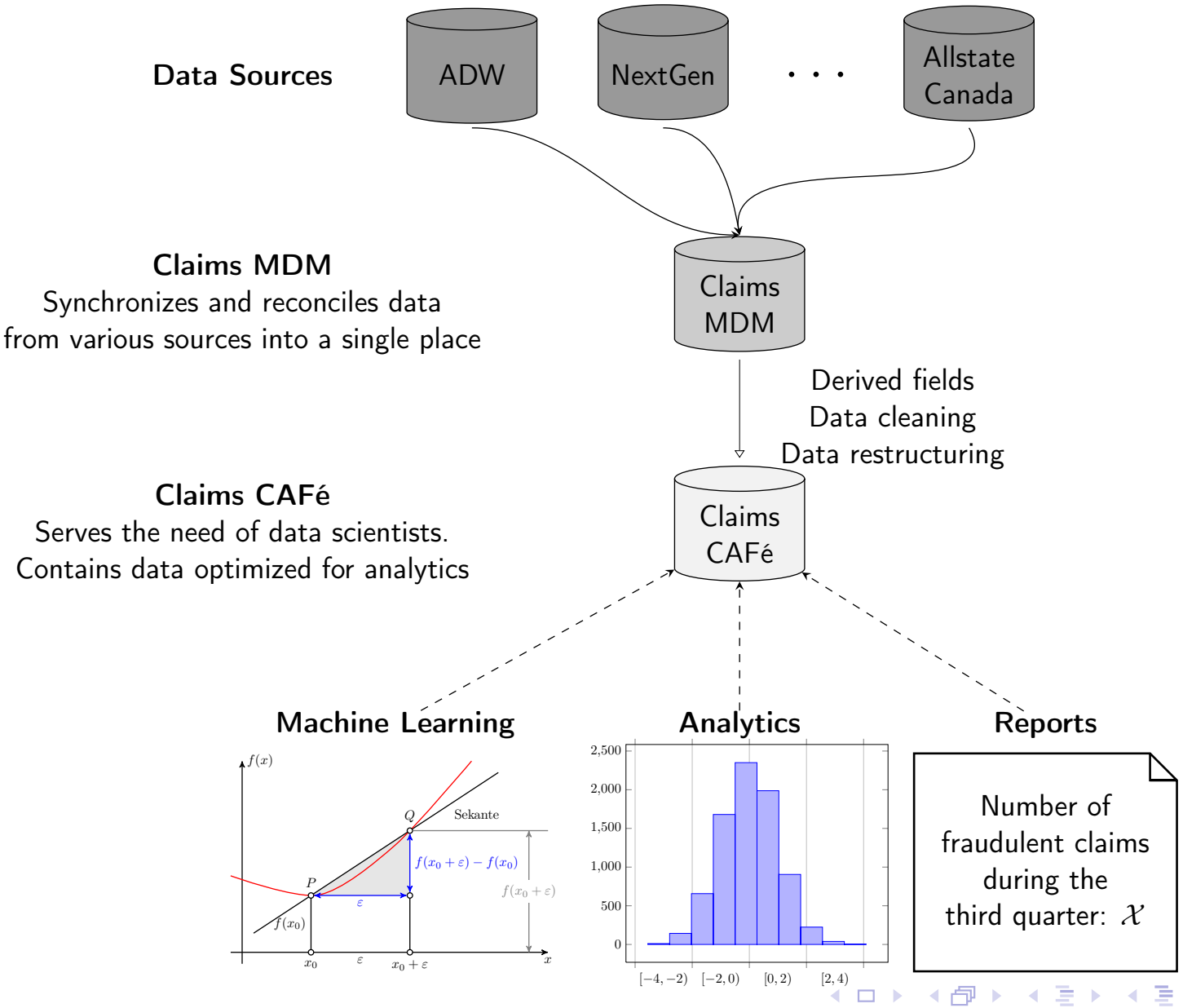
Processing:
fill in null values,
create derived fields,
data cleaning

Prepare for consumption:
flatten,
create different views,
save to Hive



Consume:
build models,
generate business reports

Claims CAFé Data Architecture



What Is a Claim From A Data Pespective?

