Nama **: Sisca Linda Anari**

NIM **: 12030123120044**

Kelas **: F**

PENGKODEAN DAN PEMROGRAMAN

PERTEMUAN 12

DATASET PERDAGANGAN

Question : buat dataset perdagangan yang didalammnya terdapat tiga tabel: tabel produksi, tabel persediaan, tabel penjualan, tabel dalam bentuk csv. penjelasan lengkap dengan struktur tabelnya.

Jawab :

Penjelasan Struktur Tabel

1. **Tabel Produksi (production.csv)**
   Tabel ini mencatat data produksi barang di suatu perusahaan perdagangan.
   Struktur Kolom:
   - production_id: ID unik untuk setiap catatan produksi (integer).
   - product_id: ID produk yang diproduksi (string, misal: PROD001).
   - product_name: Nama produk (string).
   - production_date: Tanggal produksi (format: YYYY-MM-DD).
   - quantity_produced: Jumlah unit yang diproduksi (integer).
   - production_cost: Biaya produksi per unit (decimal).
2. **Tabel Persediaan (inventory.csv)**
   Tabel ini mencatat status persediaan barang di gudang.
   Struktur Kolom:
   - inventory_id: ID unik untuk setiap catatan persediaan (integer).
   - product_id: ID produk yang ada di persediaan (string, sesuai dengan tabel produksi).
   - warehouse_id: ID gudang tempat penyimpanan (string, misal: WH001).
   - stock_quantity: Jumlah stok saat ini (integer).
   - last_updated: Tanggal terakhir pembaruan stok (format: YYYY-MM-DD).
3. **Tabel Penjualan (sales.csv)**
   Tabel ini mencatat data penjualan produk.
   Struktur Kolom:
   - sale_id: ID unik untuk setiap transaksi penjualan (integer).
   - product_id: ID produk yang dijual (string, sesuai dengan tabel produksi).
   - sale_date: Tanggal penjualan (format: YYYY-MM-DD).
   - quantity_sold: Jumlah unit yang terjual (integer).
   - sale_price: Harga jual per unit (decimal).

- total_revenue: Total pendapatan dari penjualan (quantity_sold * sale_price, decimal).

Dataset

production.csv

Production_id	Product_id	Product_name	Production_date	Quantity_produced	Production_cost
1	PROD001	Beras Premium	45667	1000	12000.5
2	PROD002	Minyak Goreng	45672	500	15000.75
3	PROD003	Gula Pasir	45689	800	13000.25
4	PROD001	Beras Premium	45698	1200	12200
5	PROD004	Tepung Terigu	45703	600	9000.3

inventory.csv

Inventory_id	Product_id	Warehouse_id	Stock_quantity	Last_updated
1	PROD001	WH001	800	45717
2	PROD002	WH002	300	45718
3	PROD003	WH001	500	45717
4	PROD004	WH003	400	45719
5	PROD001	WH002	200	45718

sales.csv

Sale_id	Product_id	Sale_date	Quantity_sold	Sale_price	Total_revenue
1	PROD001	45717	200	15000	3000000
2	PROD002	45718	100	18000.5	1800050
3	PROD003	45719	150	16000.75	2400112.5
4	PROD001	45720	300	15200	4560000
5	PROD004	45721	50	11000.25	550012.5

Catatan tambahan:

- **Konsistensi Data**

  Kolom product_id digunakan untuk menghubungkan ketiga tabel, memastikan integritas data antar tabel.

- **Format Tanggal**

  Semua tanggal menggunakan format YYYY-MM-DD untuk konsistensi.

- **Contoh Penggunaan**

  Dataset ini dapat digunakan untuk analisis perdagangan, seperti menghitung keuntungan (dari total_revenue dan production_cost), memantau stok, atau melacak tren penjualan.

Input >>colab

Question : Berdasarkan dataset perdagangan yang berisi ketiga tabel dibawah ini , buat data analitik sederhana lengkap dengan download hasil tabel dan visualisasi hasilnya.

Result :

Reasoning: tiga file CSV ke dalam pandas DataFrames dan tampilkan 5 baris pertama masing-masing untuk diverifikasi.

```
import pandas as pd

try:
    sales_df = pd.read_csv('sales.csv')
    inventory_df = pd.read_csv('inventory.csv')
    production_df = pd.read_csv('production.csv')

    display(sales_df.head())
    display(inventory_df.head())
    display(production_df.head())

except FileNotFoundError:
    print("One or more CSV files not found.")
except pd.errors.ParserError:
    print("Error parsing one or more CSV files. Check file format.")
except Exception as e:
    print(f"An unexpected error occurred: {e}")
```

	sale_id	product_id	sale_date	quantity_sold	sale_price	total_revenue
0	1	PROD001	2025-03-01	200	15000.00	3000000.0
1	2	PROD002	2025-03-02	100	18000.50	1800050.0
2	3	PROD003	2025-03-03	150	16000.75	2400112.5
3	4	PROD001	2025-03-04	300	15200.00	4560000.0
4	5	PROD004	2025-03-05	50	11000.25	550012.5

	inventory_id	product_id	warehouse_id	stock_quantity	last_updated
0	1	PROD001	WH001	800	2025-03-01
1	2	PROD002	WH002	300	2025-03-02
2	3	PROD003	WH001	500	2025-03-01
3	4	PROD004	WH003	400	2025-03-03
4	5	PROD001	WH002	200	2025-03-02

	production_id	product_id	product_name	production_date	quantity_produced	production_cost
0	1	PROD001	Beras Premium	2025-01-10	1000	12000.50
1	2	PROD002	Minyak Goreng	2025-01-15	500	15000.75
2	3	PROD003	Gula Pasir	2025-02-01	800	13000.25
3	4	PROD001	Beras Premium	2025-02-10	1200	12200.00
4	5	PROD004	Tepung Terigu	2025-02-15	600	9000.30

