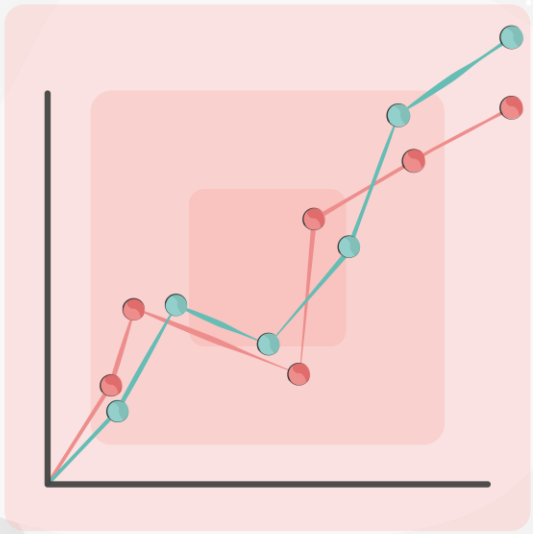
Data engineering

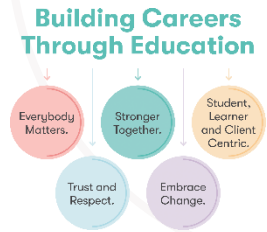
Module: Data pipelines

Topic: Testing strategies and debugging techniques



Welcome to today's webinar.

Ice breaker



Discussion…

- How are you feeling today? Motivated, happy, etc.?
- What is your key takeaway from the e-learning topic?
- What is one key skill or insight you hope to gain from today's session on testing strategies and debugging techniques?

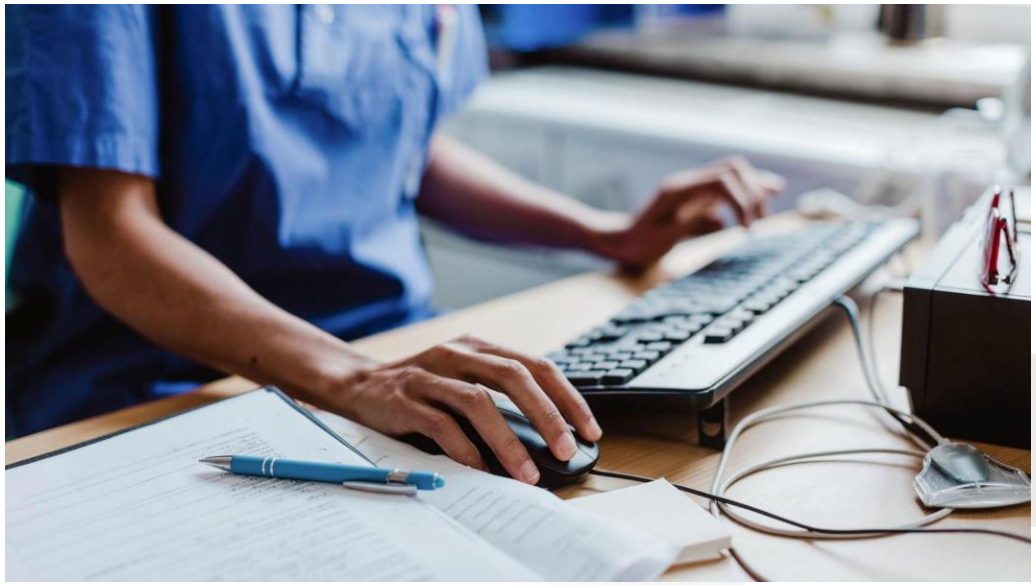
Submit your responses to the chat or turn on your microphone

BPP

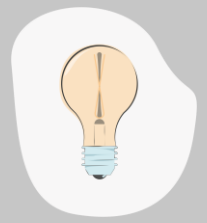
The £10 Billion NHS IT Disaster

When testing and oversight fail at scale

- Largest public-sector IT project in UK history
- Aimed to modernise NHS data systems
- Poor testing, stakeholder resistance, and delays
- Dismantled after £10B spent with limited results

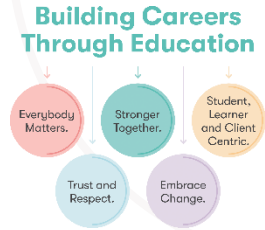


NHS computer issues linked to patient harm: **Image source:** bbc.co.uk



If you don't test with real users and real data, you risk building the wrong thing - at the highest possible cost.

e-learning recap



Reflecting on your learning…

The e-learning for this topic, covered the following areas:

1. Why Testing Matters in Data Engineering
2. Types of Testing: Unit, Integration, System, UAT
3. Automated Testing Tools & Frameworks
4. Designing Effective Test Cases & Coverage
5. Debugging Techniques & Common Pipeline Failures

Q&A discussion

- Do you have any questions about any of these areas?
- Did everything in the e-learning make sense?

