notebooks-01-data-generation

January 6, 2026

[2]:
```
import pandas as pd
import numpy as np
from faker import Faker
from datetime import timedelta
```

[3]:
```
fake = Faker()
np.random.seed(42)
```

1 Create merchants.csv

1.1 Business logic

Merchants sign up over 2 years

Higher plans → lower churn

Some churn early (first 60 days)

[5]:
```
n_merchants = 1200

merchant_ids = [f"M{i:05d}" for i in range(1, n_merchants + 1)]

signup_dates = pd.to_datetime(
    np.random.choice(
        pd.date_range("2023-01-01", "2024-12-31"),
        n_merchants
    )
)

plans = np.random.choice(
    ["Basic", "Shopify", "Advanced"],
     n_merchants,
     p=[0.55, 0.30, 0.15]
)

countries = np.random.choice(
    ["US", "Canada", "India", "Australia"],
    n_merchants
```

```
)
```

1.2 Churn logic

[13]:
```
churn_prob = {
    "Basic": 0.35,
    "Shopify": 0.22,
    "Advanced": 0.12
}

churned = [
    "Yes" if np.random.rand() < churn_prob[plan] else "No"
    for plan in plans
]

churn_dates = []
for i in range(n_merchants):
    if churned[i] == "Yes":
        churn_dates.append(
            signup_dates[i] + timedelta(days=np.random.randint(15, 360))
        )
    else:
        churn_dates.append(pd.NaT)

industries = np.random.choice(
    ["Fashion", "Electronics", "Home", "Beauty", "Food"],
    n_merchants
)
```

1.3 Build merchants dataframe

[15]:
```
merchants = pd.DataFrame({
    "merchant_id": merchant_ids,
    "signup_date": signup_dates,
    "country": countries,
    "plan_type": plans,
    "industry": industries,
    "churned": churned,
    "churn_date": churn_dates
})
```

1.3.1 Merchants data stored as merchants.csv

[17]:
```
merchants.to_csv("D:/Shopify-Revenue-Growth-Analytics/Data/merchants.csv",␣
↪index=False)
```

2 Create orders.csv (GMV DRIVER)

2.1 Business logic

Advanced merchants earn more

Orders increase over time

Churned merchants stop ordering

[19]:
```
orders = []

for _, row in merchants.iterrows():
    merchant_id = row["merchant_id"]
    start = row["signup_date"]
    end = row["churn_date"] if row["churned"] == "Yes" else pd.
 ↪Timestamp("2024-12-31")

    if pd.isna(end) or end <= start:
        continue

    n_orders = np.random.randint(10, 200)
    days_diff = (end - start).days

    # Skip merchants with too short a window
    if days_diff <= 1:
        continue

    for _ in range(n_orders):
        order_date = start + timedelta(days=np.random.randint(1, days_diff))

        base_value = {
            "Basic": np.random.uniform(20, 80),
            "Shopify": np.random.uniform(50, 150),
            "Advanced": np.random.uniform(120, 400)
        }[row["plan_type"]]

        orders.append({
            "order_id": fake.uuid4(),
            "merchant_id": merchant_id,
            "order_date": order_date,
            "order_value": round(base_value, 2),
            "channel": np.random.choice(["Web", "Mobile"]),
            "payment_method": np.random.choice(["Card", "Wallet"])
        })
```

2.1.1 Oreder data stored as order.csv

[20]:
```
orders_df = pd.DataFrame(orders)
orders_df.to_csv("D:/Shopify-Revenue-Growth-Analytics/Data/orders.csv",␣
 ↪index=False)
```

2.2 sessions.csv

This unlocks engagement & funnel analysis.

[22]:
```
sessions = []

for _, row in merchants.iterrows():
    merchant_id = row["merchant_id"]
    start = row["signup_date"]
    end = row["churn_date"] if row["churned"] == "Yes" else pd.
 ↪Timestamp("2024-12-31")

    if pd.isna(end) or end <= start:
        continue

    for day in pd.date_range(start, end, freq="7D"):
        s = np.random.randint(20, 500)
        c = int(s * np.random.uniform(0.01, 0.05))

        sessions.append({
            "merchant_id": merchant_id,
            "session_date": day,
            "sessions": s,
            "conversions": c
        })
```

2.2.1 Sessions data stored as sessions.csv

[23]:
```
sessions_df = pd.DataFrame(sessions)
sessions_df.to_csv("D:/Shopify-Revenue-Growth-Analytics/Data/sessions.csv",␣
 ↪index=False)
```

[ ]:
```

```