In [3]:
```
import pandas as pd
import numpy as np
```

In [4]:
```
df= pd.read_csv('fact_order_lines.csv',encoding= 'unicode_escape')
```

In [14]:
```
import os
print(os.getcwd())
print(os.listdir())
```

```
C:\Users\KIIT\OneDrive\Desktop\Supply Chain Project Datapool
['.ipynb_checkpoints', 'c2-input-for-participants-1.zip', 'Supply Chain Project.ipynb']
```

In [5]:
```
df.head()
```

Out[5]:

	order_id	order_placement_date	customer_id	product_id	order_qty	agreed_delivery_date	actual_delivery_date	delivery_qt
0	FMR34203601	Tuesday, March 1, 2022	789203	25891601	110	Friday, March 4, 2022	Friday, March 4, 2022	11
1	FMR32320302	Tuesday, March 1, 2022	789320	25891203	347	Wednesday, March 2, 2022	Wednesday, March 2, 2022	34
2	FMR33320501	Tuesday, March 1, 2022	789320	25891203	187	Thursday, March 3, 2022	Thursday, March 3, 2022	15
3	FMR34220601	Tuesday, March 1, 2022	789220	25891203	235	Friday, March 4, 2022	Friday, March 4, 2022	23
4	FMR33703603	Tuesday, March 1, 2022	789703	25891203	176	Thursday, March 3, 2022	Thursday, March 3, 2022	17

On-Time Deliveries

In [6]:
```
total_deliveries = len(df)
on_time_deliveries = df['On Time'].sum()  # Count of True values
ot_percentage = (on_time_deliveries / total_deliveries) * 100

print(f'On-time Delivery Percentage: {ot_percentage:.2f}%')
```

```
On-time Delivery Percentage: 71.12%
```

Daily Metrics Calculation

In [9]:
```
df['actual_delivery_date'] = pd.to_datetime(df['actual_delivery_date'])
```

In [11]:
```
daily_metrics = df.groupby(df['actual_delivery_date']).agg(
    total_deliveries=('order_id', 'count'),
    on_time_deliveries=('On Time', 'sum'),
    in_full_deliveries=('In Full', 'sum'),
    otif_deliveries=('On Time In Full', 'sum')
)
```

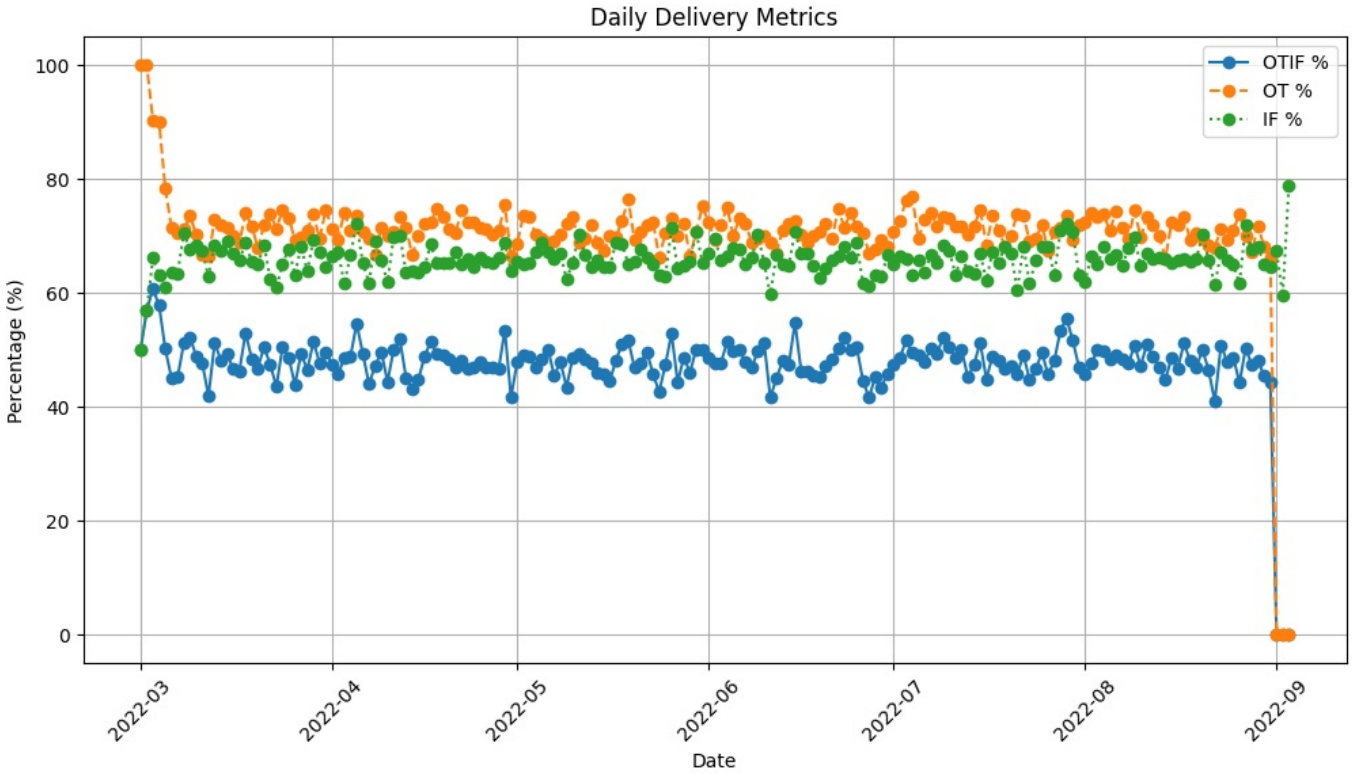
In [12]:
```
daily_metrics['OT%'] = (daily_metrics['on_time_deliveries'] / daily_metrics['total_deliveries']) * 100
daily_metrics['IF%'] = (daily_metrics['in_full_deliveries'] / daily_metrics['total_deliveries']) * 100
daily_metrics['OTIF%'] = (daily_metrics['otif_deliveries'] / daily_metrics['total_deliveries']) * 100

print(daily_metrics)
```

```
                      total_deliveries  on_time_deliveries  \
actual_delivery_date
2022-03-01                           8                   8
2022-03-02                          79                  79
2022-03-03                         186                 168
2022-03-04                         263                 237
2022-03-05                         305                 239
...                                ...                 ...
2022-08-30                         317                 216
2022-08-31                         282                 187
2022-09-01                          98                   0
2022-09-02                          42                   0
2022-09-03                          19                   0

                      in_full_deliveries  otif_deliveries         OT%  \
actual_delivery_date
2022-03-01                             4                4  100.000000
2022-03-02                            45               45  100.000000
2022-03-03                           123              113   90.322581
2022-03-04                           166              152   90.114068
2022-03-05                           186              153   78.360656
...                                  ...              ...         ...
2022-08-30                           206              144   68.138801
2022-08-31                           182              125   66.312057
2022-09-01                            66                0    0.000000
2022-09-02                            25                0    0.000000
2022-09-03                            15                0    0.000000

                            IF%      OTIF%
actual_delivery_date
2022-03-01            50.000000  50.000000
2022-03-02            56.962025  56.962025
2022-03-03            66.129032  60.752688
2022-03-04            63.117871  57.794677
2022-03-05            60.983607  50.163934
...                         ...        ...
2022-08-30            64.984227  45.425868
2022-08-31            64.539007  44.326241
2022-09-01            67.346939   0.000000
2022-09-02            59.523810   0.000000
2022-09-03            78.947368   0.000000

[187 rows x 7 columns]
```

In [13]:
```python
import matplotlib.pyplot as plt
```

Daily Delivery Metrics

In [14]:
```python
plt.figure(figsize=(12, 6))
plt.plot(daily_metrics.index, daily_metrics['OTIF%'], marker='o', label='OTIF %')
plt.plot(daily_metrics.index, daily_metrics['OT%'], marker='o', label='OT %', linestyle='--')
plt.plot(daily_metrics.index, daily_metrics['IF%'], marker='o', label='IF %', linestyle=':')
plt.title('Daily Delivery Metrics')
plt.xlabel('Date')
plt.ylabel('Percentage (%)')
plt.xticks(rotation=45)
plt.legend()
plt.grid()
plt.show()
```



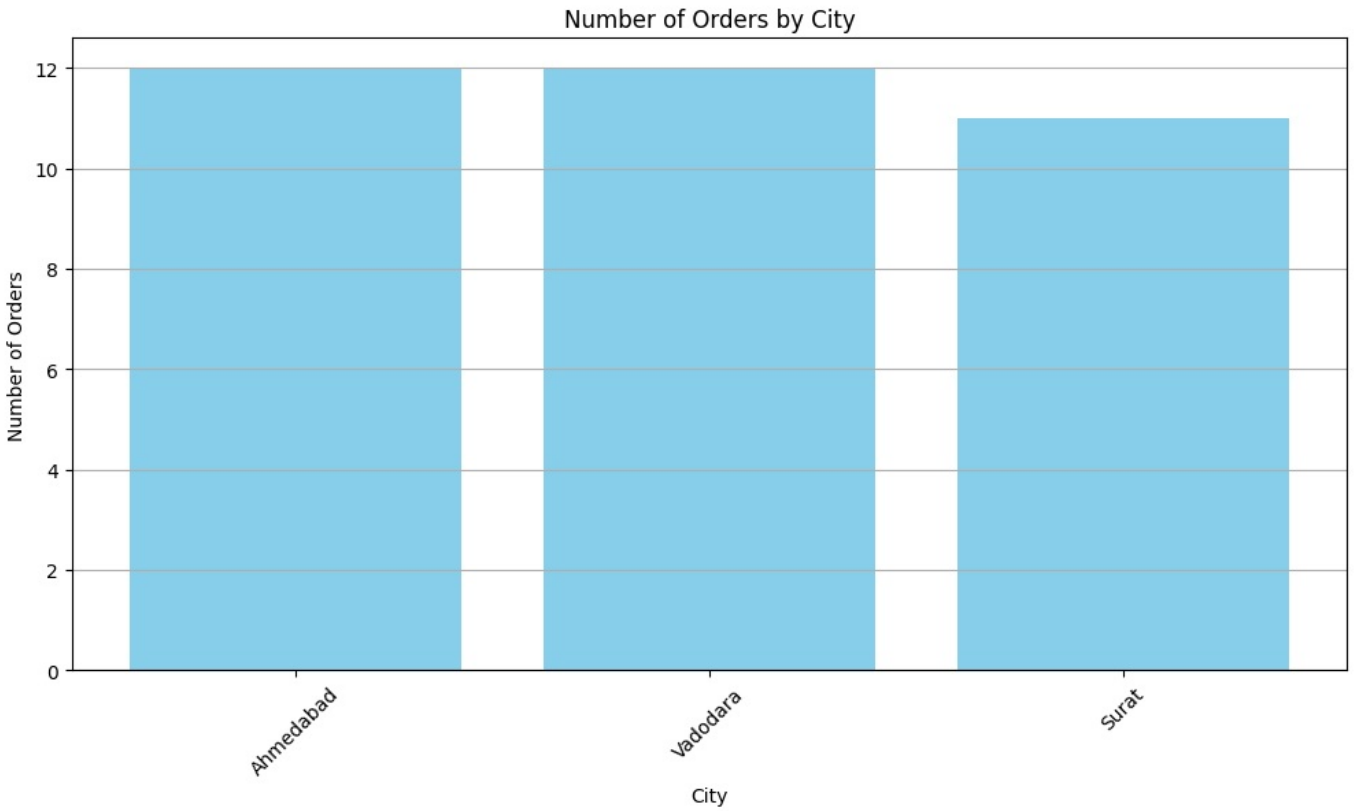
In [16]:
```
df_customers=pd.read_csv('dim_customers.csv',encoding= 'unicode_escape')
```

In [17]:
```
city_order_counts = df_customers['city'].value_counts()

city_order_counts_df = city_order_counts.reset_index()
city_order_counts_df.columns = ['city', 'number_of_orders']

print(city_order_counts_df)
```

```
       city  number_of_orders
0  Ahmedabad                12
1   Vadodara                12
2      Surat                11
```

Visualization of Most bought City

In [18]:
```
plt.figure(figsize=(12, 6))
plt.bar(city_order_counts_df['city'], city_order_counts_df['number_of_orders'], color='skyblue')
plt.title('Number of Orders by City')
plt.xlabel('City')
plt.ylabel('Number of Orders')
plt.xticks(rotation=45)
plt.grid(axis='y')
plt.show()
```



In [19]:
```
most_bought_city = city_order_counts_df.loc[city_order_counts_df['number_of_orders'].idxmax()]

print(f'The most bought city is {most_bought_city["city"]} with {most_bought_city["number_of_orders"]} orders.'
```

```
The most bought city is Ahmedabad with 12 orders.
```

In [20]:
```
df_orders= pd.read_csv('fact_orders_aggregate.csv',encoding= 'unicode_escape')
```

Total orders by OT,IF and OTIF

In [22]:
```
total_orders = len(df_orders)

on_time_orders = df_orders['on_time'].sum()

in_full_orders = df_orders['in_full'].sum()

otif_orders = df_orders['otif'].sum()

print(f'Total Orders: {total_orders}')
print(f'Total On-Time Orders (OT): {on_time_orders}')
print(f'Total In-Full Orders (IF): {in_full_orders}')
print(f'Total On-Time In-Full Orders (OTIF): {otif_orders}')
```

```
Total Orders: 31729
Total On-Time Orders (OT): 18730
Total In-Full Orders (IF): 16747
Total On-Time In-Full Orders (OTIF): 9208
```

In [23]:
```
ot_percentage = (on_time_orders / total_orders) * 100
if_percentage = (in_full_orders / total_orders) * 100
otif_percentage = (otif_orders / total_orders) * 100

print(f'On-Time Delivery Percentage (OT): {ot_percentage:.2f}%')
print(f'In-Full Delivery Percentage (IF): {if_percentage:.2f}%')
print(f'On-Time In-Full Delivery Percentage (OTIF): {otif_percentage:.2f}%')
```

```
On-Time Delivery Percentage (OT): 59.03%
In-Full Delivery Percentage (IF): 52.78%
On-Time In-Full Delivery Percentage (OTIF): 29.02%
```

In [24]:
```
summary = {
    'Metric': ['Total Orders', 'On-Time Orders (OT)', 'In-Full Orders (IF)', 'On-Time In-Full Orders (OTIF)'],
    'Count': [total_orders, on_time_orders, in_full_orders, otif_orders],
    'Percentage': [None, ot_percentage, if_percentage, otif_percentage]
}

summary_df = pd.DataFrame(summary)

print(summary_df)
```

```
                      Metric  Count  Percentage
0                Total Orders  31729         NaN
1           On-Time Orders (OT)  18730   59.031170
2           In-Full Orders (IF)  16747   52.781367
3  On-Time In-Full Orders (OTIF)   9208   29.020770
```

In [26]:
```
df_target= pd.read_csv('dim_targets_orders.csv',encoding='unicode_escape')
```

In [27]:
```
merged_data = pd.merge(df_target, df_customers, on='customer_id', how='left')

print(merged_data.head())
```

```
   customer_id  ontime_target%  infull_target%  otif_target%  \
0       789201              87              81            70
1       789202              85              81            69
2       789203              92              76            70
3       789301              89              78            69
4       789303              88              78            69

       customer_name       city
0          Rel Fresh      Surat
1          Rel Fresh  Ahmedabad
2          Rel Fresh   Vadodara
3  Expression Stores      Surat
4  Expression Stores   Vadodara
```

In [30]:
```
city_targets = merged_data.groupby('city').agg(
    average_ontime_target=('ontime_target%', 'mean'),
    average_infull_target=('infull_target%', 'mean'),
    average_otif_target=('otif_target%', 'mean')
).reset_index()

overall_targets = pd.DataFrame({
    'city': ['Overall'],
    'average_ontime_target': [merged_data['ontime_target%'].mean()],
    'average_infull_target': [merged_data['infull_target%'].mean()],
    'average_otif_target': [merged_data['otif_target%'].mean()]
})

city_targets = pd.concat([city_targets, overall_targets], ignore_index=True)

print(city_targets)
```

```
        city  average_ontime_target  average_infull_target  \
0  Ahmedabad              85.833333              77.333333
1      Surat              86.272727              76.909091
2   Vadodara              86.166667              75.333333
3    Overall              86.085714              76.514286

   average_otif_target
0            66.500000
1            66.363636
2            64.916667
3            65.914286
```

OT,IF and OTIF targets in different cities and overall

In [31]:
```
city_targets.set_index('city', inplace=True)

city_targets.plot(kind='bar', figsize=(12, 6), color=['skyblue', 'lightgreen', 'salmon'])
plt.title('Average OT, IF, and OTIF Targets by City')
plt.xlabel('City')
plt.ylabel('Average Target Percentage (%)')
plt.xticks(rotation=45)
plt.legend(title='Metrics', labels=['On-Time Target %', 'In-Full Target %', 'OTIF Target %'])
plt.grid(axis='y')
plt.tight_layout()
plt.show()
```

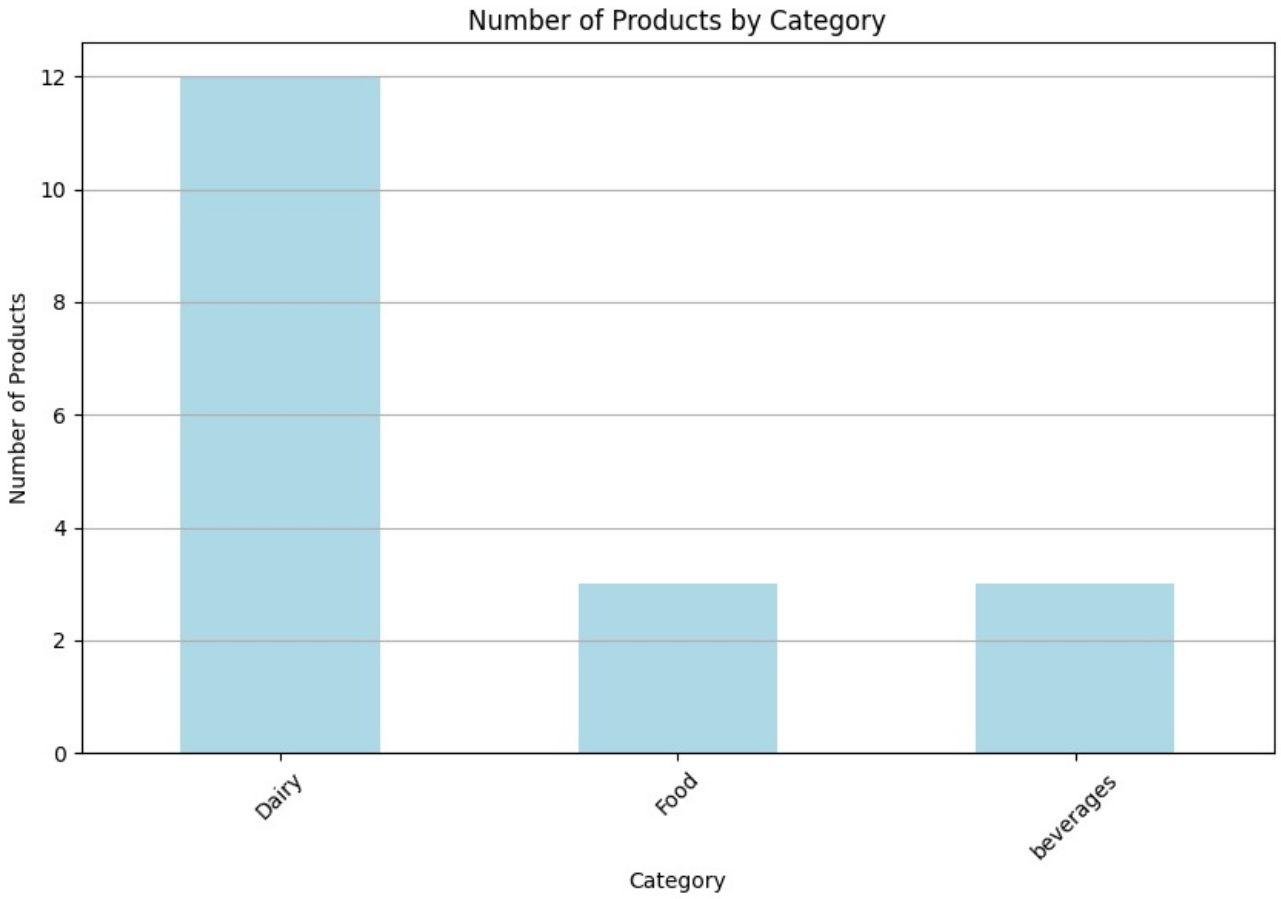
In [32]:
```
df_products=pd.read_csv('dim_products.csv',encoding='unicode_escape')
```

In [37]:
```
product_counts_by_category = df_products['category'].value_counts()
print(product_counts_by_category)
print("\n")

product_counts_by_category = df_products['category'].value_counts()
print(product_counts_by_category)
print("\n")

most_common_product = df_products['product_name'].value_counts().idxmax()
print(f'Most common product: {most_common_product}')
```

```
category
Dairy        12
Food          3
beverages     3
Name: count, dtype: int64


category
Dairy        12
Food          3
beverages     3
Name: count, dtype: int64


Most common product: AM Milk 500
```

Products by Category

In [38]:
```
category_counts = df_products['category'].value_counts()

plt.figure(figsize=(10, 6))
category_counts.plot(kind='bar', color='lightblue')
plt.title('Number of Products by Category')
plt.xlabel('Category')
plt.ylabel('Number of Products')
plt.xticks(rotation=45)
plt.grid(axis='y')
plt.show()
```



Calculating the Line Fill Rate(LiFR) and Volume Fill Rate(VoFR)

In [39]:
```
total_order_lines = len(df)

in_full_deliveries = df[df['In Full'] == True].shape[0]

line_fill_rate = (in_full_deliveries / total_order_lines) * 100

total_ordered_quantity = df['order_qty'].sum()
total_delivered_quantity = df['delivery_qty'].sum()

volume_fill_rate = (total_delivered_quantity / total_ordered_quantity) * 100

print(f'Line Fill Rate (LFR): {line_fill_rate:.2f}%')
print(f'Volume Fill Rate (VFR): {volume_fill_rate:.2f}%')
```

```
Line Fill Rate (LFR): 65.96%
Volume Fill Rate (VFR): 96.59%
```

Spiliting LiFR and VoFR by customer

In [43]:
```
def calculate_fill_rates(group):
    total_order_lines = len(group)
    in_full_deliveries = group[group['In Full'] == True].shape[0]

    line_fill_rate = (in_full_deliveries / total_order_lines) * 100 if total_order_lines > 0 else 0

    total_ordered_quantity = group['order_qty'].sum()
    total_delivered_quantity = group['delivery_qty'].sum()

    volume_fill_rate = (total_delivered_quantity / total_ordered_quantity) * 100 if total_ordered_quantity > 0 

    return pd.Series({
        'Line Fill Rate (LFR)': line_fill_rate,
        'Volume Fill Rate (VFR)': volume_fill_rate
    })
\
customer_fill_rates = df.groupby('customer_id').apply(calculate_fill_rates).reset_index()

print(customer_fill_rates)
```

	customer_id	Line Fill Rate (LFR)	Volume Fill Rate (VFR)
0	789101	74.417178	97.335499
1	789102	73.696824	97.288940
2	789103	29.891957	93.052966
3	789121	74.022850	97.388833
4	789122	29.194030	92.828532
5	789201	74.736189	97.522693
6	789202	74.733096	97.373579
7	789203	74.143302	97.389252
8	789220	75.694016	97.611520
9	789221	75.261538	97.539757
10	789301	73.273810	97.383041
11	789303	77.359655	97.702090
12	789320	75.581395	97.556068
13	789321	75.643440	97.610435
14	789401	75.046555	97.645400
15	789402	75.799638	97.760342
16	789403	76.025237	97.709254
17	789420	74.834835	97.437541
18	789421	30.774032	93.219101
19	789422	74.048659	97.273103
20	789501	74.840764	97.504943
21	789503	75.272727	97.627003
22	789520	29.374202	92.767561
23	789521	73.011016	97.281678
24	789522	73.512837	97.371244
25	789601	30.061728	92.840591
26	789603	76.228998	97.592810
27	789621	75.517661	97.424791
28	789622	75.444840	97.449931
29	789702	30.872483	92.986393
30	789703	75.883069	97.693903
31	789720	74.112607	97.320289
32	789721	74.676525	97.580252
33	789902	75.715155	97.633090
34	789903	29.737965	92.916367

```
C:\Users\KIIT\AppData\Local\Temp\ipykernel_3484\2950193285.py:17: DeprecationWarning: DataFrameGroupBy.apply operated on the grouping columns. This behavior is deprecated, and in a future version of pandas the grouping columns will be excluded from the operation. Either pass `include_groups=False` to exclude the groupings or explicitly select the grouping columns after groupby to silence this warning.
  customer_fill_rates = df.groupby('customer_id').apply(calculate_fill_rates).reset_index()
```

Thus, in conclusion we can derive that AtliQ Mart's analysis reveals strong delivery metrics with a need for improvement in On-Time and In-Full deliveries; calculation of LiFR and VoFR; customer insights highlight the most bought cities, while product distribution shows diverse offerings, guiding strategic decisions for future growth.

In [ ]: