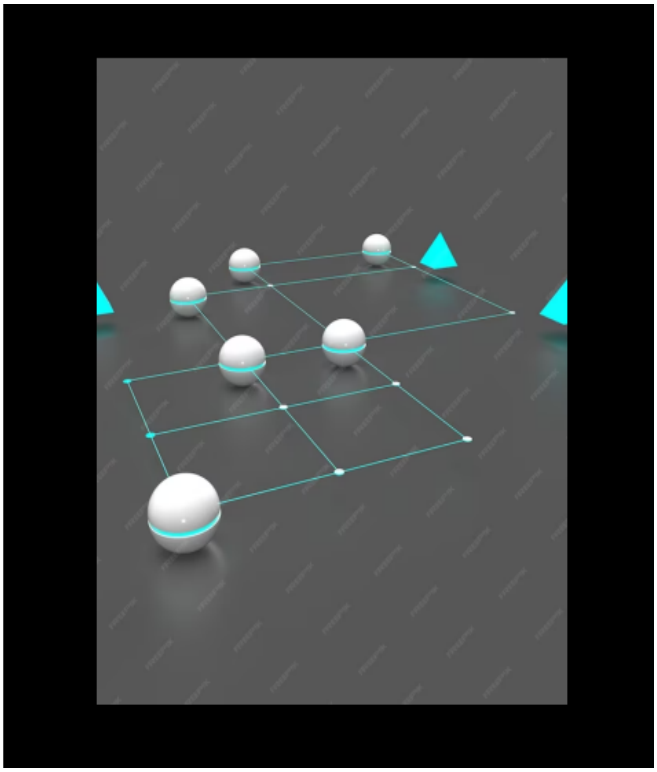
Unveiling Hidden Insights: Dataset and Preprocessing Techniques for Market Basket Analysis

Introduction

Welcome to the presentation on *Unveiling Hidden Insights: Dataset and Preprocessing Techniques for Market Basket Analysis*. This presentation will explore the importance of market basket analysis and how to effectively preprocess datasets for this analysis. Get ready to discover the secrets hidden in customer transactions!

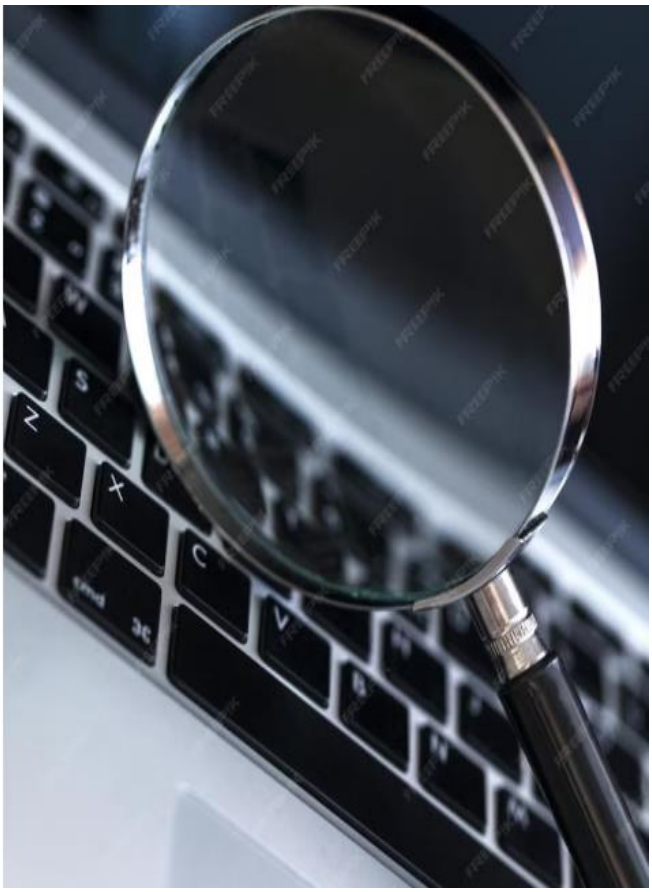
Market Basket Analysis

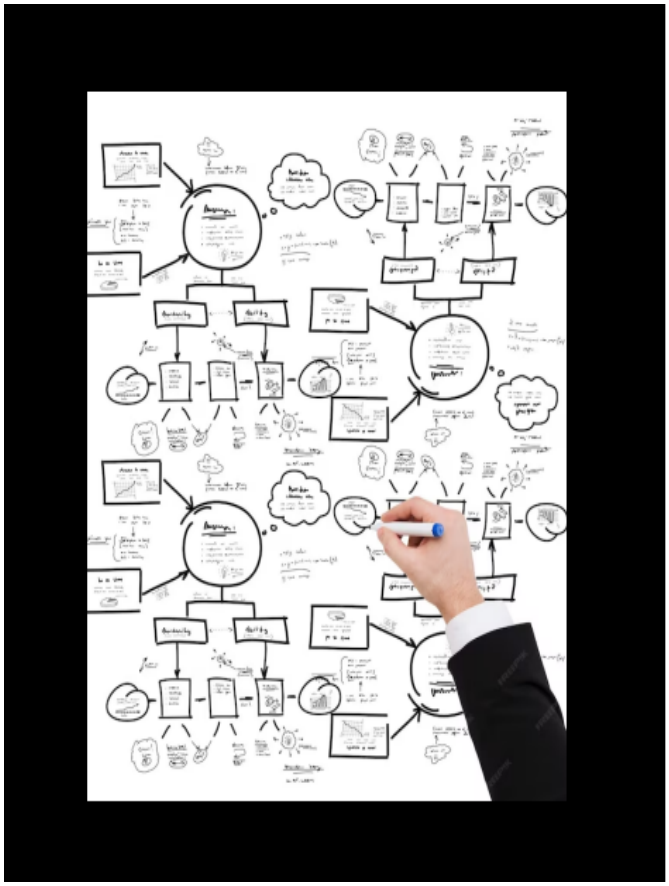
Market basket analysis is a powerful technique used to uncover *hidden patterns* and *associations* among items frequently purchased together by customers. By analyzing transactional data, we can gain valuable insights into customer behavior and make data-driven decisions to optimize business strategies.



Importance of Dataset

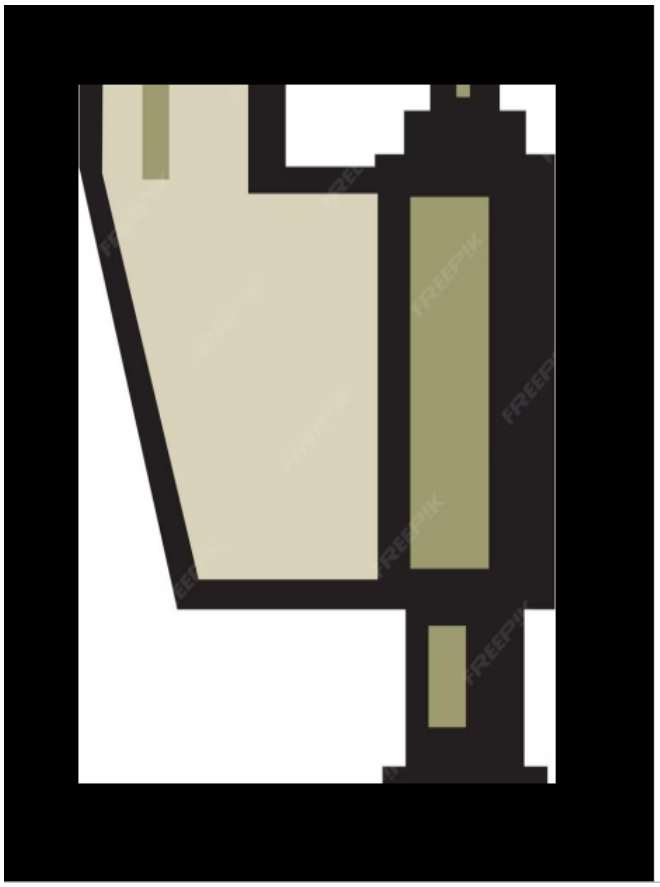
The quality and relevance of the dataset used for market basket analysis significantly impact the accuracy and usefulness of the results. It is crucial to ensure that the dataset is *comprehensive*, *clean*, and *representative* to obtain reliable insights for decision-making.





Preprocessing Techniques

Effective preprocessing techniques are essential to prepare the dataset for market basket analysis. Steps such as *data cleaning*, *data transformation*, and *data reduction* help remove noise, standardize formats, and reduce complexity, leading to improved analysis outcomes.



Popular Preprocessing Methods

Various preprocessing methods are commonly used in market basket analysis, including *one-hot encoding*, *transaction aggregation*, and *itemset filtering*. These techniques enable efficient handling of categorical data, consolidation of transactions, and reduction of itemsets for better analysis performance.

Objectives

- **Check data quality.**
- **Use exploratory data analysis to derive insights on product performance.**
- **Apply association-rule-mining to discover opportunities for cross-selling.**

```
import pandas as pd
import matplotlib as mpl
import seaborn as sns
from matplotlib.axes import Axes

sns.set_palette("autumn")

mpl.rc("axes", titlesize=18, titlepad=15, titleweight=500)
mpl.rc("axes.spines", right=False, top=False)
mpl.rc("figure", figsize=(10, 5.5))
mpl.rc("font", family="serif", size=10)


def annotate_column_chart(ax: Axes) -> Axes:
    """Add annotations to a column chart.
for p in ax.patches:
        p.set_width(0.7)
        ax.annotate(f"{p.get_height():,}", ha="center",
                xy=(p.get_x() + p.get_width() / 2, p.get_height() * 1.01)
)
    return ax


data = pd.read_csv(
header=None,
    names=[f"item_{idx}" for idx in range(1, 21)]
)

print(
)
data.head()
```

There were a total of 7,501 transactions, each containing between 1 and 20 items.

	item_1	item_2	item_3	item_4	item_5	item_6	item_7	item_8	item_9	item_10	item_11	item_12	item_13	item_14	item_15	item_16	item_17	item_18	item_19	item_20
0	shrimp	almonds	avocado	vegetables mix	green grapes	whole weat flour	yams	cottage cheese	energy drink	tomato juice	low fat yogurt	green tea	honey	salad	mineral water	salmon	antioxydant juice	frozen smoothie	spinach	olive oil
1	burgers	meatballs	eggs	NaN	NaN	NaN	NaN	NaN	NaN	NaN	NaN	NaN	NaN	NaN	NaN	NaN	NaN	NaN	NaN	NaN
2	chutney	NaN	NaN	NaN	NaN	NaN	NaN	NaN	NaN	NaN	NaN	NaN	NaN	NaN	NaN	NaN	NaN	NaN	NaN	NaN
3	turkey	avocado	NaN	NaN	NaN	NaN	NaN	NaN	NaN	NaN	NaN	NaN	NaN	NaN	NaN	NaN	NaN	NaN	NaN	NaN
4	mineral water	milk	energy bar	whole wheat rice	green tea	NaN	NaN	NaN	NaN	NaN	NaN	NaN	NaN	NaN	NaN	NaN	NaN	NaN	NaN	NaN

2. Data Cleaning

One instance of the item "asparagus" contains leading whitespace. Other than that, the data looks fine.

In [2]:
```
all_products = data.melt()["value"].dropna().sort_values()

# Find items that start or end with whitespace
all_products[all_products.str.contains("^\s|\s$")].to_list()
```

Out[2]:
```
[' asparagus']
```

3. Exploratory Data Analysis

3.1 Best-selling products

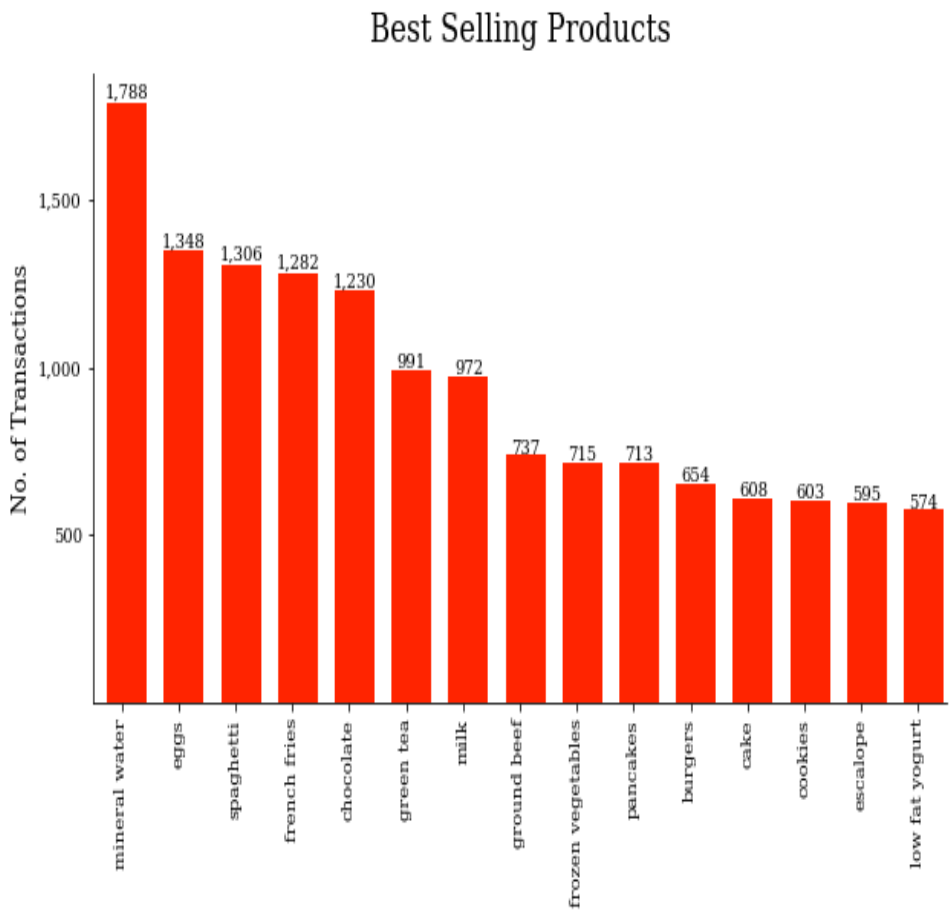
Assuming that only one unit of each item was bought in each transaction, *mineral water* is the most purchased product.

The top selling products are primarily food-stuff, but that's not at all surprising.

In [4]:
```
item_counts = all_products.value_counts()

ax = item_counts.nlargest(15).plot(kind="bar", title="Best Selling Products")
ax.set_ylabel("No. of Transactions", size=12)
ax.yaxis.set_major_formatter(mpl.ticker.StrMethodFormatter("{x:,.0f}"))
ax.yaxis.set_major_locator(mpl.ticker.FixedLocator([500, 1000, 1500]))

_ = annotate_column_chart(ax)
```



3.2 Worst performing products

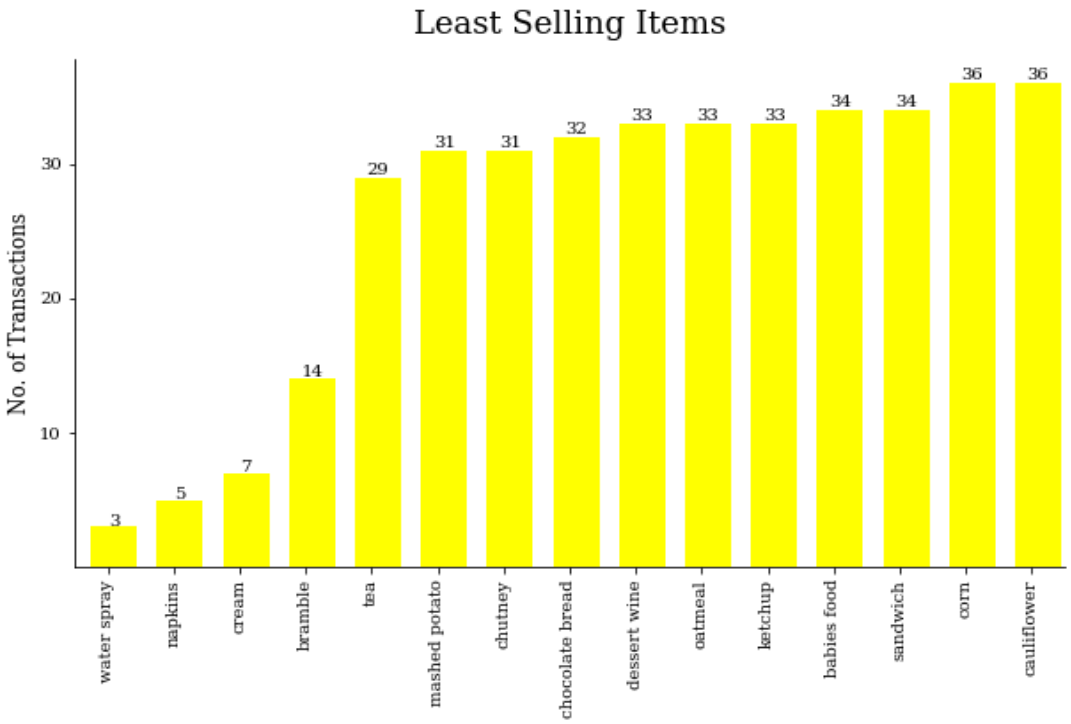
Assuming that only one unit of each item was bought in each transaction, *water spray* is sold the least.

It is quite unusual that the *tea*, *chocolate bread* and *sandwiches* are doing badly. This is worth investigating. Assuming this sample adequately captures the actual situation, then these products should probably be reviewed.

In [5]:

```
ax = item_counts.nsmallest(15).plot(kind="bar", color="yellow", title="Least Selling Items")
ax.set_ylabel("No. of Transactions", size=12)
ax.yaxis.set_major_locator(mpl.ticker.FixedLocator([10, 20, 30]))


_ = annotate_column_chart(ax)
```



3.3 Distribution of Basket sizes

The average basket-size was about 4 items. The largest transaction consisted of 20 items, and the smallest had just one.

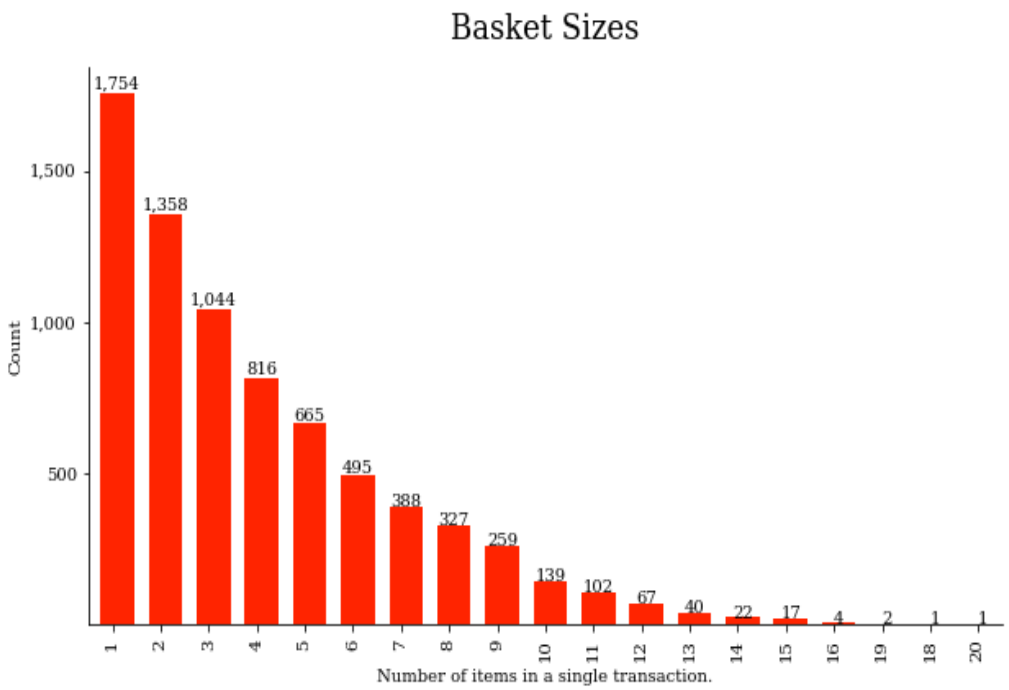
Majority of the transactions involved a single item.

In [6]:

```
basket_sizes = data.notna().apply(sum, axis=1)

ax = basket_sizes.value_counts().plot.bar(title="Basket Sizes")
ax.set_ylabel("Count")
ax.set_xlabel("Number of items in a single transaction.")
ax.yaxis.set_major_formatter(mpl.ticker.StrMethodFormatter("{x:,.0f}"))
ax.yaxis.set_major_locator(mpl.ticker.FixedLocator([500, 1000, 1500]))

_ = annotate_column_chart(ax)
```



3.4 What's in the largest transactions?

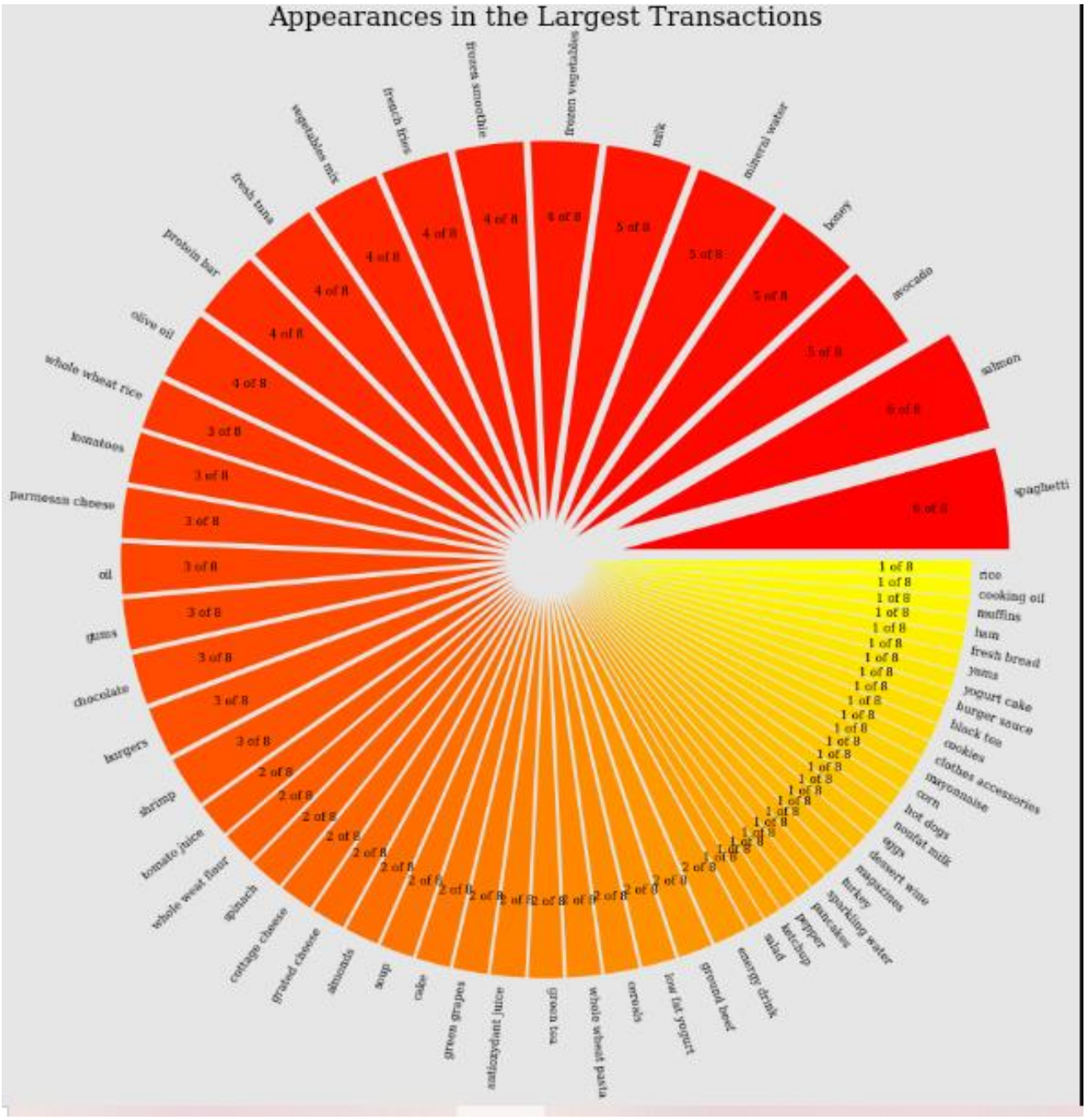
We'll consider transactions having more than 15 items (75% of maximum=20) as "large". There are 8 such transactions (16, 16, 16, 16, 18, 19, 19, 20).

***Spaghetti* and *salmon* are in 6 out of the eight largest transactions. Salmon's case is more striking, since we've already seen that spaghetti is the 3rd best seller. At face value, this might imply that customers who purchase a lot of items are more likely to buy salmon, so placing it next to the large trolleys/shopping-baskets might boost sales. But 8 out of 7501 cases doesn't inspire much confidence.**

In [9]:

```
items_in_largest_transactions = data[basket_sizes > 15].melt()['value'].dropna()

pie_data = items_in_largest_transactions.value_counts()
ax = pie_data.plot.pie(
    cmap="autumn",
    explode=[0.2] * 2 + [0.1] * 59,
    figsize=(12, 12),
    autopct=lambda pct: f" {pct * 0.01 * pie_data.sum():.0f} of 8",
    pctdistance=0.8,
    labeldistance=1.02,
    rotatelabels=True,
    textprops={"size": 9},
)
ax.set_title("Appearances in the Largest Transactions", size=20, pad=45)
ax.set_ylabel("")
ax.figure.tight_layout()
```



4.1 Preprocessing

Data input to the `efficient-apriori.apriori` function is required as a sequence of "baskets" e.g. a list of tuples containing items.

In order to find item relationships, the baskets must include more than 1 item. We'll need to discard singleton transactions.

In [12]:
```
baskets = [tuple(row.dropna()) for _, row in data[basket_sizes > 1].iterrows()]
baskets[-5:]
```

Out[12]:
```
[('pancakes', 'light mayo'),
 ('butter', 'light mayo', 'fresh bread'),
 ('burgers',
  'frozen vegetables',
  'eggs',
  'french fries',
  'magazines',
  'green tea'),
 ('escalope', 'green tea'),
 ('eggs', 'frozen smoothie', 'yogurt cake', 'low fat yogurt')]
```

4.2 Association rules

Potential opportunities for cross-selling are:

- ***frozen vegetables & spaghetti***
- ***burgers & eggs***
- ***ground beef & spaghetti***

In [13]:
```
from efficient_apriori import apriori

item_sets, association_rules = apriori(baskets, min_support=0.03, min_confidence=0.3)

# Get 1 to 1 rules e.g. {bread} -> {butter}
one_to_one_rules = filter(
    lambda rule: len(rule.lhs) == 1 and len(rule.rhs) == 1, association_rules
)
for rule in sorted(one_to_one_rules, key=lambda rule: rule.lift):
    print(rule)
```

```
{eggs} -> {mineral water} (conf: 0.304, supp: 0.066, lift: 1.030, conv: 1.013)
{shrimp} -> {mineral water} (conf: 0.339, supp: 0.031, lift: 1.150, conv: 1.067)
{low fat yogurt} -> {mineral water} (conf: 0.340, supp: 0.031, lift: 1.154, conv: 1.069)
{chocolate} -> {mineral water} (conf: 0.342, supp: 0.069, lift: 1.159, conv: 1.071)
{cake} -> {mineral water} (conf: 0.356, supp: 0.036, lift: 1.206, conv: 1.094)
{spaghetti} -> {mineral water} (conf: 0.357, supp: 0.078, lift: 1.211, conv: 1.097)
{tomatoes} -> {mineral water} (conf: 0.370, supp: 0.032, lift: 1.256, conv: 1.120)
{pancakes} -> {mineral water} (conf: 0.375, supp: 0.044, lift: 1.273, conv: 1.129)
{milk} -> {mineral water} (conf: 0.383, supp: 0.063, lift: 1.300, conv: 1.143)
{frozen vegetables} -> {mineral water} (conf: 0.385, supp: 0.047, lift: 1.306, conv: 1.147)
{frozen vegetables} -> {spaghetti} (conf: 0.300, supp: 0.036, lift: 1.376, conv: 1.117)
{ground beef} -> {mineral water} (conf: 0.429, supp: 0.053, lift: 1.454, conv: 1.234)
{olive oil} -> {mineral water} (conf: 0.439, supp: 0.036, lift: 1.490, conv: 1.258)
{burgers} -> {eggs} (conf: 0.341, supp: 0.038, lift: 1.556, conv: 1.185)
{soup} -> {mineral water} (conf: 0.466, supp: 0.030, lift: 1.581, conv: 1.321)
{ground beef} -> {spaghetti} (conf: 0.411, supp: 0.051, lift: 1.882, conv: 1.326)
```

Conclusion

In conclusion, market basket analysis is a valuable tool for understanding customer behavior and optimizing business strategies. By utilizing appropriate dataset and preprocessing techniques, businesses can uncover hidden insights and make informed decisions to enhance customer satisfaction and drive growth.

Presented by,

M. Vijayadharshini