창원 누비자 공공데이터, AI가 읽고 코드가 검산한 기록

누구나 같은 파일로 같은 숫자를 다시 얻을 수 있도록 자료 출처, 절차, 검산 코드 전문, 대조 결과, 한계를 한 묶음으로 정리했습니다.

4종	14 : 10	3	1
공공데이터포털 누비자 자료	대조 항목 : 그대로 일치	AI 말과 달랐던 항목	두 공식 자료 사이에서 새로 찾은 불일치

자료 출처와 받는 법

자료	내용	받는 곳
(2018.06)창원시 무인대여 공영자전거 누비자 대여/반납 이력	2018년 6월 건별 기록 499,525행. 자전거번호, 출발·도착 터미널, 날짜, 시각 7개 열. CSV, cp949, 약 25MB	data.go.kr/data/15014737/fileData.do
경상남도 창원시_누비자 일일 이용현황	2020.01.01~2022.06.30 하루 단위 912행	data.go.kr/data/15102075/fileData.do
경상남도 창원시_누비자 이용현황	2008.10~2023.03 월 단위 174개월, 파일 끝에 빈 줄 3개	data.go.kr/data/15075281/fileData.do
경상남도 창원시_누비자 터미널	터미널 372곳의 번호, 이름, 구·동, 주소, 사용유무, 위도, 경도. XLSX	data.go.kr/data/15000545/fileData.do
경상남도 창원시_공영자전거 누비자 대여 반납 이력	2022년 12월, 2023년, 2024년, 2025년 건별 파일. 한 해 211~240MB. 이용허락 제한 없음. 형식이 같아 같은 코드로 확장 가능	창원시 빅데이터 포털 bigdata.changwon.go.kr 데이터셋 263

공공데이터포털 파일은 로그인 없이 내려받을 수 있습니다. 모든 자료에 이용자 개인정보는 들어 있지 않습니다.

절차 다섯 단계

1. **AI가 읽는다.** 파일을 AI에게 주고 열 구조 설명, 탐색 코드, 해석 문장을 받습니다. 하루 만에 과제용 PPT까지 나왔습니다.
2. **AI가 말한 숫자를 전부 받아 적는다.** 행 수, 중복, 이상치, 주기, 오차, 급감일, 터미널 수. 이것이 대조표의 왼쪽 열입니다.
3. **검산 코드를 따로 돌린다.** 원본 파일에서 다시 세는 verify.py를 실행합니다. 코랩이면 파일을 올리고 그대로 실행하면 됩니다. 결과가 오른쪽 열입니다.
4. **불일치는 원본을 열어 확인한다.** 빈 줄, 반올림, 이름 매칭 같은 원인을 찾고, 확인이 안 되면 미확인으로 남깁니다.
5. **시민이 보는 표로 바꾼다.** 터미널은 번호와 좌표로 고르고, 사실과 해석을 문장에서 구분합니다.

> 검산 코드도 AI와 함께 썼습니다. 다만 코드는 원본 행을 그대로 세기 때문에 사람이 한 줄씩 읽을 수 있고, 같은 파일로 누구나 같은 숫자를 얻습니다. 세어 보기 전까지 AI의 숫자는 전부 주장이었습니다.

