In [29]:
```
import requests
import sys
import os
sys.path.append(os.path.abspath(os.path.join(os.getcwd(), '..', '..')))
from credential import FMP_API_KEY
from highlight_text import ax_text,fig_text
import pandas as pd
import matplotlib.pyplot as plt

import morethemes as mt
```

In [3]:
```
url = "https://financialmodelingprep.com/api/v3/income-statement/AAPL"
params = {
    "period": "annual",
    "limit": 10,
    "apikey": FMP_API_KEY
}

r = requests.get(url, params=params, timeout=30)
data = r.json()
print(data)
```

[{'date': '2024-09-28', 'symbol': 'AAPL', 'reportedCurrency': 'USD', 'cik': '0000320
193', 'fillingDate': '2024-11-01', 'acceptedDate': '2024-11-01 06:01:36', 'calendarY
ear': '2024', 'period': 'FY', 'revenue': 391035000000, 'costOfRevenue': 21035200000
0, 'grossProfit': 180683000000, 'grossProfitRatio': 0.4620634982, 'researchAndDevelo
pmentExpenses': 31370000000, 'generalAndAdministrativeExpenses': 0, 'sellingAndMarke
tingExpenses': 0, 'sellingGeneralAndAdministrativeExpenses': 26097000000, 'otherExpe
nses': 0, 'operatingExpenses': 57467000000, 'costAndExpenses': 267819000000, 'intere
stIncome': 0, 'interestExpense': 0, 'depreciationAndAmortization': 11445000000, 'ebi
tda': 134661000000, 'ebitdaratio': 0.3443707085, 'operatingIncome': 123216000000, 'o
peratingIncomeRatio': 0.3151022287, 'totalOtherIncomeExpensesNet': 269000000, 'incom
eBeforeTax': 123485000000, 'incomeBeforeTaxRatio': 0.3157901467, 'incomeTaxExpense':
29749000000, 'netIncome': 93736000000, 'netIncomeRatio': 0.2397125577, 'eps': 6.11,
'epsdiluted': 6.08, 'weightedAverageShsOut': 15343783000, 'weightedAverageShsOutDi
l': 15408095000, 'link': 'https://www.sec.gov/Archives/edgar/data/320193/00003201932
4000123/0000320193-24-000123-index.htm', 'finalLink': 'https://www.sec.gov/Archives/
edgar/data/320193/000032019324000123/aapl-20240928.htm'}, {'date': '2023-09-30', 'sy
mbol': 'AAPL', 'reportedCurrency': 'USD', 'cik': '0000320193', 'fillingDate': '2023-
11-03', 'acceptedDate': '2023-11-02 18:08:27', 'calendarYear': '2023', 'period': 'F
Y', 'revenue': 383285000000, 'costOfRevenue': 214137000000, 'grossProfit': 169148000
000, 'grossProfitRatio': 0.4413112958, 'researchAndDevelopmentExpenses': 2991500000
0, 'generalAndAdministrativeExpenses': 0, 'sellingAndMarketingExpenses': 0, 'selling
GeneralAndAdministrativeExpenses': 24932000000, 'otherExpenses': 0, 'operatingExpens
es': 54847000000, 'costAndExpenses': 268984000000, 'interestIncome': 3750000000, 'in
terestExpense': 3933000000, 'depreciationAndAmortization': 11519000000, 'ebitda': 12
5820000000, 'ebitdaratio': 0.3282674772, 'operatingIncome': 114301000000, 'operating
IncomeRatio': 0.2982141227, 'totalOtherIncomeExpensesNet': -565000000, 'incomeBefore
Tax': 113736000000, 'incomeBeforeTaxRatio': 0.2967400237, 'incomeTaxExpense': 167410
00000, 'netIncome': 96995000000, 'netIncomeRatio': 0.2530623426, 'eps': 6.16, 'epsdi
luted': 6.13, 'weightedAverageShsOut': 15744231000, 'weightedAverageShsOutDil': 1581
2547000, 'link': 'https://www.sec.gov/Archives/edgar/data/320193/000032019323000106/
0000320193-23-000106-index.htm', 'finalLink': 'https://www.sec.gov/Archives/edgar/da
ta/320193/000032019323000106/aapl-20230930.htm'}, {'date': '2022-09-24', 'symbol':
'AAPL', 'reportedCurrency': 'USD', 'cik': '0000320193', 'fillingDate': '2022-10-28',
'acceptedDate': '2022-10-27 18:01:14', 'calendarYear': '2022', 'period': 'FY', 'reve
nue': 394328000000, 'costOfRevenue': 223546000000, 'grossProfit': 170782000000, 'gro
ssProfitRatio': 0.4330963056, 'researchAndDevelopmentExpenses': 26251000000, 'genera
lAndAdministrativeExpenses': 0, 'sellingAndMarketingExpenses': 0, 'sellingGeneralAnd
AdministrativeExpenses': 25094000000, 'otherExpenses': 0, 'operatingExpenses': 51345
000000, 'costAndExpenses': 274891000000, 'interestIncome': 2825000000, 'interestExpe
nse': 2931000000, 'depreciationAndAmortization': 11104000000, 'ebitda': 13313800000
0, 'ebitdaratio': 0.3376326307, 'operatingIncome': 119437000000, 'operatingIncomeRat
io': 0.302887444, 'totalOtherIncomeExpensesNet': -334000000, 'incomeBeforeTax': 1191
03000000, 'incomeBeforeTaxRatio': 0.3020404333, 'incomeTaxExpense': 19300000000, 'ne
tIncome': 99803000000, 'netIncomeRatio': 0.2530964071, 'eps': 6.15, 'epsdiluted': 6.
11, 'weightedAverageShsOut': 16215963000, 'weightedAverageShsOutDil': 16325819000,
'link': 'https://www.sec.gov/Archives/edgar/data/320193/000032019322000108/000032019
3-22-000108-index.htm', 'finalLink': 'https://www.sec.gov/Archives/edgar/data/32019
3/000032019322000108/aapl-20220924.htm'}, {'date': '2021-09-25', 'symbol': 'AAPL',
'reportedCurrency': 'USD', 'cik': '0000320193', 'fillingDate': '2021-10-29', 'accept
edDate': '2021-10-28 18:04:28', 'calendarYear': '2021', 'period': 'FY', 'revenue': 3
65817000000, 'costOfRevenue': 212981000000, 'grossProfit': 152836000000, 'grossProfi
tRatio': 0.4177935963, 'researchAndDevelopmentExpenses': 21914000000, 'generalAndAdm
inistrativeExpenses': 0, 'sellingAndMarketingExpenses': 0, 'sellingGeneralAndAdminis
trativeExpenses': 21973000000, 'otherExpenses': 0, 'operatingExpenses': 43887000000,
'costAndExpenses': 256868000000, 'interestIncome': 2843000000, 'interestExpense': 26
45000000, 'depreciationAndAmortization': 11284000000, 'ebitda': 123136000000, 'ebitd

aratio': 0.3366054612, 'operatingIncome': 108949000000, 'operatingIncomeRatio': 0.29
78237753, 'totalOtherIncomeExpensesNet': 258000000, 'incomeBeforeTax': 109207000000,
'incomeBeforeTaxRatio': 0.2985290459, 'incomeTaxExpense': 14527000000, 'netIncome':
94680000000, 'netIncomeRatio': 0.2588179336, 'eps': 5.67, 'epsdiluted': 5.61, 'weigh
tedAverageShsOut': 16701272000, 'weightedAverageShsOutDil': 16864919000, 'link': 'ht
tps://www.sec.gov/Archives/edgar/data/320193/000032019321000105/0000320193-21-000105
-index.htm', 'finalLink': 'https://www.sec.gov/Archives/edgar/data/320193/0000320193
21000105/aapl-20210925.htm'}, {'date': '2020-09-26', 'symbol': 'AAPL', 'reportedCurr
ency': 'USD', 'cik': '0000320193', 'fillingDate': '2020-10-30', 'acceptedDate': '202
0-10-29 18:06:25', 'calendarYear': '2020', 'period': 'FY', 'revenue': 274515000000,
'costOfRevenue': 169559000000, 'grossProfit': 104956000000, 'grossProfitRatio': 0.38
23324773, 'researchAndDevelopmentExpenses': 18752000000, 'generalAndAdministrativeEx
penses': 0, 'sellingAndMarketingExpenses': 0, 'sellingGeneralAndAdministrativeExpens
es': 19916000000, 'otherExpenses': 0, 'operatingExpenses': 38668000000, 'costAndExpe
nses': 208227000000, 'interestIncome': 3763000000, 'interestExpense': 2873000000, 'd
epreciationAndAmortization': 11056000000, 'ebitda': 81020000000, 'ebitdaratio': 0.29
51386992, 'operatingIncome': 66288000000, 'operatingIncomeRatio': 0.2414731435, 'tot
alOtherIncomeExpensesNet': 803000000, 'incomeBeforeTax': 67091000000, 'incomeBeforeT
axRatio': 0.2443983025, 'incomeTaxExpense': 9680000000, 'netIncome': 57411000000, 'n
etIncomeRatio': 0.2091361128, 'eps': 3.31, 'epsdiluted': 3.28, 'weightedAverageShsOu
t': 17352119000, 'weightedAverageShsOutDil': 17528214000, 'link': 'https://www.sec.g
ov/Archives/edgar/data/320193/000032019320000096/0000320193-20-000096-index.htm', 'f
inalLink': 'https://www.sec.gov/Archives/edgar/data/320193/000032019320000096/aapl-2
0200926.htm'}]

