```
from google.colab import drive
drive.mount('/content/drive')
```

```
Mounted at /content/drive
```

```
import os

# Base project directory on Google Drive
BASE_DIR = "/content/drive/MyDrive/SkinAI_Project"

# Data dirs
DATA_DIR       = f"{BASE_DIR}/data"
RAW_DIR        = f"{DATA_DIR}/raw"
PROCESSED_DIR  = f"{DATA_DIR}/processed"
SYNTHETIC_DIR  = f"{DATA_DIR}/synthetic"

MODELS_DIR     = f"{BASE_DIR}/models"
NOTEBOOKS_DIR  = f"{BASE_DIR}/notebooks"

print("BASE_DIR      :", BASE_DIR)
print("SYNTHETIC_DIR :", SYNTHETIC_DIR)
print("\nSynthetic files:", os.listdir(SYNTHETIC_DIR))
```

```
BASE_DIR      : /content/drive/MyDrive/SkinAI_Project
SYNTHETIC_DIR : /content/drive/MyDrive/SkinAI_Project/data/synthetic

Synthetic files: ['clinical_cases_10000.csv', 'patient_profiles_10000.csv', 'treatment_records_10000.csv', 'disease_master.csv',
```

```
import pandas as pd
import os

clinical_path   = os.path.join(SYNTHETIC_DIR, "clinical_cases_10000.csv")
profiles_path   = os.path.join(SYNTHETIC_DIR, "patient_profiles_10000.csv")
treatments_path = os.path.join(SYNTHETIC_DIR, "treatment_records_10000.csv")

print("clinical_path  :", clinical_path,   "| exists:", os.path.exists(clinical_path))
print("profiles_path  :", profiles_path,   "| exists:", os.path.exists(profiles_path))
print("treatments_path:", treatments_path, "| exists:", os.path.exists(treatments_path))

df_clinical = pd.read_csv(clinical_path)
print("\nClinical cases shape:", df_clinical.shape)
print("Clinical columns    :", df_clinical.columns.tolist())

# Optional: load extra synthetic tables
if os.path.exists(profiles_path):
    df_profiles = pd.read_csv(profiles_path)
    print("\nPatient profiles shape:", df_profiles.shape)
    print("Profiles columns      :", df_profiles.columns.tolist())
else:
    df_profiles = None

if os.path.exists(treatments_path):
    df_treat_records = pd.read_csv(treatments_path)
    print("\nTreatment records shape:", df_treat_records.shape)
    print("Treatment rec columns  :", df_treat_records.columns.tolist())
else:
    df_treat_records = None
```

```
clinical_path  : /content/drive/MyDrive/SkinAI_Project/data/synthetic/clinical_cases_10000.csv | exists: True
profiles_path  : /content/drive/MyDrive/SkinAI_Project/data/synthetic/patient_profiles_10000.csv | exists: True
treatments_path: /content/drive/MyDrive/SkinAI_Project/data/synthetic/treatment_records_10000.csv | exists: True

Clinical cases shape: (10000, 20)
Clinical columns    : ['case_id', 'patient_id', 'age', 'gender', 'region', 'skin_type', 'fitzpatrick_type', 'primary_disease', '

Patient profiles shape: (10000, 9)
Profiles columns      : ['patient_id', 'age', 'gender', 'region', 'skin_type', 'fitzpatrick_type', 'allergies', 'comorbidities',

Treatment records shape: (10000, 9)
Treatment rec columns  : ['case_id', 'patient_id', 'primary_disease', 'disease_category', 'severity', 'recommended_medicine', 'i
```

```
# Normalize the treatment_outcome column
df_clinical["treatment_outcome_norm"] = (
    df_clinical["treatment_outcome"]
    .astype(str)
    .str.lower()
    .str.strip()
)

df_good = df_clinical[df_clinical["treatment_outcome_norm"] == "improved"].copy()
print("Rows with 'improved' outcome:", len(df_good))

if len(df_good) == 0:
    print("[WARN] No 'improved' rows found. Using all rows instead.")
    df_good = df_clinical.copy()

# Ensure required columns exist
required_cols = ["primary_disease", "severity", "recommended_medicine", "is_over_the_counter"]
for col in required_cols:
    if col not in df_good.columns:
        raise KeyError(f"Required column '{col}' missing in clinical dataset.")

# Group to calculate how often each treatment was successful
grouped = (
    df_good
    .groupby(["primary_disease", "severity", "recommended_medicine", "is_over_the_counter"], as_index=False)
    .size()
    .rename(columns={"size": "count"})
)

print("\nGrouped treatment rows:", len(grouped))
grouped.head()
```

```
Rows with 'improved' outcome: 3351

Grouped treatment rows: 174
```

	primary_disease	severity	recommended_medicine	is_over_the_counter	count
0	Acne rosacea	mild	metronidazole gel	yes	31
1	Acne rosacea	mild	topical ivermectin	yes	34
2	Acne rosacea	moderate	metronidazole gel	yes	16
3	Acne rosacea	moderate	topical ivermectin	yes	18
4	Acne rosacea	severe	metronidazole gel	yes	16

```
AVAILABLE_DISEASES = sorted(grouped["primary_disease"].unique())
print("Number of diseases in treatment data:", len(AVAILABLE_DISEASES))
print("First 30 diseases:")
for d in AVAILABLE_DISEASES[:30]:
    print(" -", d)
```

```
Number of diseases in treatment data: 29
First 30 diseases:
 - Acne rosacea
 - Acne vulgaris
 - Atopic dermatitis (Eczema)
 - Cellulitis
 - Chronic urticaria
 - Contact dermatitis (Allergic)
 - Contact dermatitis (Irritant)
 - Dyshidrotic eczema
 - Folliculitis
 - Herpes simplex (Cold sores)
 - Herpes zoster (Shingles)
 - Impetigo
 - Lichen planus
 - Molluscum contagiosum
 - Nummular eczema
 - Perioral dermatitis
 - Photodermatitis
 - Pityriasis versicolor
 - Psoriasis vulgaris
 - Rosacea
 - Scabies
 - Seborrheic dermatitis
 - Tinea capitis (Scalp ringworm)
```

```
- Tinea corporis (Ringworm)
- Tinea cruris (Jock itch)
- Tinea pedis (Athlete's foot)
- Urticaria (Hives)
- Vitiligo
- Warts (Verruca vulgaris)
```

```
def get_top_treatments_for(disease: str, severity: str = None, top_k: int = 5) -> pd.DataFrame:
    """
    Return top treatment rows for a given disease and optional severity from 'grouped'.
    """
    df = grouped[grouped["primary_disease"].str.lower() == disease.lower()]

    if df.empty:
        print(f"[get_top_treatments_for] No data for disease='{disease}'.")
        print("Available diseases (first 20):", AVAILABLE_DISEASES[:20])
        return pd.DataFrame()

    if severity is not None:
        df_sev = df[df["severity"].astype(str).str.lower() == severity.lower()]
        if not df_sev.empty:
            df = df_sev
        else:
            print(f"[get_top_treatments_for] No severity='{severity}' for disease='{disease}', using all severities.")

    return df.sort_values("count", ascending=False).head(top_k)


def recommend_treatment(
    disease: str,
    severity: str = None,
    age: int = None,
    allergies: list = None,
    prefer_otc: bool = True,
    top_k: int = 3
):
    """
    Recommend treatments based on synthetic clinical data.
    - disease: primary_disease name (should match or be mapped).
    - severity: e.g. 'mild', 'moderate', 'severe' (optional).
    - age: for future extensions (children/elderly filtering).
    - allergies: list of strings (e.g. ['penicillin', 'aspirin']).
    - prefer_otc: if True, prefer OTC medicines if available.
    - top_k: number of treatment suggestions to return.
    """
    if allergies is None:
        allergies = []

    # Step 1: Get candidates for disease & severity
    candidates = get_top_treatments_for(disease, severity, top_k=100).copy()

    if candidates.empty:
        print(f"[recommend_treatment] No candidates for disease='{disease}'.")
        return []

    # Step 2: Filter by allergy
    def is_safe(row):
        med = str(row["recommended_medicine"]).lower()
        return not any(a.lower() in med for a in allergies)

    candidates["is_safe"] = candidates.apply(is_safe, axis=1)
    safe_candidates = candidates[candidates["is_safe"]]

    if safe_candidates.empty:
        print("[recommend_treatment] No safe meds after allergy filtering; using all candidates.")
        safe_candidates = candidates

    # Step 3: Prefer OTC if requested
    if prefer_otc:
        otc = safe_candidates[safe_candidates["is_over_the_counter"] == True]
        if len(otc) > 0:
            safe_candidates = otc

    # Step 4: Sort & select top_k
    result = (
        safe_candidates
        .sort_values("count", ascending=False)
```

```
        .head(top_k)
        .drop(columns=["is_safe"])
        .to_dict(orient="records")
    )

    return result
```

```
def print_treatment_suggestions(
    disease: str,
    severity: str = None,
    age: int = None,
    allergies: list = None,
    prefer_otc: bool = True,
    top_k: int = 3
):
    recs = recommend_treatment(
        disease=disease,
        severity=severity,
        age=age,
        allergies=allergies,
        prefer_otc=prefer_otc,
        top_k=top_k
    )

    print(f"\n=== Treatment Suggestions for: {disease} (severity={severity}) ===")
    print(f"Age      : {age}")
    print(f"Allergies: {allergies}")
    print(f"Prefer OTC: {prefer_otc}")
    print("")

    if not recs:
        print("No treatments found in synthetic dataset for this query.")
        return

    for r in recs:
        med = r["recommended_medicine"]
        otc = "yes" if r["is_over_the_counter"] else "no"
        count = r["count"]
        sev = r["severity"]
        print(f"- {med} | severity={sev} | OTC={otc} | support_count={count}")
```

```
print("Example available diseases (first 20):")
for d in AVAILABLE_DISEASES[:20]:
    print(" -", d)

# Example 1
print_treatment_suggestions(
    disease="Acne vulgaris",
    severity="moderate",
    age=22,
    allergies=["benzoyl peroxide"],   # example
    prefer_otc=True,
    top_k=3
)

# Example 2
print_treatment_suggestions(
    disease="Atopic dermatitis (Eczema)",
    severity="mild",
    age=5,
    allergies=[],
    prefer_otc=False,
    top_k=3
)
```

```
Example available diseases (first 20):
 - Acne rosacea
 - Acne vulgaris
 - Atopic dermatitis (Eczema)
 - Cellulitis
 - Chronic urticaria
 - Contact dermatitis (Allergic)
 - Contact dermatitis (Irritant)
```

```
 - Dyshidrotic eczema
 - Folliculitis
 - Herpes simplex (Cold sores)
 - Herpes zoster (Shingles)
 - Impetigo
 - Lichen planus
 - Molluscum contagiosum
 - Nummular eczema
 - Perioral dermatitis
 - Photodermatitis
 - Pityriasis versicolor
 - Psoriasis vulgaris
 - Rosacea

=== Treatment Suggestions for: Acne vulgaris (severity=moderate) ===
Age      : 22
Allergies: ['benzoyl peroxide']
Prefer OTC: True

- topical retinoid | severity=moderate | OTC=yes | support_count=11
- salicylic acid wash | severity=moderate | OTC=yes | support_count=10

=== Treatment Suggestions for: Atopic dermatitis (Eczema) (severity=mild) ===
Age      : 5
Allergies: []
Prefer OTC: False

- emollient cream | severity=mild | OTC=yes | support_count=23
- topical steroid (low potency) | severity=mild | OTC=yes | support_count=18
- fragrance-free moisturizer | severity=mild | OTC=yes | support_count=15
```

```
grouped_path = os.path.join(SYNTHETIC_DIR, "treatment_stats_grouped.csv")
grouped.to_csv(grouped_path, index=False)
print("Saved treatment stats to:", grouped_path)
```

```
Saved treatment stats to: /content/drive/MyDrive/SkinAI_Project/data/synthetic/treatment_stats_grouped.csv
```