BPP

Webinar Agenda

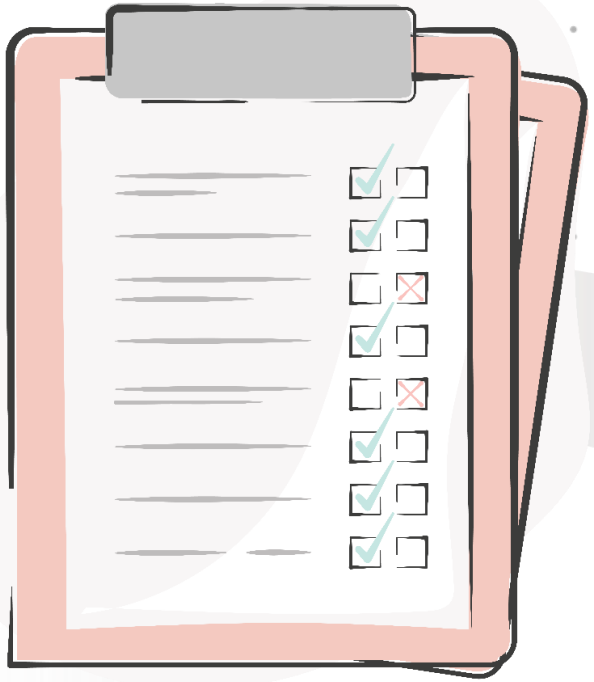
Today, we will cover the following:

Recap

1. Why testing matters in data engineering
2. Types of testing in data pipelines
3. Automated testing frameworks
4. Designing techniques and tools
5. Common pipeline failures