In [4]:
```
df = pd.DataFrame(data)
```

In [5]:
```
df
```

Out[5]:

	date	symbol	reportedCurrency	cik	fillingDate	acceptedDate	calendarYear
0	2024-09-28	AAPL	USD	0000320193	2024-11-01	2024-11-01 06:01:36	2024
1	2023-09-30	AAPL	USD	0000320193	2023-11-03	2023-11-02 18:08:27	2023
2	2022-09-24	AAPL	USD	0000320193	2022-10-28	2022-10-27 18:01:14	2022
3	2021-09-25	AAPL	USD	0000320193	2021-10-29	2021-10-28 18:04:28	2021
4	2020-09-26	AAPL	USD	0000320193	2020-10-30	2020-10-29 18:06:25	2020

5 rows × 38 columns

Data cleaning

- ☑ remove unncessary columns: `fillingDate`, `acceptedDate`, `period`, `date`, `symbol`, `reportedCurrency`, `cik`, `link`, `fina` - done
- ☑ display value in as proper format - done

- ☑ tranform the data from width to height - done

In [6]:
```
df.drop(columns=['fillingDate', 'acceptedDate', 'period', 'date', 'symbol', 'report
```

In [7]:
```
df
```

Out[7]:

	calendarYear	revenue	costOfRevenue	grossProfit	grossProfitRatio	researchAn
0	2024	391035000000	210352000000	180683000000	0.462063	
1	2023	383285000000	214137000000	169148000000	0.441311	
2	2022	394328000000	223546000000	170782000000	0.433096	
3	2021	365817000000	212981000000	152836000000	0.417794	
4	2020	274515000000	169559000000	104956000000	0.382332	

5 rows × 29 columns

In [8]:
```
df_format = df.melt(
    id_vars=["calendarYear"],         # keep the year fixed
    var_name="Metric",                # new column for metric names
    value_name="Value"                # new column for metric values
)
```

In [9]:
```
df_format
```

Out[9]:

	calendarYear	Metric	Value
0	2024	revenue	3.910350e+11
1	2023	revenue	3.832850e+11
2	2022	revenue	3.943280e+11
3	2021	revenue	3.658170e+11
4	2020	revenue	2.745150e+11
...	...	...	...
135	2024	weightedAverageShsOutDil	1.540810e+10
136	2023	weightedAverageShsOutDil	1.581255e+10
137	2022	weightedAverageShsOutDil	1.632582e+10
138	2021	weightedAverageShsOutDil	1.686492e+10
139	2020	weightedAverageShsOutDil	1.752821e+10

140 rows × 3 columns

In [10]:
```
df_revenue = df_format[df_format['Metric'] == 'revenue']
df_revenue
```

Out[10]:

	calendarYear	Metric	Value
0	2024	revenue	3.910350e+11
1	2023	revenue	3.832850e+11
2	2022	revenue	3.943280e+11
3	2021	revenue	3.658170e+11
4	2020	revenue	2.745150e+11

In [11]:
```
df_revenue['value_billion'] = df_revenue['Value']/100_000_000
df_revenue['value_billion'] = df_revenue['value_billion'].map("{:,.2f}".format)
df_revenue
```

```
C:\Users\dqthi\AppData\Local\Temp\ipykernel_3576\4090961083.py:1: SettingWithCopyWar
ning:
A value is trying to be set on a copy of a slice from a DataFrame.
Try using .loc[row_indexer,col_indexer] = value instead

See the caveats in the documentation: https://pandas.pydata.org/pandas-docs/stable/u
ser_guide/indexing.html#returning-a-view-versus-a-copy
  df_revenue['value_billion'] = df_revenue['Value']/100_000_000
C:\Users\dqthi\AppData\Local\Temp\ipykernel_3576\4090961083.py:2: SettingWithCopyWar
ning:
A value is trying to be set on a copy of a slice from a DataFrame.
Try using .loc[row_indexer,col_indexer] = value instead

See the caveats in the documentation: https://pandas.pydata.org/pandas-docs/stable/u
ser_guide/indexing.html#returning-a-view-versus-a-copy
  df_revenue['value_billion'] = df_revenue['value_billion'].map("{:,.2f}".format)
```

Out[11]:

	calendarYear	Metric	Value	value_billion
0	2024	revenue	3.910350e+11	3,910.35
1	2023	revenue	3.832850e+11	3,832.85
2	2022	revenue	3.943280e+11	3,943.28
3	2021	revenue	3.658170e+11	3,658.17
4	2020	revenue	2.745150e+11	2,745.15

In [12]:
```
df_revenue["calendarYear"] = (
    df_revenue["calendarYear"]
    .astype(str)                # make sure everything is string
    .str.strip()                # remove leading/trailing spaces
    .str.replace(r"\.0$", "", regex=True)  # drop '.0' if present
```

```
    .astype("Int32")                # finally cast to int
)
```

```
C:\Users\dqthi\AppData\Local\Temp\ipykernel_3576\1750495670.py:1: SettingWithCopyWar
ning:
A value is trying to be set on a copy of a slice from a DataFrame.
Try using .loc[row_indexer,col_indexer] = value instead

See the caveats in the documentation: https://pandas.pydata.org/pandas-docs/stable/u
ser_guide/indexing.html#returning-a-view-versus-a-copy
  df_revenue["calendarYear"] = (
```

In [13]:
```
print(df_revenue['calendarYear'].unique())
```

```
<IntegerArray>
[2024, 2023, 2022, 2021, 2020]
Length: 5, dtype: Int32
```

In [14]:
```
type(df_revenue.loc[0, "calendarYear"]), df_revenue["calendarYear"].unique()
```

Out[14]:
```
(numpy.int32,
 <IntegerArray>
 [2024, 2023, 2022, 2021, 2020]
 Length: 5, dtype: Int32)
```

In [15]:
```
df_revenue["value_billion"] = (
    df_revenue["value_billion"].astype(str).str.replace(",", "", regex=False)
    .pipe(pd.to_numeric, errors="coerce")
)
```

```
C:\Users\dqthi\AppData\Local\Temp\ipykernel_3576\3641463991.py:1: SettingWithCopyWar
ning:
A value is trying to be set on a copy of a slice from a DataFrame.
Try using .loc[row_indexer,col_indexer] = value instead

See the caveats in the documentation: https://pandas.pydata.org/pandas-docs/stable/u
ser_guide/indexing.html#returning-a-view-versus-a-copy
  df_revenue["value_billion"] = (
```

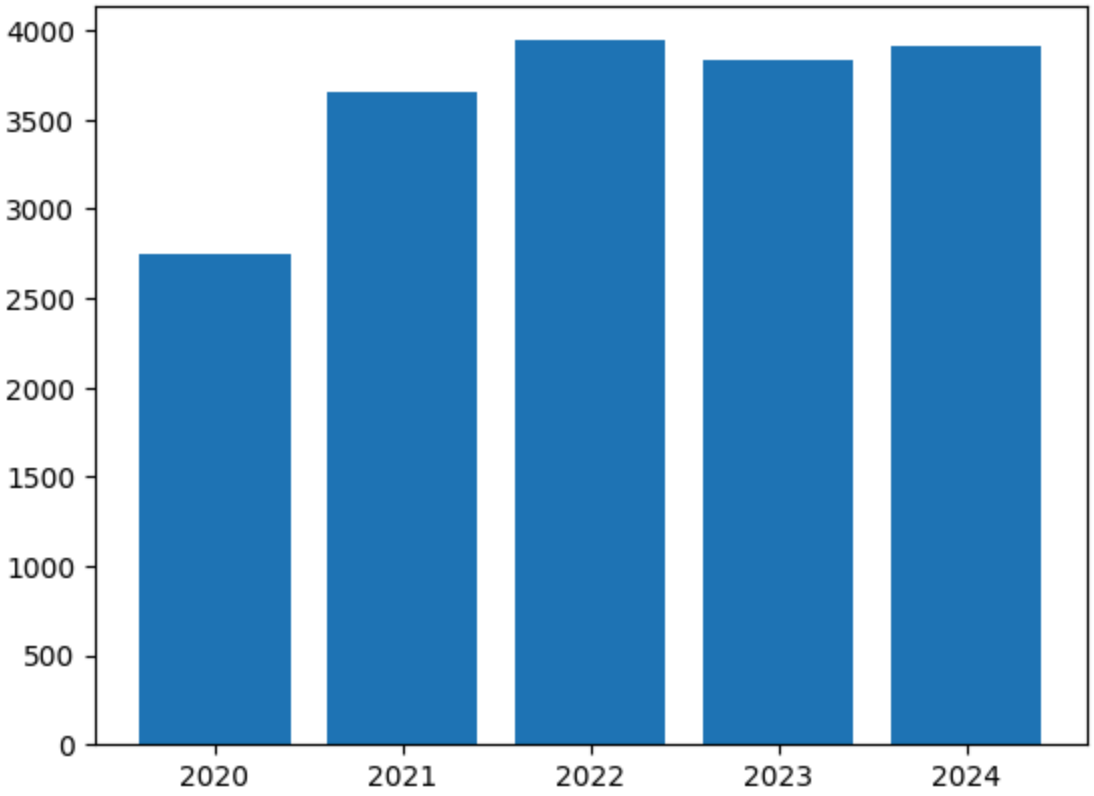
In [16]:
```
df_revenue.info()
```

```
<class 'pandas.core.frame.DataFrame'>
Index: 5 entries, 0 to 4
Data columns (total 4 columns):
 #   Column         Non-Null Count  Dtype
---  ------         --------------  -----
 0   calendarYear   5 non-null      Int32
 1   Metric         5 non-null      object
 2   Value          5 non-null      float64
 3   value_billion  5 non-null      float64
dtypes: Int32(1), float64(2), object(1)
memory usage: 357.0+ bytes
```

To follow up:

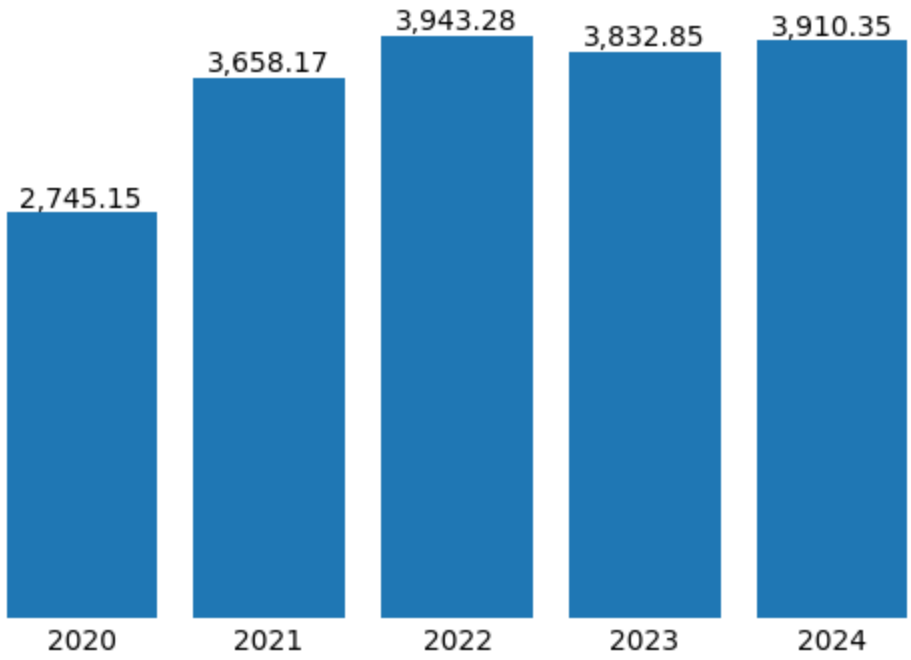
- try writing a function that appy value formatting (e.i add comma between thousand) to the select list of columns

In [17]:
```
# step 1, populate the draft chart to visualize

fig, ax = plt.subplots()
x = df_revenue['calendarYear']
y = df_revenue['value_billion']
ax.bar(
    x,y
)
ax = ax

plt.show()
```



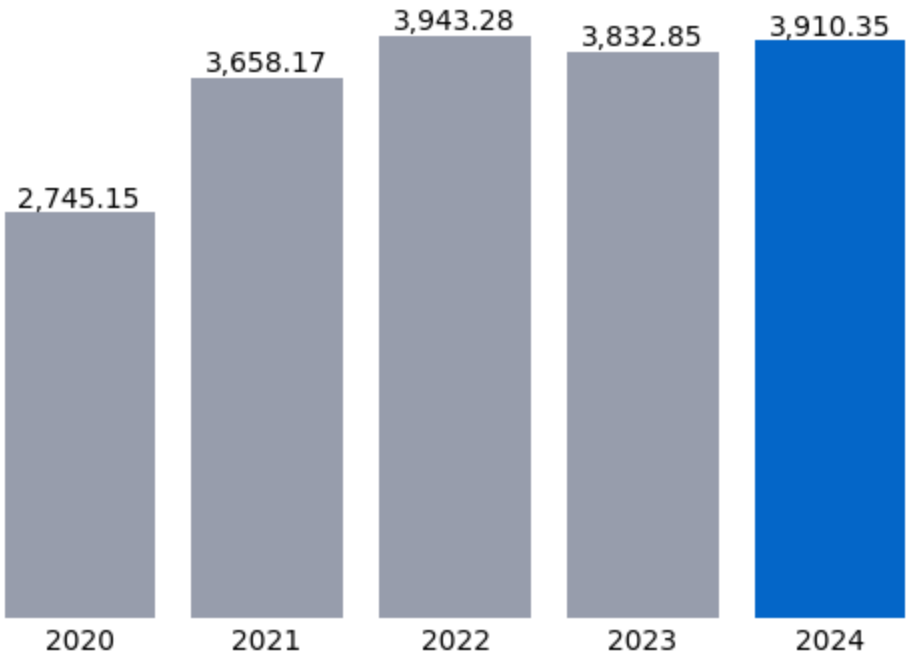
In [18]:
```
# step 2:
# remove the border line
# remove y value
# starting number of y value to be 0, while the max value is 5000

fig, ax = plt.subplots()
x = df_revenue['calendarYear']
y = df_revenue['value_billion']
bar_container = ax.bar(x,y)
for spine in ['top','right','bottom','left']:
    ax.spines[spine].set_visible(False),
# ax.set_xticks([])
ax.set_yticks([]) # to remove border line,
ax.tick_params(length=0), # to remove x - ticks,
plt.ylim(0,5000),  # starting number of y value to be 0, while the max value is 500
```

```
ax.bar_label(bar_container, label_type='edge', fmt=lambda x: f'{x:,.2f}')
ax = bar_container

plt.show()
```



In [19]:
```
# step 2:
# remove the border line
# remove y value
# starting number of y value to be 0, while the max value is 5000

# step 3:
# apply formating
# setting color for 2024, prior year to grey

blue = '#0466c8'
grey = '#979dac'
colors = [
    blue if year == 2024 else grey
    for year in df_revenue["calendarYear"]
]

fig, ax = plt.subplots()
x = df_revenue['calendarYear']
y = df_revenue['value_billion']
bar_container = ax.bar(x,y,color = colors)
for spine in ['top','right','bottom','left']:
    ax.spines[spine].set_visible(False),
ax.set_yticks([]) # to remove border line,
ax.tick_params(length=0), # to remove x - ticks,
plt.ylim(0,5000),  # starting number of y value to be 0, while the max value is 500
ax.bar_label(bar_container, label_type='edge', fmt=lambda x: f'{x:,.2f}')
```

```
ax = bar_container



plt.show()
```



In [20]:
```
def cagr(begin_value, end_value, periods):
    return (end_value / begin_value) ** (1/periods) - 1
```

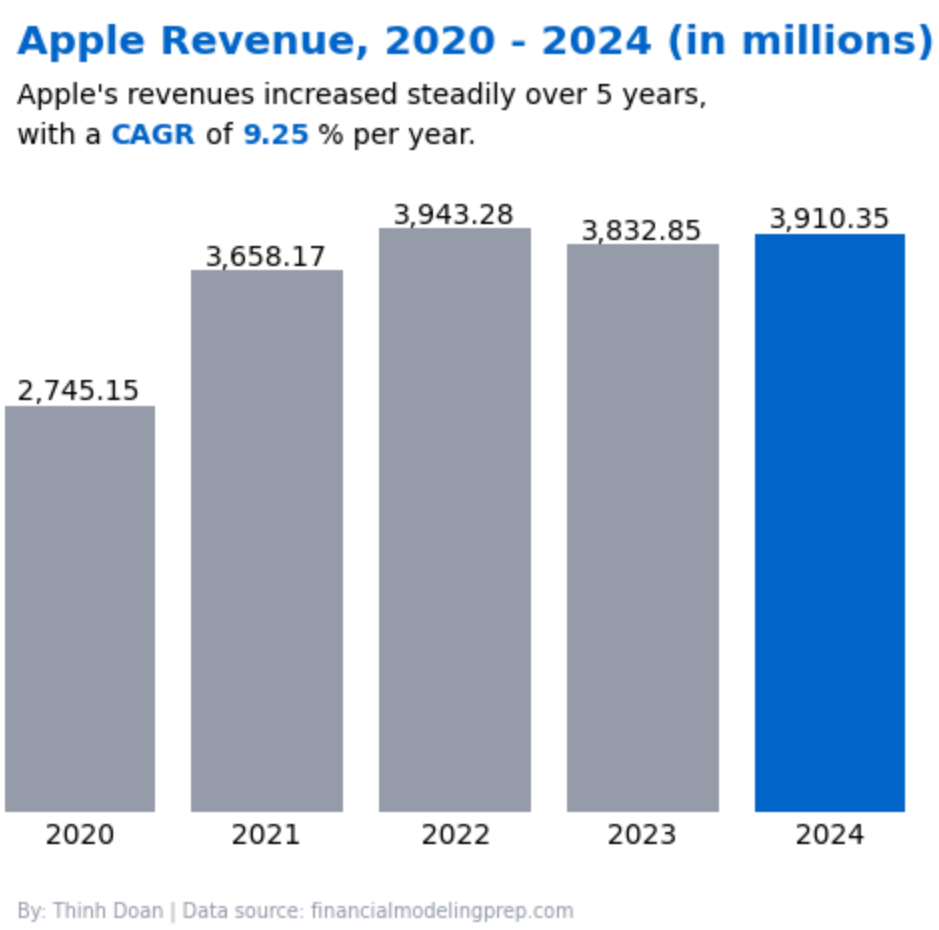
In [21]:
```
start_year = df_revenue['calendarYear'].min()
end_year = df_revenue['calendarYear'].max()

begin_value = df_revenue.loc[df_revenue['calendarYear'] == start_year,'value_billio
end_value = df_revenue.loc[df_revenue['calendarYear'] == end_year,'value_billion'].
periods = end_year - start_year
```

In [22]:
```
apple_cagr = cagr(begin_value, end_value, periods )
apple_cagr
```

Out[22]:
```
np.float64(0.09247721446491397)
```

In [34]:
```
# step 2:
## remove the border line
## remove y value
## starting number of y value to be 0, while the max value is 5000

# step 3:
## apply formating
## setting color for 2024, prior year to grey
```

```
# step 4:
## add chart title
## add chart sub title

blue = '#0466c8'
grey = '#979dac'
colors = [
    blue if year == 2024 else grey
    for year in df_revenue["calendarYear"]
]

fig, ax = plt.subplots()

x = df_revenue['calendarYear']
y = df_revenue['value_billion']
bar_container = ax.bar(x,y,color = colors)
for spine in ['top','right','bottom','left']:
    ax.spines[spine].set_visible(False),
ax.set_yticks([]) # to remove border line,
ax.tick_params(length=0), # to remove x - ticks,
plt.ylim(0,5000),  # starting number of y value to be 0, while the max value is 500
ax.bar_label(bar_container, label_type='edge', fmt=lambda x: f'{x:,.2f}')
ax = bar_container

# add title
fig.text(x = .17, y = .9 ,s='Apple Revenue, 2020 - 2024 (in millions)',color=blue,f
# add sub title
fig_text(x = .17, y = .87 ,s=f"Apple's revenues increased steadily over 5 years,\nw
# add credit
fig.text(x = .17, y = 0, s = 'By: Thinh Doan | Data source: financialmodelingprep.c
plt.show()
```



Data Visualization

What are the steps:

- Draft visual
  - Chart title: Apple 5 Year Revenue Performance (2020 - 2025)
  - Chart subtitle: Apple revenues show steady increase over 5 year with xx% a year
  - Chart credit: viz by: ThinhD, data source: FMP,
    - ☑ to calculate for CARG ratio
  - Chart type - Chart color - Color pallets
    - ☑ use https://y-sunflower.github.io/morethemes/#__tabbed_1_6
    - => Decided to not use theme as I want to make more control of the chart

Make chart showing cartoon style - https://matplotlib.org/stable/gallery/showcase/xkcd.html#sphx-glr-gallery-showcase-xkcd-py