CoffeeSalesSyntheticDataPreparation

August 13, 2025

[1]:
```
import pandas as pd
import numpy as np
import random

# Set random seed for reproducibility
np.random.seed(42)

# Parameters
dates = pd.date_range(start="2024-01-01", end="2024-03-31", freq="D")
locations = ["Tbilisi", "Batumi", "Kutaisi", "Rustavi"]
products = ["Espresso", "Latte", "Cappuccino", "Americano", "Mocha", "Cold␣
↪Brew"]
campaigns = ["Facebook Ads", "Google Ads", "Instagram Ads", "Billboard",␣
↪"Flyers"]

# Generate sales data
sales_data = []
for date in dates:
    for loc in locations:
        for prod in products:
            units_sold = np.random.poisson(lam=10)
            price = random.choice([2.5, 3.0, 3.5, 4.0])
            revenue = round(units_sold * price, 2)
            sales_data.append([date, loc, prod, units_sold, price, revenue])

sales_df = pd.DataFrame(sales_data, columns=["Date", "Location", "Product",␣
↪"Units_Sold", "Unit_Price", "Revenue"])

# Generate marketing data
ads_data = []
for date in dates:
    for camp in campaigns:
        clicks = np.random.poisson(lam=20)
        impressions = clicks * random.randint(20, 50)
        spend = round(clicks * random.uniform(0.1, 0.5), 2)
        ctr = round(clicks / impressions, 4) if impressions > 0 else 0
        cpc = round(spend / clicks, 2) if clicks > 0 else 0
        ads_data.append([date, camp, clicks, impressions, spend, ctr, cpc])

ads_df = pd.DataFrame(ads_data, columns=["Date", "Campaign", "Clicks",␣
↪"Impressions", "Spend", "CTR", "CPC"])

# Save datasets
coffee_sales_path = "coffee_sales_data.csv"
coffee_ads_path = "coffee_ads_data.csv"

sales_df.to_csv(coffee_sales_path, index=False)
ads_df.to_csv(coffee_ads_path, index=False)
```

[2]:
```
# ---------- 1) Load ----------
sales_df = pd.read_csv(coffee_sales_path)
ads_df = pd.read_csv(coffee_ads_path)
```

[2]:
```
(             Date Location     Product  Units_Sold  Unit_Price  Revenue
 2179  2024-03-31  Rustavi       Latte          11         2.5     27.5
 2180  2024-03-31  Rustavi  Cappuccino          13         4.0     52.0
 2181  2024-03-31  Rustavi   Americano           4         3.0     12.0
 2182  2024-03-31  Rustavi       Mocha           6         4.0     24.0
 2183  2024-03-31  Rustavi   Cold Brew          16         3.5     56.0,
            Date       Campaign  Clicks  Impressions  Spend     CTR   CPC
 450  2024-03-31   Facebook Ads      17          561   3.29  0.0303  0.19
 451  2024-03-31     Google Ads      20         1000   4.11  0.0200  0.21
 452  2024-03-31  Instagram Ads      22          858   8.09  0.0256  0.37
 453  2024-03-31      Billboard      21          945   4.05  0.0222  0.19
 454  2024-03-31         Flyers      14          336   6.63  0.0417  0.47)
```

[3]:
```
# ---------- 2) Quick inspect ----------
print("\n=== Raw shapes ===")
print("sales_df:", sales_df.shape, "ads_df:", ads_df.shape)

print("\n=== Sales head ===")
print(sales_df.head())

print("\n=== Ads head ===")
print(ads_df.head())

print("\n=== Dtypes (before) ===")
print("sales_df:\n", sales_df.dtypes)
print("ads_df:\n", ads_df.dtypes)
```

```
=== Raw shapes ===
sales_df: (2184, 6) ads_df: (455, 7)

=== Sales head ===
         Date Location     Product  Units_Sold  Unit_Price  Revenue
0  2024-01-01  Tbilisi    Espresso          12         3.0     36.0
1  2024-01-01  Tbilisi       Latte           6         4.0     24.0
2  2024-01-01  Tbilisi  Cappuccino          11         4.0     44.0
3  2024-01-01  Tbilisi   Americano          14         4.0     56.0
4  2024-01-01  Tbilisi       Mocha           7         3.5     24.5

=== Ads head ===
         Date       Campaign  Clicks  Impressions  Spend     CTR   CPC
0  2024-01-01   Facebook Ads      25          875   7.69  0.0286  0.31
1  2024-01-01     Google Ads      23          966   4.18  0.0238  0.18
2  2024-01-01  Instagram Ads      15          330   6.48  0.0455  0.43
3  2024-01-01       Billboard      22         1034   9.50  0.0213  0.43
4  2024-01-01          Flyers      18          882   3.19  0.0204  0.18

=== Dtypes (before) ===
sales_df:
 Date           object
Location       object
Product        object
Units_Sold      int64
Unit_Price    float64
Revenue       float64
dtype: object
ads_df:
 Date            object
Campaign        object
Clicks           int64
Impressions      int64
Spend          float64
CTR            float64
CPC            float64
dtype: object
```