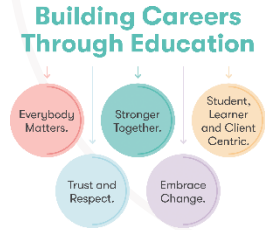
Hands on challenge

1. Hackathon
2. Summary
3. Q&A



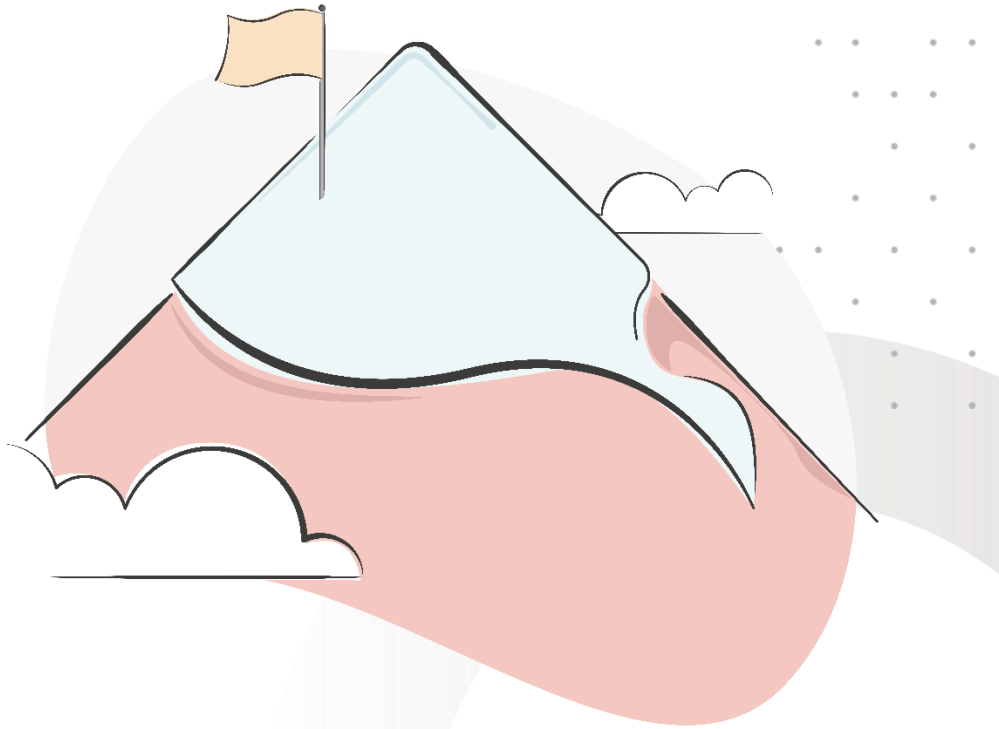
BPP

Session aim and objectives



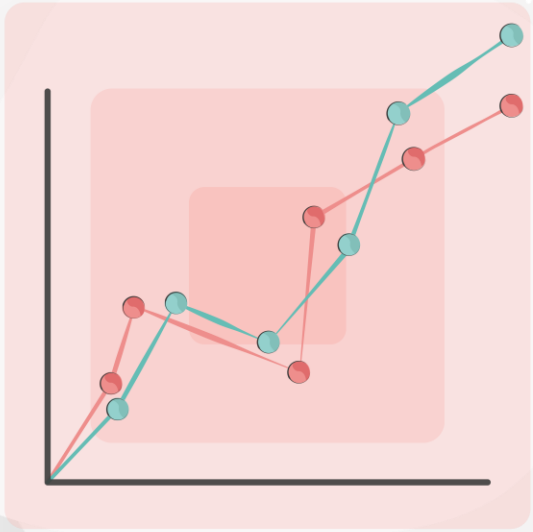
By the end of this session, you should be able to:

- Apply key testing strategies to validate the performance, reliability, and integrity of data pipelines.
- Use automated testing frameworks and debugging tools to identify and resolve common issues.
- Design effective test cases and debug plans that support the delivery of high-quality, maintainable data workflows.



BPP

Core concept recap



Knowledge check poll

Which of the following best describes the purpose of regression testing in a data pipeline?

A. Verifying that each transformation produces expected outputs in isolation

B. Checking if multiple stages of the pipeline work together end-to-end

C. Validating that the pipeline continues to produce correct results after a change

D. Identifying rows that contain null values or missing fields

Feedback: C – Regression testing ensures that recent changes—like new features, bug fixes, or refactors—haven't unintentionally broken existing functionality.

Submit your responses to the chat!

Why testing data pipelines is critical

Small bugs, big consequences

- Data errors can break trust
- Subtle bugs delay reporting
- Testing = reliability + confidence

Q&A

Have you ever had a dashboard show the wrong numbers because of a small logic error?

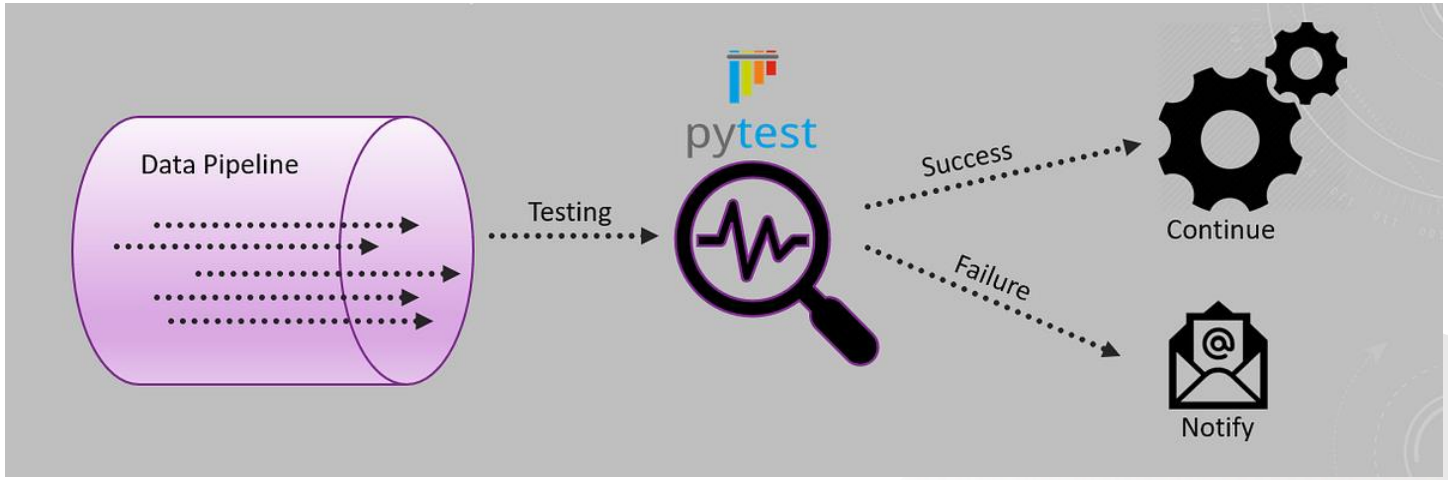


Figure: *Testing isn't just about catching errors - it's about protecting trust,* ***image source:*** *devgenius.io*

Types of testing

Layers of testing in data pipelines

- Unit → Integration → System → UAT
- Each layer builds confidence
- Failures often reveal earlier gaps