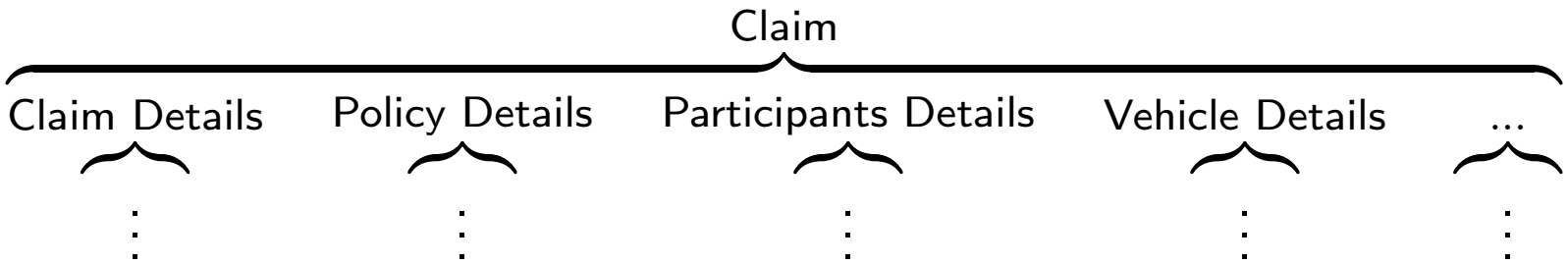


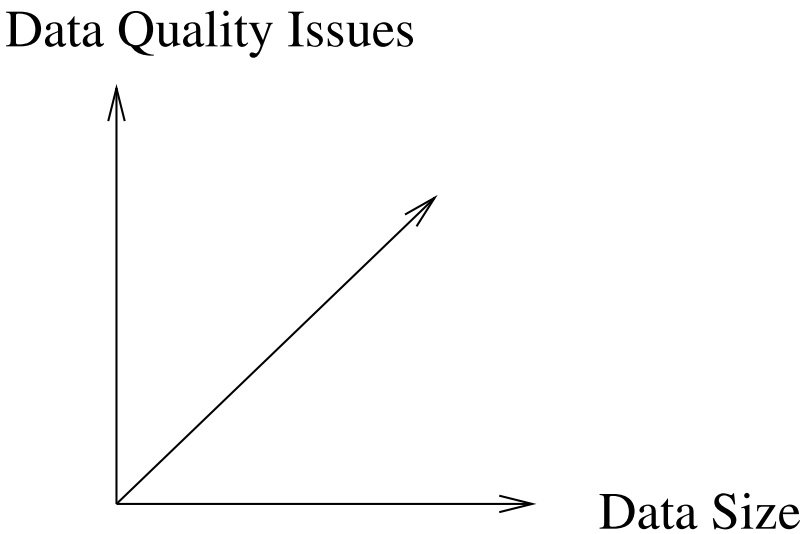
Figure: Some of the data that a single Claim in Claims CAFé contains.

Data Quality Assurance

- **Data Quality** is key to the success of Claims CAFé
- Examples of data quality issues:
  - Null values in the wrong place
  - Duplicates
  - Missing values
  - Inconsistent data types
  - ⋮

Why does my Model not work?

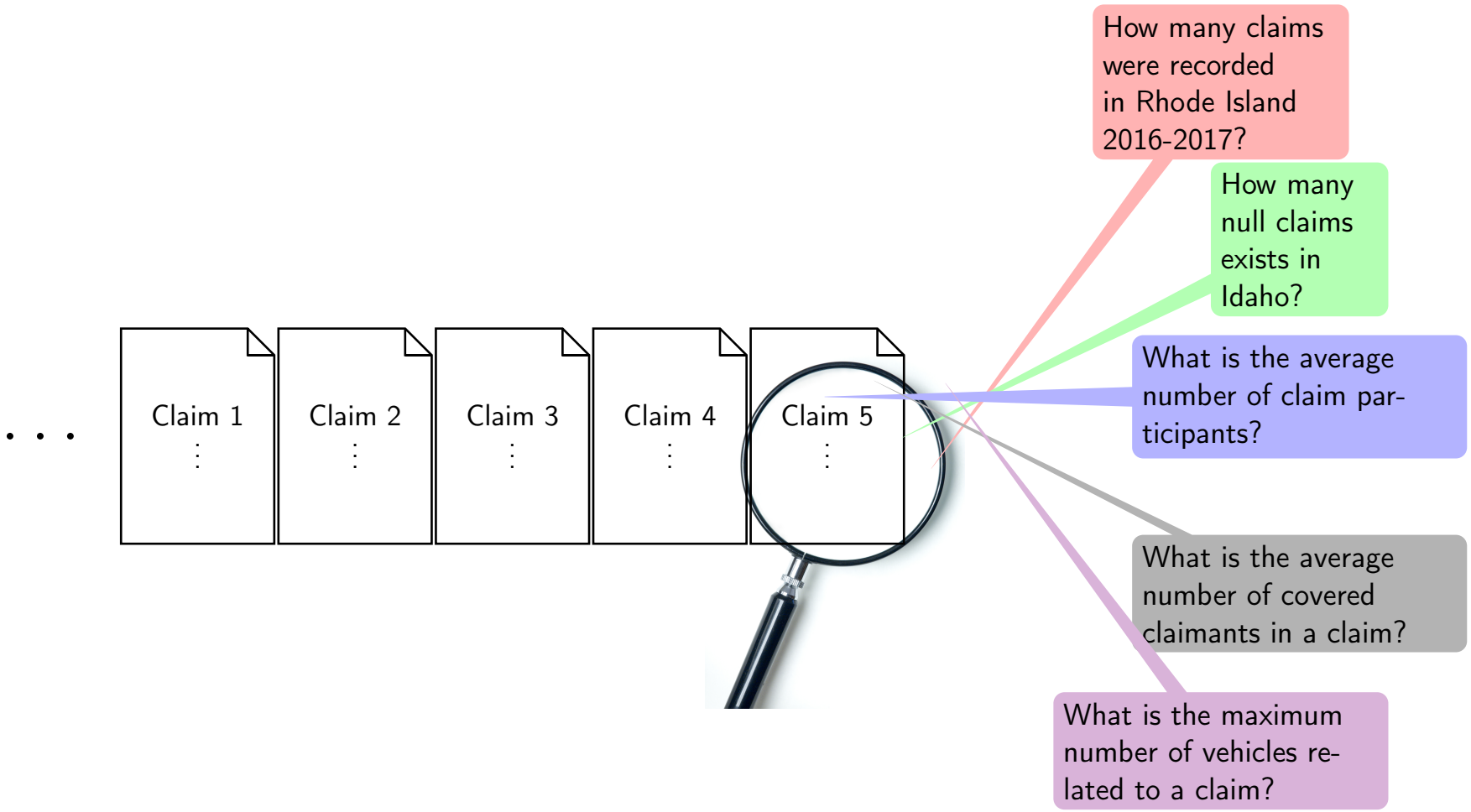
John Doe, Data Scientist



Data Quality Mechanisms Motivation

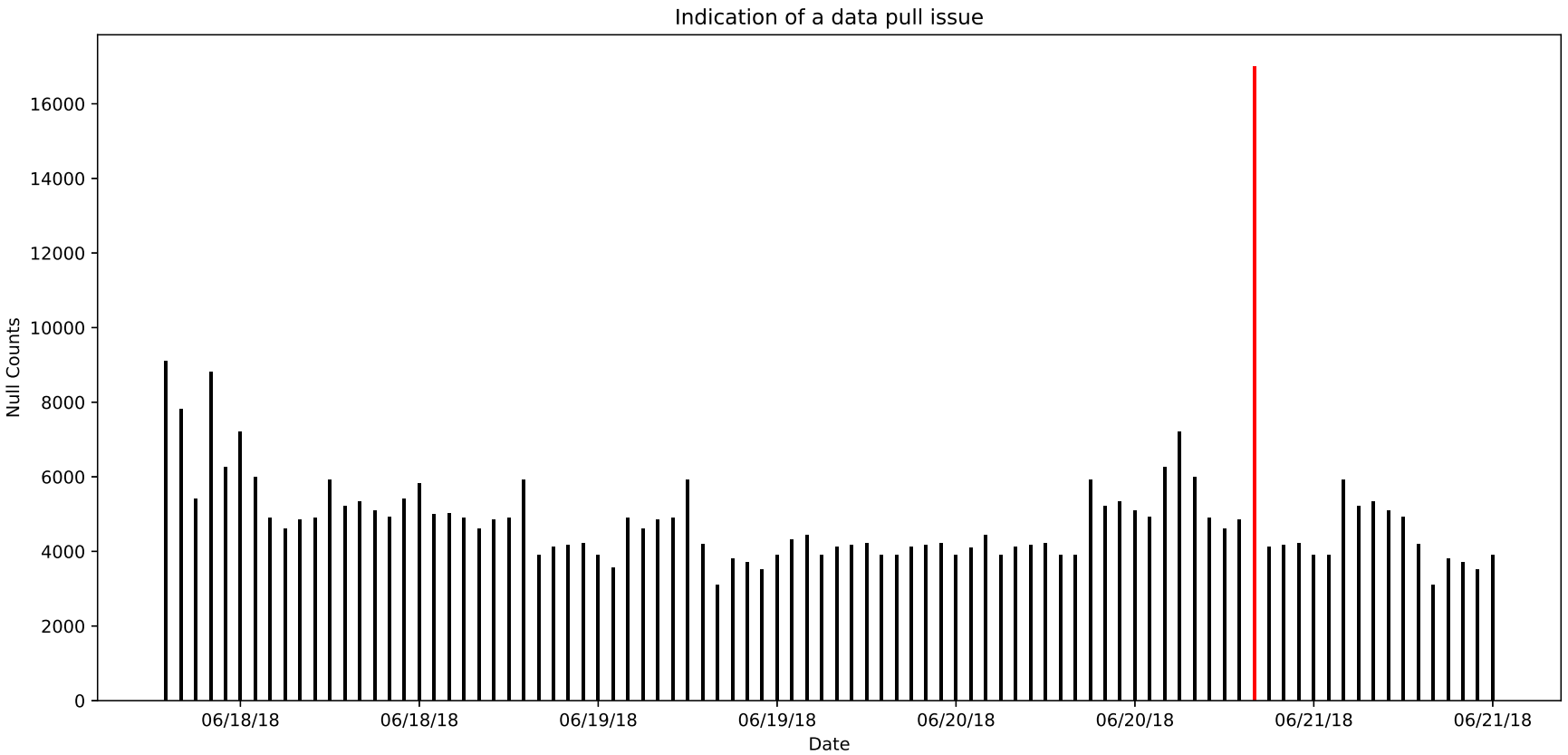
- *Failures are the **norm**, they are not an exception*
  - Optimistic error estimate: $P(\text{"CAFé error"}) = P(\text{"Data quality issue"}) \cup P(\text{"Hadoop failure"}) \cup P(\text{"Network failure"}) \cup P(\text{"ClaimsMDM failure"}) \cup P(\text{"CAFé code bug"}) \approx 1/1000 = 0.001$
  - CAFé stretch goal: near-real-time updates, say we pull data every 15 minutes $\implies$ 96 pulls per day $\implies$ 672 pulls per week
  - failure probability per week: $0.001 \times 672 = 0.672 \implies$ a failure will happen on average every other week
  - **We want built-in mechanisms in the CAFé pipeline to detect and deal with errors before they affect end-users: Tests!**

How to detect data quality issues? Know your data

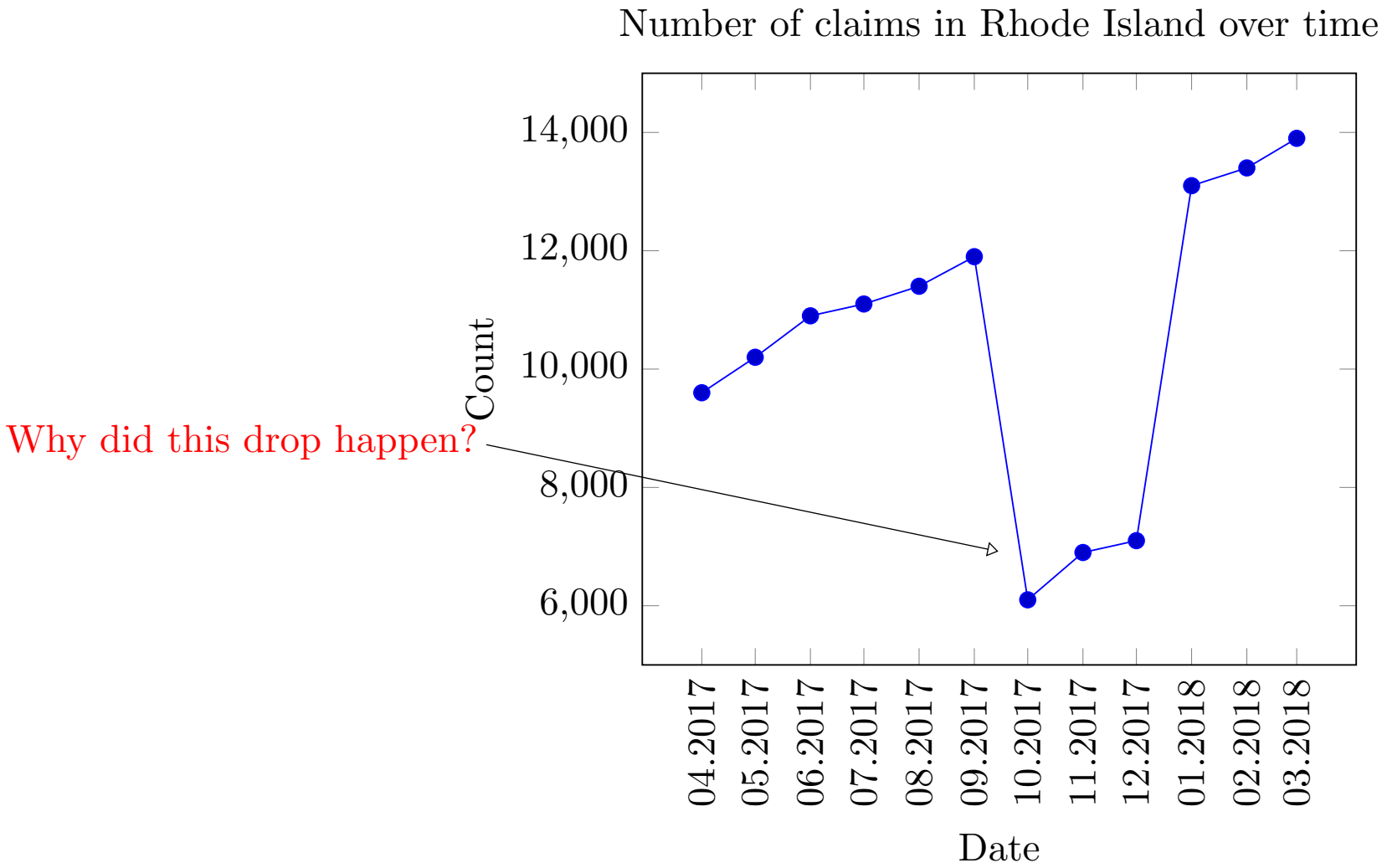


Anomaly Detection for Detecting Potential Data Issues (1/3)

