Import Library

In [611…
```
import pandas as pd
import numpy as np
import matplotlib.pyplot as plt
import seaborn as sns
import plotly.express as px
import warnings
warnings.filterwarnings('ignore')
```

Import Dataset

In [550…
```
df = pd.read_csv('ArifReport.csv')
```

In [551…
```
df.head()
```

Out[551]:

	ProductID	StoreID	ReportDate	SalesCumulativeSum
0	5469144250	53949207	2021-04-23	7
1	5936880091	17737769	2021-04-23	9
2	4845674243	237606391	2021-04-23	8
3	5726226400	124489135	2021-04-23	5
4	2892619493	156415596	2021-04-23	11

In [552…
```
# Cek informasi data

df.info()
```

```
<class 'pandas.core.frame.DataFrame'>
RangeIndex: 96753 entries, 0 to 96752
Data columns (total 4 columns):
 #   Column              Non-Null Count  Dtype 
---  ------              --------------  ----- 
 0   ProductID           96753 non-null  int64 
 1   StoreID             96753 non-null  int64 
 2   ReportDate          96753 non-null  object
 3   SalesCumulativeSum  96753 non-null  int64 
dtypes: int64(3), object(1)
memory usage: 3.0+ MB
```

In [553…
```
# cek data duplikat

df.duplicated().sum()
```

Out[553]: 0

In [554…
```
# cek data kosong

df.isnull().sum()
```

Out[554]:
```
ProductID             0
StoreID               0
ReportDate            0
SalesCumulativeSum    0
dtype: int64
```

In [555…
```
# mengubah tipe data
df['ReportDate'] = pd.to_datetime(df['ReportDate'],errors='coerce')
df['ProductID'] = df['ProductID'].astype(str)
df['StoreID'] = df['StoreID'].astype(str)
df['SalesCumulativeSum'] = df['SalesCumulativeSum'].astype(int)
```

In [556…
```
# mengurutkan data
df = df.sort_values(by=['ProductID', 'StoreID', 'ReportDate'])
```

In [557…
```
df
```

Out[557]:

	ProductID	StoreID	ReportDate	SalesCumulativeSum
13966	10000317232	39145153	2021-04-28	0
20562	10000317232	39145153	2021-04-29	0
27268	10000317232	39145153	2021-05-01	0
31286	10000317232	39145153	2021-05-04	0
36661	10000317232	39145153	2021-05-05	0
…	…	…	…	…
77257	998298981	54722738	2021-05-15	0
82381	998298981	54722738	2021-05-16	0
86586	998298981	54722738	2021-05-17	0
91709	998298981	54722738	2021-05-18	0
21119	999044353	58889343	2021-04-29	8

96753 rows × 4 columns

Pada rentang tanggal berapa Arif hilang fokus?

In [558…
```
data = df.copy()
```

In [559…
```
# step 1: mengurutkan data
data = data.sort_values(by=['ProductID', 'StoreID', 'ReportDate'])

# Step 2: Calculate DailySales
data['DailySales'] = data.groupby(['ProductID', 'StoreID'])['SalesCumulativeSum'

# Detect Anomalies
# Anomaly 1: Daily Sales Negative, sales cumulatif harusnya terus naik tidak tur
data['Anomaly_DailySales_Negative'] = data['DailySales'] < 0

# Anomaly 2: Sales on Holidays
holidays = ['2021-05-07', '2021-05-12']
```

```
data['Anomaly_Sales_On_Holiday'] = data['ReportDate'].isin(holidays) & (data['Sa

# Anomaly 3: Missing Data
data['Anomaly_Missing_Data'] = data['SalesCumulativeSum'].isna()

# Anomaly 4: Duplicate Records
data['Anomaly_Duplicate'] = data.duplicated(subset=['ProductID', 'StoreID', 'Rep

# Step 3: Output Anomalies
data['AnomalyType'] = data[['Anomaly_DailySales_Negative', 'Anomaly_Sales_On_Hol
                           'Anomaly_Missing_Data', 'Anomaly_Duplicate']].any(ax
anomalies = data[data['AnomalyType']]
anomalies
```

Out[559]:

