Wczytywanie czyszczenie i przygotowanie danych

Moduł 2 kursu Python LLM i analiza danych w notatnikach Jupyter

Materiał do samodzielnej nauki offline

Autor modułu mgr inż. Marcin Godziemba-Maliszewski

ENHANCER+ Politechnika Warszawska

Jak korzystać z podręcznika

W module 2 przygotujesz zbiór danych, którego jakość potrafisz opisać i sprawdzić. Wrócisz do przypadku „Eksploatacja urządzeń” z modułu 1, ale tym razem obejmiesz kontrolą wszystkie pomiary i kartotekę urządzeń. Rezultatem będzie nie tylko plik do dalszej analizy, lecz także zapis decyzji wyjaśniający, co zmieniono i jakie ograniczenia pozostały.

Czytaj rozdziały kolejno. Najpierw poznasz regułę i przykład, następnie zobaczysz kod oraz interpretację wyniku. Pytania „Sprawdź rozumienie” możesz rozwiązać bez komputera; odpowiedzi znajdują się na końcu książki. Ćwiczenia w notebooku pomagają zastosować tę samą regułę do danych. Nie musisz zapamiętywać wszystkich poleceń — ważne, aby umieć uzasadnić operację i sprawdzić jej skutek.

Do czytania wystarczy ten dokument. Do uruchamiania przykładów offline przygotuj wcześniej środowisko z modułu 1, biblioteki Pandas, NumPy, Matplotlib i openpyxl oraz lokalny pakiet danych. Korzystanie z modelu językowego jest opcjonalne; żaden krok obliczeniowy nie wymaga połączenia z usługą LLM.

Spis treści

Rozdział 1 Bezpieczne wczytywanie CSV i XLSX

Rozdział 2 Pierwszy audyt zbioru

Rozdział 3 Braki duplikaty i identyfikatory

Rozdział 4 Typy daty kategorie i jednostki

Rozdział 5 Zakresy dopuszczalne i wartości odstające

Rozdział 6 Transformacje łączenie i walidacja wyniku

Podsumowanie i odpowiedzi do samooceny

Słownik pojęć i pól wynikowych

Materiały modułu

Marcin Godziemba-Maliszewski, „Moduł 2 Wczytywanie, czyszczenie i przygotowanie danych”, prezentacja M2-v2.pptx. Materiały towarzyszące: karta M2_Instrukcja_przygotowanie_danych.docx oraz pakiet startowy uczestnika z pomiarami, kartoteką urządzeń, przykładami kontrolowanymi i słownikiem slownik_danych.md.

Punkt startowy i rezultat pracy

Jeden wiersz głównej tabeli opisuje pomiary jednego urządzenia w określonym czasie. Zbiór obejmuje 720 rekordów, osiem urządzeń i 90 znaczników czasu od 1 stycznia do 14 lutego 2026 roku. Kartoteka zawiera po jednym wierszu dla każdego urządzenia. Różnica między jednostką obserwacji obu tabel będzie ważna przy ich łączeniu.

Wykorzystujesz dane ćwiczeniowe, a nie uniwersalne normy eksploatacyjne. Podane zakresy i progi obowiązują w tym przypadku dydaktycznym. Przy pracy z innym źródłem należy ustalić jego własny kontrakt danych: oczekiwaną strukturę, jednostki, dopuszczalne wartości oraz reguły kompletności i unikalności.

Z folderu Dane w pakiecie startowym skopiuj trzy pliki do data/raw: pomiary_eksploatacyjne.csv, urzadzenia.xlsx i przyklady_kontrolowane_m2.csv. Słownik slownik_danych.md umieść w docs. Plik demonstracyjny zawiera celowo wprowadzone problemy i służy do osobnych eksperymentów. Nie dołączaj jego rekordów do głównej tabeli.

Zachowaj notebook M1_szkielet_analizy.ipynb. Jego kopię roboczą zapisz jako M2_przygotowanie_danych.ipynb w notebooks. Zastąp część obliczeniową przebiegiem M2: wczytaj od początku cały zbiór, a nie podzbiór jednego urządzenia utworzony w M1. Główne fragmenty kodu w tej książce wykonuj kolejno w jednej sesji; bloki opisane jako eksperyment pracują na odrębnych zmiennych.

Element projektu	Stała lokalizacja
Notebook modułu	notebooks/M2_przygotowanie_danych.ipynb
Wynik dla modułu 3	data/processed/pomiary_clean.csv
Porównanie jakości	outputs/tables/M2_audyt_jakosci.csv
Znaczenie i schemat pól	docs/slownik_danych.md
Historia pomocy modelu	docs/rejestr_uzycia_LLM.csv
Sposób odtworzenia pracy	README.md

Stosuj konsekwentnie katalog outputs, zgodnie z kartą pracy M2. Jeżeli w starszym materiale występuje output/tabele, jest to miejsce o tej samej roli; w tym podręczniku odpowiada mu outputs/tables. Prefiks M2 identyfikuje notebook i raport modułu. Nazwa pomiary_clean.csv pozostaje wspólna dla kolejnych analiz.

Rozdział 1 Bezpieczne wczytywanie CSV i XLSX

Import jest interpretacją pliku, a nie potwierdzeniem jakości danych. Program może wykonać polecenie bez wyjątku i mimo to zwrócić niewłaściwą tabelę. Dlatego najpierw potwierdzasz źródło, potem parametry odczytu, a na końcu strukturę i próbkę wyniku. Dopiero wszystkie te kontrole pozwalają przyjąć import.

Ścieżka i ochrona źródła

Ścieżki buduj względem katalogu projektu. W poniższym kodzie zakładamy, że bieżącym katalogiem roboczym jest notebooks. Samo umieszczenie tam notebooka nie zawsze wystarcza — sprawdź wynik Path.cwd(). Gdy kontrola zatrzyma wykonanie, popraw katalog roboczy lub jawnie wskaż właściwy katalog projektu, zamiast zgadywać inną nazwę pliku.

```
from pathlib import Path
import pandas as pd
import numpy as np

print("Katalog roboczy:", Path.cwd())
assert Path.cwd().name == "notebooks"
PROJECT_DIR = Path.cwd().parent
DATA_RAW = PROJECT_DIR / "data" / "raw"
plik_pomiary = DATA_RAW / "pomiary_eksploatacyjne.csv"
plik_urzadzenia = DATA_RAW / "urzadzenia.xlsx"
assert plik_pomiary.is_file()
assert plik_urzadzenia.is_file()
```

Pliki w data/raw traktuj jako niezmienny punkt odniesienia. Nie poprawiaj ich ręcznie w Excelu i nie zapisuj pod ich nazwami wyników czyszczenia. Kod będzie przekształcać ramki danych w pamięci, a rezultat zapiszesz w data/processed. Dzięki temu błędną decyzję można odrzucić i ponownie rozpocząć pracę od źródła.

Co określają parametry CSV

CSV jest plikiem tekstowym. Parametr sep wskazuje separator kolumn, encoding określa kodowanie znaków, a header — wiersz z nazwami pól. Przy jednym formacie liczb parametr decimal może wskazywać przecinek dziesiętny. Z kolei na_values pozwala dopisać oznaczenia braków, na przykład N/D. Te ustawienia powinny wynikać z formatu źródła, nie z prób wykonywanych aż do zniknięcia komunikatu błędu.

Zbyt szeroka lista na_values może zamienić poprawny tekst na brak. Błędne kodowanie może uszkodzić polskie litery bez przerwania odczytu. Dlatego nawet po udanym imporcie obejrzyj kilka wartości tekstowych i porównaj je z zapisem źródłowym.