검산 대조표

항목	AI 보고값	코드 재계산	판정	메모
2018년 6월 건별 기록 수	499,525건	499,525행	**일치**	
중복 행	2건	2행	**일치**	완전히 같은 행
1분 미만 대여	34,170건	34,170건	**일치**	그중 34,126건은 같은 터미널에 바로 반납
24시간 초과 대여	78건	78건	**일치**	
시간별 집계 점수	720점	720점	**일치**	30일 × 24시간
새벽 1~3시 대여	0건, 운영 중단	0건	**일치**	0건은 사실, 이유는 미확인
일일 이용현황	912일	912행	**일치**	2020.01.01~2022.06.30 빠진 날 없음
월별 이용현황	174개월	177행	**겉보기 불일치**	파일 끝에 빈 줄 3개, 실제 174개월. AI가 맞음
터미널 수	372곳	372행	**일치**	사용함 342, 사용안함 30, 2018년 6월 이력에 등장 276
요일·시각 평균 기준선 오차	MAE 251건	MAE 250.7건	**일치**	반올림 차이
주기	24시간, 168시간	ACF 24h 0.64, 168h 0.59	**일치**	국소 최고점 24, 48, … 192
급감일	10·19·28·30일, 비추정	10·11·28·30일 70% 미만	**부분 불일치**	19일은 73%, 11일은 67%. 기준에 따라 달라짐. 강수는 미확인
창원대 앞 터미널	3곳의 패턴	2018년 이력에는 1곳	**불일치**	이름에 창원대가 든 4곳 중 2곳은 2023년 등록, 1곳은 8km 밖 창원대로변 광장
두 공식 자료 교차	검토 없음	건별 499,523 vs 월별 471,500	**새로 발견**	차이 28,023건. 어떤 정제 규칙으로도 재현되지 않음. 미확인

AI 보고값은 2026년 10월 20~21일 AI와 함께 만든 과제 자료에 적힌 값, 코드 재계산은 10월 22일 verify.py 실행 결과.

창원대학교앞 75번 터미널, 2018년 6월 시간대별 대여·반납·순유출

시각	대여	반납	순유출	시각	대여	반납	순유출
0시	191	76	+115	12시	146	130	+16
1시	0	0	+0	13시	174	154	+20
2시	0	0	+0	14시	194	145	+49
3시	0	0	+0	15시	228	131	+97
4시	41	34	+7	16시	280	116	+164
5시	30	11	+19	17시	367	162	+205

시각	대여	반납	순유출	시각	대여	반납	순유출
6시	24	26	-2	18시	331	172	+159
7시	18	52	-34	19시	257	141	+116
8시	55	164	-109	20시	230	157	+73
9시	56	106	-50	21시	263	176	+87
10시	112	133	-21	22시	330	148	+182
11시	154	139	+15	23시	248	155	+93

한 달 합계 대여 3,729 · 반납 2,528 · 순유출 +1,201, 운영 터미널 276곳 중 순유출 11위. 순유출이 +이면 들어온 것보다 많이 나간 것. 거치대 재고는 자료에 없으므로 "저녁에 비기 쉽다"는 해석입니다.

검산 원칙

- AI가 말한 숫자는 모두 주장으로 두고, 원본에서 코드로 다시 센다.
- 불일치가 나오면 어느 쪽이 맞는지 원본 행을 열어 확인한다. AI가 맞을 때도 있다.
- 확인하지 못한 것은 미확인으로 남긴다. 비 때문이라는 추정은 추정으로 표시한다.
- 개인정보 없는 공개 자료만 쓰고 출처와 이용허락을 적는다.
- 검산 코드도 AI와 함께 썼으니 한 줄씩 읽고 직접 돌린다. 최종 판단과 공개는 사람 이름으로 한다.

한계

- 8년 전 한 달 자료입니다. 지금의 패턴은 2023년 이후 파일로 다시 돌려야 합니다.
- 급감일이 비 때문인지, 1~3시가 운영 중단인지, 두 공식 자료가 왜 다른지는 확인하지 못했습니다.
- 절약한 시간을 재지 않았고, 이 표로 재배치가 좋아졌다는 성과도 없습니다.

