Ecommerce Sales Report

Insights and Recommendations

by Nidhi Haldkar

Executive Summary

This report provides an in-depth analysis of sales performance, product categories, order fulfillment, payment methods, and customer retention. Using SQL and Python, we analyzed data to uncover insights and make recommendations for optimizing business performance.

- **Total Revenue:** $13,591,643.70
- **Total Orders:** 99,441
- **Average Order Value:** $137.75
- **Average Delivery Time:** 12.0 days
- **Retention Rate:** 0.0%

Detailed Analysis

Revenue and Orders

Total Revenue: $13,591,643.70

Total Orders: 99,441

Average Order Value: $137.75

Insights:

- High average order value suggests that customers are making substantial purchases, which is positive for revenue.
- The total revenue indicates a strong overall performance, but a deeper dive into revenue sources and trends can provide more actionable insights.

Recommendations:

- **Promotions and Bundling:** Consider offering promotions or bundling products to increase the average order value further.
- **Customer Segmentation:** Analyze customer segments to tailor marketing strategies that could boost order frequency and revenue.

Order Status Distribution

Order Status	Order Count
Shipped	1,107
Unavailable	609
Invoiced	314
Created	5
Approved	2
Processing	301
Delivered	96,478
Canceled	625

Insights:

- **Delivered Orders:** Majority are delivered, which is positive.
- **Unavailable Orders:** A significant number of orders are marked as unavailable, which could impact customer satisfaction.
- **Canceled Orders:** Low cancellation rate suggests effective order management.

Recommendations:

- **Inventory Management:** Investigate causes of unavailable statuses to improve inventory forecasting and supplier coordination.
- **Order Fulfillment:** Enhance tracking and communication for orders in "Processing" and "Invoiced" statuses to prevent delays.

Quarterly Revenue Trend

Quarter	Total Revenue
2016-3	$267.36
2016-4	$49,518.56
2017-1	$741,960.19
2017-2	$1,299,036.97
2017-3	$1,696,404.85
2017-4	$2,418,404.97
2018-1	$2,777,422.51
2018-2	$2,858,289.74
2018-3	$1,750,338.55

Insights:

- Steady revenue growth from 2016 to 2018, with peaks in late 2017 and early 2018.
- Decline in Q3 2018 needs further investigation.

Recommendations:

- **Trend Analysis:** Analyze marketing campaigns, seasonal effects, or competitive factors influencing revenue drops.
- **Strategic Planning:** Use revenue trends to forecast and plan for future periods, optimizing inventory and marketing strategies accordingly.

Product Category Revenue Share

Product Category	Total Revenue
Health Beauty	$1,258,681.34
Watches Present	$1,205,005.68
Bed Table Bath	$1,036,988.68
Sport Leisure	$988,048.97
Computer Accessories	$911,954.32
Furniture Decoration	$729,762.49
Cool Stuff	$635,290.85
Housewares	$632,248.66
Automotive	$592,720.11
Garden Tools	$485,256.46

Insights:

- **Top Categories:** Health Beauty and Watches Present are top revenue categories.
- **Lower Categories:** Garden Tools has the lowest revenue share among top categories.

Recommendations:

- **Category Focus:** Allocate more resources to top-performing categories and explore growth opportunities in lower-performing ones.
- **Market Research:** Conduct market research to understand demand in categories like Garden Tools and explore strategies to boost sales.

Delivery Time Distribution

Delivery Time Bin	Count
0-10	46,482
11-20	35,768
21-30	9,673
31-40	2,796
41-50	1,086
51-60	365
61-70	126
71-80	74
81-90	29
91-100	13
101-150	40
151-200	22
201-250	2

Insights:

- Majority of deliveries occur within 0-10 days, which is excellent.
- A small fraction of deliveries take longer than 30 days, which may be due to logistical issues.

Recommendations:

- **Logistics Improvement:** Optimize logistics and supply chain processes to minimize delivery times beyond 30 days.
- **Customer Communication:** Improve communication with customers regarding estimated delivery times to manage expectations.

Top 10 Sellers By Total Revenue

Seller ID	Total Revenue
4869f7a5dfa277a7dca6462dcf3b52b2	$229,472.63
53243585a1d6dc2643021fd1853d8905	$222,776.05
4a3ca9315b744ce9f8e9374361493884	$200,472.92
fa1c13f2614d7b5c4749cbc52fecda94	$194,042.03
7c67e1448b00f6e969d365cea6b010ab	$187,923.89
7e93a43ef30c4f03f38b393420bc753a	$176,431.87
da8622b14eb17ae2831f4ac5b9dab84a	$160,236.57
7a67c85e85bb2ce8582c35f2203ad736	$141,745.53
1025f0e2d44d7041d6cf58b6550e0bfa	$138,968.55
955fee9216a65b617aa5c0531780ce60	$135,171.70

Insights:

- **High Revenue Sellers:** Top sellers contribute a significant portion of revenue.
- **Potential Partnerships:** High-revenue sellers could be leveraged for exclusive deals or partnerships.

Recommendations:

- **Strategic Partnerships:** Explore strategic partnerships or exclusive offers with top sellers to enhance sales.
- **Seller Support:** Provide additional support and resources to top sellers to maximize their performance.

Top 10 Products By Quantity Sold

Product ID	Quantity Sold
aca2eb7d00ea1a7b8ebd4e68314663af	527
99a4788cb24856965c36a24e339b6058	488
422879e10f46682990de24d770e7f83d	484
389d119b48cf3043d311335e499d9c6b	392
368c6c730842d78016ad823897a372db	388
53759a2ecddad2bb87a079a1f1519f73	373
d1c427060a0f73f6b889a5c7c61f2ac4	343
53b36df67ebb7c41585e8d54d6772e08	323
154e7e31ebfa092203795c972e5804a6	281
3dd2a17168ec895c781a9191c1e95ad7	274

Insights:

- **High Quantity Products:** Products with high quantities sold reflect strong demand and popularity.
- **Inventory Management:** Ensure adequate inventory levels for high-quantity products to avoid stockouts.

Recommendations:

- **Marketing Focus:** Promote high-quantity products more aggressively.
- **Inventory Planning:** Optimize inventory planning to ensure availability of top-selling products.

Revenue Contribution By Payment Type

Payment Type	Total Revenue
Debit Card	$217,989.80
Voucher	$379,436.90
UPI	$2,869,361.00
Credit Card	$12,542,083.00

Insights:

- **Credit Card Dominance:** Credit card payments contribute the most to total revenue.
- **UPI Growth:** UPI payments are growing and contribute significantly.

Recommendations:

- **Payment Methods:** Encourage customers to use payment methods that are already successful, such as credit cards.
- **Expand Payment Options:** Consider expanding payment options to include more digital methods like UPI.

[1]:
```
import pandas as pd
import matplotlib.pyplot as plt
import seaborn as sns
import plotly.express as px
import numpy as np
from matplotlib.colors import LinearSegmentedColormap
import plotly.graph_objects as go
```

[2]:
```
from sqlalchemy import create_engine

# engine = create_engine('postgresql://username:password@host:port/database')

db = create_engine('postgresql://postgres:password@localhost:5432/ecommerce')
```

[3]:
```
query='''
-- 1. Total Revenue:
-- Description: Total amount of money earned from sales.
SELECT round(SUM(price)::numeric,2) AS total_revenue
FROM order_items;
'''


data = pd.read_sql(query, db)
print(f'Total Revenue (Sales) : {data['total_revenue'][0]}')
```

```
Total Revenue (Sales) : 13591643.7
```

[4]:
```
query='''
-- 2. Average Order Value (AOV):
-- Description: Average amount spent per order.

SELECT round(AVG(total_order_value)::numeric,2) AS average_order_value
FROM (
    SELECT order_id, SUM(price) AS total_order_value
    FROM order_items
    GROUP BY order_id
) AS order_totals;

'''

data = pd.read_sql(query, db)
print(f'Average Order Value : {data['average_order_value'][0]}')
```

```
Average Order Value : 137.75
```

[5]:
```
query='''
-- Total Number of Orders:
-- Description: Total count of orders placed.

SELECT COUNT(DISTINCT order_id) AS total_orders
FROM orders;
'''

data = pd.read_sql(query, db)
print(f'Total Orders : {data['total_orders'][0]}')
```

```
Total Orders : 99441
```

[7]:
```
query='''
-- Average Delivery Time:
-- Description: Average time taken to deliver an order from purchase to delivery.

SELECT
    round(AVG(EXTRACT(day FROM (order_delivered_customer_date::timestamp - order_purchase_timestamp::timestamp))),0) AS average_delivery_time
FROM orders
WHERE order_delivered_customer_date IS NOT NULL;
'''

data = pd.read_sql(query, db)
print(f'Average Delivery Time : {data['average_delivery_time'][0]}')
```

```
Average Delivery Time : 12.0
```

[9]:
```
query='''
-- Order Status Distribution:
-- Description: Distribution of different order statuses (e.g., delivered, shipped).

SELECT order_status, COUNT(*) AS order_count
FROM orders
GROUP BY order_status;
'''


data = pd.read_sql(query, db)
print(f'Order Status Distribution : \n\n{data}')
```

```
Order Status Distribution :

  order_status  order_count
0      shipped         1107
1  unavailable          609
2     invoiced          314
3      created            5
4     approved            2
5   processing          301
6    delivered        96478
7     canceled          625
```

[10]:
```
query='''
WITH customer_order_counts AS (
    SELECT customer_id, COUNT(order_id) AS order_count
    FROM orders
    GROUP BY customer_id
)
SELECT
    round((COUNT(CASE WHEN order_count > 1 THEN 1 END) * 100.0 / COUNT(*)),0) AS retention_rate
FROM customer_order_counts;
'''


data = pd.read_sql(query, db)
print(f'Retention Rate : {data['retention_rate'][0]}')
```

```
Retention Rate : 0.0
```

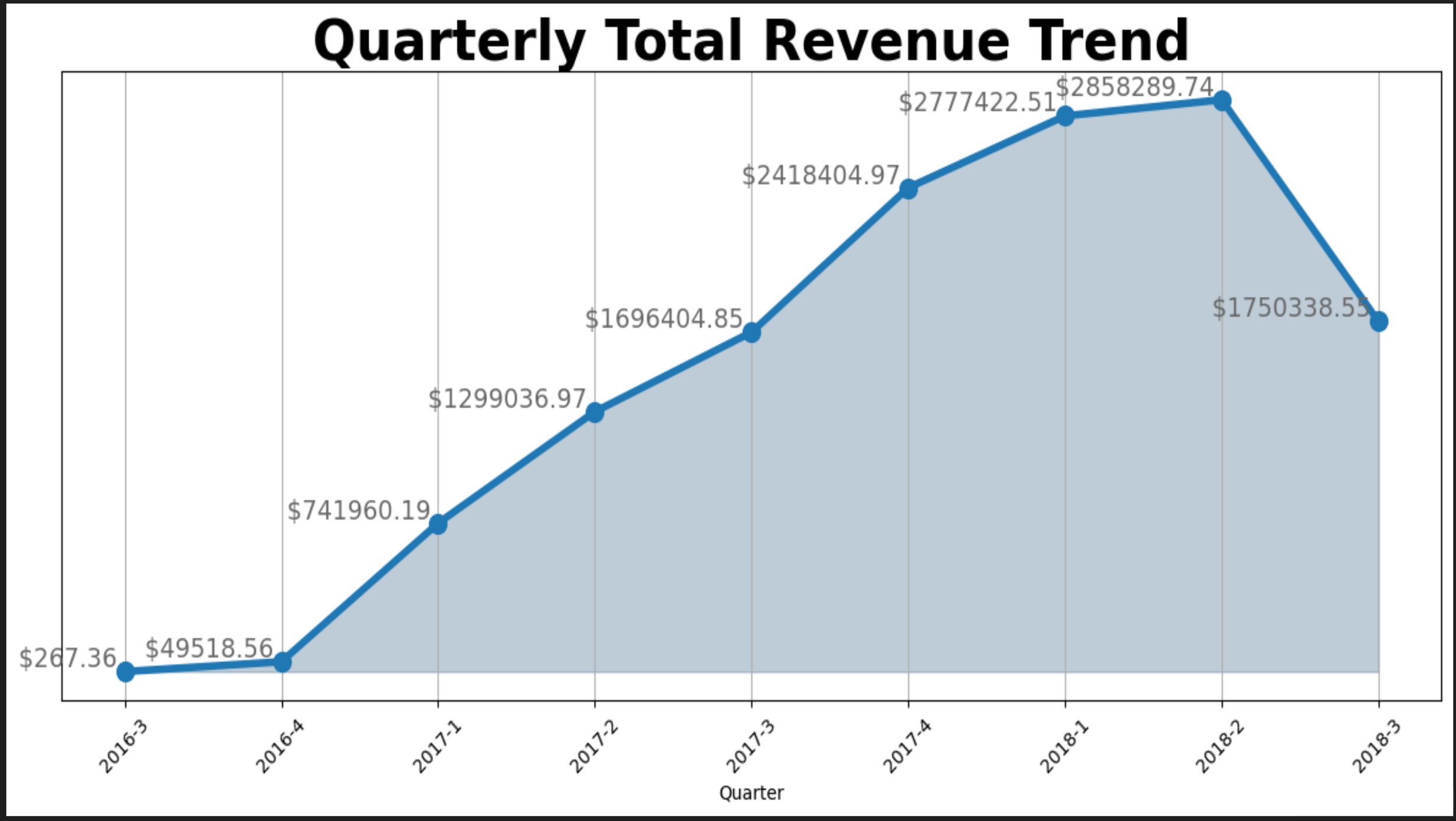
[12]:

```
query='''
        SELECT
            TO_CHAR(date_trunc('quarter', o.order_purchase_timestamp::timestamp), 'YYYY-Q') AS quarter,
            round(SUM(oi.price)::numeric,2) AS total_revenue
        FROM
            orders o
        JOIN
            order_items oi ON o.order_id = oi.order_id
        GROUP BY
            quarter
        ORDER BY
            quarter;
'''

data=pd.read_sql(query, db)
print(f'Quarterly Total Revenue Trend : \n\n{data}')
```

```
Quarterly Total Revenue Trend :

  quarter  total_revenue
0  2016-3         267.36
1  2016-4       49518.56
2  2017-1      741960.19
3  2017-2     1299036.97
4  2017-3     1696404.85
5  2017-4     2418404.97
6  2018-1     2777422.51
7  2018-2     2858289.74
```

[13]:

```
# Monthly Revenue Trend

plt.figure(figsize=(12, 6))
plt.fill_between(data['quarter'],data['total_revenue'], color='#5e819d' , alpha=0.4)  # Area
plt.plot(data['quarter'], data['total_revenue'], marker='o', markersize=10, linestyle='-', linewidth=4)
plt.title('Quarterly Total Revenue Trend', fontsize=30, fontweight='bold')
plt.xlabel('Quarter')
plt.ylabel('')
plt.yticks([])
# plt.ylabel('Total Revenue')
plt.xticks(rotation=45)
plt.grid(True)

# Annotate each data point with its value
ax = plt.gca()  # Get current axis
for i, (quarter, revenue) in enumerate(zip(data['quarter'],
                                           data['total_revenue'])):
    ax.text(quarter, revenue, f'${revenue:.2f} ', color='dimgrey', ha='right', va='bottom', fontsize=14,
            # bbox=dict(facecolor='white', edgecolor='none', alpha=0.7))
           )


plt.tight_layout()
plt.show()
```

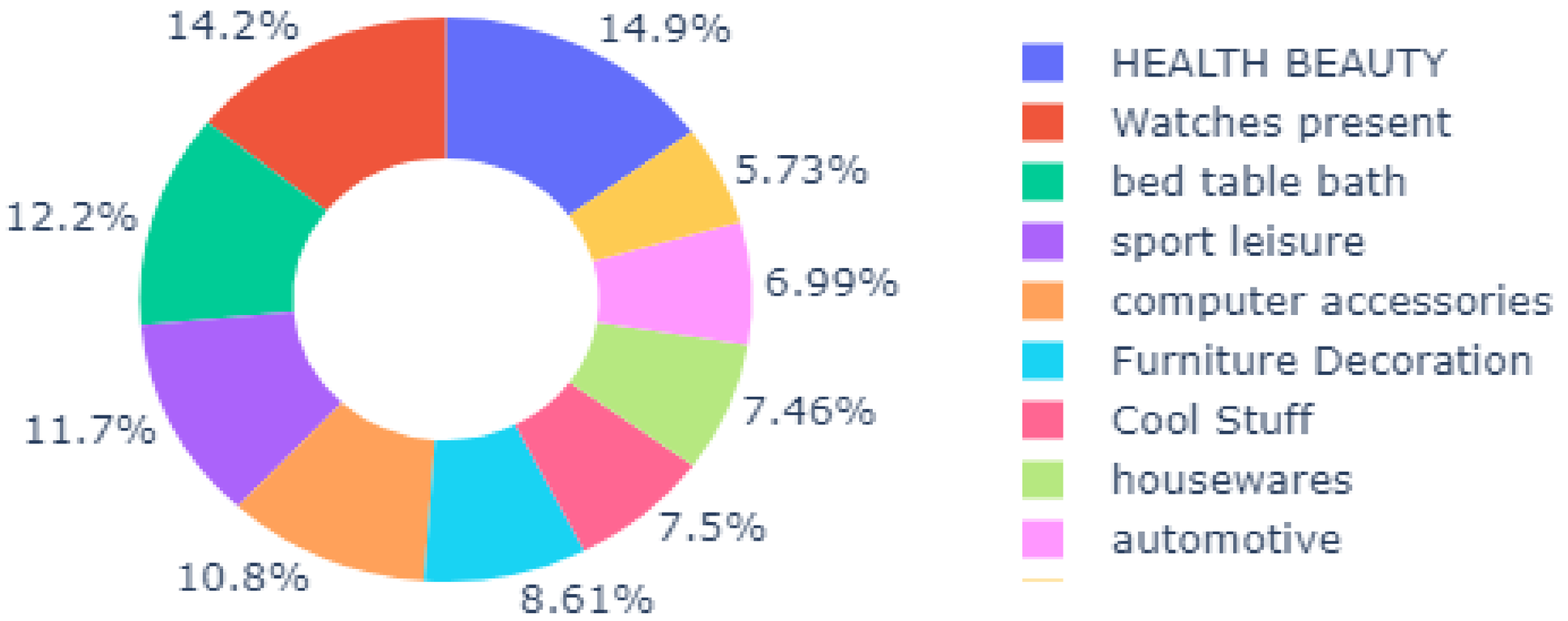

Quarterly Total Revenue Trend
$267.36
$49518.56
$741960.19
$1299036.97
$1696404.85
$2418404.97
$2777422.51
$2858289.74
$1750338.55
2016-3
2016-4
2017-1
2017-2
2017-3
2017-4
2018-1
2018-2
2018-3
Quarter

[14]:
```
query='''
        SELECT
            p.product_category,
            round(SUM(oi.price)::numeric,2) AS total_revenue
        FROM
            order_items oi
        JOIN
            products p ON oi.product_id = p.product_id
        GROUP BY
            p.product_category
        ORDER BY
            total_revenue desc
        LIMIT 10
'''

data = pd.read_sql(query, db)
print(f'Product Category  Share : \n\n{data}')
```

```
Product Category  Share :

       product_category  total_revenue
0          HEALTH BEAUTY     1258681.34
1        Watches present     1205005.68
2         bed table bath     1036988.68
3          sport leisure      988048.97
4  computer accessories      911954.32
5  Furniture Decoration      729762.49
6             Cool Stuff      635290.85
7             housewares      632248.66
8             automotive      592720.11
9           Garden tools      485256.46
```

[15]:
```python
# Product Category Share
fig = px.pie(data,
             names='product_category',
             values='total_revenue',
             title='Product Category Share',
             # color_discrete_sequence=px.colors.sequential.RdBu,
             hole=0.5  # Create a donut chart if you prefer
            )


fig.update_layout(title_font_size=30,  # Title font size
                  title_font_family='Arial',  # Title font family
                  title_font_weight='bold'
                 )

# Update traces to label data outside the pie slices
fig.update_traces(
    textinfo='percent',  # Display label, percent, and value
    textposition='outside',  # Position text outside of the pie slices
)

fig.show()
```

Product Category Share



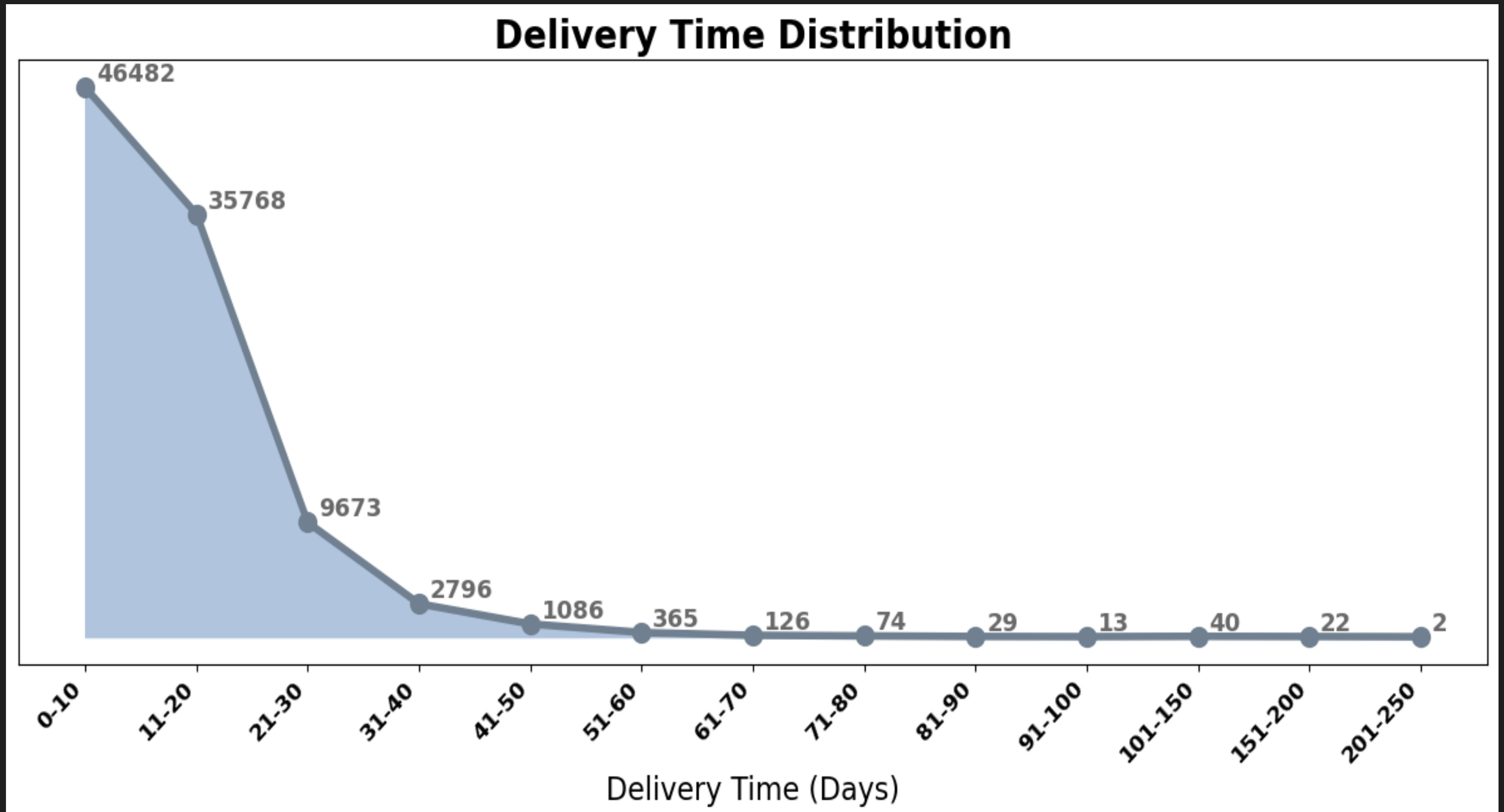
[16]:
```
query='''
        SELECT
            EXTRACT(day FROM (order_delivered_customer_date::timestamp - order_purchase_timestamp::timestamp))  delivery_time
            ,count(EXTRACT(day FROM (order_delivered_customer_date::timestamp - order_purchase_timestamp::timestamp))) count
        FROM
            orders o
        WHERE
            o.order_delivered_customer_date IS NOT NULL
        GROUP BY delivery_time
        ORDER BY delivery_time desc
'''
data = pd.read_sql(query, db)
print(f'Delivery Time Distribution : \n\n{data}')
```

```
Delivery Time Distribution :

     delivery_time  count
0            209.0      1
1            208.0      1
2            195.0      1
3            194.0      3
4            191.0      1
..             ...    ...
141            4.0   4828
142            3.0   3849
143            2.0   3168
144            1.0   1572
145            0.0     13

[146 rows x 2 columns]
```

[17]:

```
df = data
# Define bins for delivery times
bins = [0, 10, 20, 30, 40, 50, 60, 70, 80, 90, 100, 150, 200, 250]
labels = ['0-10', '11-20', '21-30', '31-40', '41-50', '51-60', '61-70', '71-80', '81-90', '91-100', '101-150', '151-200',
          '201-250']
# Create a new column 'delivery_time_bin' based on the bins
df['delivery_time_bin'] = pd.cut(df['delivery_time'], bins=bins, labels=labels, right=False)
# Aggregate counts by bins
df_grouped = df.groupby('delivery_time_bin').agg({'count': 'sum'}).reset_index()
# Display the aggregated DataFrame
print(df_grouped)# Convert labels to a numeric range for plotting
x_labels = list(range(len(labels)))
count=df_grouped['count']
# Plot the area plot
plt.figure(figsize=(12, 6))
# plt.fill_between(x_labels, count, color='#00008B', alpha=0.4)  # Area
plt.fill_between(x_labels, count, color='lightsteelblue')  # Area
# plt.plot(x_labels, count, color='#030764', alpha=0.8, linewidth=2 , marker='o')  # Line
plt.plot(x_labels, count, color='slategrey', linewidth=4 , marker='o', markersize=10)  # Line
# Add labels to each data point
for x, y in zip(x_labels, count):
    plt.text(x+0.1, y + 10, int(y), ha='left', va='bottom', fontsize=13, color='dimgrey', fontweight='bold')
# Set the x-axis with bin labels
plt.xticks(x_labels, labels, rotation=45, ha='right', fontsize=12, fontweight='semibold')
# Title and labels
plt.title('Delivery Time Distribution', fontsize=20, fontweight='bold')
plt.xlabel('Delivery Time (Days)', fontsize=16)
plt.ylabel('')
plt.yticks([])
plt.tight_layout()
plt.show()
```

	delivery_time_bin	count
0	0-10	46482
1	11-20	35768
2	21-30	9673
3	31-40	2796
4	41-50	1086
5	51-60	365
6	61-70	126
7	71-80	74
8	81-90	29
9	91-100	13
10	101-150	40
11	151-200	22
12	201-250	2


Delivery Time Distribution
46482
35768
9673
2796
1086
365
126
74
29
13
40
22
2
0-10
11-20
21-30
31-40
41-50
51-60
61-70
71-80
81-90
91-100
101-150
151-200
201-250
Delivery Time (Days)

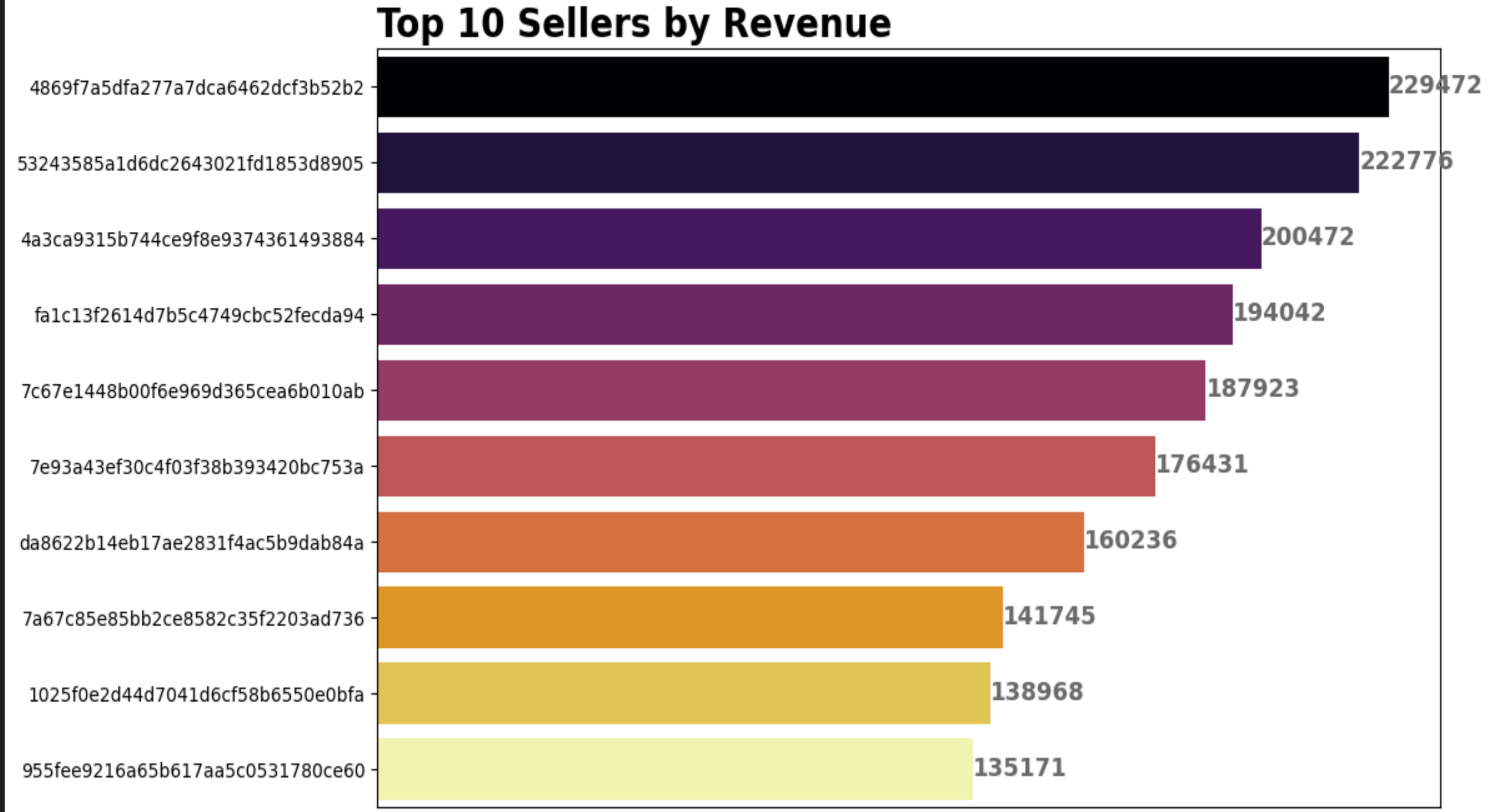
[20]:

```
query='''
 SELECT
            s.seller_id,
            round(SUM(oi.price)::numeric,2) AS total_revenue
        FROM
            order_items oi
        JOIN
            sellers s ON oi.seller_id = s.seller_id
        GROUP BY
            s.seller_id, s.seller_city, s.seller_state
        ORDER BY
            total_revenue DESC
        LIMIT 10;
'''

data=pd.read_sql(query, db)
print(f'Top 10 Sellers By Total Revenue : \n\n{data}')
```

```
Top 10 Sellers By Total Revenue :

                          seller_id  total_revenue
0  4869f7a5dfa277a7dca6462dcf3b52b2      229472.63
1  53243585a1d6dc2643021fd1853d8905      222776.05
2  4a3ca9315b744ce9f8e9374361493884      200472.92
3  fa1c13f2614d7b5c4749cbc52fecda94      194042.03
4  7c67e1448b00f6e969d365cea6b010ab      187923.89
5  7e93a43ef30c4f03f38b393420bc753a      176431.87
6  da8622b14eb17ae2831f4ac5b9dab84a      160236.57
7  7a67c85e85bb2ce8582c35f2203ad736      141745.53
8  1025f0e2d44d7041d6cf58b6550e0bfa      138968.55
9  955fee9216a65b617aa5c0531780ce60      135171.70
```

[21]:

```
# Top 10 Sellers by Revenue
cmap=plt.get_cmap('inferno')
plt.figure(figsize=(12, 6))
sns.barplot(data=data, x='total_revenue', y='seller_id', palette=cmap(np.linspace(0, 1, len(data))))
# Add labels to each data point
for index, row in data.iterrows():
    plt.text(
        row['total_revenue'] + 0.1,  # x-coordinate
        index,  # y-coordinate
        int(row['total_revenue']),  # text label
        ha='left',
        va='center',  # Align text vertically centered to the y-coordinate
        fontsize=13,
        color='dimgrey',
        fontweight='bold'
    )
plt.title('Top 10 Sellers by Revenue', fontsize=20, fontweight='bold', loc='left')
plt.xlabel('')
plt.xticks([])
plt.ylabel('')
plt.tight_layout()
plt.show()
```

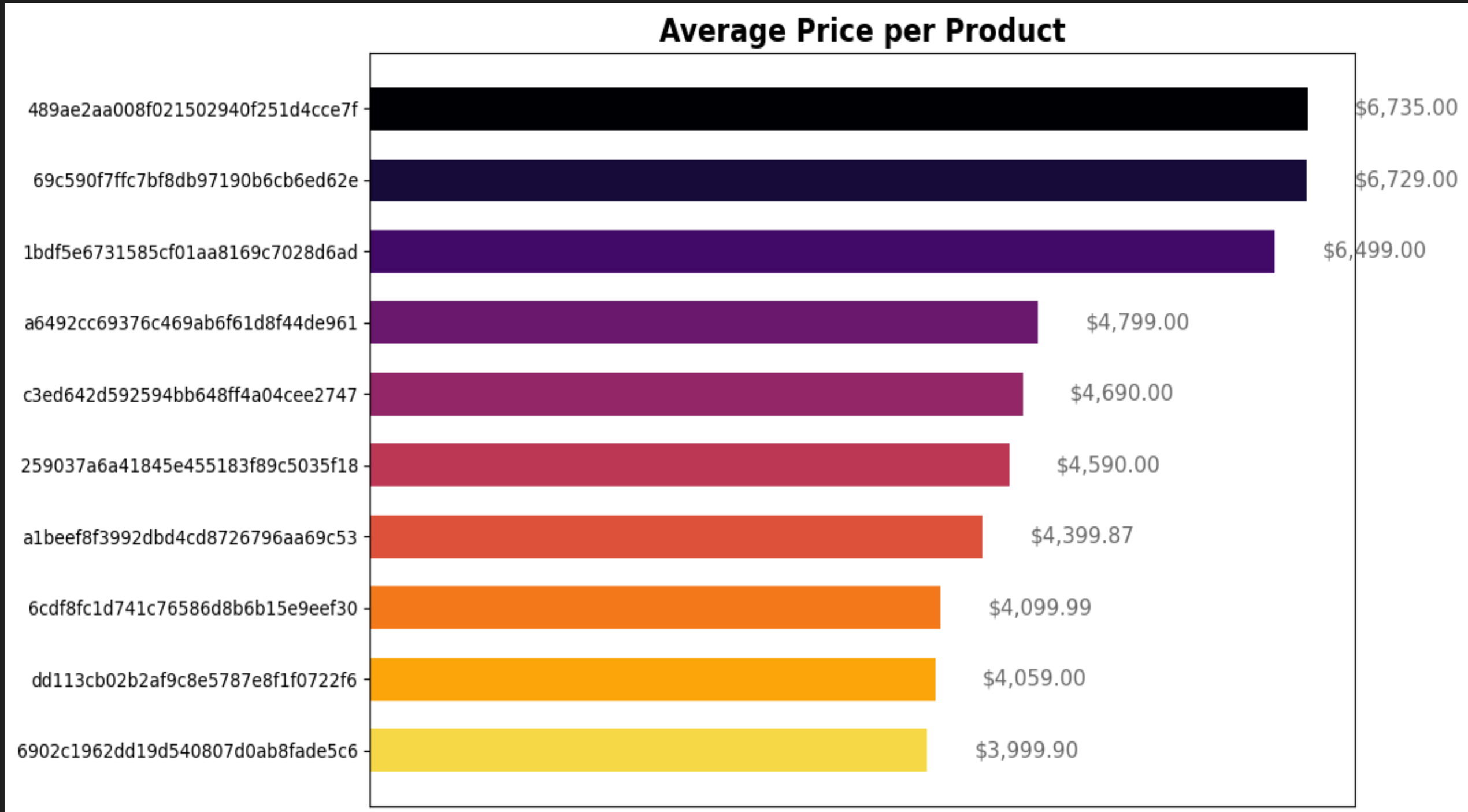

Top 10 Sellers by Revenue
4869f7a5dfa277a7dca6462dcf3b52b2
229472
53243585a1d6dc2643021fd1853d8905
222776
4a3ca9315b744ce9f8e9374361493884
200472
fa1c13f2614d7b5c4749cbc52fecda94
194042
7c67e1448b00f6e969d365cea6b010ab
187923
7e93a43ef30c4f03f38b393420bc753a
176431
da8622b14eb17ae2831f4ac5b9dab84a
160236
7a67c85e85bb2ce8582c35f2203ad736
141745
1025f0e2d44d7041d6cf58b6550e0bfa
138968
955fee9216a65b617aa5c0531780ce60
135171

[22]:

```
query='''
        SELECT
            p.product_id,
            oi.price price
        FROM
            order_items oi
        JOIN
            products p ON oi.product_id = p.product_id
        ORDER BY
            price DESC
        LIMIT 10
'''
data = pd.read_sql(query, db)
print(f'Average Price Per Products : \n\n{data}')
```

```
Average Price Per Products :

                          product_id    price
0  489ae2aa008f021502940f251d4cce7f  6735.00
1  69c590f7ffc7bf8db97190b6cb6ed62e  6729.00
2  1bdf5e6731585cf01aa8169c7028d6ad  6499.00
3  a6492cc69376c469ab6f61d8f44de961  4799.00
4  c3ed642d592594bb648ff4a04cee2747  4690.00
5  259037a6a41845e455183f89c5035f18  4590.00
6  a1beef8f3992dbd4cd8726796aa69c53  4399.87
7  6cdf8fc1d741c76586d8b6b15e9eef30  4099.99
8  dd113cb02b2af9c8e5787e8f1f0722f6  4059.00
9  6902c1962dd19d540807d0ab8fade5c6  3999.90
```

[23]:
```
# Define the number of stages (i.e., number of products)
stages = len(data)
stage_labels = data['product_id']
prices = data['price']

# Create the funnel chart
plt.figure(figsize=(12, 6))

# Create the funnel stages with bars
for i, (price, label) in enumerate(zip(prices, stage_labels)):
        # Calculate the color for the current bar
    color = cmap(i / stages)  # Use the colormap to get the gradient color
    plt.barh(i, price, color=color,  height=0.6)
    plt.text(price + 0.05 * prices.max(), i, f'${price:,.2f}', va='center', fontsize=12, color='dimgrey')

# Adjust the plot
plt.yticks(np.arange(stages), labels=stage_labels)
plt.xticks([])
# plt.xlabel('Price')
plt.title('Average Price per Product', fontsize=16, fontweight='bold')
plt.gca().invert_yaxis()  # Invert y-axis to have the largest value at the top
plt.tight_layout()

# Show the plot
plt.show()
```


Average Price per Product
489ae2aa008f021502940f251d4cce7f
69c590f7ffc7bf8db97190b6cb6ed62e
1bdf5e6731585cf01aa8169c7028d6ad
a6492cc69376c469ab6f61d8f44de961
c3ed642d592594bb648ff4a04cee2747
259037a6a41845e455183f89c5035f18
a1beef8f3992dbd4cd8726796aa69c53
6cdf8fc1d741c76586d8b6b15e9eef30
dd113cb02b2af9c8e5787e8f1f0722f6
6902c1962dd19d540807d0ab8fade5c6
$6,735.00
$6,729.00
$6,499.00
$4,799.00
$4,690.00
$4,590.00
$4,399.87
$4,099.99
$4,059.00
$3,999.90

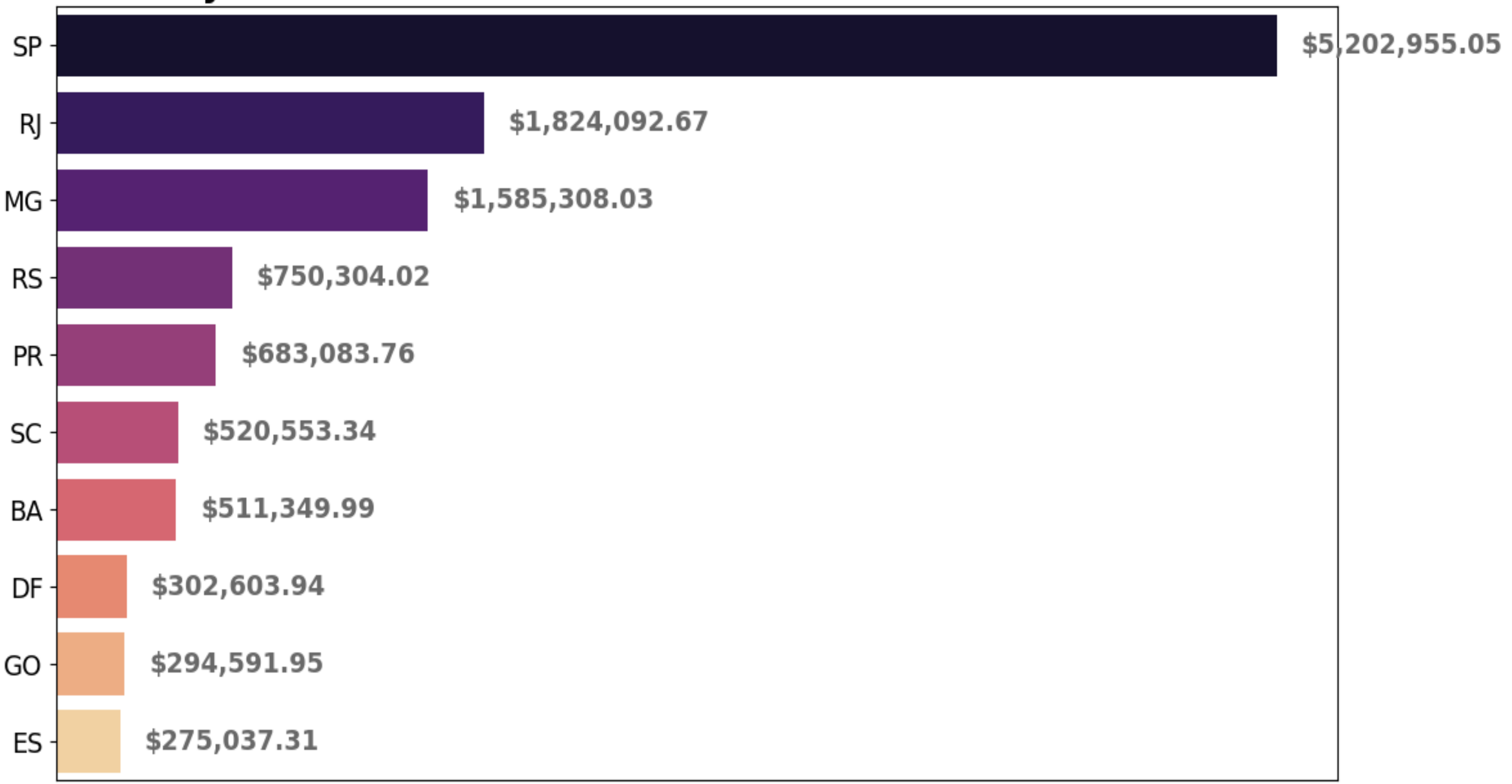
[24]:
```
query='''
  SELECT
            c.customer_state,
            round(SUM(oi.price)::numeric,2) AS total_sales
        FROM
            order_items oi
        JOIN
            orders o ON oi.order_id = o.order_id
        JOIN
            customers c ON o.customer_id = c.customer_id
        GROUP BY
            c.customer_state
        ORDER BY
            total_sales desc
        LIMIT 10;
'''
data = pd.read_sql(query, db)
print(f'Top 10 States By Total Revenue (Sales) : \n\n{data}')
```

```
Top 10 States By Total Revenue (Sales) :

  customer_state  total_sales
0             SP   5202955.05
1             RJ   1824092.67
2             MG   1585308.03
3             RS    750304.02
4             PR    683083.76
5             SC    520553.34
6             BA    511349.99
7             DF    302603.94
8             GO    294591.95
9             ES    275037.31
```

[25]:

```
# Sales by State
plt.figure(figsize=(12, 6))
sns.barplot(data=data, x='total_sales', y='customer_state', palette='magma',legend=False)
# Add data labels to each bar
for index, value in enumerate(data['total_sales']):
    plt.text(value + 0.02 * data['total_sales'].max(), index, f'${value:,.2f}', va='center', fontsize=14, color='dimgrey', fontweight='bold')

# Add title and labels
plt.title('Sales by State', loc='left', fontsize=20, fontweight='bold')
plt.ylabel('')
plt.yticks(data['customer_state'], fontsize=14)

plt.xticks([])
plt.xlabel('')

plt.grid(False)


plt.tight_layout()
plt.show()
```

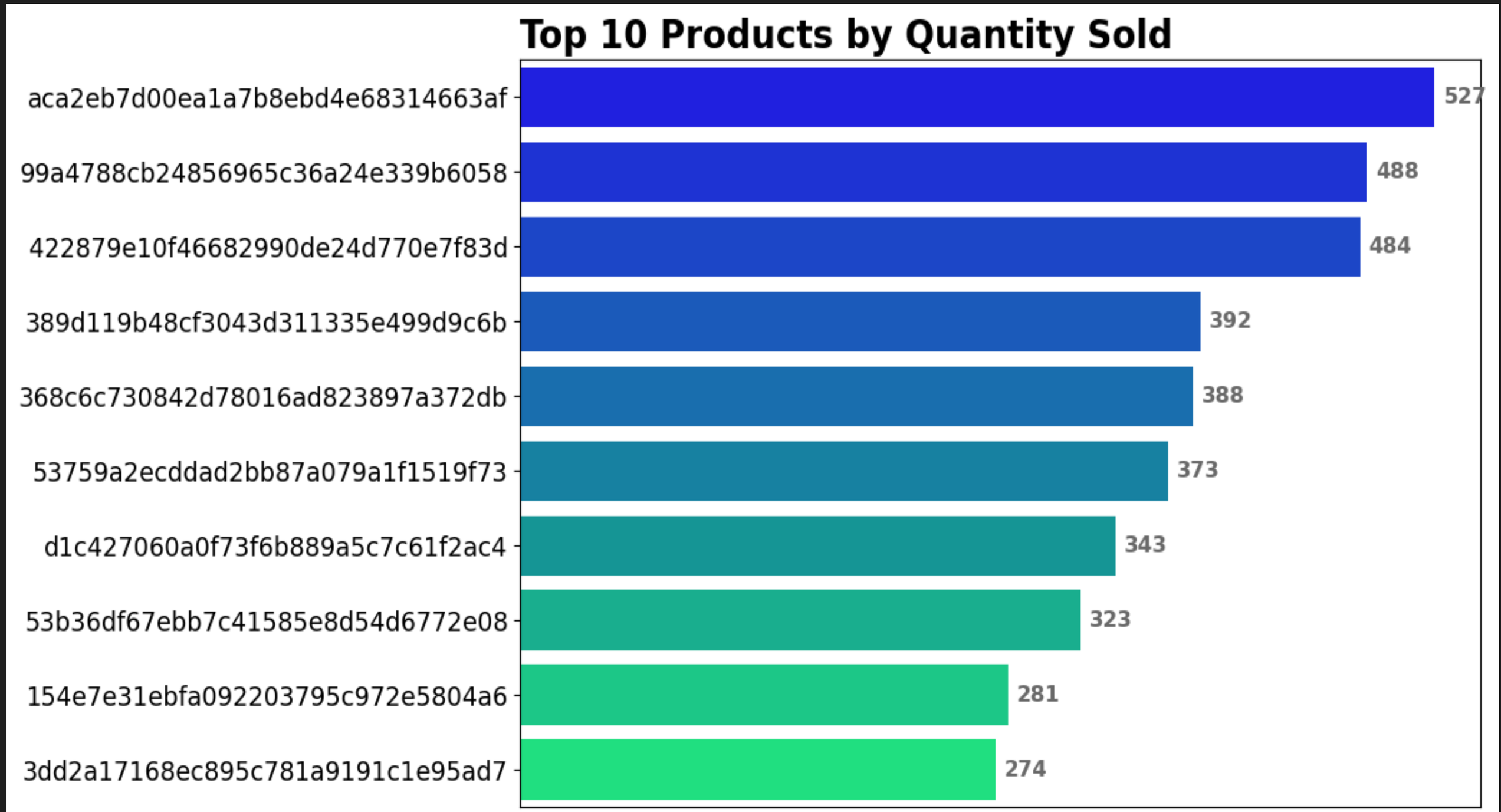
Sales by State



[26]:
```
query='''
 SELECT
            p.product_id,
            COUNT(oi.order_item_id) AS quantity_sold
        FROM
            order_items oi
        JOIN
            products p ON oi.product_id = p.product_id
        GROUP BY
            p.product_id
        ORDER BY
            quantity_sold DESC
        LIMIT 10;
'''
data= pd.read_sql(query, db)
print(f'Top 10 Products By Quantity Sold : \n\n{data}')
```

```
Top 10 Products By Quantity Sold :

                         product_id  quantity_sold
0  aca2eb7d00ea1a7b8ebd4e68314663af            527
1  99a4788cb24856965c36a24e339b6058            488
2  422879e10f46682990de24d770e7f83d            484
3  389d119b48cf3043d311335e499d9c6b            392
4  368c6c730842d78016ad823897a372db            388
5  53759a2ecddad2bb87a079a1f1519f73            373
6  d1c427060a0f73f6b889a5c7c61f2ac4            343
7  53b36df67ebb7c41585e8d54d6772e08            323
8  154e7e31ebfa092203795c972e5804a6            281
9  3dd2a17168ec895c781a9191c1e95ad7            274
```

•[27]:
```
# Top 10 Products by Quantity Sold
cmap = plt.get_cmap('winter')
# Set up the figure
plt.figure(figsize=(12, 6))

# Create the bar plot with the gradient color palette
barplot = sns.barplot(data=data, x='quantity_sold', y='product_id', palette=cmap(np.linspace(0, 1, len(data))))

# Add data labels to each bar
for index, value in enumerate(data['quantity_sold']):
    plt.text(value + 5, index, f'{value}', va='center', fontsize=12, color='dimgrey', fontweight='bold')

# Add title and labels
plt.title('Top 10 Products by Quantity Sold', loc='left', fontsize=20, fontweight='bold')

# Hide y-axis ticks and label
plt.xticks([])
plt.xlabel('')

plt.ylabel('')
plt.yticks( fontsize=14)

plt.tight_layout()
plt.show()
```

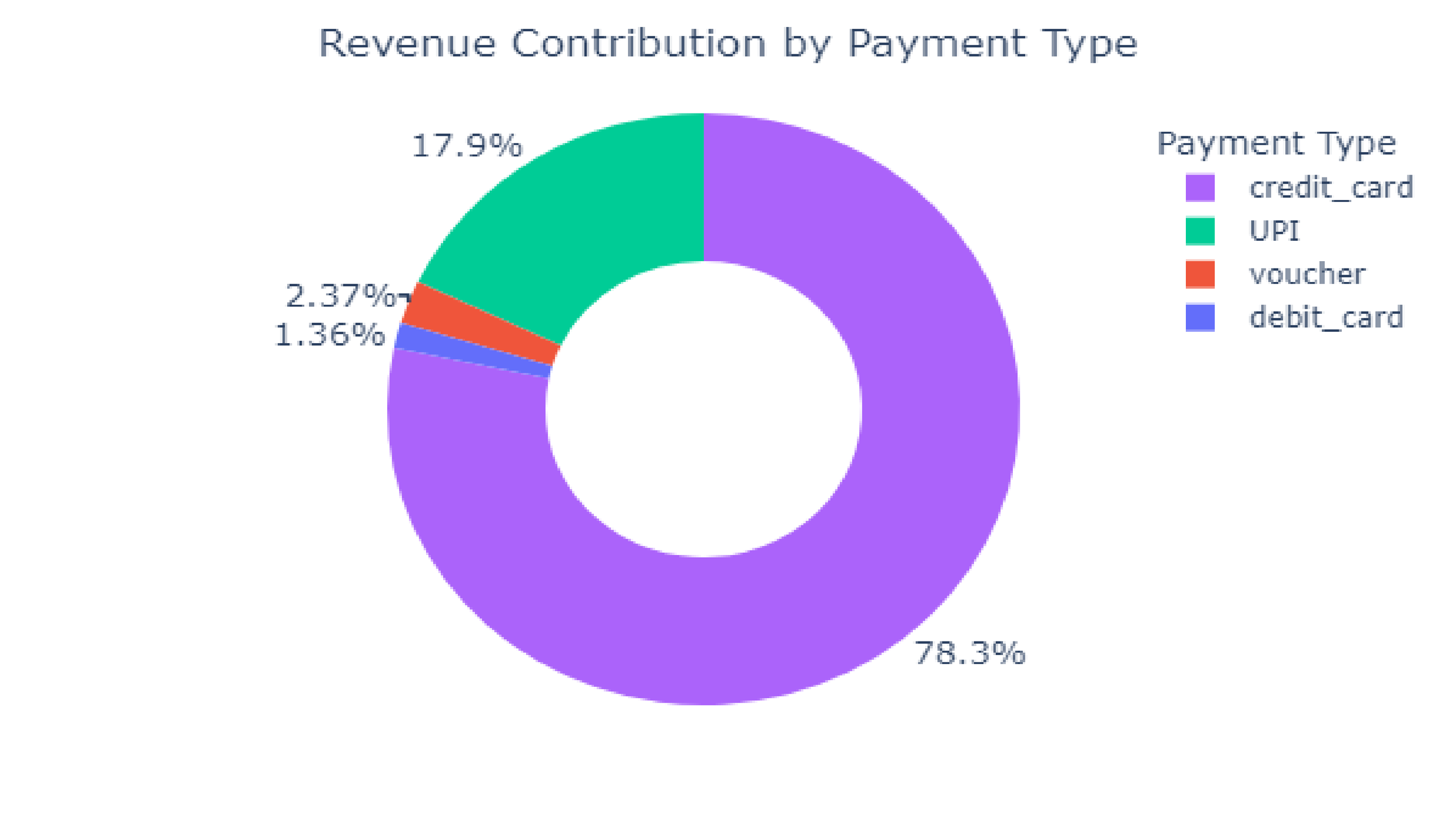

Top 10 Products by Quantity Sold
aca2eb7d00ea1a7b8ebd4e68314663af
527
99a4788cb24856965c36a24e339b6058
488
422879e10f46682990de24d770e7f83d
484
389d119b48cf3043d311335e499d9c6b
392
368c6c730842d78016ad823897a372db
388
53759a2ecddad2bb87a079a1f1519f73
373
d1c427060a0f73f6b889a5c7c61f2ac4
343
53b36df67ebb7c41585e8d54d6772e08
323
154e7e31ebfa092203795c972e5804a6
281
3dd2a17168ec895c781a9191c1e95ad7
274

[28]:
```
query='''
 SELECT
            p.payment_type,
            SUM(p.payment_value) AS total_revenue
        FROM
            payments p
        GROUP BY
            p.payment_type;
'''

data=pd.read_sql(query, db)
print(f'Revenue Contribution By Payment Type : \n\n{data}')
```

```
Revenue Contribution By Payment Type :

  payment_type  total_revenue
0  not_defined   0.000000e+00
1   debit_card   2.179898e+05
2      voucher   3.794369e+05
3          UPI   2.869361e+06
4  credit_card   1.254208e+07
```

[29]:

```
data= data[data['total_revenue']>0]
fig = px.pie( # Create the pie chart
    data,
    names='payment_type',
    values='total_revenue',
    title='Revenue Contribution by Payment Type',
    color='payment_type',  # Customize colors for different payment types
    hole=0.5,  # Create a donut chart by adding a hole in the center
    labels={'total_revenue': 'Total Revenue'},  # Customize labels
)
# Update layout to show percentages and values on the chart
fig.update_traces(
    textinfo='percent',  # Display label, percentage, and value
    textfont_size=14,  # Font size for the text
    textposition='outside'
)
# Update layout for better appearance
fig.update_layout(
    title={'text': 'Revenue Contribution by Payment Type', 'x': 0.5, 'xanchor': 'center'},  # Center the title
    legend_title='Payment Type',  # Title for the legend
    legend=dict(
        orientation='v',  # Horizontal orientation for the legend
        x=1,  # Center the legend
        xanchor='left'
    ),
    margin=dict(t=50, b=50, l=50, r=50)  # Adjust margins to make room for title and labels
)

# Show the plot
fig.show()
```


Revenue Contribution by Payment Type
17.9%
2.37%
1.36%
78.3%
Payment Type
credit_card
UPI
voucher
debit_card

Conclusions and Next Steps

This report highlights key areas for improvement and growth opportunities. By focusing on high-revenue categories, optimizing order fulfillment, enhancing payment processes, improving customer retention, and supporting seller performance, the business can drive increased profitability and customer satisfaction.

Next Steps:

- Implement the recommended strategies and monitor their impact on business performance.
- Conduct follow-up analyses to track progress and adjust strategies as needed.