Odczyt pomiarów i kontrola struktury

Główny plik pomiarów wykorzystuje przecinek jako separator i kodowanie UTF-8. Puste pola energii są rozpoznawane przez Pandas jako braki. Przyrostek raw w nazwie zmiennej przypomina, że na tym etapie nie poprawiasz zawartości.

```
pomiary_raw = pd.read_csv(
    plik_pomiary, sep=",", encoding="utf-8", header=0
)
OCZEKIWANE = [
    "czas", "urzadzenie", "temperatura", "drgania",
    "energia", "produkcja", "tryb", "status"
]
assert pomiary_raw.shape == (720, 8)
assert pomiary_raw.columns.tolist() == OCZEKIWANE
display(pomiary_raw.head(3))
```

Wynik powinien mieć 720 wierszy i osiem osobnych kolumn. W próbce sprawdź, czy czas trafił do czasu, identyfikator do urządzenia, a wartości pomiarów do właściwych pól. Rozmiar i nazwy są kontrolowane automatycznie; odczyt próbki uzupełnia je o ocenę sensu struktury. Trzy poprawne wiersze nie dowodzą jednak poprawności pozostałych 717.

Eksperyment z separatorem wykonaj w odrębnej zmiennej. Nie zastępuj poprawnego importu wynikiem tej próby.

```
pomiary_bledny = pd.read_csv(plik_pomiary, sep=";")
print(pomiary_bledny.shape)
print(pomiary_bledny.columns.tolist())
```

Otrzymasz 720 × 1. Cały nagłówek zostanie jedną długą nazwą kolumny, a rekordy pozostaną nierozdzielonym tekstem. Parser postąpił zgodnie z parametrem, ale parametr nie odpowiadał źródłu. Taki import odrzucasz, mimo że nie wystąpił wyjątek.

Nagłówek i komunikaty diagnostyczne

Ustawienie header=None oznacza brak wiersza nazw. W pliku, który ma nagłówek, dotychczasowe nazwy staną się wtedy pierwszym rekordem danych. Jeżeli pojawi się FileNotFoundError, zacznij od ścieżki i is_file(). Przy UnicodeDecodeError sprawdź kodowanie źródła. Jedna kolumna zamiast ośmiu kieruje uwagę na separator. Każdy sygnał wskazuje kontrolę do wykonania, ale sam nie stanowi gotowej diagnozy.

Skoroszyt wymaga wyboru arkusza

XLSX może zawierać kilka arkuszy o różnych rolach. W pliku urzadzenia.xlsx pierwszy arkusz nazywa się opis, a dopiero drugi zawiera kartotekę. Domyślne read_excel() odczytuje pierwszy arkusz: w tym przypadku dałoby tabelę 16 × 2, poprawnie odczytaną technicznie, lecz nieodpowiednią do łączenia z pomiarami.

```
with pd.ExcelFile(plik_urzadzenia) as xls:
    print(xls.sheet_names)
urzadzenia_raw = pd.read_excel(
    plik_urzadzenia, sheet_name="urzadzenia"
)
META_RAW = [
    "urzadzenie", "typ", "lokalizacja",
    "moc_nominalna_kw", "data_instalacji"
]
assert urzadzenia_raw.shape == (8, 5)
assert urzadzenia_raw.columns.tolist() == META_RAW
display(urzadzenia_raw.head(3))
```

Wynik zawiera osiem urządzeń i pięć pól. Nazwa arkusza jest częścią kontraktu importu. W innych skoroszytach potrzebne mogą być także header i usecols, jeżeli tabelę poprzedza opis albo chcesz ograniczyć odczyt do określonych kolumn. Błąd dotyczący openpyxl wskazuje na brak biblioteki w używanym środowisku, nie na potrzebę zmiany danych.

Decyzja po imporcie

W notebooku dodaj komórkę Markdown „Decyzja po imporcie”. Wskaż nazwę i lokalizację pliku, użyte parametry, rozmiar wyniku, zgodność nazw kolumn oraz wniosek z próbki. Dla CSV zapisz, że import z przecinkiem przyjęto, a wariant ze średnikiem odrzucono. Dla XLSX potwierdź jawny wybór arkusza urzadzenia.

Przyjęcie importu oznacza, że masz właściwie odczytaną tabelę. Nie oznacza, że jej daty, kategorie i wartości są już gotowe do analizy. To rozróżnienie pozwala później ustalić, czy problem powstał podczas interpretacji formatu, czy był obecny w samej zawartości.

Ćwiczenie w notebooku

Porównaj poprawny i błędny separator CSV oraz domyślny i jawny wybór arkusza XLSX. Dla każdego wariantu zapisz trzy dowody: rozmiar, nazwy kolumn i próbkę. Sprawdź, czy twoje uzasadnienie pozwala innej osobie odtworzyć decyzję bez oglądania prezentacji.

Sprawdź rozumienie 1

Plik zwrócił oczekiwane 720 rekordów, ale tylko jedną kolumnę. Czy możesz rozpocząć audyt wartości? Uzasadnij odpowiedź i wskaż pierwszą kontrolę, którą wykonasz.

Rozdział 2 Pierwszy audyt zbioru

Audyt opisuje stan danych przed zmianami. Bez tego punktu odniesienia łatwo uznać dowolny efekt czyszczenia za poprawę: mniej wierszy za usunięcie problemów, a brak pustych komórek za lepszą jakość. Tymczasem oba efekty mogą wynikać z utraty informacji. Na tym etapie zbierasz dowody i zapisujesz pytania, nie poprawiasz jeszcze wartości.

Profil jakości kolumn

Metoda info() łączy informację o strukturze, liczbie niepustych wartości i typach. Dtypes pozwala odczytać same typy, isna() wskazuje braki, a nunique() liczy różne wartości. Ustawienie dropna=False uwzględnia brak jako dodatkową odrębną wartość. Taki profil można później odtworzyć i porównać ze stanem końcowym.

```
pomiary_raw.info()
profil_jakosci = pd.DataFrame({
    "typ": pomiary_raw.dtypes.astype(str),
    "braki": pomiary_raw.isna().sum(),
    "braki_proc": pomiary_raw.isna().mean().mul(100),
    "unikalne_z_brakiem": pomiary_raw.nunique(dropna=False)
}).rename_axis("pole")
display(profil_jakosci.round(2))
display(pomiary_raw.describe().T)
```

W energii są 718 istniejących wartości i dwa braki, czyli około 0,28% z 720 rekordów. Pozostałe kolumny są na wejściu kompletne. Czas może zostać odczytany jako object lub typ tekstowy — zależy to od środowiska i wersji biblioteki. Nadal wymaga jawnej konwersji na datę i czas. Typ pokazany przez program nie zastępuje znaczenia pola w słowniku.

W produkcji typ całkowity nie wyklucza wartości ujemnej. Podobnie poprawny float64 nie potwierdza właściwej jednostki. Audyt musi więc badać zarówno reprezentację, jak i znaczenie danych. Liczba braków, typ i zakres to różne, uzupełniające się informacje.

Kategorie i unikalność

```
KLUCZ = ["czas", "urzadzenie"]
print("Pełne duplikaty:", pomiary_raw.duplicated().sum())
print("Powtórzenia klucza:",
      pomiary_raw.duplicated(subset=KLUCZ).sum())
for pole in ["tryb", "status"]:
    display(pomiary_raw[pole].value_counts(dropna=False))
```

Od profilu do listy sygnałów

Nie ma pełnych duplikatów ani powtórzeń pary czas i urzadzenie. W trybie występują jednak trzy zapisy: oszczedny 360 razy, auto 359 razy i AUTO jeden raz. Status występuje jako OK w 719 rekordach i ok w jednym. To sygnał niespójnego zapisu; wspólne znaczenie wariantów potwierdza słownik, a nie sama podobna pisownia.

Pole	Obserwacja	Wniosek audytu
energia	2 braki	Ustalić strategię bez wymyślania pomiarów
czas	Zapis tekstowy	Sprawdzić daty i wykonać konwersję
tryb i status	Warianty AUTO i ok	Ujednolicić po potwierdzeniu znaczenia
produkcja	Minimum −12 szt.	Naruszenie reguły nieujemnej liczby sztuk
drgania	Maksimum 4,850 mm/s	Sygnał diagnostyczny wymagający kontekstu

Minimum i maksimum pomagają znaleźć problemy, ale nie opisują całego rozkładu. Temperatura mieści się w przedziale 52,70–66,25 °C, energia wśród istniejących pomiarów w 11,48–15,89 kWh, a produkcja w −12–173 szt. Każdy zakres porównujesz z regułą ustaloną wcześniej, a nie z granicami wymyślonymi po obejrzeniu danych.

Czas i pokrycie obserwacji

Do diagnozy czasu utwórz serię tymczasową. Maska błędów rozróżnia nowy brak po konwersji od pustej wartości obecnej wcześniej. Samo policzenie wszystkich NaT nie daje tego rozróżnienia.

```
czas_tmp = pd.to_datetime(
    pomiary_raw["czas"], format="%Y-%m-%d %H:%M",
    errors="coerce"
)
blad_daty = czas_tmp.isna() & pomiary_raw["czas"].notna()
assert not blad_daty.any()
print(czas_tmp.min(), czas_tmp.max(), czas_tmp.nunique())
na_czas = pomiary_raw.groupby("czas")["urzadzenie"].nunique()
assert na_czas.eq(8).all()
assert pomiary_raw["urzadzenie"].value_counts().eq(90).all()
```

Zakres obejmuje 45 dni kalendarzowych, z dwoma pomiarami dziennie dla każdego urządzenia. Każdy z 90 znaczników ma osiem urządzeń. Dopiero w połączeniu z kontrolą kompletności i unikalności klucza te liczby potwierdzają oczekiwaną siatkę obserwacji.

Audyt kartoteki i relacji między źródłami

Kartoteka powinna zawierać jeden kompletny rekord urządzenia. Unikalność identyfikatora ma tutaj inne znaczenie niż w pomiarach: powtarzanie U-03 w pomiarach jest oczekiwane, ale dwa wiersze U-03 w kartotece mogłyby zwielokrotnić dane przy łączeniu.

```
assert urzadzenia_raw.notna().all().all()
assert urzadzenia_raw["urzadzenie"].is_unique
ids_pomiary = set(pomiary_raw["urzadzenie"])
ids_kartoteka = set(urzadzenia_raw["urzadzenie"])
print("Bez kartoteki:", ids_pomiary - ids_kartoteka)
print("Bez pomiarów:", ids_kartoteka - ids_pomiary)
```

W obu kierunkach otrzymasz pusty zbiór. Wszystkie osiem identyfikatorów występuje w obu źródłach. Różnice zbiorów odpowiadają jednak na dwa odrębne pytania: urządzenie bez kartoteki oznacza brak metadanych pomiaru, a urządzenie bez pomiarów może po prostu nie brać udziału w danym okresie obserwacji. Nie należy tych sytuacji utożsamiać.

Wnioski po audycie

Dobry zapis oddziela potwierdzone własności od spraw wymagających decyzji. Możesz napisać: „Oba źródła mają oczekiwaną strukturę. Klucz pomiarów jest unikalny, kartoteka kompletna, a identyfikatory zgodne. Do rozpatrzenia pozostają dwa braki energii, format czasu, dwa warianty kategorii, ujemna produkcja i nietypowo wysokie drgania. Audyt nie zmienił danych”.

Taki zapis jest mocniejszy niż stwierdzenie „dane wyglądają dobrze”. Wskazuje zakres wykonanej kontroli i jej granice. Nie przypisuje przyczyny brakowi energii i nie nazywa skoku drgań awarią, ponieważ sam profil danych tego nie dowodzi.

Ćwiczenie w notebooku

Zbuduj profil jakości obu źródeł, wyświetl dwa rekordy z brakami energii i dopisz do każdego sygnału właściwe pytanie. Dla braków pytaj o możliwość odtworzenia pomiaru; dla kategorii o znaczenie; dla ujemnej produkcji o regułę; dla drgań o kontekst. Zachowaj wyniki jako stan „przed”.

Sprawdź rozumienie 2

W próbce pierwszych trzech wierszy nie ma braków, a typ wszystkich pomiarów jest liczbowy. Czy to wystarcza, aby uznać cały zbiór za kompletny i poprawny? Wskaż dwie kontrole, które nadal są potrzebne.

Rozdział 3 Braki duplikaty i identyfikatory

Decyzję o czyszczeniu podejmujesz na podstawie znaczenia danych i celu analizy. Ten sam brak może być dopuszczalny w jednym zastosowaniu i uniemożliwiać inne. Podobne rekordy mogą być kopiami albo różnymi zdarzeniami. Każdą decyzję opisuj przez dowód, uzasadnienie, operację i kontrolę skutku.

Kopia robocza i świadome pozostawienie braków

```
pomiary_clean = pomiary_raw.copy()
urzadzenia_clean = urzadzenia_raw.copy()
braki_energii = pomiary_clean["energia"].isna()
display(pomiary_clean.loc[
    braki_energii, ["czas", "urzadzenie", "energia"]
])
```

Braki dotyczą U-03 w dniu 12 stycznia o 08:00 i U-04 w dniu 21 stycznia o 20:00. Nie masz dodatkowego źródła pozwalającego odtworzyć te pomiary. Pozostawiasz więc oba braki. Różne urządzenia i terminy nie wystarczają, aby ustalić przyczynę albo mechanizm powstania braków.

NaN oznacza brak wartości, zero oznacza zmierzoną wartość równą zero, a tekst N/D jest jedynie umownym markerem. Ich utożsamienie zmienia informację. W tym zbiorze wstawienie zer nie zmieni sumy obliczanej z pominięciem braków, ale zmniejszy średnią, bo jej mianownik wzrośnie z 718 do 720.

Strategia energii	Wiersze	Suma kWh	Średnia kWh
Pozostawienie braków	720	10 001,16	13,93
Usunięcie wierszy z brakami	718	10 001,16	13,93
Zastąpienie braków zerem	720	10 001,16	13,89
Uzupełnienie średnią	720	10 029,02	13,93

Wartości w tabeli są zaokrąglone do dwóch miejsc. Uzupełnienie średnią, czyli imputacja, dodaje oszacowania, nie odczyty z urządzeń. Może mieć uzasadnienie w innych zadaniach, lecz wymaga jawnej metody i oceny wpływu na wynik. Tutaj nie jest potrzebne. Usunięcie całych wierszy pozbawiłoby cię także poprawnych temperatur, drgań i produkcji.

W dalszej analizie podawaj liczbę dostępnych pomiarów energii. Suma 10 001,16 kWh opisuje 718 znanych wartości; bez dwóch brakujących odczytów nie jest potwierdzonym pełnym zużyciem dla całej siatki 720 obserwacji.

Duplikat nie jest tym samym co konflikt

Pełny duplikat powtarza wszystkie pola. Jeżeli potwierdzisz, że jest drugą kopią tego samego zapisu, usunięcie kopii nie traci treści. Powtórzony klucz czas i urzadzenie może natomiast obejmować różne wartości energii. Wtedy wiadomo, że występuje konflikt, ale nadal nie wiadomo, który odczyt jest prawidłowy.

Metoda duplicated() domyślnie oznacza kolejne wystąpienia, pozostawiając pierwsze nieoznaczone. Ustawienie keep=False pokazuje wszystkie wiersze należące do powtórzeń. Parametr subset definiuje, które pola uznajesz za klucz; jest więc częścią decyzji merytorycznej.

Eksperyment wykonaj na pliku pomocniczym, oddzielnie od pomiary_clean. W tym pliku przecinek jest separatorem dziesiętnym, a średnik rozdziela kolumny.

```
demo_raw = pd.read_csv(
    DATA_RAW / "przyklady_kontrolowane_m2.csv",
    sep=";", encoding="utf-8", decimal=",", na_values=["N/D"]
)
print(demo_raw.duplicated().sum())
demo = demo_raw.drop_duplicates().copy()
demo["do_wyjasnienia"] = demo.duplicated(
    subset=KLUCZ, keep=False
)
print(len(demo), demo["do_wyjasnienia"].sum())
```

Pełna kopia rekordu K-003 daje jedno dodatkowe wystąpienie. Po jej usunięciu pozostaje osiem wierszy, z których dwa mają konflikt klucza: K-006 i K-007, oba dla U-06 w dniu 16 lutego o 20:00. Energia wynosi odpowiednio 14,9 i 15,4 kWh. Flaga zachowuje oba zapisy do wyjaśnienia.

Przy konflikcie keep="first" wybiera pierwszy wiersz w aktualnej kolejności, a keep="last" — ostatni. Żadna z tych opcji nie dowodzi, który pomiar jest prawdziwy. Keep=False usunęłoby oba, tworząc lukę. Do rozstrzygnięcia potrzebujesz reguły źródła, na przykład wiarygodnej informacji o korekcie. W głównym zbiorze oba liczniki duplikatów wynoszą zero, więc niczego nie usuwasz.

Trzy kontrole identyfikatora

Identyfikator musi być obecny, mieć właściwy format i wskazywać istniejące urządzenie. Wzorzec U-NN opisuje zapis, lecz nie potwierdza członkostwa w kartotece. U-09 spełnia wzorzec U-NN, a jednak nie występuje w używanej kartotece. Nie wolno zmienić go na U-08 wyłącznie dlatego, że taka wartość pasowałaby do listy.

```
ids = pomiary_clean["urzadzenie"].astype("string")
karta = urzadzenia_clean["urzadzenie"].astype("string")
assert ids.notna().all()
assert ids.str.fullmatch(r"U-\d{2}", na=False).all()
assert karta.notna().all()
assert karta.is_unique
assert not (set(ids) - set(karta))
assert not pomiary_clean.duplicated(subset=KLUCZ).any()
assert len(pomiary_clean) == 720
assert pomiary_clean["energia"].isna().sum() == 2
```

Kontrole potwierdzają również świadomy brak operacji. To, że pozostawiono 720 rekordów i dwa braki energii, nie jest pominięciem etapu czyszczenia. Jest wynikiem decyzji, której skutki zmierzono. W notebooku zachowaj sekcję „Dziennik decyzji dotyczących jakości danych” i uzupełniaj ją po kolejnych rozdziałach.

Jak zapisać decyzję

Przykładowy wpis może brzmieć: „W energii występują dwa braki, dla U-03 i U-04. Brak wiarygodnego źródła do odtworzenia wartości. Pozostawiam braki; odrzucam zero i imputację, ponieważ tworzyłyby niezmierzone wartości. Nie usuwam wierszy, aby zachować pozostałe pomiary. Kontrola: 720 rekordów, dwa braki energii, suma znanych pomiarów bez zmian”.

Przy konflikcie zapisz także identyfikatory rekordów, różniące się pola i informację potrzebną do rozstrzygnięcia. Nie wpisuj, że skontaktowano się ze źródłem, jeśli to dopiero plan. Dziennik powinien odróżniać wykonane działania od spraw oczekujących.

Ćwiczenie w notebooku

Na danych demonstracyjnych wskaż pełną kopię, konflikt klucza i urządzenie spoza kartoteki. Wyjaśnij, dlaczego tylko pierwszy przypadek pozwala od razu usunąć kopię rekordu. Powtórz sprawdzenie klucza po ujednoliceniu dat w rozdziale 4 — odmienne zapisy czasu mogą ukrywać tę samą chwilę.

Sprawdź rozumienie 3

Po zastąpieniu braków energii zerem suma się nie zmieniła. Czy taki wynik dowodzi, że operacja była neutralna? Jaką drugą miarę i jaką informację o pochodzeniu wartości należy sprawdzić?

Rozdział 4 Typy daty kategorie i jednostki

Konwersja ma uporządkować reprezentację danych bez niezauważonej utraty informacji. Najpierw zachowujesz stan przed zmianą, następnie określasz regułę, wykonujesz przekształcenie i kontrolujesz wynik. Zasada ta dotyczy dat, liczb, tekstu i jednostek. Poprawny typ techniczny nadal nie oznacza poprawnej wartości domenowej.

Jawny format czasu

W głównym pliku czas ma format rok-miesiąc-dzień godzina:minuta. Errors="coerce" zamienia nierozpoznane wartości na NaT. Nie naprawia ich i nie wyjaśnia przyczyny. Nowe braki należy wykryć przed zastąpieniem pierwotnej kolumny.

```
czas_przed = pomiary_clean["czas"].copy()
czas_po = pd.to_datetime(
    czas_przed, format="%Y-%m-%d %H:%M", errors="coerce"
)
nowe_braki = czas_po.isna() & czas_przed.notna()
assert not nowe_braki.any()
assert czas_po.notna().all()
pomiary_clean["czas"] = czas_po
assert not pomiary_clean.duplicated(subset=KLUCZ).any()
```

Wszystkie 720 wartości przechodzi konwersję. Zachowanych jest 90 znaczników czasu. Kontrola klucza po konwersji jest potrzebna, bo dwa różne teksty daty mogą oznaczać tę samą chwilę. Format źródłowy określa interpretację; nie zgaduj znaczenia niejednoznacznego zapisu 03/04/2026.

W pliku demonstracyjnym dozwolone są trzy formaty. Poniższy eksperyment korzysta z ramki demo z rozdziału 3 i zapisuje datę w nowej kolumnie, pozostawiając oryginalny tekst.

```
formaty = ["%Y-%m-%d %H:%M", "%d.%m.%Y %H:%M",
           "%Y/%m/%d %H:%M"]
demo_czas = pd.Series(pd.NaT, index=demo.index,
                      dtype="datetime64[ns]")
for fmt in formaty:
    demo_czas = demo_czas.fillna(pd.to_datetime(
        demo["czas"], format=fmt, errors="coerce"))
demo["czas_po"] = demo_czas
display(demo.loc[demo_czas.isna(), ["rekord_id", "czas"]])
```

Nierozpoznany pozostaje K-005 z datą 31.02.2026 08:00. Taki dzień nie istnieje. Nie zastępuj go arbitralnie ostatnim dniem lutego: potrzebujesz informacji ze źródła, a do tego czasu zachowujesz rekord jako wymagający wyjaśnienia.

Najpierw znaczenie kategorii potem typ

Usunięcie spacji brzegowych i ujednolicenie wielkości liter porządkują zapis. Nie rozstrzygają jednak, czy eco znaczy to samo co oszczedny. Dozwolone wartości ustal na podstawie słownika. Nieznana kategoria powinna zatrzymać proces, a nie zostać cicho zastąpiona najbardziej podobną nazwą.

```
tryb_norm = (pomiary_clean["tryb"].astype("string")
             .str.strip().str.lower())
status_norm = (pomiary_clean["status"].astype("string")
               .str.strip().str.upper())
assert tryb_norm.notna().all()
assert status_norm.notna().all()
assert set(tryb_norm) <= {"auto", "oszczedny"}
assert set(status_norm) <= {"OK"}
pomiary_clean["tryb"] = tryb_norm.astype("category")
pomiary_clean["status"] = status_norm.astype("category")
```

Po normalizacji otrzymujesz 360 rekordów auto, 360 oszczedny i 720 OK. Zapis zmienił się w dwóch komórkach: AUTO na auto oraz ok na OK. Żaden rekord nie został usunięty. Samo astype("category") nie sprawdza dozwolonego słownika — dlatego kontrola znaczenia poprzedza zmianę typu.

Schemat kolumn liczbowych

Identyfikator pozostaje tekstem, nawet jeśli zawiera cyfry. Produkcja to liczba całkowita, ale po rozstrzygnięciu błędu może zawierać brak. Typ Int64 z wielką literą I obsługuje taką sytuację. Nie jest tym samym co zwykły int64, który nie przechowuje wartości pd.NA.

```
SCHEMAT = {
    "urzadzenie": "string",
    "temperatura": "float64", "drgania": "float64",
    "energia": "float64", "produkcja": "Int64"
}
pomiary_clean = pomiary_clean.astype(SCHEMAT)
assert pomiary_clean.shape == (720, 8)
assert pomiary_clean["energia"].isna().sum() == 2
```

W innym źródle zapis „12,8” może być tekstem. Przy jednolitym formacie użyj decimal="," już podczas importu. Gdy formaty są mieszane, zachowaj serię źródłową, rozpoznaj markery braków, zastosuj udokumentowaną regułę zapisu i użyj to_numeric(). Nowe braki po konwersji wykrywaj maską po.isna() & przed.notna(). Nie zamieniaj dowolnych błędnych tekstów na zero.

Jednostka jest częścią wartości

Liczba 13200 nie mówi sama, czy energia została zapisana w Wh, czy w kWh. Po poprawnej konwersji liczbowej nadal możesz mieć nieporównywalne wartości. Jednostkę potwierdza słownik, dokumentacja albo osobna kolumna źródła. Model językowy może zasugerować hipotezę, ale wielkość liczby nie stanowi dowodu jej jednostki.

W pliku demonstracyjnym pole jednostka_energii pozwala wykonać jawne przeliczenie do kWh. Energia została już odczytana liczbowo z decimal="," i na_values=["N/D"]. Nie zmieniaj kolumny wejściowej: dopisz wynik w energia_kwh.

```
przelicznik = {"Wh": 0.001, "kWh": 1.0}
jednostka = (demo["jednostka_energii"].astype("string")
             .str.strip())
wsp = jednostka.map(przelicznik)
nierozpoznane = demo["energia"].notna() & wsp.isna()
assert not nierozpoznane.any()
demo["energia_kwh"] = demo["energia"] * wsp
display(demo[["energia", "jednostka_energii", "energia_kwh"]])
```

Wartość 13200 Wh daje 13,2 kWh, a 12,8 kWh pozostaje bez zmiany. Reguła obejmuje tylko znane jednostki. Brakującą jednostkę istniejącego pomiaru trzeba wyjaśnić; nie wolno nadać jej domyślnie współczynnika 1. W głównym zbiorze energia jest już w kWh, więc żadne przeliczenie nie jest potrzebne.

Tak samo postępujesz z temperaturą: liczby w stopniach Celsjusza, Fahrenheita i kelwinach wymagają różnych reguł. Dla potwierdzonego zapisu w °F stosujesz (wartość − 32) × 5/9, a dla K odejmujesz 273,15, aby otrzymać °C. Zachowuj wartość i jednostkę źródłową, ponieważ to one pozwalają sprawdzić przeliczenie.

Ćwiczenie w notebooku

Zapisz tabelę konwersji: pole, stan przed, reguła, stan po i kontrola nowych braków. Dla głównego zbioru sprawdź 720 rekordów, dwa braki energii, brak NaT i dozwolone kategorie. W eksperymencie wskaż niemożliwą datę oraz przeliczenie 13200 Wh na 13,2 kWh. Nie łącz danych demonstracyjnych z pomiarami.

Sprawdź rozumienie 4

Konwersja dat zakończyła się bez wyjątku, lecz utworzyła jeden NaT. Czy możesz uznać ją za udaną? Co musisz sprawdzić przed zastąpieniem kolumny źródłowej?

Rozdział 5 Zakresy dopuszczalne i wartości odstające

Wartość może być nietypowa i jednocześnie poprawna. Może też mieć poprawny zapis liczbowy, ale naruszać znaczenie pola. W tym rozdziale połączysz regułę domenową, sygnał statystyczny i kontekst urządzenia. Dopiero te dowody pozwalają zdecydować, czy wartość poprawić, zachować czy pozostawić do wyjaśnienia.

Reguła twarda i próg diagnostyczny

W naszym kontrakcie temperatura mieści się od 0 do 100 °C, drgania nie mogą być ujemne, energia ma być dodatnia, gdy istnieje pomiar, a produkcja jest nieujemną liczbą sztuk. Próg drgań powyżej 2 mm/s jest diagnostyczny: wskazuje przypadek do sprawdzenia, nie automatyczny błąd ani potwierdzoną awarię.

```
reguly = {
    "temperatura_poza":
        ~pomiary_clean["temperatura"].between(0, 100),
    "drgania_ujemne": pomiary_clean["drgania"].lt(0),
    "drgania_do_sprawdzenia": pomiary_clean["drgania"].gt(2),
    "energia_niedodatnia": pomiary_clean["energia"].notna()
        & pomiary_clean["energia"].le(0),
    "produkcja_ujemna": pomiary_clean["produkcja"].lt(0)
}
display(pd.Series({n: m.sum() for n, m in reguly.items()}))
```

Maski tylko wskazują rekordy. W głównych danych otrzymasz jeden ujemny wynik produkcji i jeden pomiar drgań powyżej progu, bez pozostałych naruszeń zakresów. Kompletność pól sprawdzaj osobno: porównanie z granicą nie zawsze wychwyci brak.

Co mierzy IQR

Pierwszy kwartyl Q1 i trzeci kwartyl Q3 wyznaczają środkową część rozkładu, obejmującą około 50% obserwacji. Rozstęp międzykwartylowy IQR to różnica Q3 − Q1. Popularna reguła oznacza wartości poniżej Q1 − 1,5 × IQR i powyżej Q3 + 1,5 × IQR. Są to granice wynikające z danych, a nie dopuszczalne granice pracy urządzenia.

Dla drgań Q1 = 1,1320, Q3 = 1,3515, a IQR = 0,2195 mm/s. Granice wynoszą 0,80275 i 1,68075 mm/s. Poza nimi leżą trzy wartości: 0,781, 0,786 i 4,850 mm/s. Reguła wykryła nietypowość — nie rozstrzygnęła jej przyczyny.

Detektor i kontekst urządzenia

```
def wykryj_iqr(seria):
    q1, q3 = seria.dropna().quantile([0.25, 0.75])
    iqr = q3 - q1
    dolna, gorna = q1 - 1.5 * iqr, q3 + 1.5 * iqr
    flaga = (seria.lt(dolna) | seria.gt(gorna)).fillna(False)
    return flaga, dolna, gorna

flaga_drg, dolna, gorna = wykryj_iqr(pomiary_clean["drgania"])
print(dolna, gorna, flaga_drg.sum())
display(pomiary_clean.loc[
    flaga_drg, ["czas", "urzadzenie", "drgania"]
])
```

W energii kwartyle obliczasz z 718 istniejących wartości. Brak nie jest wartością odstającą. Dla bardzo małej próbki albo zerowego IQR interpretacja tej reguły wymaga dodatkowej ostrożności; funkcja nie staje się przez to uniwersalnym automatem czyszczącym.

Pole	Dolna granica	Górna granica	Liczba sygnałów
temperatura °C	50,525	68,245	0
drgania mm/s	0,803	1,681	3
energia kWh	9,634	18,184	0
produkcja szt. przed korektą	130,5	174,5	1

Dwa niskie wyniki drgań pochodzą z U-01. Dla samego U-01 lokalne granice IQR wynoszą około 0,765–1,309 mm/s. Wartości 0,781 i 0,786 mieszczą się w tym profilu i spełniają regułę nieujemności. W przypadku kursowym klasyfikujesz je jako prawidłowe wartości nietypowe w skali całego zbioru.

Wynik 4,850 mm/s pochodzi z U-05 z 8 lutego o 20:00. Jest pojedynczym skokiem między typowymi pomiarami. Wykres uporządkowany w czasie pozwala to zobaczyć, lecz nie odróżnia błędu czujnika od rzeczywistego zdarzenia. Potrzebne byłyby dziennik zdarzeń, stan czujnika lub zapis źródłowy. Pozostawiasz wartość z klasyfikacją „do sprawdzenia”.

Zmiana jednej wartości zamiast usunięcia wiersza

Produkcja −12 szt. dla U-03 z 2 lutego o 08:00 narusza regułę nieujemnej liczby wyprodukowanych sztuk. Nie masz jednak podstaw, aby wpisać +12, zero ani średnią. Zastępujesz wyłącznie tę komórkę brakiem. Pozostałe pomiary z tego samego rekordu pozostają dostępne.

```
blad_produkcji = pomiary_clean["produkcja"].lt(0)
assert blad_produkcji.sum() == 1
pomiary_clean.loc[blad_produkcji, "produkcja"] = pd.NA
assert pomiary_clean.shape == (720, 8)
assert pomiary_clean["produkcja"].isna().sum() == 1
assert pomiary_clean["energia"].isna().sum() == 2
assert np.isclose(pomiary_clean["drgania"].max(), 4.85)
assert pomiary_raw["produkcja"].min() == -12
```

Po korekcie zbiór ma więcej braków niż przed nią, ale wierniej przedstawia dostępną wiedzę: niepoprawna liczba nie udaje już pomiaru. Dlatego zmniejszenie liczby pustych komórek nie może być samodzielnym celem czyszczenia. W dzienniku zachowaj oryginalną wartość, regułę, decyzję i kontrolę.

Gdyby usunąć wszystkie cztery wiersze wskazane przez IQR, zostałoby 716 rekordów. Utracono by poprawne niskie drgania U-01, niewyjaśniony pomiar U-05 oraz poprawne pola z wiersza U-03. Kontrolowane postępowanie zachowuje 720 rekordów, zmienia jedną wartość i dokumentuje trzy decyzje o zachowaniu drgań.

Pomoc modelu językowego

Możesz poprosić LLM o funkcję zwracającą maskę IQR i progi, wyraźnie zakazując usuwania rekordów. Sprawdź kod na danych: oczekujesz trzech sygnałów i granic 0,80275 oraz 1,68075 mm/s. Przyjęcie detektora po teście nie oznacza przyjęcia ewentualnej sugestii automatycznego usuwania. Model proponuje narzędzie, a decyzję uzasadniają dane i reguły.

Ćwiczenie w notebooku

Oblicz granice dla czterech pól liczbowych. Dla U-01 i U-05 narysuj drgania w kolejności czasu, podpisz osie i jednostkę, a następnie porównaj sygnały globalne z lokalnymi. Opisz cztery przypadki: jedną błędną produkcję, jeden skok do sprawdzenia i dwa poprawne niskie pomiary.

Sprawdź rozumienie 5

Dlaczego 0,781 mm/s i 4,850 mm/s, wskazane przez ten sam detektor IQR, prowadzą do różnych klasyfikacji? Czy samo pozostawanie w lokalnym IQR jest gwarancją sprawności urządzenia?

Rozdział 6 Transformacje łączenie i walidacja wyniku

Wynik modułu ma być zrozumiały także dla osoby, która nie obserwowała twojej pracy. Przygotujesz więc czytelne nazwy, kolumny pochodne i jednoznaczne połączenie z kartoteką. Następnie sprawdzisz rezultat, zapiszesz plik i ponownie go odczytasz. Dokumentacja pozostanie częścią wyniku, ponieważ nazwa clean nie oznacza braku ograniczeń.

Nazwy i kolumny pochodne

Zmień nazwę typ na typ_urzadzenia, aby po połączeniu było jasne, czego dotyczy. Pole rok_miesiac utwórz z czasu pomiaru. Zapis YYYY-MM ułatwi agregowanie danych w module 3, ale nie zastępuje dokładnego znacznika czasu.

```
urzadzenia_clean = urzadzenia_clean.rename(
    columns={"typ": "typ_urzadzenia"}
)
urzadzenia_clean["urzadzenie"] = (
    urzadzenia_clean["urzadzenie"].astype("string")
)
urzadzenia_clean["data_instalacji"] = pd.to_datetime(
    urzadzenia_clean["data_instalacji"], errors="raise"
)
pomiary_clean["rok_miesiac"] = (
    pomiary_clean["czas"].dt.to_period("M").astype("string")
)
assert set(pomiary_clean["rok_miesiac"]) == {"2026-01", "2026-02"}
```

Kolumna pochodna potrzebuje definicji: danych wejściowych, reguły, typu i jednostki, jeżeli ma sens liczbowy. Zapisz, że rok_miesiac powstaje z czas i ma typ string. Pozostawienie obu pól umożliwia sprawdzenie, czy przypisanie do miesiąca jest poprawne.

Kontrakt połączenia

Wiele pomiarów jednego urządzenia łączysz z jednym rekordem kartoteki. Taka relacja to wiele do jednego, w Pandas many_to_one. Oczekujesz jednego zestawu metadanych na każdy pomiar i niezmienionej liczby wierszy. Klucz kartoteki musi więc być kompletny i unikalny.

Połączenie left zachowuje wszystkie pomiary, ujawniając brak dopasowania jako brak metadanych. Inner zachowałoby tylko dopasowane wiersze, więc przy nieznanym urządzeniu mogłoby ukryć problem przez usunięcie jego pomiarów. Wybór rodzaju połączenia wynika z celu, nie z tego, który wariant akurat działa na aktualnych danych.

Łączenie z kontrolą liczby rekordów

Przed połączeniem sprawdź klucze obu tabel i zapamiętaj liczbę pomiarów. Kartoteka może zawierać urządzenia bez pomiarów w danym okresie; taki przypadek należy odnotować, ale nie jest tym samym co pomiar bez kartoteki.

```
assert pomiary_clean["urzadzenie"].notna().all()
assert urzadzenia_clean["urzadzenie"].notna().all()
assert urzadzenia_clean["urzadzenie"].is_unique
ids_p = set(pomiary_clean["urzadzenie"])
ids_u = set(urzadzenia_clean["urzadzenie"])
assert not (ids_p - ids_u)
print("Urządzenia bez pomiarów:", ids_u - ids_p)
liczba_przed = len(pomiary_clean)
pomiary_clean = pomiary_clean.merge(
    urzadzenia_clean, on="urzadzenie", how="left",
    validate="many_to_one", indicator=True
)
assert len(pomiary_clean) == liczba_przed
assert pomiary_clean["_merge"].eq("both").all()
pomiary_clean = pomiary_clean.drop(columns="_merge")
```

Validate zatrzymuje operację, jeżeli kartoteka narusza relację wiele do jednego. Indicator tworzy kolumnę _merge do kontroli dopasowania. W aktualnych danych jest 720 dopasowań both. Przy left nie powstają rekordy right_only; urządzenia występujące wyłącznie w kartotece sprawdzaj różnicą zbiorów albo osobnym diagnostycznym outer merge.

Gdyby dopisać drugi rekord U-03 w kartotece i nie zastosować validate, 90 pomiarów tego urządzenia uzyskałoby po dwa dopasowania. Tabela urosłaby z 720 do 810 wierszy. Sam fakt, że merge() zwrócił wynik, nie potwierdza poprawności relacji.

Wiek urządzenia w dniu pomiaru

```
pomiary_clean["wiek_urzadzenia_dni"] = (
    pomiary_clean["czas"].dt.normalize()
    - pomiary_clean["data_instalacji"]
).dt.days.astype("Int64")
assert pomiary_clean["wiek_urzadzenia_dni"].notna().all()
assert pomiary_clean["wiek_urzadzenia_dni"].ge(0).all()
```

Normalize sprowadza czas pomiaru do początku dnia. Różnica oznacza wiek od daty instalacji do dnia pomiaru, nie do dzisiejszej daty. W zbiorze wynosi 1361–3516 dni. Dla instalacji 5 listopada 2020 i pomiaru 1 stycznia 2026 otrzymasz 1883 dni.

Jawny schemat pliku wynikowego

Porządek kolumn ma pokazywać model danych: identyfikację i czas, kartotekę, pomiary, a na końcu kontekst pracy. Po dodaniu czterech pól kartoteki i dwóch pól pochodnych otrzymasz 14 kolumn. Schemat zapisany w kodzie przyda się także po ponownym odczycie CSV.

```
KOLUMNY_WYNIK = [
    "czas", "rok_miesiac", "urzadzenie",
    "typ_urzadzenia", "lokalizacja", "moc_nominalna_kw",
    "data_instalacji", "wiek_urzadzenia_dni",
    "temperatura", "drgania", "energia", "produkcja",
    "tryb", "status"
]
SCHEMAT_WYNIK = {
    "rok_miesiac": "string", "urzadzenie": "string",
    "typ_urzadzenia": "string", "lokalizacja": "string",
    "moc_nominalna_kw": "float64", "temperatura": "float64",
    "drgania": "float64", "energia": "float64",
    "wiek_urzadzenia_dni": "Int64", "produkcja": "Int64",
    "tryb": pd.CategoricalDtype(
        pd.Index(["auto", "oszczedny"], dtype="string")),
    "status": pd.CategoricalDtype(pd.Index(["OK"], dtype="string"))
}
pomiary_clean = pomiary_clean[KOLUMNY_WYNIK].astype(SCHEMAT_WYNIK)
```

Dwie kolumny dat zostały przygotowane wcześniej. Pozostałe typy odtwarza SCHEMAT_WYNIK. CategoricalDtype określa również dozwolone kategorie i tekstowy typ ich etykiet, aby odczyt odtworzył ten sam schemat. Jawna lista kolumn pozwala wykryć przypadkowe dodanie, pominięcie lub przestawienie pola.

Kontrola przed zapisem

```
assert pomiary_clean.shape == (720, 14)
assert not pomiary_clean.duplicated(subset=KLUCZ).any()
assert pomiary_clean["energia"].isna().sum() == 2
assert pomiary_clean["produkcja"].isna().sum() == 1
assert pomiary_clean["produkcja"].dropna().ge(0).all()
META = ["typ_urzadzenia", "lokalizacja",
        "moc_nominalna_kw", "data_instalacji"]
assert pomiary_clean[META].notna().all().all()
assert pomiary_clean["wiek_urzadzenia_dni"].ge(0).all()
assert pomiary_clean["drgania"].gt(2).sum() == 1
assert np.isclose(pomiary_clean["energia"].sum(),
                  10001.16, rtol=0, atol=0.005)
```

Te liczby kontrolne dotyczą konkretnej wersji danych kursowych. Dla nowego źródła ustal oczekiwania na nowo. Gdy asercja zatrzyma notebook, zbadaj przyczynę; nie zmieniaj oczekiwanej liczby tylko po to, aby test przeszedł. Asercje są tutaj kontrolami wykonywanymi w zwykłej sesji notebooka, nie kompletnym systemem walidacji produkcyjnej.

Zapis i ponowny odczyt

CSV przechowuje wartości tekstowe, nie schemat Pandas. Nie zachowuje kategorii ani typu Int64. Dlatego po odczycie ponownie interpretujesz daty i ustawiasz typy. Index=False zapobiega zapisaniu technicznego indeksu jako dodatkowej kolumny.

```
DATA_PROCESSED = PROJECT_DIR / "data" / "processed"
TABLES = PROJECT_DIR / "outputs" / "tables"
DATA_PROCESSED.mkdir(parents=True, exist_ok=True)
TABLES.mkdir(parents=True, exist_ok=True)
plik_wynik = DATA_PROCESSED / "pomiary_clean.csv"
pomiary_clean.to_csv(
    plik_wynik, index=False, encoding="utf-8",
    date_format="%Y-%m-%d %H:%M"
)
wczytany = pd.read_csv(
    plik_wynik, encoding="utf-8",
    parse_dates=["czas", "data_instalacji"]
).astype(SCHEMAT_WYNIK)
assert wczytany.columns.tolist() == KOLUMNY_WYNIK
pd.testing.assert_frame_equal(
    pomiary_clean.reset_index(drop=True),
    wczytany.reset_index(drop=True),
    check_exact=False, rtol=1e-12, atol=1e-12
)
```

Porównujesz nie tylko kształt, lecz również wartości, braki i typy. Mała tolerancja liczb zmiennoprzecinkowych uwzględnia techniczny zapis i odczyt, nie służy do ukrywania merytorycznych różnic. Pomyślna kontrola samego shape nie wykryłaby na przykład zamiany dwóch wartości energii.

Tożsamość pliku i odtwarzalność

```
import hashlib
suma_sha256 = hashlib.sha256(plik_wynik.read_bytes()).hexdigest()
print("SHA-256:", suma_sha256)
```

Wynik SHA-256 ma 64 znaki szesnastkowe i identyfikuje dokładną zawartość pliku. Zapisz go w README.md wraz z datą utworzenia rezultatu. Zmiana formatowania CSV może zmienić sumę mimo równoważności danych. Zgodny skrót nie dowodzi jakości; potwierdza jedynie zgodność bajtów z wcześniej odnotowaną wersją.

Na zakończenie zapisz notebook, zrestartuj kernel i wykonaj wszystkie komórki od góry. Wyniki powinny powstać ponownie z data/raw, bez ręcznych poprawek i dodatkowych obiektów pozostałych w pamięci. Kod zapisu zastępuje wcześniejszy rezultat w data/processed; archiwizuj osobno wersję, którą chcesz zachować do porównania.

