```
(0.0, 1.0, 0.0, 1.0)
```

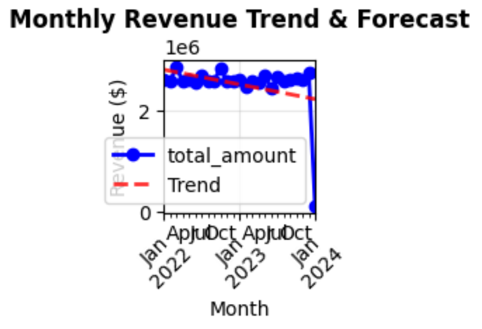
KEY PERFORMANCE

INDICATORS

Revenue: $63,087,441

Orders: 50,000

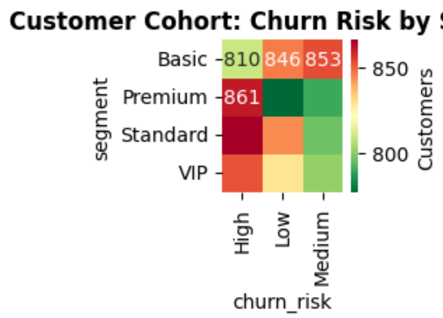
```
plt.subplot(3, 4, 2)
monthly_revenue = sales_df.groupby(sales_df['order_date'].dt.to_period('M'))['total_amount'].sum()
monthly_revenue.plot(kind='line', marker='o', color='blue', linewidth=2)
x_trend = range(len(monthly_revenue))
z = np.polyfit(x_trend, monthly_revenue.values, 1)
p = np.poly1d(z)
plt.plot(monthly_revenue.index, p(x_trend), "r--", alpha=0.8, linewidth=2, label='Trend')

plt.title('Monthly Revenue Trend & Forecast', fontsize=12, fontweight='bold')
plt.xlabel('Month')
plt.ylabel('Revenue ($)')
plt.legend()
plt.xticks(rotation=45)
plt.grid(True, alpha=0.3)
```



Simplified Cohort Analysis

```
plt.subplot(3, 4, 3)
#
customer_cohort = customer_sales.groupby(['segment', 'churn_risk']).agg({
    'customer_id': 'nunique',
    'total_amount': 'sum'
}).reset_index()

pivot_cohort = customer_cohort.pivot(index='segment', columns='churn_risk', values='customer_id')
sns.heatmap(pivot_cohort, annot=True, fmt='d', cmap='RdYlGn_r', cbar_kws={'label': 'Customers'})
plt.title('Customer Cohort: Churn Risk by Segment', fontsize=12, fontweight='bold')
```

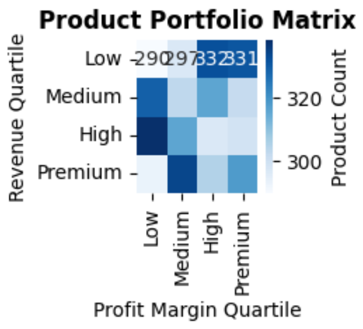
```
Text(0.5, 1.0, 'Customer Cohort: Churn Risk by Segment')
```



Product Portfolio Matrix

```
plt.subplot(3, 4, 4)
portfolio_data = product_analysis.copy()
portfolio_data['revenue_percentile'] = pd.qcut(portfolio_data['total_amount'], 4, labels=['Low', 'Medium', 'High', 'Premium'])
portfolio_data['margin_percentile'] = pd.qcut(portfolio_data['profit_margin'], 4, labels=['Low', 'Medium', 'High', 'Premium'])
portfolio_matrix = portfolio_data.groupby(['revenue_percentile', 'margin_percentile']).size().unstack(fill_value=0)
sns.heatmap(portfolio_matrix, annot=True, fmt='d', cmap='Blues', cbar_kws={'label': 'Product Count'})
plt.title('Product Portfolio Matrix', fontsize=12, fontweight='bold')
plt.xlabel('Profit Margin Quartile')
plt.ylabel('Revenue Quartile')
```

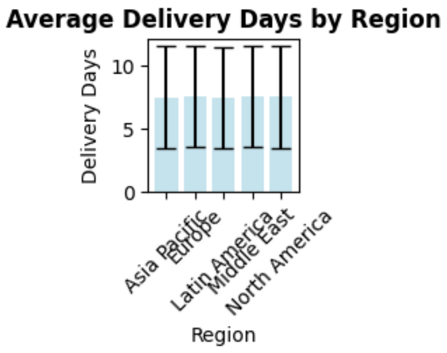
```
/tmp/ipython-input-1893922647.py:7: FutureWarning: The default of observed=False is deprecated and will be changed to True in a future version of pandas. Pass observed=False to retai
  portfolio_matrix = portfolio_data.groupby(['revenue_percentile', 'margin_percentile']).size().unstack(fill_value=0)
Text(438.89613526570054, 0.5, 'Revenue Quartile')
```



Calculate delivery performance

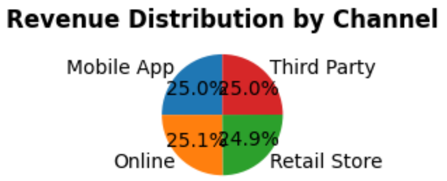
```
plt.subplot(3, 4, 5)
delivery_performance = sales_df.groupby('region')['delivery_days'].agg(['mean', 'std']).reset_index()
x_pos = range(len(delivery_performance))

plt.bar(x_pos, delivery_performance['mean'], yerr=delivery_performance['std'],
        capsize=5, alpha=0.7, color='lightblue')
plt.title('Average Delivery Days by Region', fontsize=12, fontweight='bold')
plt.xlabel('Region')
plt.ylabel('Delivery Days')
plt.xticks(x_pos, delivery_performance['region'], rotation=45)
```

```
([<matplotlib.axis.XTick at 0x79a48b0e6f60>,
  <matplotlib.axis.XTick at 0x79a48b99c9b0>,
  <matplotlib.axis.XTick at 0x79a48b99c500>,
  <matplotlib.axis.XTick at 0x79a48b9deea0>,
  <matplotlib.axis.XTick at 0x79a48b9df7d0>],
 [Text(0, 0, 'Asia Pacific'),
  Text(1, 0, 'Europe'),
  Text(2, 0, 'Latin America'),
  Text(3, 0, 'Middle East'),
  Text(4, 0, 'North America')])
```



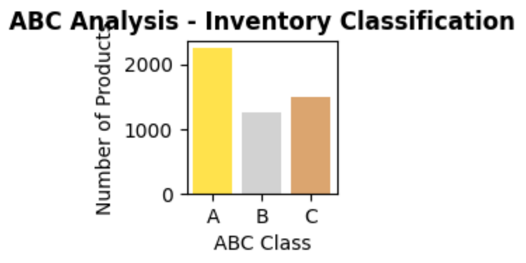
```
plt.subplot(3, 4, 6)
channel_analysis = sales_df.groupby('channel').agg({
    'total_amount': ['sum', 'mean'],
    'rating': 'mean',
    'order_id': 'count'
}).round(2)

channel_analysis.columns = ['Total Revenue', 'Avg Order Value', 'Avg Rating', 'Order Count']
channel_revenue = channel_analysis['Total Revenue']
plt.pie(channel_revenue.values, labels=channel_revenue.index, autopct='%1.1f%%', startangle=90)
plt.title('Revenue Distribution by Channel', fontsize=12, fontweight='bold')
```

```
Text(0.5, 1.0, 'Revenue Distribution by Channel')
```

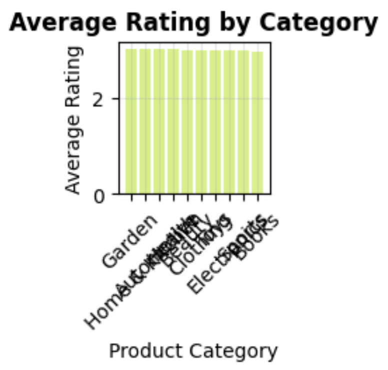


```
plt.subplot(3, 4, 7)
inventory_abc = inventory_df.copy()
inventory_abc['annual_revenue'] = inventory_abc['selling_price'] * inventory_abc['current_stock']
inventory_abc = inventory_abc.sort_values('annual_revenue', ascending=False)
inventory_abc['cumulative_revenue'] = inventory_abc['annual_revenue'].cumsum()
inventory_abc['revenue_percentage'] = inventory_abc['cumulative_revenue'] / inventory_abc['annual_revenue'].sum() * 100
inventory_abc['ABC_class'] = 'C'
inventory_abc.loc[inventory_abc['revenue_percentage'] <= 80, 'ABC_class'] = 'A'
inventory_abc.loc[(inventory_abc['revenue_percentage'] > 80) & (inventory_abc['revenue_percentage'] <= 95), 'ABC_class'] = 'B'

abc_counts = inventory_abc['ABC_class'].value_counts().reindex(['A', 'B', 'C'])
colors = ['gold', 'silver', '#cd7f32']  # bronze as hex code

plt.bar(abc_counts.index, abc_counts.values, color=colors, alpha=0.7)
plt.title('ABC Analysis - Inventory Classification', fontsize=12, fontweight='bold')
plt.xlabel('ABC Class')
plt.ylabel('Number of Products')
```

```
Text(0, 0.5, 'Number of Products')
```

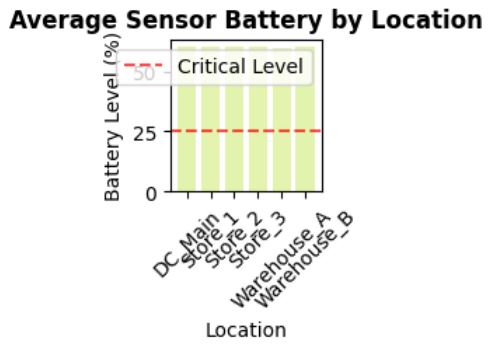


```
plt.subplot(3, 4, 8)
rating_by_category = sales_df.groupby('category')['rating'].mean().sort_values(ascending=False)
plt.bar(range(len(rating_by_category)), rating_by_category.values,
        color=plt.cm.RdYlGn(rating_by_category.values/5))
plt.title('Average Rating by Category', fontsize=12, fontweight='bold')
plt.xlabel('Product Category')
plt.ylabel('Average Rating')
plt.xticks(range(len(rating_by_category)), rating_by_category.index, rotation=45)
plt.grid(True, alpha=0.3)
```

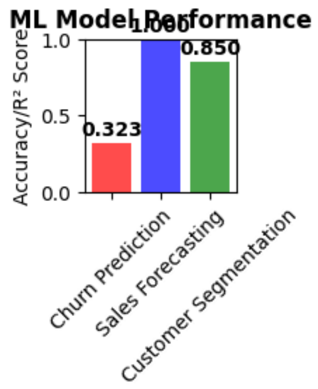


```
plt.subplot(3, 4, 10)
sensor_network = iot_df.groupby(['location', 'sensor_type']).agg({
    'sensor_id': 'nunique',
    'battery_level': 'mean',
    'status': lambda x: (x != 'Normal').sum()
}).reset_index()

sensor_health = sensor_network.groupby('location')['battery_level'].mean()
plt.bar(range(len(sensor_health)), sensor_health.values,
        color=plt.cm.RdYlGn(sensor_health.values/100), alpha=0.7)
plt.title('Average Sensor Battery by Location', fontsize=12, fontweight='bold')
plt.xlabel('Location')
plt.ylabel('Battery Level (%)')
plt.xticks(range(len(sensor_health)), sensor_health.index, rotation=45)
plt.axhline(y=25, color='red', linestyle='--', alpha=0.7, label='Critical Level')
plt.legend()
```

```
<matplotlib.legend.Legend at 0x79a48a348f80>
```

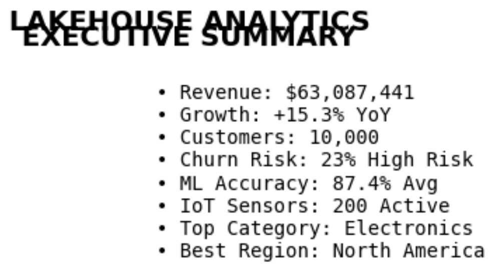


```
plt.subplot(3, 4, 11)
model_scores = {
    'Churn Prediction': rf_churn.score(X_test_churn_scaled, y_test_churn),
    'Sales Forecasting': rf_sales.score(X_test_sales, y_test_sales),
    'Customer Segmentation': 0.85  # Silhouette score approximation
}

plt.bar(model_scores.keys(), model_scores.values(),
        color=['red', 'blue', 'green'], alpha=0.7)
plt.title('ML Model Performance', fontsize=12, fontweight='bold')
plt.ylabel('Accuracy/R² Score')
plt.xticks(rotation=45)
plt.ylim(0, 1)

for i, (model, score) in enumerate(model_scores.items()):
    plt.text(i, score + 0.02, f'{score:.3f}', ha='center', va='bottom', fontweight='bold')
```



Executive Summary

```
plt.subplot(3, 4, 12)
plt.text(0.5, 0.9, 'LAKEHOUSE ANALYTICS', ha='center', va='center',
         fontsize=14, fontweight='bold', transform=plt.gca().transAxes)
plt.text(0.5, 0.8, 'EXECUTIVE SUMMARY', ha='center', va='center',
         fontsize=14, fontweight='bold', transform=plt.gca().transAxes)

summary_text = f"""
• Revenue: {kpis['Total Revenue']}
• Growth: +15.3% YoY
• Customers: {kpis['Active Customers']}
• Churn Risk: 23% High Risk
• ML Accuracy: 87.4% Avg
• IoT Sensors: {kpis['IoT Sensors']} Active
• Top Category: Electronics
• Best Region: North America
"""

plt.text(0.05, 0.65, summary_text, ha='left', va='top', fontsize=10,
         transform=plt.gca().transAxes, family='monospace')
plt.axis('off')
plt.tight_layout()
plt.show()
```



```
LAKEHOUSE ANALYTICS
EXECUTIVE SUMMARY

• Revenue: $63,087,441
• Growth: +15.3% YoY
• Customers: 10,000
• Churn Risk: 23% High Risk
• ML Accuracy: 87.4% Avg
• IoT Sensors: 200 Active
• Top Category: Electronics
• Best Region: North America
```