	ProductID	StoreID	ReportDate	SalesCumulativeSum	DailySales	Anomaly_I
40441	100023558	12621218	2021-05-05	11	-6.0	
45474	100023558	12621218	2021-05-06	10	-1.0	
55530	100023558	12621218	2021-05-09	9	-7.0	
28416	1011122810	53740401	2021-05-03	30	-17.0	
36339	1011122810	53740401	2021-05-05	27	-26.0	
...	...	...	...	...	...	
41205	9935593420	251219523	2021-05-06	11	-11.0	
51173	9935593420	251219523	2021-05-09	11	-4.0	
51758	9938865748	272054971	2021-05-09	2	-1.0	
51827	9949378375	150298732	2021-05-09	4	-4.0	
61858	9949378375	150298732	2021-05-11	4	-4.0	

3350 rows × 10 columns

In [578…

```
print('Tanggal yang memiliki anomali:')
anomalies.groupby('ReportDate').size()
```

```
Tanggal yang memiliki anomali:
```

Out[578]:

```
ReportDate
2021-05-01     35
2021-05-02     72
2021-05-03    282
2021-05-04    439
2021-05-05    523
2021-05-06    493
2021-05-08    486
2021-05-09    980
2021-05-10     25
2021-05-11     14
2021-05-13      1
dtype: int64
```

namun kita perlu mengecek untuk tanggal awal dan akhir, karena daily sales negatif belum tentu di tanggal tersebut

yang menjadi anomali, kemungkinan juga anomali di tanggal sebelumnya

In [586…
```
date_min = anomalies['ReportDate'].min()
date_max = anomalies['ReportDate'].max()
```

In [595…
```
productanom = anomalies[anomalies['ReportDate']==date_max]["ProductID"].unique()
productanom
```

Out[595]:
```
array(['2432635060'], dtype=object)
```

In [597…
```
# cari produkID anonim di tanggal 13
df[df['ProductID']=='2432635060']
```

Out[597]:

	ProductID	StoreID	ReportDate	SalesCumulativeSum
7158	2432635060	77502887	2021-04-26	19
22090	2432635060	77502887	2021-04-30	26
66107	2432635060	77502887	2021-05-13	20
71159	2432635060	77502887	2021-05-14	20
80980	2432635060	77502887	2021-05-15	20
86108	2432635060	77502887	2021-05-16	21
88137	2432635060	77502887	2021-05-17	21
93325	2432635060	77502887	2021-05-18	21

dari data, yang mungkin anomali adalah 30 april bukan 13 mei

In [601…
```
productanom = anomalies[anomalies['ReportDate']==date_min]["ProductID"].unique()
productanom
```

Out[601]:
```
array(['1825889122', '2343231517', '2671794318', '2899402331',
       '3100087879', '3341968687', '3422773117', '3459561909',
       '3633905780', '385170423', '4020080217', '4055208010',
       '4400457690', '4577315519', '4662534174', '4807923851',
       '5054475715', '5119270317', '6130032238', '6131017144',
       '6732563650', '6835417376', '7134064011', '7271595110',
       '7453403606', '7937330208', '7937931627', '848223431',
       '8500690704', '8718579163', '8746401534', '915919306',
       '9302127937', '9413973156', '9418460897'], dtype=object)
```

In [602…
```
# cari produkID anonim di tanggal 13
df[df['ProductID']=='1825889122']
```

Out[602]:

	ProductID	StoreID	ReportDate	SalesCumulativeSum
96	1825889122	10011932	2021-04-23	128
2437	1825889122	10011932	2021-04-24	128
9576	1825889122	10011932	2021-04-27	128
16120	1825889122	10011932	2021-04-28	128
21808	1825889122	10011932	2021-04-29	128
22104	1825889122	10011932	2021-04-30	238
23202	1825889122	10011932	2021-05-01	136
27403	1825889122	10011932	2021-05-02	137
28405	1825889122	10011932	2021-05-03	256
30641	1825889122	10011932	2021-05-04	234
36224	1825889122	10011932	2021-05-05	213
41171	1825889122	10011932	2021-05-06	250
46247	1825889122	10011932	2021-05-08	170
51231	1825889122	10011932	2021-05-09	128
56263	1825889122	10011932	2021-05-10	128
61306	1825889122	10011932	2021-05-11	128
66335	1825889122	10011932	2021-05-13	128
71394	1825889122	10011932	2021-05-14	128
80915	1825889122	10011932	2021-05-15	128
86043	1825889122	10011932	2021-05-16	128
88654	1825889122	10011932	2021-05-17	128
93847	1825889122	10011932	2021-05-18	128

dari data dapat disimpulkan bahwa data mulai anomali adalah di tanggal 30 april

dst. sehingga data anomali didapatkan pada rentang 30 April hingga 10 Mei 2021

mari kita cek, misalkan tanggal 30 april - 10 mei dihapus, seharusnya daily sales tidak ada yang negatif!

In [612…

```
# Pastikan kolom tanggal dalam format datetime
df2 = df.copy()

df2['ReportDate'] = pd.to_datetime(df2['ReportDate'])

# Membuat rentang tanggal yang ingin dikecualikan
exclude_dates = pd.date_range(start="2021-04-30", end="2021-05-10")
```

```
# Filter data untuk menghilangkan tanggal dalam rentang tersebut
filtered_data = df2[~df2['ReportDate'].isin(exclude_dates)]

filtered_data['DailySales'] = filtered_data.groupby(['ProductID', 'StoreID'])['S
daily_sales_negatif = (filtered_data['DailySales']<0).sum()
print(f"Jumlah daily sales negatif : {daily_sales_negatif}")
```

```
Jumlah daily sales negatif : 0
```

Sehingga Arif kehilangan fokus dari tanggal 30 April 2021 - 10 Mei 2021

Bagaimana membersihkan data agar dapat digunakan untuk analisis penjualan?

In [561…

```
data2 = df.copy()
```

In [562…

```
# Menentukan rentang tanggal yang bermasalah
start_date = pd.to_datetime('2021-04-30')
end_date = pd.to_datetime('2021-05-10')

# mengatasi masalah penjualan kumulatif pada rentang tanggal bermasalah dengan i
def fix_sales_cumulative(group):
    # Menandai tanggal sebelum 30 April dan setelah 10 Mei
    before_start = group[group['ReportDate'] < start_date]
    after_end = group[group['ReportDate'] > end_date]
    problem_dates = group[(group['ReportDate'] >= start_date) & (group['ReportDa

    # Mengganti nilai kumulatif berdasarkan kondisi
    if not before_start.empty and not after_end.empty:
        # Jika ada tanggal sebelum dan sesudah rentang
        start_value = before_start['SalesCumulativeSum'].iloc[-1]
        end_value = after_end['SalesCumulativeSum'].iloc[0]

        # Membagi nilai secara linier ke tanggal yang bermasalah
        n_dates = len(problem_dates)
        increments = (end_value - start_value) / (n_dates + 1)

        # Perbarui nilai kumulatif untuk tanggal yang bermasalah
        new_values = [start_value + increments * (i + 1) for i in range(n_dates)
        group.loc[problem_dates.index, 'SalesCumulativeSum'] = [round(val) for v
    elif not before_start.empty:
        # Jika hanya ada tanggal sebelum 30 April, gunakan nilai sebelum 30 Apri
        group.loc[problem_dates.index, 'SalesCumulativeSum'] = before_start['Sal
    elif not after_end.empty:
        # Jika hanya ada tanggal setelah 10 Mei, gunakan nilai setelah 10 Mei
        group.loc[problem_dates.index, 'SalesCumulativeSum'] = after_end['SalesC
    else:
            # Jika tidak ada nilai sebelum maupun setelah rentang, hapus baris
            group = group.drop(problem_dates.index)

    # Pastikan nilai SalesCumulativeSum adalah bilangan bulat
    group['SalesCumulativeSum'] = group['SalesCumulativeSum'].round().astype(int

    return group

# Terapkan fungsi untuk setiap pasangan ProductID dan StoreID
```

```
data2 = data2.groupby(['ProductID', 'StoreID']).apply(fix_sales_cumulative)
df_cleaned = data2[['ReportDate','SalesCumulativeSum']].reset_index().drop('leve
```

In [563…

```
def handle_missing_dates(df):
    """
    Mengisi missing date pada kolom SalesCumulativeSum berdasarkan kombinasi uni

    Parameters:
    - df: DataFrame input dengan kolom ['ProductID', 'StoreID', 'ReportDate', 'S

    Returns:
    - DataFrame dengan nilai SalesCumulativeSum yang telah diisi.
    """
    # Tentukan rentang tanggal, mengabaikan hari libur
    start_date = '2021-04-23'
    end_date = '2021-05-18'
    holidays = ['2021-05-07', '2021-05-12']
    date_range = pd.date_range(start=start_date, end=end_date).difference(pd.to_

    # Dapatkan kombinasi unik ProductID dan StoreID
    unique_combinations = df[['ProductID', 'StoreID']].drop_duplicates()

    # Buat DataFrame dengan semua kombinasi ProductID, StoreID, dan tanggal
    expanded_data = (
        unique_combinations.assign(key=1)  # Tambahkan kolom dummy 'key' untuk p
        .merge(pd.DataFrame({'ReportDate': date_range, 'key': 1}), on='key')  #
        .drop('key', axis=1)  # Hapus kolom 'key' setelah digunakan
    )

    # Gabungkan data yang diperluas dengan data asli
    df_cleaned = expanded_data.merge(df, on=['ProductID', 'StoreID', 'ReportDate

    # Isi missing value pada SalesCumulativeSum dengan nilai hari sebelumnya (fo
    df_cleaned['SalesCumulativeSum'] = df_cleaned.groupby(['ProductID', 'StoreID

    # Isi missing value yang tersisa dengan nilai hari berikutnya (backward fill
    df_cleaned['SalesCumulativeSum'] = df_cleaned.groupby(['ProductID', 'StoreID

    # Hitung DailySales sebagai selisih antar nilai SalesCumulativeSum
    df_cleaned['DailySales'] = df_cleaned.groupby(['ProductID', 'StoreID'])['Sal

    # Isi nilai yang tersisa dengan 0 (jika masih ada)
    df_cleaned = df_cleaned.fillna(0)

    return df_cleaned
```

In [564…

```
df_cleaned = handle_missing_dates(df_cleaned)
```

In [565…

```
df_kotor = handle_missing_dates(df)
```

In [566…

```
# export dataset yang telah dibersihkan

# df_cleaned.to_csv('data_bersih.csv')
# df_kotor.to_csv('data_kotor.csv')
```

Bagaimana perbandingan data sebelum dan sesudah dibersihkan?

Data sebelum dibersihkan

In [567…

```
# data kotor kumulatif

cmltv = df_kotor.groupby('ReportDate')['SalesCumulativeSum'].sum().reset_index()
fig = px.line(cmltv, x='ReportDate', y='SalesCumulativeSum', title='Trend Penjua
fig.update_traces(line=dict(color='#800020'),  fill='tozeroy')

# Mengatur sumbu y agar tidak dimulai dari 0
y_min = cmltv['SalesCumulativeSum'].min() * 0.99
y_max = cmltv['SalesCumulativeSum'].max() * 1.001
fig.update_yaxes(range=[y_min, y_max])
fig.show()
```

In [568…

```
trend = df_kotor.groupby('ReportDate')['DailySales'].sum().reset_index()
fig = px.line(trend, x='ReportDate', y='DailySales', title='Trend Penjualan Hari
fig.update_traces(line=dict(color='#800020'))
fig.show()
```

Data setelah dibersihkan

In [569…

```
# data bersih kumulatif

cmltv = df_cleaned.groupby('ReportDate')['SalesCumulativeSum'].sum().reset_index
fig = px.line(cmltv, x='ReportDate', y='SalesCumulativeSum', title='Trend Penjua
fig.update_traces(line=dict(color='#800020'),  fill='tozeroy')

# Mengatur sumbu y agar tidak dimulai dari 0
y_min = cmltv['SalesCumulativeSum'].min() * 0.99
y_max = cmltv['SalesCumulativeSum'].max() * 1.001
fig.update_yaxes(range=[y_min, y_max])
fig.show()
```

In [570…

```
trend = df_cleaned.groupby('ReportDate')['DailySales'].sum().reset_index()
fig = px.line(trend, x='ReportDate', y='DailySales', title='Trend Penjualan Hari
fig.update_traces(line=dict(color='#800020'))
fig.show()
```

Insight:

- Data bersih memiliki nilai kumulatif yang terus naik dan tidak mengalami penurunan, sedangkan data kotor memiliki nilai kumulatif yang naik turun.
- Data bersih hanya memiliki nilai penjualan harian yang positif, sedangkan data kotor memiliki nilai penjualan harian yang positif dan negatif.

Analisis Lanjutan

Data Preview

In [571…

```
total_sales = df_cleaned['DailySales'].sum()
total_pesanan = df_cleaned[df_cleaned['DailySales']>0]['DailySales'].sum()
date_max = df_cleaned['ReportDate'].max()
total_cumulative_sales = df_cleaned[df_cleaned['ReportDate']==date_max]['SalesCu
avg_sales = df_cleaned.groupby('ReportDate')['DailySales'].sum().reset_index()[1
total_produk = df_cleaned['ProductID'].nunique()
total_store = df_cleaned['StoreID'].nunique()


print(f"Total penjualan kumulatif : {int(total_cumulative_sales):,}")
print(f"Total Pesanan : {int(total_pesanan):,}")
print(f"Total penjualan harian : {int(total_sales):,}")
print(f"Rata-rata penjualan harian : {avg_sales:,.1f}")
```

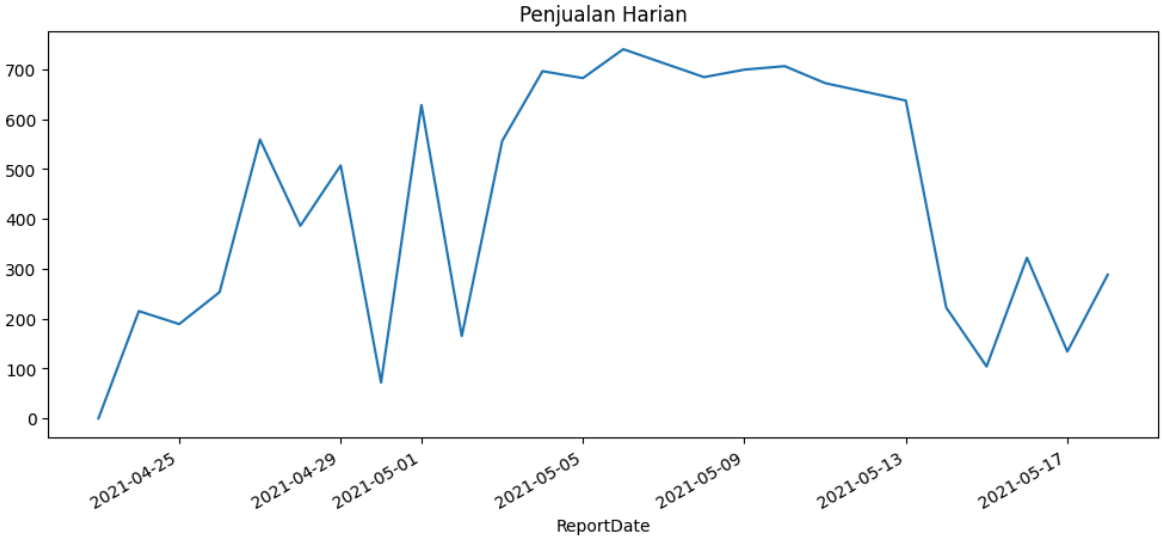
```
print(f"Total produk : {total_produk:,}")
print(f"Total toko : {total_store:,}")
```

```
Total penjualan kumulatif : 505,121
Total Pesanan : 10,116
Total penjualan harian : 10,116
Rata-rata penjualan harian : 439.8
Total produk : 6,855
Total toko : 4,985
```

Bagaimana trend penjualan harian?

In [572…

```
plt.figure(figsize=(12,5))
trend_daily = df_cleaned.groupby('ReportDate')['DailySales'].sum()
trend_daily.plot()
plt.title('Penjualan Harian');
```

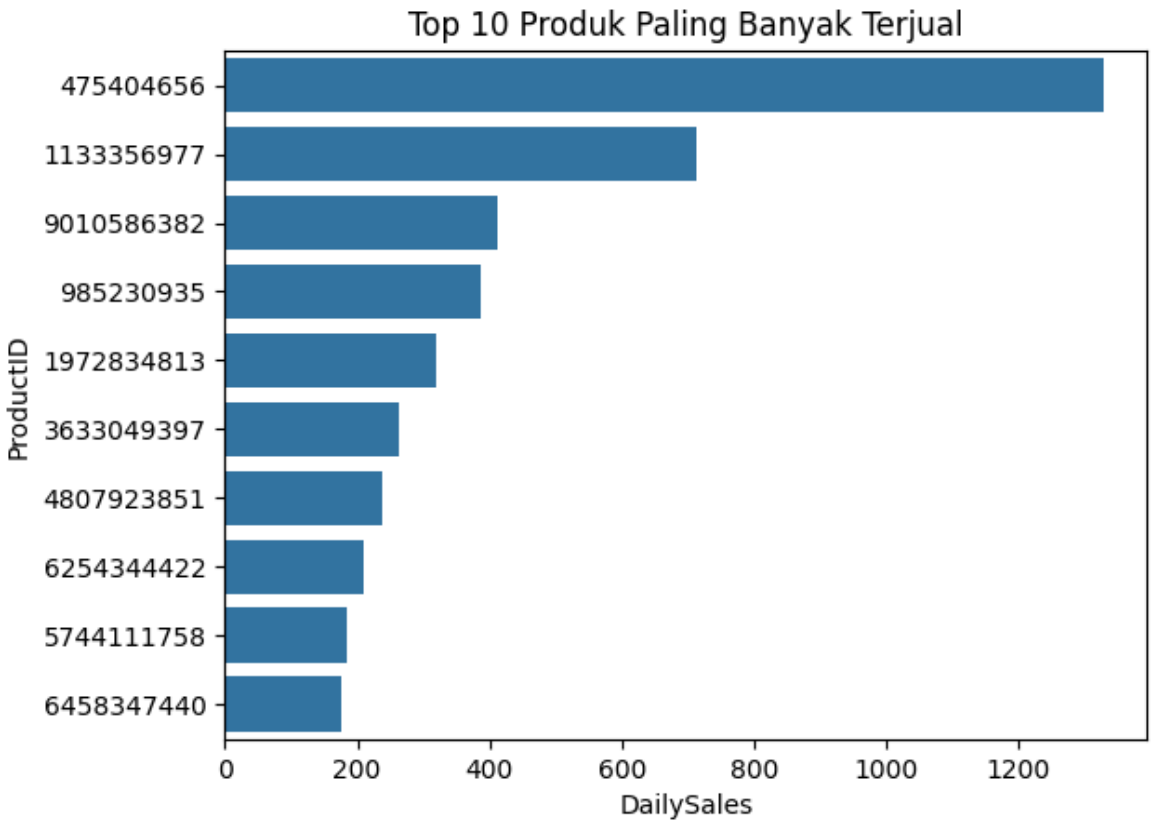


Pada awal setelah restock, penjualan cenderung meningkat, namun saat mendekati tanggal 20 (sebelum restock lagi), penjualan cenderung menurun! Hal ini mungkin terjadi karena kehabisan barang, sehingga mengakibatkan penjualan melambat.

Apa produk yang paling banyak terjual?

In [573…

```
topprod = df_cleaned.groupby('ProductID')['DailySales'].sum().nlargest(10).reset
sns.barplot(topprod, y='ProductID', x='DailySales')
plt.title('Top 10 Produk Paling Banyak Terjual');
```



Toko mana yang memiliki penjualan terbesar?

In [574…

```
topstr = df_cleaned.groupby('StoreID')['DailySales'].sum().nlargest(10).reset_in
sns.barplot(topstr, y='StoreID', x='DailySales')
plt.title('Top 10 Toko yang Memiliki Penjualan Terbesar');
```

Kontribusi Top 10 Produk Terhadap Penjualan

In [575…
```
kumulatif = df_cleaned[df_cleaned['ReportDate']==date_max]
```

In [576…
```
# Menghitung total penjualan kumulatif per toko
total_sales = kumulatif['SalesCumulativeSum'].sum()
top_10_prod = kumulatif.groupby('ProductID')['SalesCumulativeSum'].sum().nlarges

# Menghitung kontribusi persentase
top_10_contribution = top_10_prod.sum() / total_sales * 100
other_contribution = 100 - top_10_contribution

# Membuat DataFrame untuk visualisasi
contribution_data = pd.DataFrame({
    'Kategori': ['10 Produk Teratas', 'Produk Lainnya'],
    'Persentase': [top_10_contribution, other_contribution]
})

# Membuat diagram pie
fig = px.pie(
    contribution_data,
    names='Kategori',
    values='Persentase',
    title='Kontribusi Penjualan Kumulatif 10 Produk Teratas terhadap Semua Penju
    labels={'Persentase': 'Persentase Penjualan'}
)

# Menampilkan grafik
fig.show()
```

Kontribusi Top 10 Store Terhadap Penjualan

In [577…

```
# Menghitung total penjualan kumulatif per toko
total_sales = kumulatif['SalesCumulativeSum'].sum()
top_10_stores = kumulatif.groupby('StoreID')['SalesCumulativeSum'].sum().nlarges

# Menghitung kontribusi persentase
top_10_contribution = top_10_stores.sum() / total_sales * 100
other_contribution = 100 - top_10_contribution

# Membuat DataFrame untuk visualisasi
contribution_data = pd.DataFrame({
    'Kategori': ['10 Toko Teratas', 'Toko Lainnya'],
    'Persentase': [top_10_contribution, other_contribution]
})

# Membuat diagram pie
fig = px.pie(
    contribution_data,
    names='Kategori',
    values='Persentase',
    title='Kontribusi Penjualan Kumulatif 10 Toko Teratas terhadap Semua Penjual
    labels={'Persentase': 'Persentase Penjualan'}
)
```

```
# Menampilkan grafik
fig.show()
```

Rekomendasi

1. Pola Penjualan Setelah dan Sebelum Restock

- **Analisis**: Penjualan cenderung meningkat segera setelah restock, namun melambat sebelum restock.
- **Rekomendasi**:
  - **Frekuensi Restock**: Pertimbangkan untuk memperpendek siklus restock agar stok tetap tersedia di toko, terutama untuk produk dengan permintaan tinggi.
  - **Promosi Pra-Restock**: Jalankan kampanye promosi atau diskon seminggu sebelum restock untuk mendorong penjualan produk yang tersisa.
  - **Optimalisasi Stok**: Gunakan analisis data historis untuk memprediksi kebutuhan stok setiap toko dan produk, sehingga tidak ada kekurangan stok sebelum restock.

---

2. Konsentrasi Penjualan pada 10 Produk Teratas

- **Analisis**: Sepuluh produk terlaris menyumbang 42% dari total penjualan.
- **Rekomendasi**:

- **Fokus pada Produk Unggulan**: Tingkatkan produksi, stok, dan promosi untuk 10 produk terlaris ini.
- **Diversifikasi Penawaran**: Analisis produk lain yang memiliki potensi pertumbuhan untuk mengurangi ketergantungan pada produk tertentu.
- **Penempatan Strategis**: Pastikan produk-produk unggulan ini tersedia di semua toko dengan visibilitas tinggi (rak depan atau area promosi).

---

3. Dominasi Penjualan pada 10 Toko Teratas

- **Analisis**: Sepuluh toko terlaris menyumbang 46% dari total penjualan.
- **Rekomendasi**:
  - **Replikasi Keberhasilan**: Analisis strategi operasional, lokasi, dan promosi dari toko-toko ini, lalu replikasi di toko lain yang memiliki potensi serupa.
  - **Pengembangan Toko Non-Teratas**: Fokus pada toko dengan penjualan rendah tetapi potensi pasar tinggi (lokasi strategis, populasi pelanggan besar, atau permintaan belum terpenuhi).
  - **Program Insentif**: Berikan program insentif kepada toko dengan performa rendah untuk mendorong penjualan, seperti bonus target atau diskon tambahan.

Strategi Pendukung

1. **Penggunaan Data Realtime**: Implementasikan sistem monitoring stok dan penjualan secara real-time untuk merespons kebutuhan pasar lebih cepat.
2. **Pemasaran Digital**: Gunakan kampanye digital yang menargetkan produk unggulan di toko tertentu, misalnya melalui media sosial atau aplikasi.
3. **Penawaran Bundling**: Ciptakan paket produk yang menggabungkan produk unggulan dengan produk yang kurang laris untuk meningkatkan volume penjualan.

Terima Kasih

In [ ]: