

```

r"""
Load OpenFlights reference data (airports + airlines) into the airline schema.

```

Assumptions (matching your ERD):

```

- airline.airports(
    airport_id serial PK,
    iata_code varchar(3) UNIQUE,
    icao_code varchar(4) UNIQUE,
    name text,
    city text,
    country text,
    latitude double precision,
    longitude double precision,
    timezone text
)

- airline.airlines(
    airline_id serial PK,
    name text,
    iata_code varchar(3),
    icao_code varchar(3),
    country varchar(3)
)

```

We:

```

- Read the raw OpenFlights CSVs (no header, fixed column layout).
- Clean values and truncate anything that might violate length constraints.
- Skip obviously bad / placeholder values (e.g., "\N").
"""

```

```

import os
from pathlib import Path

import pandas as pd
from sqlalchemy import create_engine, text

```

```

# -----
# DB URL helper
# -----

```

```

def get_db_url() -> str:
    """
    Look up the database URL from .env / shell.

    Prefers DATABASE_URL, falls back to AIRLINE_DB_DSN.
    """
    url = os.environ.get("DATABASE_URL") or os.environ.get("AIRLINE_DB_DSN")
    if not url:
        raise RuntimeError(
            "Set either DATABASE_URL or AIRLINE_DB_DSN in your environment / .env.\n"
            "Example: postgres://user:password@localhost:5432/airline_bi"
        )
    return url

```

```

ENGINE = create_engine(get_db_url(), future=True, pool_pre_ping=True)

```

```

# -----
# Paths
# -----

```

```

PROJECT_ROOT = Path(__file__).resolve().parents[1]

```

```
DATA_DIR = PROJECT_ROOT / "data"
```

```
AIRPORTS_CSV = DATA_DIR / "openflights_airports.csv"
```

```
AIRLINES_CSV = DATA_DIR / "openflights_airlines.csv"
```

```
# -----
# Helpers
# -----
```

```
def _clean_str(value):
    """Return a stripped string or None for NaN / placeholders."""
    if pd.isna(value):
        return None
    s = str(value).strip()
    if not s or s == r"\N":
        return None
    return s
```

```
# -----
# Airports
# -----
```

```
def load_airports() -> None:
    """
    Load airports from the standard OpenFlights airports.dat layout:
```

```
0: Airport ID
1: Name
2: City
3: Country
4: IATA
5: ICAO
6: Latitude
7: Longitude
8: Altitude
9: Timezone (hours from UTC)
10: DST
11: Tz database time zone
12: type
13: source
"""
```

```
print(f"💎 Loading OpenFlights airports from: {AIRPORTS_CSV}")
```

```
df = pd.read_csv(AIRPORTS_CSV, header=None, dtype=str)
```

```
rows = []
for _, row in df.iterrows():
    name = _clean_str(row[1])
    city = _clean_str(row[2])
    country = _clean_str(row[3])
    iata = _clean_str(row[4])
    icao = _clean_str(row[5])
```

```
# Skip unusable rows
if not iata and not icao:
    continue
```

```
if iata:
    iata = iata[:3].upper()
if icao:
    icao = icao[:4].upper()
```

```
# Latitude & Longitude
```

```

    try:
        lat = float(row[6]) if row[6] not in (None, "", r"\N") else None
    except:
        lat = None
    try:
        lon = float(row[7]) if row[7] not in (None, "", r"\N") else None
    except:
        lon = None

    tz = _clean_str(row[11]) or _clean_str(row[9])

    rows.append(
        dict(
            iata=iata,
            icao=icao,
            name=name,
            city=city,
            country=country,
            lat=lat,
            lon=lon,
            tz=tz,
        )
    )

if not rows:
    print("⚠ No airport rows to insert (after filtering).")
    return

with ENGINE.begin() as con:
    con.execute(
        text(
            """
            INSERT INTO airline.airports (
                iata_code, icao_code, name, city, country,
                latitude, longitude, timezone
            )
            VALUES (
                :iata, :icao, :name, :city, :country,
                :lat, :lon, :tz
            )
            ON CONFLICT (iata_code) DO NOTHING;
            """
        ),
        rows,
    )

print(f"✅ Airports loaded: {len(rows)} candidate rows inserted (conflicts skipped).")

```

```

# -----
# Airlines
# -----

```

```

def load_airlines() -> None:
    """
    Load airlines from the standard OpenFlights airlines.dat layout:

    0: Airline ID
    1: Name
    2: Alias
    3: IATA
    4: ICAO
    5: Callsign
    6: Country
    """

```

7: Active
 """

print(f"💠 Loading OpenFlights airlines from: {AIRLINES_CSV}")

df = pd.read_csv(AIRLINES_CSV, header=None, dtype=str)

```
rows = []
for _, row in df.iterrows():
    name = _clean_str(row[1])
    if not name:
        continue

    iata = _clean_str(row[3])
    icao = _clean_str(row[4])
    country_full = _clean_str(row[6])

    # Truncate to schema limits
    if iata:
        iata = iata[:3].upper()
    if icao:
        icao = icao[:3].upper()
    country = country_full[:3].upper() if country_full else None

    rows.append(
        dict(
            name=name,
            iata=iata,
            icao=icao,
            country=country,
        )
    )
```

if not rows:
 print("⚠️ No airline rows to insert (after filtering).")
 return

```
with ENGINE.begin() as con:
    con.execute(
        text(
            """
            INSERT INTO airline.airlines (
                name, iata_code, icao_code, country
            )
            VALUES (:name, :iata, :icao, :country)
            ON CONFLICT DO NOTHING;
            """
        ),
        rows,
    )
```

print(f"✅ Airlines loaded: {len(rows)} candidate rows inserted (conflicts skipped).")

 # Entrypoint
 # -----

```
def run() -> None:
    load_airports()
    load_airlines()
    print("🎉 OpenFlights reference tables loaded.")
```

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