# 02 · Test Cleaning Strategies

Use this notebook to prototype different `DataCleanerConfig` combinations before wiring them into automated jobs. Each run reuses the production cleaner so results mirror what would happen in `scripts/data-cleaner.py`.

## Notebook goals

- Quickly compare multiple cleaning strategies without editing production code
- Capture row-retention and missing-value metrics per strategy
- Surface trade-offs (strict vs. relaxed) for stakeholders
- Provide reusable helper functions for future experiments

In [1]:
```
from __future__ import annotations

from pathlib import Path
import importlib.util
import sys
from dataclasses import dataclass

import dask.dataframe as dd
import matplotlib.pyplot as plt
from IPython.display import display
import pandas as pd
import seaborn as sns

plt.style.use("seaborn-v0_8")
sns.set_theme(style="whitegrid")
pd.set_option("display.max_columns", 50)

PROJECT_ROOT = Path.cwd().resolve().parents[1]
DATA_PATH = PROJECT_ROOT / "data" / "national_water_plan.csv"
SCRIPTS_DIR = PROJECT_ROOT / "scripts"


def load_module(module_name: str, file_path: Path):
    if module_name in sys.modules:
        return sys.modules[module_name]
    spec = importlib.util.spec_from_file_location(module_name, file_path)
    module = importlib.util.module_from_spec(spec)
    sys.modules[module_name] = module
    spec.loader.exec_module(module)
    return module


data_loader = load_module("project_data_loader", SCRIPTS_DIR / "data-loader.
DataLoader = data_loader.DataLoader
DataConfig = data_loader.DataConfig
```

```
data_cleaner = load_module("project_data_cleaner", SCRIPTS_DIR / "data-clear
DataCleanerConfig = data_cleaner.DataCleanerConfig
WaterDataCleaner = data_cleaner.WaterDataCleaner
```

```
Pydantic enhancements module not available. Using basic features only.
Pydantic enhancements module not available. Using basic features only.
Pydantic enhancements module not available. Using basic features only.
```

# 1. Load data & build a working sample

The sample keeps experiments fast while still flowing through the exact same cleaning methods.

In [2]:
```
data_config = DataConfig(filepath=str(DATA_PATH))
loader = DataLoader(data_config)
raw_ddf, exploration_report = loader.load_and_explore_data()

SAMPLE_ROWS = 20_000
sample_pdf = raw_ddf.head(SAMPLE_ROWS, compute=True)
print(f"Exploration rows: {exploration_report.metadata.rows:,}")
print(f"Sample size: {len(sample_pdf):,}")
```

```
Duplicate check failed: _sum() got an unexpected keyword argument 'skipna'
Duplicate check failed in statistics: _sum() got an unexpected keyword argum
ent 'skipna'
Duplicate check failed: _sum() got an unexpected keyword argument 'skipna'
Duplicate check skipped (not supported for this operation)
High missing data percentage: 8.10%
Exploration rows: 14,187
Sample size: 14,187
```

In [3]:
```
CleaningReport = data_cleaner.CleaningReport

@dataclass
class StrategyResult:
    name: str
    description: str
    config: DataCleanerConfig
    report: CleaningReport
    cleaned_ddf: "dd.DataFrame"


def run_cleaning_strategy(name: str, description: str, overrides: dict) -> S
    base_kwargs = dict(
        strict_mode=True,
        remove_duplicates=True,
        remove_outliers=True,
        remove_invalid_spill_years=True,
        remove_invalid_text_values=True,
        create_backup=False,
        save_cleaning_report=False,
        output_directory="/tmp",
```

```
    )
    base_kwargs.update(overrides)
    config = DataCleanerConfig(**base_kwargs)
    cleaner = WaterDataCleaner(config)
    cleaned_ddf, report = cleaner.clean_data(sample_pdf.copy(), output_dir=N
    return StrategyResult(name, description, config, report, cleaned_ddf)
```

## 2. Define strategy grid

Tweak just a handful of parameters per strategy so we can reason about the outcome of each change.

In [4]:

```
strategies = [
    {
        "name": "strict_default",
        "description": "Baseline configuration with strict column requiremer
        "overrides": {},
    },
    {
        "name": "relaxed_text",
        "description": "Allow rows that miss optional text fields while keep
        "overrides": {
            "remove_invalid_text_values": False,
            "missing_value_threshold": 0.4,
        },
    },
    {
        "name": "spill_imputation",
        "description": "Fill missing spill values with zeroes and keep rows
        "overrides": {
            "fill_missing_values": True,
            "fill_value": 0,
            "remove_invalid_spill_years": False,
            "min_valid_spill_years": 2,
        },
    },
    {
        "name": "aggressive_outliers",
        "description": "Tighten outlier filtering to 2σ and demand four vali
        "overrides": {
            "outlier_std_threshold": 2.0,
            "min_valid_spill_years": 4,
        },
    },
]

strategies
```

Out[4]:
```
[{'name': 'strict_default',
  'description': 'Baseline configuration with strict column requirements an
d duplicate removal.',
  'overrides': {}},
 {'name': 'relaxed_text',
  'description': 'Allow rows that miss optional text fields while keeping n
umeric validation strict.',
  'overrides': {'remove_invalid_text_values': False,
   'missing_value_threshold': 0.4}},
 {'name': 'spill_imputation',
  'description': 'Fill missing spill values with zeroes and keep rows even
if only two years are available.',
  'overrides': {'fill_missing_values': True,
   'fill_value': 0,
   'remove_invalid_spill_years': False,
   'min_valid_spill_years': 2}},
 {'name': 'aggressive_outliers',
  'description': 'Tighten outlier filtering to 2σ and demand four valid spi
ll years.',
  'overrides': {'outlier_std_threshold': 2.0, 'min_valid_spill_years': 4}}]
```

In [5]:
```
results: list[StrategyResult] = []
for strat in strategies:
    result = run_cleaning_strategy(strat["name"], strat["description"], stra
    results.append(result)
    print(f"✓ {strat['name']} completed — rows retained: {result.report.clea
```

```
2025-11-21 11:52:23,536 - project_data_cleaner - INFO - Starting data cleani
ng pipeline
2025-11-21 11:52:23,537 - project_data_cleaner - INFO - Converting Pandas Da
taFrame to Dask DataFrame
2025-11-21 11:52:23,552 - project_data_cleaner - INFO - Collecting initial s
tatistics
2025-11-21 11:52:23,620 - project_data_cleaner - WARNING - Could not calcula
te initial duplicates
2025-11-21 11:52:23,622 - project_data_cleaner - INFO - Step 1: Validating c
olumns
2025-11-21 11:52:23,624 - project_data_cleaner - INFO - Missing optional col
umns: ['Site Name', 'Permit Number']
2025-11-21 11:52:23,626 - project_data_cleaner - INFO - Step 2: Cleaning coo
rdinates
2025-11-21 11:52:23,697 - project_data_cleaner - INFO - Step 3: Cleaning spi
ll events
2025-11-21 11:52:23,697 - project_data_cleaner - INFO - Cleaning 3 spill eve
nt columns
2025-11-21 11:52:23,844 - project_data_cleaner - WARNING - Removing 233 outl
iers from Spill Events 2020
2025-11-21 11:52:24,112 - project_data_cleaner - WARNING - Removing 219 outl
iers from Spill Events 2021
2025-11-21 11:52:24,472 - project_data_cleaner - WARNING - Removing 238 outl
iers from Spill Events 2022
```

```
2025-11-21 11:52:24,782 - project_data_cleaner - WARNING - Removing 3253 rows with < 3 valid spill years
2025-11-21 11:52:24,785 - project_data_cleaner - INFO - Step 4: Cleaning text fields
2025-11-21 11:52:25,903 - project_data_cleaner - INFO - Step 5: Removing duplicates
2025-11-21 11:52:27,418 - project_data_cleaner - INFO - Step 6: Handling missing values
2025-11-21 11:52:28,124 - project_data_cleaner - WARNING - Removing 1990 rows exceeding missing value threshold
2025-11-21 11:52:28,127 - project_data_cleaner - INFO - Collecting final statistics
2025-11-21 11:52:30,796 - project_data_cleaner - INFO - Cleaning completed: 14187 -> 8944 rows (63.0% retained) in 7.26s
2025-11-21 11:52:30,803 - project_data_cleaner - INFO - Starting data cleaning pipeline
2025-11-21 11:52:30,804 - project_data_cleaner - INFO - Converting Pandas DataFrame to Dask DataFrame
2025-11-21 11:52:30,824 - project_data_cleaner - INFO - Collecting initial statistics
2025-11-21 11:52:30,913 - project_data_cleaner - WARNING - Could not calculate initial duplicates
2025-11-21 11:52:30,916 - project_data_cleaner - INFO - Step 1: Validating columns
2025-11-21 11:52:30,917 - project_data_cleaner - INFO - Missing optional columns: ['Site Name', 'Permit Number']
2025-11-21 11:52:30,918 - project_data_cleaner - INFO - Step 2: Cleaning coordinates
2025-11-21 11:52:30,991 - project_data_cleaner - INFO - Step 3: Cleaning spill events
2025-11-21 11:52:30,992 - project_data_cleaner - INFO - Cleaning 3 spill event columns
✓ strict_default completed — rows retained: 8,944
2025-11-21 11:52:31,112 - project_data_cleaner - WARNING - Removing 233 outliers from Spill Events 2020
2025-11-21 11:52:31,345 - project_data_cleaner - WARNING - Removing 219 outliers from Spill Events 2021
2025-11-21 11:52:31,625 - project_data_cleaner - WARNING - Removing 238 outliers from Spill Events 2022
2025-11-21 11:52:31,830 - project_data_cleaner - WARNING - Removing 3253 rows with < 3 valid spill years
2025-11-21 11:52:31,831 - project_data_cleaner - INFO - Step 4: Cleaning text fields
2025-11-21 11:52:31,836 - project_data_cleaner - INFO - Step 5: Removing duplicates
2025-11-21 11:52:33,123 - project_data_cleaner - INFO - Step 6: Handling missing values
2025-11-21 11:52:34,193 - project_data_cleaner - INFO - Collecting final statistics
```

```
2025-11-21 11:52:37,881 - project_data_cleaner - INFO - Cleaning completed: 14187 -> 10934 rows (77.1% retained) in 7.08s
2025-11-21 11:52:37,901 - project_data_cleaner - INFO - Starting data cleaning pipeline
2025-11-21 11:52:37,902 - project_data_cleaner - INFO - Converting Pandas DataFrame to Dask DataFrame
2025-11-21 11:52:37,931 - project_data_cleaner - INFO - Collecting initial statistics
```

✓ relaxed_text completed — rows retained: 10,934

```
2025-11-21 11:52:38,112 - project_data_cleaner - WARNING - Could not calculate initial duplicates
2025-11-21 11:52:38,115 - project_data_cleaner - INFO - Step 1: Validating columns
2025-11-21 11:52:38,119 - project_data_cleaner - INFO - Missing optional columns: ['Site Name', 'Permit Number']
2025-11-21 11:52:38,121 - project_data_cleaner - INFO - Step 2: Cleaning coordinates
2025-11-21 11:52:38,228 - project_data_cleaner - INFO - Step 3: Cleaning spill events
2025-11-21 11:52:38,230 - project_data_cleaner - INFO - Cleaning 3 spill event columns
2025-11-21 11:52:38,489 - project_data_cleaner - WARNING - Removing 233 outliers from Spill Events 2020
2025-11-21 11:52:38,881 - project_data_cleaner - WARNING - Removing 219 outliers from Spill Events 2021
2025-11-21 11:52:39,331 - project_data_cleaner - WARNING - Removing 238 outliers from Spill Events 2022
2025-11-21 11:52:39,332 - project_data_cleaner - INFO - Step 4: Cleaning text fields
2025-11-21 11:52:40,112 - project_data_cleaner - INFO - Step 5: Removing duplicates
2025-11-21 11:52:41,221 - project_data_cleaner - INFO - Step 6: Handling missing values
2025-11-21 11:52:41,223 - project_data_cleaner - INFO - Filled missing values with: 0
2025-11-21 11:52:41,225 - project_data_cleaner - INFO - Collecting final statistics
2025-11-21 11:52:44,097 - project_data_cleaner - INFO - Cleaning completed: 14187 -> 14187 rows (100.0% retained) in 6.20s
2025-11-21 11:52:44,105 - project_data_cleaner - INFO - Starting data cleaning pipeline
2025-11-21 11:52:44,106 - project_data_cleaner - INFO - Converting Pandas DataFrame to Dask DataFrame
2025-11-21 11:52:44,131 - project_data_cleaner - INFO - Collecting initial statistics
2025-11-21 11:52:44,272 - project_data_cleaner - WARNING - Could not calculate initial duplicates
2025-11-21 11:52:44,279 - project_data_cleaner - INFO - Step 1: Validating columns
```

```
2025-11-21 11:52:44,281 - project_data_cleaner - INFO - Missing optional columns: ['Site Name', 'Permit Number']
2025-11-21 11:52:44,283 - project_data_cleaner - INFO - Step 2: Cleaning coordinates
✓ spill_imputation completed — rows retained: 14,187
2025-11-21 11:52:44,418 - project_data_cleaner - INFO - Step 3: Cleaning spill events
2025-11-21 11:52:44,419 - project_data_cleaner - INFO - Cleaning 3 spill event columns
2025-11-21 11:52:44,535 - project_data_cleaner - WARNING - Removing 577 outliers from Spill Events 2020
2025-11-21 11:52:44,744 - project_data_cleaner - WARNING - Removing 583 outliers from Spill Events 2021
2025-11-21 11:52:45,278 - project_data_cleaner - WARNING - Removing 635 outliers from Spill Events 2022
2025-11-21 11:52:45,535 - project_data_cleaner - WARNING - Removing 14187 rows with < 4 valid spill years
2025-11-21 11:52:45,539 - project_data_cleaner - INFO - Step 4: Cleaning text fields
2025-11-21 11:52:46,212 - project_data_cleaner - INFO - Step 5: Removing duplicates
2025-11-21 11:52:47,372 - project_data_cleaner - INFO - Step 6: Handling missing values
2025-11-21 11:52:47,806 - project_data_cleaner - INFO - Collecting final statistics
2025-11-21 11:52:49,301 - project_data_cleaner - INFO - Cleaning completed: 14187 -> 0 rows (0.0% retained) in 5.20s
✓ aggressive_outliers completed — rows retained: 0
```

## 3. Compare outcomes

Aggregate the metrics that matter most (row retention, missing %, duplicates removed) and visualize trade-offs.

In [6]:
```python
summary_records = []
for result in results:
    metrics = result.report.quality_metrics
    summary_records.append(
        {
            "strategy": result.name,
            "description": result.description,
            "rows_in": result.report.original_shape[0],
            "rows_out": result.report.cleaned_shape[0],
            "rows_retained_pct": metrics.get("rows_retained_percent"),
            "initial_missing_pct": metrics.get("initial_missing_percent"),
            "final_missing_pct": metrics.get("final_missing_percent"),
            "duplicates_removed": metrics.get("duplicate_reduction"),
        }
    )
```

```
summary_df = pd.DataFrame(summary_records)
summary_df
```

Out[6]:

| | strategy | description | rows_in | rows_out | rows_retained_pct | initial_miss |
|---|---|---|---|---|---|---|
| **0** | strict_default | Baseline configuration with strict column requ... | 14187 | 8944 | 63.043631 | 8 |
| **1** | relaxed_text | Allow rows that miss optional text fields whil... | 14187 | 10934 | 77.070558 | 8 |
| **2** | spill_imputation | Fill missing spill values with zeroes and keep... | 14187 | 14187 | 100.000000 | 8 |
| **3** | aggressive_outliers | Tighten outlier filtering to 2σ and demand fou... | 14187 | 0 | 0.000000 | 8 |

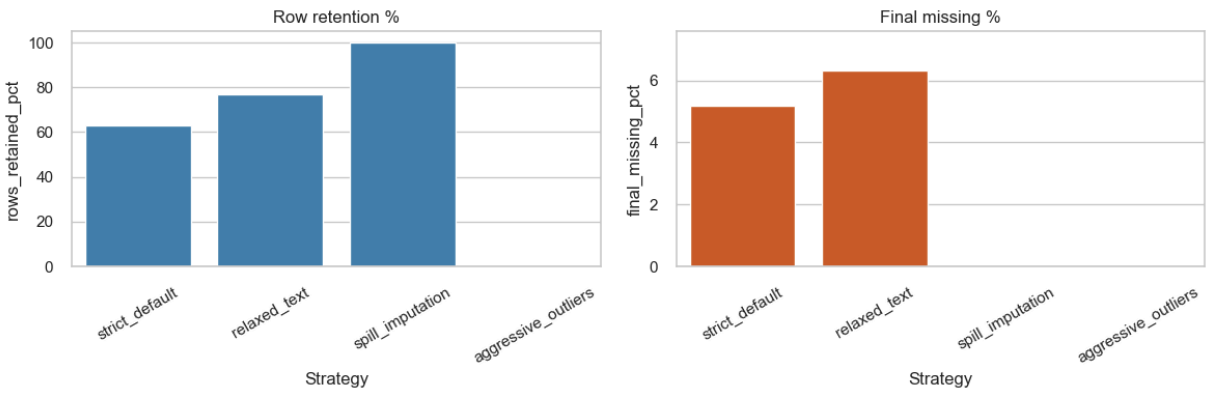
In [7]:

```
fig, axes = plt.subplots(1, 2, figsize=(12, 4))
sns.barplot(data=summary_df, x="strategy", y="rows_retained_pct", ax=axes[0]
axes[0].set_title("Row retention %")
axes[0].set_ylim(0, 105)

sns.barplot(data=summary_df, x="strategy", y="final_missing_pct", ax=axes[1]
axes[1].set_title("Final missing %")
axes[1].set_ylim(0, summary_df["final_missing_pct"].max() * 1.2)

for ax in axes:
    ax.set_xlabel("Strategy")
    ax.tick_params(axis="x", rotation=30)

plt.tight_layout()
plt.show()
```



## Removal breakdown per strategy

Inspect which rules are driving the drop in rows to tune thresholds precisely.

In [8]:
```
removal_records = []
for result in results:
    breakdown = result.report.removal_breakdown or {}
    for reason, count in breakdown.items():
        removal_records.append(
            {
                "strategy": result.name,
                "reason": reason,
                "rows_removed": count,
            }
        )

if removal_records:
    breakdown_df = (
        pd.DataFrame(removal_records)
        .pivot_table(index="reason", columns="strategy", values="rows_remove
        .reindex(columns=summary_df["strategy"], fill_value=0)
        .sort_index(ascending=False)
    )
else:
    breakdown_df = pd.DataFrame()

breakdown_df
```

Out[8]:

| strategy | strict_default | relaxed_text | spill_imputation | aggressive_outlie |
|---|---|---|---|---|
| reason | | | | |
| **insufficient_spill_years** | 3253.0 | 3253.0 | 0 | 1418 |
| **high_missing_values** | 1990.0 | 0.0 | 0 | 0 |

## Peek at cleaned sample for a chosen strategy

Use this helper to inspect the head/tail of the cleaned Dask DataFrame (converted to pandas for convenience).

In [9]:
```
def preview_strategy(strategy_name: str, n: int = 5):
    match = next((r for r in results if r.name == strategy_name), None)
    if match is None:
        raise ValueError(f"Strategy '{strategy_name}' not found. Choose from
    preview_df = match.cleaned_ddf.head(n, compute=True)
    display(preview_df)


preview_strategy("strict_default")
```

| | ID | Water company | Site name | Longitude | Latitude | Receiving Environment | River Basin District |
|---|---|---|---|---|---|---|---|
| **19** | AnW0021 | Anglian Water | ASHTON WATER RECYCLING CENTRE | -0.879539 | 52.132370 | Inland | Anglian |
| **20** | AnW0022 | Anglian Water | ASHWELL WATER RECYCLING CENTRE | -0.171398 | 52.052929 | Inland | Anglian |
| **23** | AnW0025 | Anglian Water | ATTLEBOROUGH WRC | 0.989322 | 52.515652 | Inland | Anglian |
| **24** | AnW0026 | Anglian Water | ATTLEBOROUGH NORWICH ROAD PS | 1.030924 | 52.524269 | Inland | Anglian |
| **40** | AnW0044 | Anglian Water | BARNETBY - FERNERY LA TERMINAL PS | -0.408335 | 53.574852 | Inland | Humber |

## Takeaways

- Use the summary table to document how each config impacts retention vs. quality.
- Feed promising overrides back into `DataCleanerConfig` defaults or environment-specific settings.
- Commit strategy descriptions so future analysts understand *why* thresholds changed.