Final ProjectKM

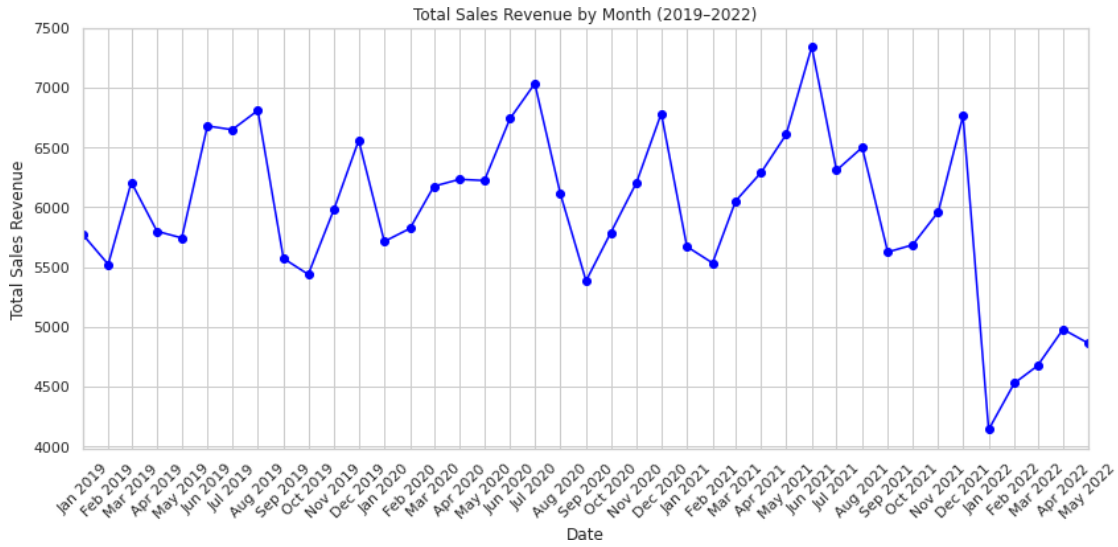
April 22, 2025

In [37]:
```
import pandas as pd
import matplotlib.pyplot as plt import
matplotlib.dates as mdates
sfm = pd.read_csv("dataset_2019_2022.csv", parse_dates=['transaction_date']) sfm =
sfm[sfm['transaction_date'] <= '2022-05-31']
sfm['month_start'] = sfm['transaction_date'].dt.to_period('M').dt.to_timestamp()
total_sales_by_month = sfm.groupby('month_start')['price'].sum().reset_index() for _, row in
total_sales_by_month.iterrows():
    print(f"Month: {row['month_start'].strftime('%Y-%m')}, Total Sales Revenue: {row[
plt.figure(figsize=(12, 6))
plt.plot(total_sales_by_month['month_start'], total_sales_by_month['price'], marker='
plt.xlabel('Date')
plt.ylabel('Total Sales Revenue')
plt.title('Total Sales Revenue by Month (20192022)')
plt.gca().set_xlim(total_sales_by_month['month_start'].min(), total_sales_by_month['m
plt.gca().xaxis.set_major_formatter(mdates.DateFormatter('%b %Y'))
plt.gca().xaxis.set_major_locator(mdates.MonthLocator(interval=1)) plt.xticks(rotation=45)
plt.grid(True) plt.tight_layout()
plt.show()
```

```
Month: 2019-01, Total Sales Revenue: 5773.51
Month: 2019-02, Total Sales Revenue: 5522.09
Month: 2019-03, Total Sales Revenue: 6208.01
Month: 2019-04, Total Sales Revenue: 5799.10
Month: 2019-05, Total Sales Revenue: 5742.71
Month: 2019-06, Total Sales Revenue: 6679.38
Month: 2019-07, Total Sales Revenue: 6648.28
Month: 2019-08, Total Sales Revenue: 6808.94
Month: 2019-09, Total Sales Revenue: 5571.46
Month: 2019-10, Total Sales Revenue: 5438.32
Month: 2019-11, Total Sales Revenue: 5980.67
```

Month: 2019-12, Total Sales Revenue: 6559.29
Month: 2020-01, Total Sales Revenue: 5713.49
Month: 2020-02, Total Sales Revenue: 5822.17
Month: 2020-03, Total Sales Revenue: 6175.00

Month: 2020-04, Total Sales Revenue: 6233.30
Month: 2020-05, Total Sales Revenue: 6222.72
Month: 2020-06, Total Sales Revenue: 6740.23
Month: 2020-07, Total Sales Revenue: 7034.01
Month: 2020-08, Total Sales Revenue: 6110.41
Month: 2020-09, Total Sales Revenue: 5384.04
Month: 2020-10, Total Sales Revenue: 5788.13
Month: 2020-11, Total Sales Revenue: 6205.28
Month: 2020-12, Total Sales Revenue: 6783.53
Month: 2021-01, Total Sales Revenue: 5670.54
Month: 2021-02, Total Sales Revenue: 5533.46
Month: 2021-03, Total Sales Revenue: 6052.53
Month: 2021-04, Total Sales Revenue: 6292.19
Month: 2021-05, Total Sales Revenue: 6606.92
Month: 2021-06, Total Sales Revenue: 7340.29
Month: 2021-07, Total Sales Revenue: 6308.48
Month: 2021-08, Total Sales Revenue: 6499.43
Month: 2021-09, Total Sales Revenue: 5625.00
Month: 2021-10, Total Sales Revenue: 5684.65
Month: 2021-11, Total Sales Revenue: 5960.48
Month: 2021-12, Total Sales Revenue: 6761.90
Month: 2022-01, Total Sales Revenue: 4140.61
Month: 2022-02, Total Sales Revenue: 4527.99
Month: 2022-03, Total Sales Revenue: 4673.15
Month: 2022-04, Total Sales Revenue: 4979.00
Month: 2022-05, Total Sales Revenue: 4864.40



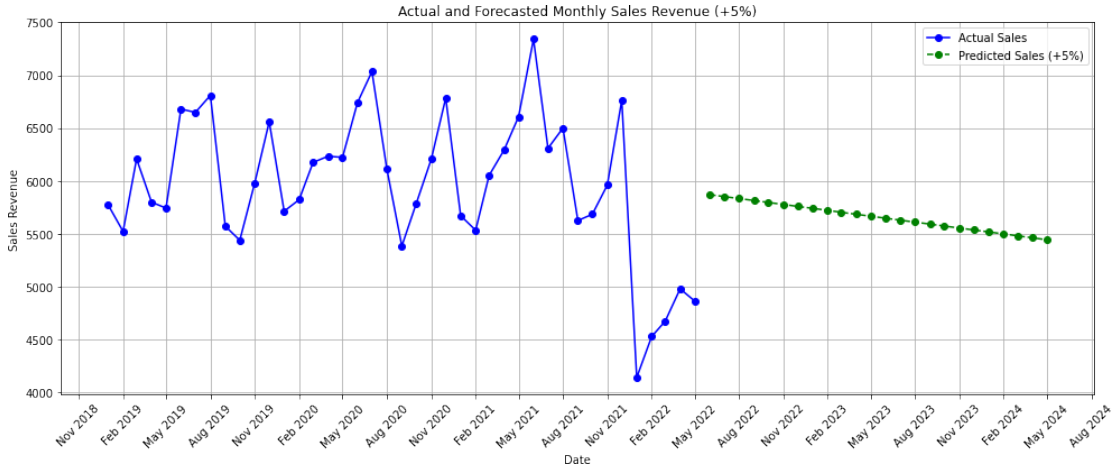
In [80]:
```
import pandas as pd
import numpy as np
```

```
sfm = pd.read_csv("dataset_2019_2022.csv", parse_dates=['transaction_date'])

sfm = sfm[sfm['transaction_date'] <= '2022-05-31']
sfm['month_start'] = sfm['transaction_date'].dt.to_period('M').dt.to_timestamp() total_sales_by_month =
sfm.groupby('month_start')['price'].sum().reset_index() total_sales_by_month.set_index('month_start', inplace=True)

total_sales_by_month.index = pd.to_datetime(total_sales_by_month.index) + pd.offsets.

jan_2021_sales = total_sales_by_month.loc['2021-01-31', 'price'] jan_2022_sales =
total_sales_by_month.loc['2022-01-31', 'price']

percentage_change_jan = ((jan_2022_sales - jan_2021_sales) / jan_2021_sales) * 100

print(f"Sales on Jan, 2021: ${jan_2021_sales:,.2f}") print(f"Sales on Jan, 2022:
${jan_2022_sales:,.2f}") print(f"Percentage change: {percentage_change_jan:.2f}%")
```

```
Sales on Jan, 2021: $5,670.54 Sales on Jan, 2022: $4,140.61
Percentage change: -26.98%
```

In [41]:
```
import pandas as pd
import numpy as np
import matplotlib.pyplot as plt import matplotlib.dates as mdates
from sklearn.linear_model import LinearRegression
sfm = pd.read_csv("dataset_2019_2022.csv", parse_dates=['transaction_date']) sfm = sfm[sfm['transaction_date']
<= '2022-05-31']
sfm['month_start'] = sfm['transaction_date'].dt.to_period('M').dt.to_timestamp() total_sales_by_month =
sfm.groupby('month_start')['price'].sum().reset_index() total_sales_by_month['time_index'] =
np.arange(len(total_sales_by_month)) X_train = total_sales_by_month[['time_index']]
y_train = total_sales_by_month['price'] model = LinearRegression()
model.fit(X_train, y_train)
future_months = pd.date_range(start='2022-06-01', end='2024-05-01', freq='MS') future_index = pd.DataFrame({
    'time_index': np.arange(len(total_sales_by_month), len(total_sales_by_month) + le
})
future_predictions = model.predict(future_index) future_predictions = future_predictions
* 1.05 future_sales_df = pd.DataFrame({
    'month_start': future_months,
    'predicted_sales': future_predictions
})
```

```
plt.figure(figsize=(14, 6))
plt.plot(total_sales_by_month['month_start'], total_sales_by_month['price'], marker='
plt.plot(future_sales_df['month_start'], future_sales_df['predicted_sales'], marker=' plt.xlabel('Date')
plt.ylabel('Sales Revenue')
plt.title('Actual and Forecasted Monthly Sales Revenue (+5%)')
plt.gca().xaxis.set_major_formatter(mdates.DateFormatter('%b %Y'))
plt.gca().xaxis.set_major_locator(mdates.MonthLocator(interval=3)) plt.xticks(rotation=45)
plt.grid(True) plt.legend() plt.tight_layout()
plt.show()
```



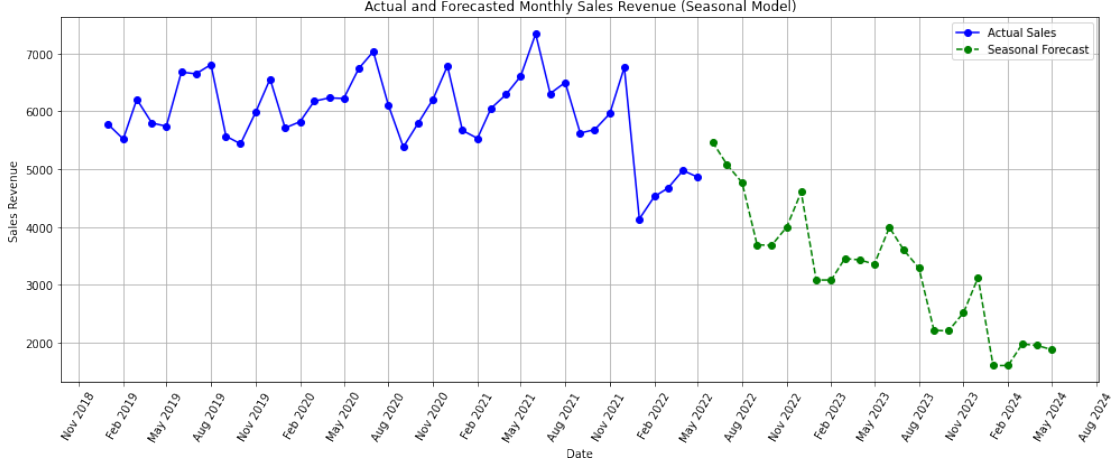
In [73]:
```
import pandas as pd
import numpy as np
import matplotlib.pyplot as plt import matplotlib.dates as mdates
from statsmodels.tsa.seasonal import STL
from statsmodels.tsa.holtwinters import ExponentialSmoothing

sfm = pd.read_csv("dataset_2019_2022.csv", parse_dates=['transaction_date'])

sfm = sfm[sfm['transaction_date'] <= '2022-05-31']
sfm['month_start'] = sfm['transaction_date'].dt.to_period('M').dt.to_timestamp()

total_sales_by_month = sfm.groupby('month_start')['price'].sum().reset_index()
total_sales_by_month.set_index('month_start', inplace=True) total_sales_by_month.index =
```

```
pd.DatetimeIndex(total_sales_by_month.index, freq='MS')

stl = STL(total_sales_by_month['price'], seasonal=13)
```

```
result = stl.fit()

model = ExponentialSmoothing( total_sales_by_month['price'],
    trend='add',
    seasonal='add', seasonal_periods=12,
    initialization_method='estimated'
)
hw_fit = model.fit()

forecast_periods = 24
forecast = hw_fit.forecast(forecast_periods)
forecast_index = pd.date_range(start=total_sales_by_month.index[-1] + pd.offsets.Mont forecast_series =
pd.Series(forecast.values, index=forecast_index)

plt.figure(figsize=(14, 6))
plt.plot(total_sales_by_month.index, total_sales_by_month['price'], label='Actual Sal plt.plot(forecast_series.index,
forecast_series, label='Seasonal Forecast', color='gr plt.xlabel('Date')
plt.ylabel('Sales Revenue')
plt.title('Actual and Forecasted Monthly Sales Revenue (Seasonal Model)')
plt.gca().xaxis.set_major_formatter(mdates.DateFormatter('%b %Y'))
plt.gca().xaxis.set_major_locator(mdates.MonthLocator(interval=3)) plt.xticks(rotation=60)
plt.grid(True) plt.legend() plt.tight_layout()
plt.show()
```

/usr/lib/python3.7/site-packages/statsmodels/tsa/base/tsa_model.py:132: FutureWarning: The 'freq' argument in Timestamp

is deprecated and will be removed in a future version.

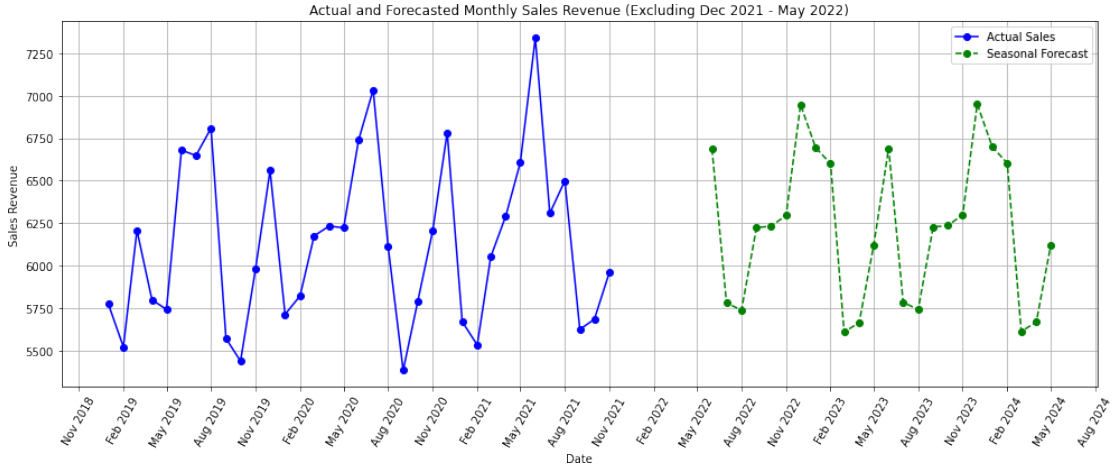


In [21]:
```
import pandas as pd
import numpy as np
import matplotlib.pyplot as plt import matplotlib.dates as mdates
from statsmodels.tsa.seasonal import STL
from statsmodels.tsa.holtwinters import ExponentialSmoothing

sfm = pd.read_csv("dataset_2019_2022.csv", parse_dates=['transaction_date'])

sfm = sfm[sfm['transaction_date'] < '2021-12-01']
sfm['month_start'] = sfm['transaction_date'].dt.to_period('M').dt.to_timestamp()

total_sales_by_month = sfm.groupby('month_start')['price'].sum().reset_index()
total_sales_by_month.set_index('month_start', inplace=True) total_sales_by_month.index =
pd.DatetimeIndex(total_sales_by_month.index, freq='MS')

seasonal_model = ExponentialSmoothing( total_sales_by_month['price'],
    trend='add',
    seasonal='add', seasonal_periods=12,
    initialization_method='estimated'
).fit()

forecast_periods = 24
forecast = seasonal_model.forecast(forecast_periods)
forecast_index = pd.date_range(start='2022-06-01', periods=forecast_periods, freq='MS forecast_series =
```

```
pd.Series(forecast.values, index=forecast_index)

plt.figure(figsize=(14, 6))
plt.plot(total_sales_by_month.index, total_sales_by_month['price'], label='Actual Sal
```

```
plt.plot(forecast_series.index, forecast_series, label='Seasonal Forecast', marker='o plt.xlabel('Date')
plt.ylabel('Sales Revenue')
plt.title('Actual and Forecasted Monthly Sales Revenue (Excluding Dec 2021 - May 2022
plt.gca().xaxis.set_major_formatter(mdates.DateFormatter('%b %Y'))
plt.gca().xaxis.set_major_locator(mdates.MonthLocator(interval=3)) plt.xticks(rotation=60)
plt.grid(True) plt.legend() plt.tight_layout()
plt.show()
```

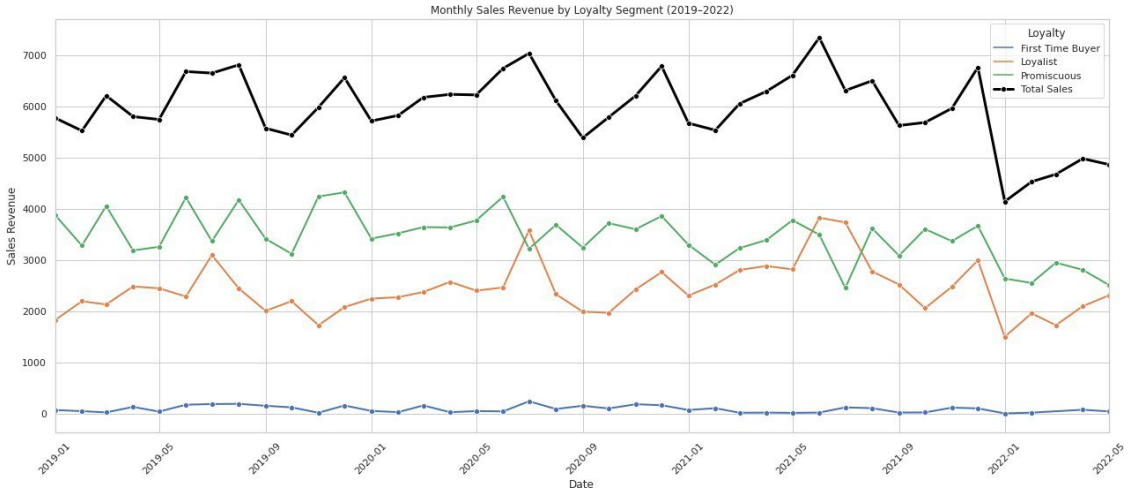
/usr/lib/python3.7/site-packages/statsmodels/tsa/base/tsa_model.py:132: FutureWarning: The 'freq' argument in Timestamp is deprecated and will be removed in a future version.



In [38]:
```
import pandas as pd
import matplotlib.pyplot as plt import seaborn as sns
sfm = pd.read_csv("dataset_2019_2022.csv", parse_dates=['transaction_date']) sfm = sfm[sfm['transaction_date']
<= '2022-05-31']
sfm['date'] = sfm['transaction_date'].dt.to_period('M').dt.to_timestamp() monthly_total =
sfm.groupby('date')['price'].sum().reset_index(name='total_sales') loyalty_monthly = sfm.groupby(['date',
'loyalty'])['price'].sum().reset_index() sns.set(style="whitegrid")
plt.figure(figsize=(18, 8))
sns.lineplot(data=loyalty_monthly, x='date', y='price', hue='loyalty', marker='o', li
sns.lineplot(data=monthly_total, x='date', y='total_sales', color='black', label='Tot min_date =
```

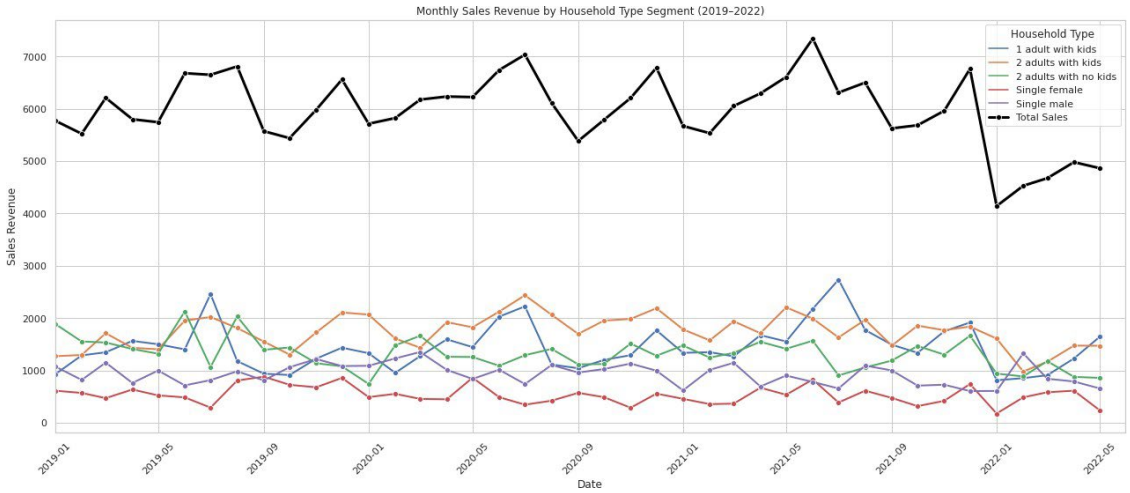
```
monthly_total['date'].min()
```

```
max_date = monthly_total['date'].max() plt.xlim(min_date, max_date)
plt.title('Monthly Sales Revenue by Loyalty Segment (20192022)') plt.xlabel('Date')
plt.ylabel('Sales Revenue') plt.xticks(rotation=45)
plt.legend(title='Loyalty') plt.tight_layout() plt.grid(True)
plt.show()
```

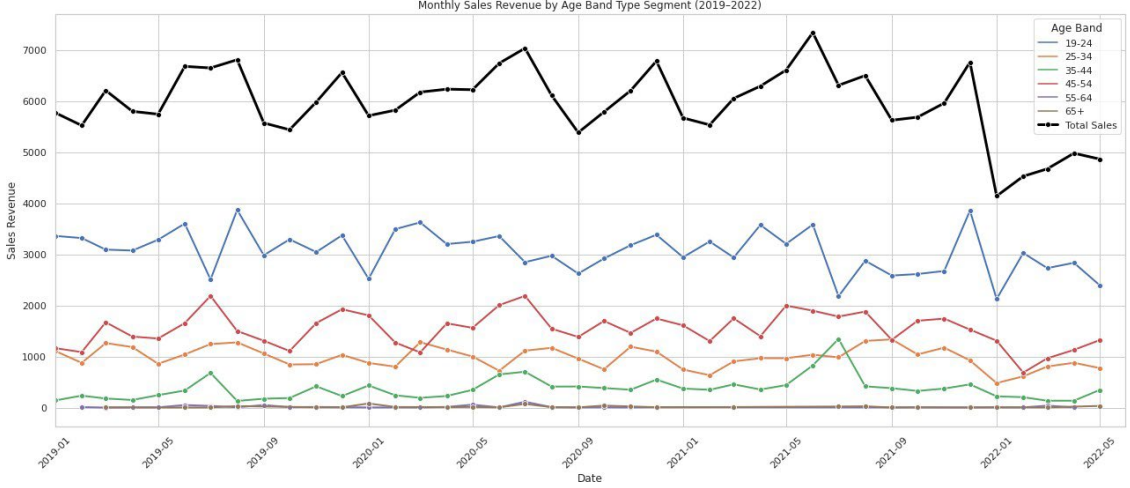


In [42]:
```
import pandas as pd
import matplotlib.pyplot as plt import seaborn as sns
sfm = pd.read_csv("dataset_2019_2022.csv", parse_dates=['transaction_date']) sfm = sfm[sfm['transaction_date']
<= '2022-05-31']
sfm['date'] = sfm['transaction_date'].dt.to_period('M').dt.to_timestamp() monthly_total =
sfm.groupby('date')['price'].sum().reset_index(name='total_sales') household_monthly = sfm.groupby(['date',
'household_type'])['price'].sum().reset_inde sns.set(style="whitegrid")
plt.figure(figsize=(18, 8))
sns.lineplot(data=household_monthly, x='date', y='price', hue='household_type', marke
sns.lineplot(data=monthly_total, x='date', y='total_sales', color='black', label='Tot plt.xlim(pd.to_datetime('2019-
01-01'), pd.to_datetime('2022-05-31')) plt.title('Monthly Sales Revenue by Household Type Segment (20192022)')
plt.xlabel('Date')
plt.ylabel('Sales Revenue') plt.xticks(rotation=45)
plt.legend(title='Household Type') plt.tight_layout()
```

```
plt.grid(True) plt.show()
```



In [40]:
```
import pandas as pd
import matplotlib.pyplot as plt import seaborn as sns
sfm = pd.read_csv("dataset_2019_2022.csv", parse_dates=['transaction_date']) sfm = sfm[sfm['transaction_date']
<= '2022-05-31']
sfm['date'] = sfm['transaction_date'].dt.to_period('M').dt.to_timestamp() monthly_total =
sfm.groupby('date')['price'].sum().reset_index(name='total_sales') household_monthly = sfm.groupby(['date',
'age_band'])['price'].sum().reset_index() sns.set(style="whitegrid")
plt.figure(figsize=(18, 8))
sns.lineplot(data=household_monthly, x='date', y='price', hue='age_band', marker='o',
sns.lineplot(data=monthly_total, x='date', y='total_sales', color='black', label='Tot plt.xlim(pd.to_datetime('2019-
01-01'), pd.to_datetime('2022-05-31')) plt.title('Monthly Sales Revenue by Age Band Type Segment (20192022)')
plt.xlabel('Date')
plt.ylabel('Sales Revenue') plt.xticks(rotation=45)
plt.legend(title='Age Band') plt.tight_layout()
plt.grid(True)
plt.show()
```



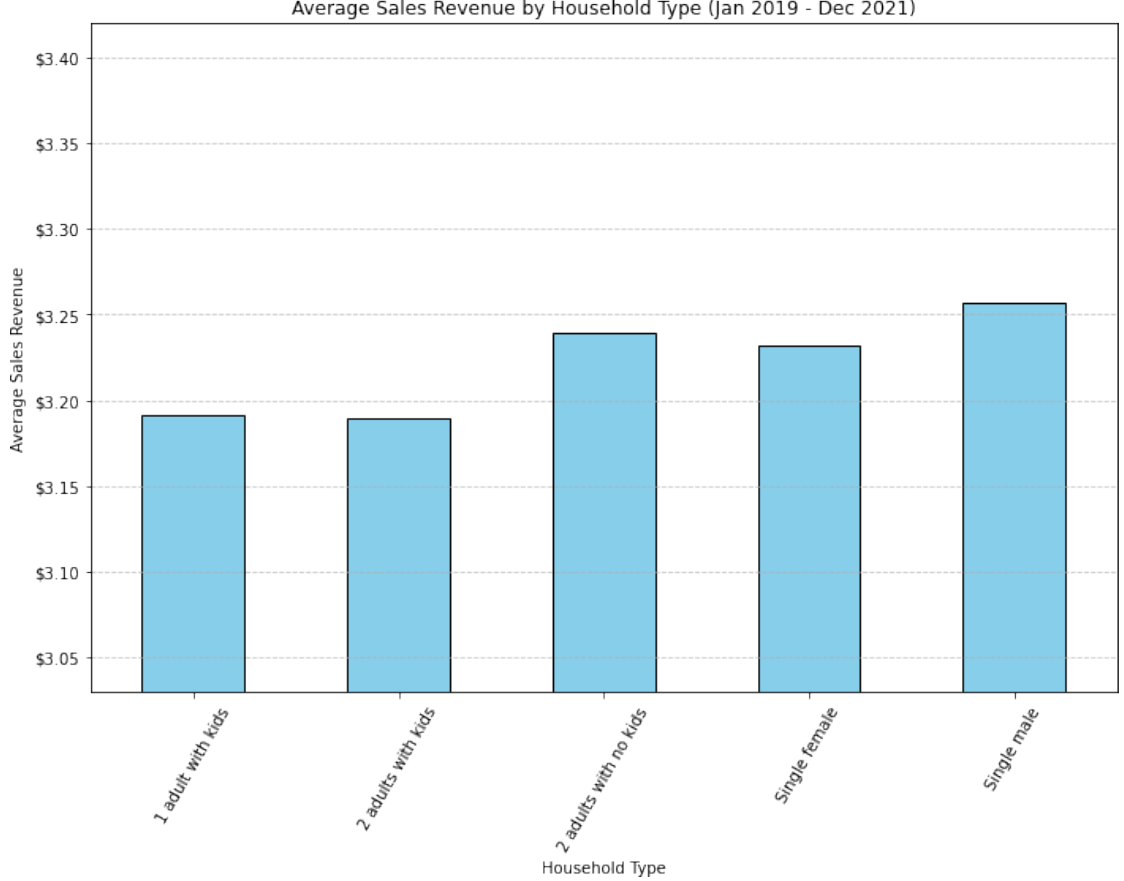
In [20]:
```
import pandas as pd
import numpy as np
sfm = pd.read_csv("dataset_2019_2022.csv", parse_dates=['transaction_date']) historical_years =
sfm[(sfm['transaction_date'] >= '2019-01-01') &
                    (sfm['transaction_date'] <= '2021-12-31')].copy() historical_years['sales_revenue'] =
historical_years['price'] historical_years_mean_sales_by_household_type = historical_years.groupby('household_t
print(historical_years_mean_sales_by_household_type)
```

	household_type	sales_revenue
0	1 adult with kids	3.191076
1	2 adults with kids	3.189138
2	2 adults with no kids	3.239422
3	Single female	3.231532
4	Single male	3.256924

In [21]:
```
import pandas as pd
import matplotlib.pyplot as plt import matplotlib.ticker as mtick
sfm = pd.read_csv("dataset_2019_2022.csv", parse_dates=['transaction_date'])

historical_years = sfm[(sfm['transaction_date'] >= '2019-01-01') &
                    (sfm['transaction_date'] <= '2021-12-31')].copy() historical_years['price'] =

pd.to_numeric(historical_years['price'], errors='coerce')
```

```
historical_years['sales_revenue'] = historical_years['price']
mean_sales_by_household = historical_years.groupby('household_type')['sales_revenue'] fig, ax =

plt.subplots(figsize=(10, 8))
```

```
mean_sales_by_household.plot( kind='bar', color='skyblue',
    edgecolor='black', x='household_type',
    y='sales_revenue',
    ax=ax
)

ax.get_legend().remove() ax.yaxis.set_major_formatter(mtick.StrMethodFormatter('${x:,.2f}'))

ymin = mean_sales_by_household['sales_revenue'].min() * 0.95 ymax =
mean_sales_by_household['sales_revenue'].max() * 1.05 plt.ylim(ymin, ymax)

plt.title('Average Sales Revenue by Household Type (Jan 2019 - Dec 2021)') plt.xlabel('Household Type')
plt.ylabel('Average Sales Revenue') plt.xticks(rotation=60)
plt.grid(axis='y', linestyle='--', alpha=0.7) plt.tight_layout()
plt.show()
```



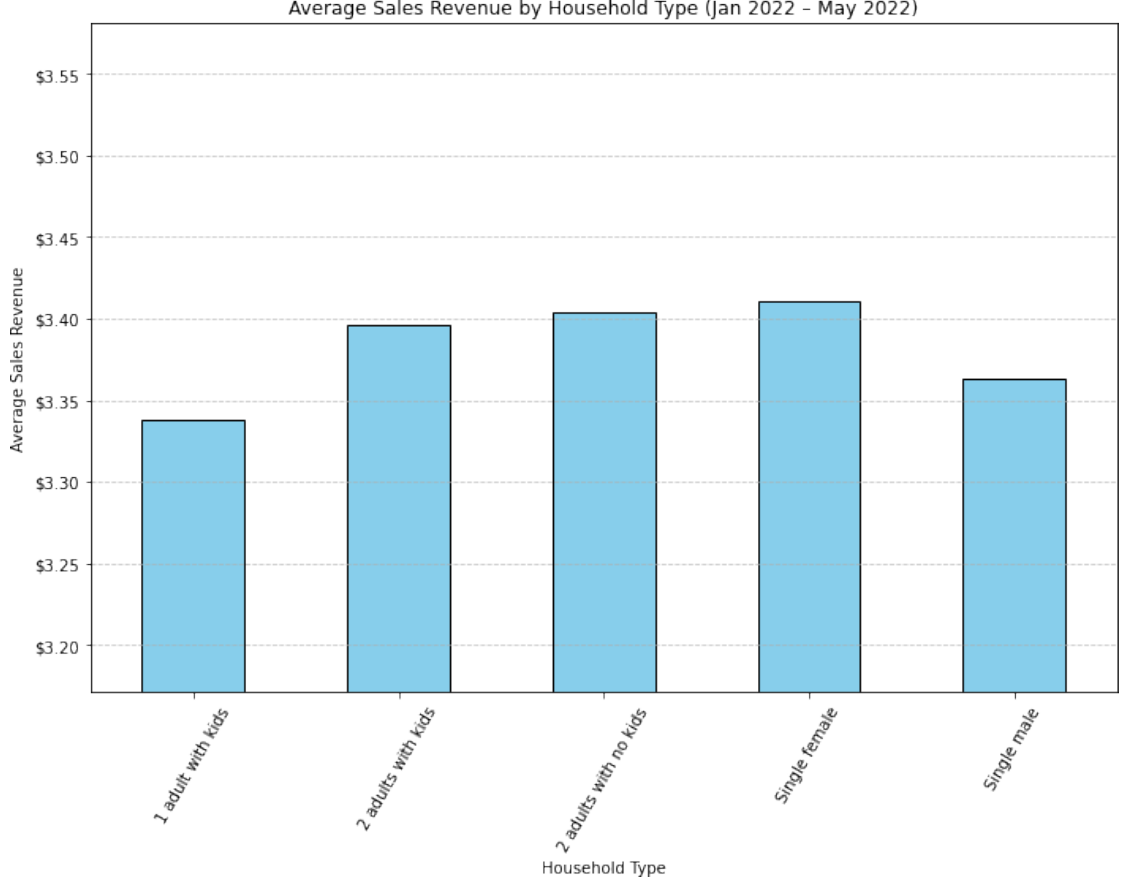

```
In [22]: import pandas as pd
         import numpy as np
         sfm = pd.read_csv("dataset_2019_2022.csv", parse_dates=['transaction_date']) recent_months =
         sfm[(sfm['transaction_date'] >= '2022-01-01') &
                                  (sfm['transaction_date'] <= '2022-05-31')].copy() recent_months['sales_revenue'] =
         recent_months['price'] recent_months_mean_sales_by_household_type = recent_months.groupby('household_type')[
         print(recent_months_mean_sales_by_household_type)
```

	household_type	sales_revenue
0	1 adult with kids	3.338080
1	2 adults with kids	3.396567
2	2 adults with no kids	3.403911
3	Single female	3.410407

4	Single	male	3.363014

In [23]:
```
import pandas as pd
import matplotlib.pyplot as plt import matplotlib.ticker as mtick
```

```
sfm = pd.read_csv("dataset_2019_2022.csv", parse_dates=['transaction_date'])

recent_months = sfm[(sfm['transaction_date'] >= '2022-01-01') &
                    (sfm['transaction_date'] <= '2022-05-31')].copy()

recent_months['price'] = pd.to_numeric(recent_months['price'], errors='coerce')

recent_months['sales_revenue'] = recent_months['price']
mean_sales_by_household = recent_months.groupby('household_type')['sales_revenue'].me fig, ax =

plt.subplots(figsize=(10, 8))

mean_sales_by_household.plot( kind='bar', color='skyblue',
    edgecolor='black', x='household_type',
    y='sales_revenue',
    ax=ax
)

ax.get_legend().remove() ax.yaxis.set_major_formatter(mtick.StrMethodFormatter('${x:,.2f}'))

ymin = mean_sales_by_household['sales_revenue'].min() * 0.95 ymax =
mean_sales_by_household['sales_revenue'].max() * 1.05 plt.ylim(ymin, ymax)

plt.title('Average Sales Revenue by Household Type (Jan 2022 May 2022)') plt.xlabel('Household Type')
plt.ylabel('Average Sales Revenue') plt.xticks(rotation=60)
plt.grid(axis='y', linestyle='--', alpha=0.7) plt.tight_layout()
plt.show()
```

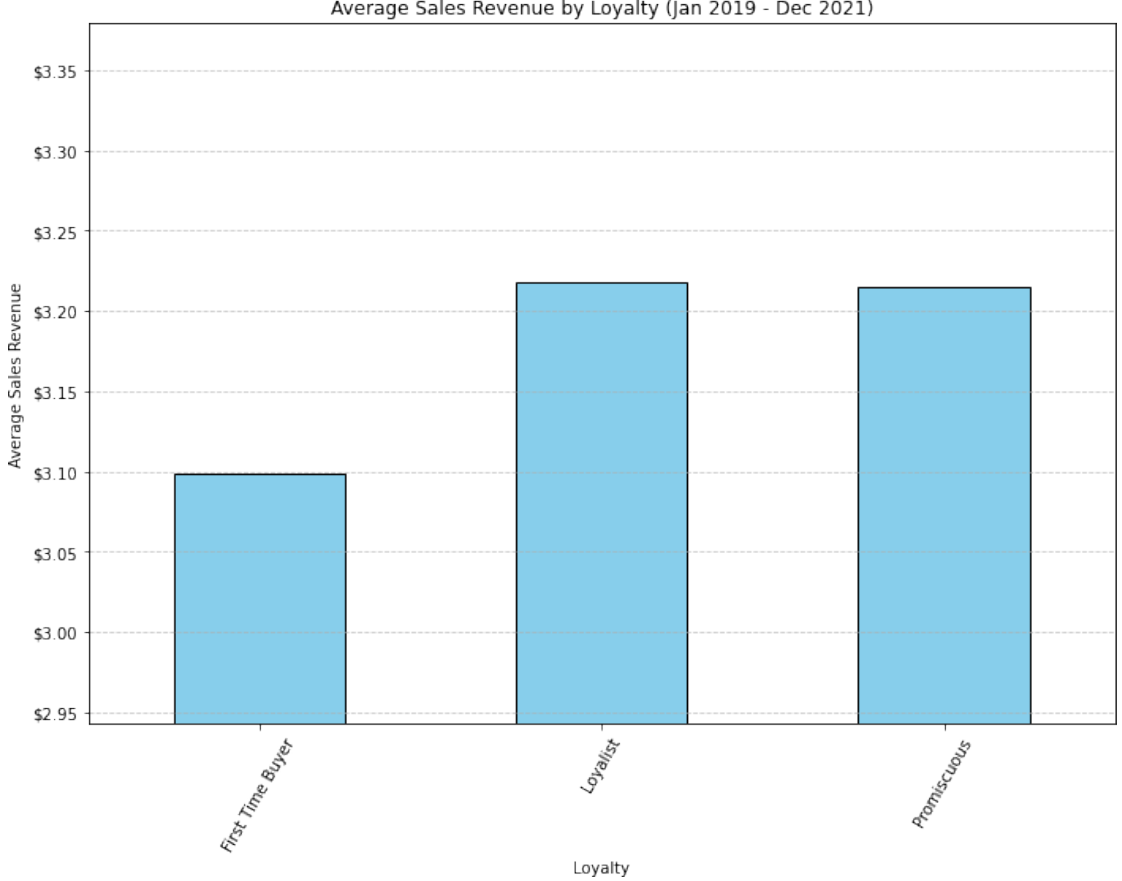


In [17]:
```
import pandas as pd
import numpy as np
sfm = pd.read_csv("dataset_2019_2022.csv", parse_dates=['transaction_date']) historical_years =
sfm[(sfm['transaction_date'] >= '2019-01-01') &
                    (sfm['transaction_date'] <= '2021-12-31')].copy() historical_years['sales_revenue'] =
historical_years['price'] historical_years_mean_sales_by_loyalty = historical_years.groupby('loyalty')['sales_r
print(historical_years_mean_sales_by_loyalty)
```

	loyalty	sales_revenue
0	First Time Buyer	3.098247
1	Loyalist	3.218279
2	Promiscuous	3.215402

In [16]:
```
import pandas as pd
import matplotlib.pyplot as plt import matplotlib.ticker as mtick
sfm = pd.read_csv("dataset_2019_2022.csv", parse_dates=['transaction_date'])
```

```
historical_years = sfm[(sfm['transaction_date'] >= '2019-01-01') &
                       (sfm['transaction_date'] <= '2021-12-31')].copy() historical_years['price'] =

pd.to_numeric(historical_years['price'], errors='coerce')

historical_years['sales_revenue'] = historical_years['price']
mean_sales_by_loyalty = historical_years.groupby('loyalty')['sales_revenue'].mean(). fig, ax =

plt.subplots(figsize=(10, 8))

mean_sales_by_loyalty.plot( kind='bar', color='skyblue',
    edgecolor='black', x='loyalty', y='sales_revenue',
    ax=ax
)

ax.get_legend().remove() ax.yaxis.set_major_formatter(mtick.StrMethodFormatter('${x:,.2f}'))

ymin = mean_sales_by_loyalty['sales_revenue'].min() * 0.95 ymax =
mean_sales_by_loyalty['sales_revenue'].max() * 1.05 plt.ylim(ymin, ymax)

plt.title('Average Sales Revenue by Loyalty (Jan 2019 - Dec 2021)') plt.xlabel('Loyalty')
plt.ylabel('Average Sales Revenue') plt.xticks(rotation=60)
plt.grid(axis='y', linestyle='--', alpha=0.7) plt.tight_layout()
plt.show()
```

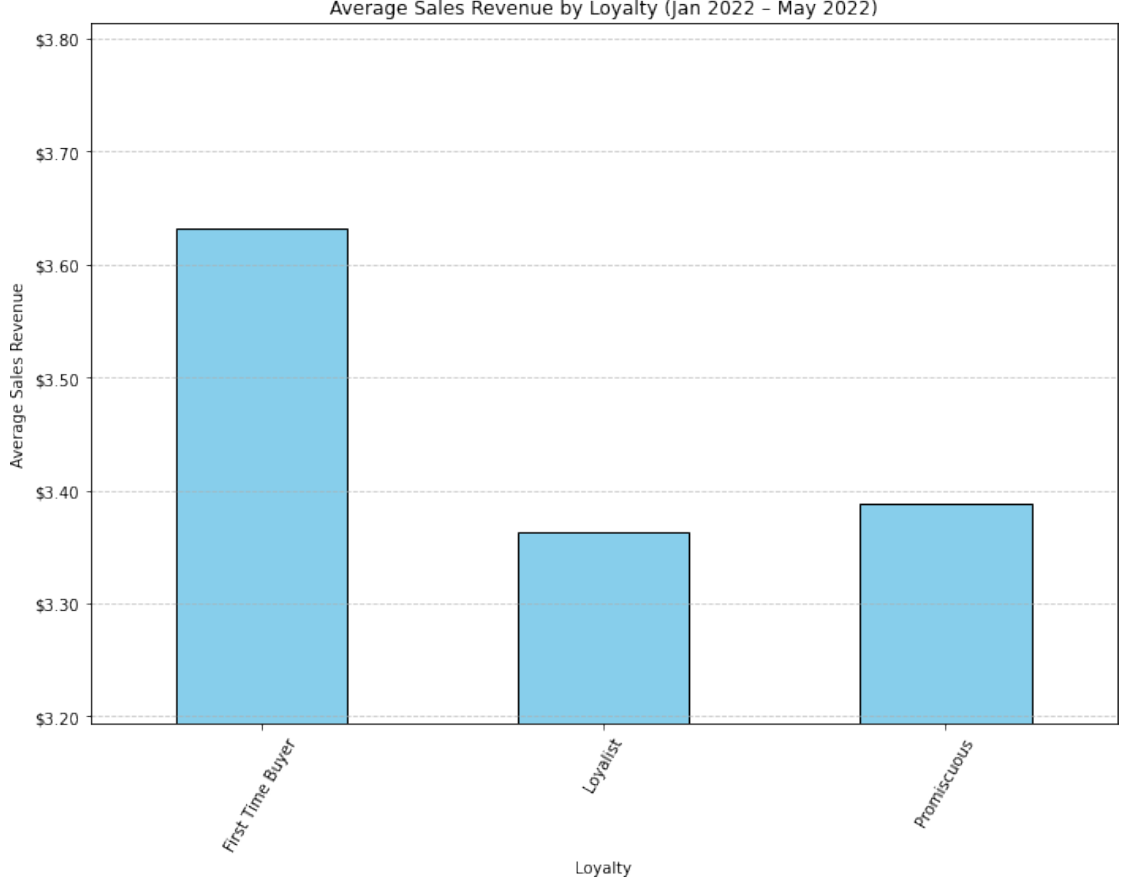


In [18]:
```
import pandas as pd
import numpy as np
sfm = pd.read_csv("dataset_2019_2022.csv", parse_dates=['transaction_date']) recent_months =
sfm[(sfm['transaction_date'] >= '2022-01-01') &
                    (sfm['transaction_date'] <= '2022-05-31')].copy() recent_months['sales_revenue'] =
recent_months['price'] recent_months_mean_sales_by_loyalty = recent_months.groupby('loyalty')['sales_revenue
print(recent_months_mean_sales_by_loyalty)
```

	loyalty	sales_revenue
0	First Time Buyer	3.631795
1	Loyalist	3.362438
2	Promiscuous	3.388869

In [19]:
```
import pandas as pd
import matplotlib.pyplot as plt import matplotlib.ticker as mtick
sfm = pd.read_csv("dataset_2019_2022.csv", parse_dates=['transaction_date'])
```

```
recent_months = sfm[(sfm['transaction_date'] >= '2022-01-01') &
                    (sfm['transaction_date'] <= '2022-05-31')].copy() recent_months['price'] =

pd.to_numeric(recent_months['price'], errors='coerce')

recent_months['sales_revenue'] = recent_months['price']
mean_sales_by_loyalty = recent_months.groupby('loyalty')['sales_revenue'].mean().rese  fig, ax =

plt.subplots(figsize=(10, 8))

mean_sales_by_loyalty.plot( kind='bar', color='skyblue',
    edgecolor='black', x='loyalty', y='sales_revenue',
    ax=ax
)

ax.get_legend().remove() ax.yaxis.set_major_formatter(mtick.StrMethodFormatter('${x:,.2f}'))

ymin = mean_sales_by_loyalty['sales_revenue'].min() * 0.95 ymax =
mean_sales_by_loyalty['sales_revenue'].max() * 1.05 plt.ylim(ymin, ymax)

plt.title('Average Sales Revenue by Loyalty (Jan 2022 May 2022)') plt.xlabel('Loyalty')
plt.ylabel('Average Sales Revenue') plt.xticks(rotation=60)
plt.grid(axis='y', linestyle='--', alpha=0.7) plt.tight_layout()
plt.show()
```



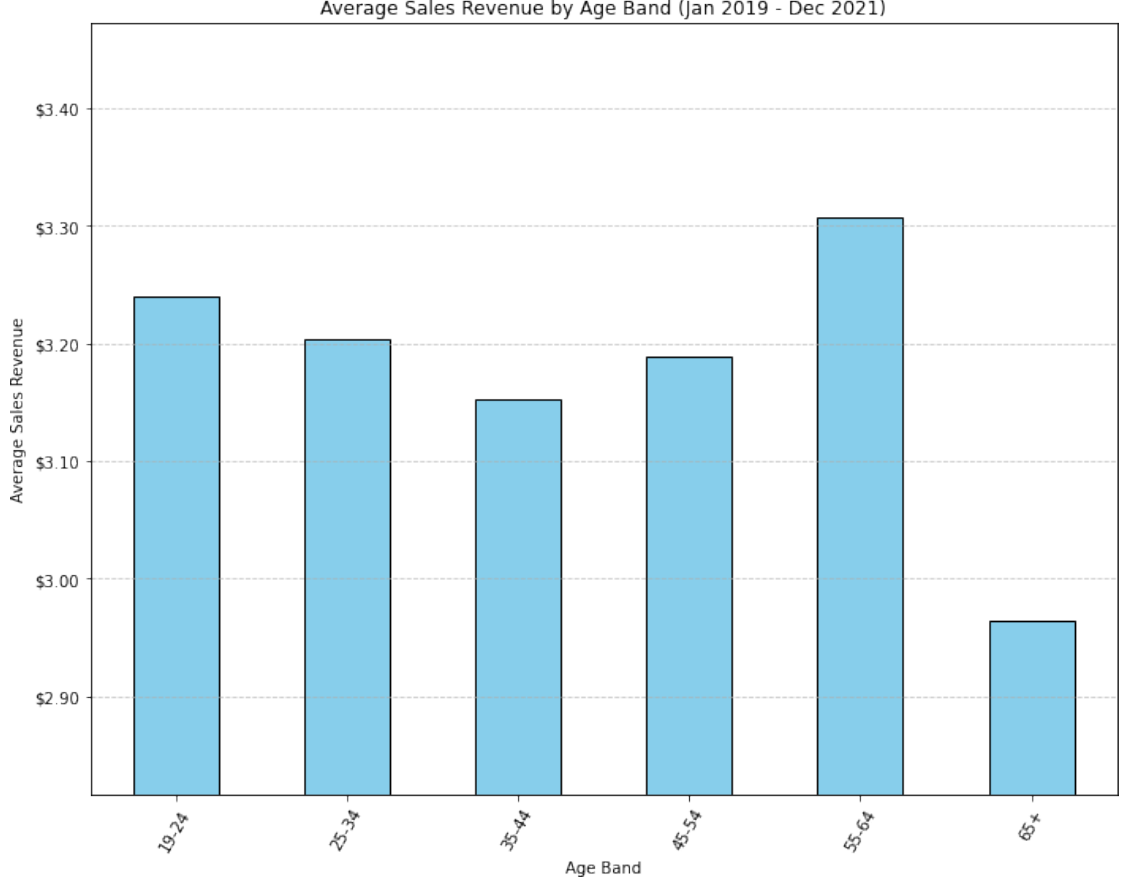
In [24]:
```
import pandas as pd
import numpy as np
sfm = pd.read_csv("dataset_2019_2022.csv", parse_dates=['transaction_date']) historical_years =
sfm[(sfm['transaction_date'] >= '2019-01-01') &
                    (sfm['transaction_date'] <= '2021-12-31')].copy() historical_years['sales_revenue'] =
historical_years['price'] historical_years_mean_sales_by_age_band = historical_years.groupby('age_band')['sales
print(historical_years_mean_sales_by_age_band)
```

	age_band	sales_revenue
0	19-24	3.240157
1	25-34	3.204240
2	35-44	3.152099
3	45-54	3.189039

4	55-64	3.307083
5	65+	2.964531

In [25]:
```
import pandas as pd
import matplotlib.pyplot as plt
```

```
import matplotlib.ticker as mtick
sfm = pd.read_csv("dataset_2019_2022.csv", parse_dates=['transaction_date'])

historical_years = sfm[(sfm['transaction_date'] >= '2019-01-01') &
                       (sfm['transaction_date'] <= '2021-12-31')].copy()

historical_years['price'] = pd.to_numeric(historical_years['price'], errors='coerce')

historical_years['sales_revenue'] = historical_years['price']
mean_sales_by_age_band = historical_years.groupby('age_band')['sales_revenue'].mean() fig, ax =

plt.subplots(figsize=(10, 8))

mean_sales_by_age_band.plot( kind='bar', color='skyblue',
    edgecolor='black', x='age_band', y='sales_revenue',
    ax=ax
)

ax.get_legend().remove() ax.yaxis.set_major_formatter(mtick.StrMethodFormatter('${x:,.2f}'))

ymin = mean_sales_by_age_band['sales_revenue'].min() * 0.95 ymax =
mean_sales_by_age_band['sales_revenue'].max() * 1.05 plt.ylim(ymin, ymax)

plt.title('Average Sales Revenue by Age Band (Jan 2019 - Dec 2021)') plt.xlabel('Age Band')
plt.ylabel('Average Sales Revenue') plt.xticks(rotation=60)
plt.grid(axis='y', linestyle='--', alpha=0.7) plt.tight_layout()
plt.show()
```



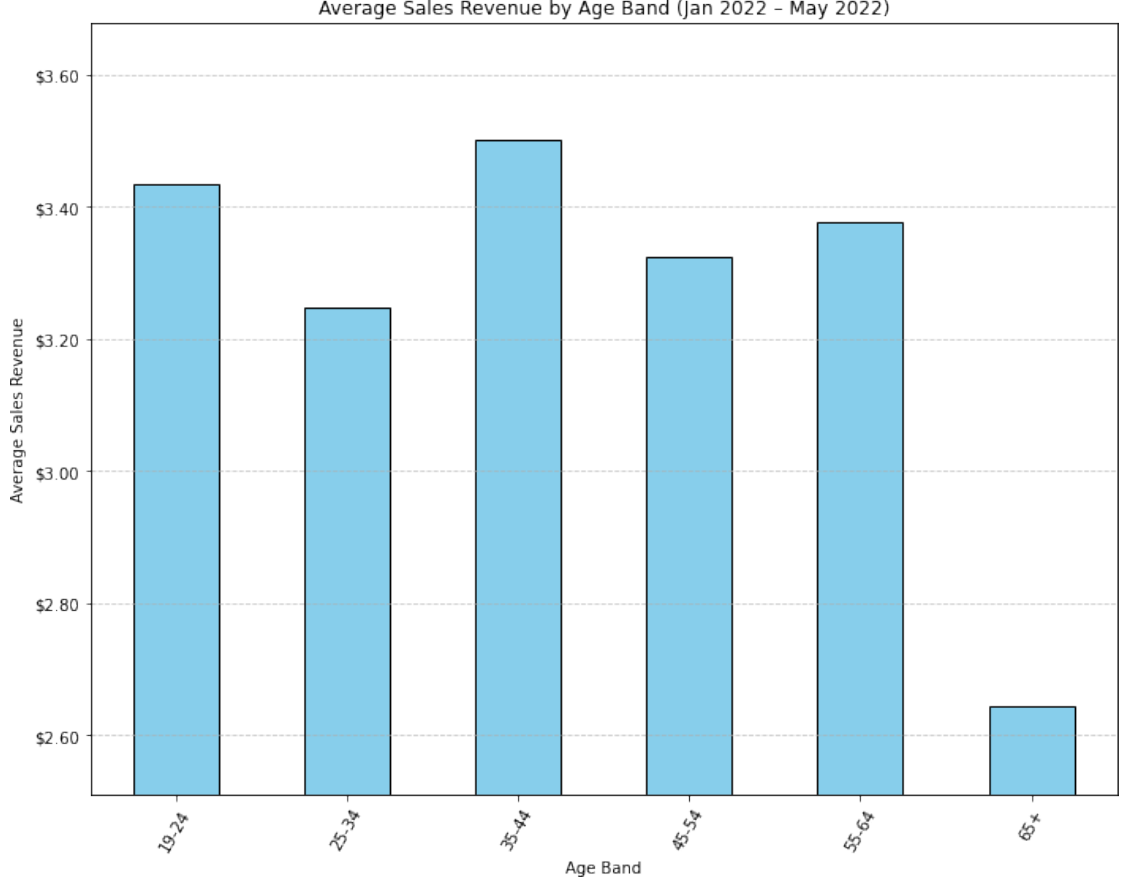
In [26]:
```
import pandas as pd
import numpy as np
sfm = pd.read_csv("dataset_2019_2022.csv", parse_dates=['transaction_date']) recent_months =
sfm[(sfm['transaction_date'] >= '2022-01-01') &
                    (sfm['transaction_date'] <= '2022-05-31')].copy() recent_months['sales_revenue'] =
recent_months['price'] recent_months_mean_sales_by_age_band = recent_months.groupby('age_band')['sales_reven
print(recent_months_mean_sales_by_age_band)
```

	age_band	sales_revenue
0	19-24	3.434380
1	25-34	3.246273
2	35-44	3.502196
3	45-54	3.324130

4	55-64	3.376154
5	65+	2.643125

In [27]:
```
import pandas as pd
import matplotlib.pyplot as plt
```

```
import matplotlib.ticker as mtick
sfm = pd.read_csv("dataset_2019_2022.csv", parse_dates=['transaction_date'])

recent_months = sfm[(sfm['transaction_date'] >= '2022-01-01') &
                    (sfm['transaction_date'] <= '2022-05-31')].copy()

recent_months['price'] = pd.to_numeric(recent_months['price'], errors='coerce')

recent_months['sales_revenue'] = recent_months['price']
mean_sales_by_age_band = recent_months.groupby('age_band')['sales_revenue'].mean(). fig, ax = plt.subplots(figsize=(10,

8))

mean_sales_by_age_band.plot( kind='bar', color='skyblue',
    edgecolor='black', x='age_band', y='sales_revenue',
    ax=ax
)

ax.get_legend().remove() ax.yaxis.set_major_formatter(mtick.StrMethodFormatter('${x:,.2f}'))

ymin = mean_sales_by_age_band['sales_revenue'].min() * 0.95 ymax =
mean_sales_by_age_band['sales_revenue'].max() * 1.05 plt.ylim(ymin, ymax)

plt.title('Average Sales Revenue by Age Band (Jan 2022  May 2022)') plt.xlabel('Age Band')
plt.ylabel('Average Sales Revenue') plt.xticks(rotation=60)
plt.grid(axis='y', linestyle='--', alpha=0.7) plt.tight_layout()
plt.show()
```




```
In [28]: import pandas as pd
         import numpy as np
         sfm = pd.read_csv("dataset_2019_2022.csv", parse_dates=['transaction_date']) historical_years =
         sfm[(sfm['transaction_date'] >= '2019-01-01') &
                                  (sfm['transaction_date']                    <=                    '2021-12-31')].copy()
         historical_years['sales_revenue'] = historical_years['price']
         mean_sales_of_commodity = historical_years.groupby('commodity')['sales_revenue'].mean mean_sales_of_commodity_sorted =
         mean_sales_of_commodity.sort_values(ascending=False) print(mean_sales_of_commodity_sorted)
```

```
                        commodity  sales_revenue
0  Prepaid wireless accessories      30.000000
1                Coupon/misc items      23.116389
```

2	Eye and ear care products	15.240769
3	Personal care appliances	14.600000
4	Audio/video products	14.156667
5	In-store photofinishing	13.642727
6	Infant formula	13.599565
7	Sports memorablilia	13.240000

```
8          Cosmetic accessories        12.990000
9          Glassware/dinnerware        12.890000
```

In [29]:
```
import pandas as pd
import numpy as np
import plotly.graph_objs as go

sfm = pd.read_csv("dataset_2019_2022.csv", parse_dates=['transaction_date']) sfm['transaction_date'] =
pd.to_datetime(sfm['transaction_date'], errors='coerce')

historical_years = sfm[(sfm['transaction_date'] >= '2019-01-01') &
                       (sfm['transaction_date'] <= '2021-12-31')].copy()

historical_years['sales_revenue'] = historical_years['price'] historical_years['product'] =
historical_years['commodity']

mean_sale_of_product = historical_years.groupby('product')['sales_revenue'].mean()

mean_sale_df = mean_sale_of_product.sort_values(ascending=False).head(10).reset_index mean_sale_df['sales_revenue'] =
mean_sale_df['sales_revenue'].round(2)

fig = go.Figure(go.Funnel( y=mean_sale_df['product'],
    x=mean_sale_df['sales_revenue'], texttemplate="%{label}: $%{value:.2f}",
    textinfo="value+label"
))

fig.update_layout(
    title_text='Average Sales Revenue by Product: Top 10 (Jan 2019 Dec 2021)', title_x=0.5
)

fig.show()
```

In [30]:
```
import pandas as pd
import numpy as np
sfm = pd.read_csv("dataset_2019_2022.csv", parse_dates=['transaction_date']) recent_months =
sfm[(sfm['transaction_date'] >= '2022-01-01') &
                       (sfm['transaction_date'] <= '2022-05-31')].copy() recent_months['sales_revenue'] =
recent_months['price']
mean_sales_of_commodity = recent_months.groupby('commodity')['sales_revenue'].mean() mean_sales_of_commodity_sorted =
mean_sales_of_commodity.sort_values(ascending=False) print(mean_sales_of_commodity_sorted)
```

		commodity	sales_revenue
0	Infant	formula	21.072857

1	Halloween	14.990000
2	Domestic wine	13.985000
3	Smoked meats	12.290667
4	Cigarettes	11.486667
5	Floral plants	11.024483
6	Diapers	10.456667
7	Prepaid wireless accessories	10.000000
8	Party trays	9.990000
9	Audio/video products	9.990000

In [31]:
```
import pandas as pd
import numpy as np
import plotly.graph_objs as go

sfm = pd.read_csv("dataset_2019_2022.csv", parse_dates=['transaction_date']) sfm['transaction_date'] =
pd.to_datetime(sfm['transaction_date'], errors='coerce') recent_months = sfm[(sfm['transaction_date'] >= '2022-01-01')
&
                                (sfm['transaction_date'] <= '2022-05-31')].copy()

recent_months['sales_revenue'] = recent_months['price'] recent_months['product'] =
recent_months['commodity']

mean_sale_of_product = recent_months.groupby('product')['sales_revenue'].mean()

mean_sale_df = mean_sale_of_product.sort_values(ascending=False).head(10).reset_index mean_sale_df['sales_revenue'] =
mean_sale_df['sales_revenue'].round(2)

fig = go.Figure(go.Funnel( y=mean_sale_df['product'],
    x=mean_sale_df['sales_revenue'], texttemplate="%{label}: $%{value:.2f}",
    textinfo="value+label"
))

fig.update_layout(
    title_text='Average Sales Revenue by Product: Top 10 (Jan 2022 - May 2022)', title_x=0.5
)

fig.show()
```

In [32]:
```
import pandas as pd
import numpy as np
```

```
sfm = pd.read_csv("dataset_2019_2022.csv", parse_dates=['transaction_date']) historical_years =
sfm[(sfm['transaction_date'] >= '2019-01-01') &
                                (sfm['transaction_date'] <= '2021-12-31')].copy() historical_years['sales_revenue'] =
historical_years['price']
```

```
mean_sales_of_department = historical_years.groupby('department')['sales_revenue'].me mean_sales_of_department_sorted =
mean_sales_of_department.sort_values(ascending=Fals print(mean_sales_of_department_sorted)
```

	department	sales_revenue
0	Floral	7.068932
1	Cosmetics	5.811010
2	Seafood	5.802921
3	Meat	4.780174
4	Deli	4.095278
5	Pharmaceutical	3.845978
6	Nutrition	3.535556
7	Salad Bar	3.348155
8	Pastry	2.924200
9	Grocery	2.491671

In [33]:
```
import pandas as pd
import numpy as np
import plotly.graph_objs as go

sfm = pd.read_csv("dataset_2019_2022.csv", parse_dates=['transaction_date']) sfm['transaction_date'] =
pd.to_datetime(sfm['transaction_date'], errors='coerce') historical_years = sfm[(sfm['transaction_date'] >= '2019-01-
01') &
                                (sfm['transaction_date'] <= '2021-12-31')].copy()

historical_years['sales_revenue'] = historical_years['price']

mean_sale_by_department = historical_years.groupby('department')['sales_revenue'].mea

mean_sale_df = mean_sale_by_department.sort_values(ascending=False).head(10).reset_in mean_sale_df['sales_revenue'] =
mean_sale_df['sales_revenue'].round(2)

fig = go.Figure(go.Funnel( y=mean_sale_df['department'],
    x=mean_sale_df['sales_revenue'], texttemplate="%{label}: $%{value:.2f}",
    textinfo="value+label"
))

fig.update_layout(
    title_text='Average Sales Revenue by Department: Top 10 (Jan 2019 - Dec 2021)', title_x=0.5
)
```

```
fig.show()
```

In [34]:
```
import pandas as pd
import numpy as np
```

```
sfm = pd.read_csv("dataset_2019_2022.csv", parse_dates=['transaction_date']) recent_months =
sfm[(sfm['transaction_date'] >= '2022-01-01') &
                    (sfm['transaction_date'] <= '2022-05-31')].copy() recent_months['sales_revenue']
= recent_months['price']
mean_sales_by_department = recent_months.groupby('department')['sales_revenue'].mean( mean_sales_by_department_sorted =
mean_sales_by_department.sort_values(ascending=Fals print(mean_sales_by_department_sorted)
```

```
<bound method Series.reset_index of department Floral    11.024483
Seafood                6.859949
Meat                   5.198131
Deli                   4.547486
Cosmetics              3.950200
Pharmaceutical         3.798002
Nutrition              3.777673
Pastry                 3.223450
Salad Bar              2.794000
Grocery                2.499606
Name: sales_revenue, dtype: float64>
```

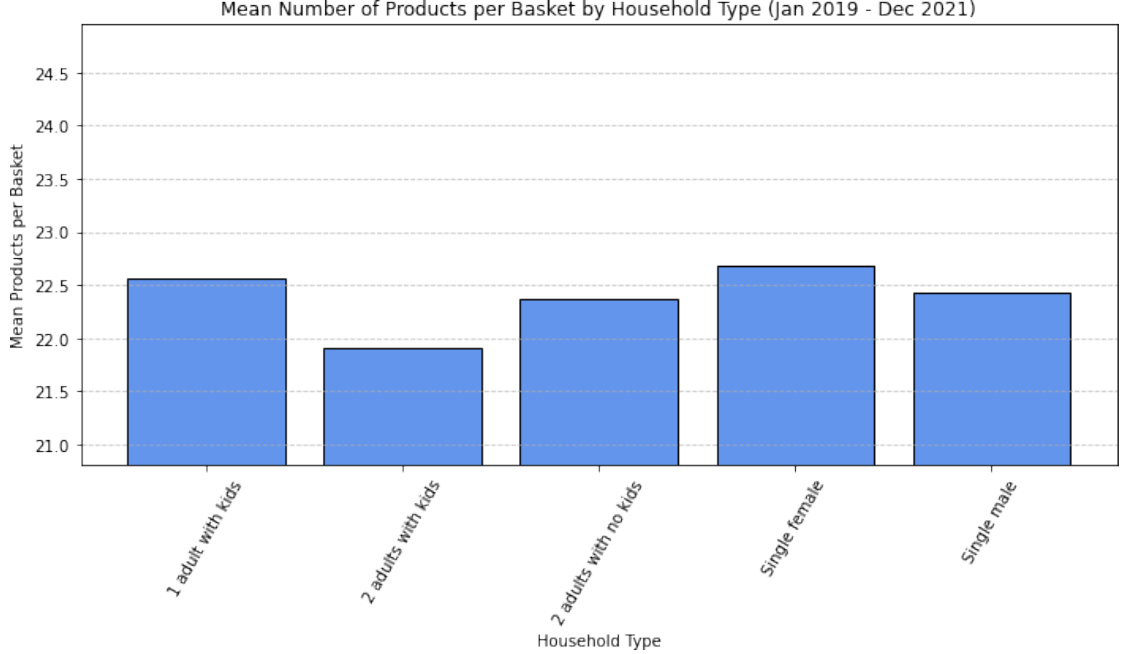
```
In [35]: import pandas as pd
         import numpy as np
         import plotly.graph_objs as go

         sfm = pd.read_csv("dataset_2019_2022.csv", parse_dates=['transaction_date'])

         recent_months = sfm[(sfm['transaction_date'] >= '2022-01-01') &
                             (sfm['transaction_date'] <= '2022-05-31')].copy() recent_months['sales_revenue'] =

         recent_months['price']

         mean_sale_by_department = recent_months.groupby('department')['sales_revenue'].mean()

         mean_sale_df = mean_sale_by_department.sort_values(ascending=False).head(10).reset_in mean_sale_df['sales_revenue'] =
         mean_sale_df['sales_revenue'].round(2)

         fig = go.Figure(go.Funnel( y=mean_sale_df['department'],
             x=mean_sale_df['sales_revenue'], texttemplate="%{label}: $%{value:.2f}",
             textinfo="value+label"
         ))
```

```
fig.update_layout(
    title_text='Average Sales Revenue by Department: Top 10 (Jan 2022 - May 2022)', title_x=0.5
```

```
)

fig.show()
```

In [37]:
```
import pandas as pd
import matplotlib.pyplot as plt
sfm = pd.read_csv("dataset_2019_2022.csv", parse_dates=['transaction_date']) historical_years =
sfm[(sfm['transaction_date'] >= '2019-01-01') &
                    (sfm['transaction_date'] <= '2021-12-31')].copy() historical_years['products'] =
historical_years['commodity'] basket_product_counts = (
    historical_years.groupby(['basket_id', 'household_type'])['products']
    .count()
    .reset_index(name='total_product_count')
)
mean_products_per_basket = ( basket_product_counts.groupby('household_type')['total_product_count']
    .mean()
    .reset_index(name='mean_products_per_basket')
)
print(mean_products_per_basket)

plt.figure(figsize=(10, 6)) plt.bar(mean_products_per_basket['household_type'],
        mean_products_per_basket['mean_products_per_basket'],    color='cornflowerblue', edgecolor='black')

ymin = mean_products_per_basket['mean_products_per_basket'].min() * 0.95 ymax =
mean_products_per_basket['mean_products_per_basket'].max() * 1.10 plt.ylim(ymin, ymax)

plt.title('Mean Number of Products per Basket by Household Type (Jan 2019 - Dec 2021) plt.xlabel('Household Type')
plt.ylabel('Mean Products per Basket') plt.xticks(rotation=60)
plt.grid(axis='y', linestyle='--', alpha=0.7) plt.tight_layout()
plt.show()
```

	household_type	mean_products_per_basket
0	1 adult with kids	22.558233
1	2 adults with kids	21.901622
2	2 adults with no kids	22.372981
3	Single female	22.682836
4	Single male	22.431034

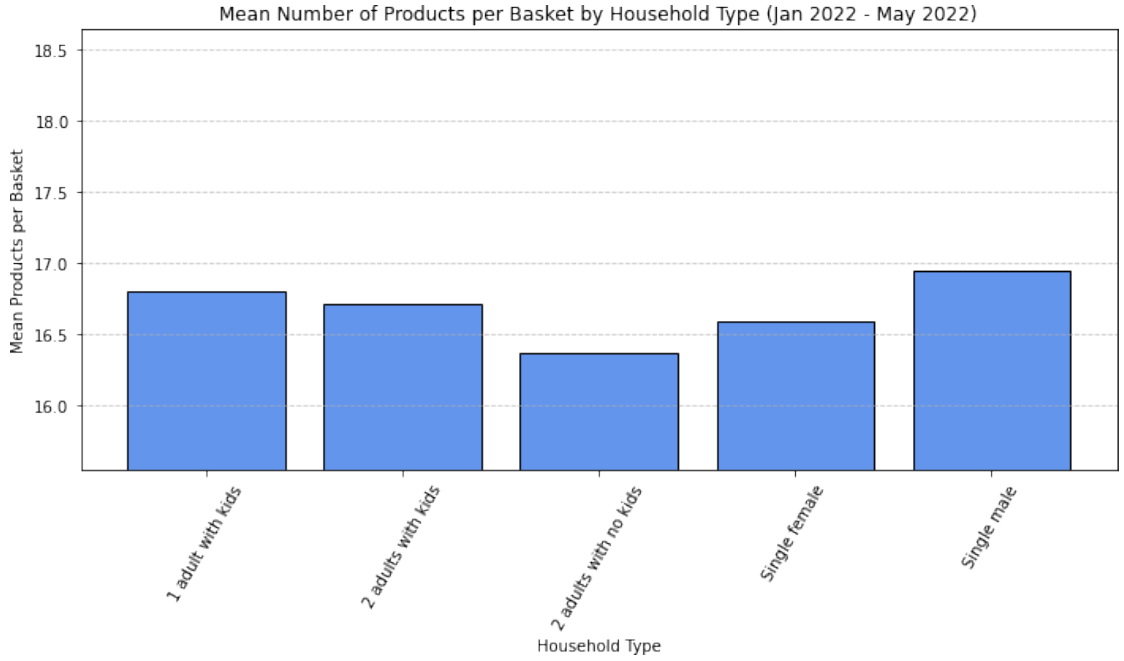


In [44]:
```
import pandas as pd
import matplotlib.pyplot as plt
sfm = pd.read_csv("dataset_2019_2022.csv", parse_dates=['transaction_date']) recent_months =
sfm[(sfm['transaction_date'] >= '2022-01-01') &
                    (sfm['transaction_date'] <= '2022-05-31')].copy() recent_months['products'] =
recent_months['commodity'] basket_product_counts = (
    recent_months.groupby(['basket_id', 'household_type'])['products']
    .count()
    .reset_index(name='total_product_count')
)
mean_products_per_basket = ( basket_product_counts.groupby('household_type')['total_product_count']
    .mean()
    .reset_index(name='mean_products_per_basket')
)
print(mean_products_per_basket)

plt.figure(figsize=(10, 6)) plt.bar(mean_products_per_basket['household_type'],
        mean_products_per_basket['mean_products_per_basket'],    color='cornflowerblue', edgecolor='black')
```

```
ymin = mean_products_per_basket['mean_products_per_basket'].min() * 0.95 ymax = mean_products_per_basket['mean_products_per_basket'].max() * 1.10 plt.ylim(ymin, ymax)
```

```
plt.title('Mean Number of Products per Basket by Household Type (Jan 2022 - May 2022) plt.xlabel('Household Type')
plt.ylabel('Mean Products per Basket') plt.xticks(rotation=60)
plt.grid(axis='y', linestyle='--', alpha=0.7) plt.tight_layout()
plt.show()
```

	household_type	mean_products_per_basket
0	1 adult with kids	16.804124
1	2 adults with kids	16.711864
2	2 adults with no kids	16.364706
3	Single female	16.594595
4	Single male	16.945946

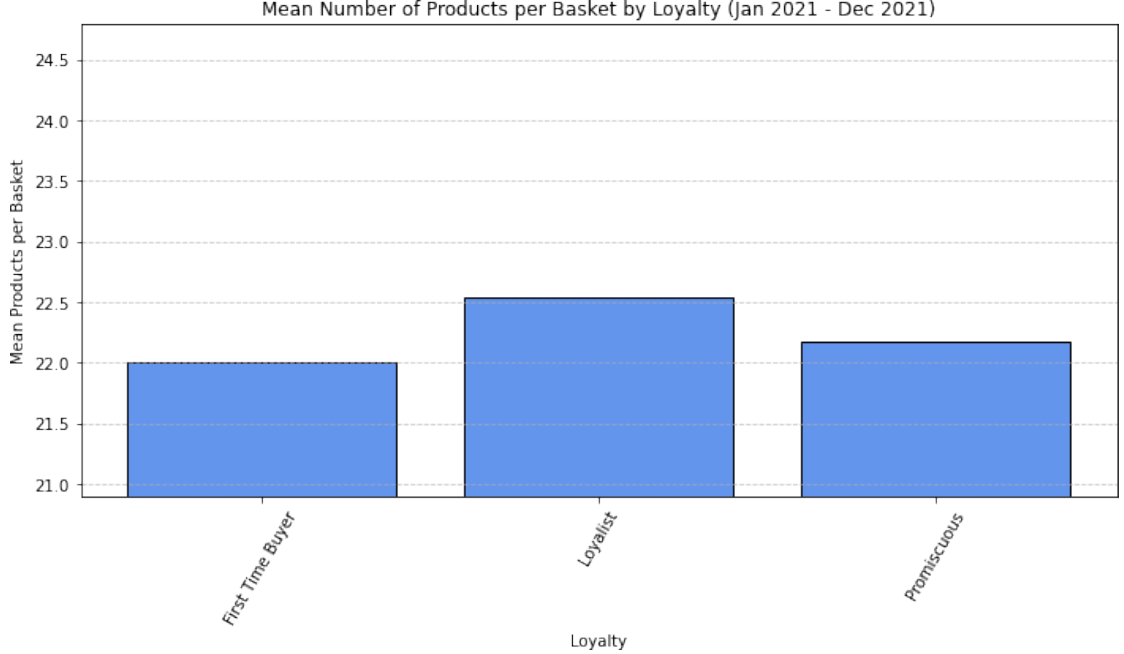


In [45]:
```
import pandas as pd
import matplotlib.pyplot as plt
sfm = pd.read_csv("dataset_2019_2022.csv", parse_dates=['transaction_date']) historical_years =
sfm[(sfm['transaction_date'] >= '2019-01-01') &
                    (sfm['transaction_date'] <= '2021-12-31')].copy() historical_years['products'] =
```

```
historical_years['commodity'] basket_product_counts = (
    historical_years.groupby(['basket_id',  'loyalty'])['products']
    .count()
```

```
        .reset_index(name='total_product_count')
    )
    mean_products_per_basket = ( basket_product_counts.groupby('loyalty')['total_product_count']
        .mean()
        .reset_index(name='mean_products_per_basket')
    )
    print(mean_products_per_basket)

    plt.figure(figsize=(10, 6)) plt.bar(mean_products_per_basket['loyalty'],
            mean_products_per_basket['mean_products_per_basket'],      color='cornflowerblue', edgecolor='black')

    ymin = mean_products_per_basket['mean_products_per_basket'].min() * 0.95 ymax =
    mean_products_per_basket['mean_products_per_basket'].max() * 1.10 plt.ylim(ymin, ymax)

    plt.title('Mean Number of Products per Basket by Loyalty (Jan 2021 - Dec 2021)') plt.xlabel('Loyalty')
    plt.ylabel('Mean Products per Basket') plt.xticks(rotation=60)
    plt.grid(axis='y', linestyle='--', alpha=0.7) plt.tight_layout()
    plt.show()
```

	loyalty	mean_products_per_basket
0	First Time Buyer	22.000000
1	Loyalist	22.535858
2	Promiscuous	22.166017

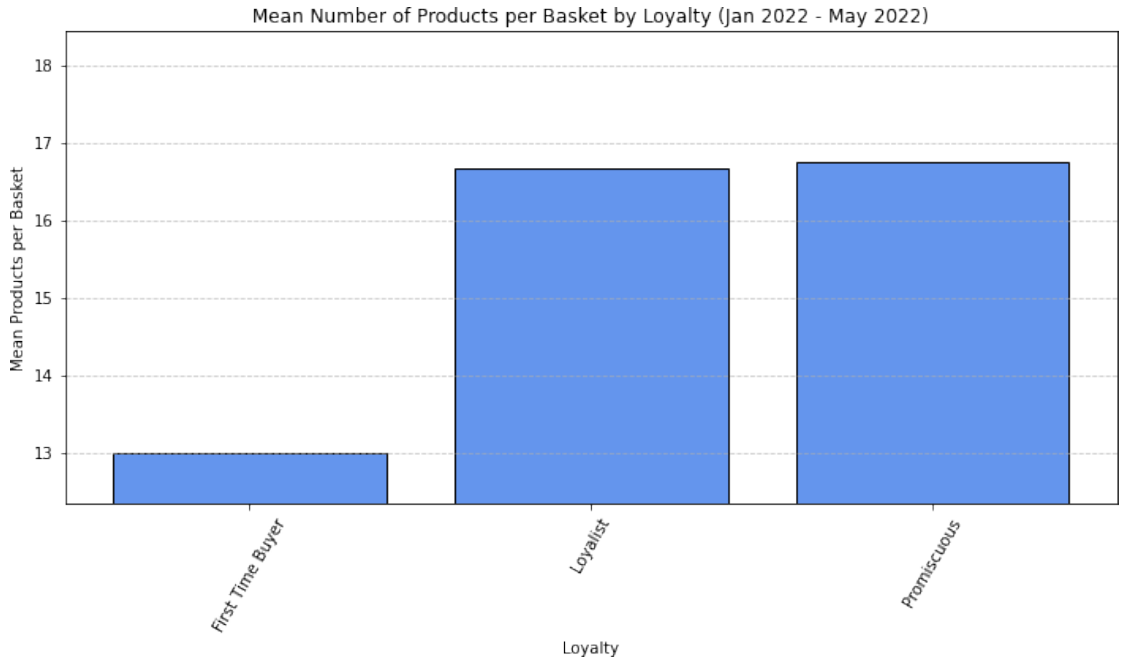


In [46]:
```
import pandas as pd
import matplotlib.pyplot as plt
sfm = pd.read_csv("dataset_2019_2022.csv", parse_dates=['transaction_date']) recent_months =
sfm[(sfm['transaction_date'] >= '2022-01-01') &
                                (sfm['transaction_date'] <= '2022-05-31')].copy() recent_months['products'] =
recent_months['commodity'] basket_product_counts = (
    recent_months.groupby(['basket_id', 'loyalty'])['products']
    .count()
    .reset_index(name='total_product_count')
)
mean_products_per_basket = ( basket_product_counts.groupby('loyalty')['total_product_count']
    .mean()
    .reset_index(name='mean_products_per_basket')
)
print(mean_products_per_basket)

plt.figure(figsize=(10, 6)) plt.bar(mean_products_per_basket['loyalty'],
        mean_products_per_basket['mean_products_per_basket'],    color='cornflowerblue', edgecolor='black')
```

```
ymin = mean_products_per_basket['mean_products_per_basket'].min() * 0.95 ymax = mean_products_per_basket['mean_products_per_basket'].max() * 1.10 plt.ylim(ymin, ymax)
```

```
plt.title('Mean Number of Products per Basket by Loyalty (Jan 2022 - May 2022)') plt.xlabel('Loyalty')
plt.ylabel('Mean Products per Basket') plt.xticks(rotation=60)
plt.grid(axis='y', linestyle='--', alpha=0.7) plt.tight_layout()
plt.show()
```

	loyalty	mean_products_per_basket
0	First Time Buyer	13.000000
1	Loyalist	16.672515
2	Promiscuous	16.755274

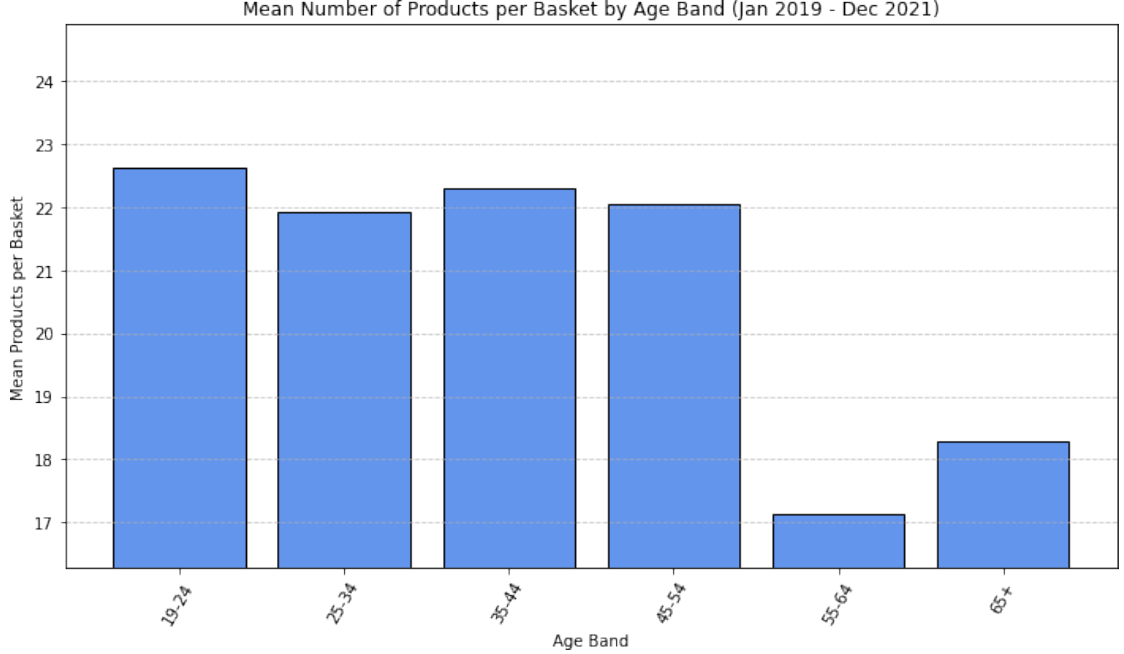


In [47]:
```
import pandas as pd
import matplotlib.pyplot as plt
sfm = pd.read_csv("dataset_2019_2022.csv", parse_dates=['transaction_date']) historical_years =
sfm[(sfm['transaction_date'] >= '2019-01-01') &
                    (sfm['transaction_date'] <= '2021-12-31')].copy() historical_years['products'] =
historical_years['commodity'] basket_product_counts = (
    historical_years.groupby(['basket_id', 'age_band'])['products']
```

```
        .count()
        .reset_index(name='total_product_count')
    )
```

```
mean_products_per_basket = ( basket_product_counts.groupby('age_band')['total_product_count']
    .mean()
    .reset_index(name='mean_products_per_basket')
)
print(mean_products_per_basket)

plt.figure(figsize=(10, 6)) plt.bar(mean_products_per_basket['age_band'],
        mean_products_per_basket['mean_products_per_basket'],  color='cornflowerblue', edgecolor='black')

ymin = mean_products_per_basket['mean_products_per_basket'].min() * 0.95 ymax =
mean_products_per_basket['mean_products_per_basket'].max() * 1.10 plt.ylim(ymin, ymax)

plt.title('Mean Number of Products per Basket by Age Band (Jan 2019 - Dec 2021)') plt.xlabel('Age Band')
plt.ylabel('Mean Products per Basket') plt.xticks(rotation=60)
plt.grid(axis='y', linestyle='--', alpha=0.7) plt.tight_layout()
plt.show()
```

	age_band	mean_products_per_basket
0	19-24	22.636660
1	25-34	21.915870
2	35-44	22.300000
3	45-54	22.038037
4	55-64	17.142857
5	65+	18.285714

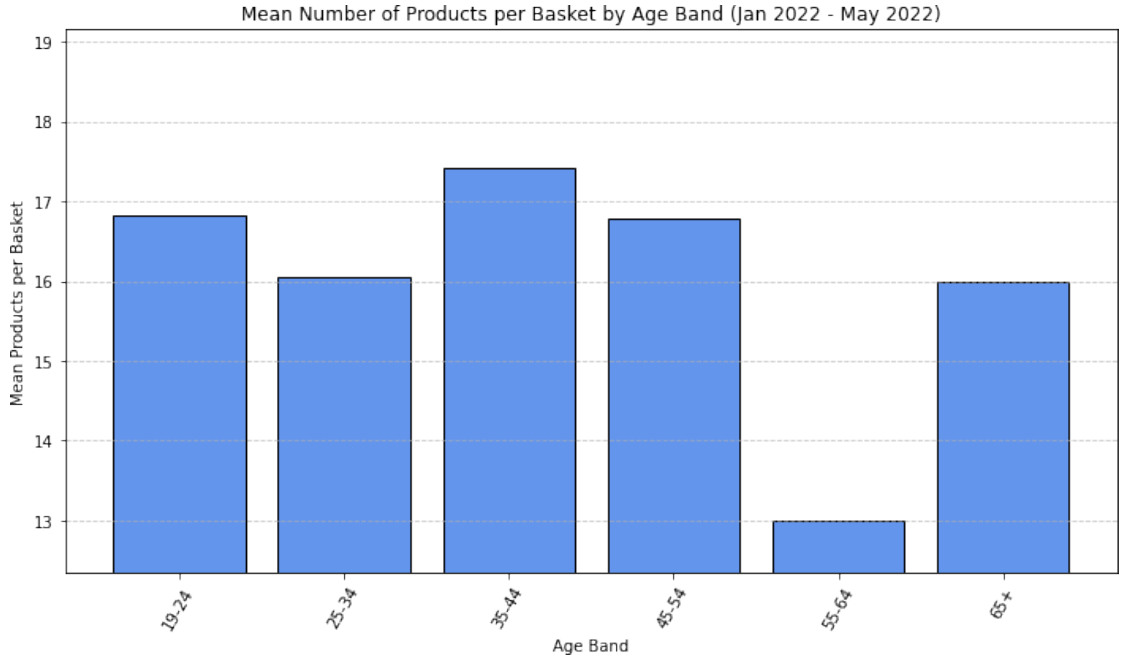


In [48]:
```
import pandas as pd
import matplotlib.pyplot as plt
sfm = pd.read_csv("dataset_2019_2022.csv", parse_dates=['transaction_date']) recent_months =
sfm[(sfm['transaction_date'] >= '2022-01-01') &
                                (sfm['transaction_date'] <= '2022-05-31')].copy() recent_months['products'] =
recent_months['commodity'] basket_product_counts = (
    recent_months.groupby(['basket_id', 'age_band'])['products']
    .count()
    .reset_index(name='total_product_count')
)
mean_products_per_basket = ( basket_product_counts.groupby('age_band')['total_product_count']
    .mean()
    .reset_index(name='mean_products_per_basket')
)
print(mean_products_per_basket)

plt.figure(figsize=(10, 6)) plt.bar(mean_products_per_basket['age_band'],
        mean_products_per_basket['mean_products_per_basket'],    color='cornflowerblue', edgecolor='black')
```

```
ymin = mean_products_per_basket['mean_products_per_basket'].min() * 0.95 ymax = mean_products_per_basket['mean_products_per_basket'].max() * 1.10 plt.ylim(ymin, ymax)
```

```
plt.title('Mean Number of Products per Basket by Age Band (Jan 2022 - May 2022)') plt.xlabel('Age Band')
plt.ylabel('Mean Products per Basket') plt.xticks(rotation=60)
plt.grid(axis='y', linestyle='--', alpha=0.7) plt.tight_layout()
plt.show()
```

	age_band	mean_products_per_basket
0	19-24	16.814978
1	25-34	16.058824
2	35-44	17.411765
3	45-54	16.773196
4	55-64	13.000000
5	65+	16.000000




```
In [1]: import pandas as pd
        import plotly.graph_objects as go

        sfm = pd.read_csv("dataset_2019_2022.csv", parse_dates=['transaction_date']) sfm['transaction_date'] =
```

```
pd.to_datetime(sfm['transaction_date'],  errors='coerce')

historical_years  =  sfm[(sfm['transaction_date']  >=  '2019-01-01')  &
                         (sfm['transaction_date'] <= '2021-12-31')].copy()
```

```
historical_years['product'] = historical_years['commodity']

product_loyalty_counts = historical_years.groupby(['loyalty', 'product']).size().reset

top_15_products = ( product_loyalty_counts.groupby('product')['count']
    .sum()
    .sort_values(ascending=False)
    .head(15)
    .index
)

top_products_df = product_loyalty_counts[product_loyalty_counts['product'].isin(top_15 heatmap_data =

top_products_df.pivot_table(index='loyalty', columns='product', values= text_labels = heatmap_data.applymap(lambda x:

f'{x:.0f}')

fig = go.Figure(data=go.Heatmap( z=heatmap_data.values,
    x=heatmap_data.columns.tolist(), y=heatmap_data.index.tolist(),
    text=text_labels.values, hoverinfo='text', colorscale='Blues',
    colorbar=dict(title="Count of Purchases")
))

fig.update_layout(
    title='Heatmap of Top 15 Products by Loyalty Type (Jan 2019 Dec 2021)', xaxis_title='Product',
    yaxis_title='Loyalty Type', xaxis=dict(tickangle=60),
    autosize=False,
    height=600, width=1000
)

fig.show()
```

In [2]:
```
import pandas as pd
import plotly.graph_objects as go

sfm = pd.read_csv("dataset_2019_2022.csv", parse_dates=['transaction_date']) sfm['transaction_date'] =
pd.to_datetime(sfm['transaction_date'], errors='coerce')

recent_months = sfm[(sfm['transaction_date'] >= '2022-01-01') &
```

```
                                (sfm['transaction_date'] <= '2022-05-30')].copy() recent_months['product'] =

recent_months['commodity']

product_loyalty_counts = recent_months.groupby(['loyalty', 'product']).size().reset_in

top_15_products = ( product_loyalty_counts.groupby('product')['count']
    .sum()
    .sort_values(ascending=False)
    .head(15)
    .index
)

top_products_df = product_loyalty_counts[product_loyalty_counts['product'].isin(top_15 heatmap_data =

top_products_df.pivot_table(index='loyalty', columns='product', values= text_labels = heatmap_data.applymap(lambda x:

f'{x:.0f}')

fig = go.Figure(data=go.Heatmap( z=heatmap_data.values,
    x=heatmap_data.columns.tolist(), y=heatmap_data.index.tolist(),
    text=text_labels.values, hoverinfo='text', colorscale='Blues',
    colorbar=dict(title="Count of Purchases")
))

fig.update_layout(
    title='Heatmap of Top 15 Products by Loyalty Type (Jan 2022 - May 2022)', xaxis_title='Product',
    yaxis_title='Loyalty Type', xaxis=dict(tickangle=60),
    autosize=False,
    height=600, width=1000
)

fig.show()
```

In [3]:
```
import pandas as pd
import plotly.graph_objects as go

sfm = pd.read_csv("dataset_2019_2022.csv", parse_dates=['transaction_date']) sfm['transaction_date'] =
pd.to_datetime(sfm['transaction_date'], errors='coerce')
```

```
historical_years = sfm[(sfm['transaction_date'] >= '2019-01-01') &
                       (sfm['transaction_date'] <= '2021-12-31')].copy() historical_years['product'] =

historical_years['commodity'] product_household_type_counts = historical_years.groupby(['household_type', 'product']

top_15_products = ( product_household_type_counts.groupby('product')['count']
    .sum()
    .sort_values(ascending=False)
    .head(15)
    .index
)

top_products_df = product_household_type_counts[product_household_type_counts['product heatmap_data =

top_products_df.pivot_table(index='household_type', columns='product', text_labels = heatmap_data.applymap(lambda x:

f'{x:.0f}')

fig = go.Figure(data=go.Heatmap( z=heatmap_data.values,
    x=heatmap_data.columns.tolist(), y=heatmap_data.index.tolist(),
    text=text_labels.values, hoverinfo='text', colorscale='Blues',
    colorbar=dict(title="Count of Purchases")
))

fig.update_layout(
    title='Heatmap of Top 15 Products by Household Type (Jan 2019Dec 2021)', xaxis_title='Product',
    yaxis_title='Household Type', xaxis=dict(tickangle=60),
    autosize=False,
    height=600, width=1000
)

fig.show()
```

In [4]:
```
import pandas as pd
import plotly.graph_objects as go

sfm = pd.read_csv("dataset_2019_2022.csv", parse_dates=['transaction_date']) sfm['transaction_date'] =
pd.to_datetime(sfm['transaction_date'], errors='coerce')
```

```
recent_months = sfm[(sfm['transaction_date'] >= '2022-01-01') &
                    (sfm['transaction_date'] <= '2022-05-31')].copy() recent_months['product'] =

recent_months['commodity']

product_household_type_counts = recent_months.groupby(['household_type', 'product']).s

top_15_products = ( product_household_type_counts.groupby('product')['count']
    .sum()
    .sort_values(ascending=False)
    .head(15)
    .index
)

top_products_df = product_household_type_counts[product_household_type_counts['product  heatmap_data =

top_products_df.pivot_table(index='household_type', columns='product', text_labels = heatmap_data.applymap(lambda x:

f'{x:.0f}')

fig = go.Figure(data=go.Heatmap( z=heatmap_data.values,
    x=heatmap_data.columns.tolist(), y=heatmap_data.index.tolist(),
    text=text_labels.values, hoverinfo='text', colorscale='Blues',
    colorbar=dict(title="Count of Purchases")
))

fig.update_layout(
    title='Heatmap of Top 15 Products by Household Type (Jan 2022 - May 2022)', xaxis_title='Product',
    yaxis_title='Household Type', xaxis=dict(tickangle=60),
    autosize=False,
    height=600, width=1000
)

fig.show()
```

In [5]:
```
import pandas as pd
import plotly.graph_objects as go

sfm = pd.read_csv("dataset_2019_2022.csv", parse_dates=['transaction_date'])
```

```
sfm['transaction_date'] = pd.to_datetime(sfm['transaction_date'], errors='coerce')

historical_years = sfm[(sfm['transaction_date'] >= '2019-01-01') &
                       (sfm['transaction_date'] <= '2021-12-31')].copy() historical_years['product'] =

historical_years['commodity']

product_age_band_counts = historical_years.groupby(['age_band', 'product']).size().res

top_15_products = ( product_age_band_counts.groupby('product')['count']
    .sum()
    .sort_values(ascending=False)
    .head(15)
    .index
)

top_products_df = product_age_band_counts[product_age_band_counts['product'].isin(top_ heatmap_data =

top_products_df.pivot_table(index='age_band', columns='product', values text_labels = heatmap_data.applymap(lambda

x: f'{x:.0f}')

fig = go.Figure(data=go.Heatmap( z=heatmap_data.values,
    x=heatmap_data.columns.tolist(), y=heatmap_data.index.tolist(),
    text=text_labels.values, hoverinfo='text', colorscale='Blues',
    colorbar=dict(title="Count of Purchases")
))

fig.update_layout(
    title='Heatmap of Top 15 Products by Age Band (Jan 2019 Dec 2021)', xaxis_title='Product',
    yaxis_title='Age Band', xaxis=dict(tickangle=60),
    autosize=False, height=600,
    width=1000
)

fig.show()
```

In [6]:
```
import pandas as pd
import plotly.graph_objects as go
```

```
sfm = pd.read_csv("dataset_2019_2022.csv", parse_dates=['transaction_date']) sfm['transaction_date'] =
pd.to_datetime(sfm['transaction_date'], errors='coerce')

recent_months = sfm[(sfm['transaction_date'] >= '2022-01-01') &
                    (sfm['transaction_date'] <= '2022-05-31')].copy() recent_months['product'] =

recent_months['commodity']

product_age_band_counts = recent_months.groupby(['age_band', 'product']).size().reset_

top_15_products = ( product_age_band_counts.groupby('product')['count']
    .sum()
    .sort_values(ascending=False)
    .head(15)
    .index
)

top_products_df = product_age_band_counts[product_age_band_counts['product'].isin(top_ heatmap_data =

top_products_df.pivot_table(index='age_band', columns='product', values text_labels = heatmap_data.applymap(lambda

x: f'{x:.0f}')

fig = go.Figure(data=go.Heatmap( z=heatmap_data.values,
    x=heatmap_data.columns.tolist(), y=heatmap_data.index.tolist(),
    text=text_labels.values, hoverinfo='text', colorscale='Blues',
    colorbar=dict(title="Count of Purchases")
))

fig.update_layout(
    title='Heatmap of Top 15 Products by Age Band (Jan 2022 May 2022)', xaxis_title='Product',
    yaxis_title='Age Band', xaxis=dict(tickangle=60),
    autosize=False, height=600,
    width=1000
)

fig.show()
```

In [7]:
```
import pandas as pd
import plotly.graph_objects as go
```

```
sfm = pd.read_csv("dataset_2019_2022.csv", parse_dates=['transaction_date']) sfm['transaction_date'] =
pd.to_datetime(sfm['transaction_date'], errors='coerce')

historical_years = sfm[(sfm['transaction_date'] >= '2019-01-01') &
                       (sfm['transaction_date'] <= '2021-12-31')].copy() historical_years['product'] =

historical_years['commodity']

product_brand_counts = historical_years.groupby(['brand', 'product']).size().reset_ind

top_15_products = ( product_brand_counts.groupby('product')['count']
    .sum()
    .sort_values(ascending=False)
    .head(15)
    .index
)

top_products_df = product_brand_counts[product_brand_counts['product'].isin(top_15_pro heatmap_data =

top_products_df.pivot_table(index='brand', columns='product', values='c text_labels = heatmap_data.applymap(lambda x:

f'{x:.0f}')

fig = go.Figure(data=go.Heatmap( z=heatmap_data.values,
    x=heatmap_data.columns.tolist(), y=heatmap_data.index.tolist(),
    text=text_labels.values, hoverinfo='text', colorscale='Blues',
    colorbar=dict(title="Count of Purchases")
))

fig.update_layout(
    title='Heatmap of Top 15 Products by Brand (Jan 2019 Dec 2021)', xaxis_title='Product',
    yaxis_title='Brand', xaxis=dict(tickangle=60),
    autosize=False, height=600,
    width=750
)

fig.show()
```

In [28]:
```
import pandas as pd
```

```
import plotly.graph_objects as go

sfm = pd.read_csv("dataset_2019_2022.csv", parse_dates=['transaction_date']) sfm['transaction_date'] =
pd.to_datetime(sfm['transaction_date'], errors='coerce')

recent_months = sfm[(sfm['transaction_date'] >= '2022-01-01') &
                    (sfm['transaction_date'] <= '2022-05-31')].copy() recent_months['product'] =

recent_months['commodity']

product_brand_counts = recent_months.groupby(['brand', 'product']).size().reset_index

top_15_products = ( product_brand_counts.groupby('product')['count']
    .sum()
    .sort_values(ascending=False)
    .head(15)
    .index
)

top_products_df = product_brand_counts[product_brand_counts['product'].isin(top_15_pr heatmap_data =

top_products_df.pivot_table(index='brand', columns='product', values=' text_labels = heatmap_data.applymap(lambda x:

f'{x:.0f}')

fig = go.Figure(data=go.Heatmap( z=heatmap_data.values,
    x=heatmap_data.columns.tolist(), y=heatmap_data.index.tolist(),
    text=text_labels.values, hoverinfo='text', colorscale='Blues',
    colorbar=dict(title="Count of Purchases")
))

fig.update_layout(
    title='Heatmap of Top 15 Products by Brand (Jan 2022 - May 2022)', xaxis_title='Product',
    yaxis_title='Brand', xaxis=dict(tickangle=60),
    autosize=False, height=600,
    width=750
)

fig.show()
```

In [8]:
```
import pandas as pd

    sfm = pd.read_csv("dataset_2019_2022.csv", parse_dates=['transaction_date']) sfm['transaction_date'] =
    pd.to_datetime(sfm['transaction_date'], errors='coerce') sfm['price'] = pd.to_numeric(sfm['price'], errors='coerce')
    sfm['loyalty'] = sfm['loyalty'].str.strip()

    promiscuous_data = sfm[sfm['loyalty'].str.lower() == 'promiscuous']

    historical = promiscuous_data[ (promiscuous_data['transaction_date'] >= '2019-01-01') &
        (promiscuous_data['transaction_date'] <= '2021-12-31')
    ]

    recent = promiscuous_data[ (promiscuous_data['transaction_date'] >= '2022-01-01') &
        (promiscuous_data['transaction_date'] <= '2022-05-31')
    ]

    historical_avg = historical['price'].mean() recent_avg = recent['price'].mean()
    change_amount = recent_avg - historical_avg
    change_percent = (change_amount / historical_avg) * 100 if historical_avg else None

    if change_amount > 0: direction = "increase"
    else:
        direction = "decrease"

    print(f"Average spend (20192021): ${historical_avg:.2f}") print(f"Average spend (JanMay
    2022): ${recent_avg:.2f}")
    print(f"{direction.capitalize()} in average spend: ${abs(change_amount):.2f} ({abs(cha
```

```
Average spend (20192021): $3.22 Average spend (JanMay 2022): $3.39
Increase in average spend: $0.17 (5.39%)
```

In [52]:
```
import pandas as pd

    sfm = pd.read_csv("dataset_2019_2022.csv", parse_dates=['transaction_date']) sfm['transaction_date'] =
    pd.to_datetime(sfm['transaction_date'], errors='coerce') sfm['price'] = pd.to_numeric(sfm['price'], errors='coerce')
    sfm['loyalty'] = sfm['loyalty'].str.strip()

    loyalist_data = sfm[sfm['loyalty'].str.lower() == 'loyalist'] historical = loyalist_data[
```

```
    (loyalist_data['transaction_date'] >= '2019-01-01') & (loyalist_data['transaction_date'] <= '2021-12-31')
]

recent = loyalist_data[  (loyalist_data['transaction_date'] >= '2022-01-01') &
    (loyalist_data['transaction_date'] <= '2022-05-31')
]

historical_avg = historical['price'].mean() recent_avg = recent['price'].mean()
change_amount = recent_avg - historical_avg
change_percent = (change_amount / historical_avg) * 100 if historical_avg else None

if change_amount > 0: direction = "increase"
else:
    direction = "decrease"

print(f"Average spend (20192021): ${historical_avg:.2f}") print(f"Average spend (JanMay
2022): ${recent_avg:.2f}")
print(f"{direction.capitalize()} in average spend: ${abs(change_amount):.2f} ({abs(ch
```

```
Average spend (20192021): $3.22 Average spend (JanMay 2022): $3.36
Increase in average spend: $0.14 (4.48%)
```

In [53]:
```
import pandas as pd

sfm = pd.read_csv("dataset_2019_2022.csv", parse_dates=['transaction_date']) sfm['transaction_date'] =
pd.to_datetime(sfm['transaction_date'], errors='coerce') sfm['price'] = pd.to_numeric(sfm['price'], errors='coerce')
sfm['loyalty'] = sfm['loyalty'].str.strip()

first_time_buyer_data = sfm[sfm['loyalty'].str.lower() == 'first time buyer']

historical = first_time_buyer_data[ (first_time_buyer_data['transaction_date'] >= '2019-01-01') &
    (first_time_buyer_data['transaction_date'] <= '2021-12-31')
]

recent = first_time_buyer_data[ (first_time_buyer_data['transaction_date'] >= '2022-01-01') &
    (first_time_buyer_data['transaction_date'] <= '2022-05-31')
]

historical_avg = historical['price'].mean()
```

```
recent_avg = recent['price'].mean() change_amount = recent_avg - historical_avg
change_percent = (change_amount / historical_avg) * 100 if historical_avg else None

if change_amount > 0: direction = "increase"
else:
    direction = "decrease"

print(f"Average spend (20192021): ${historical_avg:.2f}") print(f"Average spend (JanMay
2022): ${recent_avg:.2f}")
print(f"{direction.capitalize()} in average spend: ${abs(change_amount):.2f} ({abs(ch
```

```
Average spend (20192021): $3.10 Average spend (JanMay 2022): $3.63
Increase in average spend: $0.53 (17.22%)
```

In [54]:
```
import pandas as pd

sfm = pd.read_csv("dataset_2019_2022.csv", parse_dates=['transaction_date']) sfm['transaction_date'] =
pd.to_datetime(sfm['transaction_date'], errors='coerce') sfm['price'] = pd.to_numeric(sfm['price'], errors='coerce')
sfm['household_type'] = sfm['household_type'].str.strip()

two_adults_with_kids_data = sfm[sfm['household_type'].str.lower() == '2 adults with k

historical = two_adults_with_kids_data[ (two_adults_with_kids_data['transaction_date'] >= '2019-01-01') &
    (two_adults_with_kids_data['transaction_date'] <= '2021-12-31')
]

recent = two_adults_with_kids_data[ (two_adults_with_kids_data['transaction_date'] >= '2022-01-01') &
    (two_adults_with_kids_data['transaction_date'] <= '2022-05-31')
]

historical_avg = historical['price'].mean() recent_avg = recent['price'].mean()
change_amount = recent_avg - historical_avg
change_percent = (change_amount / historical_avg) * 100 if historical_avg else None

if change_amount > 0: direction = "increase"
else:
    direction = "decrease"

print(f"Average spend (20192021): ${historical_avg:.2f}")
```

```
print(f"Average spend (JanMay 2022): ${recent_avg:.2f}") print(f"{direction.capitalize()} in average spend: ${abs(change_amount):.2f} ({abs(ch
```

```
Average spend (20192021): $3.19 Average spend (JanMay 2022): $3.40
Increase in average spend: $0.21 (6.50%)
```

In [55]:
```
import pandas as pd

sfm = pd.read_csv("dataset_2019_2022.csv", parse_dates=['transaction_date']) sfm['transaction_date'] = pd.to_datetime(sfm['transaction_date'], errors='coerce') sfm['price'] = pd.to_numeric(sfm['price'], errors='coerce') sfm['household_type'] = sfm['household_type'].str.strip()

two_adults_with_no_kids_data = sfm[sfm['household_type'].str.lower() == '2 adults wit

historical = two_adults_with_no_kids_data[ (two_adults_with_no_kids_data['transaction_date'] >= '2019-01-01') & (two_adults_with_no_kids_data['transaction_date'] <= '2021-12-31')
]

recent = two_adults_with_no_kids_data[ (two_adults_with_no_kids_data['transaction_date'] >= '2022-01-01') & (two_adults_with_no_kids_data['transaction_date'] <= '2022-05-31')
]

historical_avg = historical['price'].mean() recent_avg = recent['price'].mean()
change_amount = recent_avg - historical_avg
change_percent = (change_amount / historical_avg) * 100 if historical_avg else None

if change_amount > 0: direction = "increase"
else:
    direction = "decrease"

print(f"Average spend (20192021): ${historical_avg:.2f}") print(f"Average spend (JanMay 2022): ${recent_avg:.2f}")
print(f"{direction.capitalize()} in average spend: ${abs(change_amount):.2f} ({abs(ch
```

```
Average spend (20192021): $3.24 Average spend (JanMay 2022): $3.40
Increase in average spend: $0.16 (5.08%)
```

In [56]:
```
import pandas as pd
```

```
sfm = pd.read_csv("dataset_2019_2022.csv", parse_dates=['transaction_date'])
```

```
sfm['transaction_date'] = pd.to_datetime(sfm['transaction_date'], errors='coerce') sfm['price'] =
pd.to_numeric(sfm['price'], errors='coerce')
sfm['household_type']   =   sfm['household_type'].str.strip()

single_female_data = sfm[sfm['household_type'].str.lower() == 'single female']

historical = single_female_data[ (single_female_data['transaction_date'] >= '2019-01-01') &
    (single_female_data['transaction_date'] <= '2021-12-31')
]

recent = single_female_data[ (single_female_data['transaction_date'] >= '2022-01-01') &
    (single_female_data['transaction_date'] <= '2022-05-31')
]

historical_avg = historical['price'].mean() recent_avg = recent['price'].mean()
change_amount = recent_avg - historical_avg
change_percent = (change_amount / historical_avg) * 100 if historical_avg else None

if change_amount > 0: direction = "increase"
else:
    direction = "decrease"

print(f"Average spend (20192021): ${historical_avg:.2f}") print(f"Average spend (JanMay
2022): ${recent_avg:.2f}")
print(f"{direction.capitalize()} in average spend: ${abs(change_amount):.2f} ({abs(ch
```

```
Average spend (20192021): $3.23 Average spend (JanMay 2022): $3.41
Increase in average spend: $0.18 (5.54%)
```

In [57]:
```
import pandas as pd

sfm = pd.read_csv("dataset_2019_2022.csv", parse_dates=['transaction_date']) sfm['transaction_date'] =
pd.to_datetime(sfm['transaction_date'], errors='coerce') sfm['price'] = pd.to_numeric(sfm['price'], errors='coerce')
sfm['household_type']   =   sfm['household_type'].str.strip()

single_male_data = sfm[sfm['household_type'].str.lower() == 'single male']

historical = single_male_data[ (single_male_data['transaction_date'] >= '2019-01-01') &
    (single_male_data['transaction_date'] <= '2021-12-31')
]
```

```
recent = single_male_data[ (single_male_data['transaction_date'] >= '2022-01-01') &
    (single_male_data['transaction_date'] <= '2022-05-31')
]

historical_avg = historical['price'].mean() recent_avg = recent['price'].mean()
change_amount = recent_avg - historical_avg
change_percent = (change_amount / historical_avg) * 100 if historical_avg else None

if change_amount > 0: direction = "increase"
else:
    direction = "decrease"

print(f"Average spend (20192021): ${historical_avg:.2f}") print(f"Average spend (JanMay
2022): ${recent_avg:.2f}")
print(f"{direction.capitalize()} in average spend: ${abs(change_amount):.2f} ({abs(ch
```

```
Average spend (20192021): $3.26 Average spend (JanMay 2022): $3.36
Increase in average spend: $0.11 (3.26%)
```

In [58]:
```
import pandas as pd

sfm = pd.read_csv("dataset_2019_2022.csv", parse_dates=['transaction_date']) sfm['transaction_date'] =
pd.to_datetime(sfm['transaction_date'], errors='coerce') sfm['price'] = pd.to_numeric(sfm['price'], errors='coerce')
sfm['household_type'] = sfm['household_type'].str.strip()

one_adult_with_kids_data = sfm[sfm['household_type'].str.lower() == '1 adult with kid

historical = one_adult_with_kids_data[ (one_adult_with_kids_data['transaction_date'] >= '2019-01-01') &
    (one_adult_with_kids_data['transaction_date'] <= '2021-12-31')
]

recent = one_adult_with_kids_data[ (one_adult_with_kids_data['transaction_date'] >= '2022-01-01') &
    (one_adult_with_kids_data['transaction_date'] <= '2022-05-31')
]

historical_avg = historical['price'].mean() recent_avg = recent['price'].mean()
change_amount = recent_avg - historical_avg
change_percent = (change_amount / historical_avg) * 100 if historical_avg else None
```

```
if change_amount > 0: direction = "increase"
else:
    direction = "decrease"

print(f"Average spend (20192021): ${historical_avg:.2f}") print(f"Average spend (JanMay
2022): ${recent_avg:.2f}")
print(f"{direction.capitalize()} in average spend: ${abs(change_amount):.2f} ({abs(ch
```

```
Average spend (20192021): $3.19 Average spend (JanMay 2022): $3.34
Increase in average spend: $0.15 (4.61%)
```

In [59]:
```
import pandas as pd

sfm = pd.read_csv("dataset_2019_2022.csv", parse_dates=['transaction_date']) sfm['transaction_date'] =
pd.to_datetime(sfm['transaction_date'], errors='coerce') sfm['price'] = pd.to_numeric(sfm['price'], errors='coerce')
sfm['age_band'] = sfm['age_band'].str.strip()

first_age_band_data = sfm[sfm['age_band'].str.lower() == '19-24']

historical = first_age_band_data[ (first_age_band_data['transaction_date'] >= '2019-01-01') &
    (first_age_band_data['transaction_date'] <= '2021-12-31')
]

recent = first_age_band_data[ (first_age_band_data['transaction_date'] >= '2022-01-01') &
    (first_age_band_data['transaction_date'] <= '2022-05-31')
]

historical_avg = historical['price'].mean() recent_avg = recent['price'].mean()
change_amount = recent_avg - historical_avg
change_percent = (change_amount / historical_avg) * 100 if historical_avg else None

if change_amount > 0: direction = "increase"
else:
    direction = "decrease"

print(f"Average spend (20192021): ${historical_avg:.2f}") print(f"Average spend (JanMay
2022): ${recent_avg:.2f}")
print(f"{direction.capitalize()} in average spend: ${abs(change_amount):.2f} ({abs(ch
```

```
Average spend (20192021): $3.24
```

```
Average spend (JanMay 2022): $3.43 Increase in average spend: $0.19
(5.99%)
```

In [61]:
```
import pandas as pd

    sfm = pd.read_csv("dataset_2019_2022.csv", parse_dates=['transaction_date']) sfm['transaction_date'] =
    pd.to_datetime(sfm['transaction_date'], errors='coerce') sfm['price'] = pd.to_numeric(sfm['price'], errors='coerce')
    sfm['age_band'] = sfm['age_band'].str.strip()

    third_age_band_data = sfm[sfm['age_band'].str.lower() == '35-44']

    historical = third_age_band_data[ (third_age_band_data['transaction_date'] >= '2019-01-01') &
        (third_age_band_data['transaction_date'] <= '2021-12-31')
    ]

    recent = third_age_band_data[ (third_age_band_data['transaction_date'] >= '2022-01-01') &
        (third_age_band_data['transaction_date'] <= '2022-05-31')
    ]

    historical_avg = historical['price'].mean() recent_avg = recent['price'].mean()
    change_amount = recent_avg - historical_avg
    change_percent = (change_amount / historical_avg) * 100 if historical_avg else None

    if change_amount > 0: direction = "increase"
    else:
        direction = "decrease"

    print(f"Average spend (20192021): ${historical_avg:.2f}") print(f"Average spend (JanMay
    2022): ${recent_avg:.2f}")
    print(f"{direction.capitalize()} in average spend: ${abs(change_amount):.2f} ({abs(ch
```

```
Average spend (20192021): $3.15 Average spend (JanMay 2022): $3.50
Increase in average spend: $0.35 (11.11%)
```

In [63]:
```
import pandas as pd

    sfm = pd.read_csv("dataset_2019_2022.csv", parse_dates=['transaction_date']) sfm['transaction_date'] =
    pd.to_datetime(sfm['transaction_date'], errors='coerce') sfm['price'] = pd.to_numeric(sfm['price'], errors='coerce')
    sfm['age_band'] = sfm['age_band'].str.strip()
```

```
fourth_age_band_data = sfm[sfm['age_band'].str.lower() == '45-54']

historical = fourth_age_band_data[ (fourth_age_band_data['transaction_date'] >= '2019-01-01') &
    (fourth_age_band_data['transaction_date'] <= '2021-12-31')
]

recent = fourth_age_band_data[ (fourth_age_band_data['transaction_date'] >= '2022-01-01') &
    (fourth_age_band_data['transaction_date'] <= '2022-05-31')
]

historical_avg = historical['price'].mean() recent_avg = recent['price'].mean()
change_amount = recent_avg - historical_avg
change_percent = (change_amount / historical_avg) * 100 if historical_avg else None

if change_amount > 0: direction = "increase"
else:
    direction = "decrease"

print(f"Average spend (20192021): ${historical_avg:.2f}") print(f"Average spend (JanMay
2022): ${recent_avg:.2f}")
print(f"{direction.capitalize()} in average spend: ${abs(change_amount):.2f} ({abs(ch
```

```
Average spend (20192021): $3.19 Average spend (JanMay 2022): $3.32
Increase in average spend: $0.14 (4.24%)
```

In [64]:
```
import pandas as pd

sfm = pd.read_csv("dataset_2019_2022.csv", parse_dates=['transaction_date']) sfm['transaction_date'] =
pd.to_datetime(sfm['transaction_date'], errors='coerce') sfm['price'] = pd.to_numeric(sfm['price'], errors='coerce')
sfm['age_band'] = sfm['age_band'].str.strip()

fifth_age_band_data = sfm[sfm['age_band'].str.lower() == '55-64']

historical = fifth_age_band_data[ (fifth_age_band_data['transaction_date'] >= '2019-01-01') &
    (fifth_age_band_data['transaction_date'] <= '2021-12-31')
]

recent = fifth_age_band_data[ (fifth_age_band_data['transaction_date'] >= '2022-01-01') &
```

```
    (fifth_age_band_data['transaction_date'] <= '2022-05-31')
]

historical_avg = historical['price'].mean() recent_avg = recent['price'].mean()
change_amount = recent_avg - historical_avg
change_percent = (change_amount / historical_avg) * 100 if historical_avg else None

if change_amount > 0: direction = "increase"
else:
    direction = "decrease"

print(f"Average spend (20192021): ${historical_avg:.2f}") print(f"Average spend (JanMay
2022): ${recent_avg:.2f}")
print(f"{direction.capitalize()} in average spend: ${abs(change_amount):.2f} ({abs(ch
```

```
Average spend (20192021): $3.31 Average spend (JanMay 2022): $3.38
Increase in average spend: $0.07 (2.09%)
```

In [65]:
```
import pandas as pd

sfm = pd.read_csv("dataset_2019_2022.csv", parse_dates=['transaction_date']) sfm['transaction_date'] =
pd.to_datetime(sfm['transaction_date'], errors='coerce') sfm['price'] = pd.to_numeric(sfm['price'], errors='coerce')
sfm['age_band'] = sfm['age_band'].str.strip()

sixth_age_band_data = sfm[sfm['age_band'].str.lower() == '65+']

historical = sixth_age_band_data[ (sixth_age_band_data['transaction_date'] >= '2019-01-01') &
    (sixth_age_band_data['transaction_date'] <= '2021-12-31')
]

recent = sixth_age_band_data[ (sixth_age_band_data['transaction_date'] >= '2022-01-01') &
    (sixth_age_band_data['transaction_date'] <= '2022-05-31')
]

historical_avg = historical['price'].mean() recent_avg = recent['price'].mean()
change_amount = recent_avg - historical_avg
change_percent = (change_amount / historical_avg) * 100 if historical_avg else None

if change_amount > 0: direction = "increase"
```

```
else:
    direction = "decrease"

print(f"Average spend (20192021): ${historical_avg:.2f}") print(f"Average spend (JanMay
2022): ${recent_avg:.2f}")
print(f"{direction.capitalize()} in average spend: ${abs(change_amount):.2f} ({abs(ch
```

```
Average spend (20192021): $2.96 Average spend (JanMay 2022): $2.64
Decrease in average spend: $0.32 (10.84%)
```

In [ ]:

In [14]:
```
import pandas as pd

sfm = pd.read_csv("dataset_2019_2022.csv", parse_dates=['transaction_date']) sfm['transaction_date'] =
pd.to_datetime(sfm['transaction_date'], errors='coerce') sfm['loyalty'] = sfm['loyalty'].str.strip().str.lower()

promiscuous_data = sfm[sfm['loyalty'] == 'promiscuous']

historical = promiscuous_data[ (promiscuous_data['transaction_date'] >= '2019-01-01') &
    (promiscuous_data['transaction_date'] <= '2021-12-31')
]

recent = promiscuous_data[ (promiscuous_data['transaction_date'] >= '2022-01-01') &
    (promiscuous_data['transaction_date'] <= '2022-05-31')
]

historical_avg = historical.groupby('basket_id').size().mean() recent_avg =
recent.groupby('basket_id').size().mean()

change_amount = recent_avg - historical_avg
change_percent = (change_amount / historical_avg) * 100 if historical_avg else None
direction = "increase" if change_amount > 0 else "decrease"

print(f"Average products per basket (20192021): {historical_avg:.2f}") print(f"Average products per basket (JanMay
2022): {recent_avg:.2f}") print(f"{direction.capitalize()} in average products per basket: {abs(change_amount):
```

```
Average products per basket (20192021): 22.17 Average products per basket (JanMay
2022): 16.76
Decrease in average products per basket: 5.41 (24.41%)
```

In [15]: **import pandas as pd**

```
sfm = pd.read_csv("dataset_2019_2022.csv", parse_dates=['transaction_date']) sfm['transaction_date'] =
pd.to_datetime(sfm['transaction_date'], errors='coerce') sfm['loyalty']  =  sfm['loyalty'].str.strip().str.lower()

loyalist_data = sfm[sfm['loyalty'] == 'loyalist']

historical = loyalist_data[ (loyalist_data['transaction_date'] >= '2019-01-01') &
    (loyalist_data['transaction_date'] <= '2021-12-31')
]

recent = loyalist_data[  (loyalist_data['transaction_date'] >= '2022-01-01') &
    (loyalist_data['transaction_date'] <= '2022-05-31')
]

historical_avg = historical.groupby('basket_id').size().mean() recent_avg =
recent.groupby('basket_id').size().mean()

change_amount = recent_avg - historical_avg
change_percent = (change_amount / historical_avg) * 100 if historical_avg else None
direction = "increase" if change_amount > 0 else "decrease"

print(f"Average products per basket (20192021): {historical_avg:.2f}") print(f"Average products per basket (JanMay
2022): {recent_avg:.2f}") print(f"{direction.capitalize()} in average products per basket: {abs(change_amount):
```

```
Average products per basket (20192021): 22.54 Average products per basket (JanMay
2022): 16.67
Decrease in average products per basket: 5.86 (26.02%)
```

In [16]:
```
import pandas as pd

sfm = pd.read_csv("dataset_2019_2022.csv", parse_dates=['transaction_date']) sfm['transaction_date'] =
pd.to_datetime(sfm['transaction_date'], errors='coerce') sfm['loyalty']  =  sfm['loyalty'].str.strip().str.lower()

first_time_buyer_data = sfm[sfm['loyalty'] == 'first time buyer']

historical = first_time_buyer_data[ (first_time_buyer_data['transaction_date'] >= '2019-01-01') &
    (first_time_buyer_data['transaction_date'] <= '2021-12-31')
]

recent = first_time_buyer_data[ (first_time_buyer_data['transaction_date'] >= '2022-01-01') &
    (first_time_buyer_data['transaction_date'] <= '2022-05-31')
```

```
]

historical_avg = historical.groupby('basket_id').size().mean() recent_avg =
recent.groupby('basket_id').size().mean()

change_amount = recent_avg - historical_avg
change_percent = (change_amount / historical_avg) * 100 if historical_avg else None
direction = "increase" if change_amount > 0 else "decrease"

print(f"Average products per basket (20192021): {historical_avg:.2f}") print(f"Average products per basket (JanMay
2022): {recent_avg:.2f}") print(f"{direction.capitalize()} in average products per basket: {abs(change_amount):
```

```
Average products per basket (20192021): 22.00 Average products per basket (JanMay
2022): 13.00
Decrease in average products per basket: 9.00 (40.91%)
```

In [19]:
```
import pandas as pd

sfm = pd.read_csv("dataset_2019_2022.csv", parse_dates=['transaction_date']) sfm['transaction_date'] =
pd.to_datetime(sfm['transaction_date'], errors='coerce') sfm['price'] = pd.to_numeric(sfm['price'], errors='coerce')

historical = sfm[
    (sfm['transaction_date'] >= '2019-01-01') & (sfm['transaction_date'] <= '2021-12-31')
]

recent = sfm[
    (sfm['transaction_date'] >= '2022-01-01') & (sfm['transaction_date'] <= '2022-05-31')
]

historical_avg = historical.groupby('basket_id')['price'].sum().mean() recent_avg =
recent.groupby('basket_id')['price'].sum().mean()

change_amount = recent_avg - historical_avg
change_percent = (change_amount / historical_avg) * 100 if historical_avg else None
direction = "increase" if change_amount > 0 else "decrease"

print(f"Average sales per basket (20192021): ${historical_avg:.2f}") print(f"Average sales per basket (JanMay
2022): ${recent_avg:.2f}")
print(f"{direction.capitalize()} in average basket sales: ${abs(change_amount):.2f} (
```

Average sales per basket (20192021): $71.73 Average sales per basket (JanMay 2022): $56.41 Decrease in average basket sales: $15.32 (21.35%)

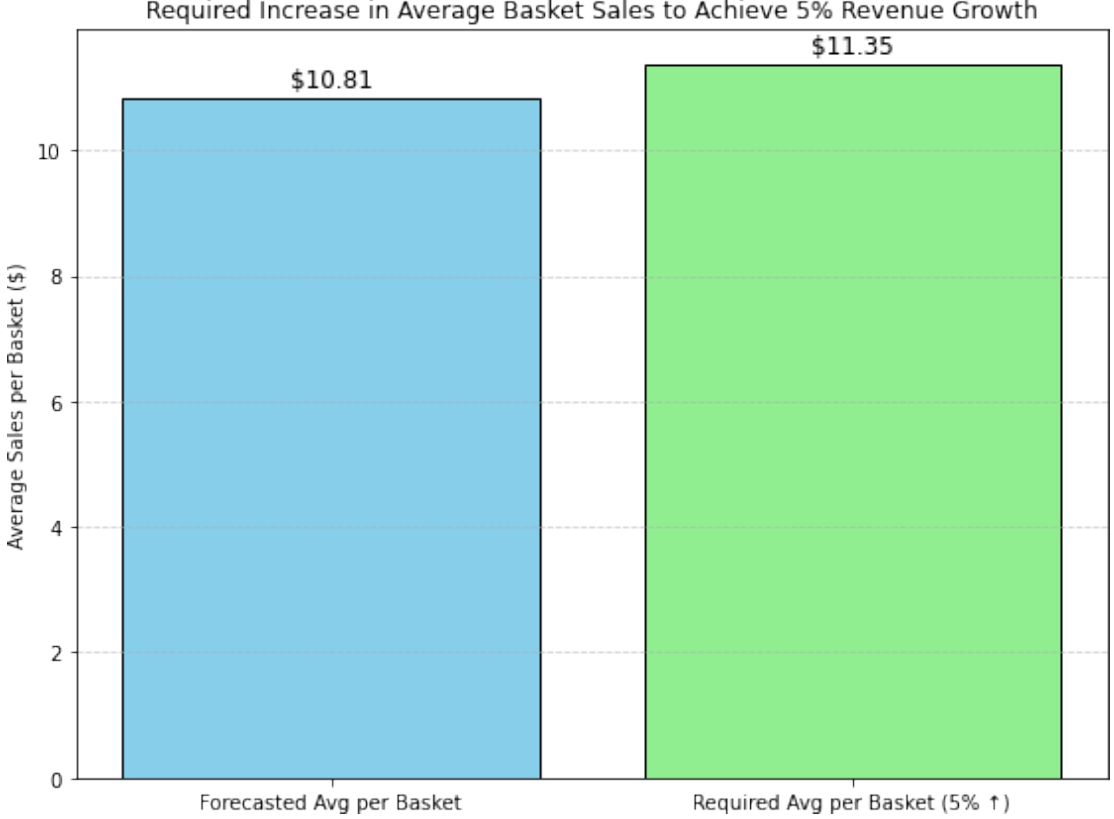
In [24]:
```
import pandas as pd
import numpy as np
from statsmodels.tsa.seasonal import STL
from statsmodels.tsa.holtwinters import ExponentialSmoothing

sfm = pd.read_csv("dataset_2019_2022.csv", parse_dates=['transaction_date'])

sfm = sfm[sfm['transaction_date'] < '2021-12-01']
sfm['month_start'] = sfm['transaction_date'].dt.to_period('M').dt.to_timestamp()

total_sales_by_month = sfm.groupby('month_start')['price'].sum().reset_index()
total_sales_by_month.set_index('month_start', inplace=True) total_sales_by_month.index =
pd.DatetimeIndex(total_sales_by_month.index, freq='MS')

seasonal_model = ExponentialSmoothing( total_sales_by_month['price'],
    trend='add',
    seasonal='add', seasonal_periods=12,
    initialization_method='estimated'
).fit()

forecast_periods = 24
forecast = seasonal_model.forecast(forecast_periods)
forecast_index = pd.date_range(start='2022-06-01', periods=forecast_periods, freq='MS forecast_series =
pd.Series(forecast.values, index=forecast_index)

basket_counts = sfm.groupby('month_start')['basket_id'].nunique() avg_baskets_per_month =

basket_counts.mean() total_forecasted_sales = forecast_series.sum()

target_sales = total_forecasted_sales * 1.05 expected_total_baskets = avg_baskets_per_month *

forecast_periods required_avg_basket_sales = target_sales / expected_total_baskets

forecasted_avg_basket_sales = total_forecasted_sales / expected_total_baskets

increase_needed = required_avg_basket_sales - forecasted_avg_basket_sales percent_increase = (increase_needed /
forecasted_avg_basket_sales) * 100

print(f"Forecasted total sales (24 months): ${total_forecasted_sales:,.2f}") print(f"Target sales for 5% increase:
${target_sales:,.2f}")
print(f"Expected total baskets (based on historical monthly avg): {expected_total_bas
```

```
print(f"Forecasted avg sales per basket: ${forecasted_avg_basket_sales:.2f}")
```

```
print(f"Required avg sales per basket: ${required_avg_basket_sales:.2f}") print(f"Increase needed per basket: ${increase_needed:.2f} ({percent_increase:.2f}%)"
```

Forecasted total sales (24 months): $149,230.17 Target sales for 5% increase: $156,691.68
Expected total baskets (based on historical monthly avg): 13,803 Forecasted avg sales per basket: $10.81
Required avg sales per basket: $11.35 Increase needed per basket: $0.54 (5.00%)

In [32]:
```
import matplotlib.pyplot as plt

labels = ['Forecasted Avg per Basket', 'Required Avg per Basket (5% )'] values = [forecasted_avg_basket_sales, required_avg_basket_sales] colors = ['skyblue', 'lightgreen']

plt.figure(figsize=(8, 6))
bars = plt.bar(labels, values, color=colors, edgecolor='black')

for bar in bars:
    height = bar.get_height()
    plt.text(bar.get_x() + bar.get_width() / 2, height + 0.1, f"${height:.2f}", ha='center', va='bottom',
             fontsize=12)

plt.title("Required Increase in Average Basket Sales to Achieve 5% Revenue Growth") plt.ylabel("Average Sales per Basket ($)")
plt.grid(axis='y', linestyle='--', alpha=0.6) plt.tight_layout()

plt.show()
```



In [33]:
```
import pandas as pd

sfm = pd.read_csv("dataset_2019_2022.csv", parse_dates=['transaction_date']) sfm['transaction_date'] =
pd.to_datetime(sfm['transaction_date'], errors='coerce') sfm['price'] = pd.to_numeric(sfm['price'], errors='coerce')
sfm['commodity'] = sfm['commodity'].str.strip()

top_products = ( sfm.groupby('commodity')['price']
    .sum()
    .sort_values(ascending=False)
    .head(10)
    .index
)

sfm_top = sfm[sfm['commodity'].isin(top_products)]
```

```
period1 = sfm_top[(sfm_top['transaction_date'] >= '2019-01-01') & (sfm_top['transaction_date'] <= '2021-
                  12-31')]
period2 = sfm_top[(sfm_top['transaction_date'] >= '2022-01-01') & (sfm_top['transaction_date'] <= '2022-
                  05-31')]
```

```
avg_sales_p1 = period1.groupby('commodity')['price'].mean().reset_index(name='avg_sal avg_sales_p2 =
period2.groupby('commodity')['price'].mean().reset_index(name='avg_sal

comparison_df = pd.merge(avg_sales_p1, avg_sales_p2, on='commodity') comparison_df['change_amount'] =
comparison_df['avg_sales_2022'] - comparison_df['avg comparison_df['change_percent'] =
(comparison_df['change_amount'] / comparison_df['av

pd.reset_option('display.float_format')

comparison_df['avg_sales_2019_2021'] = comparison_df['avg_sales_2019_2021'].apply(lam comparison_df['avg_sales_2022']
= comparison_df['avg_sales_2022'].apply(lambda x: f"$ comparison_df['change_amount'] =
comparison_df['change_amount'].apply(lambda x: f"${x comparison_df['change_percent'] =
comparison_df['change_percent'].apply(lambda x: f"{

comparison_df = comparison_df.sort_values(by='change_percent', ascending=False)

print(comparison_df[['commodity', 'avg_sales_2019_2021', 'avg_sales_2022', 'change_am
```

	commodity	avg_sales_2019_2021	avg_sales_2022	change_amount	\
2	Cheese	$3.07	$3.35	$0.28	
1	Candy	$1.61	$1.71	$0.11	
3	Cigarettes	$8.05	$11.49	$3.44	
4	Deli meats	$3.84	$4.00	$0.16	
6	Lunch meat	$2.67	$2.77	$0.10	
8	Salad	$2.94	$3.03	$0.10	
5	Frozen meat	$3.85	$4.59	$0.74	
7	Pork	$7.22	$8.56	$1.35	
9	Seafood - frozen	$6.49	$7.31	$0.82	
0	Beef	$5.86	$5.97	$0.11	

	change_percent
2	9.15%
1	6.57%
3	42.69%
4	4.27%
6	3.64%
8	3.32%
5	19.22%
7	18.66%
9	12.67%
0	1.94%

In [44]:
```
import pandas as pd

sfm = pd.read_csv("dataset_2019_2022.csv", parse_dates=['transaction_date']) sfm['transaction_date'] =
pd.to_datetime(sfm['transaction_date'], errors='coerce')
```

```
historical_sales = sfm[sfm['transaction_date'] < '2021-12-01']

sales_by_product = historical_sales.groupby('commodity')['price'].sum().reset_index() top_10_products =

sales_by_product.sort_values(by='price', ascending=False).head(10) total_sales = sales_by_product['price'].sum()

desired_sales_increase = total_sales * 0.05

top_10_products['required_increase'] = (top_10_products['price'] / total_sales) *

top_10_products['avg_required_increase'] = top_10_products['required_increase'] / len

print(top_10_products[['commodity', 'price', 'required_increase', 'avg_required_incre
```

	commodity	price	required_increase	avg_required_increase
20	Beef	14776.69	738.8345	73.88345
47	Cheese	5453.63	272.6815	27.26815
110	Frozen meat	5048.50	252.4250	25.24250
199	Salad	4976.49	248.8245	24.88245
149	Lunch meat	4904.47	245.2235	24.52235
66	Deli meats	4831.53	241.5765	24.15765
203	Seafood - frozen	4589.27	229.4635	22.94635
185	Pork	4475.81	223.7905	22.37905
37	Candy	3941.86	197.0930	19.70930
52	Cigarettes	3818.38	190.9190	19.09190

In [40]:
```
import pandas as pd

sfm = pd.read_csv("dataset_2019_2022.csv", parse_dates=['transaction_date']) sfm['transaction_date'] =
pd.to_datetime(sfm['transaction_date'], errors='coerce')

historical_sales = sfm[sfm['transaction_date'] < '2021-12-01']

sales_by_product = historical_sales.groupby('commodity').agg( total_sales=('price', 'sum'),
    sales_count=('price', 'size')
).reset_index()

top_10_products = sales_by_product.sort_values(by='total_sales', ascending=False).hea total_sales =

sales_by_product['total_sales'].sum()
```

```
desired_sales_increase = total_sales * 0.05
```

```
top_10_products['required_increase'] = (top_10_products['total_sales'] / total_sales)

top_10_products['required_price_increase'] = top_10_products['required_increase'] /

print(top_10_products[['commodity', 'total_sales', 'sales_count', 'required_increase'
```

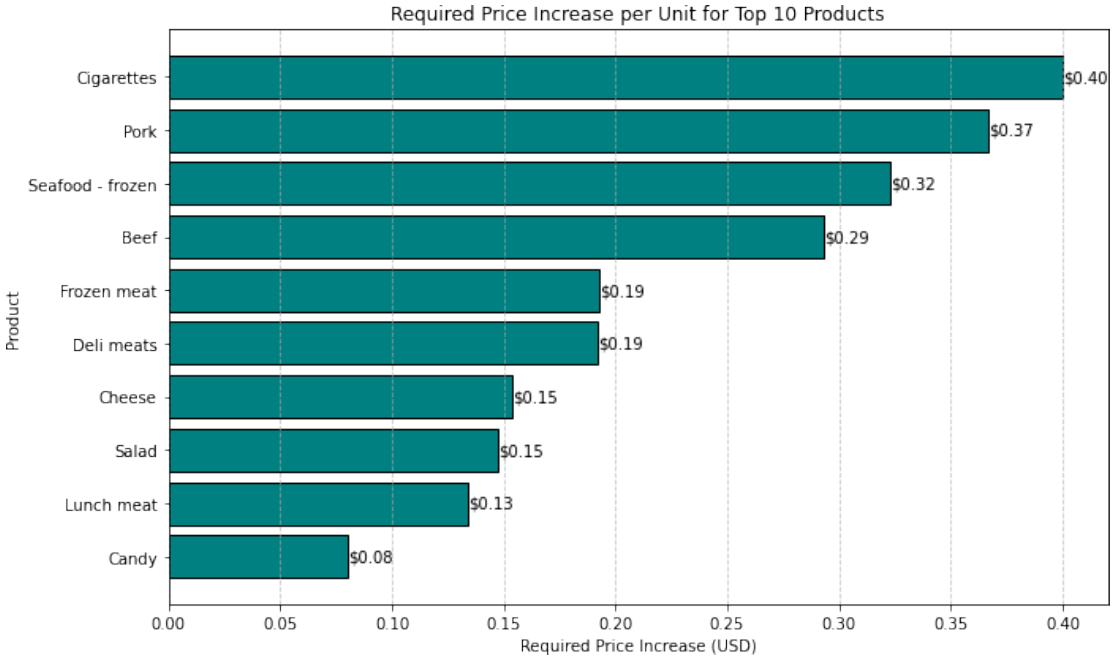
	commodity	total_sales	sales_count	required_increase	\
20	Beef	14776.69	2522	738.8345	
47	Cheese	5453.63	1772	272.6815	
110	Frozen meat	5048.50	1309	252.4250	
199	Salad	4976.49	1689	248.8245	
149	Lunch meat	4904.47	1831	245.2235	
66	Deli meats	4831.53	1258	241.5765	
203	Seafood - frozen	4589.27	711	229.4635	
185	Pork	4475.81	610	223.7905	
37	Candy	3941.86	2452	197.0930	
52	Cigarettes	3818.38	477	190.9190	

	required_price_increase
20	0.292956
47	0.153883
110	0.192838
199	0.147321
149	0.133929
66	0.192032
203	0.322733
185	0.366870
37	0.080381
52	0.400249

In [39]:
```
import matplotlib.pyplot as plt

top_10_products_sorted = top_10_products.sort_values(by='required_price_increase', as

plt.figure(figsize=(10, 6)) bars = plt.barh(
    top_10_products_sorted['commodity'], top_10_products_sorted['required_price_increase'],
    color='teal',
    edgecolor='black'
)
```

```
for bar in bars:
    width = bar.get_width()
    plt.text(width + 0.0001, bar.get_y() + bar.get_height()/2, f"${width:.2f}", va='center')
```

```
plt.title('Required Price Increase per Unit for Top 10 Products') plt.xlabel('Required Price Increase (USD)')
plt.ylabel('Product')
plt.grid(axis='x', linestyle='--', alpha=0.7) plt.tight_layout()

plt.show()
```



In [3]:
```
import pandas as pd

sfm = pd.read_csv("dataset_2019_2022.csv", parse_dates=['transaction_date'])

sfm['loyalty'] = sfm['loyalty'].str.strip().str.lower() sfm['customer_id'] =
sfm['customer_id'].astype(str) sfm['basket_id'] = sfm['basket_id'].astype(str)

first_time_buyers = sfm[sfm['loyalty'] == 'first time buyer'].copy()

unique_basket_counts = ( first_time_buyers
    .groupby('customer_id')['basket_id']
    .nunique()
    .reset_index(name='unique_basket_count')
```

```
)

print(unique_basket_counts.head())
```

	customer_id	unique_basket_count
0	15847	1
1	15909	1
2	16640	1
3	16752	2
4	17249	1

In [ ]: