

# Data Quality

And why it's especially important in the  
medallion architecture





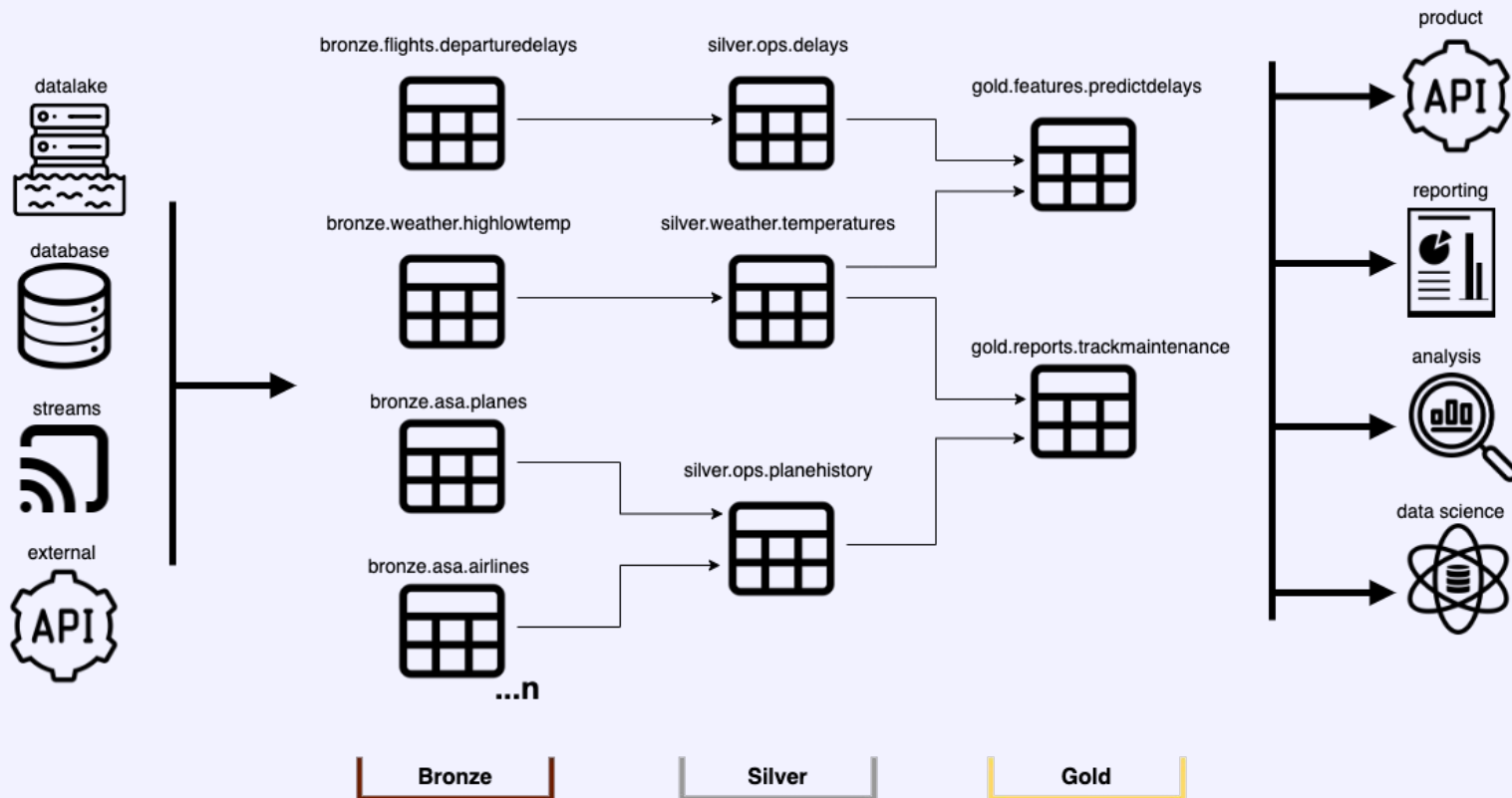
**If International food waste were a country, it would be the third leading cause to GHG emissions behind the US & China**



