```
"""
================================================
  📊 엑셀 파일 요약 분석기
  LG전자 영업마케팅 업무 자동화 도구
  작성: Axel | 버전: 1.0
================================================
사용법:
  python excel_summary.py                          # 현재 폴더의 엑셀 파일 선택
  python excel_summary.py 파일명.xlsx             # 특정 파일 지정
  python excel_summary.py 파일명.xlsx --sheet 2   # 특정 시트 지정 (번호 또는 이름)
"""

import sys
import os
import argparse
import glob

try:
    import pandas as pd
    import openpyxl
except ImportError:
    print("필요한 패키지를 설치합니다...")
    os.system("pip install pandas openpyxl --break-system-packages -q")
    import pandas as pd


def find_excel_files(folder="."):
    """현재 폴더에서 엑셀 파일 목록 반환"""
    files = glob.glob(os.path.join(folder, "*.xlsx")) + \
            glob.glob(os.path.join(folder, "*.xls")) + \
            glob.glob(os.path.join(folder, "*.xlsm"))
    return files


def summarize_sheet(df, sheet_name):
    """단일 시트 요약 분석"""
    print(f"\n{'='*55}")
    print(f"  📋 시트명: {sheet_name}")
    print(f"{'='*55}")

    # 기본 크기 정보
    rows, cols = df.shape
    print(f"\n  📐 크기: {rows:,}행 × {cols}열")
```

```
# 컬럼 정보
print(f"\n  📌 컬럼 목록 ({cols}개):")
for i, col in enumerate(df.columns, 1):
    dtype = df[col].dtype
    non_null = df[col].notna().sum()
    null_cnt = df[col].isna().sum()
    null_str = f"  ⚠️  결측값 {null_cnt:,}개" if null_cnt > 0 else ""
    print(f"     {i:2}. {str(col):<25} [{dtype}]  유효값: {non_null:,}{null_str

# 숫자형 컬럼 통계
numeric_cols = df.select_dtypes(include='number').columns.tolist()
if numeric_cols:
    print(f"\n  📊 숫자형 컬럼 통계 ({len(numeric_cols)}개):")
    print(f"  {'컬럼명':<25} {'합계':>15} {'평균':>12} {'최솟값':>12} {'최댓값':>12}
    print(f"  {'-'*25} {'-'*15} {'-'*12} {'-'*12} {'-'*12}")
    for col in numeric_cols:
        s = df[col].dropna()
        if len(s) > 0:
            print(f"  {str(col):<25} {s.sum():>15,.1f} {s.mean():>12,.1f} {s.

# 날짜형 컬럼
date_cols = df.select_dtypes(include=['datetime64']).columns.tolist()
if date_cols:
    print(f"\n  📅 날짜형 컬럼 ({len(date_cols)}개):")
    for col in date_cols:
        s = df[col].dropna()
        if len(s) > 0:
            print(f"     {col}: {s.min().strftime('%Y-%m-%d')} ~ {s.max().str

# 텍스트형 컬럼의 고유값 (적은 경우만 표시)
text_cols = df.select_dtypes(include='object').columns.tolist()
if text_cols:
    print(f"\n  🔤 텍스트형 컬럼 고유값 현황:")
    for col in text_cols:
        unique_cnt = df[col].nunique()
        if unique_cnt <= 10:
            vals = df[col].dropna().unique().tolist()
            vals_str = ", ".join(str(v) for v in vals[:10])
            print(f"     {str(col):<25} 고유값 {unique_cnt}개 → [{vals_str}]")
        else:
            print(f"     {str(col):<25} 고유값 {unique_cnt:,}개 (다양)")

# 데이터 미리보기 (상위 5행)
print(f"\n  👀 데이터 미리보기 (상위 5행):")
preview = df.head(5).to_string(index=False, max_cols=8)
for line in preview.split('\n'):
    print(f"     {line}")
```

```
    print()


def summarize_excel(filepath, target_sheet=None):
    """엑셀 파일 전체 요약"""
    filename = os.path.basename(filepath)
    filesize = os.path.getsize(filepath) / 1024  # KB

    print(f"\n{'★'*55}")
    print(f"  📂 파일명: {filename}")
    print(f"  💾 파일크기: {filesize:.1f} KB")
    print(f"{'★'*55}")

    # 시트 목록 확인
    xl = pd.ExcelFile(filepath)
    sheet_names = xl.sheet_names
    print(f"\n  📑 전체 시트 수: {len(sheet_names)}개")
    for i, name in enumerate(sheet_names, 1):
        print(f"     {i}. {name}")

    # 분석할 시트 결정
    if target_sheet is None:
        sheets_to_read = sheet_names  # 전체 시트
    else:
        # 숫자 인덱스 또는 시트명으로 지정
        try:
            idx = int(target_sheet) - 1
            sheets_to_read = [sheet_names[idx]]
        except (ValueError, IndexError):
            if target_sheet in sheet_names:
                sheets_to_read = [target_sheet]
            else:
                print(f"\n  ❌ '{target_sheet}' 시트를 찾을 수 없습니다.")
                return

    # 각 시트 분석
    for sheet in sheets_to_read:
        try:
            df = pd.read_excel(filepath, sheet_name=sheet, engine='openpyxl')
            summarize_sheet(df, sheet)
        except Exception as e:
            print(f"\n  ⚠️  '{sheet}' 시트 읽기 오류: {e}")

    print(f"\n{'✅'*27}")
    print(f"  분석 완료!")
    print(f"{'✅'*27}\n")
```

```
def main():
    parser = argparse.ArgumentParser(description="엑셀 파일 요약 분석기")
    parser.add_argument("filepath", nargs="?", help="분석할 엑셀 파일 경로")
    parser.add_argument("--sheet", "-s", help="분석할 시트 (번호 또는 시트명)", default=
    args = parser.parse_args()

    filepath = args.filepath

    # 파일 미지정 시 자동 검색
    if not filepath:
        excel_files = find_excel_files()
        if not excel_files:
            print("\n ❌ 현재 폴더에 엑셀 파일이 없습니다.")
            print("  사용법: python excel_summary.py 파일명.xlsx\n")
            sys.exit(1)
        elif len(excel_files) == 1:
            filepath = excel_files[0]
            print(f"\n ✅ 엑셀 파일 자동 감지: {os.path.basename(filepath)}")
        else:
            print("\n 📁 엑셀 파일 목록:")
            for i, f in enumerate(excel_files, 1):
                print(f"    {i}. {os.path.basename(f)}")
            choice = input("\n  분석할 파일 번호 입력: ").strip()
            try:
                filepath = excel_files[int(choice) - 1]
            except (ValueError, IndexError):
                print("  ❌ 잘못된 선택입니다.")
                sys.exit(1)

    if not os.path.exists(filepath):
        print(f"\n ❌ 파일을 찾을 수 없습니다: {filepath}\n")
        sys.exit(1)

    summarize_excel(filepath, args.sheet)


if __name__ == "__main__":
    main()
```