03 · Compare Before & After Cleaning

This notebook visualizes how the production cleaner transforms the dataset, highlighting wins in data quality and any trade-offs introduced by stricter rules.

Notebook goals

- Quantify improvements in missing data, duplicates, and quality metrics
- Visualize numeric distributions before vs. after cleaning
- Demonstrate the geographic impact of coordinate validation
- Provide an audit trail using `CleaningReport` artifacts

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

from pathlib import Path
import importlib.util
import sys

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="ticks")
pd.set_option("display.max_columns", 40)

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
CleaningReport = data_cleaner.CleaningReport
```

```
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 raw data & create paired samples

We reuse the DataLoader to keep validation consistent, then grab a manageable slice for plotting.

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

SAMPLE_ROWS = 15_000
raw_sample_pdf = raw_ddf.head(SAMPLE_ROWS, compute=True)
print(f"Sample rows: {len(raw_sample_pdf):,} / {exploration_report.metadata.
```

```
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%
Sample rows: 14,187 / 14,187
```

2. Run the cleaner with a representative config

Mirror the production defaults (strict mode, duplicates removal, coordinate checks) but skip filesystem writes for notebook speed.

In [3]:
```
comparison_config = DataCleanerConfig(
    strict_mode=True,
    remove_duplicates=True,
    remove_outliers=True,
    outlier_std_threshold=3.0,
    min_valid_spill_years=3,
    fill_missing_values=False,
    create_backup=False,
    save_cleaning_report=False,
)

cleaner = WaterDataCleaner(comparison_config)
cleaned_ddf, cleaning_report = cleaner.clean_data(raw_sample_pdf.copy(), out
clean_sample_pdf = cleaned_ddf.compute()
print(f"Rows before: {len(raw_sample_pdf):,}")
print(f"Rows after:  {len(clean_sample_pdf):,}")
```

```
2025-11-21 11:55:05,513 - project_data_cleaner - INFO - Starting data cleaning pipeline
2025-11-21 11:55:05,514 - project_data_cleaner - INFO - Converting Pandas DataFrame to Dask DataFrame
2025-11-21 11:55:05,533 - project_data_cleaner - INFO - Collecting initial statistics
2025-11-21 11:55:05,607 - project_data_cleaner - WARNING - Could not calculate initial duplicates
2025-11-21 11:55:05,610 - project_data_cleaner - INFO - Step 1: Validating columns
2025-11-21 11:55:05,611 - project_data_cleaner - INFO - Missing optional columns: ['Permit Number', 'Site Name']
2025-11-21 11:55:05,612 - project_data_cleaner - INFO - Step 2: Cleaning coordinates
2025-11-21 11:55:05,678 - project_data_cleaner - INFO - Step 3: Cleaning spill events
2025-11-21 11:55:05,679 - project_data_cleaner - INFO - Cleaning 3 spill event columns
2025-11-21 11:55:05,800 - project_data_cleaner - WARNING - Removing 233 outliers from Spill Events 2020
2025-11-21 11:55:06,077 - project_data_cleaner - WARNING - Removing 219 outliers from Spill Events 2021
2025-11-21 11:55:06,332 - project_data_cleaner - WARNING - Removing 238 outliers from Spill Events 2022
2025-11-21 11:55:06,508 - project_data_cleaner - WARNING - Removing 3253 rows with < 3 valid spill years
2025-11-21 11:55:06,513 - project_data_cleaner - INFO - Step 4: Cleaning text fields
2025-11-21 11:55:07,362 - project_data_cleaner - INFO - Step 5: Removing duplicates
2025-11-21 11:55:08,710 - project_data_cleaner - INFO - Step 6: Handling missing values
2025-11-21 11:55:09,436 - project_data_cleaner - WARNING - Removing 1990 rows exceeding missing value threshold
2025-11-21 11:55:09,437 - project_data_cleaner - INFO - Collecting final statistics
2025-11-21 11:55:12,215 - project_data_cleaner - INFO - Cleaning completed: 14187 -> 8944 rows (63.0% retained) in 6.70s
Rows before: 14,187
Rows after:  8,944
```

3. Summary metrics

Capture missing %, duplicates, and memory footprint for the paired dataframes.

In [4]:
```python
def summarize_frame(df: pd.DataFrame) -> dict:
    total_cells = df.shape[0] * df.shape[1]
    missing_pct = (df.isna().sum().sum() / total_cells * 100) if total_cells
    duplicate_rows = df.duplicated().sum()
```

```
    memory_mb = df.memory_usage(deep=True).sum() / 1e6
    return {
        "rows": df.shape[0],
        "columns": df.shape[1],
        "missing_pct": missing_pct,
        "duplicate_rows": duplicate_rows,
        "memory_mb": memory_mb,
    }

comparison_summary = pd.DataFrame(
    {
        "raw": summarize_frame(raw_sample_pdf),
        "cleaned": summarize_frame(clean_sample_pdf),
    }
).transpose()
comparison_summary
```

Out[4]:

	rows	columns	missing_pct	duplicate_rows	memory_mb
raw	14187.0	38.0	8.096367	0.0	5.871614
cleaned	8944.0	38.0	5.173418	0.0	3.855667

In [5]:
```
report_metrics = pd.Series(cleaning_report.quality_metrics)
display(report_metrics.to_frame("value"))
```

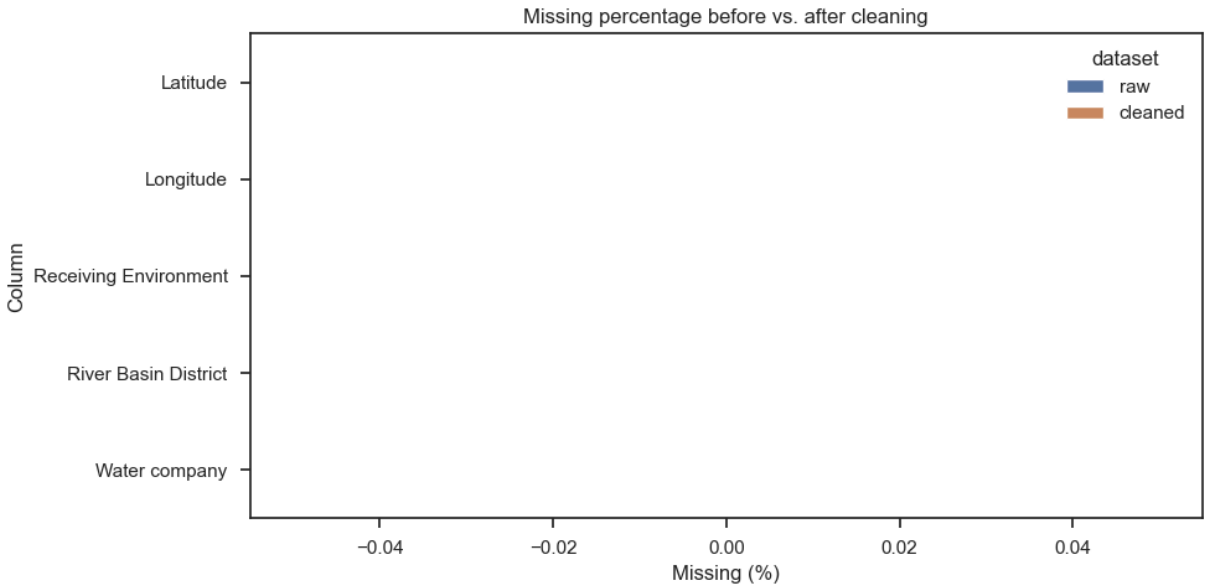
	value
rows_removed	5243.000000
rows_retained_percent	63.043631
initial_missing_percent	8.096367
final_missing_percent	5.173418
duplicate_reduction	0.000000

Missingness per key column

Visualize how missing percentages changed for critical columns from the cleaner's config.

In [6]:
```
key_columns = [
    col for col in sorted(set(comparison_config.required_columns + comparisc
    if col in raw_sample_pdf.columns
]
missing_compare = pd.DataFrame(
    {
        "raw": raw_sample_pdf[key_columns].isna().mean() * 100,
        "cleaned": clean_sample_pdf[key_columns].isna().mean() * 100,
    }
).reset_index().rename(columns={"index": "column"})
```

```
missing_compare = missing_compare.melt(id_vars="column", value_name="missing
fig, ax = plt.subplots(figsize=(10, 5))
sns.barplot(data=missing_compare, x="missing_pct", y="column", hue="dataset"
ax.set_xlabel("Missing (%)")
ax.set_ylabel("Column")
ax.set_title("Missing percentage before vs. after cleaning")
plt.tight_layout()
plt.show()
```

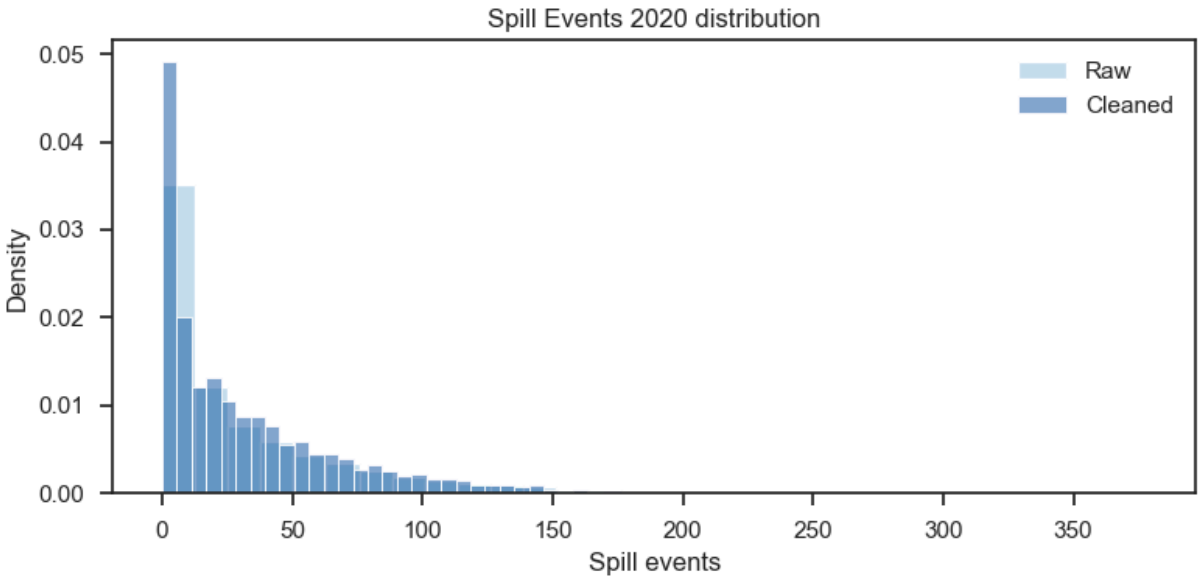


Spill event distribution shift

Pick a representative year to illustrate how outlier removal and missing handling reshape the values.

In [7]:
```
target_spill_col = next((col for col in comparison_config.spill_year_columns
if target_spill_col:
    fig, ax = plt.subplots(figsize=(8, 4))
    sns.histplot(
        raw_sample_pdf[target_spill_col].dropna(),
        bins=30,
        color="#9ecae1",
        label="Raw",
        ax=ax,
        stat="density",
        alpha=0.6,
    )
    sns.histplot(
        clean_sample_pdf[target_spill_col].dropna(),
        bins=30,
        color="#08519c",
        label="Cleaned",
        ax=ax,
        stat="density",
        alpha=0.5,
    )
    ax.set_title(f"{target_spill_col} distribution")
```

```
    ax.set_xlabel("Spill events")
    ax.legend()
    plt.tight_layout()
    plt.show()
else:
    print("No spill event columns found in the sample.")
```



Geographic footprint

Use side-by-side scatter plots to see how invalid coordinates were removed.

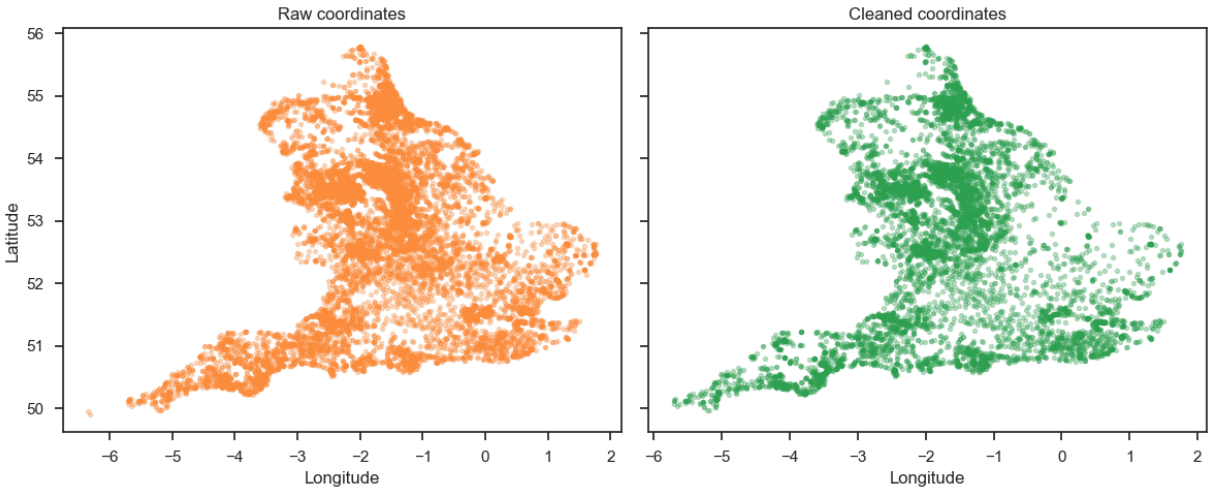
In [8]:
```
coord_cols = [col for col in ["Latitude", "Longitude"] if col in raw_sample_
if len(coord_cols) == 2:
    fig, axes = plt.subplots(1, 2, figsize=(12, 5), sharey=True)
    raw_coords = raw_sample_pdf.dropna(subset=coord_cols)
    clean_coords = clean_sample_pdf.dropna(subset=coord_cols)

    axes[0].scatter(
        raw_coords["Longitude"],
        raw_coords["Latitude"],
        s=8,
        alpha=0.3,
        color="#fd8d3c",
    )
    axes[0].set_title("Raw coordinates")
    axes[0].set_xlabel("Longitude")
    axes[0].set_ylabel("Latitude")

    axes[1].scatter(
        clean_coords["Longitude"],
        clean_coords["Latitude"],
        s=8,
        alpha=0.3,
        color="#31a354",
    )
    axes[1].set_title("Cleaned coordinates")
```

```
    axes[1].set_xlabel("Longitude")

    plt.tight_layout()
    plt.show()
else:
    print("Latitude/Longitude columns are not both available in the sample."
```



Water company coverage

Check whether the cleaning steps disproportionately impact specific companies.

In [9]:
```
if "Water company" in raw_sample_pdf.columns:
    raw_counts = raw_sample_pdf["Water company"].value_counts().head(10)
    clean_counts = clean_sample_pdf["Water company"].value_counts().head(10)

    comparison = (
        pd.concat([raw_counts, clean_counts], axis=1, keys=["raw", "cleaned"
        .fillna(0)
        .astype(int)
    )
    comparison
```

Removal breakdown & audit trail

Leverage the `CleaningReport` to explain exactly why rows were removed.

In [10]:
```
pd.Series(cleaning_report.removal_breakdown).to_frame("rows_removed")
```

Out[10]:

	rows_removed
insufficient_spill_years	3253
high_missing_values	1990

In [11]:
```
pd.DataFrame(
    {
        "errors": cleaning_report.errors or [],
```

```
        "warnings": cleaning_report.warnings or [],
    }
)
```

Out[11]:

errors	warnings

Takeaways

- Pair these visuals with `export/cleaned_data` samples when presenting to stakeholders.
- Removal breakdown + company coverage highlights whether specific regions/companies need bespoke thresholds.
- Use this notebook as a regression harness whenever `WaterDataCleaner` logic changes.