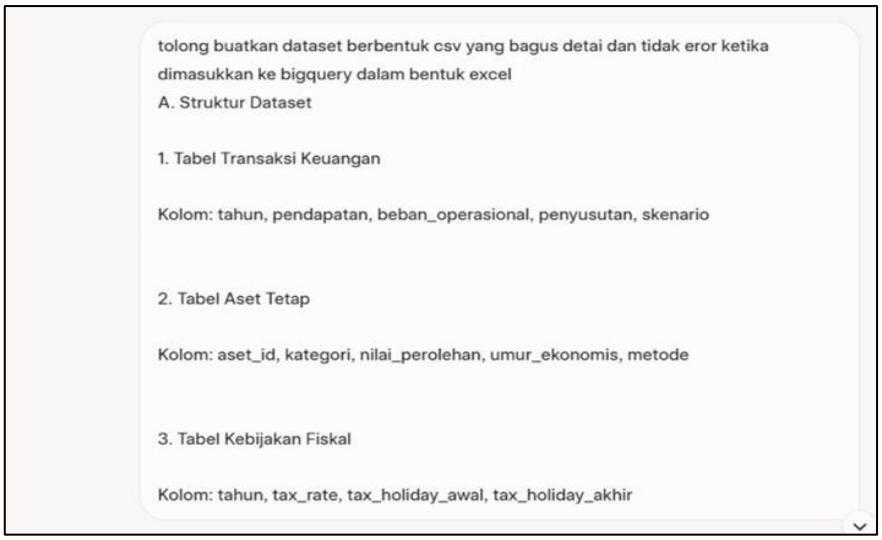
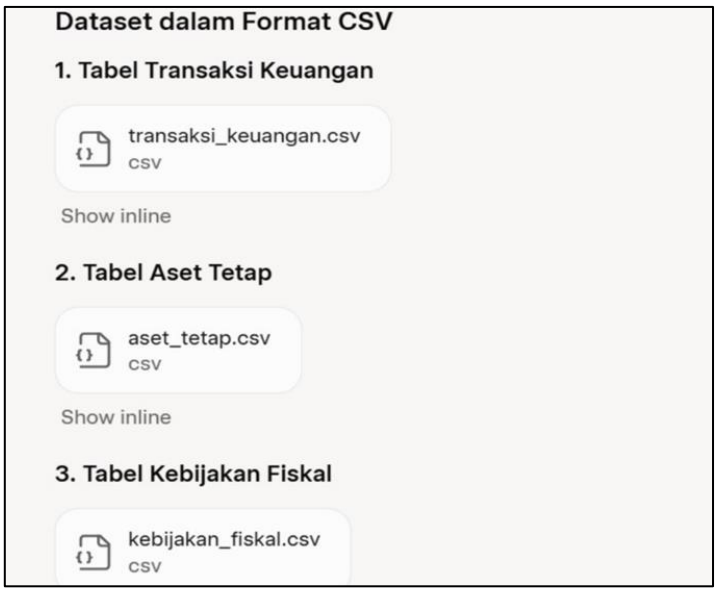
Nama : Fathinatul Hasanah
NIM : 12030123140294
Mata Kuliah : Pengkodean dan Pemrograman

Langkah-Langkah Simulasi PPh Badan Google Colab

1. **Meminta Dataset dalam Format CSV**
   Langkah pertama yang saya lakukan adalah meminta bantuan dari Grok AI untuk membuatkan dataset simulasi perhitungan PPh Badan dalam format CSV.



2. **Menerima Dataset dari Grok AI**
   Grok AI memberikan file dataset dalam bentuk CSV yang kemudian digunakan untuk proses analisis selanjutnya dan berikut isi file CSV dari Transaksi Keungan, Aset Tetap, Kenbijakan Fiskal.



- Transaksi Keuangan

tahun	pendapatan	beban_operasional	penyusutan	skenario
2020	1000000000	600000000	50000000	Optimistis
2020	900000000	650000000	45000000	Moderat
2020	800000000	700000000	40000000	Pesimistis
2021	1200000000	700000000	60000000	Optimistis
2021	1000000000	720000000	55000000	Moderat
2021	850000000	740000000	50000000	Pesimistis
2022	1500000000	800000000	70000000	Optimistis
2022	1300000000	820000000	65000000	Moderat
2022	1100000000	850000000	60000000	Pesimistis
2023	1800000000	900000000	80000000	Optimistis
2023	1600000000	920000000	75000000	Moderat
2023	1400000000	950000000	70000000	Pesimistis
2024	2000000000	1000000000	90000000	Optimistis
2024	1800000000	1020000000	85000000	Moderat
2024	1600000000	1050000000	80000000	Pesimistis

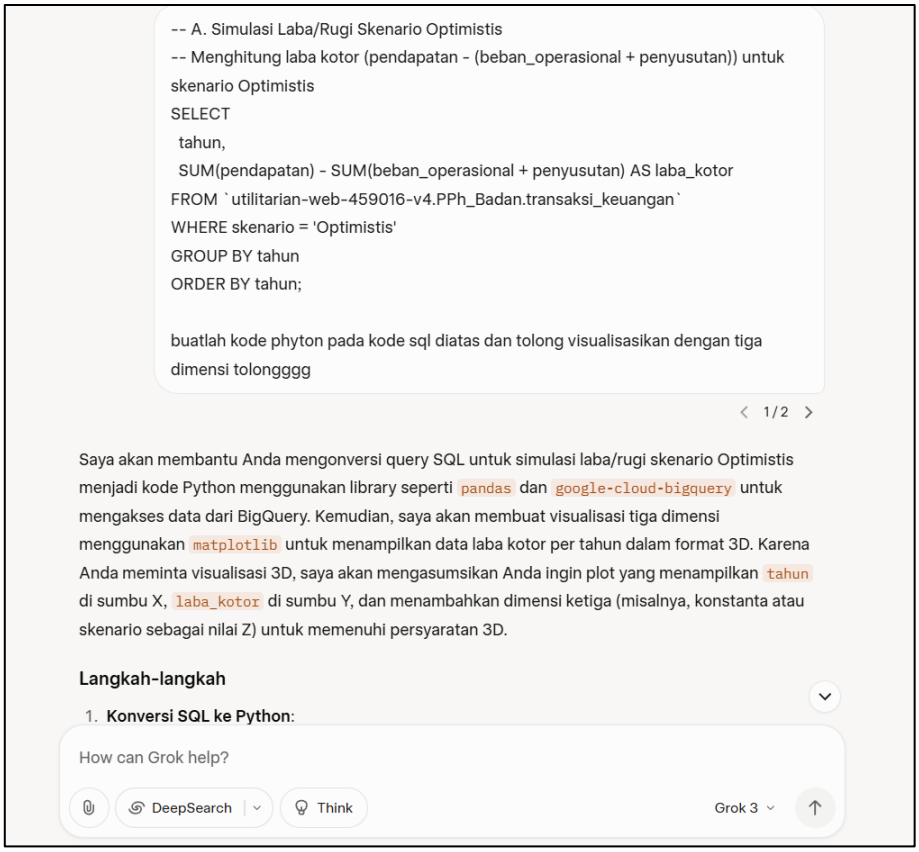
- Aset Tetap

aset_id	kategori	nilai_perolehan	umur_ekonomis	metode
AST001	Mesin	500000000	10	Garis_Lurus
AST002	Kendaraan	300000000	7	Garis_Lurus
AST003	Bangunan	2000000000	20	Garis_Lurus
AST004	Peralatan	150000000	5	Saldo_Menurun
AST005	Mesin	800000000	12	Garis_Lurus
AST006	Kendaraan	250000000	6	Garis_Lurus
AST007	Peralatan	100000000	4	Saldo_Menurun
AST008	Bangunan	2500000000	25	Garis_Lurus

- Kebijakan Fiskal

tahun	tax_rate	tax_holiday_awal	tax_holiday_akhir
2020	0.25	2018	2020
2021	0.22	2018	2020
2022	0.22	2021	2023
2023	0.20	2021	2023
2024	0.20	2024	2026
2025	0.20	2024	2026

3. **Pembuatan Kode Phyton**
   Meminta Grok AI untuk memberikan kode Phyton dari file CSV untuk simulasi perhitungan PPh Badan.



- **Simulasi Laba Rugi (Phyton)**

```
import matplotlib.pyplot as plt
from mpl_toolkits.mplot3d import Axes3D
import numpy as np
import pandas as pd

# Code to define df (copied from cell 5y6PombVps71)
try:
    from google.cloud import bigquery
    from google.colab import files
    import os
except ImportError as e:
    print(f"Error: Library tidak terinstal. Silakan jalankan: !pip install {str(e).split()[-1]}")
    # !pip install google-cloud-bigquery pandas matplotlib # Uncomment if needed
    from google.cloud import bigquery
    from google.colab import files
    import os

# Fungsi untuk mengunggah kredensial JSON di Colab
def upload_credentials():
    print("Unggah file kredensial JSON Anda:")
    uploaded = files.upload()
    if not uploaded:
```

```
        raise ValueError("File kredensial tidak diunggah!")
    credential_file = list(uploaded.keys())[0]
    os.environ['GOOGLE_APPLICATION_CREDENTIALS'] =
credential_file
    return credential_file

# Coba akses BigQuery
try:
    client = bigquery.Client()
except Exception as e:
    print(f"Error saat inisialisasi BigQuery: {str(e)}")
    print("Mari kita coba unggah kredensial JSON.")
    # upload_credentials() # Uncomment if needed
    client = None # Set client to None if upload fails

query = """
SELECT
  tahun,
  skenario,
  SUM(pendapatan) - SUM(beban_operasional + penyusutan) AS
laba_kena_pajak
FROM `utilitarian-web-459016-v4.PPh_Badan.transaksi_keuangan`
GROUP BY tahun, skenario
ORDER BY tahun, skenario;
"""

df = None # Initialize df to None
if client:
    try:
        query_job = client.query(query)
        df = query_job.to_dataframe()
    except Exception as e:
        print(f"Error saat menjalankan query: {str(e)}")
        print("Menggunakan data cadangan...")
        data = {
            'tahun': [2020, 2020, 2020, 2021, 2021, 2021,
2022, 2022, 2022, 2023, 2023, 2023, 2024, 2024, 2024],
            'skenario': ['Optimistis', 'Moderat',
'Pesimistis', 'Optimistis', 'Moderat', 'Pesimistis',
'Optimistis', 'Moderat', 'Pesimistis', 'Optimistis',
'Moderat', 'Pesimistis', 'Optimistis', 'Moderat',
'Pesimistis'],
            'laba_kena_pajak': [350000000, 205000000,
60000000, 440000000, 225000000, 60000000, 630000000,
435000000, 190000000, 820000000, 605000000, 380000000,
910000000, 695000000, 470000000]
        }
        df = pd.DataFrame(data)
```

```
else:
     print("Tidak dapat terhubung ke BigQuery. Menggunakan
data cadangan...")
     data = {
        'tahun': [2020, 2020, 2020, 2021, 2021, 2021, 2022,
2022, 2022, 2023, 2023, 2023, 2024, 2024, 2024],
        'skenario': ['Optimistis', 'Moderat', 'Pesimistis',
'Optimistis', 'Moderat', 'Pesimistis', 'Optimistis',
'Moderat', 'Pesimistis', 'Optimistis', 'Moderat',
'Pesimistis', 'Optimistis', 'Moderat', 'Pesimistis'],
        'laba_kena_pajak': [350000000, 205000000, 60000000,
440000000, 225000000, 60000000, 630000000, 435000000,
190000000, 820000000, 605000000, 380000000, 910000000,
695000000, 470000000]
     }
     df = pd.DataFrame(data)


# Map skenario ke nilai numerik untuk sumbu Z (optional for
this plot, but kept for consistency if needed later)
# skenario_map = {'Optimistis': 1, 'Moderat': 2, 'Pesimistis':
3}
# df['skenario_num'] = df['skenario'].map(skenario_map)


# Visualisasi 3D Bar Chart untuk semua skenario laba kotor
if df is not None and not df.empty:
    fig = plt.figure(figsize=(14, 10))
    ax = fig.add_subplot(111, projection='3d')

    # Dapatkan daftar tahun dan skenario unik dari DataFrame
df
    years = df['tahun'].unique()
    scenarios = df['skenario'].unique()

    # Buat posisi untuk batang pada sumbu x (tahun) dan y
(skenario)
    xpos, ypos = np.meshgrid(np.arange(len(years)),
np.arange(len(scenarios)))
    xpos = xpos.flatten()
    ypos = ypos.flatten()
    zpos = np.zeros(len(xpos))

    # Lebar dan kedalaman batang
    # dx dan dy harus sesuai dengan jumlah total batang (tahun
* skenario)
    dx = 0.5 * np.ones_like(xpos)
    dy = 0.5 * np.ones_like(ypos)
```

```
    # Tinggi batang (laba kotor) - gunakan data dari DataFrame
df
    # Need to ensure the order of dz matches the order of xpos
and ypos
    # Group by year and scenario to get the correct order
    df_grouped = df.groupby(['tahun',
'skenario'])['laba_kena_pajak'].sum().reset_index()
    dz = df_grouped['laba_kena_pajak'].values / 1e9 # Tinggi
batang (laba kotor dalam Miliar)


    # Mapping colors for each scenario - Using more vibrant
colors and edge colors
    colors = {'Optimistis': '#1F77B4', 'Moderat': '#2CA02C',
'Pesimistis': '#D62728'} # More vibrant colors
    # Create a list of colors for each bar based on the
scenario in df_grouped
    scenario_colors = [colors[skenario] for skenario in
df_grouped['skenario']]


    # Create 3D bar chart with enhanced aesthetics
    ax.bar3d(xpos, ypos, zpos, dx, dy, dz,
color=scenario_colors, alpha=0.9, edgecolors='black') #
Increased alpha and added edgecolors

    # Set axis labels
    ax.set_xlabel('Tahun', fontsize=12)
    ax.set_ylabel('Skenario', fontsize=12)
    ax.set_zlabel('Laba Kena Pajak (Miliar Rupiah)',
fontsize=12) # Changed label to Laba Kena Pajak

    # Set ticks and labels for axes
    x_tick_positions = np.arange(len(years)) + dx[0]/2
    ax.set_xticks(x_tick_positions)
    ax.set_xticklabels(years)

    y_tick_positions = np.arange(len(scenarios)) + dy[0]/2
    ax.set_yticks(y_tick_positions)
    ax.set_yticklabels(scenarios)


    # Title
    ax.set_title('Tren Laba Kena Pajak per Tahun (3D Bar
Chart)', fontsize=16) # Changed title

    # Add manual legend
    import matplotlib.patches as mpatches
```

```
    legend_patches = [mpatches.Patch(color=colors[skenario],
label=skenario) for skenario in scenarios]
    ax.legend(handles=legend_patches)

    # Adjust view angle for better perspective
    ax.view_init(elev=25, azim=-55) # Adjusted view angle

    # Add grid for better readability
    ax.grid(True, linestyle='--', alpha=0.6)

    plt.show()
else:
    print("DataFrame kosong atau tidak tersedia untuk
visualisasi.")
```

- **Simulasi Depresiasi Metode Garis Lurus (Phyton)**

```
# Impor library yang diperlukan (pastikan semua sudah
terinstal)
try:
    from google.cloud import bigquery
    import pandas as pd
    import matplotlib.pyplot as plt
    from mpl_toolkits.mplot3d import Axes3D
    from google.colab import files
    import os
except ImportError as e:
    print(f"Error: Library tidak terinstal. Silakan jalankan:
!pip install {str(e).split()[-1]}")
    !pip install google-cloud-bigquery pandas matplotlib
    from google.cloud import bigquery
    import pandas as pd
    import matplotlib.pyplot as plt
    from mpl_toolkits.mplot3d import Axes3D
    from google.colab import files
    import os

# Coba akses BigQuery atau gunakan data cadangan untuk metode
Garis Lurus
try:
    client = bigquery.Client()
    # Query SQL untuk simulasi depresiasi metode Garis Lurus
    query = """
    SELECT
      aset_id,
      kategori,
      nilai_perolehan,
      umur_ekonomis,
      ROUND(nilai_perolehan / umur_ekonomis, 2) AS
depresiasi_tahunan
```

```
    FROM `utilitarian-web-459016-v4.PPh_Badan.aset_tetap`
    WHERE metode = 'Garis_Lurus'
    ORDER BY aset_id;
    """
    query_job = client.query(query)
    df_garis_lurus = query_job.to_dataframe()
except Exception as e:
    print(f"Error saat menjalankan query BigQuery: {str(e)}")
    print("Menggunakan data cadangan dari CSV untuk metode
Garis Lurus...")
    # Data cadangan berdasarkan aset_tetap.csv untuk metode
Garis Lurus
    data_garis_lurus = {
        'aset_id': ['AST001', 'AST002', 'AST003', 'AST005',
'AST006', 'AST008'],
        'kategori': ['Mesin', 'Kendaraan', 'Bangunan',
'Mesin', 'Kendaraan', 'Bangunan'],
        'nilai_perolehan': [500000000, 300000000, 2000000000,
800000000, 250000000, 2500000000],
        'umur_ekonomis': [10, 7, 20, 12, 6, 25],
        'depresiasi_tahunan': [50000000.00, 42857142.86,
100000000.00, 66666666.67, 41666666.67, 100000000.00]
    }
    df_garis_lurus = pd.DataFrame(data_garis_lurus)

# Cetak DataFrame untuk verifikasi
print("Data Depresiasi Metode Garis Lurus:")
print(df_garis_lurus)

# Persiapkan data untuk visualisasi 3D Bar Chart
if df_garis_lurus is not None and not df_garis_lurus.empty:
    aset_id = df_garis_lurus['aset_id']
    umur_ekonomis = df_garis_lurus['umur_ekonomis']
    depresiasi_tahunan = df_garis_lurus['depresiasi_tahunan']

    # Buat visualisasi 3D Bar Chart
    try:
        fig = plt.figure(figsize=(12, 8))
        ax = fig.add_subplot(111, projection='3d')

        # Buat posisi untuk batang
        xpos = np.arange(len(aset_id))
        ypos = umur_ekonomis.values
        zpos = np.zeros_like(xpos)

        # Lebar dan kedalaman batang
        dx = 0.8 * np.ones_like(zpos)
        dy = 1.0 * np.ones_like(zpos)
```

```
            dz = depresiasi_tahunan.values / 1e6 # Tinggi batang
(depresiasi dalam Juta Rupiah)

            # Warna batang (bisa disesuaikan, misalnya berdasarkan
kategori atau nilai depresiasi)
            colors = plt.cm.viridis(depresiasi_tahunan /
depresiasi_tahunan.max())

            # Buat 3D bar chart
            ax.bar3d(xpos, ypos, zpos, dx, dy, dz, color=colors,
alpha=0.8)

            # Atur label sumbu
            ax.set_xlabel('Aset ID', fontsize=12)
            ax.set_ylabel('Umur Ekonomis (Tahun)', fontsize=12)
            ax.set_zlabel('Depresiasi Tahunan (Juta Rupiah)',
fontsize=12)

            # Atur tick dan label sumbu X (Aset ID)
            ax.set_xticks(xpos + dx[0]/2)
            ax.set_xticklabels(aset_id)

            # Judul plot
            ax.set_title('Simulasi Depresiasi Metode Garis Lurus
(3D Bar Chart)', fontsize=16)

            # Tambahkan colorbar jika menggunakan colormap
            cbar =
fig.colorbar(plt.cm.ScalarMappable(cmap='viridis',
norm=plt.Normalize(vmin=depresiasi_tahunan.min(),
vmax=depresiasi_tahunan.max())), ax=ax, pad=0.1)
            cbar.set_label('Depresiasi Tahunan (Rp)')

            # Sesuaikan sudut pandang untuk kejelasan
            ax.view_init(elev=15, azim=-55) # Mengubah sudut
pandang

            plt.show()
        except Exception as e:
            print(f"Error saat membuat visualisasi: {str(e)}")
            print("Pastikan matplotlib dan mpl_toolkits.mplot3d
terinstal dengan benar.")
    else:
        print("DataFrame untuk metode Garis Lurus kosong atau
tidak tersedia untuk visualisasi.")
```

- **Simulasi Depresiasi Metode Saldo Menurun (Phyton)**

```
# Impor library yang diperlukan (pastikan semua sudah terinstal)
try:
```

```
    from google.cloud import bigquery
    import pandas as pd
    import matplotlib.pyplot as plt
    from mpl_toolkits.mplot3d import Axes3D
    from google.colab import files
    import os
except ImportError as e:
    print(f"Error: Library tidak terinstal. Silakan jalankan:
   !pip install {str(e).split()[-1]}")
    !pip install google-cloud-bigquery pandas matplotlib
    from google.cloud import bigquery
    import pandas as pd
    import matplotlib.pyplot as plt
    from mpl_toolkits.mplot3d import Axes3D
    from google.colab import files
    import os

# Coba akses BigQuery atau gunakan data cadangan untuk metode
   Saldo Menurun
try:
    client = bigquery.Client()
    # Query SQL untuk simulasi depresiasi metode Saldo Menurun
    query = """
    SELECT
      aset_id,
      kategori,
      nilai_perolehan,
      umur_ekonomis,
      ROUND(nilai_perolehan * 0.25, 2) AS
   depresiasi_tahun_pertama,
      ROUND(nilai_perolehan * (2.0 / umur_ekonomis), 2) AS
   depresiasi_tahun_pertama_alternatif
    FROM `utilitarian-web-459016-v4.PPh_Badan.aset_tetap`
    WHERE metode = 'Saldo_Menurun'
    ORDER BY aset_id;
    """
    query_job = client.query(query)
    df_saldo_menurun = query_job.to_dataframe()
except Exception as e:
    print(f"Error saat menjalankan query BigQuery: {str(e)}")
    print("Menggunakan data cadangan dari CSV untuk metode Saldo
   Menurun...")
    # Data cadangan berdasarkan aset_tetap.csv untuk metode
   Saldo Menurun
    data_saldo_menurun = {
        'aset_id': ['AST004', 'AST007'],
        'kategori': ['Peralatan', 'Peralatan'],
        'nilai_perolehan': [150000000, 100000000],
```

```
        'umur_ekonomis': [5, 4],
        'depresiasi_tahun_pertama': [37500000.00, 25000000.00],
        'depresiasi_tahun_pertama_alternatif': [60000000.00,
    50000000.00]
     }
     df_saldo_menurun = pd.DataFrame(data_saldo_menurun)

# Cetak DataFrame untuk verifikasi
print("Data Depresiasi Metode Saldo Menurun:")
print(df_saldo_menurun)

# Persiapkan data untuk visualisasi 3D Bar Chart
if df_saldo_menurun is not None and not df_saldo_menurun.empty:
     aset_id = df_saldo_menurun['aset_id']
     umur_ekonomis = df_saldo_menurun['umur_ekonomis']
     depresiasi_tahun_pertama =
    df_saldo_menurun['depresiasi_tahun_pertama']

     # Buat visualisasi 3D Bar Chart
     try:
         fig = plt.figure(figsize=(12, 8))
         ax = fig.add_subplot(111, projection='3d')

         # Buat posisi untuk batang
         xpos = np.arange(len(aset_id))
         ypos = umur_ekonomis.values
         zpos = np.zeros_like(xpos)

         # Lebar dan kedalaman batang
         dx = 0.8 * np.ones_like(zpos)
         dy = 1.0 * np.ones_like(zpos)
         dz = depresiasi_tahun_pertama.values / 1e6 # Tinggi
    batang (depresiasi dalam Juta Rupiah)

         # Warna batang (bisa disesuaikan, misalnya berdasarkan
    kategori atau nilai depresiasi)
         colors = plt.cm.plasma(depresiasi_tahun_pertama /
    depresiasi_tahun_pertama.max())

         # Buat 3D bar chart
         ax.bar3d(xpos, ypos, zpos, dx, dy, dz, color=colors,
    alpha=0.8)

         # Atur label sumbu
         ax.set_xlabel('Aset ID', fontsize=12)
         ax.set_ylabel('Umur Ekonomis (Tahun)', fontsize=12)
         ax.set_zlabel('Depresiasi Tahun Pertama (Juta Rupiah)',
    fontsize=12)
```

```
            # Atur tick dan label sumbu X (Aset ID)
            ax.set_xticks(xpos + dx[0]/2)
            ax.set_xticklabels(aset_id)

            # Judul plot
            ax.set_title('Simulasi Depresiasi Metode Saldo Menurun
        (3D Bar Chart)', fontsize=16)

            # Tambahkan colorbar jika menggunakan colormap
            cbar = fig.colorbar(plt.cm.ScalarMappable(cmap='plasma',
        norm=plt.Normalize(vmin=depresiasi_tahun_pertama.min(),
        vmax=depresiasi_tahun_pertama.max())), ax=ax, pad=0.1)
            cbar.set_label('Depresiasi Tahun Pertama (Rp)')

            # Sesuaikan sudut pandang
            ax.view_init(elev=20, azim=-60) # Mengubah sudut pandang

            plt.show()
        except Exception as e:
            print(f"Error saat membuat visualisasi: {str(e)}")
            print("Pastikan matplotlib dan mpl_toolkits.mplot3d
        terinstal dengan benar.")
    else:
        print("DataFrame untuk metode Saldo Menurun kosong atau
        tidak tersedia untuk visualisasi.")
```

- **Simulasi Tax Holiday (Phyton)**

```
# Visualisasi 3D Bar Chart untuk simulasi Tax Holiday
fig = plt.figure(figsize=(14, 10))
ax = fig.add_subplot(111, projection='3d')

# Dapatkan daftar tahun dan skenario unik dari data tax holiday
years_th = df['tahun'].unique()
scenarios_th = df['skenario'].unique()

# Buat posisi untuk batang pada sumbu x (tahun) dan y
(skenario)
xpos_th, ypos_th = np.meshgrid(np.arange(len(years_th)),
np.arange(len(scenarios_th)))
xpos_th = xpos_th.flatten()
ypos_th = ypos_th.flatten()
zpos_th = np.zeros(len(xpos_th))

# Lebar dan kedalaman batang
dx_th = 0.5 * np.ones_like(zpos_th)
dy_th = 0.5 * np.ones_like(zpos_th)
```

```
# Tinggi batang (PPh Badan) - gunakan data dari DataFrame df
dz_th = df['pph_badan'].values

# Mapping warna untuk setiap skenario (gunakan warna yang sudah
ada atau definisikan lagi)
colors_th = {'Optimistis': 'dodgerblue', 'Moderat':
'limegreen', 'Pesimistis': 'salmon'}
scenario_colors_th = [colors_th[skenario] for skenario in
df['skenario']]

# Buat 3D bar chart
ax.bar3d(xpos_th, ypos_th, zpos_th, dx_th, dy_th, dz_th,
color=scenario_colors_th, alpha=0.8)

# Atur label sumbu
ax.set_xlabel('Tahun', fontsize=12)
ax.set_ylabel('Skenario', fontsize=12)
ax.set_zlabel('PPh Badan (Rp)', fontsize=12)

# Atur tick dan label sumbu. Ticks harus berada di tengah
kelompok batang untuk setiap tahun dan skenario.
x_tick_positions_th = np.arange(len(years_th)) + dx_th[0]/2
ax.set_xticks(x_tick_positions_th)
ax.set_xticklabels(years_th)

y_tick_positions_th = np.arange(len(scenarios_th)) + dy_th[0]/2
ax.set_yticks(y_tick_positions_th)
ax.set_yticklabels(scenarios_th)

# Judul plot
ax.set_title('Simulasi PPh Badan per Tahun (3D Bar Chart Tax
Holiday)', fontsize=16)

# Tambahkan legenda manual
import matplotlib.patches as mpatches
legend_patches_th = [mpatches.Patch(color=colors_th[skenario],
label=skenario) for skenario in scenarios_th]
ax.legend(handles=legend_patches_th)

# Sesuaikan sudut pandang
ax.view_init(elev=20, azim=-45)

plt.show()
```

- **Simulasi Tren Laba Rugi Bersih (Phyton)**

```
import matplotlib.pyplot as plt
from mpl_toolkits.mplot3d import Axes3D
```

```
import numpy as np
import pandas as pd

# Ensure df_grouped is available (from previous cell execution)
# If not, you would need to recreate it from the data
try:
    df_grouped
except NameError:
    # Recreate df_grouped using the backup data if it doesn't
exist
    print("df_grouped not found. Recreating from backup
data...")
    data = {
        'tahun': [2020, 2020, 2020, 2021, 2021, 2021, 2022,
2022, 2022, 2023, 2023, 2023, 2024, 2024, 2024],
        'skenario': ['Optimistis', 'Moderat', 'Pesimistis',
'Optimistis', 'Moderat', 'Pesimistis', 'Optimistis', 'Moderat',
'Pesimistis', 'Optimistis', 'Moderat', 'Pesimistis',
'Optimistis', 'Moderat', 'Pesimistis'],
        'laba_kotor': [350000000, 205000000, 60000000,
440000000, 225000000, 60000000, 630000000, 435000000,
190000000, 820000000, 605000000, 380000000, 910000000,
695000000, 470000000]
    }
    df = pd.DataFrame(data)
    df_grouped = df.groupby(['tahun',
'skenario'])['laba_kotor'].sum().reset_index()


if df_grouped is not None and not df_grouped.empty:
    fig = plt.figure(figsize=(14, 10))
    ax = fig.add_subplot(111, projection='3d')

    # Get the years and scenarios from the DataFrame
    years = df_grouped['tahun'].unique()
    scenarios = df_grouped['skenario'].unique()

    # Create positions for the bars on the x (year) and y
(skenario) axes
    xpos, ypos = np.meshgrid(np.arange(len(years)),
np.arange(len(scenarios)))
    xpos = xpos.flatten()
    ypos = ypos.flatten()
    zpos = np.zeros(len(xpos))

    # Width and depth of the bars
    dx = 0.5 * np.ones_like(zpos)
    dy = 0.5 * np.ones_like(zpos)
```

```
    # Heights of the bars (laba kotor)
    # Need to ensure the order of dz matches the order of xpos
and ypos
    # Create a list of laba kotor values in the correct order
(year by year, then scenario by scenario)
    dz = []
    for year in years:
        df_year = df_grouped[df_grouped['tahun'] == year]
        for skenario in scenarios:
            # Find the laba_kotor for the specific year and
skenario
            laba_kotor_value = df_year[df_year['skenario'] ==
skenario]['laba_kotor'].sum() # Use sum in case of multiple
entries (though shouldn't happen with this grouping)
            dz.append(laba_kotor_value / 1e6) # Convert to
Millions


    # Mapping colors for each scenario (using more vibrant
colors)
    colors = {'Optimistis': '#1F77B4', 'Moderat': '#2CA02C',
'Pesimistis': '#D62728'} # More vibrant colors
    # Create a list of colors corresponding to each bar based
on the scenario
    scenario_colors = [colors[skenario] for year in years for
skenario in scenarios] # Match the order of dz


    # Create 3D bar chart with enhanced aesthetics
    ax.bar3d(xpos, ypos, zpos, dx, dy, dz,
color=scenario_colors, alpha=0.9, edgecolors='black') #
Increased alpha and added edgecolors

    # Set axis labels
    ax.set_xlabel('Tahun', fontsize=12)
    ax.set_ylabel('Skenario', fontsize=12)
    ax.set_zlabel('Laba Kotor Bersih (Juta Rupiah)',
fontsize=12) # Changed label

    # Set ticks and labels for axes
    x_tick_positions = np.arange(len(years)) + dx[0]/2
    ax.set_xticks(x_tick_positions)
    ax.set_xticklabels(years)

    y_tick_positions = np.arange(len(scenarios)) + dy[0]/2
    ax.set_yticks(y_tick_positions)
    ax.set_yticklabels(scenarios)

    # Title
```

```
    ax.set_title('Tren Laba Kotor Bersih per Tahun (3D Bar
Chart Menarik)', fontsize=16) # Changed title

    # Add manual legend
    import matplotlib.patches as mpatches
    legend_patches = [mpatches.Patch(color=colors[skenario],
label=skenario) for skenario in scenarios]
    ax.legend(handles=legend_patches)

    # Adjust view angle for better perspective
    ax.view_init(elev=25, azim=-55) # Adjusted view angle

    # Add grid for better readability
    ax.grid(True, linestyle='--', alpha=0.6)

    plt.show()
else:
    print("DataFrame kosong atau tidak tersedia untuk
visualisasi.")
```

- **Simulasi Perbandingan PPh Badan per Tahun (Phyton)**

```
import matplotlib.pyplot as plt
from mpl_toolkits.mplot3d import Axes3D
import numpy as np
import pandas as pd

# Assuming df_final from cell 69cd968f contains the data for
PPh comparison
# If not, you would need to run cell 69cd968f first to get
df_final
# For robustness, let's regenerate df_final here based on the
logic in 69cd968f
try:
    from google.cloud import bigquery
    import pandas as pd
    import matplotlib.pyplot as plt
    from mpl_toolkits.mplot3d import Axes3D
    from google.colab import files
    import os
except ImportError as e:
    print(f"Error: Library tidak terinstal. Silakan jalankan:
!pip install {str(e).split()[-1]}")
    !pip install google-cloud-bigquery pandas matplotlib
    from google.cloud import bigquery
    import pandas as pd
    import matplotlib.pyplot as plt
    from mpl_toolkits.mplot3d import Axes3d
    from google.colab import files
```

```
    import os

# Dummy data for demonstration (can be replaced with actual
data loading)
data_transaksi_keuangan = {
    'tahun': [2020, 2020, 2020, 2021, 2021, 2021, 2022, 2022,
2022, 2023, 2023, 2023, 2024, 2024, 2024],
    'skenario': ['Optimistis', 'Moderat', 'Pesimistis',
'Optimistis', 'Moderat', 'Pesimistis', 'Optimistis', 'Moderat',
'Pesimistis', 'Optimistis', 'Moderat', 'Pesimistis',
'Optimistis', 'Moderat', 'Pesimistis'],
    'pendapatan': [1000000000, 900000000, 800000000,
1200000000, 1000000000, 850000000, 1500000000, 1300000000,
1100000000, 1800000000, 1600000000, 1400000000, 2000000000,
1800000000, 1600000000],
    'beban_operasional': [600000000, 650000000, 700000000,
700000000, 720000000, 740000000, 800000000, 820000000,
850000000, 900000000, 920000000, 950000000, 1000000000,
1020000000, 1050000000],
    'penyusutan': [50000000, 45000000, 40000000, 60000000,
55000000, 50000000, 70000000, 65000000, 60000000, 80000000,
75000000, 70000000, 90000000, 85000000, 80000000]
}
df_transaksi = pd.DataFrame(data_transaksi_keuangan)

data_aset_tetap = { # Include aset_tetap data for depreciation
methods
    'aset_id': ['AST001', 'AST002', 'AST003', 'AST004',
'AST005', 'AST006', 'AST007', 'AST008'],
    'metode': ['Garis_Lurus', 'Garis_Lurus', 'Garis_Lurus',
'Saldo_Menurun', 'Garis_Lurus', 'Garis_Lurus', 'Saldo_Menurun',
'Garis_Lurus'],
    'nilai_perolehan': [500000000, 300000000, 2000000000,
150000000, 800000000, 250000000, 100000000, 2500000000],
    'umur_ekonomis': [10, 7, 20, 5, 12, 6, 4, 25]
}
df_aset = pd.DataFrame(data_aset_tetap)

data_kebijakan_fiskal = {
    'tahun': [2020, 2021, 2022, 2023, 2024],
    'tax_rate': [0.22, 0.22, 0.22, 0.22, 0.22], # Example tax
rate
    'tax_holiday_awal': [2018, 2021, 2018, 2021, 2024], #
Example tax holiday start years
    'tax_holiday_akhir': [2020, 2023, 2020, 2023, 2026] #
Example tax holiday end years
}
df_kebijakan = pd.DataFrame(data_kebijakan_fiskal)
```

```
# Filter for 'Moderat' scenario to match previous PPh
comparison
df_transaksi_moderat = df_transaksi[df_transaksi['skenario'] ==
'Moderat'].copy()


# Calculate initial laba_kena_pajak components
df_laba_kotor = df_transaksi_moderat.groupby('tahun').agg(
    pendapatan=('pendapatan', 'sum'),
    beban_operasional=('beban_operasional', 'sum'),
    penyusutan_normal=('penyusutan', 'sum') # Use original
penyusutan for normal calculation
).reset_index()
df_laba_kotor['skenario'] = 'Moderat'

# Calculate depreciation based on different methods from
aset_tetap
df_aset['depresiasi_garis_lurus_tahunan'] = df_aset.apply(
    lambda row: row['nilai_perolehan'] / row['umur_ekonomis']
if row['metode'] == 'Garis_Lurus' else 0, axis=1
)
df_aset['depresiasi_saldo_menurun_tahun_pertama_alternatif'] =
df_aset.apply(
    lambda row: row['nilai_perolehan'] * (2.0 /
row['umur_ekonomis']) if row['metode'] == 'Saldo_Menurun' else
0, axis=1
)

# Sum up the depreciation across all assets (simplification)
total_depresiasi_garis_lurus_tahunan =
df_aset['depresiasi_garis_lurus_tahunan'].sum()
total_depresiasi_saldo_menurun_tahun_pertama_alternatif_sum =
df_aset['depresiasi_saldo_menurun_tahun_pertama_alternatif'].su
m()

# Calculate laba_kena_pajak for each method (applying total
depreciation to each year's laba kotor components)
df_laba_kotor['laba_kena_pajak_normal'] =
df_laba_kotor['pendapatan'] -
(df_laba_kotor['beban_operasional'] +
df_laba_kotor['penyusutan_normal'])
df_laba_kotor['laba_kena_pajak_garis_lurus'] =
df_laba_kotor['pendapatan'] -
df_laba_kotor['beban_operasional'] -
total_depresiasi_garis_lurus_tahunan
df_laba_kotor['laba_kena_pajak_saldo_menurun'] =
df_laba_kotor['pendapatan'] -
df_laba_kotor['beban_operasional'] -
```

```
total_depresiasi_saldo_menurun_tahun_pertama_alternatif_sum #
Using alternative as per SQL


# Join with kebijakan_fiskal to calculate PPh
df_result = pd.merge(df_laba_kotor, df_kebijakan, on='tahun',
how='left')

# Calculate PPh Badan for each method
df_result['pph_normal'] = df_result['laba_kena_pajak_normal'] *
df_result['tax_rate']

# Calculate PPh with Tax Holiday
df_result['pph_tax_holiday'] = df_result.apply(
    lambda row: 0 if row['tahun'] >= row['tax_holiday_awal']
and row['tahun'] <= row['tax_holiday_akhir'] else
row['laba_kena_pajak_normal'] * row['tax_rate'], axis=1
)

# Calculate PPh with Garis Lurus depreciation (applying tax
holiday)
df_result['pph_garis_lurus'] = df_result.apply(
    lambda row: 0 if row['tahun'] >= row['tax_holiday_awal']
and row['tahun'] <= row['tax_holiday_akhir'] else
row['laba_kena_pajak_garis_lurus'] * row['tax_rate'], axis=1
)

# Calculate PPh with Saldo Menurun depreciation (applying tax
holiday)
df_result['pph_saldo_menurun'] = df_result.apply(
    lambda row: 0 if row['tahun'] >= row['tax_holiday_awal']
and row['tahun'] <= row['tax_holiday_akhir'] else
row['laba_kena_pajak_saldo_menurun'] * row['tax_rate'], axis=1
)


# Select and reorder columns to match SQL output structure
df_pph_comparison = df_result[[
    'tahun',
    'skenario',
    'pph_normal',
    'pph_tax_holiday',
    'pph_garis_lurus',
    'pph_saldo_menurun'
]]


# Prepare data for 3D Bar Chart
```

```
if df_pph_comparison is not None and not
df_pph_comparison.empty:
    fig = plt.figure(figsize=(14, 10))
    ax = fig.add_subplot(111, projection='3d')

    years = df_pph_comparison['tahun'].unique()
    methods = ['pph_normal', 'pph_tax_holiday',
'pph_garis_lurus', 'pph_saldo_menurun']
    method_labels = ['Normal', 'Tax Holiday', 'Garis Lurus',
'Saldo Menurun']

    # Create positions for the bars
    xpos, ypos = np.meshgrid(np.arange(len(years)),
np.arange(len(methods)))
    xpos = xpos.flatten()
    ypos = ypos.flatten()
    zpos = np.zeros(len(xpos))

    # Width and depth of the bars
    dx = 0.5 * np.ones_like(xpos)
    dy = 0.5 * np.ones_like(ypos)

    # Heights of the bars (PPh Badan)
    # Need to ensure the order of dz matches the order of xpos
and ypos
    # Create a list of PPh values in the correct order (year by
year, then method by method)
    dz = []
    for year in years:
        df_year = df_pph_comparison[df_pph_comparison['tahun']
== year]
        for method in methods:
            # Ensure the method column exists and handle
potential missing values
            if method in df_year.columns:
                # Use .iloc[0] as there should only be one row
per year in this grouped data
                dz.append(df_year[method].iloc[0] / 1e6) #
Convert to Millions
            else:
                dz.append(0) # Append 0 if method column is
missing (shouldn't happen with this data)


    # Mapping colors for each method
    colors = ['#1f77b4', '#ff7f0e', '#2ca02c', '#d62728'] #
Example colors
    # Create a list of colors corresponding to each bar based
on the method
```

```
    method_colors = [colors[i % len(colors)] for i in ypos]


    # Create 3D bar chart with enhanced aesthetics
    ax.bar3d(xpos, ypos, zpos, dx, dy, dz, color=method_colors,
alpha=0.9, edgecolors='black')

    # Set axis labels
    ax.set_xlabel('Tahun', fontsize=12)
    ax.set_ylabel('Metode', fontsize=12)
    ax.set_zlabel('PPh Badan (Juta Rupiah)', fontsize=12)

    # Set ticks and labels for axes
    x_tick_positions = np.arange(len(years)) + dx[0]/2
    ax.set_xticks(x_tick_positions)
    ax.set_xticklabels(years)

    y_tick_positions = np.arange(len(methods)) + dy[0]/2
    ax.set_yticks(y_tick_positions)
    ax.set_yticklabels(method_labels)


    # Title
    ax.set_title('Perbandingan PPh Badan per Tahun dan Metode
(3D Bar Chart)', fontsize=16)

    # Add manual legend
    import matplotlib.patches as mpatches
    legend_patches = [mpatches.Patch(color=colors[i],
label=method_labels[i]) for i in range(len(methods))]
    ax.legend(handles=legend_patches)

    # Adjust view angle
    ax.view_init(elev=25, azim=-45)

    # Add grid
    ax.grid(True, linestyle='--', alpha=0.6)

    plt.show()
else:
    print("DataFrame untuk perbandingan PPh kosong atau tidak
tersedia untuk visualisasi.")
```

- **Tren Arus Kas Setelah Pajak per Tahun (Phyton)**

```
import matplotlib.pyplot as plt
from mpl_toolkits.mplot3d import Axes3D
import numpy as np
import pandas as pd
```

```
# Code from cell 0e1e8ee5 to define df_final
# Assuming you have DataFrames for 'transaksi_keuangan' and
'kebijakan_fiskal'
# If not, you would need to load them first (e.g., from CSV,
BigQuery, etc.)
# For demonstration, let's use dummy DataFrames based on the
structure implied by the query
# Replace this with your actual data loading code

# Dummy data for demonstration (can be replaced with actual
data loading)
data_transaksi_keuangan = {
    'tahun': [2020, 2020, 2020, 2021, 2021, 2021, 2022, 2022,
2022, 2023, 2023, 2023, 2024, 2024, 2024],
    'skenario': ['Optimistis', 'Moderat', 'Pesimistis',
'Optimistis', 'Moderat', 'Pesimistis', 'Optimistis', 'Moderat',
'Pesimistis', 'Optimistis', 'Moderat', 'Pesimistis',
'Optimistis', 'Moderat', 'Pesimistis'],
    'pendapatan': [1000000000, 900000000, 800000000,
1200000000, 1000000000, 850000000, 1500000000, 1300000000,
1100000000, 1800000000, 1600000000, 1400000000, 2000000000,
1800000000, 1600000000],
    'beban_operasional': [600000000, 650000000, 700000000,
700000000, 720000000, 740000000, 800000000, 820000000,
850000000, 900000000, 920000000, 950000000, 1000000000,
1020000000, 1050000000],
    'penyusutan': [50000000, 45000000, 40000000, 60000000,
55000000, 50000000, 70000000, 65000000, 60000000, 80000000,
75000000, 70000000, 90000000, 85000000, 80000000]
}
df_transaksi = pd.DataFrame(data_transaksi_keuangan)

data_kebijakan_fiskal = {
    'tahun': [2020, 2021, 2022, 2023, 2024],
    'tax_rate': [0.22, 0.22, 0.22, 0.22, 0.22], # Example tax
rate
    'tax_holiday_awal': [2018, 2021, 2018, 2021, 2024], #
Example tax holiday start years
    'tax_holiday_akhir': [2020, 2023, 2020, 2023, 2026] #
Example tax holiday end years
}
df_kebijakan = pd.DataFrame(data_kebijakan_fiskal)


# Equivalent of the LabaKotor CTE
df_laba_kotor = df_transaksi.groupby(['tahun',
'skenario']).agg(
```

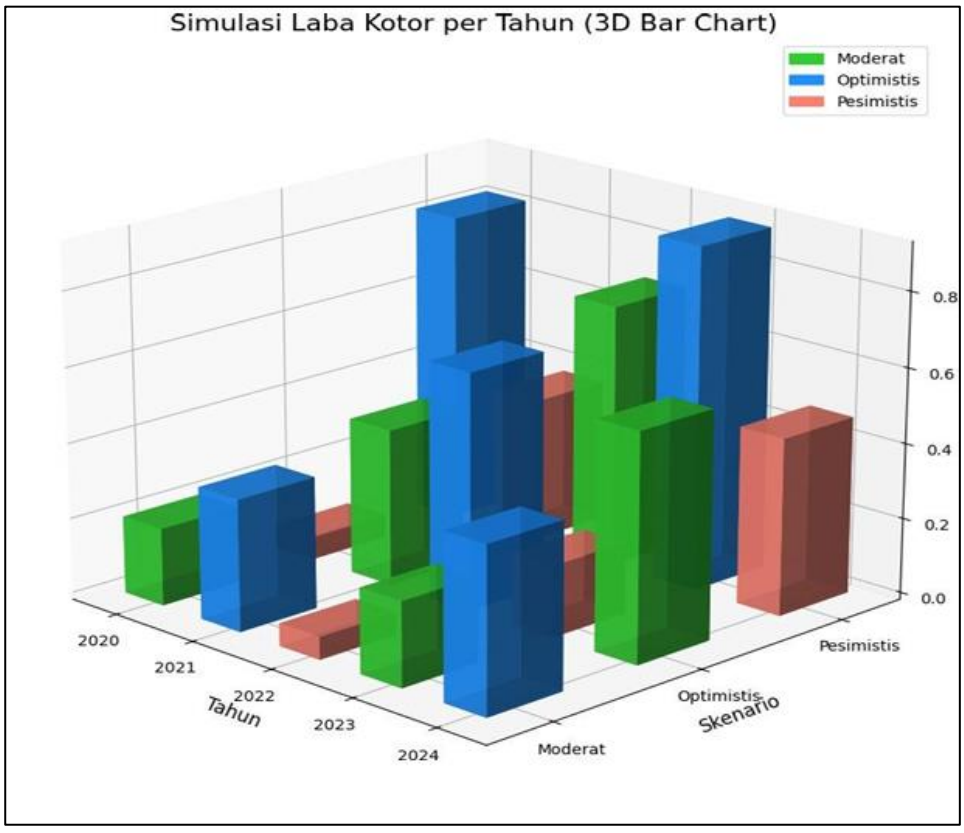
```
    laba_kena_pajak=('pendapatan', lambda x: (x.sum() -
df_transaksi.loc[x.index, 'beban_operasional'].sum() -
df_transaksi.loc[x.index, 'penyusutan'].sum()))
).reset_index()


# Equivalent of the JOIN with kebijakan_fiskal
df_result = pd.merge(df_laba_kotor, df_kebijakan, on='tahun',
how='left')

# Equivalent of the PPh calculation in the final SELECT
df_result['pph_badan'] = df_result.apply(
    lambda row: 0 if row['tahun'] >= row['tax_holiday_awal']
and row['tahun'] <= row['tax_holiday_akhir'] else
row['laba_kena_pajak'] * row['tax_rate'], axis=1
)

# Equivalent of the arus_kas_setelah_pajak calculation
df_result['arus_kas_setelah_pajak'] =
df_result['laba_kena_pajak'] - df_result['pph_badan']

# Select and order columns
df_final = df_result[['tahun', 'skenario', 'laba_kena_pajak',
'pph_badan', 'arus_kas_setelah_pajak']]


# Assuming 'df_final' contains the 'tahun', 'skenario', and
'arus_kas_setelah_pajak' data

if df_final is not None and not df_final.empty:
    fig = plt.figure(figsize=(14, 10))
    ax = fig.add_subplot(111, projection='3d')

    # Get the years and scenarios from the DataFrame
    years = df_final['tahun'].unique()
    scenarios = df_final['skenario'].unique()

    # Create positions for the bars on the x (year) and y
(skenario) axes
    xpos, ypos = np.meshgrid(np.arange(len(years)),
np.arange(len(scenarios)))
    xpos = xpos.flatten()
    ypos = ypos.flatten()
    zpos = np.zeros(len(xpos))

    # Width and depth of the bars
    dx = 0.5 * np.ones_like(xpos)
    dy = 0.5 * np.ones_like(ypos)
```

```
    # Heights of the bars (arus kas setelah pajak)
    # Need to ensure the order of dz matches the order of xpos
and ypos
    # Group by year and scenario to get the correct order and
values
    df_grouped = df_final.groupby(['tahun',
'skenario'])['arus_kas_setelah_pajak'].sum().reset_index() #
Group by arus_kas_setelah_pajak
    dz = df_grouped['arus_kas_setelah_pajak'].values / 1e6 #
Height of bars (in Millions)

    # Mapping colors for each scenario (using more vibrant
colors)
    colors = {'Optimistis': '#1F77B4', 'Moderat': '#2CA02C',
'Pesimistis': '#D62728'} # More vibrant colors
    # Create a list of colors corresponding to each bar based
on the scenario in df_grouped
    scenario_colors = [colors[skenario] for skenario in
df_grouped['skenario']]


    # Create 3D bar chart with enhanced aesthetics
    ax.bar3d(xpos, ypos, zpos, dx, dy, dz,
color=scenario_colors, alpha=0.9, edgecolors='black') #
Increased alpha and added edgecolors

    # Set axis labels
    ax.set_xlabel('Tahun', fontsize=12)
    ax.set_ylabel('Skenario', fontsize=12)
    ax.set_zlabel('Arus Kas Setelah Pajak (Juta Rupiah)',
fontsize=12) # Changed label to Arus Kas Setelah Pajak

    # Set ticks and labels for axes
    x_tick_positions = np.arange(len(years)) + dx[0]/2
    ax.set_xticks(x_tick_positions)
    ax.set_xticklabels(years)

    y_tick_positions = np.arange(len(scenarios)) + dy[0]/2
    ax.set_yticks(y_tick_positions)
    ax.set_yticklabels(scenarios)

    # Title
    ax.set_title('Tren Arus Kas Setelah Pajak per Tahun (3D Bar
Chart)', fontsize=16) # Changed title

    # Add manual legend
    import matplotlib.patches as mpatches
    legend_patches = [mpatches.Patch(color=colors[skenario],
label=skenario) for skenario in scenarios]
```

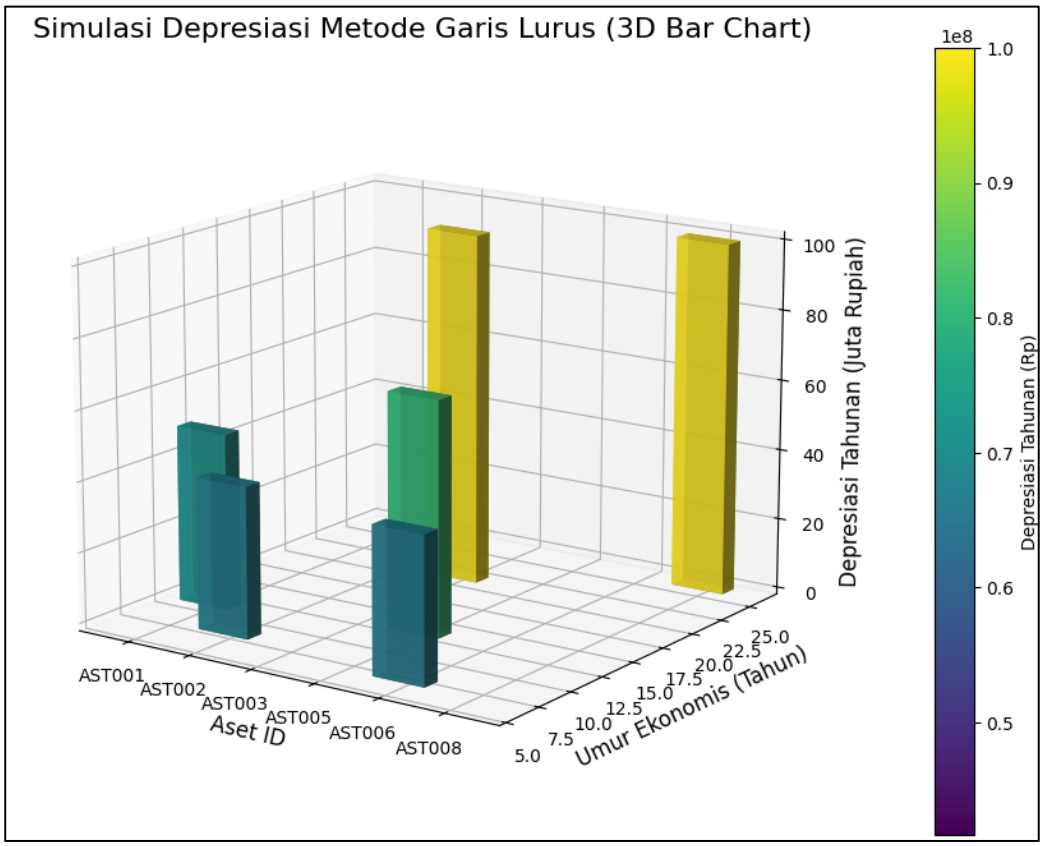
```
    ax.legend(handles=legend_patches)

    # Adjust view angle for better perspective
    ax.view_init(elev=25, azim=-55) # Adjusted view angle

    # Add grid for better readability
    ax.grid(True, linestyle='--', alpha=0.6)

    plt.show()
else:
    print("DataFrame kosong atau tidak tersedia untuk
visualisasi.")
```

4. **Visualisasi Simulasi PPh Badan Google Colab**
   Berikut adalah visualisasi simulasi PPh Badan menggunakan Google Colab dengan kode Phyton

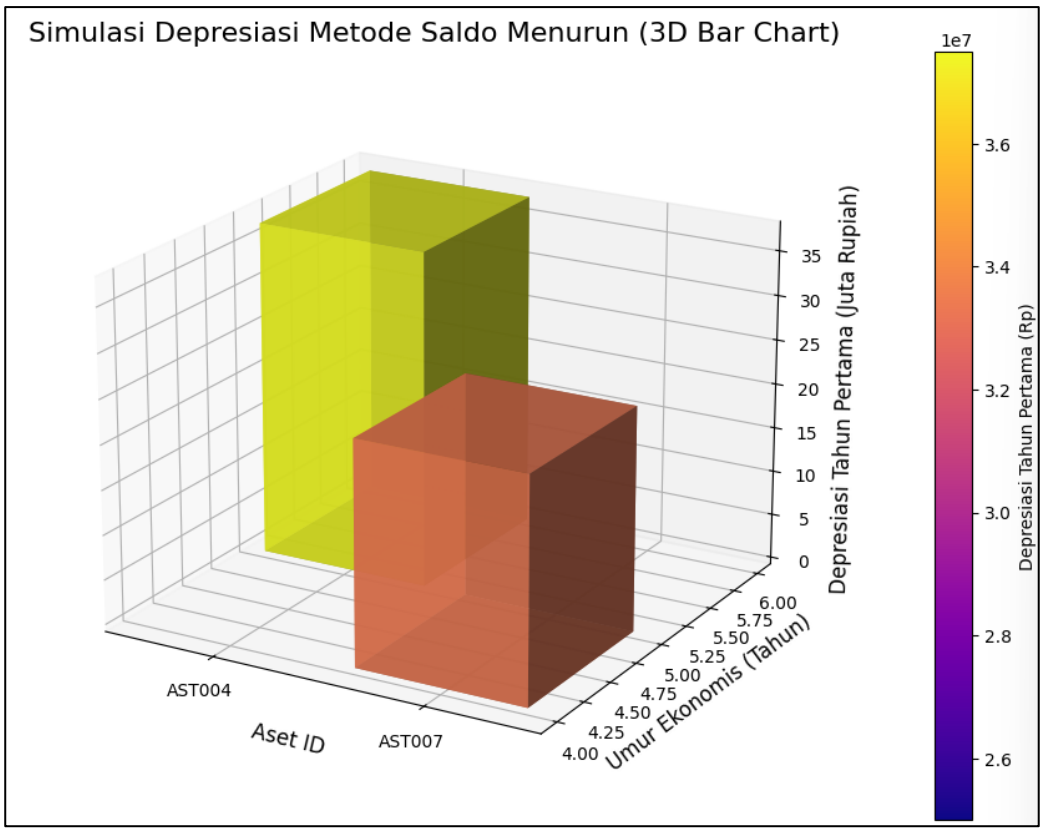
   - **Simulasi Laba Rugi (Phyton)**



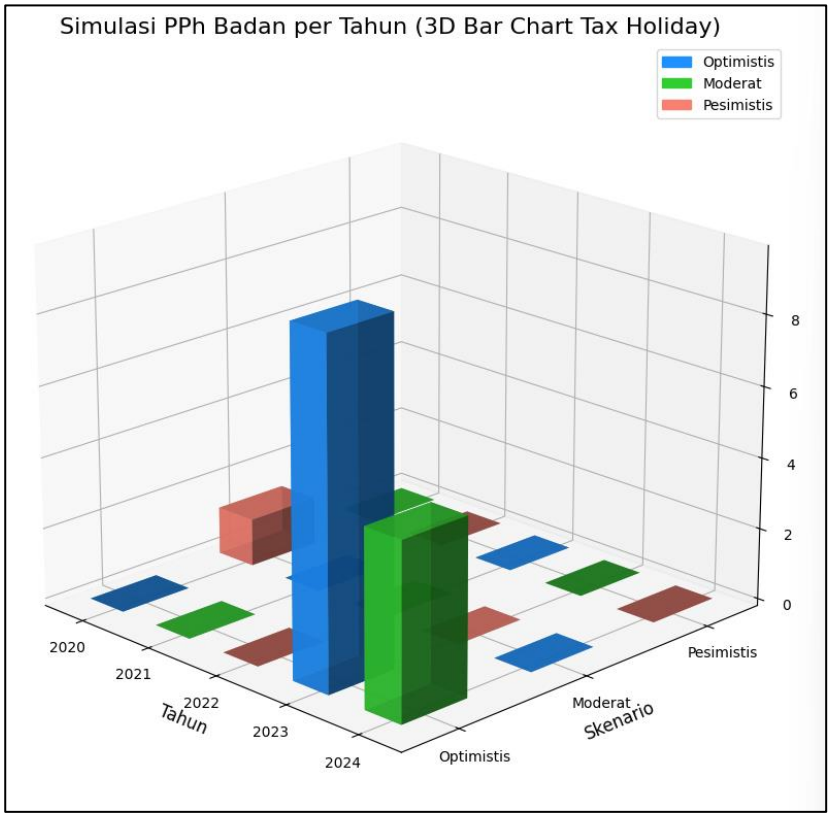
- **Simulasi Depresiasi Metode Garis Lurus (Phyton)**



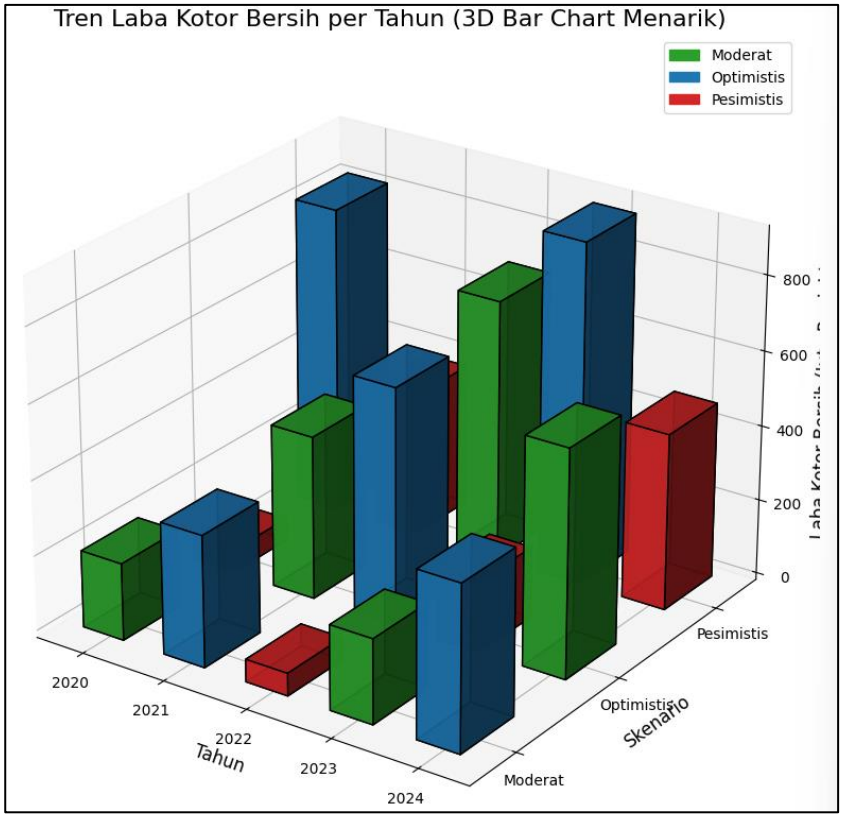
- **Simulasi Depresiasi Metode Saldo Menurun (Phyton)**



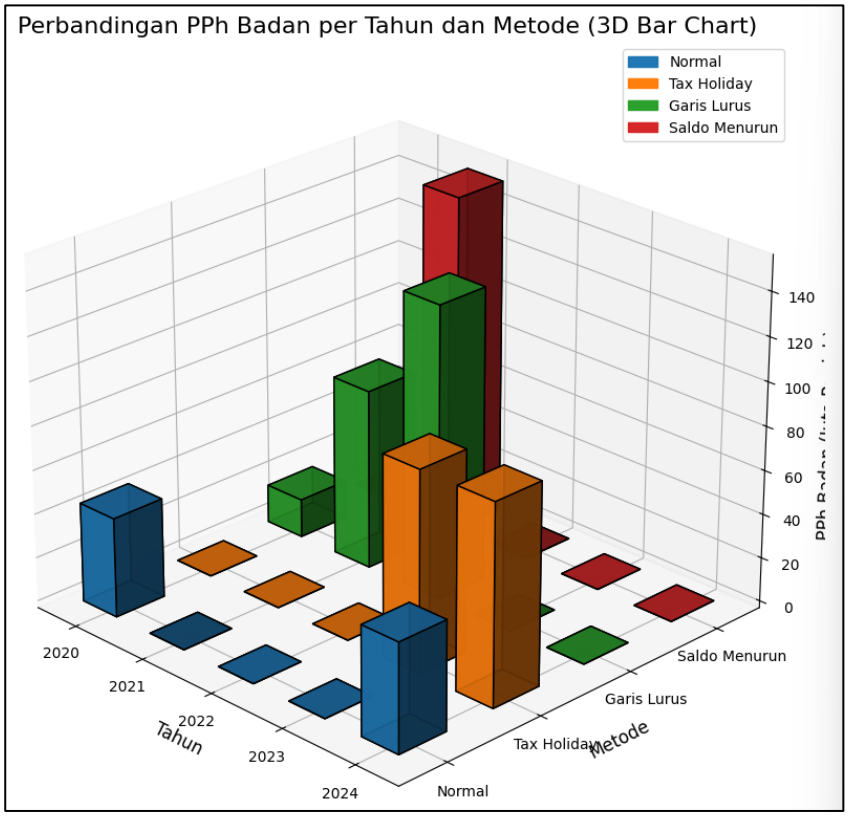
- **Simulasi Tax Holiday (Phyton)**



- **Simulasi Tren Laba Rugi Bersih (Phyton)**



- **Simulasi Perbandingan PPh Badan per Tahun (Phyton)**



- **Tren Arus Kas Setelah Pajak per Tahun (Phyton)**