[4]:
```
# ---------- 3) Basic cleaning ----------
# normalize column names
sales_df.columns = [c.strip().replace(" ", "_") for c in sales_df.columns]
ads_df.columns   = [c.strip().replace(" ", "_") for c in ads_df.columns]

# parse dates
for df in (sales_df, ads_df):
    if "Date" in df.columns:
        df["Date"] = pd.to_datetime(df["Date"], errors="coerce")

# coerce numerics
num_cols_sales = ["Units_Sold", "Unit_Price", "Revenue"]
for c in num_cols_sales:
    if c in sales_df.columns:
        sales_df[c] = pd.to_numeric(sales_df[c], errors="coerce")

num_cols_ads = ["Clicks", "Impressions", "Spend", "CTR", "CPC"]
for c in num_cols_ads:
    if c in ads_df.columns:
        ads_df[c] = pd.to_numeric(ads_df[c], errors="coerce")
```

[5]:
```
# ---------- 4) Nulls & basic fixes ----------
print("\n=== Null counts (after coercion) ===")
print("sales_df:\n", sales_df.isna().sum())
print("ads_df:\n", ads_df.isna().sum())

# Drop rows with missing Date
sales_df = sales_df.dropna(subset=["Date"])
ads_df   = ads_df.dropna(subset=["Date"])

# Negative or impossible values -> fix or drop
if "Units_Sold" in sales_df:
    sales_df["Units_Sold"] = sales_df["Units_Sold"].clip(lower=0)

if "Unit_Price" in sales_df:
    sales_df["Unit_Price"] = sales_df["Unit_Price"].clip(lower=0)

if "Revenue" in sales_df:
    missing_rev = sales_df["Revenue"].isna()
    can_compute = missing_rev & sales_df["Units_Sold"].notna() &␣
↪sales_df["Unit_Price"].notna()
    sales_df.loc[can_compute, "Revenue"] = (sales_df.loc[can_compute,␣
↪"Units_Sold"] *
                                            sales_df.loc[can_compute,␣
↪"Unit_Price"]).round(2)
    sales_df["Revenue"] = sales_df["Revenue"].fillna(0).clip(lower=0)

for c in ["Clicks", "Impressions", "Spend"]:
    if c in ads_df:
        ads_df[c] = ads_df[c].fillna(0).clip(lower=0)

# Recompute CTR/CPC if needed
if set(["Clicks", "Impressions"]).issubset(ads_df.columns):
    recompute_ctr_mask = ads_df["CTR"].isna() if "CTR" in ads_df.columns else␣
↪pd.Series(True, index=ads_df.index)
    ads_df.loc[recompute_ctr_mask, "CTR"] = np.where(ads_df["Impressions"] > 0,
                                                     ads_df["Clicks"] /␣
↪ads_df["Impressions"], 0)

if set(["Spend", "Clicks"]).issubset(ads_df.columns):
    recompute_cpc_mask = ads_df["CPC"].isna() if "CPC" in ads_df.columns else␣
↪pd.Series(True, index=ads_df.index)
    ads_df.loc[recompute_cpc_mask, "CPC"] = np.where(ads_df["Clicks"] > 0,
                                                     ads_df["Spend"] /␣
↪ads_df["Clicks"], 0)
```

```
=== Null counts (after coercion) ===
sales_df:
 Date          0
Location      0
Product       0
Units_Sold    0
Unit_Price    0
Revenue       0
dtype: int64
ads_df:
 Date           0
Campaign       0
Clicks         0
Impressions    0
Spend          0
CTR            0
CPC            0
dtype: int64
```

[6]:
```
# ---------- 5) Alignment checks: what can we join on? ----------

# 1) What columns overlap?
common_cols = set(sales_df.columns) & set(ads_df.columns)
print("\n=== Common columns between sales_df and ads_df ===")
print(common_cols)

# 2) Quick cardinality overview
print("\n=== Uniques overview ===")
if {"Location", "Product"}.issubset(sales_df.columns):
    print("sales_df uniques -> Locations:", sales_df["Location"].nunique(),
          "| Products:", sales_df["Product"].nunique())
if "Campaign" in ads_df.columns:
    print("ads_df uniques -> Campaigns:", ads_df["Campaign"].nunique())

# 3) Date coverage comparison
sales_dates = set(sales_df["Date"].dropna().unique())
ads_dates   = set(ads_df["Date"].dropna().unique())

only_sales_dates = sales_dates - ads_dates
only_ads_dates   = ads_dates - sales_dates
```

```python
both_dates       = sales_dates & ads_dates

print("\n=== Date coverage ===")
print("Sales dates:", len(sales_dates),
      "| Ads dates:", len(ads_dates),
      "| Overlap:", len(both_dates))
print("Dates only in sales_df:", len(only_sales_dates))
print("Dates only in ads_df:", len(only_ads_dates))

# 4) Per-day totals and where data is missing on one side
sales_by_day = (sales_df.groupby("Date", as_index=False)
                .agg(Daily_Revenue=("Revenue", "sum"),
                     Units_Sold=("Units_Sold", "sum")))
ads_by_day = (ads_df.groupby("Date", as_index=False)
              .agg(Ad_Spend=("Spend", "sum"),
                   Ad_Clicks=("Clicks", "sum"),
                   Ad_Impressions=("Impressions", "sum")))

day_merge = sales_by_day.merge(ads_by_day, on="Date", how="outer")

# Flags: which days have which data?
day_merge["has_sales"] = day_merge["Daily_Revenue"].fillna(0) > 0
day_merge["has_ads"]   = day_merge["Ad_Spend"].fillna(0) > 0
day_merge["where"] = np.select(
    [ day_merge["has_sales"] & day_merge["has_ads"],
      day_merge["has_sales"] & ~day_merge["has_ads"],
      ~day_merge["has_sales"] & day_merge["has_ads"]],
    ["both", "sales_only", "ads_only"],
    default="neither"
)

print("\n=== Day-level availability ===")
print(day_merge["where"].value_counts())

print("\nExamples of days with sales but no ads:")
print(day_merge.loc[day_merge["where"]=="sales_only"].head(5))

print("\nExamples of days with ads but no sales:")
print(day_merge.loc[day_merge["where"]=="ads_only"].head(5))

# 5) Sanity: do we have any negative or impossible values left?
def bad_counts(df, cols):
    out = {}
    for c in cols:
        if c in df.columns:
            out[c] = int((df[c] < 0).sum())
    return out
```

```python
print("\n=== Negative-value checks ===")
print("sales_df:", bad_counts(sales_df, ["Units_Sold", "Unit_Price",␣
↪"Revenue"]))
print("ads_df:", bad_counts(ads_df, ["Clicks", "Impressions", "Spend"]))

# 6) Optional: show a tiny summary of per-campaign completeness (ads side)
if "Campaign" in ads_df.columns:
    camp_days = (ads_df.groupby("Campaign")["Date"]
                 .nunique()
                 .sort_values(ascending=False)
                 .head(10))
    print("\nTop campaigns by active days:")
    print(camp_days)
```

```
=== Common columns between sales_df and ads_df ===
{'Date'}

=== Uniques overview ===
sales_df uniques -> Locations: 4 | Products: 6
ads_df uniques -> Campaigns: 5

=== Date coverage ===
Sales dates: 91 | Ads dates: 91 | Overlap: 91
Dates only in sales_df: 0
Dates only in ads_df: 0

=== Day-level availability ===
where
both    91
Name: count, dtype: int64

Examples of days with sales but no ads:
Empty DataFrame
Columns: [Date, Daily_Revenue, Units_Sold, Ad_Spend, Ad_Clicks, Ad_Impressions,
has_sales, has_ads, where]
Index: []

Examples of days with ads but no sales:
Empty DataFrame
Columns: [Date, Daily_Revenue, Units_Sold, Ad_Spend, Ad_Clicks, Ad_Impressions,
has_sales, has_ads, where]
Index: []

=== Negative-value checks ===
sales_df: {'Units_Sold': 0, 'Unit_Price': 0, 'Revenue': 0}
ads_df: {'Clicks': 0, 'Impressions': 0, 'Spend': 0}
```

```
Top campaigns by active days:
Campaign
Billboard        91
Facebook Ads     91
Flyers           91
Google Ads       91
Instagram Ads    91
Name: Date, dtype: int64
```

[7]:
```
# ---------- 6) Aggregate daily ----------
sales_daily = (sales_df
               .groupby("Date", as_index=False)
               .agg(Daily_Revenue=("Revenue", "sum"),
                    Units_Sold=("Units_Sold", "sum"),
                    Line_Items=("Revenue", "size")))

sales_daily["AOV_proxy"] = np.where(sales_daily["Line_Items"] > 0,
                                    sales_daily["Daily_Revenue"] /␣
 ↪sales_daily["Line_Items"], 0)

ads_daily = (ads_df
             .groupby("Date", as_index=False)
             .agg(Ad_Clicks=("Clicks", "sum"),
                  Ad_Impressions=("Impressions", "sum"),
                  Ad_Spend=("Spend", "sum")))

ads_daily["CTR"] = np.where(ads_daily["Ad_Impressions"] > 0,
                            ads_daily["Ad_Clicks"] /␣
 ↪ads_daily["Ad_Impressions"], 0)
ads_daily["CPC"] = np.where(ads_daily["Ad_Clicks"] > 0,
                            ads_daily["Ad_Spend"] / ads_daily["Ad_Clicks"], 0)
ads_daily["CPM"] = np.where(ads_daily["Ad_Impressions"] > 0,
                            (ads_daily["Ad_Spend"] /␣
 ↪ads_daily["Ad_Impressions"]) * 1000, 0)

print("\n=== sales_daily preview ===")
print(sales_daily.head())
print("\n=== ads_daily preview ===")
print(ads_daily.head())
```

```
=== sales_daily preview ===
        Date  Daily_Revenue  Units_Sold  Line_Items  AOV_proxy
0 2024-01-01          748.5         214          24  31.187500
1 2024-01-02          851.5         251          24  35.479167
2 2024-01-03          829.5         254          24  34.562500
3 2024-01-04          826.0         243          24  34.416667
```

```
4 2024-01-05          816.0         247          24  34.000000

=== ads_daily preview ===
        Date  Ad_Clicks  Ad_Impressions  Ad_Spend       CTR       CPC  \
0 2024-01-01        103            4087     31.04  0.025202  0.301359
1 2024-01-02        115            4579     31.12  0.025115  0.270609
2 2024-01-03         91            2941     30.53  0.030942  0.335495
3 2024-01-04        104            4019     25.68  0.025877  0.246923
4 2024-01-05        102            3992     33.87  0.025551  0.332059

         CPM
0   7.594813
1   6.796244
2  10.380823
3   6.389649
4   8.484469
```

