BLINKIT ORDER AND IT'S REPORT

Project submitted to

APSSDC

Bachelor of Technology

In

Computer Science Engineering

Aditya College of Engineering and Technology

Submitted By

KARNAM NAGA VENKATA SAI GANESH - 23MH1A0523
AFRID MOLLA - 23MH1A0536
GANGADHAR VALET - 23MH1A0563
ABHINAV VODURI - 223MH1A0569

Under the guidance of

K.Narmada Mani

K. Veera Vanitha

June 2025

TABLE OF CONTENTS

ABSTRACT

In the rapidly evolving landscape of e-commerce, efficient order fulfillment and customer satisfaction play a critical role in the success of any platform. This project focuses on analyzing Blinkit's order dataset to uncover key insights and patterns related to customer behavior, product demand, delivery efficiency, and feedback trends.

The analysis employs Python's powerful data manipulation and visualization libraries such as Pandas , Matplotlib, Numpy and Seaborn to perform exploratory data analysis (EDA), data cleaning, and insightful visualization.

This project identifies top products, common order types, delivery issues, and customer feedback trends. By analyzing and segmenting the data, it uncovers areas for improvement in operations and strategy. The goal is to help Blinkit enhance customer experience and streamline logistics through data-driven insights.

INTRODUCTION

In today's fast-paced digital economy, data has become one of the most valuable assets for businesses. E-commerce platforms, especially those in the quick-commerce segment like **Blinkit**, generate massive amounts of data every day from orders, payments, customer feedback, and delivery operations. To remain competitive and improve service quality, it is essential for such companies to leverage data analytics for deeper insights and smarter decisions. This project, titled **"Blinkit Order Report Data Analytics using Python"**, aims to explore and analyze transactional data from Blinkit to discover meaningful patterns and trends that can contribute to operational improvement and customer satisfaction.

Blinkit, previously known as Grofers, is an Indian online delivery service that promises to deliver groceries and daily essentials in minutes. With thousands of orders being placed and delivered daily across different regions, analyzing this order data can reveal important business insights such as product popularity, common delivery issues, regional demand trends, and customer preferences. By understanding these patterns, Blinkit can optimize its inventory, enhance delivery routes, improve customer service, and make informed business decisions.

The main objective of this project is to perform **Exploratory Data Analysis (EDA)** on Blinkit's order report using Python and its data science libraries. The key libraries used in this project include:

- **Pandas**: For data manipulation and handling missing values.
- **NumPy**: For performing numerical operations and working with arrays.
- **Matplotlib** and **Seaborn**: For creating meaningful data visualizations like bar charts, pie charts, and line graphs.

The project begins with **data collection and understanding**, followed by **data cleaning** to remove null values, correct data types, and ensure accuracy. After cleaning, we move into the **exploration phase** where we examine the structure of the dataset, look at common order types, payment methods, and delivery statuses. **Grouping and aggregation** techniques are used to identify high-performing products, analyze customer behavior across different regions, and evaluate delivery patterns.

We then use **data visualization** to present these findings in a clear and easy-to-understand format. Charts and graphs play a vital role in interpreting the data and drawing conclusions. For example, pie charts show the distribution of order statuses, bar graphs highlight the most popular products, and line graphs help us understand trends over time.

This analytical process not only supports operational decisions but also prepares the ground for future predictive analytics. The ability to transform raw order data into actionable insights demonstrates the importance of data-driven thinking in modern business.

In conclusion, this project serves as a practical example of how **data analytics tools and techniques** can be applied to real-world datasets to improve business efficiency and customer experience. It also showcases the power of Python in handling, analyzing, and visualizing data in a structured and efficient way.

SYSTEM REQUIREMENTS

Operating System

- Windows 10/11 (64-bit)
- macOS (Monterey or later)
- Linux (Ubuntu 20.04 or later)
- Jupyter Notebooks or IDEs like VS Code can run on all major OS

Programming Language

Python 3.7 or above

Software Requirements

- Python 3.8 or above
- Jupyter Notebook / JupyterLab or any IDE like VS Code, PyCharm
- Required Python libraries
    - pandas, numpy, matplotlib, seaborn, plotly, scipy, pycountry_convert

Hardware Requirements

- Minimum 4 GB RAM (8 GB or more recommended)
- At least 2 GHz Dual-core processor
- 500 MB of free disk space (for datasets and libraries)
- Internet connection (for installing packages and accessing online datasets if needed)

ARCHITECTURE OF THE PROJECT

1. Data Acquisition Layer

Order data was gathered from Blinkit's internal transaction records.
Included fields such as item names, delivery times, order outcomes, and payment types.
Data was received in structured Excel or CSV format for further processing.

2. Data Preparation Layer

Cleaning Tasks: Eliminated blank entries, standardized formats, and corrected inconsistencies.
Transformation: Reformatted date/time columns, renamed headers, and extracted key attributes.
Filtering: Narrowed down records based on specific time periods or delivery categories.

3. Analytical Processing Layer

Utilized **Pandas** and **NumPy** to explore trends and uncover distributions.
Applied **aggregation** methods to find most/least frequent values and delivery patterns.
Calculated basic descriptive statistics to understand customer and order behavior.

4. Visualization & Exploration Layer

Created visual summaries using **Seaborn** and **Matplotlib** for clarity.
Displayed data through charts like pie graphs, bar plots, and timelines.
Used color and layout to highlight patterns in order types and regional deliveries.

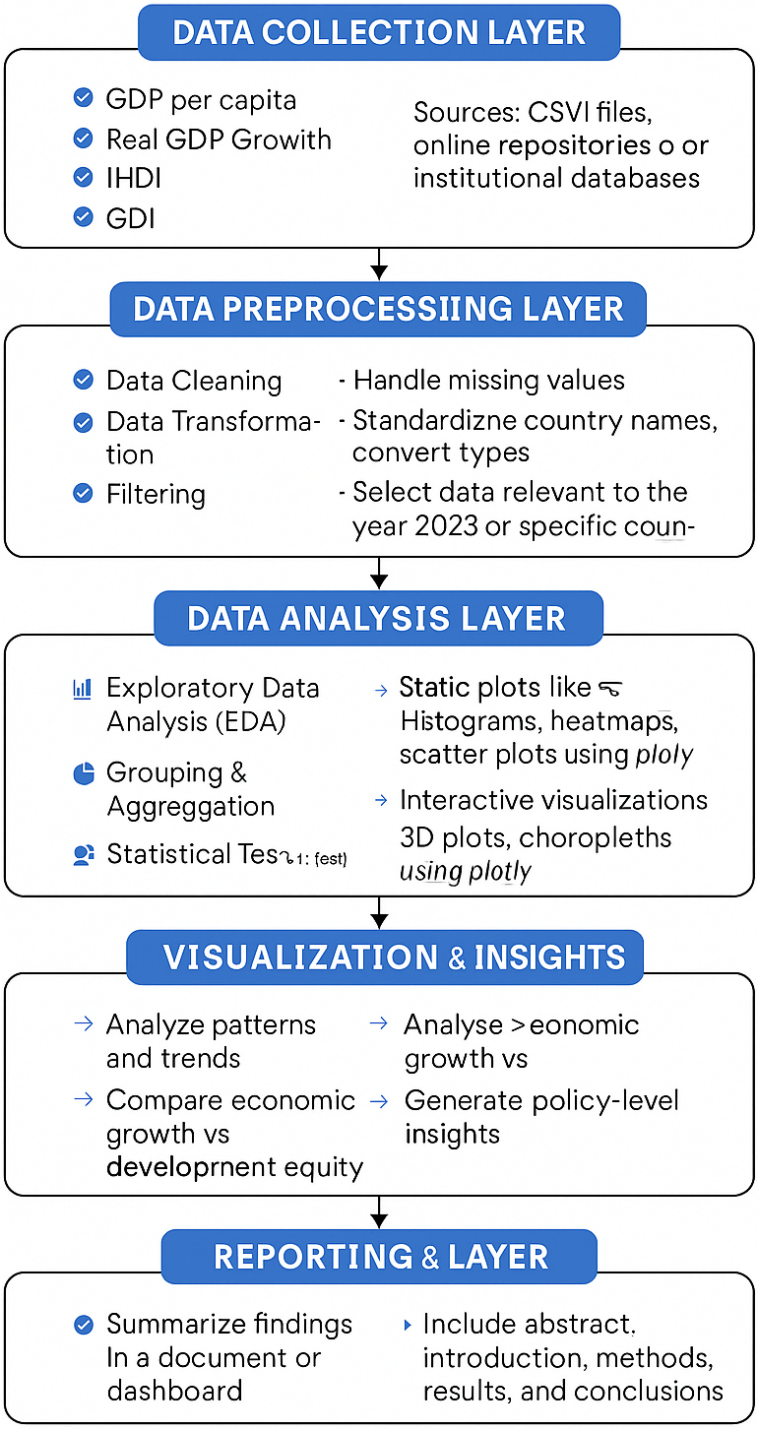
5. Insight Generation Layer

Extracted trends in product demand, payment preferences, and delay frequency.
Compared performance across different regions or delivery methods.
Formulated suggestions to enhance delivery speed and customer satisfaction.

6. Documentation & Summary Layer

Compiled findings into a structured report with visuals.
Highlighted key takeaways and business opportunities.
Outlined actionable improvements for Blinkit's operations team.

FLOWCHART OF THE ARCHITECTURE



USES OF DATA ANALYSIS LIBRARIES

1. **Pandas**

- Data manipulation: Read, write, clean, and reshape data.
- DataFrame operations: Handle tabular data easily (filtering, grouping, merging).
- Handling missing values: Fill, drop, or interpolate nulls.

2. NumPy

- Efficient numerical computations: Fast array and matrix operations.
- Mathematical functions: Supports linear algebra, statistical operations, and Fourier transforms.
- Foundation for other libraries: Many libraries like Pandas, Scikit-learn, etc., rely on NumPy arrays.

3. Matplotlib

- Data visualization: Line plots, bar charts, histograms, scatter plots, etc.
- Customization: Full control over plot styles, axes, labels, and legends.
- Static image generation: Useful for reports and publications.

4. Seaborn

- Statistical visualization: Correlation heatmaps, violin plots, boxplots, etc.
- Integration with Pandas: Works seamlessly with DataFrames.
- Attractive default styles: Better aesthetics than raw Matplotlib.

5. Plotly

- Interactive visualizations: 3D plots, geographic maps, and interactive dashboards.
- Web compatibility: Plots can be embedded in web pages or Jupyter notebooks.
- Hover & zoom features: Ideal for exploratory data analysis.

6. pycountry-convert

- Geographic classification: Convert country names to continent codes or ISO formats.
- Supports mapping: Useful for regional or global-level comparisons.

7. SciPy (stats module)

- Statistical analysis: t-tests, correlation, regression, and more.
- Scientific computing: Used in combination with NumPy for advanced mathematical functions.

PROJECT CODE

Importing libraries

Importing necessary libraries

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

Reading Dataset

The Blinkit order report dataset was imported using **Pandas' read_csv()** function, allowing the data to be loaded into a structured **DataFrame**. This step enabled efficient handling of order records and prepared the dataset for further steps like

Reading dataset

```
df = pd.read_csv('Blinkit_Sheet2.csv')
df
```

cleaning, transformation, analysis, and visualization.

Basic Functions :

```
df.isnull().sum() # gives the sum(no.of) of null values present in the each column
```

```
customer_id          0
customer_name        0
user_status          0
area                 0
order_date           0
product_id           0
product_name         0
category             0
shelf_life_days      0
campaign_name        0
brand                0
unit_price           0
quantity             0
order_total          0
delivery_status      0
payment_method       0
feedback_category    0
dtype: int64
```

```
n3=int(input("Enter a no.of rows you want:"))
df.sample(n3)  # gives n3 no.of random rows from the dataset
```

	customer_id	customer_name	user_status	area	order_date	product_id	product_name	category	shelf_life_days	campaign_name	brand	unit_price
3395	84456355	Pushti Dhillon	Inactive	Gandhidham	9/16/2024 1:11	378287	Toothpaste	Personal Care	365	Festival Offer	Naidu PLC	332.84
1384	98691989	Harita Bir	Premium	Berhampur	6/10/2024 7:16	930284	Bread	Dairy & Breakfast	7	Email Campaign	Aggarwal Group	189.47
3936	99231937	Gautami Dar	Inactive	Kirari Suleman Nagar	3/8/2024 13:40	425341	Shampoo	Personal Care	365	Flash Sale	Sant-Mahajan	486.28
2398	90621147	Rushil Gupta	All	Bijapur	7/5/2023 23:53	435929	Vitamins	Pharmacy	365	Membership Drive	Parsa-Bala	856.89
964	93507573	Vritti Halder	All	Bangalore	3/27/2024 12:45	704868	Detergent	Household Care	365	Weekend Special	Lalla-Srinivasan	193.46
803	25526479	Gauri Ramakrishnan	Premium	Anantapur	4/11/2023 6:16	34186	Vitamins	Pharmacy	365	Festival Offer	Kara-Golla	925.65

```
df.shape # gives the shape of dataset
```

```
(5000, 17)
```

```
len(df)+1 # gives no.of rows present in the dataset
```

```
5001
```

```
df['user_status'].unique().tolist() # gives the unique elements in the column user_status as a list of elements
```

```
['Premium', 'Inactive', 'New Users', 'All']
```

```
df.columns # gives all columns nmes
```

```
Index(['customer_id', 'customer_name', 'user_status', 'area', 'order_date',
       'product_id', 'product_name', 'category', 'shelf_life_days',
       'campaign_name', 'brand', 'unit_price', 'quantity', 'order_total',
       'delivery_status', 'payment_method', 'feedback_category'],
      dtype='object')
```

```
df.info() # gives information of the dataset
```

```
<class 'pandas.core.frame.DataFrame'>
RangeIndex: 5000 entries, 0 to 4999
Data columns (total 17 columns):
 #   Column             Non-Null Count  Dtype  
---  ------             --------------  -----  
 0   customer_id        5000 non-null   int64  
 1   customer_name      5000 non-null   object 
 2   user_status        5000 non-null   object 
 3   area               5000 non-null   object 
 4   order_date         5000 non-null   object 
 5   product_id         5000 non-null   int64  
 6   product_name       5000 non-null   object 
 7   category           5000 non-null   object 
 8   shelf_life_days    5000 non-null   int64  
 9   campaign_name      5000 non-null   object 
 10  brand              5000 non-null   object 
 11  unit_price         5000 non-null   float64
 12  quantity           5000 non-null   int64  
 13  order_total        5000 non-null   float64
 14  delivery_status    5000 non-null   object 
 15  payment_method     5000 non-null   object 
 16  feedback_category  5000 non-null   object 
dtypes: float64(2), int64(4), object(11)
memory usage: 664.2+ KB
```

```
df['user_status'].unique().tolist() # gives the unique elements in the column user_status as a list of elements
```

```
['Premium', 'Inactive', 'New Users', 'All']
```

```
df['order_total'].skew() # gives the skewness of the column order_total
```

```
0.36071583930198053
```

```
df.dtypes # gives datatype of each columns
```

```
customer_id          int64
customer_name       object
user_status         object
area                object
order_date          object
product_id           int64
product_name        object
category            object
shelf_life_days      int64
campaign_name       object
brand               object
unit_price         float64
quantity             int64
order_total        float64
delivery_status     object
payment_method      object
feedback_category   object
```

```
n2=int(input("Enter a no.of rows you want :"))
df.tail(n2)  # gives last n2 rows of the datset
```

```
Enter a no.of rows you want : 5
```

	customer_id	customer_name	user_status	area	order_date	product_id	product_name	category	shelf_life_days	campaign_name	brand	unit_price	quanti
4995	96001315	Daksh Mandal	Inactive	Mumbai	12/25/2023 15:46	925482	Dish Soap	Household Care	365	New User Discount	Khalsa-Bhavsar	475.04	
4996	62419778	Lavanya Jain	All	Udupi	11/27/2023 9:18	124290	Detergent	Household Care	365	Weekend Special	Dutta-Halder	39.65	
4997	60565603	Umang Dash	All	Kavali	6/21/2024 19:09	491314	Toilet Cleaner	Household Care	365	Festival Offer	Batta-Doctor	973.44	
4998	12455156	Zinal Natarajan	Premium	Alwar	6/6/2024 14:58	319388	Dish Soap	Household Care	365	Flash Sale	Hans Inc	657.80	
4999	38478423	Eiravati Sundaram	New Users	Dewas	8/23/2023 12:04	136533	Detergent	Household Care	365	Membership Drive	Edwin-Lall	826.21	

```
RangeIndex: 5000 entries, 0 to 4999
Data columns (total 17 columns):
 #   Column              Non-Null Count  Dtype
---  ------              --------------  -----
 0   customer_id         5000 non-null   int64
 1   customer_name       5000 non-null   object
 2   user_status         5000 non-null   object
 3   area                5000 non-null   object
 4   order_date          5000 non-null   object
 5   product_id          5000 non-null   int64
 6   product_name        5000 non-null   object
 7   category            5000 non-null   object
 8   shelf_life_days     5000 non-null   int64
 9   campaign_name       5000 non-null   object
 10  brand               5000 non-null   object
 11  unit_price          5000 non-null   float64
 12  quantity            5000 non-null   int64
 13  order_total         5000 non-null   float64
 14  delivery_status     5000 non-null   object
 15  payment_method      5000 non-null   object
 16  feedback_category   5000 non-null   object
dtypes: float64(2), int64(4), object(11)
memory usage: 664.2+ KB
```

```
df.isnull() # shows where the null value is present
```

	customer_id	customer_name	user_status	area	order_date	product_id	product_name	category	shelf_life_days	campaign_name	brand	unit_price	quantity	or
0	False	False	False	False	False	False	False	False	False	False	False	False	False	
1	False	False	False	False	False	False	False	False	False	False	False	False	False	
2	False	False	False	False	False	False	False	False	False	False	False	False	False	
3	False	False	False	False	False	False	False	False	False	False	False	False	False	
4	False	False	False	False	False	False	False	False	False	False	False	False	False	
...	...	...	...	...	...	...	...	...	...	...	...	...	...	
4995	False	False	False	False	False	False	False	False	False	False	False	False	False	
4996	False	False	False	False	False	False	False	False	False	False	False	False	False	
4997	False	False	False	False	False	False	False	False	False	False	False	False	False	
4998	False	False	False	False	False	False	False	False	False	False	False	False	False	
4999	False	False	False	False	False	False	False	False	False	False	False	False	False	

5000 rows × 17 columns

Data Cleaning :

```
df.duplicated() # gives the status of the duplicate rows in the dataset
```

```
0       False
1       False
2       False
3       False
4       False
        ...
4995    False
4996    False
4997    False
4998    False
4999    False
Length: 5000, dtype: bool
```

```
df['product_name'].mode()[0] # gives the product which was purchased max no.of times
```

```
'Pet Treats'
```

```
# drops the duplicate rows from the dataset
before = len(df)
df.drop_duplicates(inplace=True)
after = len(df)
if before > after:
    print(f"{before - after} duplicate rows removed.")
else:
    print("No duplicate rows found. No changes made.")
```

```
df['order_total'].head()
```

```
0    3197.07
1     976.55
2     839.05
3     440.23
4    2526.68
Name: order_total, dtype: float64
```

```
# replaces all the values in the 'order_total' column which are less than 2000
df.loc[df['order_total'] < 2000, 'order_total'] = np.nan
df['order_total'].head()
```

```
0    3197.07
1        NaN
2        NaN
3        NaN
4    2526.68
Name: order_total, dtype: float64
```

```
before = len(df)            # removes entire row if any null value found
df.dropna(inplace=True)
after = len(df)
if before > after:
    print(f"{before - after} row(s) with null values dropped.")
else:
    print(" No null values found. No changes made.")
```

```
No null values found. No changes made.
```

Data Filtering :

```
df.columns
```

```
Index(['customer_id', 'customer_name', 'user_status', 'area', 'order_date',
       'product_id', 'product_name', 'category', 'shelf_life_days',
       'campaign_name', 'brand', 'unit_price', 'quantity', 'order_total',
       'delivery_status', 'payment_method', 'feedback_category'],
      dtype='object')
```

```
# replaces all the '_' in the columnswith empty space
df.columns=df.columns.str.strip().str.title().str.replace('_',' ')
df.columns
```

```
Index(['Customer Id', 'Customer Name', 'User Status', 'Area', 'Order Date',
       'Product Id', 'Product Name', 'Category', 'Shelf Life Days',
       'Campaign Name', 'Brand', 'Unit Price', 'Quantity', 'Order Total',
       'Delivery Status', 'Payment Method', 'Feedback Category'],
      dtype='object')
```

```
df
```

ıct_id	product_name	category	shelf_life_days	campaign_name	brand	unit_price	quantity	order_total	delivery_status	payment_method	feedback_category
42612	Onions	Fruits & Vegetables	3	New User Discount	Aurora LLC	517.03	3	3197.07	On Time	Cash	Delivery
78676	Potatoes	Fruits & Vegetables	3	Weekend Special	Ramaswamy-Tata	881.42	1	976.55	On Time	Cash	App Experience
41341	Potatoes	Fruits & Vegetables	3	Festival Offer	Chadha and Sons	923.84	2	839.05	On Time	UPI	App Experience
51860	Tomatoes	Fruits & Vegetables	3	Flash Sale	Barad and Sons	874.78	1	440.23	On Time	Card	App Experience
02241	Onions	Fruits & Vegetables	3	Membership Drive	Sangha, Nagar and Varty	976.55	2	2526.68	On Time	Cash	Delivery

```
df.drop(['order_total'], axis=1, inplace=True)
df
```

r_date	product_id	product_name	category	shelf_life_days	campaign_name	brand	unit_price	quantity	delivery_status	payment_method	feedback_category
/2024 8:34	642612	Onions	Fruits & Vegetables	3	New User Discount	Aurora LLC	517.03	3	On Time	Cash	Delivery
/2024 13:14	378676	Potatoes	Fruits & Vegetables	3	Weekend Special	Ramaswamy-Tata	881.42	1	On Time	Cash	App Experience
/2024 13:07	741341	Potatoes	Fruits & Vegetables	3	Festival Offer	Chadha and Sons	923.84	2	On Time	UPI	App Experience

```
# returns all the rows which has given starting letters
name=input("Enter the customer starts with:")
df[df['customer_name'].str.startswith(name)]
```

```
Enter the customer starts with: Oviya
```

	customer_id	customer_name	user_status	area	order_date	product_id	product_name	category	shelf_life_days	campaign_name	brand	unit_price
15	12059049	Oviya Mitra	Inactive	Mira-Bhayandar	2/3/2024 8:50	455755	Onions	Fruits & Vegetables	3	App Push Notification	Ganesan Ltd	44.09
123	8955057	Oviya Luthra	New Users	Kadapa	6/24/2024 5:17	118820	Frozen Biryani	Instant & Frozen Food	180	App Push Notification	Sabharwal-Mital	12.32
1761	38902193	Oviya Rau	Premium	Shivpuri	4/4/2023 9:05	398340	Dish Soap	Household Care	365	App Push Notification	Ramesh Inc	651.93
1979	40587811	Oviya Bhandari	Premium	Chapra	3/1/2024 21:18	146840	Orange Juice	Cold Drinks & Juices	180	Referral Program	Toor-Nagar	786.85
		Oviya			9/10/2024					App Push	Prakash,	

```
n9=int(input("enter no.of rows you want:"))
df.nsmallest(n9, 'unit_price')   #    fetches top n9 rows with lowest order_total
```

```
enter no.of rows you want: 5
```

order_date	product_id	product_name	category	shelf_life_days	campaign_name	brand	unit_price	quantity	order_total	delivery_status	payment_method	feed
6/24/2024 5:17	118820	Frozen Biryani	Instant & Frozen Food	180	App Push Notification	Sabharwal-Mital	12.32	3	2487.83	On Time	Wallet	
1/25/2024 0:04	118820	Detergent	Household Care	365	Weekend Special	Dutta-Halder	12.32	2	3603.94	On Time	Card	
10/24/2024 13:19	118820	Wheat Flour	Grocery & Staples	365	Festival Offer	Nadig Ltd	12.32	2	1869.79	On Time	Cash	Cu
8/18/2023 11:32	118820	Lotion	Personal Care	365	New User Discount	Dalal-Shan	12.32	3	4205.08	On Time	Card	
6/12/2023 14:45	118820	Baby Food	Baby Care	365	Category Promotion	Karnik PLC	12.32	1	2144.77	On Time	Cash	

```
n8=int(input("enter no.of rows you want:"))
df.nlargest(n8, 'order_total')  # fetches top n8 rows with highest order_total
```

```
enter no.of rows you want: 5
```

	area	order_date	product_id	product_name	category	shelf_life_days	campaign_name	brand	unit_price	quantity	order_total	delivery_status	payment
	Muzaffarpur	5/14/2023 15:42	600767	Orange Juice	Cold Drinks & Juices	180	Referral Program	Murthy-Kaul	722.42	3	6721.46	Slightly Delayed	
	Kottayam	7/12/2024 13:13	961614	Cough Syrup	Pharmacy	365	Flash Sale	Balan-Madan	227.22	1	6543.19	Slightly Delayed	
	Raurkela Industrial Township	8/31/2023 1:45	15314	Vitamins	Pharmacy	365	Referral Program	Shah and Sons	578.85	2	6458.90	Slightly Delayed	
	Karawal Nagar	8/20/2024 7:33	903336	Rice	Grocery & Staples	365	Email Campaign	Ramanathan, Agarwal and Baria	551.88	3	6173.45	Slightly Delayed	
	Hapur	5/3/2023 12:11	762527	Baby Wipes	Baby Care	365	Festival Offer	Dora-Pillai	690.08	1	6161.48	On Time	

Data Grouping :

```
df['user_status'].value_counts()
```

```
user_status
New Users    1270
Inactive     1251
Premium      1244
All          1235
Name: count, dtype: int64
```

```
# Total quatity per category
df.groupby('category')['quantity'].sum().sort_values(ascending=False)
```

```
category
Dairy & Breakfast        1149
Snacks & Munchies        1049
Household Care           1029
Fruits & Vegetables      1026
Pharmacy                  927
Personal Care             901
Grocery & Staples         894
Pet Care                  894
Cold Drinks & Juices      859
Instant & Frozen Food     754
Baby Care                 552
Name: quantity, dtype: int64
```

```python
# groupby().nunique() to get unique brands per category
df.groupby('category')['brand'].nunique()
```

```
category
Baby Care                16
Cold Drinks & Juices     22
Dairy & Breakfast        30
Fruits & Vegetables      27
Grocery & Staples        24
Household Care           27
Instant & Frozen Food    20
Personal Care            25
Pet Care                 25
Pharmacy                 25
Snacks & Munchies        27
Name: brand, dtype: int64
```

```python
# groupby() + idxmax() — get index of top spender per area
df.groupby('area')['order_total'].idxmax()
```

```
area
Adoni            2250
Agartala         3454
Agra             3802
Ahmedabad         903
Ahmednagar       1584
                 ...
Vijayanagaram    2882
Vijayawada       3502
Visakhapatnam    4680
Warangal         3048
Yamunanagar      2765
Name: order_total, Length: 316, dtype: int64
```

Data Sorting :

```
# Sort based on value_counts of a column (area frequency)
df['area'].value_counts().sort_values(ascending=False)
```

```
area
Jalna          36
Orai           34
Bathinda       34
Deoghar        34
Ratlam         32
               ..
Raichur         6
Muzaffarpur     6
Gopalpur        6
Phusro          4
Mangalore       2
Name: count, Length: 316, dtype: int64
```

```
#  groupby().sum().sort_values() – total order_total by area, descending
df.groupby('area')['order_total'].sum().sort_values(ascending=False)
```

```
area
Bathinda     85997.77
Ghaziabad    84547.87
Deoghar      79993.59
Ratlam       77475.44
Jalna        76449.23
               ...
Gopalpur     10070.17
Phusro        8932.43
Kurnool       8421.80
Bharatpur     6192.85
Mangalore     2802.77
Name: order_total, Length: 316, dtype: float64
```

```
# Sort product_id and keep top 3 unique after dropping duplicates of product_id
df.drop_duplicates('product_id').sort_values(by='product_id').head(3)
```

area	order_date	product_id	product_name	category	shelf_life_days	campaign_name	brand	unit_price	quantity	order_total	delivery_status	payment_metho
Chennai	9/5/2023 10:02	4452	Pet Treats	Pet Care	365	Flash Sale	Dyal, Saraf and Goel	169.88	1	1146.95	On Time	Car
zaribagh	7/17/2024 18:55	6405	Orange Juice	Cold Drinks & Juices	180	Weekend Special	Baral-Kamdar	236.51	3	1164.09	On Time	UI
Srinagar	10/12/2023 14:44	9436	Pet Treats	Pet Care	365	Referral Program	Narayanan-Kumar	88.43	2	1264.98	On Time	Walle

```
#  Multi-column sorting: by category (A-Z), then price (high to low)
df.sort_values(by=['category', 'unit_price'], ascending=[True, False])
```

uct_id	product_name	category	shelf_life_days	campaign_name	brand	unit_price	quantity	order_total	delivery_status	payment_method	feedback_category
'39534	Baby Food	Baby Care	365	Flash Sale	Srinivas PLC	994.56	1	3069.89	On Time	UPI	Delivery
'39534	Diapers	Baby Care	365	Referral Program	Kurian, Patla and Sanghvi	994.56	3	2575.94	On Time	Cash	Delivery
183013	Baby Food	Baby Care	365	Membership Drive	Mallick PLC	989.88	1	3710.55	On Time	Card	App Experience
183013	Baby Food	Baby Care	365	Email Campaign	Srinivas PLC	989.88	1	697.53	On Time	UPI	Customer Service
105754	Baby Wipes	Baby Care	365	App Push Notification	Narula Group	979.99	3	3453.62	On Time	Wallet	Customer Service
...	...	...	...	...	...	...	...	...	...	...	...

Aggregation :

```
# Lambda used as mode
df['quantity'].agg(['sum','mean','median',lambda x: x.mode().iloc[0],'min','max','var','std'])
```

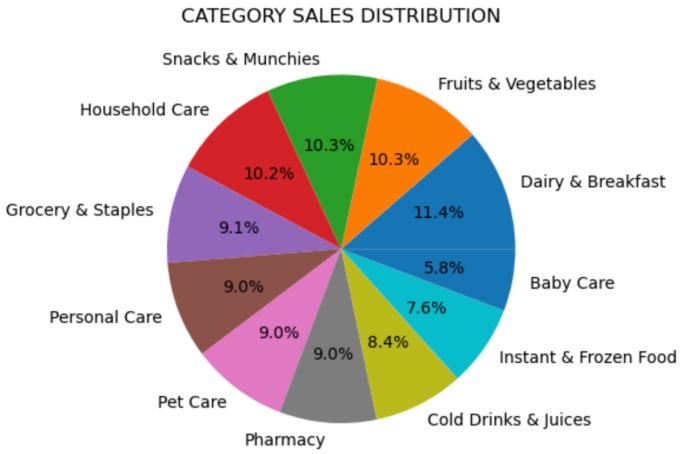
```
sum         10034.000000
mean            2.006800
median          2.000000
<lambda>        3.000000
min             1.000000
max             3.000000
var             0.673288
std             0.820542
Name: quantity, dtype: float64
```

Data Visualization:

Visualizations

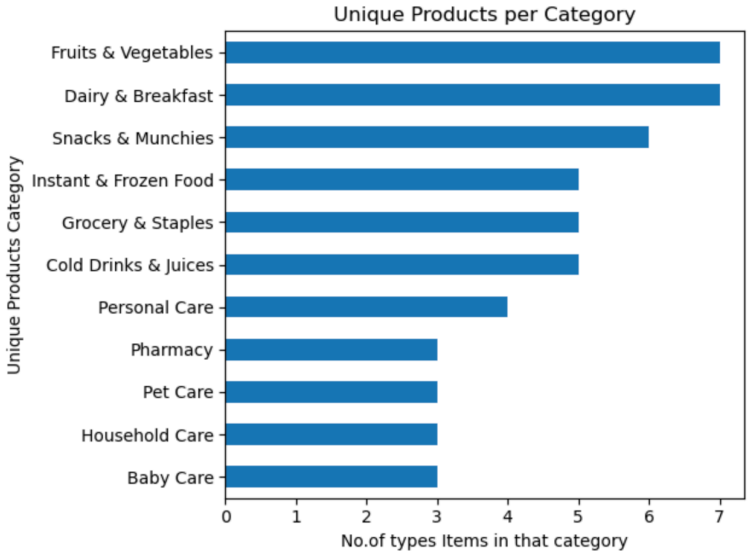
1.) MOST SOLDOUT ITEM ANALYSIS

[162]:
```
# CATEGORY SALES DISTRIBUTION
counts=df['category'].value_counts()
plt.pie(counts,labels=counts.index,autopct='%1.1f%%')
plt.title('CATEGORY SALES DISTRIBUTION')
plt.show()
```



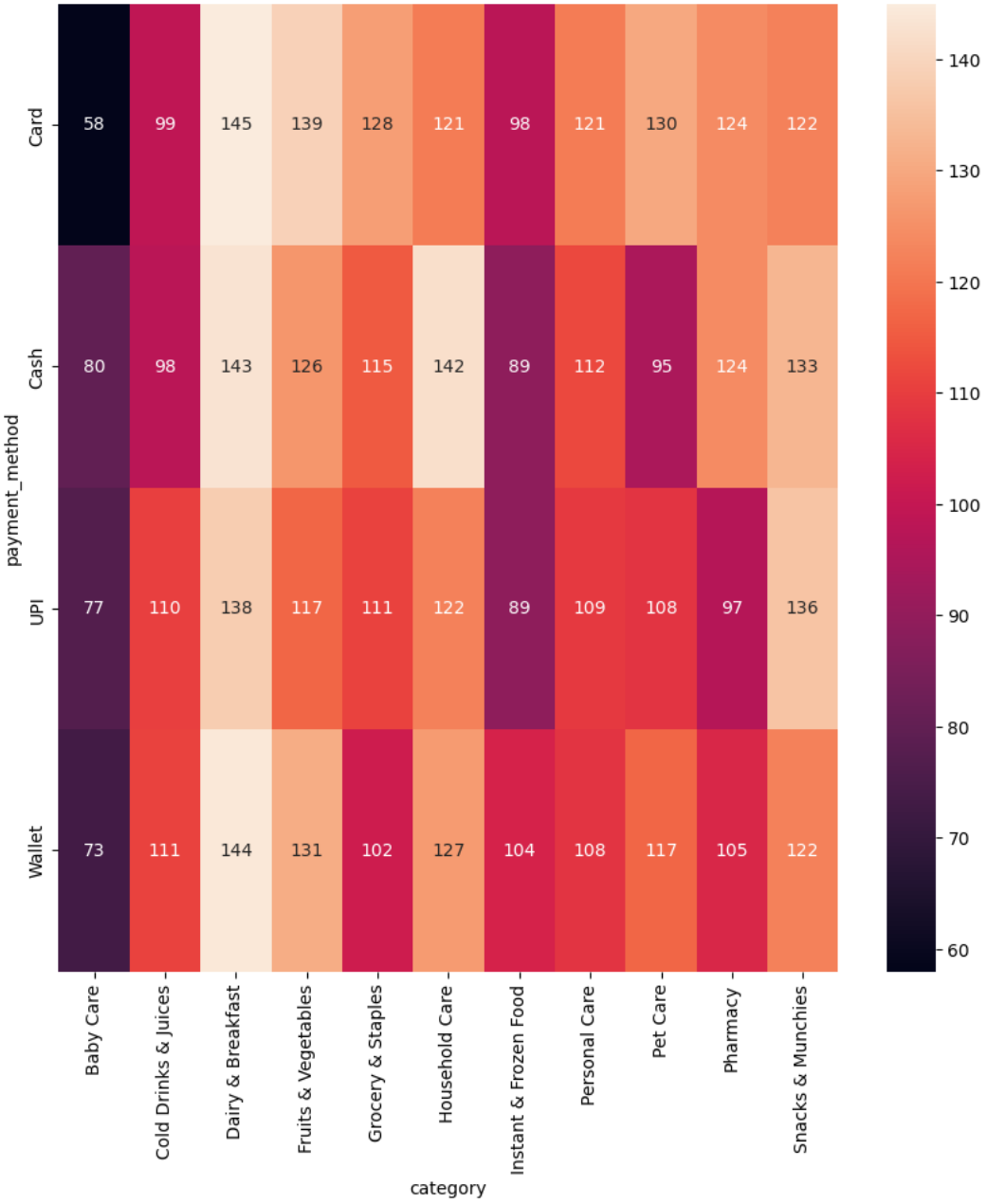
3.) NO.OF PRODUCTS IN EACH CATEGORY

[174]:
```
# PRODUCT'S SALES DISTRIBUTION AND ANALYSIS
df.groupby('category')['product_name'].nunique().sort_values().plot(kind='barh', title='Unique Products per Category')
plt.ylabel('Unique Products Category')
plt.xlabel('No.of types Items in that category')
plt.tight_layout()
plt.show()
```



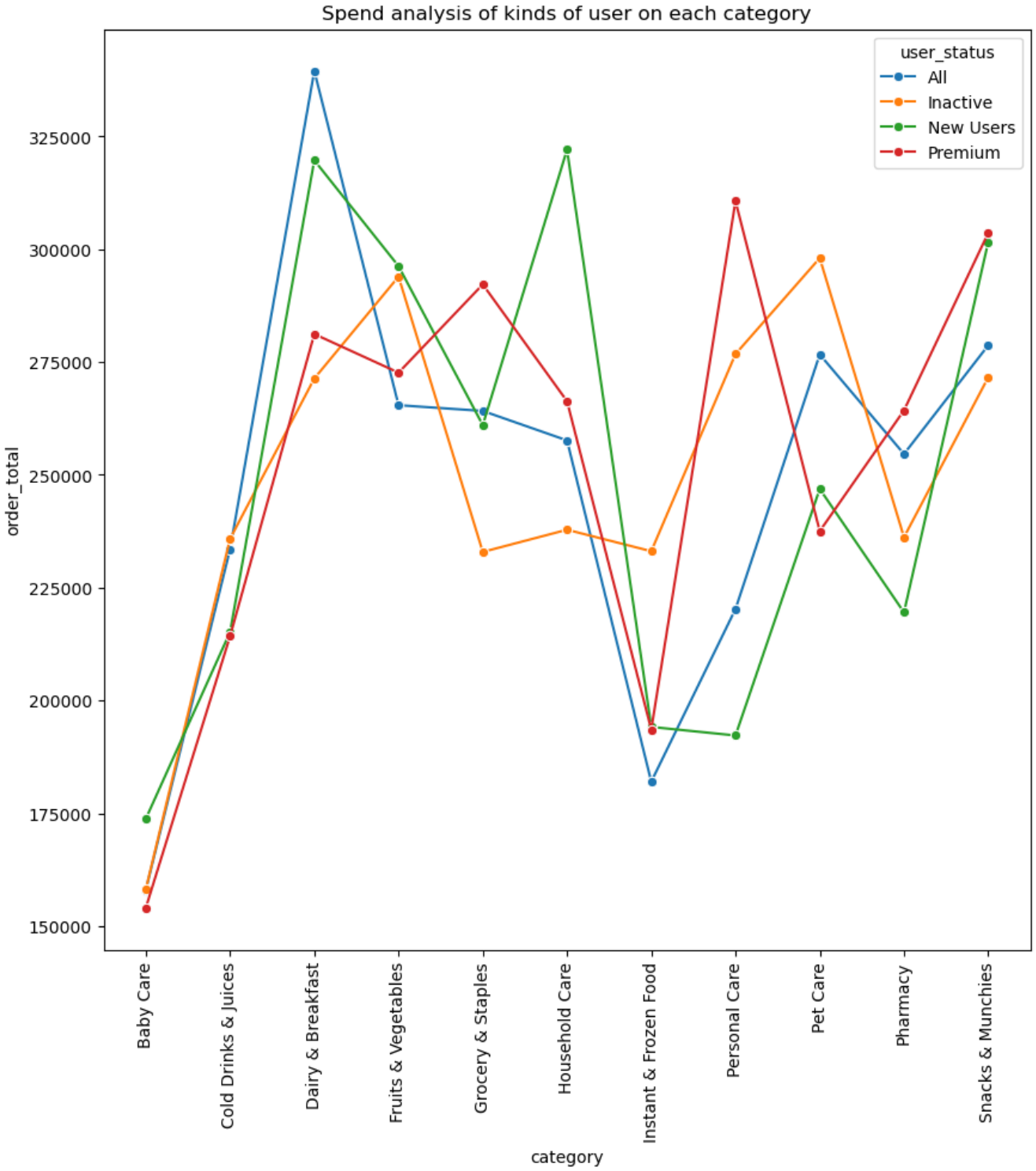
6.) Number of times each payment method done on each category analysis

[208]:
```
# correlation between payment_method and product name
data = df.pivot_table(index='payment_method', columns='category', aggfunc='size', fill_value=0)
plt.figure(figsize=(10, 10))
sns.heatmap(data, annot=True,fmt='d')
plt.show()
```



7.) Total spent analysis of each kind of user(all,inactive,newuser,premium) on each category

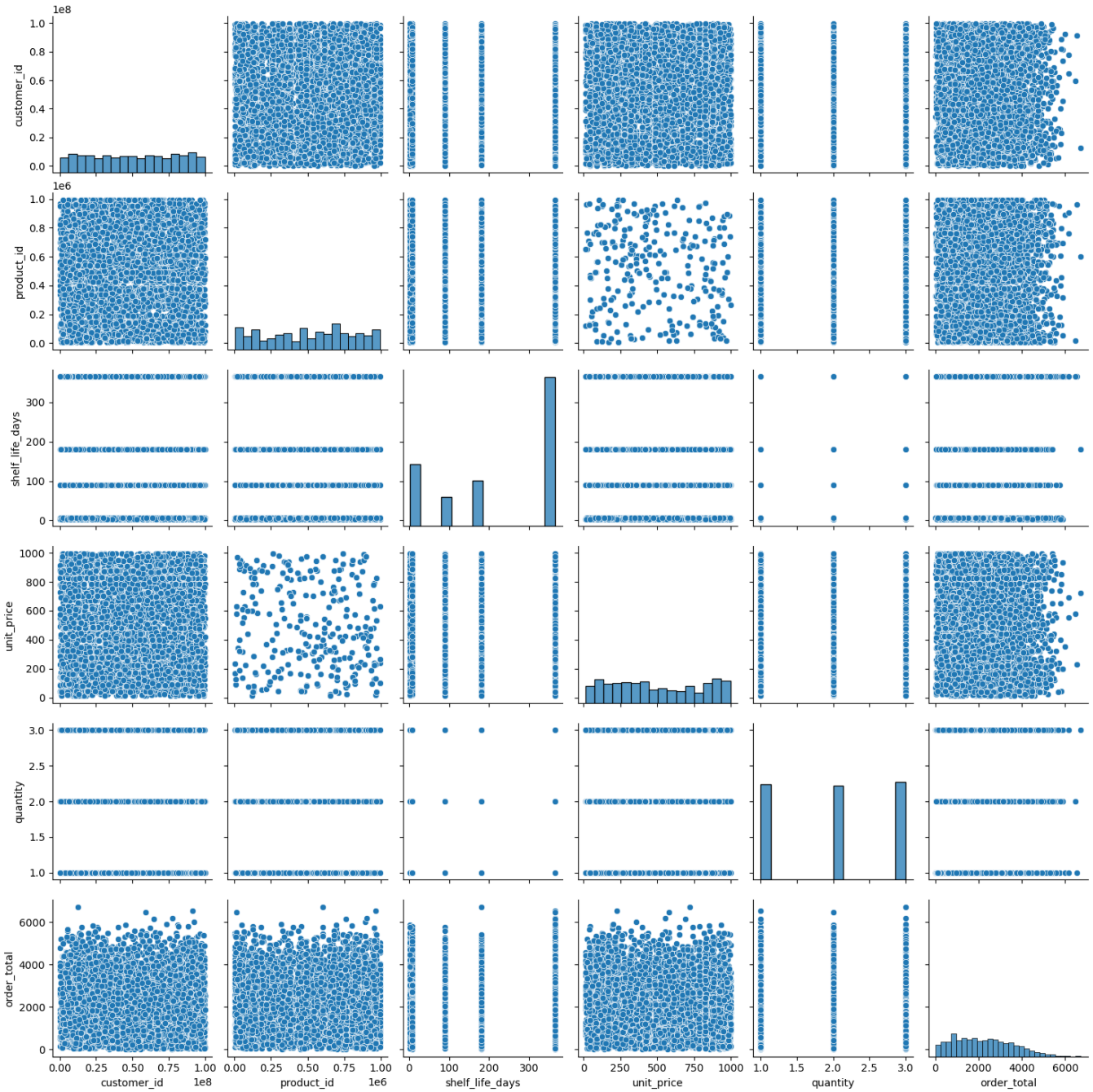
211]:
```
# relation between user_status , product category and their order_total
data = df.groupby(['category', 'user_status'])['order_total'].sum().reset_index()
plt.figure(figsize=(10,10))
plt.title("Spend analysis of kinds of user on each category")
sns.lineplot(x='category', y='order_total', hue='user_status', data=data,marker='o')
plt.xticks(rotation=90)
plt.show()
```



9.) Scatterplot between the every column of the dataset

[424]:
```
sns.pairplot(df)
```

[424]:
```
<seaborn.axisgrid.PairGrid at 0x1d4e86ef9e0>
```



Advantages :

1. **Better Decision Making**
   The analysis provides real insights to support Blinkit's business strategies.
   It helps in making informed, data-driven decisions.

2. **Understanding Customer Preferences**
   Reveals most used payment methods and popular products.
   Useful for improving user experience and targeting offers.

3. **Improved Delivery Performance**
   Highlights areas with frequent delivery delays or issues.
   Helps optimize routes and reduce customer complaints.

4. **Clear Visual Interpretation**
   Uses graphs and charts to make complex data understandable.
   Aids in presenting insights to teams and stakeholders.

5. **Support for Inventory Management**
   Tracks high-demand items and order trends over time.
   Improves stock planning and reduces overstock or shortages.
6. **Data Cleaning Practice**
   Improves skills in handling real-world messy data.
   Ensures accurate and reliable analysis results.

7. **Performance Monitoring**
   Tracks order volume, status, and delivery times over time.
   Helps evaluate business growth and service quality.

8. **Customer Feedback Insights**
   Analyzes feedback to understand user satisfaction and pain points.
   Supports improvements in customer service.

9. **Area-wise Demand Analysis**
   Finds out which locations have higher or lower order rates.
   Useful for region-specific planning and marketing.

10. **Foundation for Advanced Analytics**
    Sets the stage for predictive modeling and trend forecasting.
    Can be extended into deeper analytics in future projects.

Conclusion :

This project effectively utilized Python libraries such as **Pandas**, **NumPy**, **Matplotlib**, and **Seaborn** to clean, explore, and analyze Blinkit's order dataset. The dataset, which included key attributes like product names, payment methods, delivery statuses, and customer feedback, was first cleaned to ensure accuracy. Missing values were handled, incorrect data formats were corrected, and necessary transformations were made to prepare the data for proper analysis.

Using grouping and aggregation techniques, we were able to identify important patterns such as the most frequently ordered products, commonly used payment methods, and areas with higher rates of delivery delays or cancellations. These patterns were further brought to life through effective data visualizations. Bar charts, pie graphs, and line plots helped represent the data in a visually clear and informative manner, making trends easier to understand and communicate.

The analysis led to several meaningful insights that can be used to improve Blinkit's performance. For example, recognizing top-selling products helps with better inventory management, while identifying delay-prone regions can guide improvements in logistics and delivery operations. Furthermore, analyzing customer feedback and order statuses provides a clear view of user satisfaction, revealing areas where the service can be enhanced.

In addition to improving customer satisfaction and operational efficiency, this project also provides a solid foundation for making smarter, **data-driven business decisions**. These insights can influence marketing strategies, delivery planning, regional expansion, and product stocking. The techniques and processes used in this analysis can also be extended further for more advanced analytics, such as forecasting demand or predicting delays.

Overall, this project demonstrates the value of data analytics in a real-world business setting. It highlights how structured analysis of order data can support growth, improve decision-making, and enhance the overall customer experience for a fast-paced, customer-focused platform like Blinkit.

References :

1. Blinkit Official Website – Company Overview and Services
https://blinkit.com

2. Kaggle – Online Retail Datasets (for data structure reference and analysis approach)
https://www.kaggle.com/datasets

3. Pandas Documentation – Python Data Analysis Library
https://pandas.pydata.org/docs/

4. Seaborn Documentation – Statistical Data Visualization in Python
https://seaborn.pydata.org/

5. Matplotlib Documentation – Data Plotting Library
https://matplotlib.org/stable/contents.html

6. Towards Data Science – Data Cleaning and EDA Best Practices
https://towardsdatascience.com

7. GeeksforGeeks – Python Data Analysis Tutorials
https://www.geeksforgeeks.org/python-data-analysis/