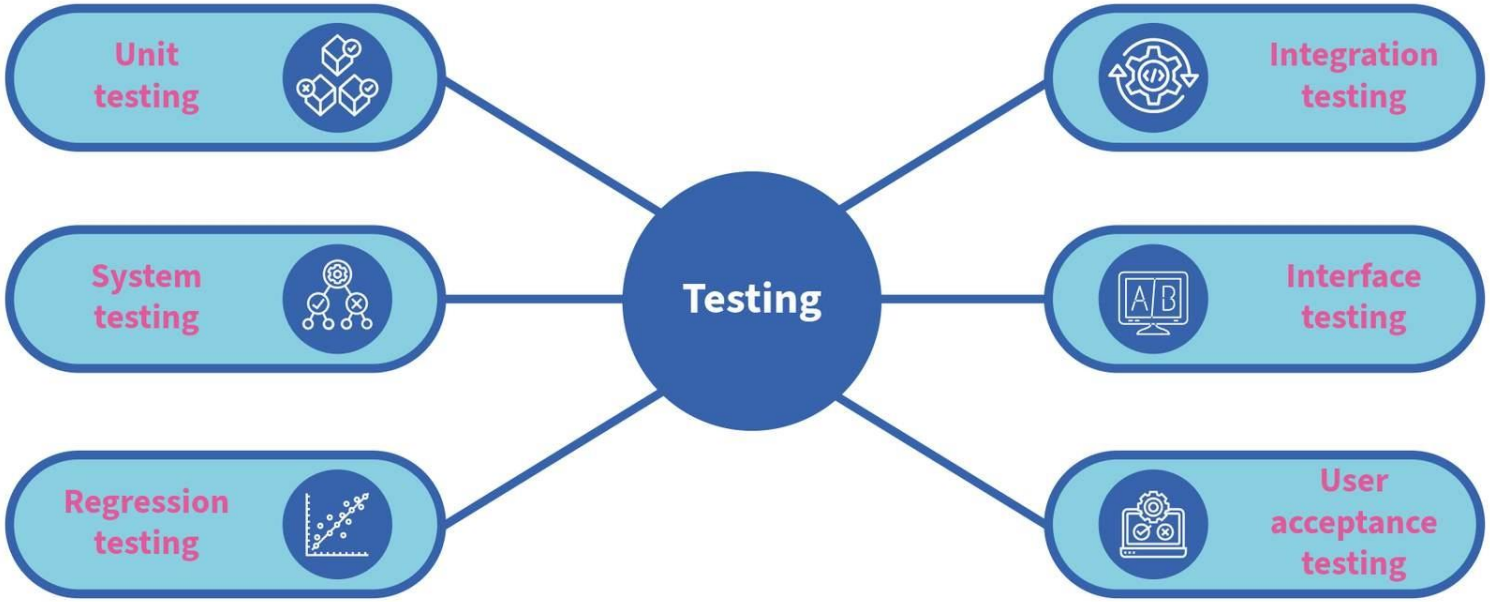


Figure: Different types of testing in data pipelines

Unit & integration testing

Isolated logic & interactions

- Unit: Test small functions with mock data
- Integration: Test modules working together
- Tools: pytest, unittest.mock

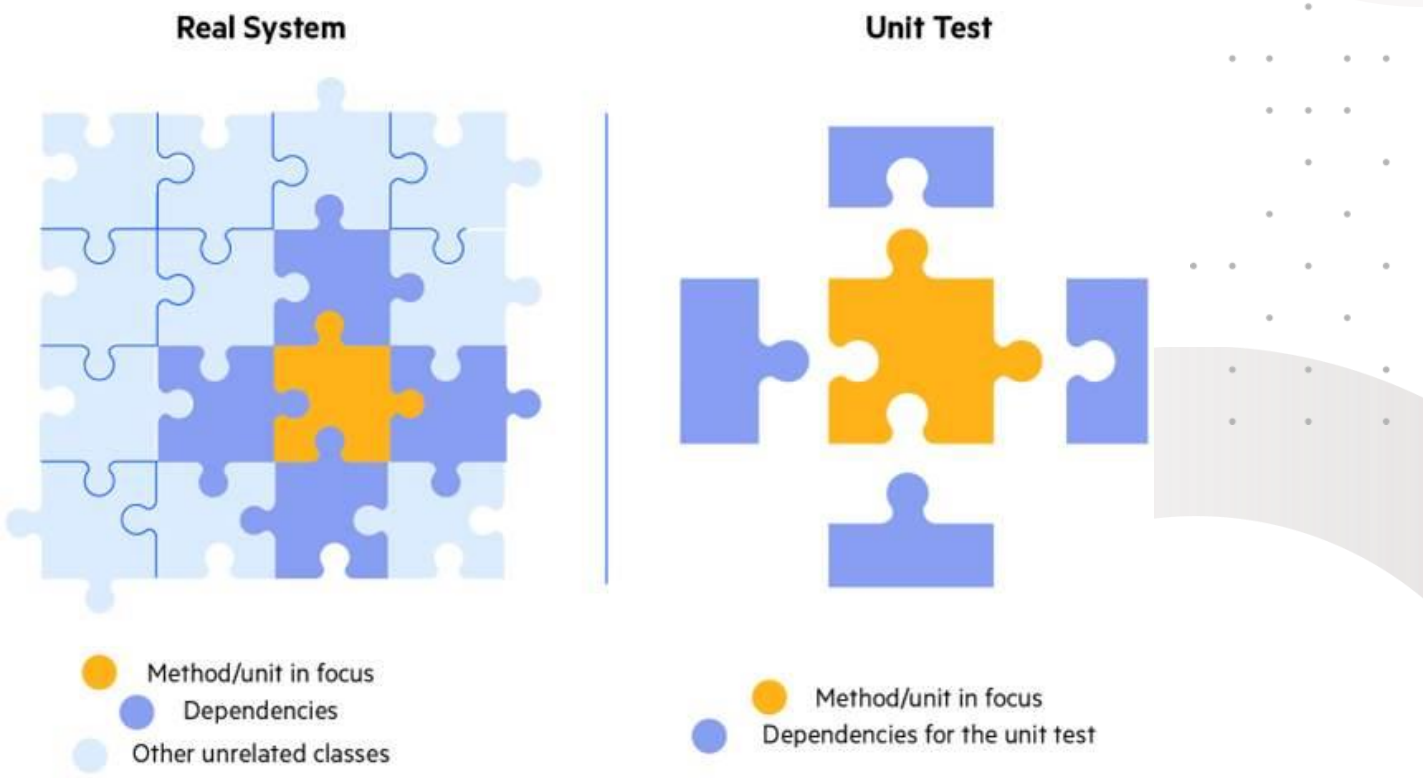


Figure: The whole system versus unit testing

System & UAT

Testing the whole pipeline

- **System:** Validate full pipeline behavior
- **UAT:** Ensure business needs are met
- Stakeholder sign-off is key

Note

Emphasise that UAT is often the last line of defence before production.

Figure: From Unit Testing to Integration Testing to finally System, or Functionality, Testing, image source: *headspin*

Data Validation

Is the data right?

- Schema checks, null checks, uniqueness
- Tools: Great Expectations, Deequ
- Automate validation in pipelines

Note

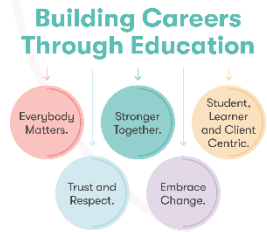
Validation should be automated and run regularly.

```yaml
expectation_suite_name: validate_customer_data
expectations:
  - expectation_type: expect_column_values_to_not_be_null
    kwargs:
      column: email
  - expectation_type: expect_column_values_to_be_between
    kwargs:
      column: age
      min_value: 18
      max_value: 99
  - expectation_type: expect_column_values_to_be_unique
    kwargs:
      column: customer_id
```

Figure: Automated data validation ensures your pipeline's integrity by continuously checking schema, nulls, and uniqueness

Automation & CI

CI/CD for Data Pipelines



- Run tests on every commit
- Tools: GitHub Actions, Jenkins
- Prevent broken code in production

Q&A

How many of you have tests that only run on your laptop?

```yaml
name: CI - Test Pipeline

on: [push]

jobs:
  test:
    runs-on: ubuntu-latest
    steps:
      - uses: actions/checkout@v3
      - name: Run tests
        run: |
          pip install -r requirements.txt
          pytest

```

Figure: Run your tests on every push - automated checks catch issues before they reach production.

