Installation & Setup

```
!pip install plotly kaleido -q

import pandas as pd
import numpy as np
import matplotlib.pyplot as plt
import seaborn as sns
import plotly.express as px
import plotly.graph_objects as go
from plotly.subplots import make_subplots
from datetime import datetime
import warnings
warnings.filterwarnings('ignore')
sns.set_style('whitegrid')
plt.rcParams['figure.figsize'] = (14, 7)

print("Setup complete! All libraries loaded successfully.")
```

```
   ━━━━━━━━━━━━━━━━━━━━━━━━━━━━━━━━━━━━━━━━ 66.3/66.3 kB 3.0 MB/s eta 0:00:00
   ━━━━━━━━━━━━━━━━━━━━━━━━━━━━━━━━━━━━━━━━ 53.3/53.3 kB 3.5 MB/s eta 0:00:00
Setup complete! All libraries loaded successfully.
```

Data Loading with Progress

```
print("\n📥 Loading datasets...\n")

# Load required datasets
orders = pd.read_csv('/content/olist_orders_dataset.csv.zip')
print("   ✓ Orders loaded          -", f"{len(orders):,} records")

order_items = pd.read_csv('/content/olist_order_items_dataset.csv.zip')
print("   ✓ Order items loaded     -", f"{len(order_items):,} records")

customers = pd.read_csv('/content/olist_customers_dataset.csv.zip')
print("   ✓ Customers loaded       -", f"{len(customers):,} records")

products = pd.read_csv('/content/olist_products_dataset.csv.zip')
print("   ✓ Products loaded        -", f"{len(products):,} records")

payments = pd.read_csv('/content/olist_order_payments_dataset.csv.zip')
print("   ✓ Payments loaded        -", f"{len(payments):,} records")

# Load optional datasets
HAS_REVIEWS = False
try:
    reviews = pd.read_csv('/content/olist_order_reviews_dataset.csv.zip')
    HAS_REVIEWS = True
    print("   ✓ Reviews loaded         -", f"{len(reviews):,} records")
except:
    print("   ○ Reviews not available")

print("\n✅ All datasets loaded successfully!\n")
```

```
📥 Loading datasets...

   ✓ Orders loaded          - 99,441 records
   ✓ Order items loaded     - 112,650 records
   ✓ Customers loaded       - 99,441 records
   ✓ Products loaded        - 32,951 records
   ✓ Payments loaded        - 103,886 records
   ✓ Reviews loaded         - 99,224 records

✅ All datasets loaded successfully!
```

Data Cleaning

```
print("🧹 Cleaning and preprocessing data...\n")

# Convert date columns
date_cols = ['order_purchase_timestamp', 'order_approved_at',
             'order_delivered_carrier_date', 'order_delivered_customer_date',
             'order_estimated_delivery_date']

for col in date_cols:
    if col in orders.columns:
```

```
        orders[col] = pd.to_datetime(orders[col], errors='coerce')

print("   ✓ Date columns converted")

# Merge datasets
df = orders.merge(order_items, on='order_id', how='left')
df = df.merge(payments, on='order_id', how='left')
df = df.merge(customers, on='customer_id', how='left')
df = df.merge(products, on='product_id', how='left')

if HAS_REVIEWS:
    df = df.merge(reviews[['order_id', 'review_score']], on='order_id', how='left')

print("   ✓ Datasets merged")

# Clean data - keep only completed orders
initial_rows = len(df)
df = df[df['order_status'].isin(['delivered', 'shipped', 'invoiced'])]
removed_rows = initial_rows - len(df)

print(f"   ✓ Removed {removed_rows:,} incomplete orders")

# Calculate derived metrics
df['total_price'] = df['price'] + df['freight_value']
df['delivery_time'] = (df['order_delivered_customer_date'] -
                       df['order_purchase_timestamp']).dt.days
df['year_month'] = df['order_purchase_timestamp'].dt.to_period('M')

print("   ✓ Calculated derived metrics")

# Remove price outliers (top 1%)
initial_rows = len(df)
df = df[df['total_price'] < df['total_price'].quantile(0.99)]
outliers_removed = initial_rows - len(df)

print(f"   ✓ Removed {outliers_removed:,} price outliers")

print(f"\n✅ Data cleaning complete!")
print(f"   Final dataset shape: {df.shape[0]:,} rows × {df.shape[1]} columns")
```

```
🧹 Cleaning and preprocessing data...

   ✓ Date columns converted
   ✓ Datasets merged
   ✓ Removed 1,786 incomplete orders
   ✓ Calculated derived metrics
   ✓ Removed 1,177 price outliers

✅ Data cleaning complete!
   Final dataset shape: 116,180 rows × 34 columns
```

Data Quality Check

```
print("\n" + "="*80)
print("   📊 DATA QUALITY DASHBOARD")
print("="*80 + "\n")

# Date range
date_min = df['order_purchase_timestamp'].min()
date_max = df['order_purchase_timestamp'].max()
date_range_days = (date_max - date_min).days

print(f"📅 Date Range:")
print(f"   From: {date_min.strftime('%B %d, %Y')}")
print(f"   To:   {date_max.strftime('%B %d, %Y')}")
print(f"   Span: {date_range_days} days ({date_range_days/30:.1f} months)")

# Completeness check
print(f"\n📈 Data Completeness:")
completeness = (1 - df.isnull().sum() / len(df)) * 100
key_columns = ['total_price', 'customer_state', 'product_category_name',
               'delivery_time', 'payment_type']

for col in key_columns:
    if col in df.columns:
        pct = completeness[col]
        status = "✓" if pct > 90 else "⚠"
        print(f"   {status} {col:<30}: {pct:>5.1f}%")

overall_completeness = completeness.mean()
print(f"\n   Overall Data Quality: {overall_completeness:.1f}% {'✅' if overall_completeness > 85 else '⚠️'}")
```

```
================================================================================
  📊 DATA QUALITY DASHBOARD
================================================================================

📅 Date Range:
   From: September 04, 2016
   To:   September 03, 2018
   Span: 728 days (24.3 months)

📈 Data Completeness:
   ✓ total_price                  : 100.0%
   ✓ customer_state               : 100.0%
   ✓ product_category_name        :  98.6%
   ✓ delivery_time                :  98.6%
   ✓ payment_type                 : 100.0%

   Overall Data Quality: 99.7% ✅
```

Executive Summary Generation

```
print("\n" + "="*80)
print("  🎯 EXECUTIVE SUMMARY - AUTO GENERATED")
print("="*80 + "\n")

# Calculate key metrics
total_revenue = df['total_price'].sum()
total_orders = df['order_id'].nunique()
avg_order_value = total_revenue / total_orders
total_customers = df['customer_unique_id'].nunique()
avg_delivery_time = df['delivery_time'].mean()

# Calculate monthly metrics for trend
monthly_data = df.groupby('year_month').agg({
    'total_price': 'sum',
    'order_id': 'nunique'
}).reset_index()

# Calculate growth using 3-month rolling average
if len(monthly_data) >= 6:
    # Last 3 months average
    recent_avg = monthly_data['total_price'].iloc[-3:].mean()
    # Previous 3 months average
    prev_avg = monthly_data['total_price'].iloc[-6:-3].mean()

    revenue_growth = ((recent_avg - prev_avg) / prev_avg * 100)

    recent_orders_avg = monthly_data['order_id'].iloc[-3:].mean()
    prev_orders_avg = monthly_data['order_id'].iloc[-6:-3].mean()
    order_growth = ((recent_orders_avg - prev_orders_avg) / prev_orders_avg * 100)
else:
    revenue_growth = 0
    order_growth = 0

# Display metrics in cards
print("┌" + "─"*76 + "┐")
print("│" + " "*28 + "KEY METRICS" + " "*37 + "│")
print("├" + "─"*76 + "┤")
print(f"│  💰 Total Revenue              │  R$ {total_revenue:>20,.2f}              │")
print(f"│  📦 Total Orders               │  {total_orders:>27,}              │")
print(f"│  💵 Average Order Value        │  R$ {avg_order_value:>20,.2f}              │")
print(f"│  👥 Total Customers            │  {total_customers:>27,}              │")
print(f"│  🚚 Avg Delivery Time          │  {avg_delivery_time:>24,.1f} days              │")
print("└" + "─"*76 + "┘")

# Growth indicators
print("\n┌" + "─"*76 + "┐")
print("│" + " "*26 + "GROWTH TRENDS" + " "*37 + "│")
print("├" + "─"*76 + "┤")

revenue_arrow = "📈" if revenue_growth > 0 else "📉"
order_arrow = "📈" if order_growth > 0 else "📉"

print(f"│  {revenue_arrow} Revenue Growth (MoM)      │  {revenue_growth:>+24,.1f}%              │")
print(f"│  {order_arrow} Order Growth (MoM)        │  {order_growth:>+24,.1f}%              │")
print("└" + "─"*76 + "┘")

# Auto-generated insights
print("\n📌 KEY INSIGHTS:\n")

insights_generated = []
```

```
# Insight 1: Revenue trend
if revenue_growth > 5:
    insights_generated.append(f"   ✓ Strong revenue growth of {revenue_growth:.1f}% - business is scaling well")
elif revenue_growth < -5:
    insights_generated.append(f"   ⚠ Revenue declined {abs(revenue_growth):.1f}% - investigate seasonality or market factors")
else:
    insights_generated.append(f"   ○ Revenue stable at {revenue_growth:.1f}% growth - focus on optimization")

# Insight 2: AOV
if avg_order_value > 150:
    insights_generated.append(f"   ✓ High AOV of R$ {avg_order_value:.2f} - customers buying premium products")
else:
    insights_generated.append(f"   💡 AOV at R$ {avg_order_value:.2f} - opportunity for upselling strategies")

# Insight 3: Delivery
if avg_delivery_time < 12:
    insights_generated.append(f"   ✓ Fast delivery ({avg_delivery_time:.1f} days) - competitive advantage")
elif avg_delivery_time > 20:
    insights_generated.append(f"   ⚠ Slow delivery ({avg_delivery_time:.1f} days) - optimize logistics")
else:
    insights_generated.append(f"   ○ Standard delivery time ({avg_delivery_time:.1f} days) - meets expectations")

for insight in insights_generated:
    print(insight)

print("\n" + "="*80)
```

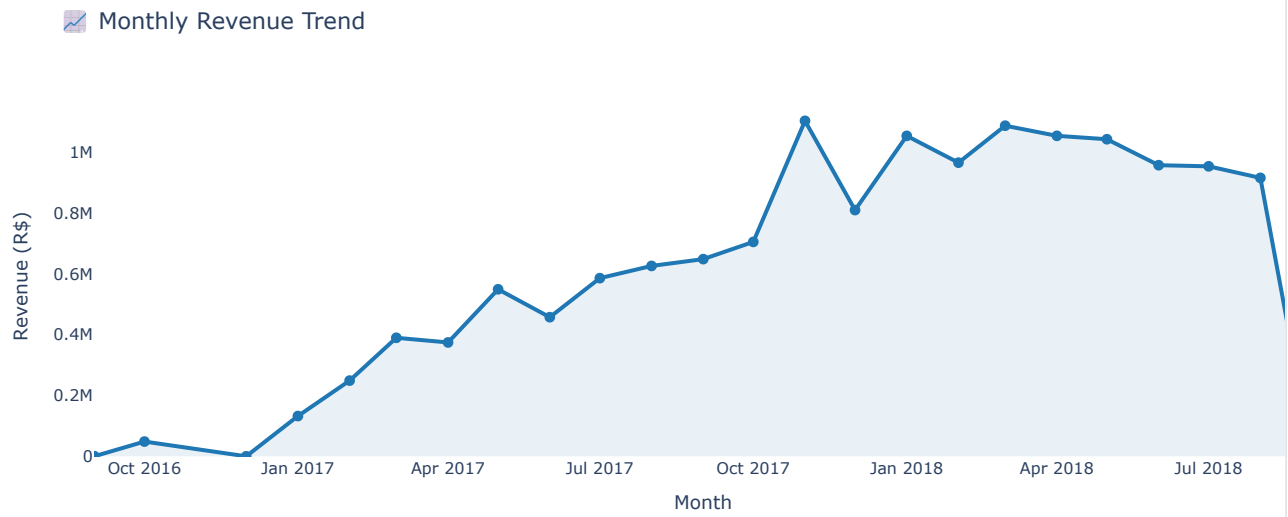
```
================================================================================
  🎯 EXECUTIVE SUMMARY - AUTO GENERATED
================================================================================

┌──────────────────────────────────────────────────────────────────────────┐
│                              KEY METRICS                                 │
├──────────────────────────────────────────────────────────────────────────┤
│  💰 Total Revenue              │  R$        14,715,991.61              │
│  📦 Total Orders               │                     96,839              │
│  💵 Average Order Value        │  R$               151.96               │
│  👥 Total Customers            │                     93,665              │
│  🚚 Avg Delivery Time          │                    12.0 days            │
└──────────────────────────────────────────────────────────────────────────┘

┌──────────────────────────────────────────────────────────────────────────┐
│                             GROWTH TRENDS                                │
├──────────────────────────────────────────────────────────────────────────┤
│  📉 Revenue Growth (MoM)       │                    -38.8%               │
│  📉 Order Growth (MoM)         │                    -36.3%               │
└──────────────────────────────────────────────────────────────────────────┘

📌 KEY INSIGHTS:

   ⚠ Revenue declined 38.8% - investigate seasonality or market factors
   ✓ High AOV of R$ 151.96 - customers buying premium products
   ✓ Fast delivery (12.0 days) - competitive advantage

================================================================================
```

Revenue Trend Analysis

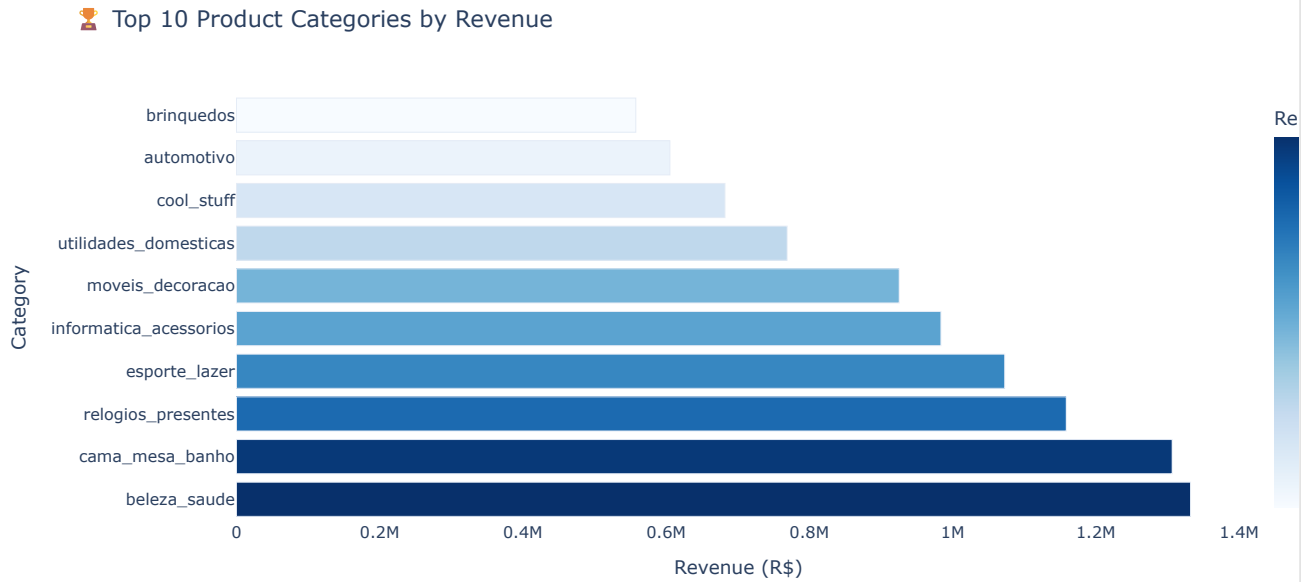
```
print("\n" + "="*80)
print("  📈 REVENUE & SALES ANALYSIS")
print("="*80 + "\n")

# Monthly revenue trend
monthly_rev = df.groupby('year_month').agg({
    'total_price': 'sum',
    'order_id': 'nunique'
}).reset_index()
monthly_rev.columns = ['month', 'revenue', 'orders']
monthly_rev['month'] = monthly_rev['month'].astype(str)

# Find peak month
peak_month_idx = monthly_rev['revenue'].idxmax()
peak_month = monthly_rev.loc[peak_month_idx, 'month']
peak_revenue = monthly_rev.loc[peak_month_idx, 'revenue']

print(f"📊 Peak Performance:")
print(f"   Best Month: {peak_month}")
print(f"   Revenue: R$ {peak_revenue:,.2f}\n")

# Create revenue trend chart
fig1 = go.Figure()

fig1.add_trace(go.Scatter(
```

```
    x=monthly_rev['month'],
    y=monthly_rev['revenue'],
    mode='lines+markers',
    name='Revenue',
    line=dict(color='#1f77b4', width=3),
    marker=dict(size=8),
    fill='tozeroy',
    fillcolor='rgba(31, 119, 180, 0.1)'
))

fig1.update_layout(
    title='📈 Monthly Revenue Trend',
    xaxis_title='Month',
    yaxis_title='Revenue (R$)',
    height=450,
    hovermode='x unified',
    plot_bgcolor='white'
)

fig1.show()

# Top product categories
print("\n🏆 Top Performing Categories:\n")
top_categories = df.groupby('product_category_name').agg({
    'total_price': 'sum',
    'order_id': 'nunique'
}).reset_index()
top_categories.columns = ['category', 'revenue', 'orders']
top_categories = top_categories.nlargest(10, 'revenue')

for idx, row in top_categories.head(5).iterrows():
    pct_of_total = (row['revenue'] / total_revenue * 100)
    print(f"   {idx+1}. {row['category'][:35]:<35} - R$ {row['revenue']:>12,.2f} ({pct_of_total:>4.1f}%)")

# Category chart
fig2 = px.bar(
    top_categories,
    y='category',
    x='revenue',
    orientation='h',
    title='🏆 Top 10 Product Categories by Revenue',
    labels={'revenue': 'Revenue (R$)', 'category': 'Category'},
    color='revenue',
    color_continuous_scale='Blues'
)

fig2.update_layout(
    height=500,
    showlegend=False,
    plot_bgcolor='white'
)

fig2.show()

print("\n💡 INSIGHT: Focus inventory and marketing on top 3 categories for maximum ROI")
```

```
================================================================================
📈 REVENUE & SALES ANALYSIS
================================================================================

📊 Peak Performance:
   Best Month: 2017-11
   Revenue: R$ 1,103,676.64
```




```
🏆 Top Performing Categories:

   12. beleza_saude                      - R$ 1,331,990.16 ( 9.1%)
   14. cama_mesa_banho                   - R$ 1,306,374.00 ( 8.9%)
   67. relogios_presentes                - R$ 1,158,535.31 ( 7.9%)
   33. esporte_lazer                     - R$ 1,072,529.59 ( 7.3%)
   45. informatica_acessorios            - R$   983,497.89 ( 6.7%)
```



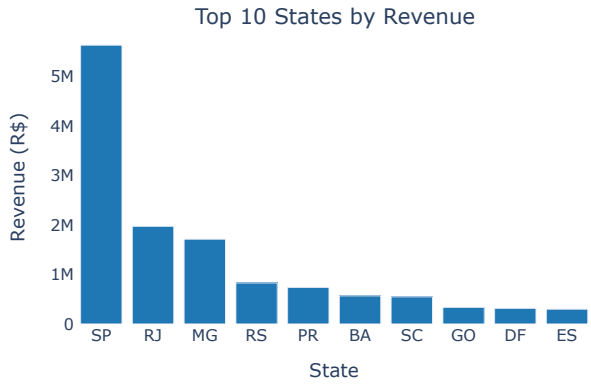

```
💡 INSIGHT: Focus inventory and marketing on top 3 categories for maximum ROI
```

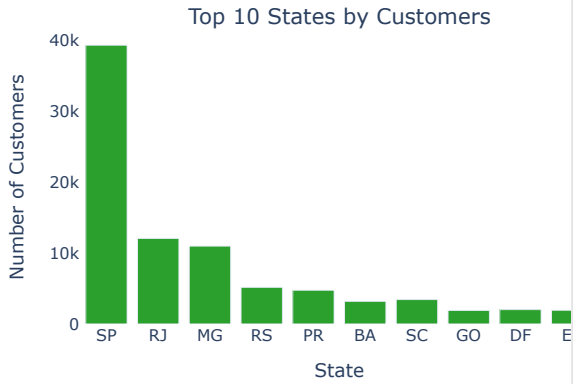
Geographic & Payment Analysis

```
print("\n" + "="*80)
print("   🗺️ GEOGRAPHIC & PAYMENT ANALYSIS")
print("="*80 + "\n")

# Customer distribution by state
customer_state = df.groupby('customer_state').agg({
    'customer_unique_id': 'nunique',
    'total_price': 'sum',
    'order_id': 'nunique'
}).reset_index()
```

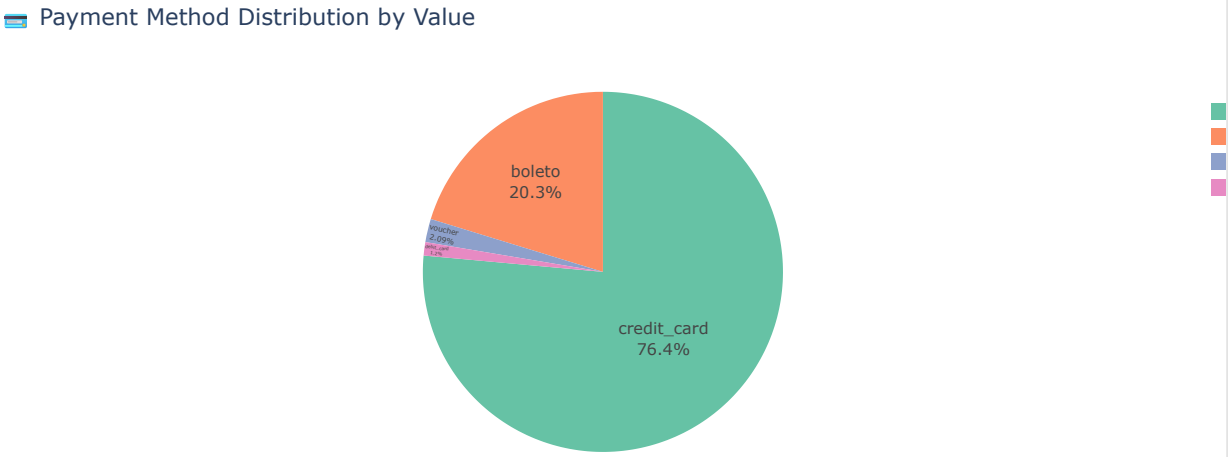
```
customer_state.columns = ['state', 'customers', 'revenue', 'orders']
customer_state = customer_state.nlargest(10, 'revenue')

print("📍 Top 5 States by Revenue:\n")
for idx, row in customer_state.head(5).iterrows():
    print(f"   {row['state']}: R$ {row['revenue']:>12,.2f} | {row['customers']:>6,} customers | {row['orders']:>6,} orders")

# Geographic visualization
fig3 = make_subplots(
    rows=1, cols=2,
    subplot_titles=('Top 10 States by Revenue', 'Top 10 States by Customers'),
    horizontal_spacing=0.15
)

fig3.add_trace(
    go.Bar(x=customer_state['state'], y=customer_state['revenue'],
           marker_color='#1f77b4', name='Revenue'),
    row=1, col=1
)

fig3.add_trace(
    go.Bar(x=customer_state['state'], y=customer_state['customers'],
           marker_color='#2ca02c', name='Customers'),
    row=1, col=2
)

fig3.update_layout(height=400, showlegend=False, plot_bgcolor='white')
fig3.update_xaxes(title_text="State", row=1, col=1)
fig3.update_xaxes(title_text="State", row=1, col=2)
fig3.update_yaxes(title_text="Revenue (R$)", row=1, col=1)
fig3.update_yaxes(title_text="Number of Customers", row=1, col=2)

fig3.show()

# Payment analysis
print("\n💳 Payment Method Analysis:\n")
payment_dist = df.groupby('payment_type').agg({
    'payment_value': 'sum',
    'order_id': 'nunique'
}).reset_index()
payment_dist.columns = ['payment_type', 'total_value', 'num_orders']
payment_dist = payment_dist.sort_values('total_value', ascending=False)

for idx, row in payment_dist.iterrows():
    pct = (row['total_value'] / payment_dist['total_value'].sum() * 100)
    print(f"   {row['payment_type'].title():<20} - R$ {row['total_value']:>12,.2f} ({pct:>5.1f}%)")

# Payment pie chart
fig4 = px.pie(
    payment_dist,
    values='total_value',
    names='payment_type',
    title='💳 Payment Method Distribution by Value',
    color_discrete_sequence=px.colors.qualitative.Set2
)

fig4.update_traces(textposition='inside', textinfo='percent+label')
fig4.update_layout(height=450)
fig4.show()

# Identify dominant payment
dominant_payment = payment_dist.iloc[0]['payment_type']
dominant_pct = (payment_dist.iloc[0]['total_value'] / payment_dist['total_value'].sum() * 100)

print(f"\n💡 INSIGHT: {dominant_payment.title()} dominates at {dominant_pct:.1f}% - ensure this channel is optimized")
```

```
================================================================================
  🗺️ GEOGRAPHIC & PAYMENT ANALYSIS
================================================================================

📍 Top 5 States by Revenue:

   SP: R$ 5,630,149.86 | 39,313 customers | 40,670 orders
   RJ: R$ 1,971,814.91 | 12,064 customers | 12,517 orders
   MG: R$ 1,711,491.73 | 10,981 customers | 11,336 orders
   RS: R$   828,271.66 |  5,164 customers |  5,345 orders
   PR: R$   738,041.46 |  4,755 customers |  4,912 orders
```






```
💳 Payment Method Analysis:

   Credit_Card          - R$ 13,999,536.17 ( 76.4%)
   Boleto               - R$ 3,710,354.55 ( 20.3%)
   Voucher              - R$   383,041.34 (  2.1%)
   Debit_Card           - R$   219,796.84 (  1.2%)
```



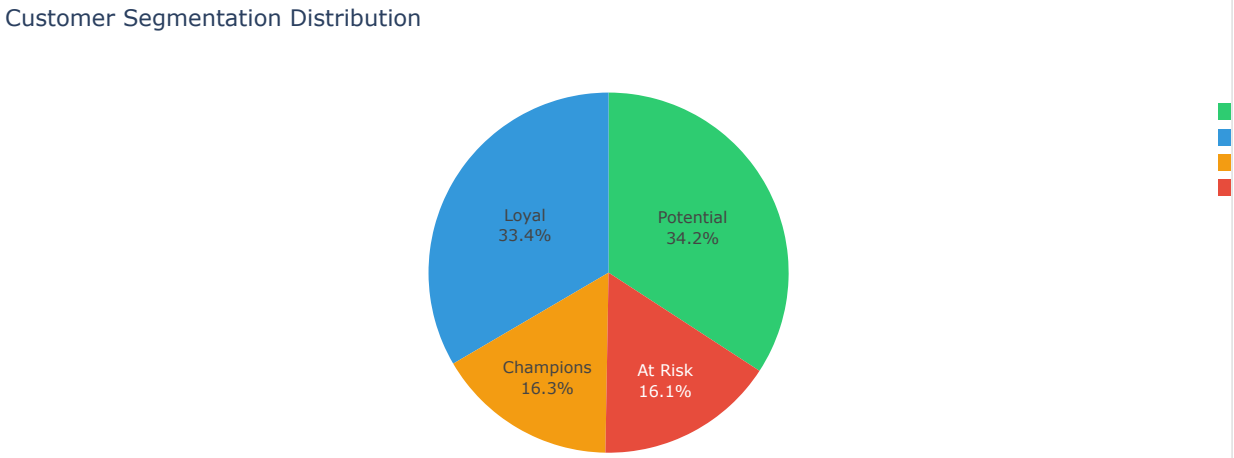

```
💡 INSIGHT: Credit_Card dominates at 76.4% - ensure this channel is optimized
```

RFM Customer Segmentation

```
print("\n" + "="*80)
print("  👥 CUSTOMER SEGMENTATION - RFM ANALYSIS")
print("="*80 + "\n")

print("🔄 Calculating RFM scores...\n")

# Calculate RFM metrics
analysis_date = df['order_purchase_timestamp'].max()

rfm = df.groupby('customer_unique_id').agg({
    'order_purchase_timestamp': lambda x: (analysis_date - x.max()).days,
    'order_id': 'nunique',
    'total_price': 'sum'
```

```
}).reset_index()

rfm.columns = ['customer_id', 'recency', 'frequency', 'monetary']

# Create RFM scores (1-4 scale)
rfm['R_score'] = pd.qcut(rfm['recency'], 4, labels=[4, 3, 2, 1])
rfm['F_score'] = pd.qcut(rfm['frequency'].rank(method='first'), 4, labels=[1, 2, 3, 4])
rfm['M_score'] = pd.qcut(rfm['monetary'], 4, labels=[1, 2, 3, 4])

rfm['RFM_Score'] = (rfm['R_score'].astype(int) +
                    rfm['F_score'].astype(int) +
                    rfm['M_score'].astype(int))

# Segment customers
def segment_customers(score):
    if score >= 10:
        return 'Champions'
    elif score >= 8:
        return 'Loyal'
    elif score >= 6:
        return 'Potential'
    else:
        return 'At Risk'

rfm['segment'] = rfm['RFM_Score'].apply(segment_customers)

# Display segment distribution
segment_stats = rfm.groupby('segment').agg({
    'customer_id': 'count',
    'monetary': 'mean',
    'frequency': 'mean',
    'recency': 'mean'
}).reset_index()
segment_stats.columns = ['segment', 'count', 'avg_monetary', 'avg_frequency', 'avg_recency']
segment_stats = segment_stats.sort_values('count', ascending=False)

print("📊 Customer Segment Distribution:\n")
print("┌" + "─"*78 + "┐")
print(f"│ {'Segment':<18} │ {'Count':>10} │ {'Avg Spend':>15} │ {'Avg Orders':>12} │ {'Avg Recency':>12} │")
print("├" + "─"*78 + "┤")

for idx, row in segment_stats.iterrows():
    pct = (row['count'] / len(rfm) * 100)
    print(f"│ {row['segment']:<18} │ {row['count']:>10,} │ R$ {row['avg_monetary']:>11,.2f} │ {row['avg_frequency']:>12.1f} │

print("└" + "─"*78 + "┘")

# Segment pie chart
segment_dist = rfm['segment'].value_counts()

fig5 = px.pie(
    values=segment_dist.values,
    names=segment_dist.index,
    title='Customer Segmentation Distribution',
    color_discrete_sequence=['#2ecc71', '#3498db', '#f39c12', '#e74c3c']
)

fig5.update_traces(textposition='inside', textinfo='percent+label')
fig5.update_layout(height=450)
fig5.show()

# Segment-specific insights
champions_count = segment_dist.get('Champions', 0)
at_risk_count = segment_dist.get('At Risk', 0)
champions_pct = (champions_count / len(rfm) * 100)
at_risk_pct = (at_risk_count / len(rfm) * 100)

print(f"\nKEY INSIGHTS:")
print(f"   • {champions_pct:.1f}% Champions - Reward with VIP programs and exclusive offers")
print(f"   • {at_risk_pct:.1f}% At Risk - Launch re-engagement campaigns immediately")
print(f"   • Focus: Convert Potential customers to Loyal through targeted promotions")
```

```
================================================================================
  👥 CUSTOMER SEGMENTATION - RFM ANALYSIS
================================================================================

🔄 Calculating RFM scores...

📊 Customer Segment Distribution:

┌──────────────────────────────────────────────────────────────────────────┐
│ Segment            │      Count │       Avg Spend │   Avg Orders │  Avg Recency │
├──────────────────────────────────────────────────────────────────────────┤
│ Potential          │     32,016 │ R$      119.87 │          1.0 │       278 days │
│ Loyal              │     31,315 │ R$      182.74 │          1.0 │       200 days │
│ Champions          │     15,254 │ R$      273.15 │          1.1 │       120 days │
│ At Risk            │     15,080 │ R$       65.60 │          1.0 │       376 days │
└──────────────────────────────────────────────────────────────────────────┘
```




```
KEY INSIGHTS:
   • 16.3% Champions - Reward with VIP programs and exclusive offers
   • 16.1% At Risk - Launch re-engagement campaigns immediately
   • Focus: Convert Potential customers to Loyal through targeted promotions
```

Business Recommendations

```
print("\n" + "="*80)
print("  🎯 ACTIONABLE BUSINESS RECOMMENDATIONS")
print("="*80 + "\n")

recommendations = []

# Recommendation 1: Revenue Growth
if revenue_growth > 5:
    rec1 = {
        'priority': '🟢 HIGH',
        'category': 'Growth Strategy',
        'recommendation': f'Sustain momentum with {revenue_growth:.1f}% growth',
        'actions': [
            'Scale marketing in high-performing categories',
            'Expand successful campaigns to new regions',
            'Increase inventory for top products'
        ]
    }
elif revenue_growth < -5:
    rec1 = {
        'priority': '🔴 CRITICAL',
        'category': 'Revenue Recovery',
        'recommendation': f'Address {abs(revenue_growth):.1f}% revenue decline',
        'actions': [
            'Investigate root causes (seasonality, competition, quality)',
            'Launch aggressive promotional campaigns',
            'Re-engage dormant customers with win-back offers'
        ]
    }
else:
    rec1 = {
        'priority': '🟡 MEDIUM',
        'category': 'Optimization',
```

```
        'recommendation': 'Revenue stable - focus on efficiency',
        'actions': [
            'Optimize pricing strategy for higher margins',
            'Reduce operational costs',
            'Test upselling and cross-selling tactics'
        ]
    }
recommendations.append(rec1)

# Recommendation 2: Customer Segments
rec2 = {
    'priority': '🟢 HIGH',
    'category': 'Customer Retention',
    'recommendation': f'Activate segmented marketing strategies',
    'actions': [
        f'Champions ({champions_pct:.0f}%): Exclusive loyalty rewards, early access',
        f'At Risk ({at_risk_pct:.0f}%): Personalized email campaigns, discount codes',
        'Potential: Onboarding sequences, product recommendations'
    ]
}
recommendations.append(rec2)

# Recommendation 3: Geographic Expansion
top_state = customer_state.iloc[0]['state']
top_state_revenue_pct = (customer_state.iloc[0]['revenue'] / total_revenue * 100)

rec3 = {
    'priority': '🟡 MEDIUM',
    'category': 'Market Expansion',
    'recommendation': f'{top_state} dominates ({top_state_revenue_pct:.1f}% of revenue)',
    'actions': [
        f'Replicate {top_state} success in underperforming states',
        'Launch localized campaigns in top 5 states',
        'Analyze barriers in low-performing regions'
    ]
}
recommendations.append(rec3)

# Recommendation 4: Operations
if avg_delivery_time > 15:
    rec4 = {
        'priority': '🔴 CRITICAL',
        'category': 'Logistics',
        'recommendation': f'Delivery time ({avg_delivery_time:.1f} days) exceeds expectations',
        'actions': [
            'Partner with faster logistics providers',
            'Optimize warehouse locations',
            'Implement regional fulfillment centers'
        ]
    }
    recommendations.append(rec4)

# Display recommendations
for idx, rec in enumerate(recommendations, 1):
    print(f"{rec['priority']} PRIORITY #{idx}: {rec['category'].upper()}")
    print(f"{'-'*78}")
    print(f"📋 {rec['recommendation']}\n")
    print(f"✅ Action Items:")
    for action in rec['actions']:
        print(f"   • {action}")
    print("\n")

print("="*80)
```

```
================================================================================
  🎯 ACTIONABLE BUSINESS RECOMMENDATIONS
================================================================================

🔴 CRITICAL PRIORITY #1: REVENUE RECOVERY
------------------------------------------------------------------------------
📋 Address 38.8% revenue decline

✅ Action Items:
   • Investigate root causes (seasonality, competition, quality)
   • Launch aggressive promotional campaigns
   • Re-engage dormant customers with win-back offers


🟢 HIGH PRIORITY #2: CUSTOMER RETENTION
------------------------------------------------------------------------------
📋 Activate segmented marketing strategies

✅ Action Items:
```

```
    • Champions (16%): Exclusive loyalty rewards, early access
    • At Risk (16%): Personalized email campaigns, discount codes
    • Potential: Onboarding sequences, product recommendations


🟡 MEDIUM PRIORITY #3: MARKET EXPANSION
────────────────────────────────────────────────────────────────────────────────
📋 SP dominates (38.3% of revenue)

✅ Action Items:
    • Replicate SP success in underperforming states
    • Launch localized campaigns in top 5 states
    • Analyze barriers in low-performing regions


================================================================================
```

Export Package

```
from google.colab import files
import json

# Create Excel writer object
excel_filename = 'E-Commerce_Business_Report.xlsx'

with pd.ExcelWriter(excel_filename, engine='openpyxl') as writer:

    # Sheet 1: Executive Summary
    summary_data = {
        'Metric': [
            'Total Revenue (R$)',
            'Total Orders',
            'Average Order Value (R$)',
            'Total Customers',
            'Average Delivery Time (days)',
            'Revenue Growth MoM (%)',
            'Order Growth MoM (%)',
            'Top Product Category',
            'Top State',
            'Dominant Payment Method',
            'Champions Customers (%)',
            'Loyal Customers (%)',
            'Potential Customers (%)',
            'At Risk Customers (%)',
            'Data Date Range',
            'Overall Data Quality (%)',
            'Analysis Date'
        ],
        'Value': [
            f"{total_revenue:,.2f}",
            f"{total_orders:,}",
            f"{avg_order_value:,.2f}",
            f"{total_customers:,}",
            f"{avg_delivery_time:.1f}",
            f"{revenue_growth:+.1f}",
            f"{order_growth:+.1f}",
            top_categories.iloc[0]['category'],
            customer_state.iloc[0]['state'],
            dominant_payment,
            f"{champions_pct:.1f}",
            f"{(segment_dist.get('Loyal', 0) / len(rfm) * 100):.1f}",
            f"{(segment_dist.get('Potential', 0) / len(rfm) * 100):.1f}",
            f"{at_risk_pct:.1f}",
            f"{date_min.strftime('%Y-%m-%d')} to {date_max.strftime('%Y-%m-%d')}",
            f"{overall_completeness:.1f}",
            datetime.now().strftime('%Y-%m-%d %H:%M:%S')
        ]
    }
    summary_df = pd.DataFrame(summary_data)
    summary_df.to_excel(writer, sheet_name='Executive Summary', index=False)


    # Sheet 2: Monthly Performance
    monthly_export = monthly_rev.copy()
    monthly_export.columns = ['Month', 'Revenue (R$)', 'Orders']
    monthly_export.to_excel(writer, sheet_name='Monthly Performance', index=False)


    # Sheet 3: Top Products/Categories
    top_products_export = top_categories.copy()
    top_products_export.columns = ['Category', 'Revenue (R$)', 'Orders']
    top_products_export.to_excel(writer, sheet_name='Top Categories', index=False)
```

```
    # Sheet 4: Geographic Analysis
    geographic_export = customer_state.copy()
    geographic_export.columns = ['State', 'Customers', 'Revenue (R$)', 'Orders']
    geographic_export.to_excel(writer, sheet_name='Geographic Analysis', index=False)


    # Sheet 5: Payment Methods
    payment_export = payment_dist.copy()
    payment_export.columns = ['Payment Type', 'Total Value (R$)', 'Number of Orders']
    payment_export['Percentage'] = (payment_export['Total Value (R$)'] / payment_export['Total Value (R$)'].sum() * 100).roun
    payment_export.to_excel(writer, sheet_name='Payment Analysis', index=False)


    # Sheet 6: Customer Segments Summary
    segment_summary = segment_stats.copy()
    segment_summary.columns = ['Segment', 'Customer Count', 'Avg Spend (R$)', 'Avg Orders', 'Avg Recency (days)']
    segment_summary['Percentage'] = (segment_summary['Customer Count'] / segment_summary['Customer Count'].sum() * 100).round
    segment_summary.to_excel(writer, sheet_name='Customer Segments', index=False)


    # Sheet 7: Customer Segment Details (Top 1000 customers)
    segment_details = rfm[['customer_id', 'recency', 'frequency', 'monetary', 'RFM_Score', 'segment']].copy()
    segment_details.columns = ['Customer ID', 'Recency (days)', 'Frequency', 'Monetary (R$)', 'RFM Score', 'Segment']
    segment_details = segment_details.sort_values('Monetary (R$)', ascending=False).head(1000)
    segment_details.to_excel(writer, sheet_name='Top 1000 Customers', index=False)


    # Sheet 8: Business Recommendations
    recommendations_list = []
    for idx, rec in enumerate(recommendations, 1):
        recommendations_list.append({
            'Priority': rec['priority'],
            'Category': rec['category'],
            'Recommendation': rec['recommendation'],
            'Action Items': ' | '.join(rec['actions'])
        })

    recommendations_df = pd.DataFrame(recommendations_list)
    recommendations_df.to_excel(writer, sheet_name='Recommendations', index=False)


    # Sheet 9: Key Insights
    insights_list = []

    # Add all auto-generated insights
    for insight in insights_generated:
        insight_clean = insight.strip().replace('   ', '')
        insights_list.append({'Insight': insight_clean})

    # Add segment insights
    insights_list.append({'Insight': f"Champions ({champions_pct:.1f}%): Reward with VIP programs and exclusive offers"})
    insights_list.append({'Insight': f"At Risk ({at_risk_pct:.1f}%): Launch re-engagement campaigns immediately"})
    insights_list.append({'Insight': f"Top category '{top_categories.iloc[0]['category']}' generates {(top_categories.iloc[0]
    insights_list.append({'Insight': f"{dominant_payment.title()} dominates payment methods at {dominant_pct:.1f}%"})

    insights_df = pd.DataFrame(insights_list)
    insights_df.to_excel(writer, sheet_name='Key Insights', index=False)


# Download the single Excel file
files.download(excel_filename)

print("\n" + "="*80)
print("   ✅ ANALYSIS COMPLETE!")
print("="*80)
print("\n📊 Summary:")
print(f"   • Analyzed {df.shape[0]:,} transactions")
print(f"   • Segmented {len(rfm):,} customers")
print(f"   • Generated {len(recommendations)} strategic recommendations")
print(f"   • Exported 1 comprehensive Excel file with 9 sheets")
```

```
================================================================================
   ✅ ANALYSIS COMPLETE!
================================================================================

📊 Summary:
   • Analyzed 116,180 transactions
   • Segmented 93,665 customers
   • Generated 3 strategic recommendations
   • Exported 1 comprehensive Excel file with 9 sheets
```

Output

```
from IPython.display import display, HTML

# Combine key metrics and growth trends from the executive summary
executive_summary_html = summary_df.to_html(index=False)

# Combine customer segment distribution
segment_stats_html = segment_stats.to_html(index=False)

# Combine top categories
top_categories_html = top_categories.to_html(index=False)

# Combine top states by revenue and customers
customer_state_html = customer_state.to_html(index=False)

# Combine payment method analysis
payment_dist_html = payment_dist.to_html(index=False)

# Get plotly figures as HTML
fig1_html = fig1.to_html(full_html=False, include_plotlyjs='cdn')
fig2_html = fig2.to_html(full_html=False, include_plotlyjs='cdn')
fig3_html = fig3.to_html(full_html=False, include_plotlyjs='cdn')
fig4_html = fig4.to_html(full_html=False, include_plotlyjs='cdn')
fig5_html = fig5.to_html(full_html=False, include_plotlyjs='cdn')


# Create a single HTML output
all_outputs_html = f"""
<h1>E-Commerce Business Analysis Report</h1>

<h2>Executive Summary</h2>
{executive_summary_html}

<h2>Revenue & Sales Analysis</h2>
<h3>Monthly Revenue Trend</h3>
{fig1_html}
<h3>Top Performing Categories</h3>
{top_categories_html}
{fig2_html}


<h2>Geographic & Payment Analysis</h2>
<h3>Top States by Revenue and Customers</h3>
{customer_state_html}
{fig3_html}
<h3>Payment Method Analysis</h3>
{payment_dist_html}
{fig4_html}

<h2>Customer Segmentation</h2>
<h3>Customer Segment Distribution</h3>
{segment_stats_html}
{fig5_html}

"""

display(HTML(all_outputs_html))
```

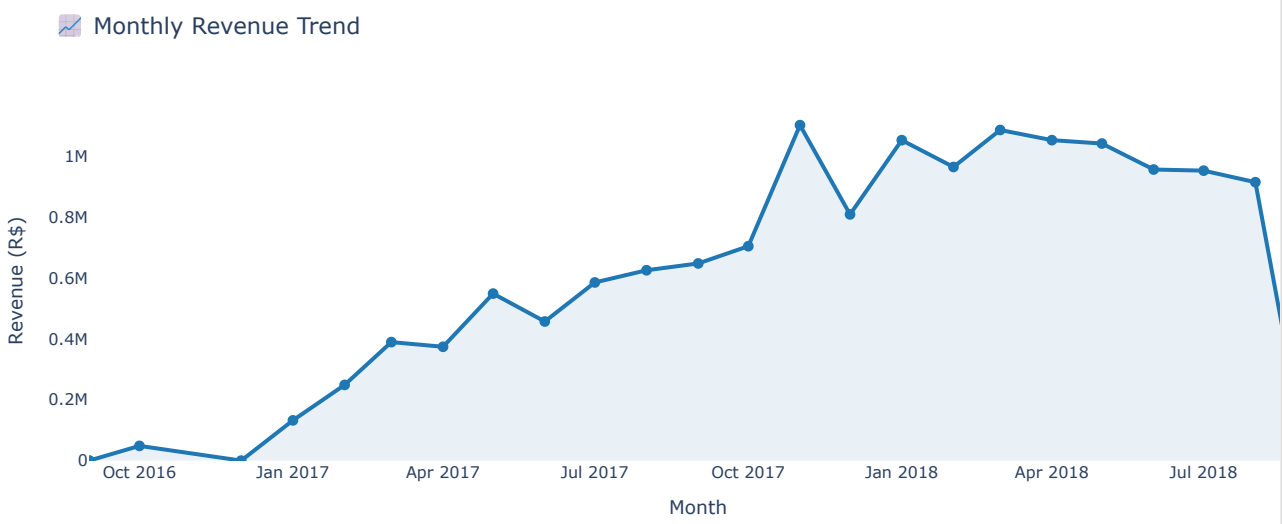
E-Commerce Business Analysis Report

Executive Summary

Metric	Value
Total Revenue (R$)	14,715,991.61
Total Orders	96,839
Average Order Value (R$)	151.96
Total Customers	93,665
Average Delivery Time (days)	12.0
Revenue Growth MoM (%)	-38.8
Order Growth MoM (%)	-36.3
Top Product Category	beleza_saude
Top State	SP
Dominant Payment Method	credit_card
Champions Customers (%)	16.3
Loyal Customers (%)	33.4
Potential Customers (%)	34.2
At Risk Customers (%)	16.1
Data Date Range	2016-09-04 to 2018-09-03
Overall Data Quality (%)	99.7
Analysis Date	2025-10-08 05:00:35

Revenue & Sales Analysis

Monthly Revenue Trend



Top Performing Categories

category	revenue	orders
beleza_saude	1331990.16	8679
cama_mesa_banho	1306374.00	9379
relogios_presentes	1158535.31	5451
esporte_lazer	1072529.59	7580
informatica_acessorios	983497.89	6579
moveis_decoracao	925268.77	6386
utilidades_domesticas	768883.65	5804
cool_stuff	682012.40	3575
automotivo	605150.20	3799