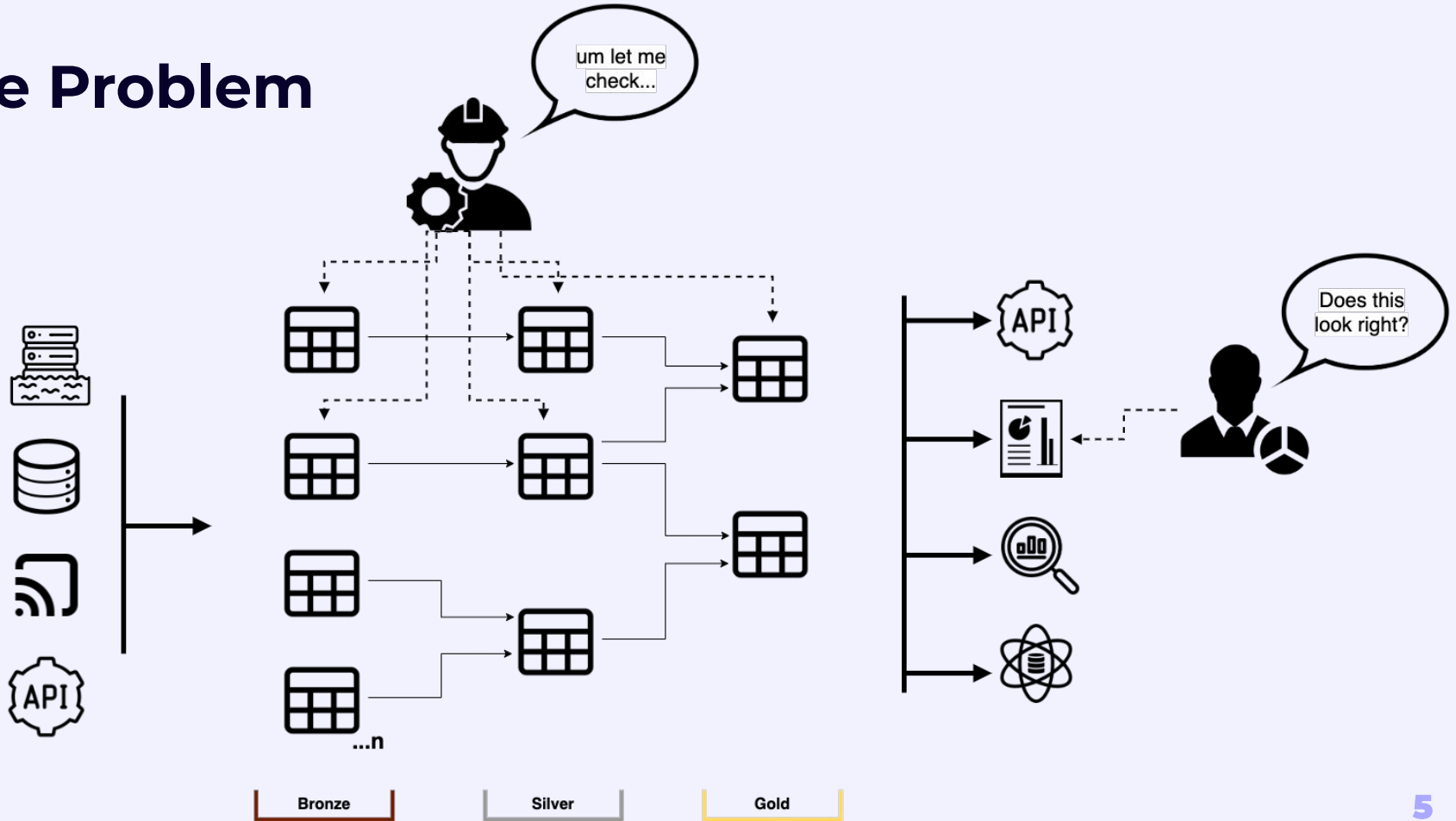
# In Theory



# In Practice



# The Problem



# Motivators

Know of an issue before the end user

Debug the source of the issue

Don't regress as modifications are made

Provide an environment for QA and Analysts to encode business logic



# Demo



# Folder Structure

```
tests/
├── __init__.py
├── conftest.py
├── bronze
│   ├── asa
│   │   ├── airlines_test.py
│   │   └── planes_test.py
│   ├── flights
│   │   └── departedelays_test.py
│   └── weather
│       ├── hightemps_test.py
│       └── lowtemps_test.py
├── gold
│   ├── features
│   │   ├── predictdelays_test.py
│   │   └── predictmaintenance_test.py
│   └── reporting
│       └── planesupply_test.py
└── silver
    ├── ops
    │   ├── delays_test.py
    │   └── planehistory_test.py
    └── weather
        └── temperatures_test.py
```



# conftest.py

```
from pipelines.utils import configmanagement as cm
from datetime import datetime
import platform
import os

def pytest_cmdline_preparse(config, args):

    datetime_now_str = datetime.now().strftime('%Y%m%dT%H%M%S%f')
    html_file = f'tests/reports/{datetime_now_str}-quality-report.html'

    if not cm.isLocal():
        html_file = f'/{html_file}'

    args.extend(['--html', html_file, '--self-contained-html'])

def pytest_unconfigure(config):
    if not cm.isLocal():
        html_report_path = os.path.join(config.invocation_dir.strpath, config.option.htmlpath)
        test_path = config.option.file_or_dir[0].replace("./", "").replace(".py", "").replace("/dbfs/datatest/code/", "")

        file_name = html_report_path.split("/")[-1]
        cm.get_dbutils().fs.cp(f"file:///html_report_path", f"/mnt/{test_path}/{file_name}")
```

# test\_temperatures.py

```
import pytest
import pyspark.sql.functions as f
from datetime import datetime, timedelta
from pyspark.sql import SparkSession
```

✓ Run Test | ✓ Debug Test

```
class Test_temperatures(object):
```

✓ Run Test | ✓ Debug Test

```
    def test_data_is_recent(self):
```

```
        spark = SparkSession.builder.getOrCreate()
```

```
        silver_weather_temperatures_df = spark.read.load("/mnt/data/silver/weather/temperatures")
```

```
        print('most recent 10 records are:')
```

```
        silver_weather_temperatures_df.orderBy(f.col("date").desc()).show()
```

```
        yesterday_date = datetime.utcnow().date() - timedelta(days=1)
```

```
        record_as_of_yesterday_df = silver_weather_temperatures_df.where(f.col("date") >= yesterday_date)
```

```
        yesterday_str = yesterday_date.strftime("%Y/%m/%d")
```

```
        print(f'checking records exist as of {yesterday_str}')
```

```
        assert record_as_of_yesterday_df.count() > 0, "records do not exist recently"
```

# Executing

```
databricks fs cp -r --overwrite tests dbfs:/datatest/code/tests
```

```
databricks fs cp --overwrite dist/datatest-0.0.1-py3-none-any.whl dbfs:/datatest/code
```

```
import pytest
from datetime import datetime

def etl(table_path):
    test_path = f"/dbfs/datatest/code/tests/{table_path}_test.py"

    exit_code = pytest.main([test_path])

    if exit_code.value in (1,2,3,4,5):
        raise Exception("Test(s) failed - check test report")
```

# Execution

Save as template ✓ Validate ▶ Debug ⚙ Trigger (1)

```
graph LR; J1[Python: bronze_weather_highlowtemps] --> J2[Python: silver_weather_temperatures]; J2 --> J3[Python: gold_features_predictions]; J4[Python: bronze_asa_planes] --> J5[Python: silver_ops_planehistory]; J6[Python: bronze_asa_airlines] --> J5; J5 --> J7[Python: gold_features_predict_maintenance];
```

General Azure Databricks Settings User properties

Python file \*

Parameters

+ New | Delete


















☐ PARAMETERS

Append libraries

+ New | Delete

LIBRARY TYPE	LIBRARY CONFIGURATION
wheel	DBFS URI * <input type="text" value="dbfs:/data/test/code/datatest-0.0.1-py3-r"/>
pypi	Package * <input type="text" value="pytest"/>
	Repository <input type="text"/>
pypi	Package * <input type="text" value="pytest-html"/>
	Repository <input type="text"/>

# Result

← → ∨ ↑ Active blobs (default) tests > silver > weather > temperatures_test			
Name	Access Tier	Access Tier Last Modified	Last Modified
 20201119T072417015222-quality-report.html	Hot (inferred)		11/19/2020, 7:24:25 AM
 20201119T120044117414-quality-report.html	Hot (inferred)		11/19/2020, 7:00:44 AM
 20201119T110053265419-quality-report.html	Hot (inferred)		11/19/2020, 6:00:53 AM
 20201119T100051017419-quality-report.html	Hot (inferred)		11/19/2020, 5:00:51 AM
 20201119T090046337048-quality-report.html	Hot (inferred)		11/19/2020, 4:00:46 AM
 20201119T080101903490-quality-report.html	Hot (inferred)		11/19/2020, 3:01:02 AM
 20201119T070104255610-quality-report.html	Hot (inferred)		11/19/2020, 2:01:04 AM
 20201119T060053089895-quality-report.html	Hot (inferred)		11/19/2020, 1:00:53 AM
 20201119T002022278242-quality-report.html	Hot (inferred)		11/19/2020, 12:20:27 AM
 20201119T001858695988-quality-report.html	Hot (inferred)		11/19/2020, 12:19:02 AM
 20201119T001824003556-quality-report.html	Hot (inferred)		11/19/2020, 12:18:47 AM
 20201119T050058086960-quality-report.html	Hot (inferred)		11/19/2020, 12:00:58 AM
 20201119T042539221785-quality-report.html	Hot (inferred)		11/18/2020, 11:25:39 PM
 20201119T012505498176-quality-report.html	Hot (inferred)		11/18/2020, 8:25:05 PM
 20201119T002535144210-quality-report.html	Hot (inferred)		11/18/2020, 7:25:35 PM
 20201119T001845097631-quality-report.html	Hot (inferred)		11/18/2020, 7:18:45 PM
 20201118T235122295774-quality-report.html	Hot (inferred)		11/18/2020, 6:51:22 PM

# Results

## 20201119T072417015222-quality-report.html

Report generated on 19-Nov-2020 at 07:24:24 by [pytest-html](#) v2.1.1

### Environment

Packages	{'pluggy': '0.13.1', 'py': '1.9.0', 'pytest': '6.1.1'}
Platform	Darwin-19.6.0-x86_64-i386-64bit
Plugins	{'html': '2.1.1', 'metadata': '1.10.0'}
Python	3.7.9

### Summary

2 tests ran in 7.85 seconds.

(Un)check the boxes to filter the results.

☒ 1 passed, ☒ 0 skipped, ☒ 1 failed, ☒ 0 errors, ☒ 0 expected failures, ☒ 0 unexpected passes

### Results

[Show all details](#) / [Hide all details](#)

▲ Result	▼ Test	▼ Duration
Failed <a href="#">(show details)</a>	tests/silver/weather/temperatures_test.py::Test_temperatures::test_data_is_recent	7.77
Passed <a href="#">(show details)</a>	tests/silver/weather/temperatures_test.py::Test_temperatures::test_data_in_range	0.00

# Results

Show all details / Hide all details

Result

Test

Failed (hide details)

tests/silver/weather/temperatures\_test.py::Test\_temperatures::test\_data\_is\_recent

```
self = <temperatures_test.Test_temperatures object at 0x7faea0dcd50>

def test_data_is_recent(self):

    spark = SparkSession.builder.getOrCreate()

    silver_weather_temperatures_df = spark.read.load("/mnt/data/silver/weather/temperatures")

    print('most recent 10 records are:')
    silver_weather_temperatures_df.orderBy(f.col("date").desc()).show()

    yesterday_date = datetime.utcnow().date() - timedelta(days=1)
    record_as_of_yesterday_df = silver_weather_temperatures_df.where(f.col("date") >= yesterday_date)

    yesterday_str = yesterday_date.strftime("%Y/%m/%d")
    print(f'checking records exist as of {yesterday_str}')

>     assert record_as_of_yesterday_df.count() > 0, "records do not exist recently"
E     AssertionError: records do not exist recently
E     assert 0 > 0
E       + where 0 = <bound method DataFrame.count of DataFrame[date: string, high_temp: string, low_temp: string, delta_t
E       +       where <bound method DataFrame.count of DataFrame[date: string, high_temp: string, low_temp: string, delta_t
location: string].count

tests/silver/weather/temperatures_test.py:23: AssertionError
-----Captured stdout call-----
most recent 10 records are:
+-----+-----+-----+-----+
|      date|high_temp|low_temp|delta_temp|location|
+-----+-----+-----+-----+
|2018-09-30|    65|    56|      9.0|    SEA|
|2018-09-29|    66|    52|     14.0|    SEA|
|2018-09-28|    79|    57|     22.0|    SEA|
|2018-09-27|    73|    52|     21.0|    SEA|
|2018-09-26|    70|    51|     19.0|    SEA|
|2018-09-25|    69|    50|     19.0|    SEA|
|2018-09-24|    68|    49|     19.0|    SEA|
|2018-09-23|    65|    53|     12.0|    SEA|
|2018-09-22|    68|    55|     13.0|    SEA|
|2018-09-21|    72|    59|     13.0|    SEA|
|2018-09-20|    66|    53|     13.0|    SEA|
|2018-09-19|    68|    48|     20.0|    SEA|
|2018-09-18|    67|    48|     19.0|    SEA|
|2018-09-17|    65|    51|     14.0|    SEA|
|2018-09-16|    65|    51|     14.0|    SEA|
|2018-09-15|    63|    53|     10.0|    SEA|
|2018-09-14|    67|    55|     12.0|    SEA|
|2018-09-13|    70|    55|     15.0|    SEA|
|2018-09-12|    67|    55|     12.0|    SEA|
|2018-09-11|    67|    56|     11.0|    SEA|
+-----+-----+-----+-----+

only showing top 20 rows

checking records exist as of 2020/11/18
```

Passed (show details)

tests/silver/weather/temperatures\_test.py::Test\_temperatures::test\_data\_in\_range

# Results

