In [1]:
```python
from faker import Faker
from datetime import datetime
import pandas as pd
import random
```

In [2]:
```python
fake = Faker(locale='en_us')
```

In [3]:
```python
Fact_Sales = []

# Define date ranges for each year
yearly_date_ranges = {
    2020: (datetime(2020, 1, 1), datetime(2020, 12, 31)),
    2021: (datetime(2021, 1, 1), datetime(2021, 12, 31)),
    2022: (datetime(2022, 1, 1), datetime(2022, 12, 31)),
    2023: (datetime(2023, 1, 1), datetime(2023, 12, 31)),
    2024: (datetime(2024, 1, 1), datetime(2024, 12, 31))
}

product_choices = [
    "Produce", "Dairy", "Meat", "Bakery", "Frozen",
    "Beverages", "Snacks", "Grains", "Condiments", "Canned"
]
product_probabilities = [10, 20, 15, 6, 14, 12, 8, 5, 4, 6]

store_choices = [
    "Farmer's Market", "Gourmet Food Store", "Health Food Store",
    "Butcher Shop", "Bakery", "Online Food Retailer", "Supermarket"
]
store_probabilities = [15, 10, 10, 10, 10, 20, 25]

Promotion_choices = ["Discount Percentage", "BOGO 50% Off", "Free Shipping", "Coupon Code", "No Promotion"]
Promotion_probabilities = [20, 15, 10, 10, 45]

# Define function to generate unit price with variation over years
def generate_unit_price(year):
    if year == 2020:
        return round(random.uniform(1, 40), 2)
    elif year == 2021:
        return round(random.uniform(1, 45), 2)
    elif year == 2022:
        return round(random.uniform(1, 50), 2)
    elif year == 2023:
        return round(random.uniform(1, 50), 2)
    else:
        return round(random.uniform(1, 50.5), 2)

def generate_Quantity(year):
    if year == 2020:
        return fake.random_int(1, 55)
    elif year == 2021:
        return fake.random_int(1, 50)
    elif year == 2022:
        return fake.random_int(1, 45)
    elif year == 2023:
        return fake.random_int(1, 45)
    else:
        return fake.random_int(1, 42)


# Define function to generate promotion with variation over years
def generate_promotion(year):
    if year in [2020, 2021]:
        return random.choices(Promotion_choices, weights=Promotion_probabilities, k=1)[0]
    elif year == 2022:
        return random.choices(Promotion_choices, weights=[25, 20, 15, 10, 30], k=1)[0]
    elif year == 2023:
        return random.choices(Promotion_choices, weights=[30, 25, 20, 10, 15], k=1)[0]
    else:
        return random.choices(Promotion_choices, weights=[35, 30, 25, 5, 5], k=1)[0]

def generate_prduct(year):
    if year in [2020, 2021]:
        return random.choices(product_choices, weights=product_probabilities, k=1)[0]
    elif year == 2022:
        return random.choices(product_choices, weights=[10, 20, 15, 6, 9, 7, 8, 5, 4, 6] , k=1)[0]
    elif year == 2023:
        return random.choices(product_choices, weights=[6, 10, 9, 9, 12, 13, 16, 4, 11, 10] , k=1)[0]
    else:
        return random.choices(product_choices, weights=[5, 16, 5, 8, 18, 6, 17, 5, 4, 8] , k=1)[0]


# Generate data for each year
for year, date_range in yearly_date_ranges.items():
    for i in range(1, 50000):  # Adjust the number of rows per year as needed
```

```
        row = {}
        row['Date'] = fake.date_between_dates(date_start=date_range[0], date_end=date_range[1])
        row['Product'] = generate_prduct(year)
        row['Store'] = random.choices(store_choices, weights=store_probabilities, k=1)[0]
        row['Promotion'] = generate_promotion(year)
        row['State'] = fake.state()
        row['Quantity'] = generate_Quantity(year)
        row['UnitPrice'] = generate_unit_price(year)
        Fact_Sales.append(row)

# Convert data to DataFrame
data = pd.DataFrame(Fact_Sales)

# Save DataFrame to CSV
data.to_csv('PremiumFoodSales.csv', index=False)
```

In [4]:

```
data.head()
```

Out[4]:

	Date	Product	Store	Promotion	State	Quantity	UnitPrice
0	2020-08-11	Bakery	Supermarket	No Promotion	Minnesota	37	31.08
1	2020-03-25	Grains	Gourmet Food Store	No Promotion	Oregon	28	38.19
2	2020-04-02	Meat	Bakery	Discount Percentage	Virginia	2	36.79
3	2020-03-24	Beverages	Health Food Store	No Promotion	Louisiana	9	9.70
4	2020-06-05	Meat	Farmer's Market	No Promotion	North Dakota	9	12.76

In [ ]:

Loading [MathJax]/jax/output/CommonHTML/fonts/TeX/fontdata.js