검산 코드 verify.py 전문

data 폴더에 네 파일을 nubija_2018_06.csv, nubija_daily.csv, nubija_month.csv, nubija_terminal.xlsx 이름으로 두고 실행합니다. pandas, numpy, openpyxl이 필요합니다.

```
# -*- coding: utf-8 -*-
"""
AI가 앞서 보고한 누비자 수치를 독립 코드로 다시 계산한다.
입력: data/ 폴더의 공공데이터포털 원본 4개
출력: out/verify_results.json, out/daily_2018_06.csv, out/hourly_2018_06.csv
"""
import json, os, sys
import numpy as np, pandas as pd

ROOT = os.path.dirname(os.path.dirname(os.path.abspath(__file__)))
D = os.path.join(ROOT, 'data'); O = os.path.join(ROOT, 'out'); os.makedirs(O, exist_ok=True)

# AI가 지난 분석에서 보고한 값 — 검산 대상
AI_CLAIMS = {
    '2018년 6월 건별 기록 수': 499525,
    '중복 행': 2,
    '1분 미만 대여': 34170,
    '24시간 초과 대여': 78,
    '시간별 집계 점 수': 720,
    '일일 이용현황 일수': 912,
    '월별 이용현황 개월 수': 174,
    '터미널 수': 372,
    '요일·시각 평균 기준선 MAE': 251,
    '새벽 1~3시 대여 0건': '운영 중단',
    '급감일': '6월 10, 19, 28, 30일',
    'ACF 재상승 주기': '24시간, 168시간',
}

res = {}

# ---------- 건별 이력 ----------
r = pd.read_csv(os.path.join(D, 'nubija_2018_06.csv'), encoding='cp949')
res['원본 행 수'] = int(len(r))
dup_mask = r.duplicated(keep='first')
res['중복 행'] = int(dup_mask.sum())
r = r[~dup_mask].copy()
res['중복 제거 후 행 수'] = int(len(r))

r['start'] = pd.to_datetime(r['출발일'] + ' ' + r['출발시간'], format='%Y.%m.%d %H:%M:%S', errors='coerce')
r['end']   = pd.to_datetime(r['도착일'] + ' ' + r['도착시간'], format='%Y.%m.%d %H:%M:%S', errors='coerce')
res['시각 파싱 실패'] = int(r['start'].isna().sum() + r['end'].isna().sum())
r['dur_min'] = (r['end'] - r['start']).dt.total_seconds() / 60
res['이용시간 음수'] = int((r['dur_min'] < 0).sum())
res['1분 미만 대여'] = int((r['dur_min'] < 1).sum())
res['24시간 초과 대여'] = int((r['dur_min'] > 1440).sum())
res['출발일 범위'] = [str(r['start'].min()), str(r['start'].max())]
res['6월 이외 출발 행'] = int(((r['start'] < '2018-06-01') | (r['start'] >= '2018-07-01')).sum())

clean = r[(r['dur_min'] >= 1) & (r['dur_min'] <= 1440)].copy()
res['정제 후 행 수'] = int(len(clean))

hourly = clean.set_index('start').resample('h').size()
hourly = hourly.reindex(pd.date_range('2018-06-01', '2018-06-30 23:00', freq='h'), fill_value=0)
res['시간별 집계 점 수'] = int(len(hourly))
h = hourly.copy(); h.index.name = 'hour'
h.rename('rentals').to_csv(os.path.join(O, 'hourly_2018_06.csv'))

# 새벽 1~3시 — 시각별 합계
by_hr = clean.groupby(clean['start'].dt.hour).size().reindex(range(24), fill_value=0)
res['시각별 대여 합계'] = {int(k): int(v) for k, v in by_hr.items()}
res['새벽 1~3시 대여 합계'] = int(by_hr.loc[[1, 2, 3]].sum())
res['0시 대여 합계'] = int(by_hr.loc[0]); res['4시 대여 합계'] = int(by_hr.loc[4])
```

```