[8]:

```
# ---------- 7) Merge & KPIs ----------
daily = pd.merge(sales_daily, ads_daily, on="Date", how="left").fillna(0)

daily["ROAS"] = np.where(daily["Ad_Spend"] > 0,
                         daily["Daily_Revenue"] / daily["Ad_Spend"], np.nan)

daily["ROI"] = np.where(daily["Ad_Spend"] > 0,
                        (daily["Daily_Revenue"] - daily["Ad_Spend"]) /␣
 ↪daily["Ad_Spend"], np.nan)

daily["ConvRate_proxy"] = np.where(daily["Ad_Clicks"] > 0,
                                   daily["Line_Items"] / daily["Ad_Clicks"], np.
 ↪nan)

print("\n=== daily merged preview ===")
print(daily.head())

print("\n=== Summary ranges ===")
print(daily.describe(include="all").T)
```

```
=== daily merged preview ===
        Date  Daily_Revenue  Units_Sold  Line_Items  AOV_proxy  Ad_Clicks  \
0 2024-01-01          748.5         214          24  31.187500        103
1 2024-01-02          851.5         251          24  35.479167        115
2 2024-01-03          829.5         254          24  34.562500         91
3 2024-01-04          826.0         243          24  34.416667        104
4 2024-01-05          816.0         247          24  34.000000        102

   Ad_Impressions  Ad_Spend       CTR       CPC        CPM       ROAS  \
0            4087     31.04  0.025202  0.301359   7.594813  24.114046
```

```
1           4579     31.12  0.025115  0.270609   6.796244  27.361825
2           2941     30.53  0.030942  0.335495  10.380823  27.169997
3           4019     25.68  0.025877  0.246923   6.389649  32.165109
4           3992     33.87  0.025551  0.332059   8.484469  24.092117

         ROI  ConvRate_proxy
0  23.114046        0.233010
1  26.361825        0.208696
2  26.169997        0.263736
3  31.165109        0.230769
4  23.092117        0.235294

=== Summary ranges ===
               count                 mean                  min  \
Date              91  2024-02-15 00:00:00  2024-01-01 00:00:00
Daily_Revenue   91.0           781.708791                670.0
Units_Sold      91.0           239.043956                194.0
Line_Items      91.0                 24.0                 24.0
AOV_proxy       91.0              32.5712            27.916667
Ad_Clicks       91.0           100.065934                 79.0
Ad_Impressions  91.0          3529.241758               2534.0
Ad_Spend        91.0            29.274945                16.23
CTR             91.0             0.028642             0.023207
CPC             91.0             0.292115             0.175567
CPM             91.0             8.343425             4.508869
ROAS            91.0            27.864668            17.538393
ROI             91.0            26.864668            16.538393
ConvRate_proxy  91.0             0.241985                  0.2

                                25%                  50%                  75%  \
Date            2024-01-23 12:00:00  2024-02-15 00:00:00  2024-03-08 12:00:00
Daily_Revenue                734.25                779.5                825.5
Units_Sold                    226.0                239.0                251.0
Line_Items                     24.0                 24.0                 24.0
AOV_proxy                  30.59375            32.479167            34.395833
Ad_Clicks                      93.5                101.0                106.0
Ad_Impressions               3195.0               3507.0               3837.0
Ad_Spend                     25.175                30.17                32.71
CTR                        0.026192             0.028348              0.03065
CPC                         0.25889             0.286847              0.32543
CPM                        7.221094             8.331432              9.27033
ROAS                      23.596522            26.131814            32.325659
ROI                       22.596522            25.131814            31.325659
ConvRate_proxy             0.226415             0.237624             0.256692

                                max        std
Date            2024-03-31 00:00:00        NaN
Daily_Revenue                 918.5  58.753726
```

```
Units_Sold                   277.0    16.663274
Line_Items                    24.0          0.0
AOV_proxy                38.270833     2.448072
Ad_Clicks                    120.0     9.380597
Ad_Impressions              4694.0   481.047014
Ad_Spend                     44.93     5.809625
CTR                       0.035769     0.002968
CPC                       0.417905     0.049155
CPM                      12.942092     1.536206
ROAS                     48.890943     6.491257
ROI                      47.890943     6.491257
ConvRate_proxy            0.303797     0.023258
```

[9]:
```
sales_daily.rename(columns={
    "Daily_Revenue": "Daily Revenue",
    "Units_Sold": "Units Sold",
    "Line_Items": "Line Items",
    "AOV_proxy": "AOV (Proxy)"
}, inplace=True)

ads_daily.rename(columns={
    "Ad_Clicks": "Ad Clicks",
    "Ad_Impressions": "Ad Impressions",
    "Ad_Spend": "Ad Spend",
    "CTR": "Click-Through Rate",
    "CPC": "Cost Per Click",
    "CPM": "Cost Per 1000 Impressions"
}, inplace=True)


daily.rename(columns={
    "Daily_Revenue": "Daily Revenue",
    "Units_Sold": "Units Sold",
    "Line_Items": "Line Items",
    "AOV_proxy": "AOV (Proxy)",
    "Ad_Clicks": "Ad Clicks",
    "Ad_Impressions": "Ad Impressions",
    "Ad_Spend": "Ad Spend",
    "ConvRate_proxy": "Conversion Rate (Proxy)"
}, inplace=True)
```

[10]:
```
# ---------- 8) Save outputs ----------
sales_clean_path = "coffee_sales_clean.csv"
ads_clean_path   = "coffee_ads_clean.csv"
daily_path       = "coffee_daily_kpis.csv"

sales_df.to_csv(sales_clean_path, index=False)
```

```python
ads_df.to_csv(ads_clean_path, index=False)
daily.to_csv(daily_path, index=False)

print(f"\nSaved:\n- {sales_clean_path}\n- {ads_clean_path}\n- {daily_path}")
```

```
Saved:
- coffee_sales_clean.csv
- coffee_ads_clean.csv
- coffee_daily_kpis.csv
```

[ ]:

[ ]:

[ ]:

[ ]:

[ ]:

[ ]:

[ ]:

[ ]:

[ ]: