FINAL PROJECT

E-Commerce Sales Analysis

Presented By Faridz Salman Al Parissy

MySkill

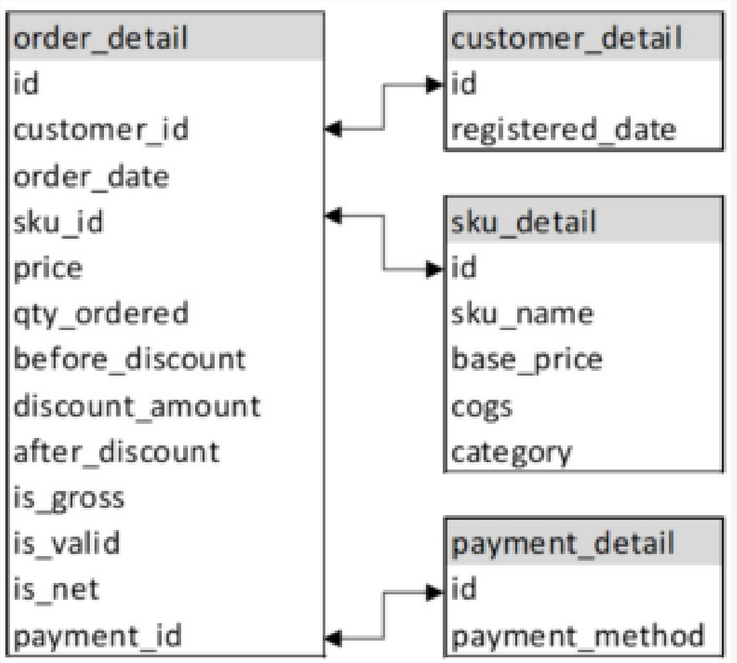
About the dataset

The dataset countaints 4 tables

1. Order_detail table stores comprehensive information about each item within a customer's order
2. Customer_detail provides information about customers and when they registered as a customer.
3. Payment_detail table provides information about the payment method of each customer
4. Sku_detail table provides detailed information about each product offered, going beyond basic identification to include production cost and selling price data as well as categories.

The ecommerce dataset make up 14 relevant features

1. Order_id : Unique identifier for each order placed by a table
2. Customer_id : Identifier for each customer
3. Order_date : Date the order was placed by the customer
4. Category : Product Category
5. Sku_name : Name of product
6. Price : Price of each product
7. Cogs : Cost of production before marketing
8. Qty_ordered : Number of orders
9. Before_discount : (price*qty_ordered) Product price of each order
10. Discount_amount : Discounted price every order
11. After_discount : The total transaction price of each order
12. Is_gross : Customers who have not made payments
13. Is_valid : Customers who have made payments
14. Is_net : Customers who have completed transactions



MySkill

Data Overview

All data use after_discount and is_valid = 1

Total Customer
2864

Total Revenue
Rp 3,67 Miliar

Total Orders
3955

Net Profit
Rp 820 Juta

Total Quantity
8307

Average Order Value
Rp 927,326.25

MySkill

python™

Data Preparation

Library

```
import pandas as pd
import numpy as np
import matplotlib.pyplot as plt
import seaborn as sns
```

Import **Pandas**, **Numpy**, **Matplotlib**, and **Seaborn** libraries to facilitate dataset analysis.

Import data from git

```
path_od = "https://raw.githubusercontent.com/dataskillsboost/FinalProjectDA11/main/order_detail.csv"
path_pd = "https://raw.githubusercontent.com/dataskillsboost/FinalProjectDA11/main/payment_detail.csv"
path_cd = "https://raw.githubusercontent.com/dataskillsboost/FinalProjectDA11/main/customer_detail.csv"
path_sd = "https://raw.githubusercontent.com/dataskillsboost/FinalProjectDA11/main/sku_detail.csv"
```

Reading Csv

```
order_df = pd.read_csv(path_od)
payment_df = pd.read_csv(path_pd)
customer_df = pd.read_csv(path_cd)
sku_df = pd.read_csv(path_sd)
```

Merge table

```
merged_df = pd.merge(order_df, payment_df, left_on='payment_id', right_on='id', how='left', suffixes=('', '_payment'))
merged_df = pd.merge(merged_df, customer_df, left_on='customer_id', right_on='id', how='left', suffixes=('', '_customer'))
merged_df = pd.merge(merged_df, sku_df, left_on='sku_id', right_on='id', how='left', suffixes=('', '_sku'))
```

Drop column

```
merged_df = merged_df.drop(columns=['id_payment','id_sku','id_customer', 'base_price'], errors='ignore')
```

To prepare for data analysis, data from GitHub is imported into a dataframe, then the **order_df**, **payment_df**, **customer_df**, and **sku_df** data are merged into **merge_df** using a *left-join*. Duplicate columns after the merge are removed to maintain data cleanliness.

Data Cleaning

- Check the type of each column
- Check missing value
- Check duplicate data
- Fix unnecessary text

Before

```
Data columns (total 18 columns):
 #   Column            Non-Null Count  Dtype
---  ------            --------------  -----
 0   id                5884 non-null   object
 1   customer_id       5884 non-null   object
 2   order_date        5884 non-null   object
 3   sku_id            5884 non-null   object
 4   price             5884 non-null   int64
 5   qty_ordered       5884 non-null   int64
 6   before_discount   5884 non-null   float64
 7   discount_amount   5884 non-null   float64
 8   after_discount    5884 non-null   float64
 9   is_gross          5884 non-null   int64
 10  is_valid          5884 non-null   int64
 11  is_net            5884 non-null   int64
 12  payment_id        5884 non-null   int64
 13  payment_method    5884 non-null   object
 14  registered_date   5884 non-null   object
 15  sku_name          5884 non-null   object
 16  cogs              5884 non-null   int64
 17  category          5884 non-null   object
dtypes: float64(3), int64(7), object(8)
```

Check Duplicate

```
merged_df.duplicated().sum()
✓ 0.0s
np.int64(0)
```

Float to INT

```
merged_df = merged_df.astype({"before_discount":'int',
                              "discount_amount":'int',
                              "after_discount":'int'})
```

String to Datetime

```
merged_df[['order_date','registered_date']] = merged_df[
    ['order_date','registered_date']].apply(pd.to_datetime)
```

After

```
Data columns (total 18 columns):
 #   Column            Non-Null Count  Dtype
---  ------            --------------  -----
 0   id                5884 non-null   object
 1   customer_id       5884 non-null   object
 2   order_date        5884 non-null   datetime64[ns]
 3   sku_id            5884 non-null   object
 4   price             5884 non-null   int64
 5   qty_ordered       5884 non-null   int64
 6   before_discount   5884 non-null   int64
 7   discount_amount   5884 non-null   int64
 8   after_discount    5884 non-null   int64
 9   is_gross          5884 non-null   int64
 10  is_valid          5884 non-null   int64
 11  is_net            5884 non-null   int64
 12  payment_id        5884 non-null   int64
 13  payment_method    5884 non-null   object
 14  registered_date   5884 non-null   datetime64[ns]
 15  sku_name          5884 non-null   object
 16  cogs              5884 non-null   int64
 17  category          5884 non-null   object
dtypes: datetime64[ns](2), int64(10), object(6)
```

Data Cleaning

- Check the type of each column ✓
- Check missing value
- Check duplicate data ✓
- Fix unnecessary text

Check missing value

```
merged_df.isna().sum()
```

✓ 0.0s

```
id                  0
customer_id         0
order_date          0
sku_id              0
price               0
qty_ordered         0
before_discount     0
discount_amount     0
after_discount      0
is_gross            0
is_valid            0
is_net              0
payment_id          0
payment_method      0
registered_date     0
sku_name            0
cogs                0
category            0
dtype: int64
```

Replace symbol

```
merged_df[['sku_name']] = merged_df[['sku_name']].apply(lambda col: col.str.replace('_|-', ' ', regex=True))
```

Replace the symbol _ or - with " "

Fixing text

```
merged_df.payment_method = merged_df.payment_method.replace({
    'cod': 'COD', 'Payaxis': 'Pay axis', 'Easypay':'Easy Pay',
    'customercredit': 'Customer Credit', 'ublcreditcard':'UBL Credit Card',
    'jazzwallet':'Jazz Wallet','mygateway': 'My Gateway', 'jazzvoucher': 'Jazz Voucher',
    'internetbanking':'Internet Banking', 'mcblite': 'MCB Lite', 'Easypay_MA':'Easy Pay MA',
    'easypay_voucher':'Easy Pay Voucher', 'cashatdoorstep': 'Cash at Door Step', 'productcredit':'Product Credit',
    'financesettlement': 'Finance Settlement', 'marketingexpense': 'Marketing Expense'})
```

Changed the format of the values in the **payment_method** column from no spaces to using spaces. The goal is to improve readability and ease data interpretation. **Example** payaxis to Pay Axis, ublcreditcard to UBL Credit Card, etc

Store dataframe

```
merged_clean = merged_df.copy()
```

To avoid changing the original dataframe, created a copy of the **merged_df** dataframe and stored it in the **merged_clean** variable.

Question/Task

01

At the end of this year, the company will give prizes to customers who win the Year-End Festival competition. The Marketing Team needs help to determine the estimated prizes that will be given to the winners of the competition later. The prizes will be taken from the TOP 5 Products from the Mobiles & Tablets Category during 2022, with the highest sales quantity (valid = 1).

02

Following up on the joint meeting of the Werehouse Team and Marketing Team, we found that the availability of product stock with the Others Category at the end of 2022 was still high.

1. We ask for your assistance in checking the sales data for this category with 2021 in terms of sales quantity. Our temporary suspicion is that there has been a decrease in sales quantity in 2022 compared to 2021. (Please also display data for the 15 categories)
2. If there is indeed a decrease in sales quantity in the Others category, we ask for your assistance in providing data on the TOP 20 product names that experienced the highest decrease in 2022 compared to 2021. We will use this as discussion material at the next meeting.

03

Regarding the company's anniversary in the next 2 months, the Digital Marketing Team will provide promotional information for customers at the end of this month. The customer criteria that we will need are those who have checked out but have not made a payment (is_gross = 1) during 2022. The data we need is Customer ID and Registered Date.

04

From October to December 2022, we run a campaign every Saturday and Sunday. We want to assess whether the campaign has a significant impact on increasing sales (before_discount). Please help us display the following data:

1. Average daily sales for weekends (Saturday and Sunday) vs. average daily sales for weekdays (Monday-Friday) per month. Is there an increase in sales in each of these months?
2. Average daily sales for weekends (Saturday and Sunday) vs. average daily sales for weekdays (Monday-Friday) for the entire 3 months.

MySkill python

01

Showing top 5 best-selling mobile and tablet products of 2022 will be awarded as prizes in the Year-End Festival competition.

Logical Thinking

- Create new column **year**, extract from **order_date**
- Filtering data **valid** by 1, **year** = 2022, and grouping by **category** = Mobiles & Tablets
- Create aggregation **total_qty**
- Grouping data by **year** and **sku_name**
- Sort data by **total_qty**
- Limit data to 5

Syntax top 5 qty

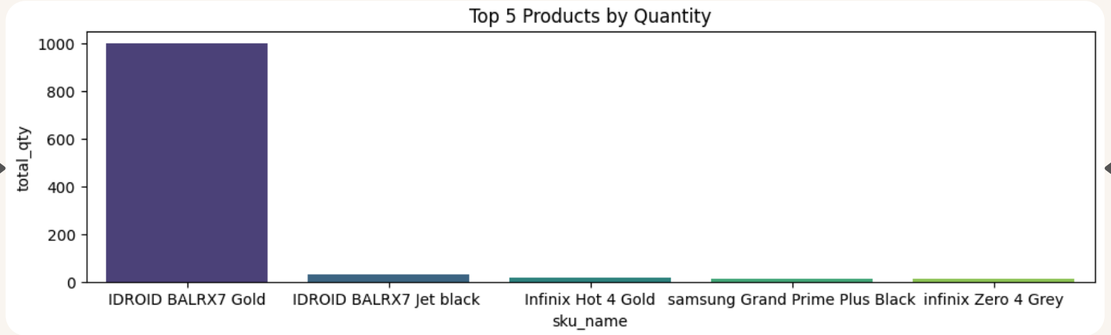
```python
merged_clean['year'] = merged_clean['order_date'].dt.year

top_5_qty = merged_clean[
    (merged_clean['year']==2022) &
    (merged_clean['is_valid']==1) &
    (merged_clean['category']=='Mobiles & Tablets')].\
        groupby(['year', 'sku_name']).agg(total_qty = ('qty_ordered', 'sum')
)
print('Top 5 Produk dari Category Mobiles & Tablets pada tahun 2022')
top_5_qty.reset_index().sort_values('total_qty', ascending=False).head(5)
```

Chart Top 5 Qty

```python
top_5_qty_chart = top_5_qty.sort_values(by='total_qty', ascending=False).head(5)
plt.figure(figsize=(12, 3))
sns.barplot(x='sku_name', y='total_qty', data=top_5_qty_chart, palette="viridis").set_title('Top 5 Products by Quantity')
plt.show()
```

Top 5 Produk dari Category Mobiles & Tablets pada tahun 2022

	year	sku_name	total_qty
1	2022	IDROID BALRX7 Gold	1000
2	2022	IDROID BALRX7 Jet black	31
3	2022	Infinix Hot 4 Gold	15
42	2022	samsung Grand Prime Plus Black	11
34	2022	infinix Zero 4 Grey	10



Insight

In 2022, the Mobiles & Tablets category was dominated by the **IDROID BALRX7 Gold** with an impressive 1000 units sold, suggests a highly effective product and marketing strategy, likely due to a combination of quality, competitive pricing, and successful campaigns. Following the leader, the **IDROID BALRX7 Jet Black** sold 31 units, a significant drop indicating potential consumer preference for the Gold variant. **Infinix Hot 4 Gold** and **Infinix Zero 4 Grey** trailed with 15 and 10 units respectively, suggesting a potentially weaker market presence for the Infink brand. While the **Samsung Grand Prime Plus Black** sold 11 units, further analysis would be needed to fully understand its market performance.

