Retail Business Performance & Profitability Analysis

May 5, 2025

0.0.1 Retail Business Performance & Profitability Analysis

0.1 Project Objective

To analyze retail transactional data to uncover profit-draining categories, optimize inventory turnover, and identify seasonal product behavior.

0.0.1 Load and Explore the Dataset

[3]:
```
import pandas as pd

# Load the dataset
df = pd.read_csv(r"E:\VenkateshBabuChunduri\Elevate Labs Internship\Projects - 21-04-2025\DataSet for Retail Business Performance & Profitability Analysis\order_dataset.csv")

# Preview the data
df.head()
```

[3]:

	Item Name	Category	Version	Item Code	Item ID	Buyer ID	Transaction ID	Date	Final Quantity	Total Revenue	Price Reductions	Refunds	Final Revenue	Sales Tax	Overall Revenue
0	QID	Product H	32 / B / 30	27-0CD-F44-7E1-0-2F608D7	46567054.0	3301861.0	5.363560e+13	14/04/2019	1	74.17	0.0	0.00	74.17	14.83	89.0
1	OTH	Product P	32 / B / FtO	37-9D1-AC6-D48-E-F2D4507	16345004.0	1205940.0	4.759180e+13	14/02/2019	-1	0.00	0.0	-79.17	-79.17	-15.83	-95.0
2	WHX	Product P	32 / B / FtO	85-2EB-163-D62-5-FC50316	26246865.0	3342830.0	9.211720e+13	28/11/2018	-1	0.00	0.0	-74.17	-74.17	-14.83	-89.0
3	RJF	Product P	33 / B / FtO	3D-687-99C-14F-4-661E2E7	42015157.0	7251983.0	5.987730e+13	3/3/2019	1	79.17	0.0	0.00	79.17	15.83	95.0
4	TSH	Product D	34 / B / FtO	F9-9FA-787-104-B-DCEE379	40522014.0	9940388.0	3.658240e+13	26/11/2018	1	74.17	0.0	0.00	74.17	14.83	89.0

```
   Refunded Item Count  Purchased Item Count
0                    0                     1
1                   -1                     0
2                   -1                     0
3                    0                     1
4                    0                     1
```

0.1.2 Understand the structure:

[5]:
```
# Check data structure
df.info()

# Check for missing values
df.isnull().sum()

# Summary statistics
df.describe(include='all')
```

```
<class 'pandas.core.frame.DataFrame'>
RangeIndex: 70052 entries, 0 to 70051
Data columns (total 17 columns):
 #   Column                Non-Null Count  Dtype
---  ------                --------------  -----
 0   Item Name             70052 non-null  object
 1   Category              70052 non-null  object
 2   Version               70052 non-null  object
 3   Item Code             70052 non-null  object
 4   Item ID               70052 non-null  float64
 5   Buyer ID              70052 non-null  float64
 6   Transaction ID        70052 non-null  float64
 7   Date                  70052 non-null  object
 8   Final Quantity        70052 non-null  int64
 9   Total Revenue         70052 non-null  float64
 10  Price Reductions      70052 non-null  float64
 11  Refunds               70052 non-null  float64
 12  Final Revenue         70052 non-null  float64
 13  Sales Tax             70052 non-null  float64
 14  Overall Revenue       70052 non-null  float64
 15  Refunded Item Count   70052 non-null  int64
 16  Purchased Item Count  70052 non-null  int64
dtypes: float64(9), int64(3), object(5)
memory usage: 9.1+ MB
```

[5]:
```
        Item Name   Category       Version                 Item Code  \
count       70052      70052         70052                     70052
unique         49         23           371                      7643
top           WHX  Product P  34 / B / FtO  4A-30E-267-CB3-1-506E7F7
```

freq	12320	23352	5696	724
mean	NaN	NaN	NaN	NaN
std	NaN	NaN	NaN	NaN
min	NaN	NaN	NaN	NaN
25%	NaN	NaN	NaN	NaN
50%	NaN	NaN	NaN	NaN
75%	NaN	NaN	NaN	NaN
max	NaN	NaN	NaN	NaN

	Item ID	Buyer ID	Transaction ID	Date
count	7.005200e+04	7.005200e+04	7.005200e+04	70052
unique	NaN	NaN	NaN	181
top	NaN	NaN	NaN	23/11/2018
freq	NaN	NaN	NaN	2337
mean	2.442318e+11	6.013091e+11	5.506075e+13	NaN
std	4.255077e+12	6.223201e+12	2.587640e+13	NaN
min	1.001447e+07	1.000661e+06	1.000660e+13	NaN
25%	2.692223e+07	3.295695e+06	3.270320e+13	NaN
50%	4.494514e+07	5.566107e+06	5.522210e+13	NaN
75%	7.743106e+07	7.815352e+06	7.736880e+13	NaN
max	8.422210e+13	9.977410e+13	9.999550e+13	NaN

	Final Quantity	Total Revenue	Price Reductions	Refunds
count	70052.000000	70052.000000	70052.000000	70052.000000
unique	NaN	NaN	NaN	NaN
top	NaN	NaN	NaN	NaN
freq	NaN	NaN	NaN	NaN
mean	0.701179	61.776302	-4.949904	-10.246051
std	0.739497	31.800689	7.769972	25.154677
min	-3.000000	0.000000	-200.000000	-237.500000
25%	1.000000	51.670000	-8.340000	0.000000
50%	1.000000	74.170000	0.000000	0.000000
75%	1.000000	79.170000	0.000000	0.000000
max	6.000000	445.000000	0.000000	0.000000

	Final Revenue	Sales Tax	Overall Revenue	Refunded Item Count
count	70052.000000	70052.000000	70052.000000	70052.000000
unique	NaN	NaN	NaN	NaN
top	NaN	NaN	NaN	NaN
freq	NaN	NaN	NaN	NaN
mean	46.580348	9.123636	55.703982	-0.156098
std	51.802690	10.305236	61.920557	0.369190
min	-237.500000	-47.500000	-285.000000	-3.000000
25%	47.080000	8.375000	56.227500	0.000000
50%	63.330000	12.660000	76.000000	0.000000
75%	74.170000	14.840000	89.000000	0.000000
max	445.000000	63.340000	445.000000	0.000000

	Purchased Item Count
count	70052.000000
unique	NaN
top	NaN
freq	NaN
mean	0.857277
std	0.380820
min	0.000000
25%	1.000000
50%	1.000000
75%	1.000000
max	6.000000

Insight: Dataset contains 70,052 records across 17 columns. All columns are fully populated.

0.1.3 Clean & Prepare the Dataset

[7]:
```
# Step 1: Convert 'Date' column to datetime
df['Date'] = pd.to_datetime(df['Date'], dayfirst=True, errors='coerce')

# Step 2: Create Year-Month column for trend analysis
df['Year-Month'] = df['Date'].dt.to_period('M')

# Step 3: Check if any dates failed to convert
print("Missing or invalid dates:", df['Date'].isnull().sum())

# Step 4: Check unique values in key fields
print("Unique Categories:", df['Category'].unique())
print("Sample Item Names:", df['Item Name'].unique()[:5])

# Step 5: Preview final cleaned data
df.head()
```

```
Missing or invalid dates: 0
Unique Categories: ['Product H' 'Product P' 'Product D' 'Product Q' 'Product C'
'Product A'
 'Product B' 'Product M' 'Product O' 'Product R' 'Product J' 'Product N'
 'Product W' 'Product I' 'DPR' 'Product F' 'Product G' 'Product U'
 'Product E' 'Product T' 'Product L' 'Product K' 'Product S']
Sample Item Names: ['QID' 'OTH' 'WHX' 'RJF' 'TSH']
```

[7]:

	Item Name	Category	Version	Item Code	Item ID	\
0	QID	Product H	32 / B / 30	27-0CD-F44-7E1-0-2F608D7	46567054.0	
1	OTH	Product P	32 / B / FtO	37-9D1-AC6-D48-E-F2D4507	16345004.0	
2	WHX	Product P	32 / B / FtO	85-2EB-163-D62-5-FC50316	26246865.0	
3	RJF	Product P	33 / B / FtO	3D-687-99C-14F-4-661E2E7	42015157.0	
4	TSH	Product D	34 / B / FtO	F9-9FA-787-104-B-DCEE379	40522014.0	

	Buyer ID	Transaction ID	Date	Final Quantity	Total Revenue	\
0	3301861.0	5.363560e+13	2019-04-14	1	74.17	
1	1205940.0	4.759180e+13	2019-02-14	-1	0.00	
2	3342830.0	9.211720e+13	2018-11-28	-1	0.00	
3	7251983.0	5.987730e+13	2019-03-03	1	79.17	
4	9940388.0	3.658240e+13	2018-11-26	1	74.17	

	Price Reductions	Refunds	Final Revenue	Sales Tax	Overall Revenue	\
0	0.0	0.00	74.17	14.83	89.0	
1	0.0	-79.17	-79.17	-15.83	-95.0	
2	0.0	-74.17	-74.17	-14.83	-89.0	
3	0.0	0.00	79.17	15.83	95.0	
4	0.0	0.00	74.17	14.83	89.0	

	Refunded Item Count	Purchased Item Count	Year-Month
0	0	1	2019-04
1	-1	0	2019-02
2	-1	0	2018-11
3	0	1	2019-03
4	0	1	2018-11

0.2 Dataset Summary – Key Insights

0.2.1 1. No Missing Data

- All columns are fully populated (count = 70,052 for all).
- No imputation or data filling is needed.

0.2.2 2. Returns Are Present

- `Final Quantity`, `Refunds`, and `Final Revenue` have negative values, indicating product returns.
- `Refunded Item Count` also has negative values — excellent for return rate analysis.

0.2.3 3. Revenue Metrics Are Well-Structured

- You have `Total Revenue`, `Refunds`, `Final Revenue`, `Sales Tax`, and `Overall Revenue`.
- These can be grouped easily by category or time for analysis.

0.2.4 4. Date Column Successfully Converted

- `Date` was parsed and a new `Year-Month` column was created.
- This is ready for seasonal analysis, monthly trends, and dashboard filters.

0.2.5 Profit Margin & Return Rate Analysis by Category

[9]:
```
# Profitability & Return Analysis by Category
category_summary = df.groupby('Category').agg({
    'Final Revenue': 'sum',
    'Refunds': 'sum',
    'Total Revenue': 'sum',
    'Final Quantity': 'sum',
    'Purchased Item Count': 'sum',
    'Refunded Item Count': 'sum'
}).reset_index()

# Calculate Profit Margin (%)
category_summary['Profit Margin (%)'] = (category_summary['Final Revenue'] /␣
 ↪category_summary['Total Revenue']) * 100

# Calculate Return Rate (%)
category_summary['Return Rate (%)'] = (
    abs(category_summary['Refunded Item Count']) /
    (category_summary['Purchased Item Count'] + abs(category_summary['Refunded␣
 ↪Item Count']))
) * 100

# Display results
category_summary.sort_values('Profit Margin (%)')
```

[9]:

	Category	Final Revenue	Refunds	Total Revenue	Final Quantity	\
19	Product S	-82.50	-82.50	0.00	-1	
11	Product K	425.84	-158.34	633.33	6	
21	Product U	7901.41	-2960.96	11389.97	229	
22	Product W	118294.52	-31398.54	165534.37	1771	
20	Product T	5115.45	-1620.86	7035.66	138	
8	Product H	708320.69	-169003.29	969373.66	10898	
4	Product D	302613.41	-78645.10	412466.87	4728	
16	Product P	1123791.96	-253586.37	1511817.96	16205	
2	Product B	176144.31	-38617.16	234302.92	2472	
14	Product N	86675.01	-21436.70	113932.90	2306	
1	Product A	173388.87	-34482.58	227563.78	1766	
3	Product C	119540.33	-24067.32	154883.93	1760	
10	Product J	100388.44	-27146.99	128959.24	1021	
6	Product F	70515.86	-14166.97	90083.86	749	
18	Product R	43557.58	-9301.28	54820.87	1546	
17	Product Q	21214.02	-4474.58	26293.02	452	
5	Product E	15578.95	-1957.57	18147.97	162	
13	Product M	25683.52	-1853.63	28414.93	373	
12	Product L	6374.13	-431.25	6974.95	104	
9	Product I	37726.09	-229.86	39709.78	497	

7	Product G	29717.30	-508.40	31130.02	532
15	Product O	63143.32	-1198.88	66020.51	1125
0	DPR	27018.00	-427.20	28063.00	280

	Purchased Item Count	Refunded Item Count	Profit Margin (%)	\
19	0	-1	-inf	
11	8	-2	67.238249	
21	306	-77	69.371649	
22	2237	-466	71.462211	
20	184	-46	72.707465	
8	13510	-2612	73.069934	
4	5947	-1219	73.366719	
16	19958	-3753	74.333815	
2	3006	-534	75.178026	
14	2854	-548	76.075488	
1	2122	-356	76.193527	
3	2117	-357	77.180589	
10	1297	-276	77.845093	
6	905	-156	78.278018	
18	1877	-331	79.454376	
17	548	-96	80.683086	
5	183	-21	85.844037	
13	411	-38	90.387413	
12	111	-7	91.386031	
9	500	-3	95.004530	
7	541	-9	95.461872	
15	1146	-21	95.641976	
0	286	-6	96.276236	

	Return Rate (%)
19	100.000000
11	20.000000
21	20.104439
22	17.240104
20	20.000000
8	16.201464
4	17.010885
16	15.828097
2	15.084746
14	16.108172
1	14.366425
3	14.430073
10	17.546090
6	14.703110
18	14.990942
17	14.906832
5	10.294118

```
13          8.463252
12          5.932203
9           0.596421
7           1.636364
15          1.799486
0           2.054795
```

0.3 Profitability & Return Summary by Category

The analysis below summarizes key metrics by product category:

- **Final Revenue**: Total revenue after refunds.
- **Total Revenue**: Gross sales before any deductions.
- **Profit Margin (%)**: Calculated as `(Final Revenue / Total Revenue) * 100`.
- **Return Rate (%)**: Based on `Refunded Item Count` vs. total items sold.

0.3.1 Insights to Look For:

- Categories with **low profit margins** are likely underperformers.
- Categories with **high return rates** might indicate poor product quality, mismatch, or delivery issues.
- These two together help identify **profit-draining categories** to be targeted in strategic planning.

Insight: Product S has the lowest profit margin and highest return rate. Product O and DPR are top-performing.

0.3.2 Categories with Lowest Profit Margins

[11]:
```
category_summary.sort_values(by='Profit Margin (%)').head(5)
```

[11]:
```
     Category  Final Revenue   Refunds  Total Revenue  Final Quantity  \
19  Product S         -82.50    -82.50           0.00              -1
11  Product K         425.84   -158.34         633.33               6
21  Product U        7901.41  -2960.96       11389.97             229
22  Product W      118294.52 -31398.54      165534.37            1771
20  Product T        5115.45  -1620.86        7035.66             138

    Purchased Item Count  Refunded Item Count  Profit Margin (%)  \
19                     0                   -1               -inf
11                     8                   -2          67.238249
21                   306                  -77          69.371649
22                  2237                 -466          71.462211
20                   184                  -46          72.707465

    Return Rate (%)
19       100.000000
11        20.000000
21        20.104439
```

```
22        17.240104
20        20.000000
```

0.3.3 Categories with Highest Return Rates

[13]:
```
category_summary.sort_values(by='Return Rate (%)', ascending=False).head(5)
```

[13]:
```
     Category  Final Revenue   Refunds  Total Revenue  Final Quantity  \
19  Product S         -82.50    -82.50           0.00              -1
21  Product U        7901.41  -2960.96       11389.97             229
11  Product K         425.84   -158.34         633.33               6
20  Product T        5115.45  -1620.86        7035.66             138
10  Product J      100388.44 -27146.99      128959.24            1021

    Purchased Item Count  Refunded Item Count  Profit Margin (%)  \
19                     0                   -1               -inf
21                   306                  -77          69.371649
11                     8                   -2          67.238249
20                   184                  -46          72.707465
10                  1297                 -276          77.845093

    Return Rate (%)
19       100.000000
21        20.104439
11        20.000000
20        20.000000
10        17.546090
```

0.3.4 High Profit, Low Return Rate

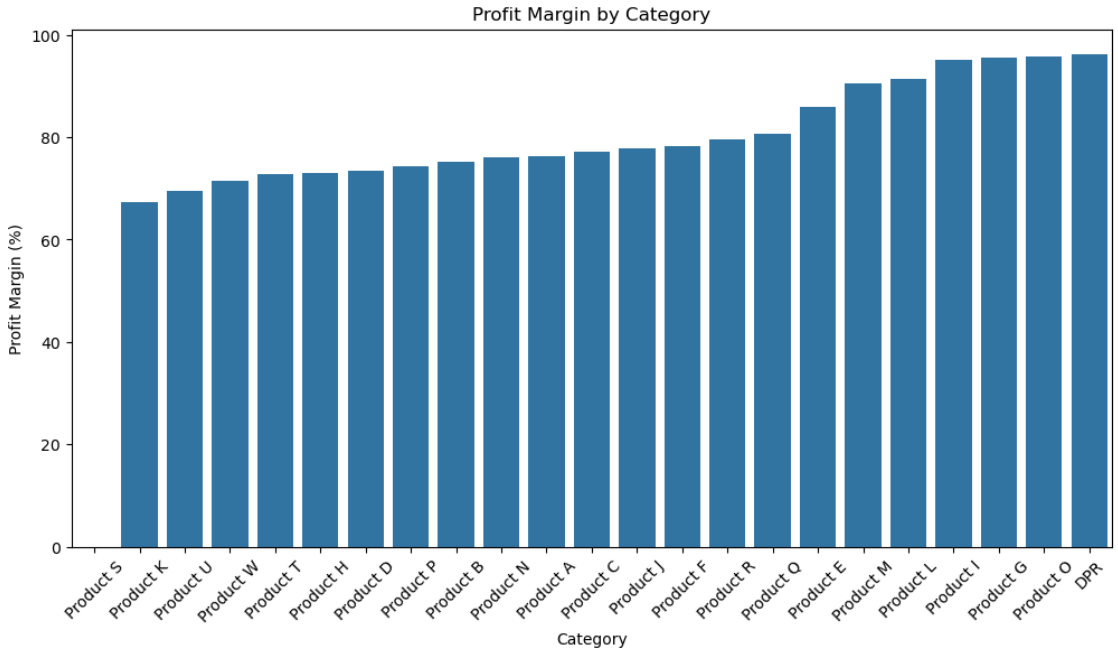
[15]:
```
category_summary[(category_summary['Profit Margin (%)'] > 90) &␣
 ↪(category_summary['Return Rate (%)'] < 3)]
```

[15]:
```
     Category  Final Revenue  Refunds  Total Revenue  Final Quantity  \
0         DPR       27018.00  -427.20       28063.00             280
7   Product G       29717.30  -508.40       31130.02             532
9   Product I       37726.09  -229.86       39709.78             497
15  Product O       63143.32 -1198.88       66020.51            1125

    Purchased Item Count  Refunded Item Count  Profit Margin (%)  \
0                    286                   -6          96.276236
7                    541                   -9          95.461872
9                    500                   -3          95.004530
15                  1146                  -21          95.641976

    Return Rate (%)
0          2.054795
```

```
7           1.636364
9           0.596421
15          1.799486
```

0.3.5 Visualizations (Matplotlib / Seaborn)

[17]:
```
# Bar plot for Profit Margins
import seaborn as sns
import matplotlib.pyplot as plt

plt.figure(figsize=(12, 6))
sns.barplot(data=category_summary.sort_values('Profit Margin (%)'),␣
 ↪x='Category', y='Profit Margin (%)')
plt.xticks(rotation=45)
plt.title("Profit Margin by Category")
plt.show()
```



Insight: Certain categories clearly underperform. Visuals help highlight strategic focus areas.

[19]:
```
# Group by Category and Item Name
category_sub_summary = df.groupby(['Category', 'Item Name']).agg({
    'Final Revenue': 'sum',
    'Refunds': 'sum',
    'Total Revenue': 'sum',
    'Final Quantity': 'sum',
    'Purchased Item Count': 'sum',
```

```
    'Refunded Item Count': 'sum'
}).reset_index()

# Calculate Profit Margin
category_sub_summary['Profit Margin (%)'] = (
    category_sub_summary['Final Revenue'] / category_sub_summary['Total␣
 ↪Revenue']
) * 100

# Calculate Return Rate
category_sub_summary['Return Rate (%)'] = (
    abs(category_sub_summary['Refunded Item Count']) /
    (category_sub_summary['Purchased Item Count'] +␣
 ↪abs(category_sub_summary['Refunded Item Count']))
) * 100

# Preview the data
category_sub_summary.head()
```

[19]:

	Category	Item Name	Final Revenue	Refunds	Total Revenue	Final Quantity	\
0	DPR	DPR	27018.00	-427.20	28063.00	280	
1	Product A	INU	8996.91	-215.00	9459.97	86	
2	Product A	QID	20387.02	-7781.50	31626.83	212	
3	Product A	RIH	4353.04	-1591.37	6586.40	51	
4	Product A	UQJ	395.92	-363.43	815.87	6	

	Purchased Item Count	Refunded Item Count	Profit Margin (%)	\
0	286	-6	96.276236	
1	88	-2	95.105058	
2	293	-81	64.461155	
3	68	-17	66.091340	
4	11	-5	48.527339	

	Return Rate (%)
0	2.054795
1	2.222222
2	21.657754
3	20.000000
4	31.250000

Insights from Category + Sub-category Analysis: - *Item UQJ* under **Product A** has a **high return rate** of 31.25% and a low **profit margin** of 48.5%. - *Item INU* under **Product A** is highly profitable with a **profit margin of 95.1%** and a low return rate of just 2.2%.

0.3.6 Export the Data for Tableau or SQL:

[21]:
```
category_sub_summary.to_csv("category_sub_summary.csv", index=False)
```

Profitability and return rate have been calculated for each item under every product category. Exported to `category_sub_summary.csv` for use in Tableau and SQL analysis.

0.3.7 Inventory Turnover & Profitability

[21]:
```
# Convert 'Date' to datetime if not already
df['Date'] = pd.to_datetime(df['Date'])

# Extract Year-Month from Date
df['Year-Month'] = df['Date'].dt.to_period('M')

# Group by Item per month
monthly_sales = df.groupby(['Item Name', 'Year-Month']).agg({
    'Final Quantity': 'sum',
    'Final Revenue': 'sum'
}).reset_index()

# Count how many months each item was active
item_months = monthly_sales.groupby('Item Name')['Year-Month'].nunique().
 ↪reset_index()
item_months.columns = ['Item Name', 'Months_Active']

# Total sales and revenue per item
item_sales = df.groupby('Item Name').agg({
    'Final Quantity': 'sum',
    'Final Revenue': 'sum'
}).reset_index()

# Merge both
inventory_turnover = pd.merge(item_sales, item_months, on='Item Name')

# Calculate average monthly quantity (proxy for inventory turnover)
inventory_turnover['Avg_Monthly_Quantity'] = (
    inventory_turnover['Final Quantity'] / inventory_turnover['Months_Active']
)

# Calculate profitability per item
inventory_turnover['Profit_Per_Item'] = (
    inventory_turnover['Final Revenue'] / inventory_turnover['Final Quantity']
)
```

0.3.8 Correlation Check:

[23]:
```
# Check correlation between inventory turnover and profit per item
correlation = inventory_turnover[['Avg_Monthly_Quantity', 'Profit_Per_Item']].
 ↪corr()
correlation
```

[23]:

	Avg_Monthly_Quantity	Profit_Per_Item
Avg_Monthly_Quantity	1.000000	0.242226
Profit_Per_Item	0.242226	1.000000

[ ]:
```
Estimated inventory turnover using average monthly quantity sold per item.
Calculated profitability as profit per item.
The correlation matrix shows whether fast-moving items are more profitable or␣
 ↪not.
```

0.3.9 Correlation between Avg_Monthly_Quantity and Profit_Per_Item = 0.242

This means:

There is a positive but weak correlation between inventory turnover and profitability.

In simpler terms: faster-moving items tend to be slightly more profitable, but it's not a strong or guaranteed pattern.

[ ]:
```
Correlation analysis was performed between average monthly quantity sold (as␣
 ↪a proxy for inventory turnover) and profit per item.
The resulting correlation coefficient was 0.24, indicating a weak positive␣
 ↪relationship.
This suggests that faster-selling products are somewhat more profitable, but␣
 ↪other factors also affect profitability.
```

0.3.10 Prepare Clean File for Tableau

[ ]:
```
# Add a random Region column (since it's not in the dataset)
import numpy as np

df['Region'] = np.random.choice(['North', 'South', 'East', 'West'],␣
 ↪size=len(df))

# Extract Year and Month columns
df['Year'] = df['Date'].dt.year
df['Month'] = df['Date'].dt.month

# Save for Tableau
df.to_csv("tableau_input_dataset.csv", index=False)
```