BPP

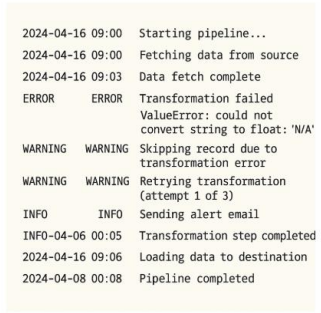
Debugging Techniques

Debugging like a Pro

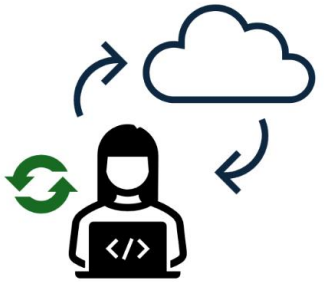
- Start with logs
- Reproduce locally
- Use breakpoint() and Git diffs

Note

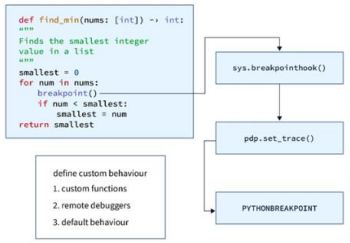
Debugging is a structured process – not guesswork.



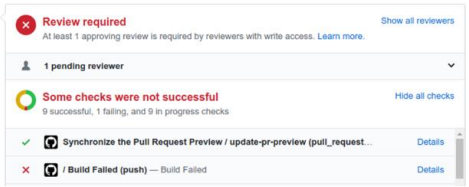
1 - Start with the logs



2 - Reproduce the error locally



3 - Use breakpoints and debuggers

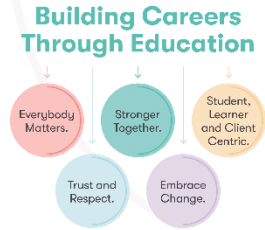


4 - Review recent changes (Version Control)

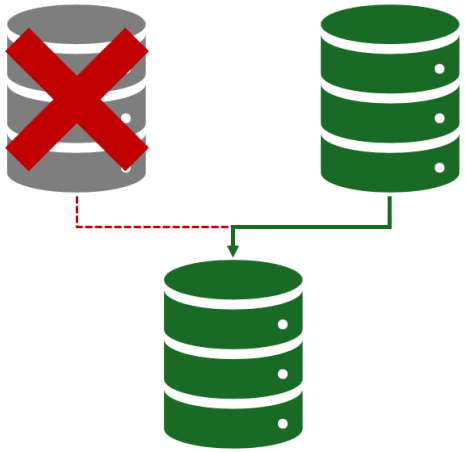
5 - Trace the data

Figure: Debugging is a structured process - start with logs, isolate locally, and trace changes with confidence.

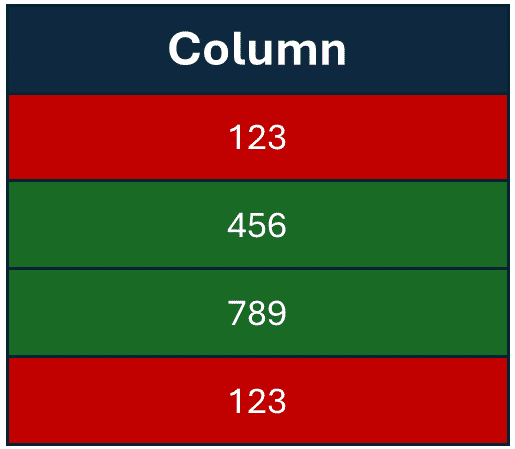
Common pipeline failures



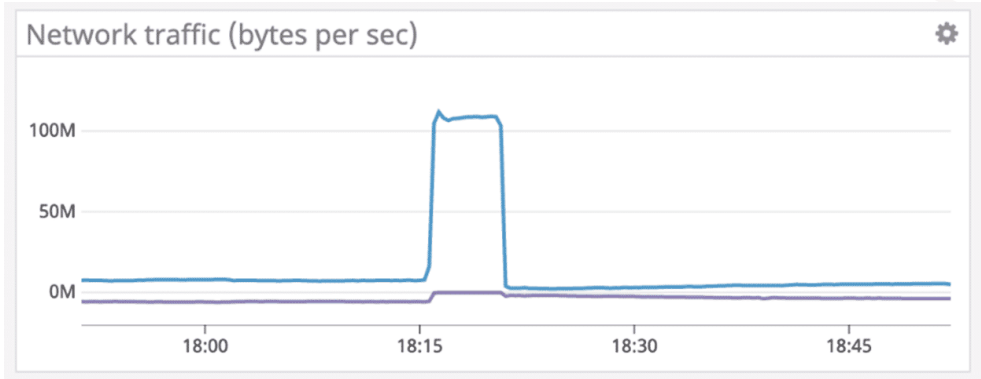
What breaks pipelines?



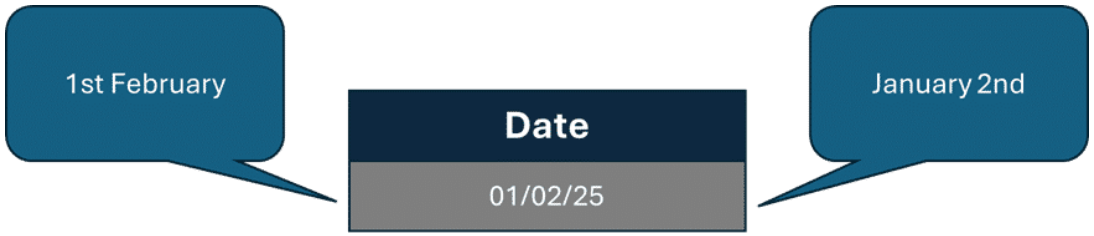
Schema Drift and Source Changes



Nulls, Duplicates and Dirty Data



Data Volume Spikes and Performance Bottlenecks

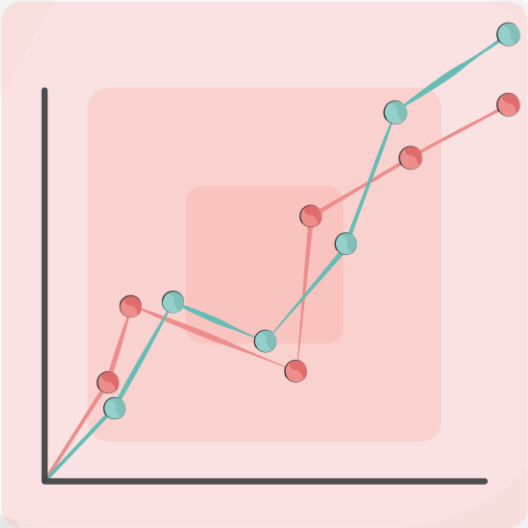


Logic Errors in Tansformations

Downstream Failures and Broken Dependencies

BPP

Practical lab



Hackathon

Part A and B…

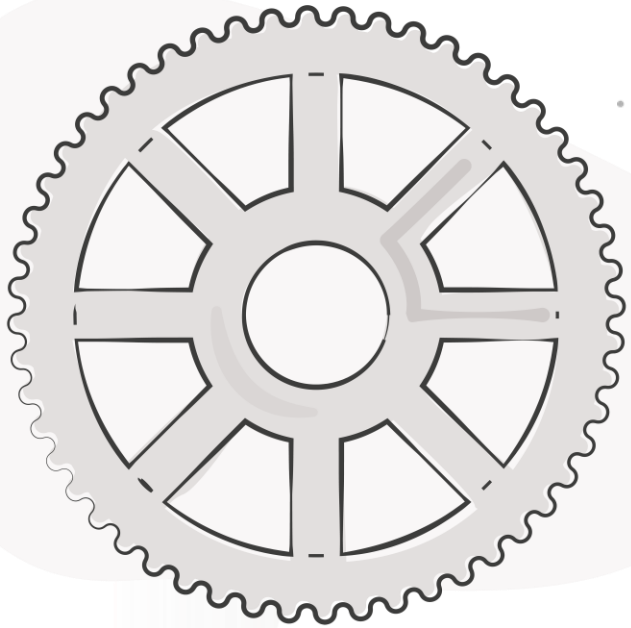
Your step-by-step tasks are as follows:

Part A – Pipeline Design

- Design, implement, and test a data pipeline
- Merge sunrise/sunset and weather data for Edinburgh
- Generate minute-level temperature estimates for 2012

Part B – Pipeline Reuse & Adaptation

- Execute a peer's pipeline on new data (London & Hertfordshire)
- Assess documentation, adapt for schema differences
- Validate output and provide feedback



Practical challenge

Files Provided in the Hub:

- Hackathon brief

BPP

Hackathon extension

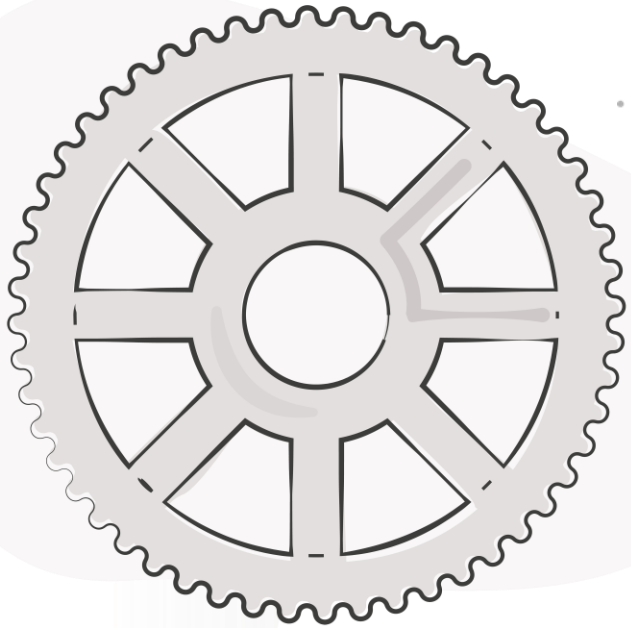
Part C and D…

Part C – Big Data Model Building (Extension)

- Model relationship: temperature ↔ equipment performance
- Link model to customer complaints
- Simulate global warming impact on mast reliability
- Present findings to a non-technical audience
- Bonus: Identify peak usage times (weekday vs weekend)

Part D – Compare Big Data Sets (Extension)

- Apply & adapt model to 8 new masts (Hertfordshire)
- Adjust for local temperature differences
- Identify unusual mast behavior
- Evaluate significance of anomalies
- Present conclusions clearly to a lay audience



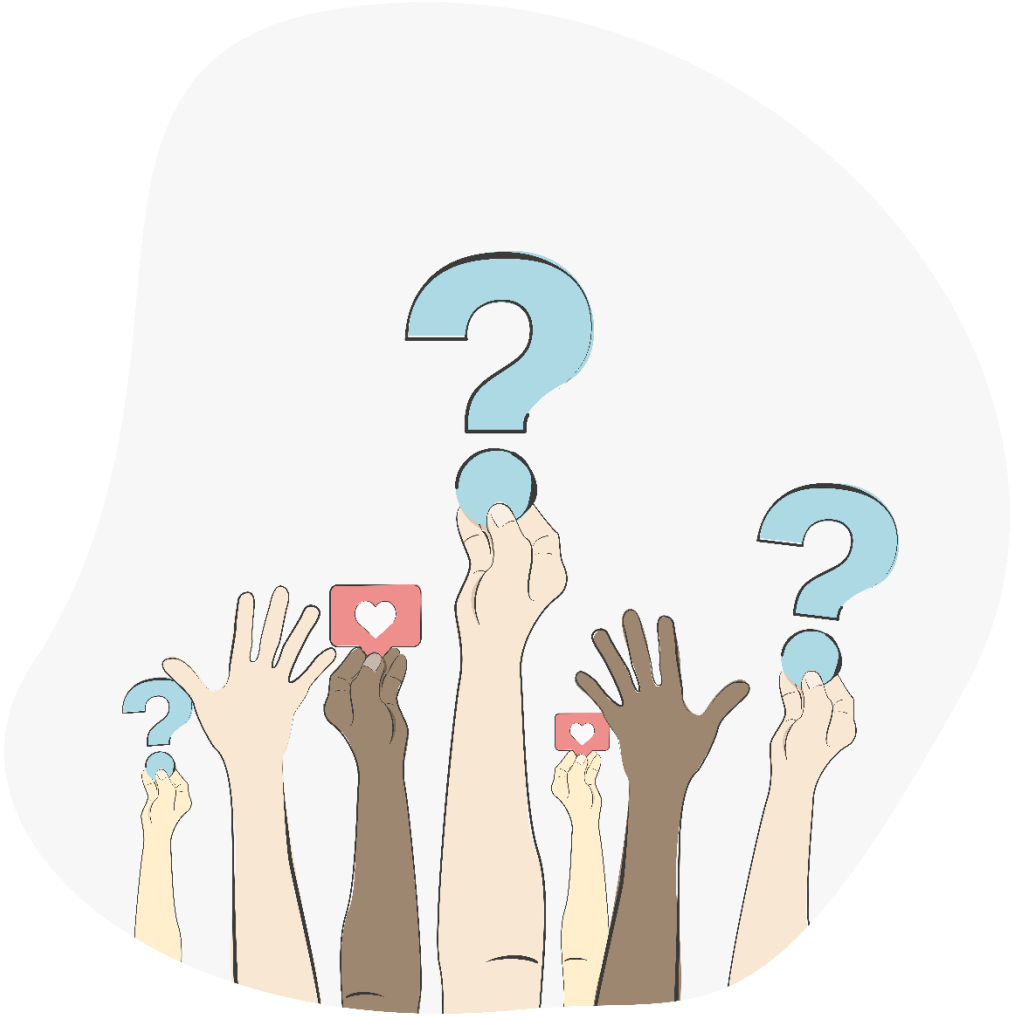
Practical challenge

Files Provided in the Hub:

- Hackathon brief

BPP

Any questions or feedback?



Building Careers Through Education

Everybody Matters. Stronger Together. Student, Learner and Client Centric. Trust and Respect. Embrace Change.

BPP

Thank you