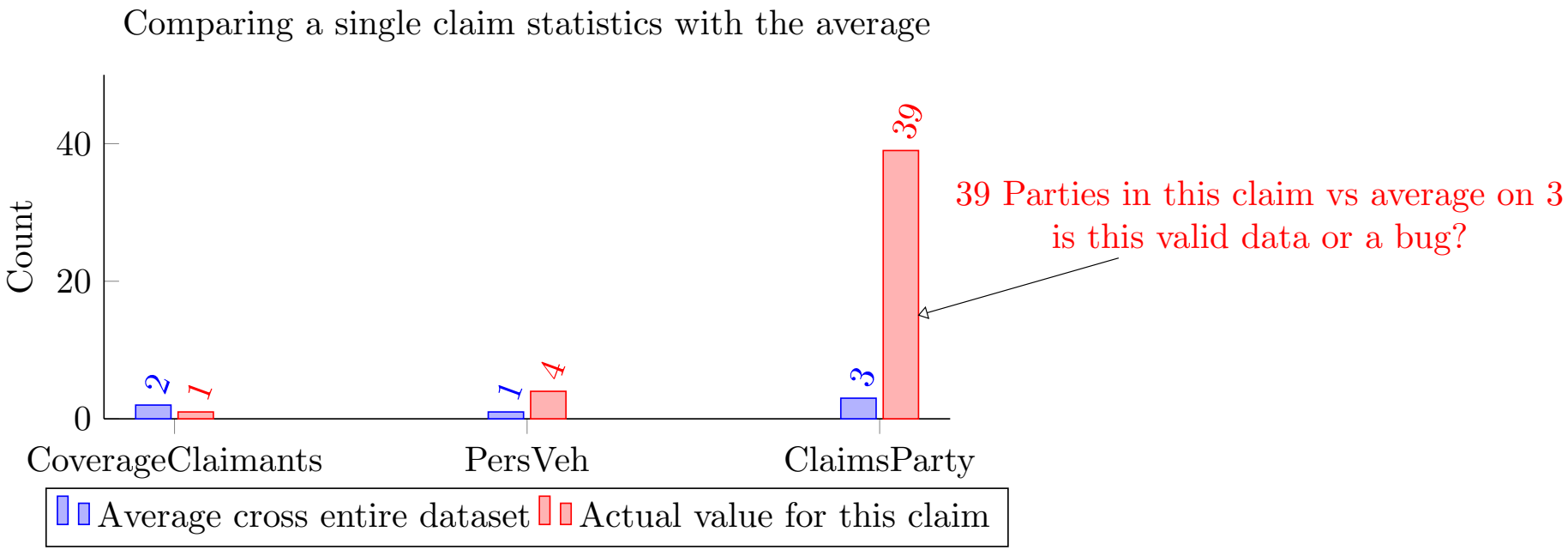
Spikes in the number of null values indicate a data issue.



Anomaly Detection for Detecting Potential Data Issues (2/3)



Anomaly Detection for Detecting Potential Data Issues (3/3)



Regression Tests Background (1/2)

What is regression testing?

- Regression tests verify **modifications** of a program or data.
- If the modification fail the tests, the program can ***regress*** back.

I just refactored the CAFé pipeline, how do I know that I didn't break anything?

I just pulled new data into CAFé, how do I know I did not introduce data quality issues?

Why do we want regression tests?

- It increases the confidence in making code and data changes
- We can detect bugs before they bother end-users
- We can avoid unnecessary work: If the tests fail we can abort early and save time.

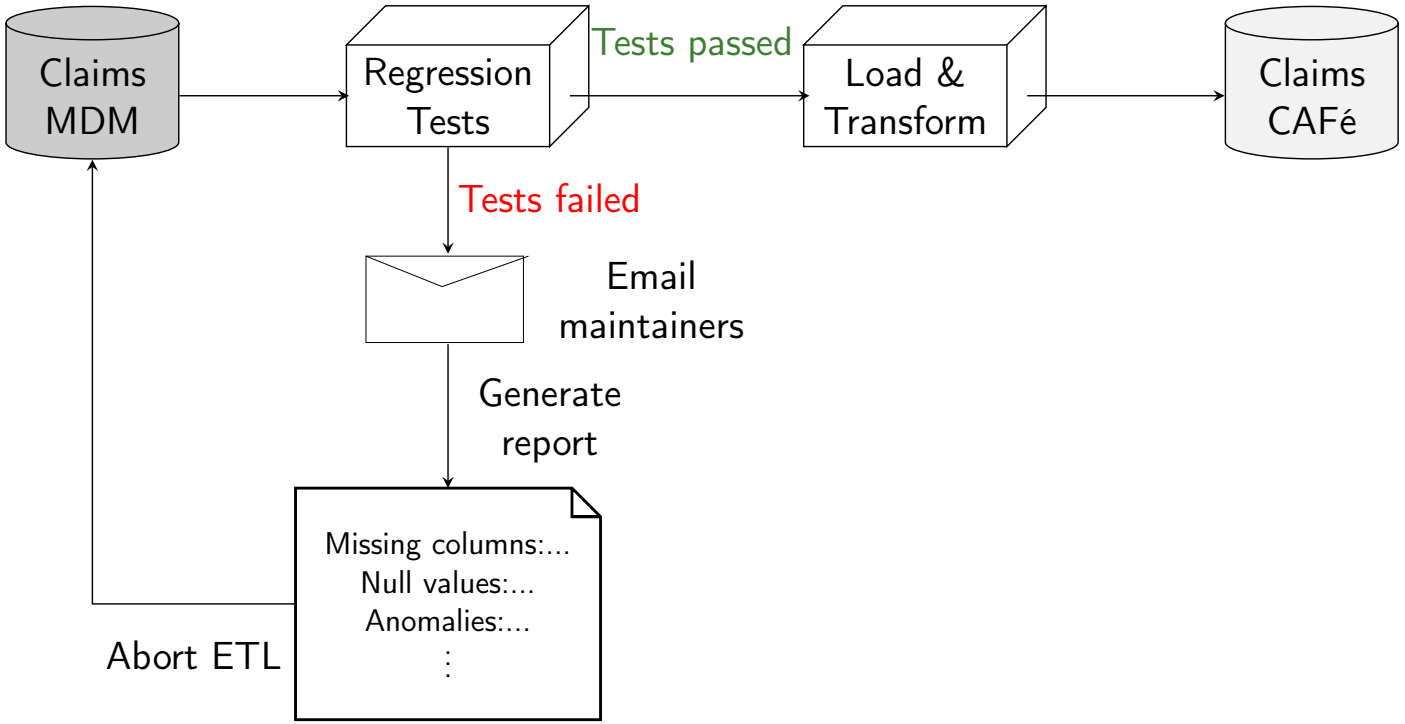
Regression Tests Background (2/2)