Data exploration

```
# Data Exploration

# 1. Shape of DataFrames
print("Sales DataFrame Shape:", sales_df.shape)
print("Inventory DataFrame Shape:", inventory_df.shape)
print("Production DataFrame Shape:", production_df.shape)

# 2. Data Types
print("\nSales DataFrame Data Types:\n", sales_df.dtypes)
print("\nInventory DataFrame Data Types:\n", inventory_df.dtypes)
print("\nProduction DataFrame Data Types:\n", production_df.dtypes)

# 3. Descriptive Statistics
print("\nSales DataFrame Descriptive Statistics:\n",
sales_df.describe(include='all'))
print("\nInventory DataFrame Descriptive Statistics:\n",
inventory_df.describe(include='all'))
print("\nProduction DataFrame Descriptive Statistics:\n",
production_df.describe(include='all'))

# 4. Missing Values
print("\nSales DataFrame Missing Values:\n", sales_df.isnull().sum())
print("\nInventory DataFrame Missing Values:\n", inventory_df.isnull().sum())
print("\nProduction DataFrame Missing Values:\n", production_df.isnull().sum())

# 5. Duplicate Rows
print("\nSales DataFrame Duplicate Rows:", sales_df.duplicated().sum())
print("\nInventory DataFrame Duplicate Rows:", inventory_df.duplicated().sum())
print("\nProduction DataFrame Duplicate Rows:", production_df.duplicated().sum())

# 6. Unique Values in Key Columns
print("\nSales DataFrame Unique Product IDs:", sales_df['product_id'].nunique())
print("Inventory DataFrame Unique Product IDs:",
inventory_df['product_id'].nunique())
print("Production DataFrame Unique Product IDs:",
production_df['product_id'].nunique())

print("\nSales DataFrame Unique Sale Dates:", sales_df['sale_date'].nunique())
print("Inventory DataFrame Unique Update Dates:",
inventory_df['last_updated'].nunique())
print("Production DataFrame Unique Production Dates:",
production_df['production_date'].nunique())

# Check for product_ids in sales that are not in production
sales_products = set(sales_df['product_id'].unique())
production_products = set(production_df['product_id'].unique())
missing_products = sales_products - production_products
print("\nProduct IDs in sales but not in production:", missing_products)
```

```
Sales DataFrame Shape: (5, 6)
Inventory DataFrame Shape: (5, 5)
Production DataFrame Shape: (5, 6)

Sales DataFrame Data Types:
 sale_id            int64
product_id        object
sale_date         object
quantity_sold      int64
sale_price       float64
total_revenue    float64
dtype: object

Inventory DataFrame Data Types:
 inventory_id       int64
product_id        object
warehouse_id      object
stock_quantity     int64
last_updated      object
dtype: object

Production DataFrame Data Types:
 production_id          int64
product_id            object
product_name          object
production_date       object
quantity_produced      int64
production_cost      float64
dtype: object

Sales DataFrame Descriptive Statistics:
         sale_id product_id   sale_date  quantity_sold    sale_price  \
count   5.000000          5           5        5.00000      5.000000
unique       NaN          4           5            NaN           NaN
top          NaN    PROD001  2025-03-01            NaN           NaN
freq         NaN          2           1            NaN           NaN
mean    3.000000        NaN         NaN      160.00000  15040.300000
std     1.581139        NaN         NaN       96.17692   2551.194839
min     1.000000        NaN         NaN       50.00000  11000.250000
25%     2.000000        NaN         NaN      100.00000  15000.000000
50%     3.000000        NaN         NaN      150.00000  15200.000000
75%     4.000000        NaN         NaN      200.00000  16000.750000
max     5.000000        NaN         NaN      300.00000  18000.500000

        total_revenue
count    5.000000e+00
unique            NaN
top               NaN
freq              NaN
mean     2.462035e+06
std      1.482291e+06
min      5.500125e+05
25%      1.800050e+06
50%      2.400112e+06
75%      3.000000e+06
max      4.560000e+06

Inventory DataFrame Descriptive Statistics:
        inventory_id product_id warehouse_id  stock_quantity last_updated
count       5.000000          5            5        5.000000            5
unique           NaN          4            3             NaN            3
top              NaN    PROD001        WH001             NaN   2025-03-01
freq             NaN          2            2             NaN            2
mean        3.000000        NaN          NaN      440.000000          NaN
std         1.581139        NaN          NaN      230.217289          NaN
min         1.000000        NaN          NaN      200.000000          NaN
```

```
25%         2.000000        NaN          NaN      300.000000          NaN
50%         3.000000        NaN          NaN      400.000000          NaN
75%         4.000000        NaN          NaN      500.000000          NaN
max         5.000000        NaN          NaN      800.000000          NaN

Production DataFrame Descriptive Statistics:
         production_id product_id   product_name production_date  \
count         5.000000          5              5               5
unique             NaN          4              4               5
top                NaN    PROD001  Beras Premium      2025-01-10
freq               NaN          2              2               1
mean          3.000000        NaN            NaN             NaN
std           1.581139        NaN            NaN             NaN
min           1.000000        NaN            NaN             NaN
25%           2.000000        NaN            NaN             NaN
50%           3.000000        NaN            NaN             NaN
75%           4.000000        NaN            NaN             NaN
max           5.000000        NaN            NaN             NaN

        quantity_produced  production_cost
count            5.000000         5.000000
unique                NaN              NaN
top                   NaN              NaN
freq                  NaN              NaN
mean           820.000000     12240.360000
std            286.356421      2165.313852
min            500.000000      9000.300000
25%            600.000000     12000.500000
50%            800.000000     12200.000000
75%           1000.000000     13000.250000
max           1200.000000     15000.750000

Sales DataFrame Missing Values:
 sale_id          0
product_id       0
sale_date        0
quantity_sold    0
sale_price       0
total_revenue    0
dtype: int64

Inventory DataFrame Missing Values:
 inventory_id      0
product_id        0
warehouse_id      0
stock_quantity    0
last_updated      0
dtype: int64

Production DataFrame Missing Values:
 production_id        0
product_id           0
product_name         0
production_date      0
quantity_produced    0
production_cost      0
dtype: int64

Sales DataFrame Duplicate Rows: 0

Inventory DataFrame Duplicate Rows: 0

Production DataFrame Duplicate Rows: 0

Sales DataFrame Unique Product IDs: 4
Inventory DataFrame Unique Product IDs: 4
```

```
Production DataFrame Unique Product IDs: 4

Sales DataFrame Unique Sale Dates: 5
Inventory DataFrame Unique Update Dates: 3
Production DataFrame Unique Production Dates: 5

Product IDs in sales but not in production: set()
```

```python
# Data Exploration (Continued)

# 3. Descriptive Statistics (Continued)
print("\nProduction DataFrame Descriptive Statistics:\n",
production_df.describe(include='all'))

# 4. Missing Values
print("\nSales DataFrame Missing Values:\n", sales_df.isnull().sum())
print("\nInventory DataFrame Missing Values:\n", inventory_df.isnull().sum())
print("\nProduction DataFrame Missing Values:\n", production_df.isnull().sum())

# 5. Duplicate Rows
print("\nSales DataFrame Duplicate Rows:", sales_df.duplicated().sum())
print("\nInventory DataFrame Duplicate Rows:", inventory_df.duplicated().sum())
print("\nProduction DataFrame Duplicate Rows:", production_df.duplicated().sum())

# 6. Unique Values in Key Columns
print("\nSales DataFrame Unique Product IDs:", sales_df['product_id'].nunique())
print("Inventory DataFrame Unique Product IDs:",
inventory_df['product_id'].nunique())
print("Production DataFrame Unique Product IDs:",
production_df['product_id'].nunique())

print("\nSales DataFrame Unique Sale Dates:", sales_df['sale_date'].nunique())
print("Inventory DataFrame Unique Update Dates:",
inventory_df['last_updated'].nunique())
print("Production DataFrame Unique Production Dates:",
production_df['production_date'].nunique())

# Check for product_ids in sales that are not in production
sales_products = set(sales_df['product_id'].unique())
production_products = set(production_df['product_id'].unique())
missing_products = sales_products - production_products
print("\nProduct IDs in sales but not in production:", missing_products)
```

```
Production DataFrame Descriptive Statistics:
         production_id product_id   product_name production_date  \
count         5.000000          5              5               5
unique             NaN          4              4               5
top                NaN    PROD001  Beras Premium      2025-01-10
freq               NaN          2              2               1
mean          3.000000        NaN            NaN             NaN
std           1.581139        NaN            NaN             NaN
min           1.000000        NaN            NaN             NaN
25%           2.000000        NaN            NaN             NaN
50%           3.000000        NaN            NaN             NaN
75%           4.000000        NaN            NaN             NaN
max           5.000000        NaN            NaN             NaN

        quantity_produced  production_cost
count            5.000000         5.000000
unique                NaN              NaN
```