Tabela jakości i dokumentacja

Utwórz jedno porównanie stanu przed i po. Nie przepisuj ręcznie samych oczekiwanych liczb — oblicz je z ramek. Przykładowa funkcja zbiera podstawowe miary, a przyrost liczby kolumn pokaże efekt dołączenia kartoteki i pól pochodnych.

```
def miary_jakosci(df):
    return pd.Series({
        "rekordy": len(df), "kolumny": df.shape[1],
        "duplikaty_klucza": df.duplicated(KLUCZ).sum(),
        "braki_energia": df["energia"].isna().sum(),
        "braki_produkcja": df["produkcja"].isna().sum(),
        "produkcja_ujemna": df["produkcja"].lt(0).sum(),
        "warianty_tryb": df["tryb"].nunique(),
        "warianty_status": df["status"].nunique(),
        "drgania_iqr": wykryj_iqr(df["drgania"])[0].sum(),
        "drgania_powyzej_2": df["drgania"].gt(2).sum()
    })
audyt = pd.DataFrame({"przed": miary_jakosci(pomiary_raw),
                      "po": miary_jakosci(wczytany)})
audyt.loc["braki_metadanych", "po"] = (
    wczytany[META].isna().any(axis=1).sum())
audyt.rename_axis("kontrola").to_csv(
    TABLES / "M2_audyt_jakosci.csv", encoding="utf-8"
)
```

Brak wartości „przed” dla metadanych znaczy „nie dotyczy tabeli wejściowej”, a nie zero. W notebooku dopisz interpretację: rekordów nadal jest 720, kolumn 14, wariantów trybu i statusu odpowiednio dwa i jeden, braków energii dwa, produkcji jeden. Trzy sygnały IQR drgań i jeden pomiar powyżej 2 mm/s pozostają zachowane.

Uzupełnij docs/slownik_danych.md o typ_urzadzenia, rok_miesiac i wiek_urzadzenia_dni oraz zasady dopuszczenia braków. Zachowaj opis źródłowego pola typ, wskazując jego nową nazwę w wyniku. Dziennik decyzji w notebooku powinien obejmować także zachowane nietypowe wartości, a README — środowisko i kolejność odtworzenia.

Jeśli korzystasz z LLM, zapisuj prompt, odpowiedź, test, werdykt i uzasadnienie w docs/rejestr_uzycia_LLM.csv, z wartością M2 w polu modul. Rejestruj rzeczywiste użycie, nie przykładową rozmowę jako wykonaną. Przekazuj modelowi tylko potrzebny, dozwolony kontekst; nie wysyłaj haseł ani poufnych danych.

Sprawdź rozumienie 6

Po połączeniu otrzymujesz 810 zamiast 720 wierszy. Dlaczego to sygnał do zatrzymania pracy? Która kontrola powinna wykryć przyczynę przed zapisaniem wyniku?

Podsumowanie i odpowiedzi do samooceny

Moduł 2 kończysz z przygotowanym zbiorem i udokumentowanymi decyzjami. Potrafisz wykazać, że import, czyszczenie, łączenie i zapis przeszły odpowiednie kontrole.

Odpowiedzi do pytań

1. Najpierw popraw interpretację CSV. 720 wierszy przy jednej kolumnie nie spełnia kontraktu 720 × 8. Sprawdź separator, nazwy kolumn i próbkę. Audyt wartości w źle podzielonej tabeli nie będzie miarodajny.

2. Próbka nie potwierdza kompletności, a typ liczbowy nie potwierdza zakresu ani jednostki. Policz braki w całym zbiorze i porównaj wartości z regułami słownika.

3. Niezmieniona suma nie oznacza neutralnej operacji. Wstawienie zer zwiększa liczbę wartości uwzględnianych w średniej i obniża ją. Przede wszystkim zamienia niewiedzę na pozorny pomiar. Sprawdź count(), średnią i informację o pochodzeniu wartości.

4. Errors="coerce" nie zgłasza wyjątku także dla błędnych dat. Sprawdź nowe NaT i źródłowy zapis. Bez dowodu nie poprawiaj daty; zachowaj ją do wyjaśnienia.

5. Niski pomiar mieści się w profilu U-01 i spełnia regułę domenową, a wysoki wynik U-05 jest niewyjaśnionym skokiem ponad próg diagnostyczny. Lokalny IQR daje kontekst statystyczny, nie gwarancję sprawności technicznej. Do diagnozy potrzebne są dodatkowe dowody.

6. Przyrost wierszy oznacza, że pomiary mogły zostać zwielokrotnione. W kontrprzykładzie drugi rekord U-03 w kartotece dodał 90 dopasowań. Unikalność klucza kartoteki i validate="many_to_one" powinny zatrzymać połączenie.

Sprawdzenie gotowości do modułu 3

- Notebook wykonuje się od początku po restarcie kernela, a źródła pozostają niezmienione.
- Wynik ma 720 rekordów i 14 opisanych kolumn oraz unikalny klucz czas i urzadzenie.
- Dwa braki energii, jeden brak produkcji i zachowane nietypowe drgania mają zapisane uzasadnienie.
- Połączenie ma kompletne metadane, a ponowny odczyt odtwarza wartości, braki i typy.
- Tabela jakości, słownik, dziennik decyzji i README odpowiadają zapisanej wersji pliku.

W module 3 punktem startowym będzie data/processed/pomiary_clean.csv. Dane są przygotowane do stawiania pytań analitycznych, ale nadal mają jawne ograniczenia. Uwzględniaj je w agregacjach i interpretacji wyników.

Słownik pojęć i pól wynikowych

Kontrakt danych to zestaw sprawdzalnych oczekiwań wobec struktury i zawartości. Audyt mierzy stan zbioru przed zmianami. Walidacja sprawdza, czy wynik spełnia ustalone oczekiwania. Te etapy są powiązane, lecz nie zastępują się wzajemnie.

Klucz złożony identyfikuje obserwację przez kilka pól — tutaj czas i urzadzenie. Pełny duplikat powtarza wszystkie wartości, natomiast konflikt klucza może zawierać sprzeczne pomiary. Kardynalność opisuje liczbę dopasowań między tabelami; many_to_one oznacza wiele rekordów z lewej tabeli przypisanych do jednego rekordu z prawej.

Imputacja zastępuje brak oszacowaniem. IQR mierzy rozstęp między pierwszym i trzecim kwartylem. Maska logiczna wskazuje rekordy spełniające warunek, lecz sama ich nie zmienia. Asercja zatrzymuje wykonanie, gdy oczekiwany warunek nie jest spełniony. SHA-256 identyfikuje zawartość pliku, nie jego jakość.

Pole wyniku	Znaczenie i jednostka
czas	Data i czas pomiaru; część klucza
rok_miesiac	Rok i miesiąc z pola czas; zapis YYYY-MM
urzadzenie	Tekstowy identyfikator zgodny z kartoteką
typ_urzadzenia	Typ z kartoteki; źródłowa nazwa typ
lokalizacja	Lokalizacja urządzenia z kartoteki
moc_nominalna_kw	Moc nominalna urządzenia w kW
data_instalacji	Data instalacji z kartoteki
wiek_urzadzenia_dni	Dni od instalacji do dnia pomiaru
temperatura	Temperatura w °C
drgania	Drgania w mm/s; sygnały nietypowe zachowane
energia	Energia w kWh na interwał; dwa braki
produkcja	Liczba sztuk na interwał; jeden brak po korekcie
tryb	Kategoria auto lub oszczedny
status	Kategoria OK