Naive ETL Pipeline For Updating Claims CAFé:

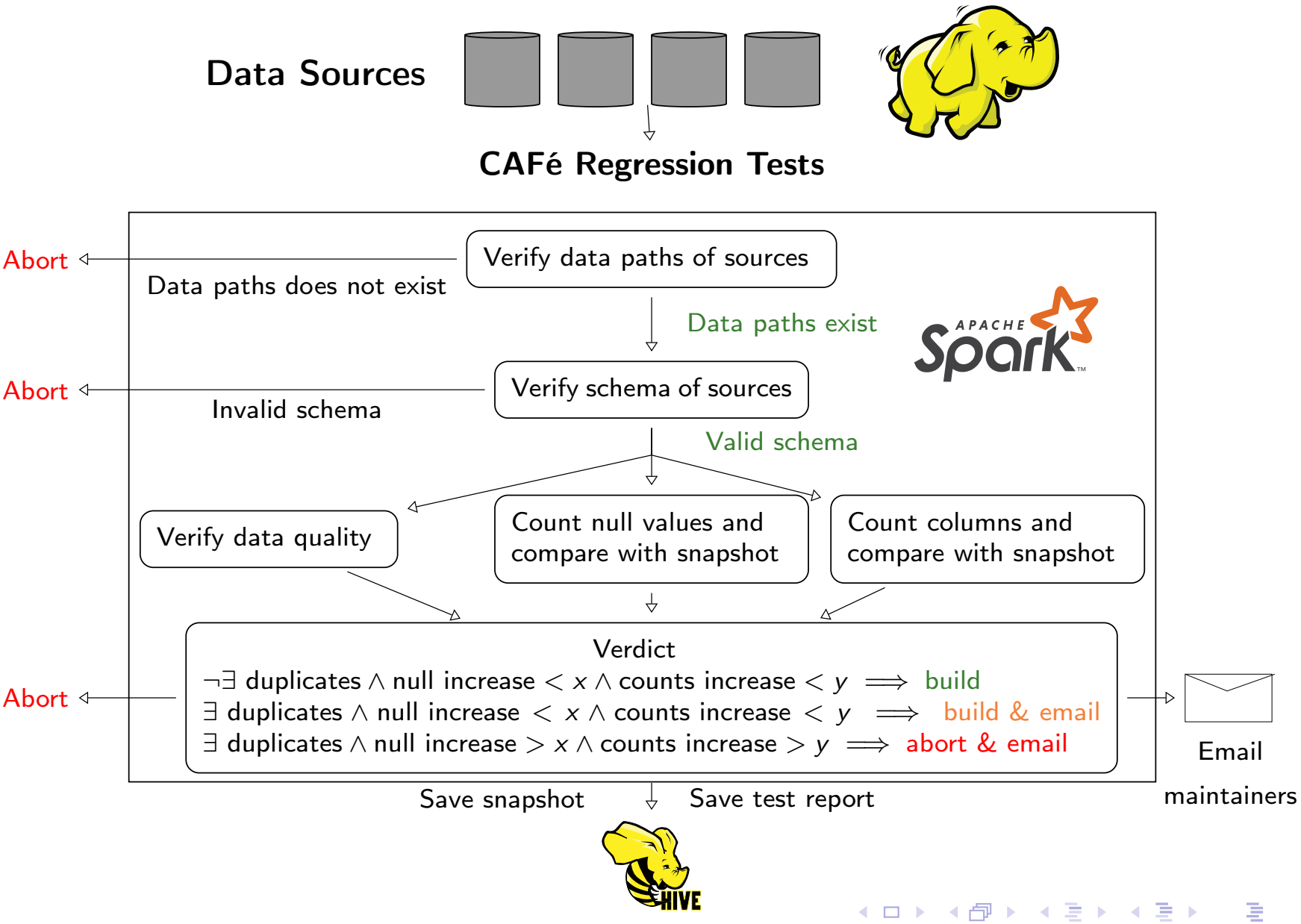
Regression Tests Background (2/2)

Naive ETL Pipeline For Updating Claims CAFé:

A More Robust ETL Pipeline:



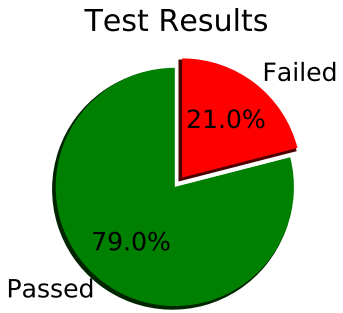
Regression Tests Pipeline for CAFé

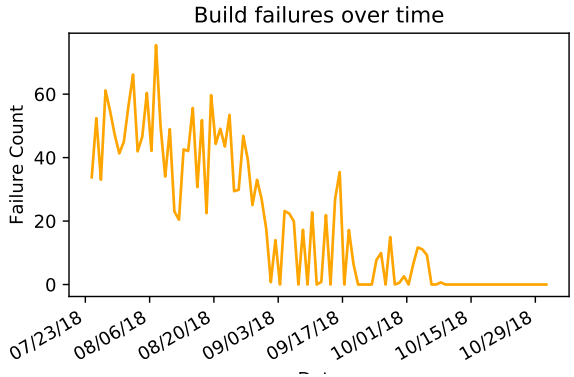


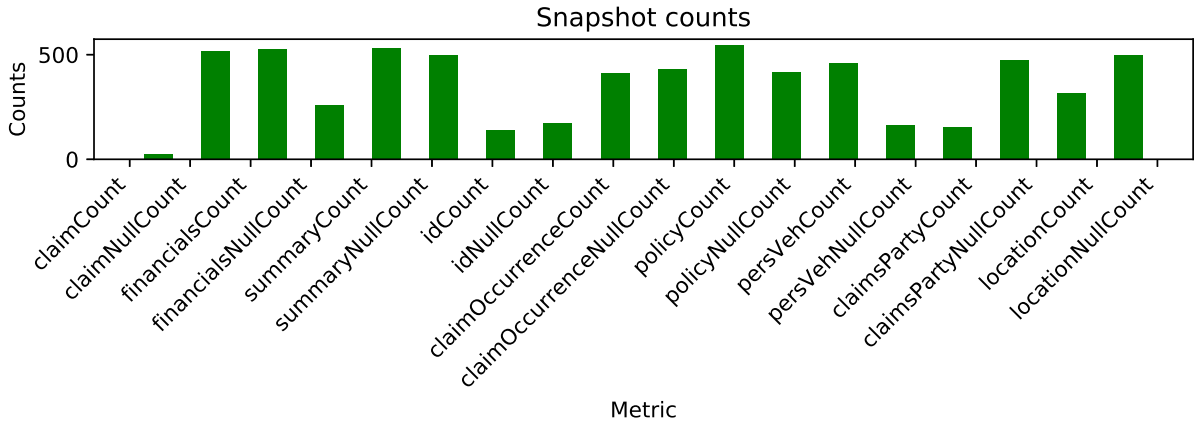
Regression Tests Reporting

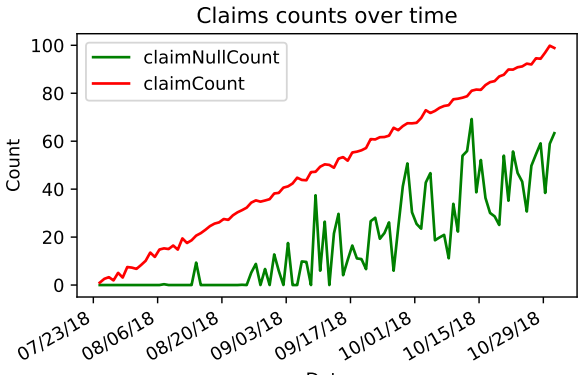
Test Report: 26/7-2018

valid_data_paths: passed, time: 7s
valid_source_schema: passed, time: 19s
no_duplicates: failed, time: 26s
claim_counts: passed, time: 41s
claim_null_counts: passed, time: 38s
⋮









Data Quality Mechanisms: Putting It All Together

Push Commit

Trigger Jenkins Job

Merge Master Branch

Build → Unit tests → Regression Tests

Jenkins

Daily Data Pull

Trigger Jenkins Job

Cluster

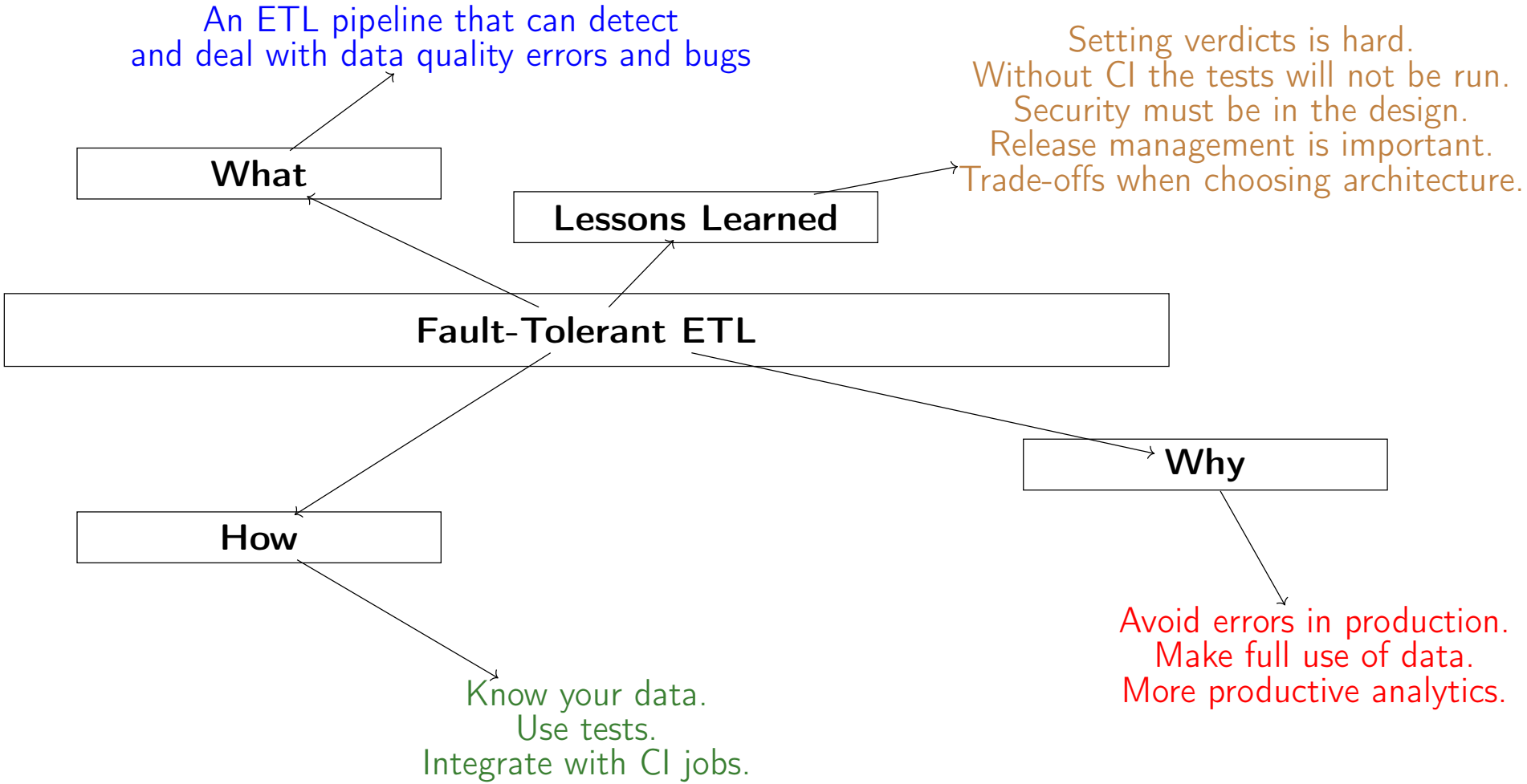
Abort

Build → Unit tests → Regression Tests

Update CAFé

DEMO

Conclusion


An ETL pipeline that can detect
and deal with data quality errors and bugs
What
Setting verdicts is hard.
Without CI the tests will not be run.
Security must be in the design.
Release management is important.
Trade-offs when choosing architecture.
Lessons Learned
Fault-Tolerant ETL
Why
How
Avoid errors in production.
Make full use of data.
More productive analytics.
Know your data.
Use tests.
Integrate with CI jobs.

Thank You!
Questions?

Links

- Claims CAFé Schema Repository[1]
- Claims CAFé Schema Confluence Page[2]
- Claims CAFé Data Pipeline and Tests Repository[3]
- Claims CAFé Tests Pipeline Confluence Page[4]
- Claims CAFé Continuous Integration Confluence Page[5]
- Claims CAFé API Documentation Confluence Page[6]

[1] https://github.allstate.com/d3-cafe/ClaimsCAFeSchema

[2] http://conflu.allstate.com/display/DOMF2/Schema

[3] https://github.allstate.com/d3-cafe/CAFe

[4] http://conflu.allstate.com/display/DOMF2/Tests

[5] http://conflu.allstate.com/display/DOMF2/Continuous+Integration

[6] http://conflu.allstate.com/display/DOMF2/API+Documentation