Context

``` fase_a.ipynb
{'cells': [{'cell_type': 'markdown', 'metadata': {}, 'source': ['# Fase A: Exploratory Data Analysis (EDA) - BigMart Sales Dataset\n', '\n', '## Objetivo\n', 'Realizar un análisis exploratorio completo del dataset BigMart Sales para entender la estructura, calidad y características de los datos antes de proceder con el clustering de productos.\n', '\n', '## Contexto de Negocio\n', 'El dataset contiene información de ventas de 1,559 productos distribuidos en 10 tiendas de diferentes ciudades. Incluye atributos del producto (peso, contenido graso, visibilidad, tipo, precio) y de la tienda (año de establecimiento, tamaño, ubicación, tipo). El objetivo final es segmentar productos según su comportamiento de ventas y características para analizar la mezcla de productos por tienda.']}, {'cell_type': 'code', 'execution_count': 1, 'metadata': {}, 'outputs': [], 'source': ['# Importar librerías necesarias\n', 'import pandas as pd\n', 'import numpy as np\n', 'import matplotlib.pyplot as plt\n', 'import seaborn as sns\n', 'import warnings\n', "warnings.filterwarnings('ignore')\n", '\n', '# Configuración de visualización\n', "plt.style.use('seaborn-v0_8')\n", 'sns.set_palette("husl")\n', '%matplotlib inline']}, {'cell_type': 'code', 'execution_count': 2, 'metadata': {}, 'outputs': [{'name': 'stdout', 'output_type': 'stream', 'text': ['✓ Dataset cargado exitosamente\n', 'Dimensiones del dataset: (8523, 12)\n', 'Número de registros: 8523\n', 'Número de variables: 12\n']}], 'source': ['# Cargar datos\n', "df = pd.read_csv('../Data/Raw/train.csv')\n", 'print("✓ Dataset cargado exitosamente")\n', 'print(f"Dimensiones del dataset: {df.shape}")\n', 'print(f"Número de registros: {df.shape[0]}")\n', 'print(f"Número de variables: {df.shape[1]}")']}, {'cell_type': 'markdown', 'metadata': {}, 'source': ['## 1. Información General del Dataset']}, {'cell_type': 'code', 'execution_count': 3, 'metadata': {}, 'outputs': [{'name': 'stdout', 'output_type': 'stream', 'text': ['INFORMACIÓN GENERAL DEL DATASET\n', '==================================================\n', "<class 'pandas.core.frame.DataFrame'>\n", 'RangeIndex: 8523 entries, 0 to 8522\n', 'Data columns (total 12 columns):\n', ' #   Column                     Non-Null Count  Dtype  \n', '---  ------                     --------------  -----  \n', ' 0   Item_Identifier            8523 non-null   object \n', ' 1   Item_Weight                7060 non-null   float64\n', ' 2   Item_Fat_Content           8523 non-null   object \n', ' 3   Item_Visibility            8523 non-null   float64\n', ' 4   Item_Type                  8523 non-null   object \n', ' 5   Item_MRP                   8523 non-null   float64\n', ' 6   Outlet_Identifier          8523 non-null   object \n', ' 7   Outlet_Establishment_Year  8523 non-null   int64  \n', ' 8   Outlet_Size                6113 non-null   object \n', ' 9   Outlet_Location_Type       8523 non-null   object \n', ' 10  Outlet_Type                8523 non-null   object \n', ' 11  Item_Outlet_Sales          8523 non-null   float64\n', 'dtypes: float64(4), int64(1), object(7)\n', 'memory usage: 799.2+ KB\n', '\n', '==================================================\n', 'PRIMERAS FILAS DEL DATASET\n', '==================================================\n']}, {'data': {'text/html': ['<div>\n', '<style scoped>\n', '    .dataframe tbody tr th:only-of-type {\n', '        vertical-align: middle;\n', '    }\n', '\n', '    .dataframe tbody tr th {\n', '        vertical-align: top;\n', '    }\n', '\n', '    .dataframe thead th {\n', '        text-align: right;\n', '    }\n', '</style>\n', '<table border="1" class="dataframe">\n', '  <thead>\n', '    <tr style="text-align: right;">\n', '      <th></th>\n', '      <th>Item_Identifier</th>\n', '      <th>Item_Weight</th>\n', '      <th>Item_Fat_Content</th>\n', '      <th>Item_Visibility</th>\n', '      <th>Item_Type</th>\n', '      <th>Item_MRP</th>\n', '      <th>Outlet_Identifier</th>\n', '      <th>Outlet_Establishment_Year</th>\n', '      <th>Outlet_Size</th>\n', '      <th>Outlet_Location_Type</th>\n', '
```

<th>Outlet_Type</th>\n', ' <th>Item_Outlet_Sales</th>\n', ' </tr>\n', ' </thead>\n', '
<tbody>\n', ' <tr>\n', ' <th>0</th>\n', ' <td>FDA15</td>\n', ' <td>9.30</td>\n', '
<td>Low Fat</td>\n', ' <td>0.016047</td>\n', ' <td>Dairy</td>\n', '
<td>249.8092</td>\n', ' <td>OUT049</td>\n', ' <td>1999</td>\n', '
<td>Medium</td>\n', ' <td>Tier 1</td>\n', ' <td>Supermarket Type1</td>\n', '
<td>3735.1380</td>\n', ' </tr>\n', ' <tr>\n', ' <th>1</th>\n', ' <td>DRC01</td>\n', '
<td>5.92</td>\n', ' <td>Regular</td>\n', ' <td>0.019278</td>\n', ' <td>Soft
Drinks</td>\n', ' <td>48.2692</td>\n', ' <td>OUT018</td>\n', ' <td>2009</td>\n', '
<td>Medium</td>\n', ' <td>Tier 3</td>\n', ' <td>Supermarket Type2</td>\n', '
<td>443.4228</td>\n', ' </tr>\n', ' <tr>\n', ' <th>2</th>\n', ' <td>FDN15</td>\n', '
<td>17.50</td>\n', ' <td>Low Fat</td>\n', ' <td>0.016760</td>\n', '
<td>Meat</td>\n', ' <td>141.6180</td>\n', ' <td>OUT049</td>\n', '
<td>1999</td>\n', ' <td>Medium</td>\n', ' <td>Tier 1</td>\n', ' <td>Supermarket
Type1</td>\n', ' <td>2097.2700</td>\n', ' </tr>\n', ' <tr>\n', ' <th>3</th>\n', '
<td>FDX07</td>\n', ' <td>19.20</td>\n', ' <td>Regular</td>\n', '
<td>0.000000</td>\n', ' <td>Fruits and Vegetables</td>\n', ' <td>182.0950</td>\n', '
<td>OUT010</td>\n', ' <td>1998</td>\n', ' <td>NaN</td>\n', ' <td>Tier 3</td>\n', '
<td>Grocery Store</td>\n', ' <td>732.3800</td>\n', ' </tr>\n', ' <tr>\n', '
<th>4</th>\n', ' <td>NCD19</td>\n', ' <td>8.93</td>\n', ' <td>Low Fat</td>\n', '
<td>0.000000</td>\n', ' <td>Household</td>\n', ' <td>53.8614</td>\n', '
<td>OUT013</td>\n', ' <td>1987</td>\n', ' <td>High</td>\n', ' <td>Tier 3</td>\n', '
<td>Supermarket Type1</td>\n', ' <td>994.7052</td>\n', ' </tr>\n', ' </tbody>\n',
'</table>\n', '</div>'], 'text/plain': [' Item_Identifier Item_Weight Item_Fat_Content
Item_Visibility \\\n', '0 FDA15 9.30 Low Fat 0.016047 \n', '1
DRC01 5.92 Regular 0.019278 \n', '2 FDN15 17.50 Low
Fat 0.016760 \n', '3 FDX07 19.20 Regular 0.000000 \n', '4
NCD19 8.93 Low Fat 0.000000 \n', '\n', ' Item_Type Item_MRP
Outlet_Identifier \\\n', '0 Dairy 249.8092 OUT049 \n', '1 Soft
Drinks 48.2692 OUT018 \n', '2 Meat 141.6180 OUT049 \n', '3
Fruits and Vegetables 182.0950 OUT010 \n', '4 Household 53.8614
OUT013 \n', '\n', ' Outlet_Establishment_Year Outlet_Size Outlet_Location_Type \\\n', '0
1999 Medium Tier 1 \n', '1 2009 Medium Tier 3 \n', '2
1999 Medium Tier 1 \n', '3 1998 NaN Tier 3 \n', '4
1987 High Tier 3 \n', '\n', ' Outlet_Type Item_Outlet_Sales \n', '0
Supermarket Type1 3735.1380 \n', '1 Supermarket Type2 443.4228 \n', '2
Supermarket Type1 2097.2700 \n', '3 Grocery Store 732.3800 \n', '4
Supermarket Type1 994.7052 ']}, 'metadata': {}, 'output_type': 'display_data'}],
'source': ['# Información general\n', 'print("INFORMACIÓN GENERAL DEL DATASET")\n',
'print("="*50)\n', 'df.info()\n', 'print("\\n" + "="*50)\n', 'print("PRIMERAS FILAS DEL
DATASET")\n', 'print("="*50)\n', 'display(df.head())']}, {'cell_type': 'code', 'execution_count': 4,
'metadata': {}, 'outputs': [{'name': 'stdout', 'output_type': 'stream', 'text': ['DISTRIBUCIÓN DE
TIPOS DE DATOS\n', '==\n',
'object 7\n', 'float64 4\n', 'int64 1\n', 'Name: count, dtype: int64\n', '\n', '\n', 'NÚMERO
DE VALORES ÚNICOS POR COLUMNA\n',
'==\n', 'Item_Identifier: 1559
valores únicos\n', 'Item_Weight: 415 valores únicos\n', 'Item_Fat_Content: 5 valores
únicos\n', " Valores: ['Low Fat' 'Regular' 'low fat' 'LF' 'reg']\n", 'Item_Visibility: 7880 valores
únicos\n', 'Item_Type: 16 valores únicos\n', 'Item_MRP: 5938 valores únicos\n',

'Outlet_Identifier: 10 valores únicos\n', "   Valores: ['OUT049' 'OUT018' 'OUT010' 'OUT013' 'OUT027' 'OUT045' 'OUT017' 'OUT046'\n", " 'OUT035' 'OUT019']\n", 'Outlet_Establishment_Year: 9 valores únicos\n', '   Valores: [1999 2009 1998 1987 1985 2002 2007 1997 2004]\n', 'Outlet_Size: 3 valores únicos\n', "   Valores: ['Medium' nan 'High' 'Small']\n", 'Outlet_Location_Type: 3 valores únicos\n', "   Valores: ['Tier 1' 'Tier 3' 'Tier 2']\n", 'Outlet_Type: 4 valores únicos\n', "   Valores: ['Supermarket Type1' 'Supermarket Type2' 'Grocery Store'\n", " 'Supermarket Type3']\n", 'Item_Outlet_Sales: 3493 valores únicos\n']}], 'source': ['# Tipos de datos\n', 'print("DISTRIBUCIÓN DE TIPOS DE DATOS")\n', 'print("="*50)\n', 'print(df.dtypes.value_counts())\n', 'print("\\n")\n', '\n', '# Valores únicos por columna\n', 'print("NÚMERO DE VALORES ÚNICOS POR COLUMNA")\n', 'print("="*50)\n', 'for col in df.columns:\n', '   unique_count = df[col].nunique()\n', '   print(f"{col}: {unique_count} valores únicos")\n', '   if unique_count <= 15:\n', '   print(f"   Valores: {df[col].unique()}")']}, {'cell_type': 'markdown', 'metadata': {}, 'source': ['## 2. Estadísticas Descriptivas - Variables Numéricas']}, {'cell_type': 'code', 'execution_count': 5, 'metadata': {}, 'outputs': [{'name': 'stdout', 'output_type': 'stream', 'text': ["Variables numéricas: ['Item_Weight', 'Item_Visibility', 'Item_MRP', 'Outlet_Establishment_Year', 'Item_Outlet_Sales']\n", '\n', 'ESTADÍSTICAS DESCRIPTIVAS - VARIABLES NUMÉRICAS\n', '==\n']}, {'data': {'text/html': ['<div>\n', '<style scoped>\n', '   .dataframe tbody tr th:only-of-type {\n', '   vertical-align: middle;\n', '   }\n', '\n', '   .dataframe tbody tr th {\n', '   vertical-align: top;\n', '   }\n', '\n', '   .dataframe thead th {\n', '   text-align: right;\n', '   }\n', '</style>\n', '<table border="1" class="dataframe">\n', ' <thead>\n', '   <tr style="text-align: right;">\n', '   <th></th>\n', '   <th>Item_Weight</th>\n', '   <th>Item_Visibility</th>\n', '   <th>Item_MRP</th>\n', '   <th>Outlet_Establishment_Year</th>\n', '   <th>Item_Outlet_Sales</th>\n', '   </tr>\n', ' </thead>\n', ' <tbody>\n', '   <tr>\n', '   <th>count</th>\n', '   <td>7060.000000</td>\n', '   <td>8523.000000</td>\n', '   <td>8523.000000</td>\n', '   <td>8523.000000</td>\n', '   <td>8523.000000</td>\n', '   </tr>\n', '   <tr>\n', '   <th>mean</th>\n', '   <td>12.857645</td>\n', '   <td>0.066132</td>\n', '   <td>140.992782</td>\n', '   <td>1997.831867</td>\n', '   <td>2181.288914</td>\n', '   </tr>\n', '   <tr>\n', '   <th>std</th>\n', '   <td>4.643456</td>\n', '   <td>0.051598</td>\n', '   <td>62.275067</td>\n', '   <td>8.371760</td>\n', '   <td>1706.499616</td>\n', '   </tr>\n', '   <tr>\n', '   <th>min</th>\n', '   <td>4.555000</td>\n', '   <td>0.000000</td>\n', '   <td>31.290000</td>\n', '   <td>1985.000000</td>\n', '   <td>33.290000</td>\n', '   </tr>\n', '   <tr>\n', '   <th>25%</th>\n', '   <td>8.773750</td>\n', '   <td>0.026989</td>\n', '   <td>93.826500</td>\n', '   <td>1987.000000</td>\n', '   <td>834.247400</td>\n', '   </tr>\n', '   <tr>\n', '   <th>50%</th>\n', '   <td>12.600000</td>\n', '   <td>0.053931</td>\n', '   <td>143.012800</td>\n', '   <td>1999.000000</td>\n', '   <td>1794.331000</td>\n', '   </tr>\n', '   <tr>\n', '   <th>75%</th>\n', '   <td>16.850000</td>\n', '   <td>0.094585</td>\n', '   <td>185.643700</td>\n', '   <td>2004.000000</td>\n', '   <td>3101.296400</td>\n', '   </tr>\n', '   <tr>\n', '   <th>max</th>\n', '   <td>21.350000</td>\n', '   <td>0.328391</td>\n', '   <td>266.888400</td>\n', '   <td>2009.000000</td>\n', '   <td>13086.964800</td>\n', '   </tr>\n', ' </tbody>\n', '</table>\n', '</div>'], 'text/plain': ['   Item_Weight Item_Visibility   Item_MRP Outlet_Establishment_Year \\\n', 'count 7060.000000   8523.000000 8523.000000   8523.000000   \n', 'mean   12.857645   0.066132   140.992782   1997.831867   \n', 'std   4.643456

0.051598 62.275067 8.371760 \n', 'min 4.555000 0.000000 31.290000 1985.000000 \n', '25% 8.773750 0.026989 93.826500 1987.000000 \n', '50% 12.600000 0.053931 143.012800 1999.000000 \n', '75% 16.850000 0.094585 185.643700 2004.000000 \n', 'max 21.350000 0.328391 266.888400 2009.000000 \n', '\n', ' Item_Outlet_Sales \n', 'count 8523.000000 \n', 'mean 2181.288914 \n', 'std 1706.499616 \n', 'min 33.290000 \n', '25% 834.247400 \n', '50% 1794.331000 \n', '75% 3101.296400 \n', 'max 13086.964800 ']}, 'metadata': {}, 'output_type': 'display_data'}], 'source': ['# Seleccionar variables numéricas\n', 'numeric_cols = df.select_dtypes(include=[np.number]).columns.tolist()\n', 'print(f"Variables numéricas: {numeric_cols}")\n', '\n', '# Estadísticas descriptivas\n', 'print("\\nESTADÍSTICAS DESCRIPTIVAS - VARIABLES NUMÉRICAS")\n', 'print("="*70)\n', 'display(df[numeric_cols].describe())']}, {'cell_type': 'code', 'execution_count': 6, 'metadata': {}, 'outputs': [{'data': {'text/plain': ['<Figure size 1500x1000 with 5 Axes>']}, 'metadata': {}, 'output_type': 'display_data'}, {'name': 'stdout', 'output_type': 'stream', 'text': ['\n', 'ANÁLISIS DE DISTRIBUCIONES:\n', '------------------------------\n', 'Item_Weight: Asimetría = 0.08\n', ' → Distribución aproximadamente simétrica\n', 'Item_Visibility: Asimetría = 1.17\n', ' → Distribución altamente sesgada\n', 'Item_MRP: Asimetría = 0.13\n', ' → Distribución aproximadamente simétrica\n', 'Outlet_Establishment_Year: Asimetría = -0.40\n', ' → Distribución aproximadamente simétrica\n', 'Item_Outlet_Sales: Asimetría = 1.18\n', ' → Distribución altamente sesgada\n']}], 'source': ['# Distribución de variables numéricas\n', 'fig, axes = plt.subplots(2, 3, figsize=(15, 10))\n', 'axes = axes.ravel()\n', '\n', 'for i, col in enumerate(numeric_cols):\n', ' if i < len(axes):\n', ' df[col].hist(bins=30, ax=axes[i], alpha=0.7)\n', " axes[i].set_title(f'Distribución de {col}')\n", ' axes[i].set_xlabel(col)\n', " axes[i].set_ylabel('Frecuencia')\n", '\n', '# Eliminar ejes vacíos\n', 'for i in range(len(numeric_cols), len(axes)):\n', ' fig.delaxes(axes[i])\n', '\n', 'plt.tight_layout()\n', 'plt.show()\n', '\n', 'print("\\nANÁLISIS DE DISTRIBUCIONES:")\n', 'print("-" * 30)\n', 'for col in numeric_cols:\n', ' skewness = df[col].skew()\n', ' print(f"{col}: Asimetría = {skewness:.2f}")\n', ' if abs(skewness) > 1:\n', ' print(f" → Distribución altamente sesgada")\n', ' elif abs(skewness) > 0.5:\n', ' print(f" → Distribución moderadamente sesgada")\n', ' else:\n', ' print(f" → Distribución aproximadamente simétrica")']}, {'cell_type': 'markdown', 'metadata': {}, 'source': ['## 3. Estadísticas Descriptivas - Variables Categóricas']}, {'cell_type': 'code', 'execution_count': 7, 'metadata': {}, 'outputs': [{'name': 'stdout', 'output_type': 'stream', 'text': ["Variables categóricas: ['Item_Identifier', 'Item_Fat_Content', 'Item_Type', 'Outlet_Identifier', 'Outlet_Size', 'Outlet_Location_Type', 'Outlet_Type']\n", '\n', 'ESTADÍSTICAS DESCRIPTIVAS - VARIABLES CATEGÓRICAS\n', '==\n', '\n', 'Item_Identifier:\n', ' Número de categorías: 1559\n', " Categorías: ['FDA15' 'DRC01' 'FDN15' ... 'NCF55' 'NCW30' 'NCW05']\n", ' Frecuencias:\n']}, {'data': {'text/html': ['<div>\n', '<style scoped>\n', ' .dataframe tbody tr th:only-of-type {\n', ' vertical-align: middle;\n', ' }\n', '\n', ' .dataframe tbody tr th {\n', ' vertical-align: top;\n', ' }\n', '\n', ' .dataframe thead th {\n', ' text-align: right;\n', ' }\n', '</style>\n', '<table border="1" class="dataframe">\n', ' <thead>\n', ' <tr style="text-align: right;">\n', ' <th>Item_Identifier</th>\n', ' <th>FDW13</th>\n', ' <th>FDG33</th>\n', ' <th>FDX31</th>\n', ' <th>FDT07</th>\n', ' <th>NCY18</th>\n', ' <th>FDW26</th>\n', ' <th>NCQ06</th>\n', ' <th>DRN47</th>\n', ' <th>FDV38</th>\n', ' <th>FDX20</th>\n', ' <th>...</th>\n', ' <th>NCM42</th>\n', ' <th>FDQ60</th>\n', ' <th>FDY43</th>\n', ' <th>DRF48</th>\n', '

<th>FDC23</th>\n', '   <th>FDO33</th>\n', '   <th>FDK57</th>\n', '   <th>FDT35</th>\n', '   <th>FDN52</th>\n', '   <th>FDE52</th>\n', '   </tr>\n', ' </thead>\n', ' <tbody>\n', '   <tr>\n', '   <th>count</th>\n', '   <td>10</td>\n', '   <td>10</td>\n', '   <td>9</td>\n', '   <td>9</td>\n', '   <td>9</td>\n', '   <td>9</td>\n', '   <td>9</td>\n', '   <td>9</td>\n', '   <td>9</td>\n', '   <td>9</td>\n', '   <td>...</td>\n', '   <td>2</td>\n', '   <td>1</td>\n', '   <td>1</td>\n', '   <td>1</td>\n', '   <td>1</td>\n', '   <td>1</td>\n', '   <td>1</td>\n', '   <td>1</td>\n', '   <td>1</td>\n', '   <td>1</td>\n', '   </tr>\n', ' </tbody>\n', '</table>\n', '<p>1 rows × 1559 columns</p>\n', '</div>'], 'text/plain': ['Item_Identifier FDW13 FDG33 FDX31 FDT07 NCY18 FDW26 NCQ06 DRN47 \\\n', 'count   10   10   9   9   9   9   9   9   \n', '\n', 'Item_Identifier FDV38 FDX20 ... NCM42 FDQ60 FDY43 DRF48 FDC23 FDO33 \\\n', 'count   9   9 ...   2   1   1   1   1   1   \n', '\n', 'Item_Identifier FDK57 FDT35 FDN52 FDE52 \n', 'count   1   1   1   1 \n', '\n', '[1 rows x 1559 columns]']}, 'metadata': {}, 'output_type': 'display_data'}, {'name': 'stdout', 'output_type': 'stream', 'text': ['\n', 'Item_Fat_Content:\n', ' Número de categorías: 5\n', " Categorías: ['Low Fat' 'Regular' 'low fat' 'LF' 'reg']\n", ' Frecuencias:\n']}, {'data': {'text/html': ['<div>\n', '<style scoped>\n', '   .dataframe tbody tr th:only-of-type {\n', '   vertical-align: middle;\n', '   }\n', '\n', '   .dataframe tbody tr th {\n', '   vertical-align: top;\n', '   }\n', '\n', '   .dataframe thead th {\n', '   text-align: right;\n', '   }\n', '</style>\n', '<table border="1" class="dataframe">\n', ' <thead>\n', '   <tr style="text-align: right;">\n', '   <th>Item_Fat_Content</th>\n', '   <th>Low Fat</th>\n', '   <th>Regular</th>\n', '   <th>LF</th>\n', '   <th>reg</th>\n', '   <th>low fat</th>\n', '   </tr>\n', ' </thead>\n', ' <tbody>\n', '   <tr>\n', '   <th>count</th>\n', '   <td>5089</td>\n', '   <td>2889</td>\n', '   <td>316</td>\n', '   <td>117</td>\n', '   <td>112</td>\n', '   </tr>\n', ' </tbody>\n', '</table>\n', '</div>'], 'text/plain': ['Item_Fat_Content Low Fat Regular   LF reg low fat\n', 'count   5089   2889 316 117   112']}, 'metadata': {}, 'output_type': 'display_data'}, {'name': 'stdout', 'output_type': 'stream', 'text': ['\n', 'Item_Type:\n', ' Número de categorías: 16\n', " Categorías: ['Dairy' 'Soft Drinks' 'Meat' 'Fruits and Vegetables' 'Household'\n", " 'Baking Goods' 'Snack Foods' 'Frozen Foods' 'Breakfast'\n", " 'Health and Hygiene' 'Hard Drinks' 'Canned' 'Breads' 'Starchy Foods'\n", " 'Others' 'Seafood']\n", ' Frecuencias:\n']}, {'data': {'text/html': ['<div>\n', '<style scoped>\n', '   .dataframe tbody tr th:only-of-type {\n', '   vertical-align: middle;\n', '   }\n', '\n', '   .dataframe tbody tr th {\n', '   vertical-align: top;\n', '   }\n', '\n', '   .dataframe thead th {\n', '   text-align: right;\n', '   }\n', '</style>\n', '<table border="1" class="dataframe">\n', ' <thead>\n', '   <tr style="text-align: right;">\n', '   <th>Item_Type</th>\n', '   <th>Fruits and Vegetables</th>\n', '   <th>Snack Foods</th>\n', '   <th>Household</th>\n', '   <th>Frozen Foods</th>\n', '   <th>Dairy</th>\n', '   <th>Canned</th>\n', '   <th>Baking Goods</th>\n', '   <th>Health and Hygiene</th>\n', '   <th>Soft Drinks</th>\n', '   <th>Meat</th>\n', '   <th>Breads</th>\n', '   <th>Hard Drinks</th>\n', '   <th>Others</th>\n', '   <th>Starchy Foods</th>\n', '   <th>Breakfast</th>\n', '   <th>Seafood</th>\n', '   </tr>\n', ' </thead>\n', ' <tbody>\n', '   <tr>\n', '   <th>count</th>\n', '   <td>1232</td>\n', '   <td>1200</td>\n', '   <td>910</td>\n', '   <td>856</td>\n', '   <td>682</td>\n', '   <td>649</td>\n', '   <td>648</td>\n', '   <td>520</td>\n', '   <td>445</td>\n', '   <td>425</td>\n', '   <td>251</td>\n', '   <td>214</td>\n', '   <td>169</td>\n', '   <td>148</td>\n', '   <td>110</td>\n', '   <td>64</td>\n', '   </tr>\n', ' </tbody>\n', '</table>\n', '</div>'], 'text/plain': ['Item_Type Fruits and Vegetables Snack Foods Household Frozen Foods Dairy \\\n', 'count   1232   1200   910   856   682   \n', '\n', 'Item_Type Canned Baking Goods Health and Hygiene Soft Drinks Meat \\\n', 'count   649   648   520

445 425 \n', '\n', 'Item_Type Breads Hard Drinks Others Starchy Foods Breakfast Seafood \n', 'count 251 214 169 148 110 64 ']}, 'metadata': {}, 'output_type': 'display_data'}, {'name': 'stdout', 'output_type': 'stream', 'text': ['\n', 'Outlet_Identifier:\n', ' Número de categorías: 10\n', " Categorías: ['OUT049' 'OUT018' 'OUT010' 'OUT013' 'OUT027' 'OUT045' 'OUT017' 'OUT046'\n", " 'OUT035' 'OUT019']\n", ' Frecuencias:\n']}, {'data': {'text/html': ['<div>\n', '<style scoped>\n', ' .dataframe tbody tr th:only-of-type {\n', ' vertical-align: middle;\n', ' }\n', '\n', ' .dataframe tbody tr th {\n', ' vertical-align: top;\n', ' }\n', '\n', ' .dataframe thead th {\n', ' text-align: right;\n', ' }\n', '</style>\n', '<table border="1" class="dataframe">\n', ' <thead>\n', ' <tr style="text-align: right;">\n', ' <th>Outlet_Identifier</th>\n', ' <th>OUT027</th>\n', ' <th>OUT013</th>\n', ' <th>OUT035</th>\n', ' <th>OUT049</th>\n', ' <th>OUT046</th>\n', ' <th>OUT045</th>\n', ' <th>OUT018</th>\n', ' <th>OUT017</th>\n', ' <th>OUT010</th>\n', ' <th>OUT019</th>\n', ' </tr>\n', ' </thead>\n', ' <tbody>\n', ' <tr>\n', ' <th>count</th>\n', ' <td>935</td>\n', ' <td>932</td>\n', ' <td>930</td>\n', ' <td>930</td>\n', ' <td>930</td>\n', ' <td>929</td>\n', ' <td>928</td>\n', ' <td>926</td>\n', ' <td>555</td>\n', ' <td>528</td>\n', ' </tr>\n', ' </tbody>\n', '</table>\n', '</div>'], 'text/plain': ['Outlet_Identifier OUT027 OUT013 OUT035 OUT049 OUT046 OUT045 OUT018 \\\n', 'count 935 932 930 930 930 929 928 \n', '\n', 'Outlet_Identifier OUT017 OUT010 OUT019 \n', 'count 926 555 528 ']}, 'metadata': {}, 'output_type': 'display_data'}, {'name': 'stdout', 'output_type': 'stream', 'text': ['\n', 'Outlet_Size:\n', ' Número de categorías: 3\n', " Categorías: ['Medium' nan 'High' 'Small']\n", ' Frecuencias:\n']}, {'data': {'text/html': ['<div>\n', '<style scoped>\n', ' .dataframe tbody tr th:only-of-type {\n', ' vertical-align: middle;\n', ' }\n', '\n', ' .dataframe tbody tr th {\n', ' vertical-align: top;\n', ' }\n', '\n', ' .dataframe thead th {\n', ' text-align: right;\n', ' }\n', '</style>\n', '<table border="1" class="dataframe">\n', ' <thead>\n', ' <tr style="text-align: right;">\n', ' <th>Outlet_Size</th>\n', ' <th>Medium</th>\n', ' <th>Small</th>\n', ' <th>High</th>\n', ' </tr>\n', ' </thead>\n', ' <tbody>\n', ' <tr>\n', ' <th>count</th>\n', ' <td>2793</td>\n', ' <td>2388</td>\n', ' <td>932</td>\n', ' </tr>\n', ' </tbody>\n', '</table>\n', '</div>'], 'text/plain': ['Outlet_Size Medium Small High\n', 'count 2793 2388 932']}, 'metadata': {}, 'output_type': 'display_data'}, {'name': 'stdout', 'output_type': 'stream', 'text': ['\n', 'Outlet_Location_Type:\n', ' Número de categorías: 3\n', " Categorías: ['Tier 1' 'Tier 3' 'Tier 2']\n", ' Frecuencias:\n']}, {'data': {'text/html': ['<div>\n', '<style scoped>\n', ' .dataframe tbody tr th:only-of-type {\n', ' vertical-align: middle;\n', ' }\n', '\n', ' .dataframe tbody tr th {\n', ' vertical-align: top;\n', ' }\n', '\n', ' .dataframe thead th {\n', ' text-align: right;\n', ' }\n', '</style>\n', '<table border="1" class="dataframe">\n', ' <thead>\n', ' <tr style="text-align: right;">\n', ' <th>Outlet_Location_Type</th>\n', ' <th>Tier 3</th>\n', ' <th>Tier 2</th>\n', ' <th>Tier 1</th>\n', ' </tr>\n', ' </thead>\n', ' <tbody>\n', ' <tr>\n', ' <th>count</th>\n', ' <td>3350</td>\n', ' <td>2785</td>\n', ' <td>2388</td>\n', ' </tr>\n', ' </tbody>\n', '</table>\n', '</div>'], 'text/plain': ['Outlet_Location_Type Tier 3 Tier 2 Tier 1\n', 'count 3350 2785 2388']}, 'metadata': {}, 'output_type': 'display_data'}, {'name': 'stdout', 'output_type': 'stream', 'text': ['\n', 'Outlet_Type:\n', ' Número de categorías: 4\n', " Categorías: ['Supermarket Type1' 'Supermarket Type2' 'Grocery Store'\n", " 'Supermarket Type3']\n", ' Frecuencias:\n']}, {'data': {'text/html': ['<div>\n', '<style scoped>\n', ' .dataframe tbody tr th:only-of-type {\n', ' vertical-align: middle;\n', ' }\n', '\n', ' .dataframe tbody tr th {\n', ' vertical-align: top;\n', ' }\n', '\n', ' .dataframe thead th {\n', ' text-align: right;\n', ' }\n', '</style>\n', '<table border="1" class="dataframe">\n', ' <thead>\n', ' <tr style="text-align: right;">\n', '

<th>Outlet_Type</th>\n', '   <th>Supermarket Type1</th>\n', '   <th>Grocery Store</th>\n', '   <th>Supermarket Type3</th>\n', '   <th>Supermarket Type2</th>\n', '   </tr>\n', ' </thead>\n', ' <tbody>\n', '   <tr>\n', '   <th>count</th>\n', '   <td>5577</td>\n', '   <td>1083</td>\n', '   <td>935</td>\n', '   <td>928</td>\n', '   </tr>\n', ' </tbody>\n', '</table>\n', '</div>'], 'text/plain': ['Outlet_Type Supermarket Type1 Grocery Store Supermarket Type3 \\\n', 'count   5577   1083   935   \n', '\n', 'Outlet_Type Supermarket Type2 \n', 'count   928 ']}, 'metadata': {}, 'output_type': 'display_data'}], 'source': ['# Seleccionar variables categóricas\n', "categorical_cols = df.select_dtypes(include=['object']).columns.tolist()\n", 'print(f"Variables categóricas: {categorical_cols}")\n', '\n', '# Estadísticas de variables categóricas\n', 'print("\\nESTADÍSTICAS DESCRIPTIVAS - VARIABLES CATEGÓRICAS")\n', 'print("="*70)\n', 'for col in categorical_cols:\n', '   print(f"\\n{col}:")\n', '   print(f" Número de categorías: {df[col].nunique()}")\n', '   print(f" Categorías: {df[col].unique()}")\n', '   print(f" Frecuencias:")\n', '   display(pd.DataFrame(df[col].value_counts()).T)']}, {'cell_type': 'code', 'execution_count': 8, 'metadata': {}, 'outputs': [{'data': {'text/plain': ['<Figure size 1800x1200 with 6 Axes>']}, 'metadata': {}, 'output_type': 'display_data'}, {'name': 'stdout', 'output_type': 'stream', 'text': ['\n', 'HALLazgos principales variables categóricas:\n', '--\n', '• Item_Identifier: 1559 productos únicos (variable de identificación)\n', '• Item_Fat_Content: Se observan inconsistencias en categorías (Low Fat, LF, low fat)\n', '• Item_Type: 16 categorías de productos, siendo Fruits and Vegetables la más común\n', '• Outlet_Identifier: 10 tiendas únicas\n', '• Outlet_Size: Presencia de valores nulos\n', '• Outlet_Location_Type: 3 tipos de ubicación\n', '• Outlet_Type: 4 tipos de tienda\n']}], 'source': ['# Visualización de variables categóricas\n', 'fig, axes = plt.subplots(2, 3, figsize=(18, 12))\n', 'axes = axes.ravel()\n', '\n', 'for i, col in enumerate(categorical_cols):\n', '   if i < len(axes):\n', '   # Para variables con muchas categorías, mostrar solo las top 10\n', '   if df[col].nunique() > 10:\n', '   top_categories = df[col].value_counts().head(10)\n', "   top_categories.plot(kind='bar', ax=axes[i], alpha=0.7)\n", "   axes[i].set_title(f'Top 10 {col}')\n", '   else:\n', "   df[col].value_counts().plot(kind='bar', ax=axes[i], alpha=0.7)\n", "   axes[i].set_title(f'Distribución de {col}')\n", "   axes[i].tick_params(axis='x', rotation=45)\n", "   axes[i].set_ylabel('Frecuencia')\n", '\n', '# Eliminar ejes vacíos\n', 'for i in range(len(categorical_cols), len(axes)):\n', '   fig.delaxes(axes[i])\n', '\n', 'plt.tight_layout()\n', 'plt.show()\n', '\n', 'print("\\nHALLazgos principales variables categóricas:")\n', 'print("-" * 50)\n', 'print("• Item_Identifier: 1559 productos únicos (variable de identificación)")\n', 'print("• Item_Fat_Content: Se observan inconsistencias en categorías (Low Fat, LF, low fat)")\n', 'print("• Item_Type: 16 categorías de productos, siendo Fruits and Vegetables la más común")\n', 'print("• Outlet_Identifier: 10 tiendas únicas")\n', 'print("• Outlet_Size: Presencia de valores nulos")\n', 'print("• Outlet_Location_Type: 3 tipos de ubicación")\n', 'print("• Outlet_Type: 4 tipos de tienda")']}, {'cell_type': 'markdown', 'metadata': {}, 'source': ['## 4. Análisis de Valores Faltantes']}, {'cell_type': 'code', 'execution_count': 9, 'metadata': {}, 'outputs': [{'name': 'stdout', 'output_type': 'stream', 'text': ['ANÁLISIS DE VALORES FALTANTES\n', '==\n', 'Columnas con valores faltantes:\n']}, {'data': {'text/html': ['<div>\n', '<style scoped>\n', '   .dataframe tbody tr th:only-of-type {\n', '   vertical-align: middle;\n', '   }\n', '\n', '   .dataframe tbody tr th {\n', '   vertical-align: top;\n', '   }\n', '\n', '   .dataframe thead th {\n', '   text-align: right;\n', '   }\n', '</style>\n', '<table border="1" class="dataframe">\n', ' <thead>\n', '   <tr style="text-align: right;">\n', '   <th></th>\n', '   <th>Columna</th>\n', '   <th>Valores Faltantes</th>\n', '   <th>Porcentaje</th>\n', '   </tr>\n', ' </thead>\n', '

<tbody>\n', '   <tr>\n', '   <th>8</th>\n', '   <td>Outlet_Size</td>\n', '   <td>2410</td>\n', '   <td>28.276428</td>\n', '   </tr>\n', '   <tr>\n', '   <th>1</th>\n', '   <td>Item_Weight</td>\n', '   <td>1463</td>\n', '   <td>17.165317</td>\n', '   </tr>\n', ' </tbody>\n', '</table>\n', '</div>'], 'text/plain': ['   Columna Valores Faltantes Porcentaje\n', '8 Outlet_Size   2410   28.276428\n', '1 Item_Weight   1463   17.165317']}, 'metadata': {}, 'output_type': 'display_data'}, {'data': {'text/plain': ['<Figure size 1000x600 with 2 Axes>']}, 'metadata': {}, 'output_type': 'display_data'}, {'name': 'stdout', 'output_type': 'stream', 'text': ['\n', 'ANÁLISIS POR COLUMNA:\n', '------------------------------\n', 'Outlet_Size: 2410 valores faltantes (28.28%)\n', 'Item_Weight: 1463 valores faltantes (17.17%)\n']}], 'source': ['# Análisis de valores faltantes\n', 'print("ANÁLISIS DE VALORES FALTANTES")\n', 'print("="*50)\n', '\n', 'missing_data = df.isnull().sum()\n', 'missing_percent = (missing_data / len(df)) * 100\n', '\n', 'missing_df = pd.DataFrame({\n', "   'Columna': missing_data.index,\n", "   'Valores Faltantes': missing_data.values,\n", "   'Porcentaje': missing_percent.values\n", '})\n', "missing_df = missing_df[missing_df['Valores Faltantes'] > 0].sort_values('Porcentaje', ascending=False)\n", '\n', 'if len(missing_df) > 0:\n', '   print("Columnas con valores faltantes:")\n', '   display(missing_df)\n', 'else:\n', '   print("✓ No hay valores faltantes en el dataset")\n', '\n', '# Visualización de valores faltantes\n', 'if len(missing_df) > 0:\n', '   plt.figure(figsize=(10, 6))\n', "   sns.heatmap(df.isnull(), cbar=True, yticklabels=False, cmap='viridis')\n", "   plt.title('Mapa de Valores Faltantes')\n", '   plt.show()\n', '\n', '   print("\\nANÁLISIS POR COLUMNA:")\n', '   print("-" * 30)\n', "   for col in missing_df['Columna']:\n", '   print(f"{col}: {missing_df[missing_df[\'Columna\'] == col][\'Valores Faltantes\'].values[0]} valores faltantes "\n', '   f"({missing_df[missing_df[\'Columna\'] == col][\'Porcentaje\'].values[0]:.2f}%)")']}, {'cell_type': 'markdown', 'metadata': {}, 'source': ['## 5. Análisis de Registros Duplicados']}, {'cell_type': 'code', 'execution_count': 10, 'metadata': {}, 'outputs': [{'name': 'stdout', 'output_type': 'stream', 'text': ['ANÁLISIS DE REGISTROS DUPLICADOS\n', '==\n', 'Registros duplicados exactos: 0\n', 'Duplicados en combinación Producto-Tienda: 0\n', '\n', '✓ No hay duplicados en la combinación Producto-Tienda\n']}], 'source': ['# Análisis de duplicados\n', 'print("ANÁLISIS DE REGISTROS DUPLICADOS")\n', 'print("="*50)\n', '\n', '# Duplicados exactos\n', 'exact_duplicates = df.duplicated().sum()\n', 'print(f"Registros duplicados exactos: {exact_duplicates}")\n', '\n', '# Duplicados por combinación producto-tienda (que deberían ser únicos)\n', "key_duplicates = df.duplicated(subset=['Item_Identifier', 'Outlet_Identifier']).sum()\n", 'print(f"Duplicados en combinación Producto-Tienda: {key_duplicates}")\n', '\n', 'if key_duplicates > 0:\n', '   print("\\n¡ALERTA! Se encontraron duplicados en la llave natural (Producto-Tienda)")\n', '   print("Esto podría indicar problemas en la calidad de los datos")\n', '   \n', '   # Mostrar duplicados\n', "   duplicates = df[df.duplicated(subset=['Item_Identifier', 'Outlet_Identifier'], keep=False)]\n", '   print("\\nRegistros duplicados:")\n', "   display(duplicates.sort_values(['Item_Identifier', 'Outlet_Identifier']))\n", 'else:\n', '   print("\\n✓ No hay duplicados en la combinación Producto-Tienda")']}, {'cell_type': 'markdown', 'metadata': {}, 'source': ['## 6. Análisis de la Variable Objetivo (Item_Outlet_Sales)']}, {'cell_type': 'code', 'execution_count': 11, 'metadata': {}, 'outputs': [{'name': 'stdout', 'output_type': 'stream', 'text': ['ANÁLISIS DE LA VARIABLE OBJETIVO: Item_Outlet_Sales\n', '==\n', 'Estadísticas descriptivas:\n']}, {'data': {'text/html': ['<div>\n', '<style scoped>\n', '   .dataframe tbody tr th:only-of-type {\n', '   vertical-align: middle;\n', '   }\n', '\n', '   .dataframe tbody tr th {\n', '   vertical-align: top;\n', '   }\n', '\n', '   .dataframe thead th {\n', '   text-align:

right;\n', '   }\n', '</style>\n', '<table border="1" class="dataframe">\n', ' <thead>\n', '   <tr style="text-align: right;">\n', '   <th></th>\n', '   <th>count</th>\n', '   <th>mean</th>\n', '   <th>std</th>\n', '   <th>min</th>\n', '   <th>25%</th>\n', '   <th>50%</th>\n', '   <th>75%</th>\n', '   <th>max</th>\n', '   </tr>\n', ' </thead>\n', ' <tbody>\n', '   <tr>\n', '   <th>Item_Outlet_Sales</th>\n', '   <td>8523.0</td>\n', '   <td>2181.288914</td>\n', '   <td>1706.499616</td>\n', '   <td>33.29</td>\n', '   <td>834.2474</td>\n', '   <td>1794.331</td>\n', '   <td>3101.2964</td>\n', '   <td>13086.9648</td>\n', '   </tr>\n', ' </tbody>\n', '</table>\n', '</div>'], 'text/plain': ['   count   mean   std   min   25% \\\n', 'Item_Outlet_Sales 8523.0 2181.288914 1706.499616 33.29 834.2474   \n', '\n', '   50%   75%   max \n', 'Item_Outlet_Sales 1794.331 3101.2964 13086.9648 ']}, 'metadata': {}, 'output_type': 'display_data'}, {'data': {'text/plain': ['<Figure size 1500x500 with 2 Axes>']}, 'metadata': {}, 'output_type': 'display_data'}, {'name': 'stdout', 'output_type': 'stream', 'text': ['\n', 'ANÁLISIS DE BALANCE:\n', '------------------------------\n', 'Asimetría: 1.18\n', '\n', 'Distribución por rangos de ventas:\n', ' (20.236, 2644.025]: 5728 registros (67.2%)\n', ' (2644.025, 5254.76]: 2256 registros (26.5%)\n', ' (5254.76, 7865.495]: 483 registros (5.7%)\n', ' (7865.495, 10476.23]: 52 registros (0.6%)\n', ' (10476.23, 13086.965]: 4 registros (0.0%)\n', '\n', '→ La variable objetivo está altamente sesgada, considerando transformación logarítmica\n']}], 'source': ['# Análisis de la variable objetivo\n', "target_var = 'Item_Outlet_Sales'\n", 'print(f"ANÁLISIS DE LA VARIABLE OBJETIVO: {target_var}")\n', 'print("="*60)\n', '\n', '# Estadísticas descriptivas\n', 'print("Estadísticas descriptivas:")\n', 'target_stats = df[target_var].describe()\n', 'display(pd.DataFrame(target_stats).T)\n', '\n', '# Distribución\n', 'fig, axes = plt.subplots(1, 2, figsize=(15, 5))\n', '\n', '# Histograma\n', 'df[target_var].hist(bins=50, ax=axes[0], alpha=0.7)\n', "axes[0].axvline(df[target_var].mean(), color='red', linestyle='--', label=f'Media: {df[target_var].mean():.2f}')\n", "axes[0].axvline(df[target_var].median(), color='green', linestyle='--', label=f'Mediana: {df[target_var].median():.2f}')\n", "axes[0].set_title('Distribución de Item_Outlet_Sales')\n", "axes[0].set_xlabel('Ventas')\n", "axes[0].set_ylabel('Frecuencia')\n", 'axes[0].legend()\n', '\n', '# Boxplot\n', 'df.boxplot(column=target_var, ax=axes[1])\n', "axes[1].set_title('Boxplot de Item_Outlet_Sales')\n", '\n', 'plt.tight_layout()\n', 'plt.show()\n', '\n', '# Análisis de balance\n', 'print("\\nANÁLISIS DE BALANCE:")\n', 'print("-" * 30)\n', 'target_skew = df[target_var].skew()\n', 'print(f"Asimetría: {target_skew:.2f}")\n', '\n', '# Crear categorías para análisis de distribución\n', 'sales_bins = pd.cut(df[target_var], bins=5)\n', 'bin_counts = sales_bins.value_counts().sort_index()\n', 'print("\\nDistribución por rangos de ventas:")\n', 'for bin_range, count in bin_counts.items():\n', '   percentage = (count / len(df)) * 100\n', '   print(f" {bin_range}: {count} registros ({percentage:.1f}%)")\n', '\n', 'if abs(target_skew) > 1:\n', '   print("\\n→ La variable objetivo está altamente sesgada, considerando transformación logarítmica")\n', 'elif abs(target_skew) > 0.5:\n', '   print("\\n→ La variable objetivo está moderadamente sesgada")']}, {'cell_type': 'markdown', 'metadata': {}, 'source': ['## 7. Análisis de Correlaciones entre Variables Numéricas']}, {'cell_type': 'code', 'execution_count': 12, 'metadata': {}, 'outputs': [{'name': 'stdout', 'output_type': 'stream', 'text': ['MATRIZ DE CORRELACIÓN - VARIABLES NUMÉRICAS\n', '==\n']}, {'data': {'text/plain': ['<Figure size 1000x800 with 2 Axes>']}, 'metadata': {}, 'output_type': 'display_data'}, {'name': 'stdout', 'output_type': 'stream', 'text': ['\n', 'CORRELACIONES CON LA VARIABLE OBJETIVO (Item_Outlet_Sales):\n', '--\n', 'Item_MRP: 0.568 (FUERTE correlación positiva)\n', 'Item_Weight: 0.014 (DÉBIL correlación positiva)\n', 'Outlet_Establishment_Year: -0.049 (DÉBIL correlación negativa)\n', 'Item_Visibility: -0.129 (DÉBIL correlación negativa)\n', '\n', 'CORRELACIONES ENTRE VARIABLES PREDICTORAS (|corr| > 0.5):\n', '--\n', 'No hay correlaciones fuertes entre variables predictoras\n']}], 'source': ['# Matriz de correlación\n', 'print("MATRIZ DE CORRELACIÓN - VARIABLES NUMÉRICAS")\n', 'print("="*60)\n', '\n', 'correlation_matrix = df[numeric_cols].corr()\n', '\n', '# Visualización\n', 'plt.figure(figsize=(10, 8))\n', "sns.heatmap(correlation_matrix, annot=True, cmap='coolwarm', center=0,\n", '   square=True, linewidths=0.5)\n', "plt.title('Matriz de Correlación entre Variables Numéricas')\n", 'plt.tight_layout()\n', 'plt.show()\n', '\n', '# Correlaciones más fuertes con la variable objetivo\n', 'print("\\nCORRELACIONES CON LA VARIABLE OBJETIVO (Item_Outlet_Sales):")\n', 'print("-" * 70)\n', 'target_correlations = correlation_matrix[target_var].sort_values(ascending=False)\n', 'for var, corr in target_correlations.items():\n', '   if var != target_var:\n', '   strength = "FUERTE" if abs(corr) > 0.5 else "MODERADA" if abs(corr) > 0.3 else "DÉBIL"\n', '   direction = "positiva" if corr > 0 else "negativa"\n', '   print(f"{var}: {corr:.3f} ({strength} correlación {direction})")\n', '\n', '# Correlaciones entre variables predictoras\n', 'print("\\nCORRELACIONES ENTRE VARIABLES PREDICTORAS (|corr| > 0.5):")\n', 'print("-" * 70)\n', 'high_corr_pairs = []\n', 'for i in range(len(correlation_matrix.columns)):\n', '   for j in range(i+1, len(correlation_matrix.columns)):\n', '   if abs(correlation_matrix.iloc[i, j]) > 0.5 and correlation_matrix.columns[i] != target_var and correlation_matrix.columns[j] != target_var:\n', '   high_corr_pairs.append((\n', '   correlation_matrix.columns[i], \n', '   correlation_matrix.columns[j], \n', '   correlation_matrix.iloc[i, j]\n', '   ))\n', '\n', 'if high_corr_pairs:\n', '   for var1, var2, corr in high_corr_pairs:\n', '   print(f"{var1} - {var2}: {corr:.3f}")\n', 'else:\n', '   print("No hay correlaciones fuertes entre variables predictoras")']}, {'cell_type': 'markdown', 'metadata': {}, 'source': ['## 8. Análisis de Outliers']}, {'cell_type': 'code', 'execution_count': 13, 'metadata': {}, 'outputs': [{'name': 'stdout', 'output_type': 'stream', 'text': ['ANÁLISIS DE OUTLIERS EN VARIABLES NUMÉRICAS\n', '==\n']}, {'data': {'text/html': ['<div>\n', '<style scoped>\n', '   .dataframe tbody tr th:only-of-type {\n', '   vertical-align: middle;\n', '   }\n', '\n', '   .dataframe tbody tr th {\n', '   vertical-align: top;\n', '   }\n', '\n', '   .dataframe thead th {\n', '   text-align: right;\n', '   }\n', '</style>\n', '<table border="1" class="dataframe">\n', ' <thead>\n', '   <tr style="text-align: right;">\n', '   <th></th>\n', '   <th>outliers_count</th>\n', '   <th>outliers_percent</th>\n', '   <th>lower_bound</th>\n', '   <th>upper_bound</th>\n', '   </tr>\n', ' </thead>\n', ' <tbody>\n', '   <tr>\n', '   <th>Item_Outlet_Sales</th>\n', '   <td>186.0</td>\n', '   <td>2.182330</td>\n', '   <td>-2566.326100</td>\n', '   <td>6501.869900</td>\n', '   </tr>\n', '   <tr>\n', '   <th>Item_Visibility</th>\n', '   <td>144.0</td>\n', '   <td>1.689546</td>\n', '   <td>-0.074404</td>\n', '   <td>0.195979</td>\n', '   </tr>\n', '   <tr>\n', '   <th>Item_Weight</th>\n', '   <td>0.0</td>\n', '   <td>0.000000</td>\n', '   <td>-3.340625</td>\n', '   <td>28.964375</td>\n', '   </tr>\n', '   <tr>\n', '   <th>Item_MRP</th>\n', '   <td>0.0</td>\n', '   <td>0.000000</td>\n', '   <td>-43.899300</td>\n', '   <td>323.369500</td>\n', '   </tr>\n', '   <tr>\n', '   <th>Outlet_Establishment_Year</th>\n', '   <td>0.0</td>\n', '   <td>0.000000</td>\n', '   <td>1961.500000</td>\n', '   <td>2029.500000</td>\n', '   </tr>\n', ' </tbody>\n', '</table>\n', '</div>'], 'text/plain': ['   outliers_count outliers_percent lower_bound \\\n', 'Item_Outlet_Sales   186.0   2.182330 -2566.326100   \n', 'Item_Visibility   144.0   1.689546   -0.074404   \n', 'Item_Weight   0.0   0.000000   -3.340625   \n', 'Item_MRP   0.0   0.000000

-43.899300 \n', 'Outlet_Establishment_Year 0.0 0.000000 1961.500000 \n', '\n', ' upper_bound \n', 'Item_Outlet_Sales 6501.869900 \n', 'Item_Visibility 0.195979 \n', 'Item_Weight 28.964375 \n', 'Item_MRP 323.369500 \n', 'Outlet_Establishment_Year 2029.500000 ']}, 'metadata': {}, 'output_type': 'display_data'}, {'data': {'text/plain': ['<Figure size 1500x1000 with 5 Axes>']}, 'metadata': {}, 'output_type': 'display_data'}, {'name': 'stdout', 'output_type': 'stream', 'text': ['\n', 'VARIABLES CON MÁS OUTLIERS:\n', '--\n']}], 'source': ['# Análisis de outliers usando IQR\n', 'print("ANÁLISIS DE OUTLIERS EN VARIABLES NUMÉRICAS")\n', 'print("="*60)\n', '\n', 'outliers_summary = {}\n', '\n', 'for col in numeric_cols:\n', '   Q1 = df[col].quantile(0.25)\n', '   Q3 = df[col].quantile(0.75)\n', '   IQR = Q3 - Q1\n', '   lower_bound = Q1 - 1.5 * IQR\n', '   upper_bound = Q3 + 1.5 * IQR\n', '   \n', '   outliers = df[(df[col] < lower_bound) | (df[col] > upper_bound)]\n', '   outliers_count = len(outliers)\n', '   outliers_percent = (outliers_count / len(df)) * 100\n', '   \n', '   outliers_summary[col] = {\n', "   'outliers_count': outliers_count,\n", "   'outliers_percent': outliers_percent,\n", "   'lower_bound': lower_bound,\n", "   'upper_bound': upper_bound\n", '   }\n', '\n', '# Mostrar resumen\n', 'outliers_df = pd.DataFrame(outliers_summary).T\n', "outliers_df = outliers_df.sort_values('outliers_percent', ascending=False)\n", 'display(outliers_df)\n', '\n', '# Visualización de outliers\n', 'fig, axes = plt.subplots(2, 3, figsize=(15, 10))\n', 'axes = axes.ravel()\n', '\n', 'for i, col in enumerate(numeric_cols):\n', '   if i < len(axes):\n', '   df.boxplot(column=col, ax=axes[i])\n', "   axes[i].set_title(f'Outliers en {col}')\n", '\n', '# Eliminar ejes vacíos\n', 'for i in range(len(numeric_cols), len(axes)):\n', '   fig.delaxes(axes[i])\n', '\n', 'plt.tight_layout()\n', 'plt.show()\n', '\n', 'print("\\nVARIABLES CON MÁS OUTLIERS:")\n', 'print("-" * 40)\n', 'for col, stats in outliers_summary.items():\n', "   if stats['outliers_percent'] > 5:\n", '   print(f"{col}: {stats[\'outliers_count\']} outliers ({stats[\'outliers_percent\']:.1f}%)")']}, {'cell_type': 'markdown', 'metadata': {}, 'source': ['## 9. Análisis Bivariado: Relación entre Variables y Target']}, {'cell_type': 'code', 'execution_count': 14, 'metadata': {}, 'outputs': [{'name': 'stdout', 'output_type': 'stream', 'text': ['ANÁLISIS BIVARIADO: VARIABLES NUMÉRICAS VS TARGET\n', '==\n']}, {'data': {'text/plain': ['<Figure size 1500x1200 with 4 Axes>']}, 'metadata': {}, 'output_type': 'display_data'}], 'source': ['# Análisis bivariado: Variables numéricas vs Target\n', 'print("ANÁLISIS BIVARIADO: VARIABLES NUMÉRICAS VS TARGET")\n', 'print("="*70)\n', '\n', '# Scatter plots para variables numéricas vs target\n', 'predictor_numeric = [col for col in numeric_cols if col != target_var]\n', '\n', 'fig, axes = plt.subplots(2, 2, figsize=(15, 12))\n', 'axes = axes.ravel()\n', '\n', 'for i, col in enumerate(predictor_numeric):\n', '   if i < len(axes):\n', '   axes[i].scatter(df[col], df[target_var], alpha=0.5)\n', '   axes[i].set_xlabel(col)\n', '   axes[i].set_ylabel(target_var)\n', "   axes[i].set_title(f'{col} vs {target_var}')\n", '   \n', '   # Calcular correlación\n', '   corr = df[col].corr(df[target_var])\n', "   axes[i].text(0.05, 0.95, f'Corr: {corr:.3f}', \n", '   transform=axes[i].transAxes, bbox=dict(boxstyle="round", facecolor=\'wheat\', alpha=0.8))\n', '\n', 'plt.tight_layout()\n', 'plt.show()']}, {'cell_type': 'code', 'execution_count': 15, 'metadata': {}, 'outputs': [{'name': 'stdout', 'output_type': 'stream', 'text': ['ANÁLISIS BIVARIADO: VARIABLES CATEGÓRICAS VS TARGET\n', '==\n', '\n', 'VENTAS PROMEDIO POR ITEM_FAT_CONTENT:\n', ' Regular: 2235.19\n', ' Low Fat: 2164.48\n', ' low fat: 2087.74\n', ' LF: 2073.55\n', ' reg: 1962.19\n', '\n', 'VENTAS PROMEDIO POR OUTLET_SIZE:\n', ' Medium: 2681.60\n', ' High: 2299.00\n', ' Small: 1912.15\n', '\n', 'VENTAS PROMEDIO POR OUTLET_LOCATION_TYPE:\n', ' Tier 2:

2323.99\n', ' Tier 3: 2279.63\n', ' Tier 1: 1876.91\n', '\n', 'VENTAS PROMEDIO POR OUTLET_TYPE:\n', ' Supermarket Type3: 3694.04\n', ' Supermarket Type1: 2316.18\n', ' Supermarket Type2: 1995.50\n', ' Grocery Store: 339.83\n']}, {'data': {'text/plain': ['<Figure size 1500x1200 with 4 Axes>']}, 'metadata': {}, 'output_type': 'display_data'}], 'source': ['# Análisis bivariado: Variables categóricas vs Target\n', 'print("ANÁLISIS BIVARIADO: VARIABLES CATEGÓRICAS VS TARGET")\n', 'print("="*70)\n', '\n', '# Seleccionar variables categóricas con menos categorías para mejor visualización\n', 'categorical_for_analysis = [col for col in categorical_cols \n', "   if col not in ['Item_Identifier', 'Outlet_Identifier'] \n", '   and df[col].nunique() <= 10]\n', '\n', 'fig, axes = plt.subplots(2, 2, figsize=(15, 12))\n', 'axes = axes.ravel()\n', '\n', 'for i, col in enumerate(categorical_for_analysis):\n', '   if i < len(axes):\n', '   # Boxplot por categoría\n', '   df.boxplot(column=target_var, by=col, ax=axes[i])\n', "   axes[i].set_title(f'Ventas por {col}')\n", "   axes[i].set_ylabel('Ventas')\n", '   \n', '   # Calcular ventas promedio por categoría\n', '   avg_sales = df.groupby(col)[target_var].mean().sort_values(ascending=False)\n', '   print(f"\\nVENTAS PROMEDIO POR {col.upper()}:")\n', '   for category, sales in avg_sales.items():\n', '   print(f" {category}: {sales:.2f}")\n', '\n', "plt.suptitle('') # Eliminar título automático\n", 'plt.tight_layout()\n', 'plt.show()']}, {'cell_type': 'markdown', 'metadata': {}, 'source': ['## Resumen Ejecutivo del EDA']}, {'cell_type': 'code', 'execution_count': 16, 'metadata': {}, 'outputs': [{'name': 'stdout', 'output_type': 'stream', 'text': ['RESUMEN EJECUTIVO - ANÁLISIS EXPLORATORIO DE DATOS\n', '==\n', '\n', 'HALLazgos PRINCIPALES:\n', '--\n', '1. ESTRUCTURA DEL DATASET:\n', '   • Dimensiones: 8523 registros, 12 variables\n', '   • 1559 productos únicos distribuidos en 10 tiendas\n', '   • Variables: 5 numéricas, 7 categóricas\n', '\n', '2. CALIDAD DE DATOS:\n', '   • Variables con valores faltantes: 2\n', '   • Columnas afectadas: Outlet_Size, Item_Weight\n', '   • Registros duplicados: 0\n', '   • Duplicados en clave producto-tienda: 0\n', '\n', '3. VARIABLE OBJETIVO (Item_Outlet_Sales):\n', '   • Rango: 33.29 - 13086.96\n', '   • Media: 2181.29, Mediana: 1794.33\n', '   • Asimetría: 1.18 (distribución sesgada a la derecha)\n', '\n', '4. CORRELACIONES DESTACADAS:\n', '   • Item_MRP: 0.568\n', '   • Item_Weight: 0.014\n', '\n', '5. OUTLIERS CRÍTICOS (>10%):\n', '   • No hay variables con más del 10% de outliers\n', '\n', '6. RECOMENDACIONES PARA FASE DE CLUSTERING:\n', '   • Tratar valores faltantes en Item_Weight y Outlet_Size\n', '   • Estandarizar variables numéricas debido a diferentes escalas\n', '   • Considerar transformación logarítmica para variables sesgadas\n', '   • Codificar variables categóricas para el algoritmo de clustering\n', '   • Considerar reducir dimensionalidad si hay alta correlación entre variables\n', '\n', '==\n', '✓ ANÁLISIS EXPLORATORIO COMPLETADO\n']}], 'source': ['print("RESUMEN EJECUTIVO - ANÁLISIS EXPLORATORIO DE DATOS")\n', 'print("="*70)\n', 'print("\\nHALLazgos PRINCIPALES:")\n', 'print("-" * 50)\n', '\n', '# 1. Estructura del dataset\n', 'print("1. ESTRUCTURA DEL DATASET:")\n', 'print(f"   • Dimensiones: {df.shape[0]} registros, {df.shape[1]} variables")\n', 'print(f"   • 1559 productos únicos distribuidos en 10 tiendas")\n', 'print(f"   • Variables: {len(numeric_cols)} numéricas, {len(categorical_cols)} categóricas")\n', '\n', '# 2. Calidad de datos\n', "missing_cols = missing_df['Columna'].tolist() if 'missing_df' in locals() and len(missing_df) > 0 else []\n", 'print(f"\\n2. CALIDAD DE DATOS:")\n', 'print(f"   • Variables con valores faltantes: {len(missing_cols)}")\n', 'if missing_cols:\n', '   print(f"   • Columnas afectadas: {\', \'.join(missing_cols)}")\n', 'print(f"   •

Registros duplicados: {exact_duplicates}")\n', 'print(f"   • Duplicados en clave producto-tienda: {key_duplicates}")\n', '\n', '# 3. Variable objetivo\n', 'print(f"\\n3. VARIABLE OBJETIVO (Item_Outlet_Sales):")\n', 'print(f"   • Rango: {df[target_var].min():.2f} - {df[target_var].max():.2f}")\n', 'print(f"   • Media: {df[target_var].mean():.2f}, Mediana: {df[target_var].median():.2f}")\n', 'print(f"   • Asimetría: {target_skew:.2f} (distribución sesgada a la derecha)")\n', '\n', '# 4. Correlaciones importantes\n', 'print(f"\\n4. CORRELACIONES DESTACADAS:")\n', 'top_corr = target_correlations.head(3)\n', 'for var, corr in top_corr.items():\n', '   if var != target_var:\n', '   print(f"   • {var}: {corr:.3f}")\n', '\n', '# 5. Outliers\n', "high_outliers = [(col, stats['outliers_percent']) for col, stats in outliers_summary.items() \n", "   if stats['outliers_percent'] > 10]\n", 'print(f"\\n5. OUTLIERS CRÍTICOS (>10%):")\n', 'if high_outliers:\n', '   for col, percent in high_outliers:\n', '   print(f"   • {col}: {percent:.1f}%")\n', 'else:\n', '   print("   • No hay variables con más del 10% de outliers")\n', '\n', '# 6. Recomendaciones para clustering\n', 'print(f"\\n6. RECOMENDACIONES PARA FASE DE CLUSTERING:")\n', 'print("   • Tratar valores faltantes en Item_Weight y Outlet_Size")\n', 'print("   • Estandarizar variables numéricas debido a diferentes escalas")\n', 'print("   • Considerar transformación logarítmica para variables sesgadas")\n', 'print("   • Codificar variables categóricas para el algoritmo de clustering")\n', 'print("   • Considerar reducir dimensionalidad si hay alta correlación entre variables")\n', '\n', 'print("\\n" + "="*70)\n', 'print("✓ ANÁLISIS EXPLORATORIO COMPLETADO")']}], 'metadata': {'kernelspec': {'display_name': 'Python 3', 'language': 'python', 'name': 'python3'}, 'language_info': {'codemirror_mode': {'name': 'ipython', 'version': 3}, 'file_extension': '.py', 'mimetype': 'text/x-python', 'name': 'python', 'nbconvert_exporter': 'python', 'pygments_lexer': 'ipython3', 'version': '3.12.8'}}, 'nbformat': 4, 'nbformat_minor': 4}
```

``` fase_b.ipynb
{'cells': [{'cell_type': 'markdown', 'metadata': {}, 'source': ['# Fase B: Construcción del Dataset a Nivel Producto\n', '\n', '## Objetivos de la Notebook\n', '1.  **Limpieza Granular:** Corregir inconsistencias en categorías y tratar valores nulos con lógica de negocio (no imputación ciega).\n', '2.  **Feature Engineering:** Crear categorías amplias y ajustar variables semánticas.\n', '3.  **Agregación (Paso Crítico):** Transformar el dataset de transacciones (Tienda-Producto) a un dataset de entidades (Producto único).\n', '4.  **Preparación para Clustering:** Generar dos versiones de los datos: una legible para humanos (Interpretación) y una numérica escalada para máquinas (Modelado).']}, {'cell_type': 'code', 'execution_count': 1, 'metadata': {}, 'outputs': [{'name': 'stdout', 'output_type': 'stream', 'text': ['Datos cargados correctamente. Dimensiones iniciales: (8523, 12)\n']}], 'source': ['import pandas as pd\n', 'import numpy as np\n', 'from sklearn.preprocessing import StandardScaler, OneHotEncoder\n', 'import matplotlib.pyplot as plt\n', 'import seaborn as sns\n', '\n', '# Configuración de visualización\n', "pd.set_option('display.max_columns', None)\n", '\n', '# Carga de datos\n', '# Asumimos que la estructura de carpetas es: Notebooks/fase_b.ipynb y Data/Raw/train.csv\n', 'try:\n', "    df = pd.read_csv('../Data/Raw/train.csv')\n", '    print(f"Datos cargados correctamente. Dimensiones iniciales: {df.shape}")\n', 'except FileNotFoundError:\n', '    print("Error: No se encontró el archivo. Verifica la ruta relativa.")']}, {'cell_type': 'markdown', 'metadata': {}, 'source': ['--- \n', '## 1. Limpieza de Datos Granular\n', '\n', 'Antes de agregar, debemos limpiar a nivel fila para asegurar que los promedios y conteos sean precisos.\n', '\n', '### 1.1 Estandarización de Categorías\n', "La columna `Item_Fat_Content` tiene etiquetas inconsistentes ('LF', 'low fat', 'Low Fat')."]}, {'cell_type': 'code', 'execution_count': 2, 'metadata': {}, 'outputs': [{'name': 'stdout', 'output_type': 'stream',

'text': ['Distribución corregida de Fat Content:\n', 'Item_Fat_Content\n', 'Low Fat 5517\n', 'Regular 3006\n', 'Name: count, dtype: int64\n']}], 'source': ['# Mapeo de corrección\n', 'fat_content_map = {\n', " 'low fat': 'Low Fat',\n", " 'LF': 'Low Fat',\n", " 'reg': 'Regular'\n", '}\n', '\n', "# Aplicar corrección (respetando los que ya están bien como 'Low Fat' y 'Regular')\n", "df['Item_Fat_Content'] = df['Item_Fat_Content'].replace(fat_content_map)\n", '\n', 'print("Distribución corregida de Fat Content:")\n', "print(df['Item_Fat_Content'].value_counts())"]}, {'cell_type': 'markdown', 'metadata': {}, 'source': ['### 1.2 Tratamiento de Nulos: Lógica de Negocio\n', '\n', '**Item_Weight:** Un mismo producto (`Item_Identifier`) debe pesar lo mismo en todas las tiendas. Usaremos esto para imputar.\n', '**Item_Visibility:** Una visibilidad de 0.0 es imposible. La trataremos como nulo y la imputaremos con el promedio de visibilidad de ese producto.']}, {'cell_type': 'code', 'execution_count': 3, 'metadata': {}, 'outputs': [{'name': 'stdout', 'output_type': 'stream', 'text': ['Nulos restantes tras imputación lógica:\n', 'Item_Visibility 0\n', 'Item_Weight 0\n', 'dtype: int64\n']}], 'source': ['# 1. Tratamiento de Visibilidad\n', '# Reemplazar 0.0 con NaN para que no afecte el cálculo del promedio\n', "df['Item_Visibility'] = df['Item_Visibility'].replace(0, np.nan)\n", '\n', '# Imputar la visibilidad con el promedio DE ESE PRODUCTO ESPECÍFICO\n', "df['Item_Visibility'] = df.groupby('Item_Identifier')['Item_Visibility'].transform(lambda x: x.fillna(x.mean()))\n", '\n', '# 2. Tratamiento de Peso (Weight)\n', '# Estrategia Primaria: Rellenar con el peso existente del mismo ID\n', "df['Item_Weight'] = df.groupby('Item_Identifier')['Item_Weight'].transform(lambda x: x.fillna(x.mean()))\n", '\n', '# Estrategia de Respaldo (Fallback): Si un producto NO tiene peso en ninguna tienda (todos nulos para ese ID),\n', "# imputamos con la media global de su 'Item_Type'\n", "df['Item_Weight'] = df['Item_Weight'].fillna(df.groupby('Item_Type')['Item_Weight'].transform('mean'))\n", '\n', '# Verificación final de nulos\n', 'print("Nulos restantes tras imputación lógica:")\n', "print(df[['Item_Visibility', 'Item_Weight']].isnull().sum())"]}, {'cell_type': 'markdown', 'metadata': {}, 'source': ['---\n', '## 2. Feature Engineering (Previo a Agregación)\n', '\n', 'Derivaremos características que son inherentes al producto.\n', '\n', '* **Broad_Category:** Las primeras dos letras del ID (FD, DR, NC) nos dicen la categoría macro.\n', "* **Ajuste Semántico:** Si es 'NC' (Non-Consumable), no tiene sentido que tenga 'Low Fat'. Lo cambiaremos a 'Non-Edible'."]}, {'cell_type': 'code', 'execution_count': 4, 'metadata': {}, 'outputs': [{'name': 'stdout', 'output_type': 'stream', 'text': ['Item_Fat_Content Low Fat Non-Edible Regular\n', 'Broad_Category \n', 'Drinks 728 0 71\n', 'Food 3190 0 2935\n', 'Non-Consumable 0 1599 0\n']}], 'source': ['# Crear Broad Category\n', "df['Broad_Category'] = df['Item_Identifier'].apply(lambda x: x[:2])\n", "category_map = {'FD': 'Food', 'NC': 'Non-Consumable', 'DR': 'Drinks'}\n", "df['Broad_Category'] = df['Broad_Category'].map(category_map)\n", '\n', '# Ajuste lógico: Si es Non-Consumable, Fat_Content = Non-Edible\n', "df.loc[df['Broad_Category'] == 'Non-Consumable', 'Item_Fat_Content'] = 'Non-Edible'\n", '\n', "print(pd.crosstab(df['Broad_Category'], df['Item_Fat_Content']))"]}, {'cell_type': 'markdown', 'metadata': {}, 'source': ['---\n', '## 3. Construcción del Dataset Agregado (Nivel Producto)\n', '\n', 'Este es el paso core de la Fase B. Reduciremos la granularidad de Tienda-Producto a solo Producto.\n', '\n', '**Métricas a calcular:**\n', '1. `Total_Sales`: Suma de ventas (¿Qué tanto volumen mueve este producto en total?)\n', '2. `Avg_Sales`: Promedio de ventas por tienda (¿Qué tan bien performa individualmente?)\n', '3. `Store_Count`: Conteo único de tiendas (¿Qué tanta penetración de mercado tiene?)\n', '4. `Avg_MRP`: Precio promedio (Indica su gama: económico vs premium).\n', '5. `Avg_Visibility`: Visibilidad promedio.\n', '6. `Item_Weight`: Promedio (que será igual al valor único).\n', '7. Variables Categóricas: Tomaremos el "primero" ya que son constantes por producto.']}, {'cell_type': 'code', 'execution_count': 5, 'metadata': {}, 'outputs': [{'name': 'stdout', 'output_type': 'stream', 'text': ['Dimensiones del dataset agregado: (1559, 10)\n', 'Debería ser (1559, 10). Resultado: (1559, 10)\n']}, {'data': {'text/html': ['<div>\n', '<style scoped>\n', ' .dataframe tbody tr th:only-of-type {\n', ' vertical-align: middle;\n', ' }\n', '\n', ' .dataframe tbody tr th {\n', ' vertical-align: top;\n', ' }\n', '\n', ' .dataframe thead th {\n', ' text-align: right;\n', ' }\n', '</style>\n', '<table border="1" class="dataframe">\n', ' <thead>\n', ' <tr style="text-align: right;">\n', ' <th></th>\n', ' <th>Item_Identifier</th>\n', ' <th>Total_Sales</th>\n', ' <th>Avg_Sales</th>\n', ' <th>Store_Count</th>\n', ' <th>Avg_MRP</th>\n', ' <th>Avg_Visibility</th>\n', ' <th>Item_Weight</th>\n', ' <th>Item_Fat_Content</th>\n', ' <th>Broad_Category</th>\n', ' <th>Item_Type</th>\n', ' </tr>\n', ' </thead>\n', ' <tbody>\n', ' <tr>\n', ' <th>0</th>\n', ' <td>DRA12</td>\n', ' <td>11061.6012</td>\n', ' <td>1843.600200</td>\n', ' <td>6</td>\n', ' <td>141.865400</td>\n', ' <td>0.047934</td>\n', ' <td>11.600</td>\n', ' <td>Low Fat</td>\n', ' <td>Drinks</td>\n', ' <td>Soft Drinks</td>\n', ' </tr>\n', ' <tr>\n', ' <th>1</th>\n', ' <td>DRA24</td>\n', ' <td>15723.5328</td>\n', ' <td>2246.218971</td>\n', ' <td>7</td>\n', ' <td>164.086800</td>\n', ' <td>0.048062</td>\n', ' <td>19.350</td>\n', ' <td>Regular</td>\n', ' <td>Drinks</td>\n', ' <td>Soft Drinks</td>\n', ' </tr>\n', ' <tr>\n', ' <th>2</th>\n', ' <td>DRA59</td>\n', ' <td>20915.4412</td>\n', ' <td>2614.430150</td>\n', ' <td>8</td>\n', ' <td>185.179900</td>\n', ' <td>0.153963</td>\n', ' <td>8.270</td>\n', ' <td>Regular</td>\n', ' <td>Drinks</td>\n', ' <td>Soft Drinks</td>\n', ' </tr>\n', ' <tr>\n', ' <th>3</th>\n', ' <td>DRB01</td>\n', ' <td>4554.0720</td>\n', ' <td>1518.024000</td>\n', ' <td>3</td>\n', ' <td>189.586333</td>\n', ' <td>0.082126</td>\n', ' <td>7.390</td>\n', ' <td>Low Fat</td>\n', ' <td>Drinks</td>\n', ' <td>Soft Drinks</td>\n', ' </tr>\n', ' <tr>\n', ' <th>4</th>\n', ' <td>DRB13</td>\n', ' <td>12144.1920</td>\n', ' <td>2428.838400</td>\n', ' <td>5</td>\n', ' <td>189.693000</td>\n', ' <td>0.008002</td>\n', ' <td>6.115</td>\n', ' <td>Regular</td>\n', ' <td>Drinks</td>\n', ' <td>Soft Drinks</td>\n', ' </tr>\n', ' </tbody>\n', '</table>\n', '</div>'], 'text/plain': [' Item_Identifier Total_Sales Avg_Sales Store_Count Avg_MRP \\\n', '0 DRA12 11061.6012 1843.600200 6 141.865400 \n', '1 DRA24 15723.5328 2246.218971 7 164.086800 \n', '2 DRA59 20915.4412 2614.430150 8 185.179900 \n', '3 DRB01 4554.0720 1518.024000 3 189.586333 \n', '4 DRB13 12144.1920 2428.838400 5 189.693000 \n', '\n', ' Avg_Visibility Item_Weight Item_Fat_Content Broad_Category Item_Type \n', '0 0.047934 11.600 Low Fat Drinks Soft Drinks \n', '1 0.048062 19.350 Regular Drinks Soft Drinks \n', '2 0.153963 8.270 Regular Drinks Soft Drinks \n', '3 0.082126 7.390 Low Fat Drinks Soft Drinks \n', '4 0.008002 6.115 Regular Drinks Soft Drinks ']}, 'execution_count': 5, 'metadata': {}, 'output_type': 'execute_result'}], 'source': ['# Definir diccionario de agregaciones\n', 'aggs = {\n', " 'Item_Outlet_Sales': ['sum', 'mean'], # Total Volumen y Rendimiento Promedio\n", " 'Outlet_Identifier': 'nunique', # Store Count\n", " 'Item_MRP': 'mean', # Precio Promedio\n", " 'Item_Visibility': 'mean', # Visibilidad Promedio\n", " 'Item_Weight': 'first', # Peso (Constante)\n", " 'Item_Fat_Content': 'first', # Categórica (Constante)\n", " 'Broad_Category': 'first',

# Categórica (Constante)\n", "    'Item_Type': 'first'                      # Categórica (Constante - opcional pero útil)\n", '}\n', '\n', '# Agrupar\n', "df_product = df.groupby('Item_Identifier').agg(aggs).reset_index()\n", '\n', '# Aplanar los nombres de columnas (MultiIndex a Single Index)\n', 'df_product.columns = [\n', "    'Item_Identifier', \n", "    'Total_Sales', \n", "    'Avg_Sales', \n", "    'Store_Count', \n", "    'Avg_MRP', \n", "    'Avg_Visibility', \n", "    'Item_Weight', \n", "    'Item_Fat_Content', \n", "    'Broad_Category', \n", "    'Item_Type'\n", ']\n', '\n', 'print(f"Dimensiones del dataset agregado: {df_product.shape}")\n', 'print(f"Debería ser (1559, 10). Resultado: {df_product.shape}")\n', 'df_product.head()']}, {'cell_type': 'markdown', 'metadata': {}, 'source': ['---\n', '## 4. Preprocesamiento para Clustering (Encoding y Scaling)\n', '\n', 'Generaremos dos datasets:\n', '1.  **Interpretación:** Mantiene los valores originales.\n', '2.  **Modelado:** Numérico y escalado, listo para K-Means.\n', '\n', '### 4.1 Encoding\n', 'Transformaremos `Item_Fat_Content` y `Broad_Category`. No usaremos `Item_Type` para el clustering inicial para evitar la maldición de la dimensionalidad (muchas columnas dummy), usaremos `Broad_Category` como proxy general.']}, {'cell_type': 'code', 'execution_count': 6, 'metadata': {}, 'outputs': [{'name': 'stdout', 'output_type': 'stream', 'text': ["Index(['Total_Sales', 'Avg_Sales', 'Store_Count', 'Avg_MRP', 'Avg_Visibility',\n", "       'Item_Weight', 'Item_Fat_Content_Low Fat',\n", "       'Item_Fat_Content_Non-Edible', 'Item_Fat_Content_Regular',\n", "       'Broad_Category_Drinks', 'Broad_Category_Food',\n", "       'Broad_Category_Non-Consumable'],\n", "      dtype='object')\n"]}], 'source': ['# Copia para modelado\n', "df_modeling = df_product.copy().set_index('Item_Identifier')\n", '\n', '# 1. Drop de columnas que no usaremos en el modelo numérico estricto\n', '# Item_Type es muy granular, Broad_Category captura la esencia mejor para clustering de alto nivel\n', "df_modeling = df_modeling.drop(columns=['Item_Type'])\n", '\n', '# 2. One Hot Encoding para categoricas\n', "df_modeling = pd.get_dummies(df_modeling, columns=['Item_Fat_Content', 'Broad_Category'], drop_first=False)\n", '\n', '# Visualizar columnas resultantes\n', 'print(df_modeling.columns)']}, {'cell_type': 'markdown', 'metadata': {}, 'source': ['### 4.2 Scaling\n', 'Los algoritmos de clustering basados en distancia (como K-Means) son sensibles a la magnitud. `Total_Sales` (miles) dominaría sobre `Item_Visibility` (0.01). Usaremos `StandardScaler`.']}, {'cell_type': 'code', 'execution_count': 7, 'metadata': {}, 'outputs': [{'data': {'text/html': ['<div>\n', '<style scoped>\n', '    .dataframe tbody tr th:only-of-type {\n', '        vertical-align: middle;\n', '    }\n', '\n', '    .dataframe tbody tr th {\n', '        vertical-align: top;\n', '    }\n', '\n', '    .dataframe thead th {\n', '        text-align: right;\n', '    }\n', '</style>\n', '<table border="1" class="dataframe">\n', '  <thead>\n', '    <tr style="text-align: right;">\n', '      <th></th>\n', '      <th>Total_Sales</th>\n', '      <th>Avg_Sales</th>\n', '      <th>Store_Count</th>\n', '      <th>Avg_MRP</th>\n', '      <th>Avg_Visibility</th>\n', '      <th>Item_Weight</th>\n', '      <th>Item_Fat_Content_Low Fat</th>\n', '      <th>Item_Fat_Content_Non-Edible</th>\n', '      <th>Item_Fat_Content_Regular</th>\n', '      <th>Broad_Category_Drinks</th>\n', '      <th>Broad_Category_Food</th>\n', '      <th>Broad_Category_Non-Consumable</th>\n', '    </tr>\n', '    <tr>\n', '      <th>Item_Identifier</th>\n', '      <th></th>\n', '      <th></th>\n', '      <th></th>\n', '      <th></th>\n', '      <th></th>\n', '      <th></th>\n', '      <th></th>\n', '      <th></th>\n', '      <th></th>\n', '      <th></th>\n', '      <th></th>\n', '      <th></th>\n', '    </tr>\n', '  </thead>\n', '  <tbody>\n', '    <tr>\n', '      <th>DRA12</th>\n', '      <td>-0.122721</td>\n', '      <td>-0.308001</td>\n', '      <td>0.348838</td>\n', '      <td>0.013769</td>\n', '      <td>-0.472216</td>\n', '      <td>-0.260236</td>\n', '      <td>1.089282</td>\n', '      <td>-0.4831</td>\n', '      <td>-0.739342</td>\n', '      <td>3.122775</td>\n', '      <td>-1.594736</td>\n', '      <td>-0.4831</td>\n', '    </tr>\n', '

<tr>\n', '      <th>DRA24</th>\n', '      <td>0.539887</td>\n', '      <td>0.047767</td>\n', '
<td>1.003278</td>\n', '      <td>0.371701</td>\n', '      <td>-0.469498</td>\n', '
<td>1.408344</td>\n', '      <td>-0.918036</td>\n', '      <td>-0.4831</td>\n', '
<td>1.352554</td>\n', '      <td>3.122775</td>\n', '      <td>-1.594736</td>\n', '
<td>-0.4831</td>\n', '    </tr>\n', '    <tr>\n', '      <th>DRA59</th>\n', '
<td>1.277820</td>\n', '      <td>0.373131</td>\n', '      <td>1.657717</td>\n', '
<td>0.711459</td>\n', '      <td>1.779100</td>\n', '      <td>-0.977187</td>\n', '
<td>-0.918036</td>\n', '      <td>-0.4831</td>\n', '      <td>1.352554</td>\n', '
<td>3.122775</td>\n', '      <td>-1.594736</td>\n', '      <td>-0.4831</td>\n', '    </tr>\n', '
</tbody>\n', '</table>\n', '</div>'], 'text/plain': ['                Total_Sales  Avg_Sales
Store_Count   Avg_MRP  \\\n', 'Item_Identifier                                                \n', 'DRA12
-0.122721  -0.308001    0.348838  0.013769   \n', 'DRA24             0.539887   0.047767
1.003278  0.371701   \n', 'DRA59             1.277820   0.373131    1.657717  0.711459   \n',
'\n', '                Avg_Visibility  Item_Weight  Item_Fat_Content_Low Fat  \\\n', 'Item_Identifier
\n', 'DRA12               -0.472216    -0.260236                 1.089282   \n', 'DRA24
-0.469498     1.408344                -0.918036   \n', 'DRA59                1.779100    -0.977187
-0.918036   \n', '\n', '                Item_Fat_Content_Non-Edible  Item_Fat_Content_Regular
\\\n', 'Item_Identifier                                                        \n', 'DRA12
-0.4831                -0.739342   \n', 'DRA24                           -0.4831                 1.352554
\n', 'DRA59                           -0.4831                 1.352554   \n', '\n', '
Broad_Category_Drinks  Broad_Category_Food  \\\n', 'Item_Identifier
\n', 'DRA12                      3.122775            -1.594736   \n', 'DRA24                      3.122775
-1.594736   \n', 'DRA59                      3.122775            -1.594736   \n', '\n', '
Broad_Category_Non-Consumable  \n', 'Item_Identifier                                  \n', 'DRA12
-0.4831  \n', 'DRA24                            -0.4831  \n', 'DRA59                            -0.4831
']}, 'execution_count': 7, 'metadata': {}, 'output_type': 'execute_result'}], 'source': ['scaler =
StandardScaler()\n', '\n', '# Escalamos todo el dataframe de modelado\n',
'df_modeling_scaled = pd.DataFrame(scaler.fit_transform(df_modeling), \n', '
columns=df_modeling.columns, \n', '                                  index=df_modeling.index)\n', '\n',
'df_modeling_scaled.head(3)']}, {'cell_type': 'markdown', 'metadata': {}, 'source': ['---\n', '## 5.
Exportación y Resumen\n', '\n', 'Guardaremos los archivos en `../Data/Processed/`.']},
{'cell_type': 'code', 'execution_count': 8, 'metadata': {}, 'outputs': [{'name': 'stdout',
'output_type': 'stream', 'text': ['Archivos guardados exitosamente.\n', '1.
product_level_interpretation.csv (Para análisis y dashboard)\n', '2.
product_level_modeling.csv (Para algoritmos de clustering)\n']}], 'source': ['# Guardar
Dataset para Interpretación (Valores Reales)\n',
"df_product.to_csv('../Data/Processed/product_level_interpretation.csv', index=False)\n", '\n',
'# Guardar Dataset para Modelado (Escalado y Encodeado)\n',
"df_modeling_scaled.to_csv('../Data/Processed/product_level_modeling.csv', index=True)\n",
'\n', 'print("Archivos guardados exitosamente.")\n', 'print("1. product_level_interpretation.csv
(Para análisis y dashboard)")\n', 'print("2. product_level_modeling.csv (Para algoritmos de
clustering)")']}, {'cell_type': 'markdown', 'metadata': {}, 'source': ['### Resumen de
Transformaciones Aplicadas\n', '\n', '1.  **Limpieza:**\n', '    * Unificación de etiquetas
`Item_Fat_Content`.\n', '    * Imputación de `Item_Weight` basada en ID (lógica de producto
único).\n', '    * Imputación de `Item_Visibility` basada en promedio del producto (corrigiendo
ceros).\n', '\n', '2.  **Ingeniería:**\n', '    * Creación de `Broad_Category` (Food, Drink,
Non-Consumable).\n', "    * Corrección semántica: Non-Consumable ahora es 'Non-Edible'
en contenido de grasa.\n", '\n', '3.  **Agregación:**\n', '    * Reducción de 8523 filas a **1559

productos únicos**.\n', '    * Nuevas métricas generadas: `Store_Count`, `Total_Sales`,
`Avg_Visibility`, etc.\n', '\n', '4.  **Preprocesamiento:**\n', '    * One-Hot Encoding aplicado a
categorías.\n', '    * Standard Scaling aplicado para normalizar rangos de ventas vs
visibilidad.']}], 'metadata': {'kernelspec': {'display_name': 'Python 3', 'language': 'python',
'name': 'python3'}, 'language_info': {'codemirror_mode': {'name': 'ipython', 'version': 3},
'file_extension': '.py', 'mimetype': 'text/x-python', 'name': 'python', 'nbconvert_exporter':
'python', 'pygments_lexer': 'ipython3', 'version': '3.12.8'}}, 'nbformat': 4, 'nbformat_minor': 4}
```

``` fase_c.ipynb
{'cells': [{'cell_type': 'markdown', 'metadata': {}, 'source': ['# Fase C: Clustering de
Productos\n', '\n', '## Objetivos de la Notebook\n', '1.  **Evaluación de Algoritmos:**
Comparar K-Means con métricas de cohesión.\n', '2.  **Determinación de K Óptimo:**
Analizar el rango de 2 a 30 clusters. **Punto Crítico:** Resolver la discrepancia entre la
visualización (que sugiere 3 grupos) y las métricas matemáticas (que sugieren 4).\n', '3.
**Análisis de Granularidad:** Justificar por qué seleccionamos K=4 sobre K=6 a pesar de la
búsqueda de detalle.\n', '4.  **Perfilamiento de Negocio:** Traducir los clusters matemáticos
a etiquetas de negocio ("Premium", "Mass Market") para la Fase D.']}, {'cell_type': 'code',
'execution_count': 1, 'metadata': {}, 'outputs': [{'name': 'stdout', 'output_type': 'stream', 'text':
['✅ Datos cargados correctamente.\n']}], 'source': ['import pandas as pd\n', 'import numpy as
np\n', 'import matplotlib.pyplot as plt\n', 'import seaborn as sns\n', 'from sklearn.cluster import
KMeans\n', 'from sklearn.metrics import silhouette_score\n', 'from sklearn.decomposition
import PCA\n', '\n', '# Configuración de visualización\n', 'sns.set(style="whitegrid")\n',
"plt.rcParams['figure.figsize'] = (12, 6)\n", '\n', '# Carga de datos procesados en Fase B\n',
'try:\n', '    # Dataset numérico escalado (Para los algoritmos, index=Item_Identifier)\n', "
df_model = pd.read_csv('../Data/Processed/product_level_modeling.csv',
index_col='Item_Identifier')\n", '    # Dataset con valores reales (Para la interpretación
humana)\n', "    df_interpret =
pd.read_csv('../Data/Processed/product_level_interpretation.csv')\n", '    print("✅ Datos
cargados correctamente.")\n', 'except FileNotFoundError:\n', '    print("❌ Error: Verifica la
ruta de los archivos.")']}, {'cell_type': 'markdown', 'metadata': {}, 'source': ['---\n', '## 1.
Determinación de K (Inercia vs Silhouette)\n', '\n', 'Ejecutaremos el algoritmo en un rango de
**2 a 30**. Buscamos un K que tenga un Silhouette Score alto (buena separación) y una
Inercia baja (buena compactación).\n', '\n', '**Hipótesis de entrada:** Visualmente el PCA
suele mostrar 3 grandes grupos (Categorías), pero buscamos matices de precio/ventas
dentro de esos grupos.']}, {'cell_type': 'code', 'execution_count': 2, 'metadata': {}, 'outputs':
[{'name': 'stdout', 'output_type': 'stream', 'text': ['🔄 Calculando métricas para K=2 a
K=30...\n', 'K=2 | Silhouette=0.3156 | Inercia=14103.2\n', 'K=3 | Silhouette=0.2776 |
Inercia=11347.1\n', 'K=4 | Silhouette=0.3109 | Inercia=9413.4\n', 'K=5 | Silhouette=0.2927 |
Inercia=8044.4\n', 'K=6 | Silhouette=0.2927 | Inercia=7234.0\n', 'K=10 | Silhouette=0.2144 |
Inercia=5923.2\n', 'K=15 | Silhouette=0.2116 | Inercia=4940.5\n']}, {'data': {'text/plain':
['<Figure size 1400x600 with 2 Axes>']}, 'metadata': {}, 'output_type': 'display_data'}],
'source': ['range_n_clusters = range(2, 31)\n', 'inertias = []\n', 'silhouette_scores = []\n', '\n',
'print("🔄 Calculando métricas para K=2 a K=30...")\n', '\n', 'for k in range_n_clusters:\n', '
kmeans = KMeans(n_clusters=k, random_state=42, n_init=10)\n', '    cluster_labels =
kmeans.fit_predict(df_model)\n', '    inertias.append(kmeans.inertia_)\n', '
silhouette_scores.append(silhouette_score(df_model, cluster_labels))\n', '    \n', '    if k in [2,
3, 4, 5, 6, 10, 15]: # Prints clave\n', '        print(f"K={k} | Silhouette={silhouette_scores[-1]:.4f}

| Inercia={kmeans.inertia_:.1f}")\n', '\n', '# Gráfica Dual\n', 'fig, ax1 = plt.subplots(figsize=(14, 6))\n', "color = 'tab:blue'\n", "ax1.set_xlabel('Número de Clusters (k)')\n", "ax1.set_ylabel('Inercia', color=color)\n", "ax1.plot(range_n_clusters, inertias, marker='o', color=color)\n", "ax1.tick_params(axis='y', labelcolor=color)\n", '\n', 'ax2 = ax1.twinx()\n', "color = 'tab:red'\n", "ax2.set_ylabel('Silhouette Score', color=color)\n", "ax2.plot(range_n_clusters, silhouette_scores, marker='x', linestyle='--', color=color)\n", "ax2.tick_params(axis='y', labelcolor=color)\n", '\n', "plt.title('Decisión de K: Balanceando Cohesión y Complejidad')\n", 'plt.xticks(range(2, 31, 2))\n', 'plt.show()']}, {'cell_type': 'markdown', 'metadata': {}, 'source': ['### Análisis de Decisión: ¿Por qué K=4 y no K=6?\n', '\n', '**Los Datos:**\n', '* **K=4:** Silhouette **0.3109** (Pico local alto).\n', '* **K=6:** Silhouette **0.2927** (Caída significativa).\n', '\n', '**Interpretación:**\n', 'Aunque K=6 ofrece más granularidad, la caída en el Silhouette indica que los nuevos grupos creados no están bien definidos; se están solapando. En negocio, crear categorías difusas ("Ruido") es peligroso porque lleva a estrategias confusas.\n', '\n', '**Decisión Final:** Nos quedamos con **K=4**. Es el punto óptimo donde maximizamos la separación matemática antes de empezar a degradar la calidad de los grupos por exceso de fragmentación.']}, {'cell_type': 'markdown', 'metadata': {}, 'source': ['---\n', '## 2. Entrenamiento y la "Ilusión Visual" del PCA\n', '\n', 'Entrenaremos con K=4. Al graficar en PCA, veremos algo interesante: **Visualmente hay 3 bloques, pero el modelo detecta 4.**']}, {'cell_type': 'code', 'execution_count': 3, 'metadata': {}, 'outputs': [{'data': {'text/plain': ['<Figure size 1000x800 with 1 Axes>']}, 'metadata': {}, 'output_type': 'display_data'}], 'source': ['# Entrenar Modelo Final\n', 'kmeans_final = KMeans(n_clusters=4, random_state=42, n_init=10)\n', 'clusters = kmeans_final.fit_predict(df_model)\n', "df_model['Cluster'] = clusters\n", '\n', '# PCA para visualización\n', 'pca = PCA(n_components=2)\n', "pca_data = pca.fit_transform(df_model.drop('Cluster', axis=1))\n", "df_pca = pd.DataFrame(pca_data, columns=['PC1', 'PC2'])\n", "df_pca['Cluster'] = clusters\n", '\n', '# Visualización\n', 'plt.figure(figsize=(10, 8))\n', "sns.scatterplot(x='PC1', y='PC2', hue='Cluster', data=df_pca, palette='viridis', s=60, alpha=0.8)\n", "plt.title('Distribución de Clusters (K=4) en PCA')\n", "plt.xlabel('Componente Principal 1')\n", "plt.ylabel('Componente Principal 2')\n", "plt.legend(title='Cluster ID')\n", 'plt.show()']}, {'cell_type': 'markdown', 'metadata': {}, 'source': ['### Interpretación de la Discrepancia Visual\n', '\n', 'Si observamos la gráfica:\n', '1. **Bloque Derecho (Verde):** Muy separado. Seguramente `Non-Consumable`.\n', '2. **Bloque Central (Amarillo):** Muy separado. Seguramente `Drinks`.\n', '3. **Bloque Izquierdo (Morado/Azul):** Aquí está la clave. El PCA (2D) los muestra juntos, pero K-Means los separó.\n', '\n', 'Esto significa que **dentro de la categoría dominante (probablemente Comida)**, existen dos sub-comportamientos muy distintos (ej. Comida Barata vs Comida Cara) que están separados en las dimensiones que el PCA "aplanó". **Esta es la ganancia de usar K=4 sobre K=3.**']}, {'cell_type': 'markdown', 'metadata': {}, 'source': ['---\n', '## 3. Perfilamiento de Negocio\n', '\n', 'Validaremos nuestra hipótesis revisando los promedios reales.']}, {'cell_type': 'code', 'execution_count': 4, 'metadata': {}, 'outputs': [{'name': 'stdout', 'output_type': 'stream', 'text': ['Perfil de los 4 Clusters:\n']}, {'data': {'text/html': ['<style type="text/css">\n', '#T_f3888_row0_col0, #T_f3888_row0_col1 {\n', ' background-color: #f7fcf5;\n', ' color: #000000;\n', '}\n', '#T_f3888_row1_col0 {\n', ' background-color: #005723;\n', ' color: #f1f1f1;\n', '}\n', '#T_f3888_row1_col1 {\n', ' background-color: #004e1f;\n', ' color: #f1f1f1;\n', '}\n', '#T_f3888_row2_col0, #T_f3888_row3_col1 {\n', ' background-color: #00441b;\n', ' color: #f1f1f1;\n', '}\n', '#T_f3888_row2_col1 {\n', ' background-color: #077331;\n', ' color: #f1f1f1;\n', '}\n', '#T_f3888_row3_col0 {\n', ' background-color: #6abf71;\n', ' color: #000000;\n', '}\n',

'</style>\n', '<table id="T_f3888">\n', '  <thead>\n', '    <tr>\n', '      <th class="blank level0" > </th>\n', '      <th id="T_f3888_level0_col0" class="col_heading level0 col0" >Total_Sales</th>\n', '      <th id="T_f3888_level0_col1" class="col_heading level0 col1" >Avg_MRP</th>\n', '      <th id="T_f3888_level0_col2" class="col_heading level0 col2" >Store_Count</th>\n', '      <th id="T_f3888_level0_col3" class="col_heading level0 col3" >Broad_Category</th>\n', '      <th id="T_f3888_level0_col4" class="col_heading level0 col4" >Item_Identifier</th>\n', '    </tr>\n', '    <tr>\n', '      <th class="index_name level0" >Cluster</th>\n', '      <th class="blank col0" > </th>\n', '      <th class="blank col1" > </th>\n', '      <th class="blank col2" > </th>\n', '      <th class="blank col3" > </th>\n', '      <th class="blank col4" > </th>\n', '    </tr>\n', '  </thead>\n', '  <tbody>\n', '    <tr>\n', '      <th id="T_f3888_level0_row0" class="row_heading level0 row0" >3</th>\n', '      <td id="T_f3888_row0_col0" class="data row0 col0" >11005.995421</td>\n', '      <td id="T_f3888_row0_col1" class="data row0 col1" >132.370867</td>\n', '      <td id="T_f3888_row0_col2" class="data row0 col2" >5.510345</td>\n', '      <td id="T_f3888_row0_col3" class="data row0 col3" >Drinks</td>\n', '      <td id="T_f3888_row0_col4" class="data row0 col4" >145</td>\n', '    </tr>\n', '    <tr>\n', '      <th id="T_f3888_level0_row1" class="row_heading level0 row1" >0</th>\n', '      <td id="T_f3888_row1_col0" class="data row1 col0" >12092.504109</td>\n', '      <td id="T_f3888_row1_col1" class="data row1 col1" >142.266641</td>\n', '      <td id="T_f3888_row1_col2" class="data row1 col2" >5.500000</td>\n', '      <td id="T_f3888_row1_col3" class="data row1 col3" >Food</td>\n', '      <td id="T_f3888_row1_col4" class="data row1 col4" >580</td>\n', '    </tr>\n', '    <tr>\n', '      <th id="T_f3888_level0_row2" class="row_heading level0 row2" >1</th>\n', '      <td id="T_f3888_row2_col0" class="data row2 col0" >12162.137719</td>\n', '      <td id="T_f3888_row2_col1" class="data row2 col1" >141.100732</td>\n', '      <td id="T_f3888_row2_col2" class="data row2 col2" >5.445269</td>\n', '      <td id="T_f3888_row2_col3" class="data row2 col3" >Food</td>\n', '      <td id="T_f3888_row2_col4" class="data row2 col4" >539</td>\n', '    </tr>\n', '    <tr>\n', '      <th id="T_f3888_level0_row3" class="row_heading level0 row3" >2</th>\n', '      <td id="T_f3888_row3_col0" class="data row3 col0" >11614.276138</td>\n', '      <td id="T_f3888_row3_col1" class="data row3 col1" >142.623000</td>\n', '      <td id="T_f3888_row3_col2" class="data row3 col2" >5.420339</td>\n', '      <td id="T_f3888_row3_col3" class="data row3 col3" >Non-Consumable</td>\n', '      <td id="T_f3888_row3_col4" class="data row3 col4" >295</td>\n', '    </tr>\n', '  </tbody>\n', '</table>\n'], 'text/plain': ['<pandas.io.formats.style.Styler at 0x7f3332d523f0>']}, 'metadata': {}, 'output_type': 'display_data'}], 'source': ["df_interpret['Cluster'] = clusters\n", '\n', '# Perfilamiento Numérico\n', "profile = df_interpret.groupby('Cluster').agg({\n", "    'Total_Sales': 'mean',\n", "    'Avg_MRP': 'mean',\n", "    'Store_Count': 'mean',\n", "    'Broad_Category': lambda x: x.mode()[0],\n", "    'Item_Identifier': 'count'\n", "}).sort_values('Broad_Category')\n", '\n', 'print("Perfil de los 4 Clusters:")\n', "display(profile.style.background_gradient(cmap='Greens', subset=['Total_Sales', 'Avg_MRP']))"]}, {'cell_type': 'code', 'execution_count': 5, 'metadata': {}, 'outputs': [{'name': 'stdout', 'output_type': 'stream', 'text': ['Validación de Etiquetas:\n', '     Item_Identifier Broad_Category    Avg_MRP  Cluster  \\\n', '1263           FDZ60           Food  107.819600        0   \n', '539            FDJ56           Food  100.250000        0   \n', '477            FDI14           Food  140.363886        0   \n', '114            DRL11         Drinks  158.354600        3   \n', '940            FDT28           Food  150.904133        0   \n', '\n', '                   Cluster_Label  \n', '1263  Food: Low Viz / Economy  \n', '539       Food: Low Viz / Economy  \n', '477   Food: Mass

Market / High Rev  \n', '114                    Drinks  \n', '940   Food: Mass Market / High Rev  \n']}], 'source': ['# Etiquetado Automático basado en reglas (Ajustar lógica según el output anterior)\n', 'def label_cluster(row):\n', '    # Ejemplo de lógica basada en tus resultados previos:\n', '    # Cluster 0 y 1 son Food. Uno tiene más ventas/MRP que el otro?\n', '    # Cluster 2 es Non-Consumable\n', '    # Cluster 3 es Drinks\n', '    \n', "    cat = row['Broad_Category']\n", "    if cat == 'Non-Consumable':\n", "        return 'Non-Edible Goods'\n", "    elif cat == 'Drinks':\n", "        return 'Drinks'\n", "    elif cat == 'Food':\n", '        # Diferenciación dentro del bloque "Food" (La separación oculta en PCA)\n', "        if row['Avg_MRP'] > 140: # Umbral hipotético, revisar tabla\n", "            return 'Food: Mass Market / High Rev'\n", '        else:\n', "            return 'Food: Low Viz / Economy'\n", "    return 'Other'\n", '\n', '# Aplicar etiquetas\n', "df_interpret['Cluster_Label'] = df_interpret.apply(label_cluster, axis=1)\n", '\n', 'print("Validación de Etiquetas:")\n', "print(df_interpret[['Item_Identifier', 'Broad_Category', 'Avg_MRP', 'Cluster', 'Cluster_Label']].sample(5))"]}, {'cell_type': 'markdown', 'metadata': {}, 'source': ['---\n', '## 4. Exportación para Fase D\n', '\n', 'Este archivo será crucial para entender el "ADN" de cada tienda en la siguiente fase.']}, {'cell_type': 'code', 'execution_count': 6, 'metadata': {}, 'outputs': [{'name': 'stdout', 'output_type': 'stream', 'text': ['✅ Archivo exportado: ../Data/Processed/product_level_with_clusters.csv\n']}], 'source': ["output_path = '../Data/Processed/product_level_with_clusters.csv'\n", 'df_interpret.to_csv(output_path, index=False)\n', 'print(f"✅ Archivo exportado: {output_path}")']}], 'metadata': {'kernelspec': {'display_name': 'Python 3', 'language': 'python', 'name': 'python3'}, 'language_info': {'codemirror_mode': {'name': 'ipython', 'version': 3}, 'file_extension': '.py', 'mimetype': 'text/x-python', 'name': 'python', 'nbconvert_exporter': 'python', 'pygments_lexer': 'ipython3', 'version': '3.12.8'}}, 'nbformat': 4, 'nbformat_minor': 4}
```

``` fase_d.ipynb
{'cells': [{'cell_type': 'markdown', 'metadata': {}, 'source': ['# Fase D: Análisis Estratégico Integral (Tienda + Entorno + Producto)\n', '\n', '## Resumen Ejecutivo del Proyecto hasta el momento\n', '1.  **Fase A (EDA):** Entendimos la distribución de datos y detectamos nulos.\n', '2.  **Fase B (Feature Eng):** Limpiamos y creamos un dataset a nivel producto único.\n', '3.  **Fase C (Clustering):** Agrupamos productos en 4 clusters clave (Drinks, Food Economy, Food Premium, Non-Edible).\n', '\n', '## Objetivos de esta Notebook (Fase D - Iteración Final)\n', 'Esta notebook consolida todo el análisis de negocio para alimentar el Dashboard de la Fase E. Abordaremos 3 dimensiones:\n', '\n', '1.  **Dimensión Tienda (Store DNA):** \n', '    * Validar que el Mix de productos es homogéneo (Estandarización).\n', '    * Confirmar que la diferenciación viene por **Escala** (Volumen de ventas) y **Eficiencia**.\n', '2.  **Dimensión Estratégica (Precio):**\n', '    * Analizar la sensibilidad al precio. (Validamos previamente que Premium vende más en promedio, incluso en Grocery Stores).\n', '3.  **Dimensión Entorno y Operación (Nuevos Insights):**\n', '    * **Evolución:** Crecimiento de tiendas en el tiempo.\n', '    * **Geografía:** Distribución de formatos por tipo de ciudad.\n', '    * **Visibilidad:** Impacto real de la exhibición en las ventas.\n', '\n', '## Entregable\n', 'Un archivo JSON jerárquico (`store_hierarchy_final.json`) listo para D3.js que contiene toda esta inteligencia de negocio.']}, {'cell_type': 'code', 'execution_count': 1, 'metadata': {}, 'outputs': [{'name': 'stdout', 'output_type': 'stream', 'text': ['✅ Entorno listo. Iniciando análisis integral.\n']}], 'source': ['import pandas as pd\n', 'import numpy as np\n', 'import matplotlib.pyplot as plt\n', 'import seaborn as sns\n', 'import json\n', '\n', '# Configuración de visualización\n', 'sns.set(style="whitegrid")\n', "plt.rcParams['figure.figsize']

= (14, 7)\n", "pd.set_option('display.float_format', lambda x: '%.2f' % x)\n", '\n', '# Rutas de archivos\n', "RAW_DATA_PATH = '../Data/Raw/train.csv'\n", "CLUSTERS_PATH = '../Data/Processed/product_level_with_clusters.csv'\n", '\n', 'print("✅ Entorno listo. Iniciando análisis integral.")']}, {'cell_type': 'markdown', 'metadata': {}, 'source': ['---\n', '## 1. Integración de Datos y Segmentación de Precios\n', '\n', 'Unimos las transacciones con los clusters y creamos la segmentación de precios (Economy/Standard/Premium) que resultó ser un hallazgo clave.']}, {'cell_type': 'code', 'execution_count': 2, 'metadata': {}, 'outputs': [{'name': 'stdout', 'output_type': 'stream', 'text': ['Dataset Integrado: (8523, 16)\n']}, {'data': {'text/html': ['<div>\n', '<style scoped>\n', ' .dataframe tbody tr th:only-of-type {\n', ' vertical-align: middle;\n', ' }\n', '\n', ' .dataframe tbody tr th {\n', ' vertical-align: top;\n', ' }\n', '\n', ' .dataframe thead th {\n', ' text-align: right;\n', ' }\n', '</style>\n', '<table border="1" class="dataframe">\n', ' <thead>\n', ' <tr style="text-align: right;">\n', ' <th></th>\n', ' <th>Item_Identifier</th>\n', ' <th>Item_Weight</th>\n', ' <th>Item_Fat_Content</th>\n', ' <th>Item_Visibility</th>\n', ' <th>Item_Type</th>\n', ' <th>Item_MRP</th>\n', ' <th>Outlet_Identifier</th>\n', ' <th>Outlet_Establishment_Year</th>\n', ' <th>Outlet_Size</th>\n', ' <th>Outlet_Location_Type</th>\n', ' <th>Outlet_Type</th>\n', ' <th>Item_Outlet_Sales</th>\n', ' <th>Cluster_Label</th>\n', ' <th>Broad_Category</th>\n', ' <th>Avg_Visibility</th>\n', ' <th>Price_Tier</th>\n', ' </tr>\n', ' </thead>\n', ' <tbody>\n', ' <tr>\n', ' <th>0</th>\n', ' <td>FDA15</td>\n', ' <td>9.30</td>\n', ' <td>Low Fat</td>\n', ' <td>0.02</td>\n', ' <td>Dairy</td>\n', ' <td>249.81</td>\n', ' <td>OUT049</td>\n', ' <td>1999</td>\n', ' <td>Medium</td>\n', ' <td>Tier 1</td>\n', ' <td>Supermarket Type1</td>\n', ' <td>3735.14</td>\n', ' <td>Food: Mass Market / High Rev</td>\n', ' <td>Food</td>\n', ' <td>0.02</td>\n', ' <td>Premium</td>\n', ' </tr>\n', ' <tr>\n', ' <th>1</th>\n', ' <td>DRC01</td>\n', ' <td>5.92</td>\n', ' <td>Regular</td>\n', ' <td>0.02</td>\n', ' <td>Soft Drinks</td>\n', ' <td>48.27</td>\n', ' <td>OUT018</td>\n', ' <td>2009</td>\n', ' <td>Medium</td>\n', ' <td>Tier 3</td>\n', ' <td>Supermarket Type2</td>\n', ' <td>443.42</td>\n', ' <td>Drinks</td>\n', ' <td>Drinks</td>\n', ' <td>0.02</td>\n', ' <td>Economy</td>\n', ' </tr>\n', ' </tbody>\n', '</table>\n', '</div>'], 'text/plain': ['  Item_Identifier  Item_Weight Item_Fat_Content  Item_Visibility    Item_Type \\\n', '0           FDA15         9.30          Low Fat             0.02        Dairy   \n', '1           DRC01         5.92          Regular             0.02  Soft Drinks   \n', '\n', '   Item_MRP Outlet_Identifier  Outlet_Establishment_Year Outlet_Size  \\\n', '0    249.81            OUT049                       1999      Medium   \n', '1     48.27            OUT018                       2009      Medium   \n', '\n', '  Outlet_Location_Type        Outlet_Type  Item_Outlet_Sales  \\\n', '0               Tier 1  Supermarket Type1            3735.14   \n', '1               Tier 3  Supermarket Type2             443.42   \n', '\n', '                  Cluster_Label Broad_Category  Avg_Visibility Price_Tier  \n', '0  Food: Mass Market / High Rev           Food            0.02    Premium  \n', '1                        Drinks         Drinks            0.02    Economy  ']}, 'execution_count': 2, 'metadata': {}, 'output_type': 'execute_result'}], 'source': ['# 1. Carga\n', 'df_trans = pd.read_csv(RAW_DATA_PATH)\n', 'df_clusters = pd.read_csv(CLUSTERS_PATH)\n', '\n', '# 2. Merge\n', "cols_cluster = ['Item_Identifier', 'Cluster_Label', 'Broad_Category', 'Avg_Visibility'] \n", '# Nota: Traemos Avg_Visibility del cluster para comparar vs la real de la tienda\n', "df_merged = df_trans.merge(df_clusters[cols_cluster], on='Item_Identifier', how='left')\n", '\n', '# 3. Price Tiers (Feature Engineering)\n', "quartiles = df_merged['Item_MRP'].quantile([0.33, 0.66]).values\n", 'def classify_price(mrp):\n', "    if mrp < quartiles[0]: return 'Economy'\n", "    elif mrp < quartiles[1]: return 'Standard'\n", "    else: return 'Premium'\n", '\n', "df_merged['Price_Tier'] =

df_merged['Item_MRP'].apply(classify_price)\n", '\n', 'print(f"Dataset Integrado: {df_merged.shape}")\n', 'df_merged.head(2)']}, {'cell_type': 'markdown', 'metadata': {}, 'source': ['---\n', '## 2. Dimensión Tienda: DNA, Escala y Eficiencia\n', '\n', 'Replicamos los gráficos que confirmaron que **"Todos venden lo mismo (Mix), pero a escalas muy diferentes"**.']}, {'cell_type': 'code', 'execution_count': 3, 'metadata': {}, 'outputs': [{'data': {'text/plain': ['<Figure size 1800x600 with 2 Axes>']}, 'metadata': {}, 'output_type': 'display_data'}], 'source': ['# Preparación de datos agregados\n', "store_sales = df_merged.groupby(['Outlet_Identifier', 'Cluster_Label'])['Item_Outlet_Sales'].sum().reset_index()\n", "df_absolute = store_sales.pivot(index='Outlet_Identifier', columns='Cluster_Label', values='Item_Outlet_Sales').fillna(0)\n", "df_absolute['Total_Sales'] = df_absolute.sum(axis=1)\n", '\n', '# Datos Relativos (%)\n', "cluster_cols = [c for c in df_absolute.columns if c != 'Total_Sales']\n", "df_relative = df_absolute[cluster_cols].div(df_absolute['Total_Sales'], axis=0) * 100\n", '\n', '# Metadata\n', "store_meta = df_merged[['Outlet_Identifier', 'Outlet_Type', 'Outlet_Size', 'Outlet_Location_Type', 'Outlet_Establishment_Year']].drop_duplicates().set_index('Outlet_Identifier')\n", 'df_dna = df_absolute.join(store_meta)\n', '\n', '# Visualización Dual (Mix vs Escala)\n', "sum_mix_absolute = df_absolute.join(store_meta['Outlet_Type']).groupby('Outlet_Type').sum()\n", "avg_mix_relative = df_relative.join(store_meta['Outlet_Type']).groupby('Outlet_Type').mean()\n", '\n', 'fig, axes = plt.subplots(1, 2, figsize=(18, 6))\n', "avg_mix_relative.plot(kind='bar', stacked=True, ax=axes[0], colormap='viridis')\n", "axes[0].set_title('A. Mix Porcentual (Homogéneo)')\n", "axes[0].set_ylabel('% Ventas')\n", 'axes[0].get_legend().remove()\n', '\n', "sum_mix_absolute.drop('Total_Sales', axis=1).plot(kind='bar', stacked=True, ax=axes[1], colormap='viridis')\n", "axes[1].set_title('B. Volumen Absoluto (Escala Dispar)')\n", "axes[1].set_ylabel('Ventas Totales ($)')\n", 'plt.legend(bbox_to_anchor=(1,1))\n', 'plt.tight_layout()\n', 'plt.show()']}, {'cell_type': 'markdown', 'metadata': {}, 'source': ['---\n', '## 3. Dimensión Estratégica: El Poder de lo Premium\n', '\n', 'Validamos tu hallazgo visual: **Los productos Premium generan más revenue promedio por unidad en TODOS los formatos**, aunque la Grocery Store tenga un volumen total bajo.']}, {'cell_type': 'code', 'execution_count': 4, 'metadata': {}, 'outputs': [{'data': {'text/plain': ['<Figure size 1000x600 with 1 Axes>']}, 'metadata': {}, 'output_type': 'display_data'}], 'source': ["price_sensitivity = df_merged.groupby(['Outlet_Type', 'Price_Tier'])['Item_Outlet_Sales'].mean().reset_index()\n", "order = ['Economy', 'Standard', 'Premium']\n", '\n', 'plt.figure(figsize=(10, 6))\n', "sns.barplot(x='Outlet_Type', y='Item_Outlet_Sales', hue='Price_Tier', hue_order=order, data=price_sensitivity, palette='Blues')\n", "plt.title('Ventas Promedio por SKU según Rango de Precio')\n", "plt.ylabel('Ventas Promedio ($)')\n", 'plt.show()']}, {'cell_type': 'markdown', 'metadata': {}, 'source': ['---\n', '## 4. Dimensión Entorno: Tiempo y Espacio (Nuevos Requerimientos)\n', '\n', 'Aquí respondemos las nuevas preguntas:\n', '1.  **Evolución:** ¿Cómo crecieron los formatos en el tiempo?\n', '2.  **Distribución:** ¿Qué tiendas hay en cada ciudad?']}, {'cell_type': 'code', 'execution_count': 13, 'metadata': {}, 'outputs': [{'data': {'text/plain': ['<Figure size 1800x600 with 3 Axes>']}, 'metadata': {}, 'output_type': 'display_data'}], 'source': ['fig, axes = plt.subplots(1, 2, figsize=(18, 6))\n', '\n', '# 4.1 Crecimiento de Tiendas (Countplot por Año)\n', '# Nota: Esto muestra cuándo se fundaron las tiendas existentes.\n', "sns.countplot(x='Outlet_Establishment_Year', hue='Outlet_Type', data=df_merged.drop_duplicates(subset=['Outlet_Identifier']), ax=axes[0], palette='muted')\n", "axes[0].set_title('Apertura de Tiendas por Año')\n", "axes[0].set_ylabel('Cantidad de Tiendas Nuevas')\n", "axes[0].tick_params(axis='x', rotation=45)\n", '\n', '# 4.2 Distribución por Tipo de Ciudad (Heatmap)\n', "city_store_matrix = pd.crosstab(store_meta['Outlet_Location_Type'], store_meta['Outlet_Type'])\n", "sns.heatmap(city_store_matrix, annot=True, fmt='d', cmap='YlGnBu', ax=axes[1])\n", "axes[1].set_title('Distribución de Formatos por Tipo de Ciudad')\n", '\n', 'plt.tight_layout()\n', 'plt.show()']}, {'cell_type': 'markdown', 'metadata': {}, 'source': ['### Insights de Entorno\n', '* **Evolución:** Hubo un pico de expansión en 1985 (muchas Grocery Stores y Supermarket Type3). Luego una pausa y reactivación a finales de los 90s con Supermarket Type1.\n', '* **Geografía:**\n', '    * **Tier 2** es territorio exclusivo de `Supermarket Type1`.\n', '    * **Tier 3** es el más diverso: Tiene todos los tipos (Grocery, Type1, Type2, Type3).\n', '    * **Tier 1** mezcla Grocery y Type1.\n', '    * *Insight:* Tier 3 parece ser el mercado de prueba o el más saturado.']}, {'cell_type': 'markdown', 'metadata': {}, 'source': ['---\n', '## 5. Dimensión Operativa: Visibilidad y Ventas\n', '\n', 'Preguntas a responder:\n', '1.  ¿El tamaño de la tienda influye en la visibilidad?\n', '2.  ¿Más visibilidad = Más ventas?']}, {'cell_type': 'code', 'execution_count': 6, 'metadata': {}, 'outputs': [{'name': 'stderr', 'output_type': 'stream', 'text': ['/tmp/ipykernel_876680/918682932.py:6: FutureWarning: \n', '\n', 'Passing `palette` without assigning `hue` is deprecated and will be removed in v0.14.0. Assign the `x` variable to `hue` and set `legend=False` for the same effect.\n', '\n', "  sns.boxplot(x='Outlet_Size', y='Item_Visibility', data=df_merged, order=['Small', 'Medium', 'High'], ax=axes[0], palette='Set2')\n"]}, {'data': {'text/plain': ['<Figure size 1800x600 with 2 Axes>']}, 'metadata': {}, 'output_type': 'display_data'}], 'source': ['fig, axes = plt.subplots(1, 2, figsize=(18, 6))\n', '\n', '# 5.1 Tamaño vs Visibilidad Promedio\n', '# Hipótesis: Tiendas chicas tienen menos espacio, ¿quizás muestran menos productos?\n', '# O al revés: al tener menos productos, cada uno ocupa más % del total relativo.\n', "sns.boxplot(x='Outlet_Size', y='Item_Visibility', data=df_merged, order=['Small', 'Medium', 'High'], ax=axes[0], palette='Set2')\n", "axes[0].set_title('Distribución de Visibilidad por Tamaño de Tienda')\n", '\n', '# 5.2 Visibilidad vs Ventas\n', "sns.scatterplot(x='Item_Visibility', y='Item_Outlet_Sales', hue='Outlet_Type', alpha=0.3, data=df_merged, ax=axes[1])\n", "axes[1].set_title('Correlación: Visibilidad vs Ventas')\n", 'axes[1].set_ylim(0, 8000)\n', '\n', 'plt.tight_layout()\n', 'plt.show()']}, {'cell_type': 'markdown', 'metadata': {}, 'source': ['### Insights Operativos\n', '1.  **Tamaño vs Visibilidad:** ¡Sorpresa! Las tiendas **Small** tienen una visibilidad promedio *mayor*. \n', '    * *Razón Técnica:* La visibilidad es un ratio (Espacio del producto / Espacio Total). En una tienda chica con menos inventario total, cada producto individual ocupa proporcionalmente más "atención" del cliente.\n', '2.  **Visibilidad vs Ventas:** Existe una relación extraña. Muchos productos con alta visibilidad tienen ventas bajas (la "panza" de puntos azules/naranjas a la derecha). Esto suele indicar que dar mucha visibilidad a productos que no rotan es desperdicio de espacio.']}, {'cell_type': 'markdown', 'metadata': {}, 'source': ['---\n', '## 6. Preferencias Geográficas (Tier vs Categoría)\n', '\n', '¿Influye el tipo de ciudad en *qué* se vende más?']}, {'cell_type': 'code', 'execution_count': 7, 'metadata': {}, 'outputs': [{'data': {'text/plain': ['<Figure size 1000x400 with 2 Axes>']}, 'metadata': {}, 'output_type': 'display_data'}], 'source': ['# Tabla pivote: Ventas Promedio por Categoría en cada Tier\n', "tier_pref = df_merged.groupby(['Outlet_Location_Type', 'Broad_Category'])['Item_Outlet_Sales'].mean().unstack()\n", '\n', 'plt.figure(figsize=(10, 4))\n', "sns.heatmap(tier_pref, annot=True, fmt='.0f', cmap='Greens')\n", "plt.title('Ventas Promedio ($) por Categoría y Tipo de Ciudad')\n", 'plt.show()']}, {'cell_type': 'markdown', 'metadata': {}, 'source': ['### Insight Geográfico\n', 'Las preferencias son bastante estables.

No se ve que Tier 1 prefiera drásticamente "Drinks" sobre Tier 3. Esto refuerza la teoría inicial: **El mercado es homogéneo en gustos, la diferencia la marca la capacidad operativa de la tienda (Supermarket Type 3 en Tier 3 arrasa).**']}, {'cell_type': 'markdown', 'metadata': {}, 'source': ['---\n', '## 7. Generación del JSON Final para Dashboard\n', '\n', 'Consolidamos todo en un JSON. Incluiremos:\n', '1.  **Perfil:** Ventas, Eficiencia, Año Estab, Ubicación.\n', '2.  **Breakdown Clusters:** Valores y Porcentajes.\n', '3.  **Insight Estratégico:** Tier de Precio Dominante.\n', '4.  **Top Products:** Los 3 mejores SKUs.']}, {'cell_type': 'code', 'execution_count': 8, 'metadata': {}, 'outputs': [{'name': 'stdout', 'output_type': 'stream', 'text': ['✅ JSON Final Generado: ../Data/Processed/store_hierarchy_final.json\n', 'Este archivo contiene la estructura completa para visualizar Mix, Escala, Top Items e Info de Entorno.\n']}], 'source': ['# Función auxiliar para obtener Top Products\n', 'def get_top_items(df_subset, n=3):\n', "    return df_subset.groupby(['Item_Identifier', 'Broad_Category'])['Item_Outlet_Sales'].sum() \\\n", "    .sort_values(ascending=False).head(n).reset_index().to_dict('records')\n", '\n', 'json_data = []\n', '\n', 'for store_id, row in df_dna.iterrows():\n', '    # Subconjunto de datos para esta tienda\n', "    subset = df_merged[df_merged['Outlet_Identifier'] == store_id]\n", '    \n', '    # Insight de Precio\n', "    best_tier = subset.groupby('Price_Tier')['Item_Outlet_Sales'].mean().idxmax()\n", '    \n', '    store_obj = {\n', '        "id": store_id,\n', '        "type": row[\'Outlet_Type\'],\n', '        "size": str(row[\'Outlet_Size\']),\n', '        "location": row[\'Outlet_Location_Type\'],\n', '        "year_established": int(row[\'Outlet_Establishment_Year\']),\n', '        "total_sales": row[\'Total_Sales\'],\n', '        "dominant_price_tier": best_tier,\n', '        "top_products": get_top_items(subset),\n', '        "breakdown": []\n', '    }\n', '    \n', '    # Breakdown de Clusters\n', '    for cluster in cluster_cols:\n', '        abs_val = row[cluster]\n', '        pct_val = df_relative.loc[store_id, cluster]\n', '        if abs_val > 0:\n', "            store_obj['breakdown'].append({\n", '                "cluster": cluster,\n', '                "value": round(abs_val, 2),\n', '                "percent": round(pct_val, 2)\n', '            })\n', '    \n', '    json_data.append(store_obj)\n', '\n', "output_path = '../Data/Processed/store_hierarchy_final.json'\n", "with open(output_path, 'w') as f:\n", '    json.dump(json_data, f, indent=4)\n', '\n', 'print(f"✅ JSON Final Generado: {output_path}")\n', 'print("Este archivo contiene la estructura completa para visualizar Mix, Escala, Top Items e Info de Entorno.")']}, {'cell_type': 'markdown', 'metadata': {}, 'source': ['## Resumen Final de la Notebook\n', '\n', 'Hemos completado el análisis integrando las dimensiones solicitadas:\n', '\n', '1.  **Crecimiento:** Identificamos olas de apertura en 1985, 1997-1999 y 2000s.\n', '2.  **Ubicación:** Tier 3 es el mercado más saturado y diverso.\n', '3.  **Visibilidad:** Mayor en tiendas pequeñas (por ratio de espacio), pero no garantiza linealmente mayores ventas.\n', '4.  **Estrategia de Precio:** Confirmamos que productos Premium tienen mejor rendimiento unitario en todos los formatos.\n', '5.  **Output:** El archivo `store_hierarchy_final.json` está listo para ser la columna vertebral de tu Dashboard Web en la Fase E.']}], 'metadata': {'kernelspec': {'display_name': 'Python 3', 'language': 'python', 'name': 'python3'}, 'language_info': {'codemirror_mode': {'name': 'ipython', 'version': 3}, 'file_extension': '.py', 'mimetype': 'text/x-python', 'name': 'python', 'nbconvert_exporter': 'python', 'pygments_lexer': 'ipython3', 'version': '3.12.8'}}, 'nbformat': 4, 'nbformat_minor': 4}
```

``` Folder structure

.

├── Data

│   ├── Processed
│   │   ├── product_level_interpretation.csv
│   │   ├── product_level_modeling.csv
│   │   ├── product_level_with_clusters.csv
│   │   └── store_hierarchy_final.json
│   └── Raw
│       └── train.csv
├── Docs
│   ├── EDA.md
│   └── Instructions.pdf
├── index.html
├── Notebooks
│   ├── fase_a.ipynb
│   ├── fase_b.ipynb
│   ├── fase_c.ipynb
│   └── fase_d.ipynb
├── PROMPT.md
├── README.md
└── Scripts
    └── stringify.py

7 directories, 15 files
```

``` Dataset Overview
Train Dataset (8,523 records)
Includes both input features and the target variable (Item_Outlet_Sales).

Product Features
* Item_Identifier: Unique product ID
* Item_Weight: Weight of the product
* Item_Fat_Content: Fat level (low-fat or regular)
* Item_Visibility: Percentage of display area allocated to the product
* Item_Type: Category of the product
* Item_MRP: Maximum Retail Price

Store Features
* Outlet_Identifier: Unique store ID
* Outlet_Establishment_Year: Year the store was established
* Outlet_Size: Store size (small, medium, large)
* Outlet_Location_Type: City tier classification
* Outlet_Type: Type of outlet (grocery store, supermarket, etc.)

Target Variable
* Item_Outlet_Sales: Sales of the product at a particular store (to be predicted if a regression problem is needed)
```
```