Instacart Analysis: Python Code Snippets & Advanced Data Visualization

Executive Summary

Objective: Leverage Instacart's retail transaction dataset (3M+ rows) to extract customer behavior insights, optimize product segmentation strategies, and showcase advanced data engineering and statistical analysis capabilities.

Key Findings: Data profiling revealed major structural challenges—mixed data types, duplicate records, and missing values—systematically resolved through cleaning protocols and memory optimization. Multi-level groupings and cross-tabulations highlighted distinct segmentation patterns across departments and price ranges, while statistical visualizations exposed department-specific purchasing behaviors and spending trends. Strategic feature engineering introduced loyalty classifications, distinguishing new versus repeat customers to enable granular behavioral analysis.

Strategic Outcome: The project establishes a scalable framework for retail analytics and customer intelligence, demonstrating how complex transactional data can be transformed into strategic business insights. It validates technical expertise in Python (pandas, seaborn, matplotlib) while delivering business intelligence outcomes including customer segmentation models, dynamic pricing tier analysis, and loyalty-based behavioral metrics.

Advanced Data Manipulation & Processing

Complex Multi-Column Operations

```
# data transformation with multiple conditional logic
df_ords_prods_comb.loc[df_ords_prods_comb['prices'] > 15, 'price_range_loc'] = 'High-range product'
df_ords_prods_comb.loc[(df_ords_prods_comb['prices'] <= 15) & (df_ords_prods_comb['prices'] > 5), 'price_range_loc'] = 'Mid-range product'
df_ords_prods_comb.loc[df_ords_prods_comb['prices'] <= 5, 'price_range_loc'] = 'Low-range product'
```

Memory-Efficient Data Operations

```
# Strategic column selection for large datasets
df_ords_prods_comb = df_ords_prods_comb[['order_id', 'user_id', 'order_number', 'orders_day_of_week',
'order_hour_of_day', 'days_since_prior_order', 'product_id',
'add_to_cart_order', 'reordered', 'product_name', 'aisle_id',
'department_id', 'prices', 'price_range_loc']]
```

📊 Data Profiling & Data Validation

```
# Multi-dimensional data profiling
print('First 5 rows:')
print(df.head())
print('Data types:')
print(df.dtypes)
print('Dataset shape:')
print(df.shape)
print('Mixed-type columns info:')
print(df.info(memory_usage='deep'))
```

Advanced Categorical Analysis

```
# Frequency distribution with statistical insights
df['department'].value_counts(dropna=False)
```

Date Integrity Verification

```
# Duplicate detection and validation
df_dups = df[df.duplicated()]
print(df_dups.shape)
df_dups
```

🎯 Advanced Groupby Operations & Business Intelligence

Complex Multi-Level Aggregations

```
# Business metric calculation with grouped operations
df_ords_prods_comb.groupby(['department', 'price_range_loc']).agg({'order_id': ['mean', 'min', 'max']})
```

Customer Segmentation Analysis

```
Customer Segmentation Analysis
# Advanced customer behavior profiling
crosstab = pd.crosstab(df_ords_prods_comb['department'], df_ords_prods_comb['price_range_loc'], margins=True)
```

📈 Data Visualization & Reporting

Professional Statistical Visualizations

```
# Advanced plotting with seaborn for business insights
import seaborn as sns
import matplotlib.pyplot as plt

# Multi-dimensional analysis visualization
sns.barplot(data=df_ords_prods_comb, x='department', y='prices')
plt.xticks(rotation=45)
plt.tight_layout()
plt.show()
```

Interactive Data Exploration

```
# Dynamic filtering and analysis
loyalty_flag = df_ords_prods_comb.loc[df_ords_prods_comb['loyalty_flag'] == 'Loyal customer']
```

🔍 Data Engineering & Pipeline Operations

Efficient Data Merging Strategies

```
# Large-scale dataset joins with validation
df_merged_large = df_ords.merge(df_ords_prods_comb, on='order_id', indicator=True)
```

Column Engineering & Feature Creation

```
# Automated feature engineering
df_ords_prods_comb['loyalty_flag'] = 'New customer' # Default assignment
df_ords_prods_comb.loc[condition, 'loyalty_flag'] = 'Loyal customer' # Conditional update
```

💡 Code Quality Highlights

- Memory Management: Strategic column selection and efficient data operations
- Error Handling: Comprehensive data validation and duplicate detection
- Scalability: Operations designed for large retail datasets (3M+ rows)
- Business Logic: Complex conditional operations for customer segmentation
- Statistical Rigor: Multi-dimensional aggregations and cross-tabulations
- Visualization: Professional plotting with proper formatting and rotation
- Documentation: Clear variable naming and logical code structure

🎯 Technical Skills Demonstrated

- Data Manipulation: Advanced pandas operations, conditional logic, multi-level indexing
- Statistical Analysis: Descriptive statistics, frequency distributions, cross-tabulations
- Data Engineering: Large dataset merges, memory optimization, pipeline operations
- Visualization: Seaborn integration, professional chart formatting
- Business Intelligence: Customer segmentation, price range analysis, loyalty metrics

---

This case study demonstrates comprehensive data engineering capabilities, advanced statistical analysis, and scalable business intelligence extraction from large-scale retail datasets. Through systematic methods, the analysis transformed raw transactional data into actionable customer insights while establishing robust technical frameworks to support ongoing retail analytics and strategic decision-making.
The full project code and documentation are available on GitHub at: https://github.com/bea-bijanne