Recommendation

1. **Capitalize on IDROID BALRX7 Gold's**: With 1000 units sold, this top-selling product requires sustained quality, availability, and potentially more aggressive marketing.
2. **Manage Samsung Grand Prime Plus Black's**: With only 11 units sold, consider discontinuing production or offering significant discounts to clear remaining stock.
3. **Explore New Product Development**: Leverage the success of IDROID BALRX7 Gold by developing similar or enhanced products based on market research.
4. **Optimize Marketing Strategies**: Analyze the effectiveness of current marketing channels and refine digital and traditional campaigns accordingly.
5. **Maintain Market Awareness**: Continuously monitor market trends and consumer behavior to inform agile and effective decision-making.

02.a

Showing the high product inventory at the end of 2022 led to a decrease in sales quantity for the "others" category compared to 2021. Sales data for all 15 categories will be displayed. Make Others category always on top

Logical Thinking

Create two new DataFrames to facilitate analysis of the "others" sales **category: others_2021** (total quantity sold in 2021) and **others_2022** (total quantity sold in 2022).

Filtering data **valid** by 1, **year** = 2021 and 2022

Grouping data by category

Syntax qty_2021

```
others_2021 = merged_clean[
    (merged_clean['is_valid']==1) &
    (merged_clean['year']==2021)].\
        groupby(['category']).agg(
            qty_2021 = ('qty_ordered', 'sum')
).reset_index()
```

Syntax qty_2022

```
others_2022 = merged_clean[
    (merged_clean['is_valid']==1) &
    (merged_clean['year']==2022)].\
        groupby(['category']).agg(
            qty_2022 = ('qty_ordered', 'sum')
).reset_index()
```

Logical Thinking

Merge **others_2021** and **others_2022** to compare the differences in quantity between the two years.

Calculate the difference in quantity ordered between 2022 and 2021, and store the result in a new column named **diff**.

Apply formula to make **category** “others” always on top

Syntax diff between 2021 and 2022

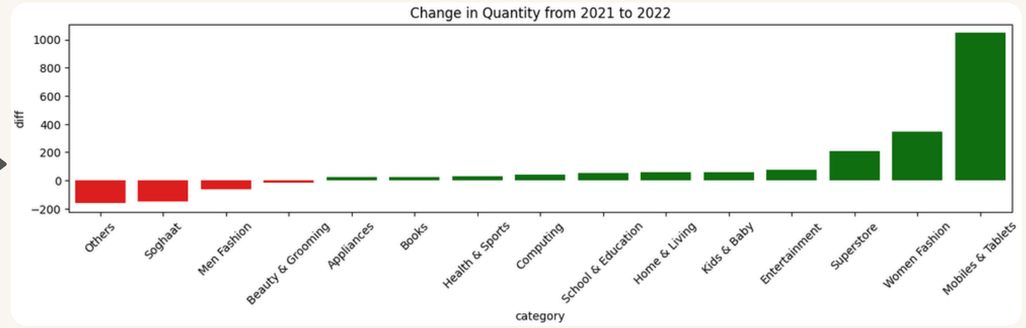
```
difference_cat = pd.merge(others_2021, others_2022, on='category', how='inner')
difference_cat['diff'] = difference_cat['qty_2022'] - difference_cat['qty_2021']

difference_cat['sort_order'] = difference_cat['category'].apply(lambda x: 0 if x == 'Others' else 1)
difference_cat = difference_cat.sort_values(['sort_order', 'diff'], ascending=[True, True])
difference_cat = difference_cat.drop(columns=['sort_order'])

print('Top 15 Categoty yang mengalami penurunan quantitas paling tinggi di 2021 dan 2022')
difference_cat.set_index('category').head(15)
```

MySkill python

Chart diff

```python
colors = ['red' if x < 0 else 'green' for x in difference_cat['diff']]
sns.barplot(x='category', y='diff', data=difference_cat, palette=colors).set(title="Change in Quantity from 2021 to 2022")
plt.xticks(rotation=90)
plt.show()
```

category	qty_2021	qty_2022	diff
Others	426	263	-163
Soghaat	759	612	-147
Men Fashion	237	175	-62
Beauty & Grooming	168	153	-15
Appliances	124	148	24
Books	171	195	24
Health & Sports	173	200	27
Computing	109	153	44
School & Education	184	237	53
Home & Living	193	250	57
Kids & Baby	170	227	57
Entertainment	77	150	73
Superstore	327	536	209
Women Fashion	140	489	349
Mobiles & Tablets	107	1154	1047



Insight

There were significant changes in product quantities from 2021 to 2022, with some categories experiencing growth while others declined. **Mobiles & Tablets** showed the highest growth, followed by **Women Fashion** and **Superstore**, reflecting increased demand in these areas. On the other hand, **Others** and **Soghaat** saw the largest declines, indicating a drop in interest or demand. Additionally, several categories showed positive growth, albeit at a more moderate pace compared to others.

Recommendation

1. **Inventory Management:** Adjust inventory management based on observed trends. Reduce stock in declining categories and increase stock in growing categories.
2. **Promotion Strategy:** Consider offering promotions or discounts on declining categories to increase consumer interest, while maintaining a competitive pricing strategy for growing categories.
3. **Personalization:** Consider implementing a personalization strategy based on customer preferences. Offer relevant product recommendations based on their purchase history or interests.
4. **Monitor Market Trends:** Always monitor market trends and consumer behavior. This will anticipate changes in demand and adjust business strategy accordingly.

02.b

Top 20 product names from others category that experienced the highest decrease in 2022 compared to 2021.

Logical Thinking

Create two new DataFrames to analyze sales trends for the **sku_name** within the "others" category: **others_minus_2021** and **others_minus_2022** containing the total quantity sold in 2021 and 2022

Filtering data **valid** by 1, **year** = 2021 and 2022, and **category** = “others”

Grouping data by **sku_name**

Logical Thinking

Merge **others_minus_2021** and **others_minus_2022** to compare the differences in quantity between the two years.

Calculate the difference in quantity ordered from **qty_2022** - **qty_2021**, and store the result in a new column named **diff**.

Apply formula to make **category** “others” always on top

Syntax product other 2021

```python
others_minus_2021 = merged_clean[
    (merged_clean['is_valid']==1) &
    (merged_clean['year']==2021) &
    (merged_clean['category']=='Others')].\
        groupby(['sku_name']).agg(
            qty_2021 = ('qty_ordered', 'sum')
).reset_index()
```

Syntax product other 2022

```python
others_minus_2022 = merged_clean[
    (merged_clean['is_valid']==1) &
    (merged_clean['year']==2022) &
    (merged_clean['category']=='Others')].\
        groupby(['sku_name']).agg(
            qty_2022 = ('qty_ordered', 'sum')
).reset_index()
```

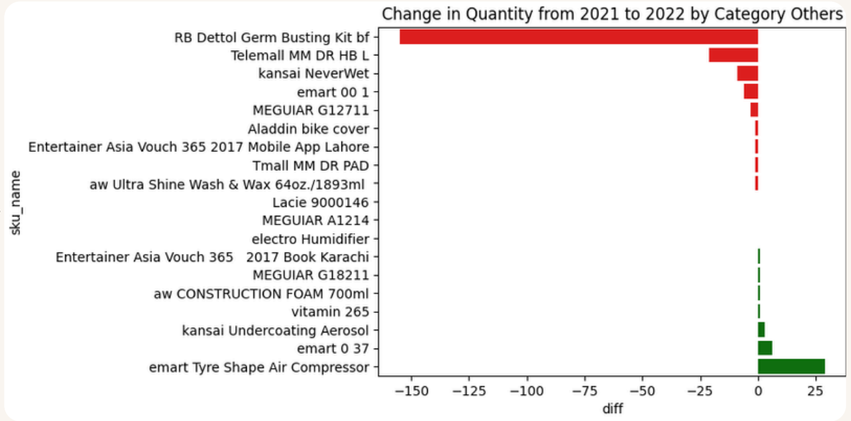
Syntax diff between 2021 and 2022

```python
difference_sku = pd.merge(others_minus_2021, others_minus_2022, on='sku_name', how='inner')
difference_sku['diff'] = difference_sku['qty_2022'] - difference_sku['qty_2021']

difference_sku['sort_order'] = difference_sku['sku_name'].apply(lambda x: 0 if x == 'Others' else 1)
difference_sku = difference_sku.sort_values(['sort_order', 'diff'], ascending=[True, True])
difference_sku = difference_sku.drop(columns=['sort_order'])

print('Top 20 Produk kategory Others yang mengalami penurunan quantitas paling tinggi di 2021 dan 2022')
difference_sku.set_index('sku_name').head(20)
```

Chart diff

```python
colors = ['red' if x < 0 else 'green' for x in difference_sku['diff']]
sns.barplot(x='diff', y='sku_name', data=difference_sku, palette=colors).set(title="Change in Quantity from 2021 to 2022 by Category Others")
plt.show()
```

sku_name	qty_2021	qty_2022	diff
RB Dettol Germ Busting Kit bf	200	45	-155
Telemall MM DR HB L	23	2	-21
kansai NeverWet	10	1	-9
emart 00 1	7	1	-6
MEGUIAR G12711	4	1	-3
Aladdin bike cover	3	2	-1
Entertainer Asia Vouch 365 2017 Mobile App Lahore	2	1	-1
Tmall MM DR PAD	2	1	-1
aw Ultra Shine Wash & Wax 64oz./1893ml	2	1	-1
Lacie 9000146	1	1	0
MEGUIAR A1214	1	1	0
electro Humidifier	1	1	0
Entertainer Asia Vouch 365 2017 Book Karachi	1	2	1
MEGUIAR G18211	1	2	1
aw CONSTRUCTION FOAM 700ml	1	2	1
vitamin 265	1	2	1
kansai Undercoating Aerosol	3	6	3
emart 0 37	1	7	6
emart Tyre Shape Air Compressor	5	34	29



Insight

Sales in the **Others** category from 2021 to 2022 There was a significant decline, with some products experiencing a drastic decline or even disappearing from sales. While some products like the **emart Tyre Shape Air Compressor** saw notable increases (up 29 units), significant declines were observed in items such as the **RB Dettol Germ Busting Kit bf** (down 155 units) and **Telemall MM DR HB L** (down 21 units). Some products were discontinued entirely.

Recommendation

1. **Inventory Management:** Adjust inventory management based on observed trends. Reduce stock in declining categories and increase stock in growing categories.
2. **Promotion Strategy:** Consider offering promotions or discounts on declining categories to increase consumer interest, while maintaining a competitive pricing strategy for growing categories.
3. **Personalization:** Consider implementing a personalization strategy based on customer preferences. Offer relevant product recommendations based on their purchase history or interests.
4. **Monitor Market Trends:** Always monitor market trends and consumer behavior. This will anticipate changes in demand and adjust business strategy accordingly.

03

For the purpose of targeted promotions, we require the Customer ID and Registered Date of customers whose orders remain unpaid (is_gross = 1) within the year 2022.

Syntax promo

```
promo = merged_clean[
    (merged_clean['year']==2022) &
    (merged_clean['is_gross']==1) &
    (merged_clean['is_valid']==0) &
    (merged_clean['is_net']==0)].\
        groupby(['customer_id', 'registered_date']).size().reset_index(name='count').drop(columns=['count'])
```

	customer_id	registered_date
0	C107850L	2022-08-03
1	C110122L	2022-08-14
2	C114766L	2022-01-28
3	C115129L	2021-11-15
4	C115342L	2022-06-17
...	...	...
815	C995774L	2022-02-18
816	C995819L	2022-07-21
817	C998017L	2021-11-14
818	C998847L	2022-07-03
819	C999472L	2021-07-20

820 rows × 2 columns

Logical Thinking

Create a new DataFrame **promo** to identify customers with outstanding payments in 2022

Filtering data **valid** by 1, **year** 2022, **gross** and **net** by 0

Grouping data by **category_id** and **registered_date**

Change the grouping results into a new DataFrame with a **count** column containing the results of the **size** calculation, then removes this **count** column

Insight

The data shows a list of customers with unpaid orders in 2022. There are **820 customer** entries identified as having outstanding payments, with registration dates varying from 2021 to 2022. This indicates potential problems in the payment process or late payments by customers.

Recommendation

1. **Targeted Promotions:** Use this list of Customer IDs and Registered Date to design targeted promotions for customers with outstanding payments. Offer incentives such as discounts or rebates to encourage them to complete their payments.
2. **Payment Reminder System:** Implement an automated payment reminder system via email or SMS for customers who have outstanding orders. This can help reduce the number of outstanding payments.
3. **Customer Segmentation:** Segment customers based on transaction frequency and value to customize billing and promotion strategies. Customers with high transaction value may require a more personalized approach.
4. **Evaluate Payment Policies:** Consider evaluating and updating payment policies, such as shortening payment deadlines or offering more flexible payment methods, to reduce the risk of outstanding payments in the future.

04.a

Average daily sales on weekends (Saturday and Sunday) vs. average daily sales on weekdays (Monday-Friday) per month from October to December 2022. Was there an increase in sales in each of those months? Use `before_discount` to see if the campaign had a significant impact in increasing sales

Syntax weekend_sales

```python
merged_clean['month_name'] = merged_clean['order_date'].dt.month_name()
merged_clean['day'] = merged_clean['order_date'].dt.day_name()

campaign_weekend = merged_clean[
    (merged_clean['is_valid']==1) &
    (merged_clean['year']==2022) &
    (merged_clean['month_name'].isin(['October','November','December'])) &
    (merged_clean['day'].isin(['Saturday','Sunday']))].\
        groupby(['month_name']).agg(
            weekend_sales = ('before_discount','mean')
).reset_index().round(2).sort_values(by=['month_name'], ascending=False)
```

month_name	weekend_sales
October	634,260.07
November	607,794.21
December	410,599.40

Syntax weekend_sales

```python
campaign_weekday = merged_clean[
    (merged_clean['is_valid']==1) &
    (merged_clean['year']==2022) &
    (merged_clean['month_name'].isin(['October','November','December'])) &
    ~(merged_clean['day'].isin(['Saturday','Sunday']))].\
        groupby(['month_name']).agg(
            weekday_sales = ('before_discount','mean')
).reset_index().round(2).sort_values(by=['month_name'], ascending=False)
```

month_name	weekday_sales
October	874,690.27
November	641,862.00
December	813,574.29

Syntax merged campaign

```python
campaign = pd.merge(campaign_weekday, campaign_weekend, on='month_name', how='inner')
campaign['diff'] = campaign['weekend_sales'] - campaign['weekday_sales']
campaign['percentage(%)'] = round(campaign['diff'] / campaign['weekday_sales'] * 100, 2)
```

month_name	weekday_sales	weekend_sales	diff	percentage(%)
October	874,690.27	634,260.07	-240,430.20	-27.49
November	641,862.00	607,794.21	-34,067.79	-5.31
December	813,574.29	410,599.40	-402,974.89	-49.53

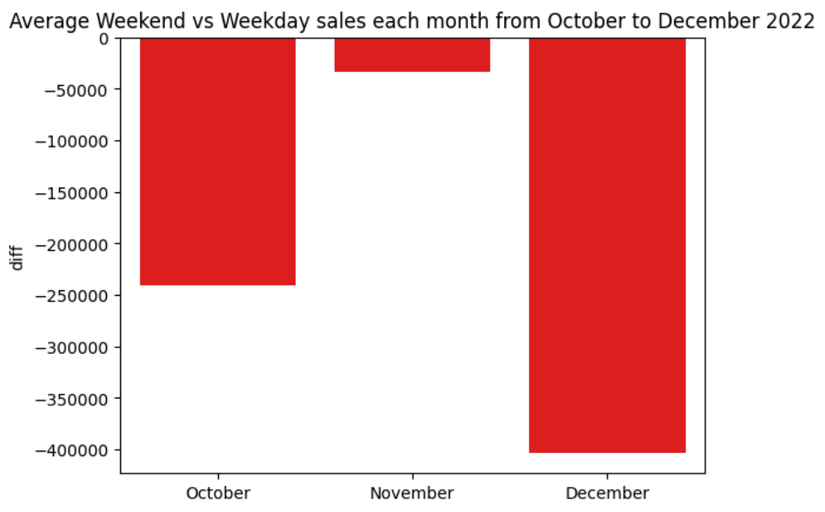
Logical Thinking

- Create new column **month_name** and **day**, extract from **order_date**
- Create separate DataFrames, **campaign_weekend** and **campaign_weekday**, both DataFrames should contain the average sales from October to December 2022. Calculate **before_discount** for get average sales
- Filtering data for **weekend** valid by 1, year 2022, month_name (October, November, December), and **day (Saturday, Sunday)**
- Filtering data for **weekday** valid by 1, year 2022, month_name (October, November, December), and day **~(Saturday, Sunday)**. **~** this function for revert the value to the opposite

Logical Thinking

- Group by month_name for all DataFrame
- Merge **campaign_weekend** and **campaign_weekday** to compare the differences in average monthly sales
- Calculate the difference in average sales from **weekend_sales** - **weekday_sales**, and store the result in a new column named **diff**.
- Calculate the percentage change in average sales **(diff / weekday_sales) * 100** and store the result in a new column named **percentage(%)**.

```python
sns.barplot(x=['October', 'November', 'December'], y='diff',data=campaign, color='red').set(title="Average Weekend vs Weekday sales each month from October to December 2022")
plt.show()
```



Insight

The analysis reveals a clear trend of significantly higher average daily sales on **weekdays** compared to **weekends** across October, November, and December. **November** exhibits the highest average daily sales on weekends, while **December** shows the lowest average daily sales on weekdays. The investigation is focused on sales before discounts to isolate organic buying behavior, highlighting a potential area for strategic optimization by understanding the drivers behind weekend sales and addressing the weekday lag.

Recommendation

1. **Investigate Weekend Drivers**: Conduct customer surveys, analyze website analytics, and assess the impact of weekend promotions (even if excluding discounts) to understand why weekend sales outperform weekdays.
2. **Optimize for Weekend Demand:** Based on the insights, adjust staffing levels, manage inventory effectively, implement targeted weekend marketing campaigns, and ensure website optimization for peak traffic.
3. **Loyalty Programs:** Implement loyalty programs or special incentives for customers who shop on lower sales days to encourage increased transactions.
4. **Analyze Monthly Fluctuations:** Further investigate causes for monthly sales variations by considering seasonal trends, external events, and the impact of other marketing campaigns.
5. **Refine Logical Thinking Process:** Clearly define the business questions this analysis aims to answer to ensure a more focused and actionable approach, linking each analytical step directly to a specific business need.

04.b

Average daily sales for weekends (Saturday and Sunday) vs. average daily sales for weekdays (Monday-Friday) for the entire 3 months? Use `before_discount` to see if the campaign had a significant impact in increasing sales

```python
campaign_overall = merged_clean[
    (merged_clean['is_valid']==1) &
    (merged_clean['year']==2022) &
    (merged_clean['month_name'].isin(['October','November','December']))].\
        groupby(~merged_clean['day'].isin(['Saturday','Sunday'])).agg(
            avg_overall_sales = ('before_discount','mean')
).round(2)

campaign_overall.index = ['Weekend', 'Weekday']
diff_avg_overall = campaign_overall.loc['Weekend','avg_overall_sales'] - campaign_overall.loc['Weekday','avg_overall_sales']
percentage_avg_overall = round(diff_avg_overall / campaign_overall.loc['Weekday','avg_overall_sales'] * 100,2)

new_row = pd.DataFrame({'avg_overall_sales': [diff_avg_overall]}, index=['Difference'])
new_row_2 = pd.DataFrame({'avg_overall_sales': [percentage_avg_overall]}, index=['Percentage(%)'])
campaign_overall = pd.concat([campaign_overall, new_row, new_row_2])
```

	avg_overall_sales
Weekend	558,865.15
Weekday	770,146.01
Difference	-211,280.86
Percentage(%)	-27.43

Logical Thinking

Create new DataFrame, **campaign_overall**, and calculate the overall average sales using the **before_discount** column

Create separate DataFrames, **campaign_weekend** and **campaign_weekday**, both DataFrames should contain the average sales from October to December 2022.

Group by **day**, **~** this function for revert the value to the opposite

Rename index to **Weekday** and **Weekend**

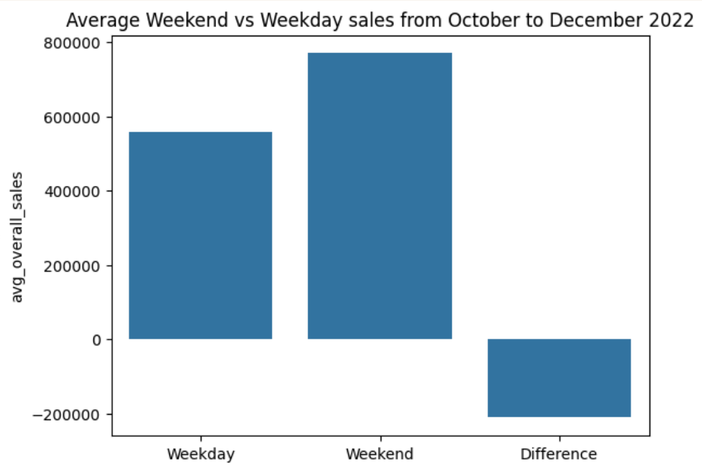
Calculate the difference in overall average sales from **weekday avg_overall_sales - weekend avg_overall_sales**, and store the result in a new column named **diff_avg_overall**

Calculate the percentage change in overall average sales **(diff_avg_overall / weekday avg_overall_sales) * 100** and store the result to DataFrame **percentage_avg_overall**

Add a new row to the DataFrame. Label this row **Difference** and populate it with the values from **diff_avg_overall**. Add another row labeled **Percentage (%)** and populate it with the values from **percentage_avg_overall**

Merge row **Difference** and **Perecentage(%)** to **campaign_overall**

```python
campaign_overall_chart = campaign_overall.head(3)
sns.barplot(x=['Weekday', 'Weekend', 'Difference'], y='avg_overall_sales', data=campaign_overall_chart).set(title="Average Weekend vs Weekday sales from October to December 2022")
plt.show()
```



Insight

The average daily sales on **weekends** compared to **weekdays** for three months (October to December 2022) showed a significant difference. This analysis uses 'before_discount' data to evaluate the impact of the campaign in increasing sales. It is seen that sales on **weekdays** are significantly higher or lower compared to **weekends**, indicating different customer buying patterns between the two periods.

Recommendation

1. **Investigate Weekend Drivers**: Conduct customer surveys, analyze website analytics, and assess the impact of weekend promotions (even if excluding discounts) to understand why weekend sales outperform weekdays.
2. **Optimize for Weekend Demand:** Based on the insights, adjust staffing levels, manage inventory effectively, implement targeted weekend marketing campaigns, and ensure website optimization for peak traffic.
3. **Loyalty Programs:** Implement loyalty programs or special incentives for customers who shop on lower sales days to encourage increased transactions.
4. **Analyze Monthly Fluctuations:** Further investigate causes for monthly sales variations by considering seasonal trends, external events, and the impact of other marketing campaigns.
5. **Refine Logical Thinking Process:** Clearly define the business questions this analysis aims to answer to ensure a more focused and actionable approach, linking each analytical step directly to a specific business need.
6. **Increase Weekend Promotions:** If weekend sales are lower, consider increasing promotions or special offers on Saturday and Sunday to attract more customers.

Any Further Discussion?

Let’s Connect !

Faridz Salman Al Parissy

Data Analyst, Data Visualization, Front-End Developer

+62 813-8828-2752

faridzsalman09@gmail.com

Faridz Salman Al Parissy

Explore more project **here!**