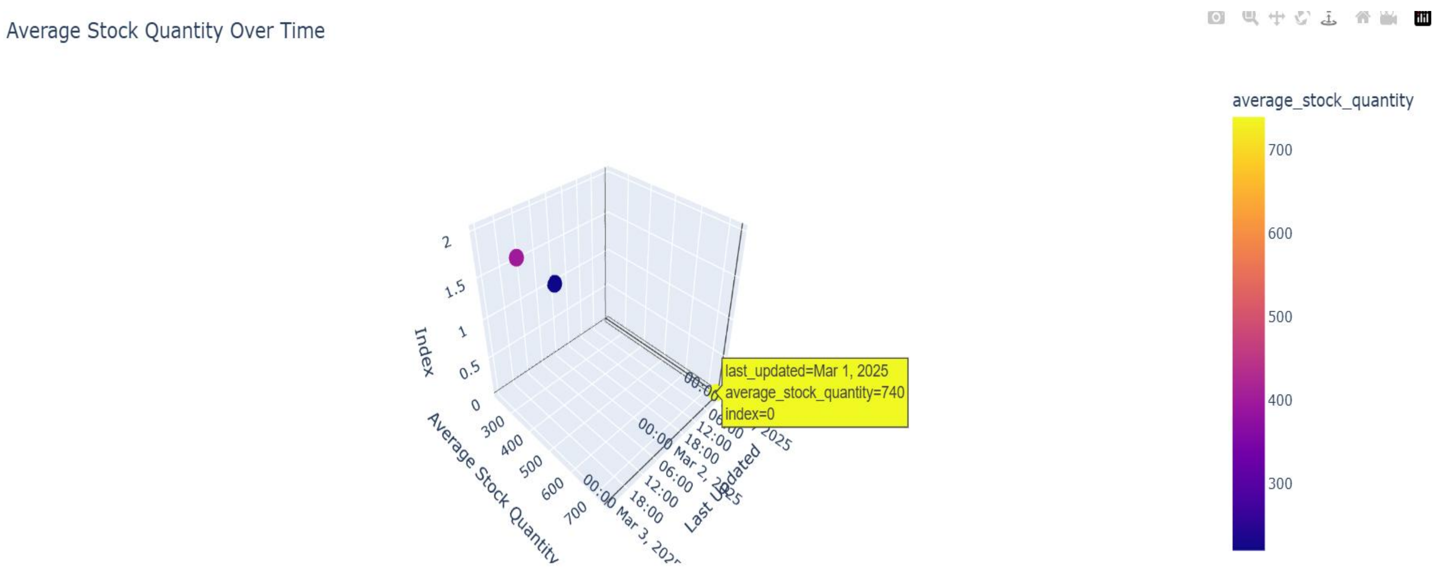
```
top                     NaN              NaN
freq                    NaN              NaN
mean             820.000000     12240.360000
std              286.356421      2165.313852
min              500.000000      9000.300000
25%              600.000000     12000.500000
50%              800.000000     12200.000000
75%             1000.000000     13000.250000
max             1200.000000     15000.750000

Sales DataFrame Missing Values:
 sale_id          0
product_id       0
sale_date        0
quantity_sold    0
sale_price       0
total_revenue    0
dtype: int64

Inventory DataFrame Missing Values:
 inventory_id      0
product_id        0
warehouse_id      0
stock_quantity    0
last_updated      0
dtype: int64

Production DataFrame Missing Values:
 production_id        0
product_id           0
product_name         0
production_date      0
quantity_produced    0
production_cost      0
dtype: int64

Sales DataFrame Duplicate Rows: 0

Inventory DataFrame Duplicate Rows: 0

Production DataFrame Duplicate Rows: 0

Sales DataFrame Unique Product IDs: 4
Inventory DataFrame Unique Product IDs: 4
Production DataFrame Unique Product IDs: 4

Sales DataFrame Unique Sale Dates: 5
Inventory DataFrame Unique Update Dates: 3
Production DataFrame Unique Production Dates: 5

Product IDs in sales but not in production: set()
```

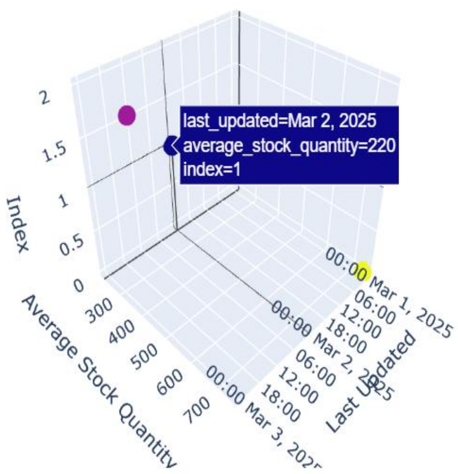
Data Analysis & Visualization

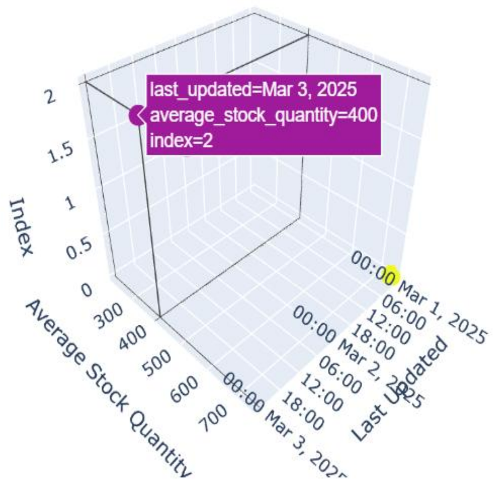
```
# Calculate total sales, revenue, production costs, and profit for each product
combined_df['total_sales'] =
combined_df.groupby('product_id')['quantity_sold'].transform('sum')
combined_df['total_revenue_per_product'] =
combined_df.groupby('product_id')['total_revenue'].transform('sum')
combined_df['total_production_cost_per_product'] =
combined_df.groupby('product_id')['production_cost'].transform('sum')
combined_df['total_profit_per_product'] = combined_df['total_revenue_per_product'] -
combined_df['total_production_cost_per_product']
```

```
# Analyze sales trends
sales_trends_df = combined_df.groupby('sale_date').agg({'quantity_sold': 'sum',
'total_revenue': 'sum'}).reset_index()
sales_trends_df.rename(columns={'quantity_sold': 'total_sales', 'total_revenue':
'total_revenue_on_date'}, inplace=True)

# Investigate inventory levels
inventory_trends_df = combined_df.groupby('last_updated').agg({'stock_quantity':
'mean'}).reset_index()
inventory_trends_df.rename(columns={'stock_quantity': 'average_stock_quantity'},
inplace=True)

# Calculate average profit margin
combined_df['profit_margin'] = (combined_df['total_revenue_per_product'] -
combined_df['total_production_cost_per_product']) /
combined_df['total_revenue_per_product']

# Explore correlations (example: sales quantity and production quantity)
correlation_sales_production =
combined_df['quantity_sold'].corr(combined_df['quantity_produced'])
print(f"Correlation between sales quantity and production quantity:
{correlation_sales_production}")

# Identify potential outliers (example: using IQR for profit margin)
Q1 = combined_df['profit_margin'].quantile(0.25)
Q3 = combined_df['profit_margin'].quantile(0.75)
IQR = Q3 - Q1
lower_bound = Q1 - 1.5 * IQR
upper_bound = Q3 + 1.5 * IQR
outliers = combined_df[(combined_df['profit_margin'] < lower_bound) |
(combined_df['profit_margin'] > upper_bound)]

display(combined_df.head())
display(sales_trends_df.head())
display(inventory_trends_df.head())
display(outliers.head())
import matplotlib.pyplot as plt
from mpl_toolkits.mplot3d import Axes3D
import plotly.express as px

# 3D Scatter Plot: total_revenue_per_product, total_production_cost_per_product,
total_profit_per_product
fig = plt.figure(figsize=(10, 8))
ax = fig.add_subplot(111, projection='3d')
scatter = ax.scatter(combined_df['total_revenue_per_product'],
combined_df['total_production_cost_per_product'],
combined_df['total_profit_per_product'],
c=combined_df['product_id'].astype('category').cat.codes)
ax.set_xlabel('Total Revenue per Product')
ax.set_ylabel('Total Production Cost per Product')
ax.set_zlabel('Total Profit per Product')
ax.set_title('3D Scatter Plot of Revenue, Cost, and Profit by Product')
plt.savefig('3d_scatter_revenue_cost_profit.png')
plt.show()
```

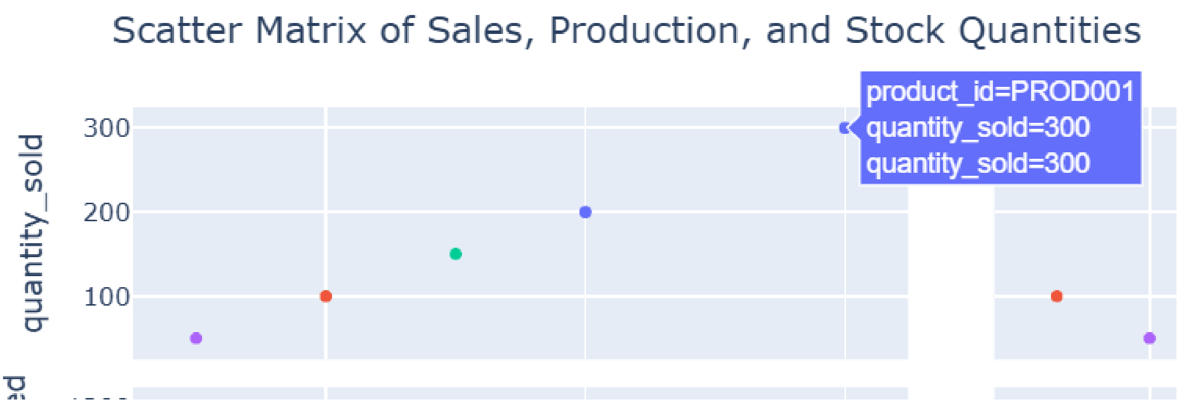
```python
# 3D Surface Plot: average_stock_quantity over time
fig = px.scatter_3d(inventory_trends_df, x='last_updated',
y='average_stock_quantity', z=inventory_trends_df.index,
color='average_stock_quantity')
fig.update_layout(title='Average Stock Quantity Over Time',
scene=dict(xaxis_title='Last Updated', yaxis_title='Average Stock Quantity',
zaxis_title='Index'))
fig.write_html('3d_surface_inventory_trends.html')
fig.show()


# Another 3D visualization (e.g., scatter matrix)
fig = px.scatter_matrix(combined_df, dimensions=['quantity_sold',
'quantity_produced', 'stock_quantity'], color='product_id')
fig.update_layout(title='Scatter Matrix of Sales, Production, and Stock Quantities')
fig.write_html("3d_scatter_matrix_sales_production_stock.html")
fig.show()
```

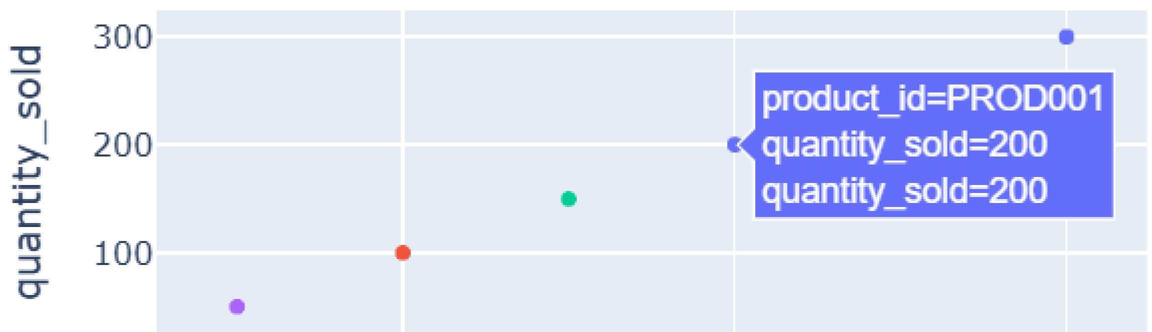


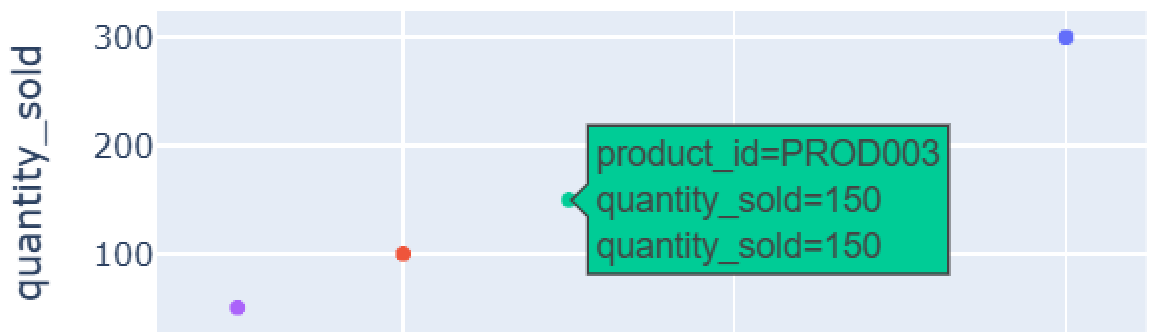




Sample (quantity_sold)







Summary

Temuan Kunci Analisis Data

Integrasi Data

Data penjualan, persediaan, dan produksi berhasil digabungkan ke dalam satu DataFrame (combined_df) menggunakan *inner join* berdasarkan product_id. Kolom tanggal diubah menjadi objek datetime. Tidak ditemukan nilai yang hilang dalam dataset gabungan.

Analisis Penjualan

Total penjualan, pendapatan, biaya produksi, dan laba dihitung per produk. Tren penjualan dianalisis berdasarkan tanggal, menunjukkan total penjualan dan pendapatan per hari.

Analisis Persediaan

Rata-rata tingkat persediaan dilacak dari waktu ke waktu.

Analisis Profitabilitas

Margin laba dihitung untuk setiap produk. *Outlier* dalam margin laba diidentifikasi menggunakan metode IQR.
Analisis Korelasi

Koefisien korelasi dihitung antara jumlah penjualan dan jumlah produksi.

Visualisasi:

Visualisasi 3D dibuat, termasuk *scatter plot* untuk pendapatan, biaya, dan laba; *surface plot* untuk rata-rata jumlah stok dari waktu ke waktu; dan *scatter matrix* untuk jumlah penjualan, produksi, dan stok.
Pembuatan Laporan

Laporan PDF dibuat yang berisi visualisasi dan deskripsi teks. DataFrame utama (combined_df, sales_trends_df, inventory_trends_df, dan outliers) disimpan sebagai file CSV.

Wawasan

Menyelidiki lebih lanjut korelasi antara jumlah penjualan dan jumlah produksi untuk mengoptimalkan perencanaan produksi. Koefisien korelasi telah dihitung, tetapi analisis lebih lanjut diperlukan untuk memahami implikasi praktisnya.