# 원본 기준으로도 확인 — 정제 때문에 0이 된 것인지 구분
by_hr_raw = r.groupby(r['start'].dt.hour).size().reindex(range(24), fill_value=0)
res['새벽 1~3시 대여 합계 원본'] = int(by_hr_raw.loc[[1, 2, 3]].sum())

# 일별 합계와 급감일
daily = clean.set_index('start').resample('D').size()
daily.index.name = 'date'; daily.rename('rentals').to_csv(os.path.join(O, 'daily_2018_06.csv'))
dd = daily.to_frame('n'); dd['dow'] = dd.index.dayofweek
med = dd.groupby('dow')['n'].transform('median')
dd['ratio'] = dd['n'] / med
res['일별 대여'] = {str(k.date()): int(v) for k, v in daily.items()}
res['같은 요일 중앙값 대비 70% 미만인 날'] = [f"{k.day}일 {v:.0%}" for k, v in dd['ratio'].items() if v < 0.70]
res['급감일 상위 5'] = [f"{k.day}일 {v:.0%}" for k, v in dd['ratio'].nsmallest(5).items()]

# 기준선: 6/1~6/23 학습, 6/24~6/30 검증, 요일·시각 평균으로 예측
train, test = hourly[:'2018-06-23 23:00'], hourly['2018-06-24':]
tf = train.to_frame('n'); tf['dow'] = tf.index.dayofweek; tf['hr'] = tf.index.hour
prof = tf.groupby(['dow', 'hr'])['n'].mean()
pred = pd.Series([prof.loc[(t.dayofweek, t.hour)] for t in test.index], index=test.index)
res['요일·시각 평균 기준선 MAE'] = round(float((test - pred).abs().mean()), 1)
res['기준선 검증 기간 점 수'] = int(len(test))
res['검증 기간 시간당 평균 대여'] = round(float(test.mean()), 1)

# ACF — 시간별 시계열, 지연 1~200
x = hourly.values.astype(float); x = x - x.mean(); n = len(x)
acf = np.array([1.0] + [np.sum(x[:-k] * x[k:]) / np.sum(x * x) for k in range(1, 200)])
peaks = [k for k in range(2, 199) if acf[k] > acf[k - 1] and acf[k] > acf[k + 1] and acf[k] > 0.3]
res['ACF 0.3 이상 국소 최고점 지연'] = peaks
res['ACF 24'] = round(float(acf[24]), 3); res['ACF 168'] = round(float(acf[168]), 3)
res['ACF 12'] = round(float(acf[12]), 3)

# ---------- 일별·월별 집계 자료 ----------
dy = pd.read_csv(os.path.join(D, 'nubija_daily.csv'), encoding='cp949')
res['일일 이용현황 일수'] = int(len(dy)); res['일일 이용현황 범위'] = [str(dy['년월일'].min()), str(dy['년월일'].max())]
res['일일 이용현황 중복 날짜'] = int(dy['년월일'].duplicated().sum())
exp_days = (pd.to_datetime(dy['년월일'].max()) - pd.to_datetime(dy['년월일'].min())).days + 1
res['일일 이용현황 기대 일수'] = int(exp_days)

mo = pd.read_csv(os.path.join(D, 'nubija_month.csv'), encoding='cp949')
res['월별 이용현황 개월 수'] = int(len(mo)); res['월별 이용현황 범위'] = [str(mo['년월'].min()), str(mo['년월'].max())]
m201806 = mo.loc[mo['년월'].astype(str).str.startswith('2018-06'), '이용건수']
res['월별 자료의 2018년 6월 이용건수'] = int(m201806.iloc[0]) if len(m201806) else None
res['건별 자료 원본 행 수와 월별 자료 차이'] = (res['원본 행 수'] - res['월별 자료의 2018년 6월 이용건수']) if len(m201806) else None
res['건별 자료 정제 후와 월별 자료 차이'] = (res['정제 후 행 수'] - res['월별 자료의 2018년 6월 이용건수']) if len(m201806) else None

# ---------- 터미널 ----------
t = pd.read_excel(os.path.join(D, 'nubija_terminal.xlsx'))
res['터미널 수'] = int(len(t)); res['터미널 사용유무 분포'] = {str(k): int(v) for k, v in t['사용유무'].value_counts().items()}
res['터미널 번호 중복'] = int(t['터미널번호'].duplicated().sum())
used_ids = set(clean['출발터미널']).union(set(clean['도착터미널']))
res['2018년 6월 이력에 등장한 터미널 수'] = int(len(used_ids))
res['이력에는 있으나 터미널 목록에 없는 번호 수'] = int(len(used_ids - set(t['터미널번호'])))
res['목록에는 있으나 이력에 없는 터미널 수'] = int(len(set(t['터미널번호']) - used_ids))
cw = t[t['터미널명'].str.contains('창원대', na=False)]
res['이름에 창원대가 들어간 터미널'] = [f"{int(a)} {b}" for a, b in zip(cw['터미널번호'], cw['터미널명'])]

# 터미널별 대여·반납 합계 — 시민용 표 재료
rent_by = clean.groupby('출발터미널').size(); ret_by = clean.groupby('도착터미널').size()
top = rent_by.sort_values(ascending=False).head(10)
res['대여 상위 10 터미널'] = [f"{int(k)} {t.set_index('터미널번호')['터미널명'].get(k, '?')} {int(v)}" for k, v in top.items()]

# ---------- 대조표 ----------
compare = []
def row(name, ai, code):
    compare.append({'항목': name, 'AI 보고값': ai, '코드 재계산': code, '판정': '일치' if str(ai) == str(code) else '불일치'})
row('2018년 6월 건별 기록 수', AI_CLAIMS['2018년 6월 건별 기록 수'], res['원본 행 수'])
row('중복 행', AI_CLAIMS['중복 행'], res['중복 행'])
row('1분 미만 대여', AI_CLAIMS['1분 미만 대여'], res['1분 미만 대여'])
row('24시간 초과 대여', AI_CLAIMS['24시간 초과 대여'], res['24시간 초과 대여'])
row('시간별 집계 점 수', AI_CLAIMS['시간별 집계 점 수'], res['시간별 집계 점 수'])
row('일일 이용현황 일수', AI_CLAIMS['일일 이용현황 일수'], res['일일 이용현황 일수'])
```

```
row('월별 이용현황 개월 수', AI_CLAIMS['월별 이용현황 개월 수'], res['월별 이용현황 개월 수'])
row('터미널 수', AI_CLAIMS['터미널 수'], res['터미널 수'])
row('요일·시각 평균 기준선 MAE', AI_CLAIMS['요일·시각 평균 기준선 MAE'], res['요일·시각 평균 기준선 MAE'])
res['대조표'] = compare

json.dump(res, open(os.path.join(0, 'verify_results.json'), 'w', encoding='utf-8'), ensure_ascii=False, indent=1)
for c in compare: print(f"{c['판정']:4s} | {c['항목']:22s} | AI {c['AI 보고값']} | 코드 {c['코드 재계산']}")
print()
for k in ['원본 행 수', '중복 제거 후 행 수', '정제 후 행 수', '시각 파싱 실패', '이용시간 음수', '출발일 범위', '6월 이외 출발 행',
          '새벽 1~3시 대여 합계', '새벽 1~3시 대여 합계 원본', '0시 대여 합계', '4시 대여 합계',
          '같은 요일 중앙값 대비 70% 미만인 날', '급감일 상위 5', '기준선 검증 기간 점 수', '검증 기간 시간당 평균 대여',
          'ACF 0.3 이상 국소 최고점 지연', 'ACF 12', 'ACF 24', 'ACF 168',
          '일일 이용현황 범위', '일일 이용현황 기대 일수', '일일 이용현황 중복 날짜', '월별 이용현황 범위',
          '월별 자료의 2018년 6월 이용건수', '건별 자료 원본 행 수와 월별 자료 차이', '건별 자료 정제 후와 월별 자료 차이',
          '터미널 사용유무 분포', '터미널 번호 중복', '2018년 6월 이력에 등장한 터미널 수', '이력에는 있으나 터미널 목록에 없는 번호 수',
          '목록에는 있으나 이력에 없는 터미널 수', '이름에 창원대가 들어간 터미널', '대여 상위 10 터미널']:
    print(f"{k}: {res[k]}")
```

작성 2026년 10월 22일 · 도구 Claude, Python 3 pandas · 자료 창원시, 공공데이터포털