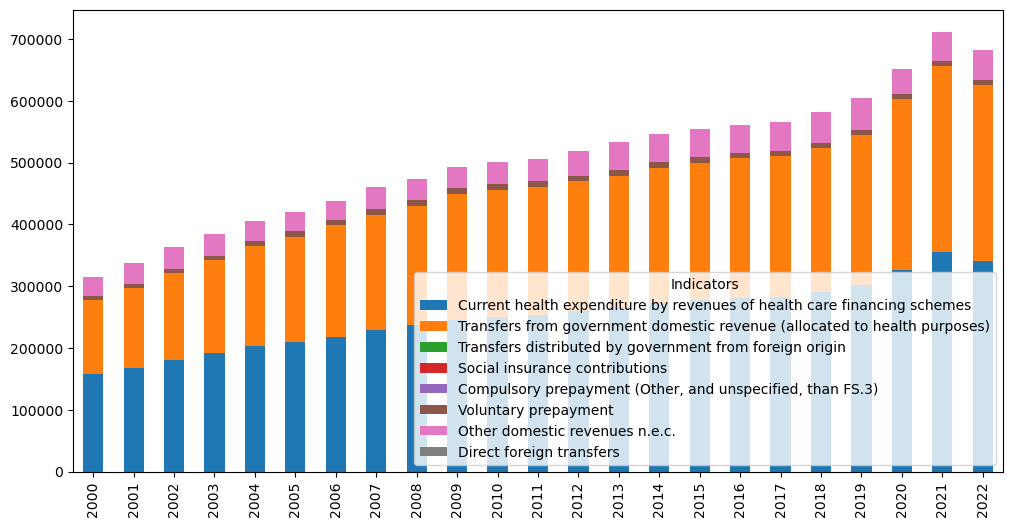
## Current Health Expenditure (CHE) In million constant (2022) US$

%%pyspark
  
  
import pandas as pd
  
import matplotlib.pyplot as plt
  
  
df = spark.sql("SELECT \* FROM `default`.`nhaindicators`")
  
  
#df = spark.read.load('abfss://healthdata-fs@distributedanalytics.dfs.core.windows.net/data/Current Health Expenditure (CHE) as % Gross Domestic Product (GDP).csv', format='csv'
  
## If header exists uncomment line below
  
#, header=True
  
#)
  
  
# select un-necessary columns to remove
  
cols\_to\_drop = [df.columns[0], df.columns[2]]
  
  
# Drop the specified columns
  
df = df.drop(\*cols\_to\_drop)
  
df.show(truncate=False)
  
  
  
# convery spark df to pandas
  
pandas\_df = df.toPandas()
  
  
# as it is loaded from csv,Identify the columns that need to be converted to numeric
  
columns\_to\_convert = list(pandas\_df.columns)[1:]
  
  
# Convert the specified columns to numeric
  
for col in columns\_to\_convert:
  
 pandas\_df[col] = pd.to\_numeric(pandas\_df[col], errors='coerce')
  
  
df\_transposed = pandas\_df.set\_index('Indicators').T
  
  
df\_transposed.plot(kind='bar', stacked=True, figsize=(12, 6))
  
  
plt.show()

{"execution\_finish\_time":"2025-03-20T23:42:17.7311443Z","execution\_start\_time":"2025-03-20T23:41:08.6014715Z","livy\_statement\_state":"available","normalized\_state":"finished","parent\_msg\_id":"9107ce3b-4746-4232-a0cd-d8dfb2f22407","queued\_time":"2025-03-20T23:36:00.4323694Z","session\_id":"47","session\_start\_time":"2025-03-20T23:36:00.4337018Z","spark\_jobs":null,"spark\_pool":"ghparkcluster","state":"finished","statement\_id":2,"statement\_ids":[2]}

+-------------------------------------------------------------------------+------+------+------+------+------+------+------+------+------+------+------+------+------+------+------+------+------+------+------+------+------+------+------+  
|Indicators |2000 |2001 |2002 |2003 |2004 |2005 |2006 |2007 |2008 |2009 |2010 |2011 |2012 |2013 |2014 |2015 |2016 |2017 |2018 |2019 |2020 |2021 |2022 |  
+-------------------------------------------------------------------------+------+------+------+------+------+------+------+------+------+------+------+------+------+------+------+------+------+------+------+------+------+------+------+  
|Current health expenditure by revenues of health care financing schemes |157882|168633|181476|191934|202928|209876|218748|230046|236950|246200|250221|253247|259598|266565|273011|277190|280482|283055|291161|302490|325950|355811|341370|  
|Transfers from government domestic revenue (allocated to health purposes)|119672|127825|139253|150057|162517|170492|180301|185324|192641|203749|205454|207397|210333|212109|218863|222834|227162|227998|232698|241813|277140|299994|283563|  
|Transfers distributed by government from foreign origin |0 |0 |0 |0 |0 |0 |0 |0 |0 |0 |0 |0 |0 |0 |0 |0 |0 |0 |0 |0 |0 |0 |0 |  
|Social insurance contributions |0 |0 |0 |0 |0 |0 |0 |0 |0 |0 |0 |0 |0 |0 |0 |0 |0 |0 |0 |0 |0 |0 |0 |  
|Compulsory prepayment (Other, and unspecified, than FS.3) |0 |0 |0 |0 |0 |0 |0 |0 |0 |0 |0 |0 |0 |0 |0 |0 |0 |0 |0 |0 |0 |0 |0 |  
|Voluntary prepayment |6775 |7531 |7736 |7716 |7788 |8380 |8663 |8926 |9515 |9104 |8972 |9068 |9157 |9176 |9338 |9121 |7922 |8507 |8575 |8667 |7817 |7983 |8565 |  
|Other domestic revenues n.e.c. |31423 |33265 |34472 |34145 |32607 |30987 |29765 |35778 |34775 |33329 |35774 |36760 |40085 |45253 |44784 |45206 |45367 |46514 |49854 |51976 |40962 |47801 |49207 |  
|Direct foreign transfers |12 |13 |15 |16 |16 |17 |18 |18 |19 |18 |20 |22 |24 |26 |26 |29 |32 |36 |35 |34 |31 |33 |34 |  
+-------------------------------------------------------------------------+------+------+------+------+------+------+------+------+------+------+------+------+------+------+------+------+------+------+------+------+------+------+------+

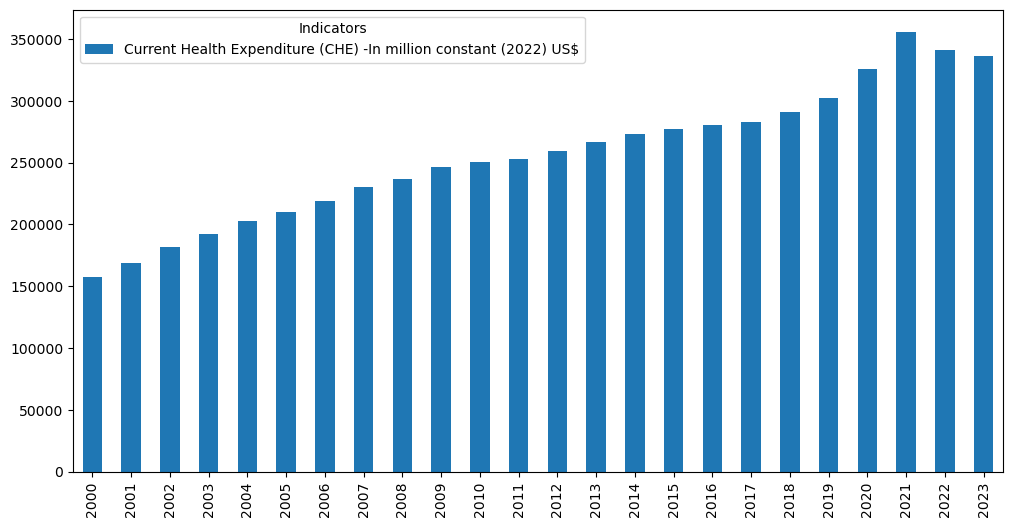


## Current Health Expenditure (CHE) -In million constant (2022) US$ ( Standarised based on USD 2022 to accomodate Inflation)- revenue from health care financilin schemes

%%pyspark
  
  
import pandas as pd
  
import matplotlib.pyplot as plt
  
  
#df = spark.read.load('abfss://healthdata-fs@distributedanalytics.dfs.core.windows.net/data/CHE\_In million constant\_2022\_USD.csv', format='csv'
  
## If header exists uncomment line below
  
#, header=True
  
#)
  
df = spark.sql("SELECT \* FROM `default`.`cheinmillionconstant2022usd`")
  
  
df.show(truncate=False)
  
  
# convery spark df to pandas
  
pandas\_df = df.toPandas()
  
  
# as it is loaded from csv,Identify the columns that need to be converted to numeric
  
columns\_to\_convert = list(pandas\_df.columns)[1:]
  
  
# Convert the specified columns to numeric
  
for col in columns\_to\_convert:
  
 pandas\_df[col] = pd.to\_numeric(pandas\_df[col], errors='coerce')
  
  
df\_transposed = pandas\_df.set\_index('Indicators').T
  
  
df\_transposed.plot(kind='bar', stacked=True, figsize=(12, 6))
  
  
plt.show()

{"execution\_finish\_time":"2025-03-20T23:42:20.6740873Z","execution\_start\_time":"2025-03-20T23:42:17.7473014Z","livy\_statement\_state":"available","normalized\_state":"finished","parent\_msg\_id":"c095b9a8-64cb-41a6-8bd9-248457a60015","queued\_time":"2025-03-20T23:36:00.4331591Z","session\_id":"47","session\_start\_time":null,"spark\_jobs":null,"spark\_pool":"ghparkcluster","state":"finished","statement\_id":3,"statement\_ids":[3]}

+----------------------------------------------------------------+------+------+------+------+------+------+------+------+------+------+------+------+------+------+------+------+------+------+------+------+------+------+------+------+  
|Indicators |2000 |2001 |2002 |2003 |2004 |2005 |2006 |2007 |2008 |2009 |2010 |2011 |2012 |2013 |2014 |2015 |2016 |2017 |2018 |2019 |2020 |2021 |2022 |2023 |  
+----------------------------------------------------------------+------+------+------+------+------+------+------+------+------+------+------+------+------+------+------+------+------+------+------+------+------+------+------+------+  
|Current Health Expenditure (CHE) -In million constant (2022) US$|157882|168633|181476|191934|202928|209876|218748|230046|236950|246200|250221|253247|259598|266565|273011|277190|280482|283055|291161|302490|325950|355811|341370|336329|  
+----------------------------------------------------------------+------+------+------+------+------+------+------+------+------+------+------+------+------+------+------+------+------+------+------+------+------+------+------+------+

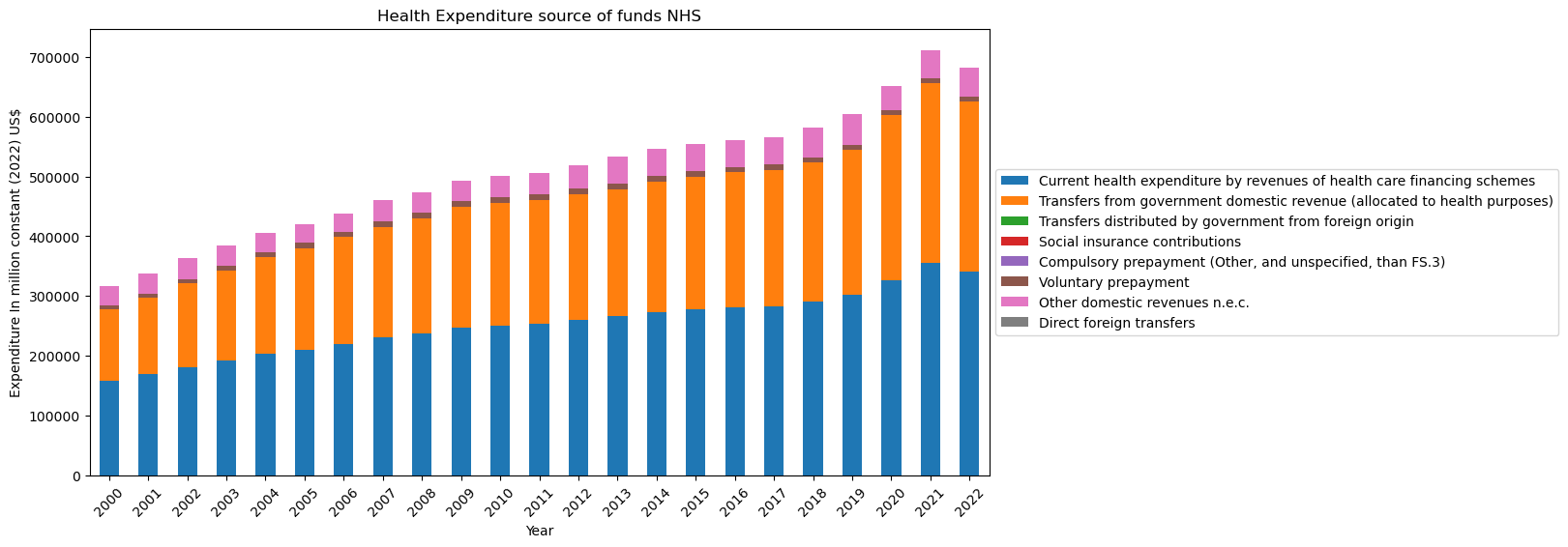


EDA NHS FUNDING SOURCES

%%pyspark
  
from pyspark.sql.functions import monotonically\_increasing\_id,col, struct, when, isnan, cast
  
import pandas as pd
  
import matplotlib.pyplot as plt
  
  
  
df = spark.sql("SELECT \* FROM `default`.`nhaindicators`")
  
  
# select un-necessary columns to remove
  
cols\_to\_drop = [df.columns[0], df.columns[2]]
  
  
# Drop the specified columns
  
df = df.drop(\*cols\_to\_drop)
  
df.show(truncate=False)
  
  
# convery spark df to pandas
  
pandas\_df = df.toPandas()
  
  
# as it is loaded from csv,Identify the columns that need to be converted to numeric
  
columns\_to\_convert = list(pandas\_df.columns)[1:]
  
  
  
# Convert the specified columns to numeric
  
for col in columns\_to\_convert:
  
 pandas\_df[col] = pd.to\_numeric(pandas\_df[col], errors='coerce')
  
  
df\_transposed = pandas\_df.set\_index('Indicators').T
  
  
df\_transposed.plot(kind='bar', stacked=True, figsize=(12, 6))
  
  
# Customize plot
  
plt.title('Health Expenditure source of funds NHS')
  
plt.xlabel('Year')
  
plt.ylabel('Expenditure In million constant (2022) US$')
  
plt.xticks(rotation=45)
  
plt.legend(loc='center left', bbox\_to\_anchor=(1, 0.5))
  
  
plt.show()

{"execution\_finish\_time":"2025-03-20T23:42:23.6195175Z","execution\_start\_time":"2025-03-20T23:42:20.6888512Z","livy\_statement\_state":"available","normalized\_state":"finished","parent\_msg\_id":"2f2d0acf-24b5-49aa-8759-35b0b191943e","queued\_time":"2025-03-20T23:36:00.4338537Z","session\_id":"47","session\_start\_time":null,"spark\_jobs":null,"spark\_pool":"ghparkcluster","state":"finished","statement\_id":4,"statement\_ids":[4]}

+-------------------------------------------------------------------------+------+------+------+------+------+------+------+------+------+------+------+------+------+------+------+------+------+------+------+------+------+------+------+  
|Indicators |2000 |2001 |2002 |2003 |2004 |2005 |2006 |2007 |2008 |2009 |2010 |2011 |2012 |2013 |2014 |2015 |2016 |2017 |2018 |2019 |2020 |2021 |2022 |  
+-------------------------------------------------------------------------+------+------+------+------+------+------+------+------+------+------+------+------+------+------+------+------+------+------+------+------+------+------+------+  
|Current health expenditure by revenues of health care financing schemes |157882|168633|181476|191934|202928|209876|218748|230046|236950|246200|250221|253247|259598|266565|273011|277190|280482|283055|291161|302490|325950|355811|341370|  
|Transfers from government domestic revenue (allocated to health purposes)|119672|127825|139253|150057|162517|170492|180301|185324|192641|203749|205454|207397|210333|212109|218863|222834|227162|227998|232698|241813|277140|299994|283563|  
|Transfers distributed by government from foreign origin |0 |0 |0 |0 |0 |0 |0 |0 |0 |0 |0 |0 |0 |0 |0 |0 |0 |0 |0 |0 |0 |0 |0 |  
|Social insurance contributions |0 |0 |0 |0 |0 |0 |0 |0 |0 |0 |0 |0 |0 |0 |0 |0 |0 |0 |0 |0 |0 |0 |0 |  
|Compulsory prepayment (Other, and unspecified, than FS.3) |0 |0 |0 |0 |0 |0 |0 |0 |0 |0 |0 |0 |0 |0 |0 |0 |0 |0 |0 |0 |0 |0 |0 |  
|Voluntary prepayment |6775 |7531 |7736 |7716 |7788 |8380 |8663 |8926 |9515 |9104 |8972 |9068 |9157 |9176 |9338 |9121 |7922 |8507 |8575 |8667 |7817 |7983 |8565 |  
|Other domestic revenues n.e.c. |31423 |33265 |34472 |34145 |32607 |30987 |29765 |35778 |34775 |33329 |35774 |36760 |40085 |45253 |44784 |45206 |45367 |46514 |49854 |51976 |40962 |47801 |49207 |  
|Direct foreign transfers |12 |13 |15 |16 |16 |17 |18 |18 |19 |18 |20 |22 |24 |26 |26 |29 |32 |36 |35 |34 |31 |33 |34 |  
+-------------------------------------------------------------------------+------+------+------+------+------+------+------+------+------+------+------+------+------+------+------+------+------+------+------+------+------+------+------+

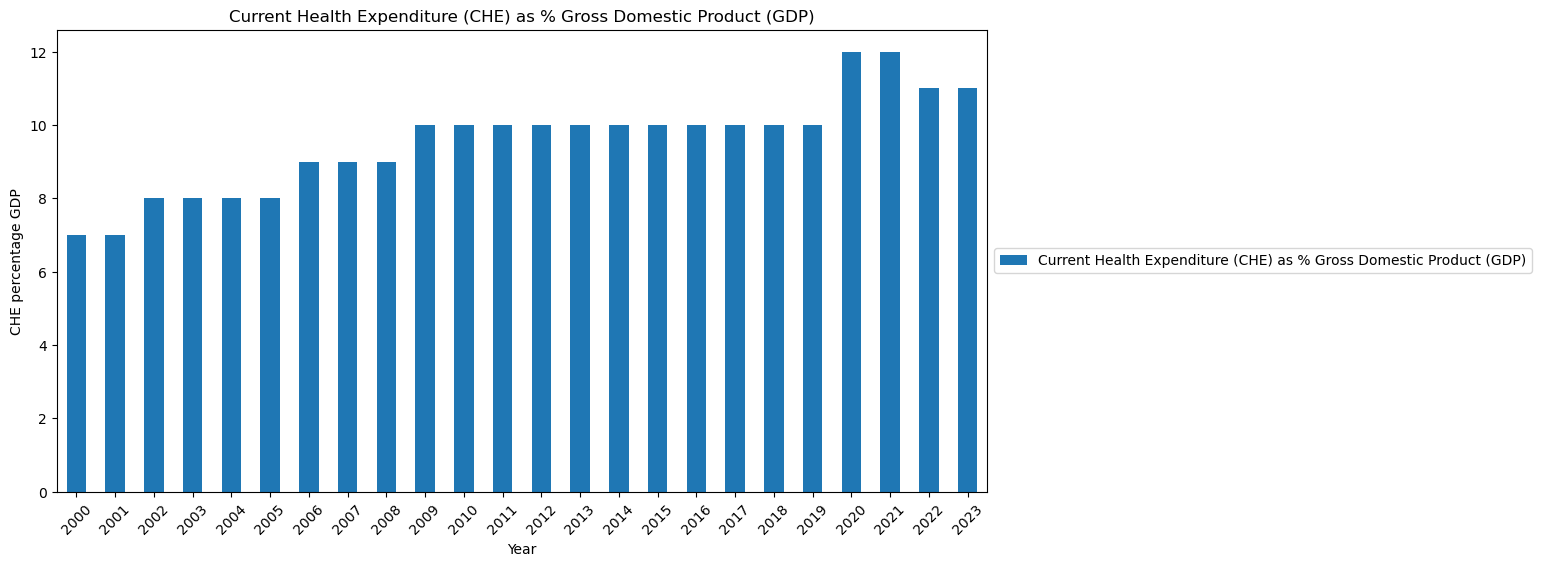


## CHE - Current Health Expenses as percentage of GDP

%%pyspark
  
  
import pandas as pd
  
import matplotlib.pyplot as plt
  
  
df = spark.sql("SELECT \* FROM `default`.`cheaspercentgdp`")
  
df.show(10)
  
  
# convery spark df to pandas
  
pandas\_df = df.toPandas()
  
  
# as it is loaded from csv,Identify the columns that need to be converted to numeric
  
columns\_to\_convert = list(pandas\_df.columns)[1:]
  
  
  
# Convert the specified columns to numeric
  
for col in columns\_to\_convert:
  
 pandas\_df[col] = pd.to\_numeric(pandas\_df[col], errors='coerce')
  
  
df\_transposed = pandas\_df.set\_index('Indicators').T
  
  
df\_transposed.plot(kind='bar', stacked=True, figsize=(12, 6))
  
  
# Customize plot
  
plt.title('Current Health Expenditure (CHE) as % Gross Domestic Product (GDP)')
  
plt.xlabel('Year')
  
plt.ylabel('CHE percentage GDP')
  
plt.xticks(rotation=45)
  
plt.legend(loc='center left', bbox\_to\_anchor=(1, 0.5))
  
  
plt.show()

{"execution\_finish\_time":"2025-03-20T23:42:25.6316862Z","execution\_start\_time":"2025-03-20T23:42:23.6358914Z","livy\_statement\_state":"available","normalized\_state":"finished","parent\_msg\_id":"7df689b5-b78a-478e-bf60-a584cc841493","queued\_time":"2025-03-20T23:36:00.4345273Z","session\_id":"47","session\_start\_time":null,"spark\_jobs":null,"spark\_pool":"ghparkcluster","state":"finished","statement\_id":5,"statement\_ids":[5]}

+--------------------+----+----+----+----+----+----+----+----+----+----+----+----+----+----+----+----+----+----+----+----+----+----+----+----+  
| Indicators|2000|2001|2002|2003|2004|2005|2006|2007|2008|2009|2010|2011|2012|2013|2014|2015|2016|2017|2018|2019|2020|2021|2022|2023|  
+--------------------+----+----+----+----+----+----+----+----+----+----+----+----+----+----+----+----+----+----+----+----+----+----+----+----+  
|Current Health Ex...| 7| 7| 8| 8| 8| 8| 9| 9| 9| 10| 10| 10| 10| 10| 10| 10| 10| 10| 10| 10| 12| 12| 11| 11|  
+--------------------+----+----+----+----+----+----+----+----+----+----+----+----+----+----+----+----+----+----+----+----+----+----+----+----+

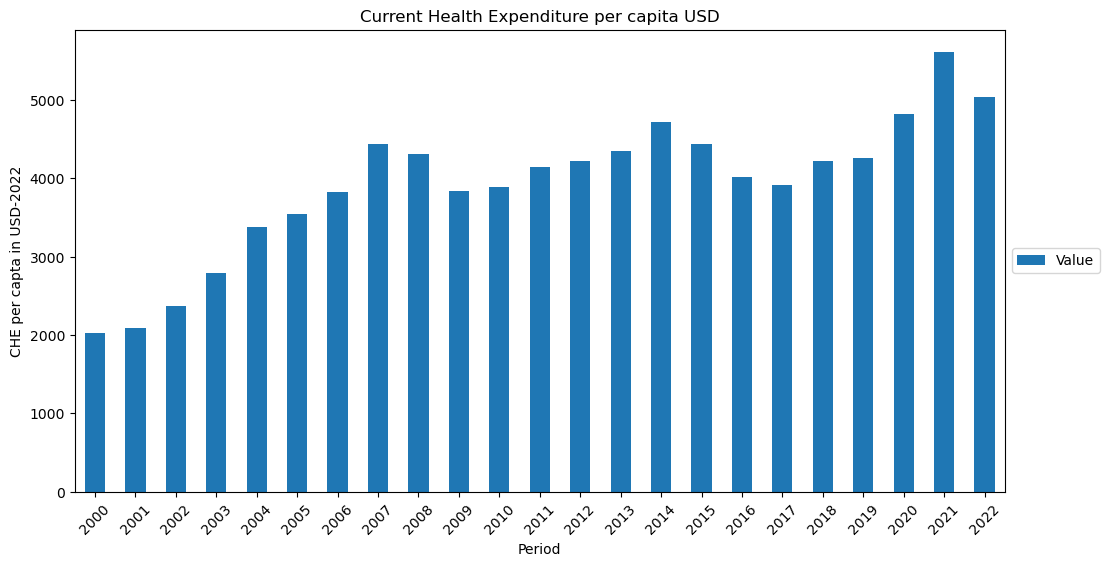


## CHE per capita USD

%%pyspark
  
import pandas as pd
  
import matplotlib.pyplot as plt
  
  
df = spark.sql("SELECT \* FROM `default`.`chepercapitausd`where SpatialDimValueCode='GBR' order by Period ")
  
  
df.show(25)
  
  
  
# select un-necessary columns to remove
  
cols\_to\_drop = [df.columns[0], df.columns[1], df.columns[2],df.columns[5]]
  
  
# Drop the specified columns
  
df = df.drop(\*cols\_to\_drop)
  
  
df.show(25)
  
  
# convery spark df to pandas
  
pandas\_df = df.toPandas()
  
  
# as it is loaded from csv,Identify the columns that need to be converted to numeric
  
columns\_to\_convert = list(pandas\_df.columns)[1:]
  
  
  
# Convert the specified columns to numeric
  
for col in columns\_to\_convert:
  
 pandas\_df[col] = pd.to\_numeric(pandas\_df[col], errors='coerce')
  
  
#df\_transposed = pandas\_df.set\_index('Indicators').T
  
##
  
  
pandas\_df = pandas\_df.set\_index('Period')
  
  
  
pandas\_df.plot(kind='bar', stacked=True, figsize=(12, 6))
  
  
  
# Customize plot
  
plt.title('Current Health Expenditure per capita USD')
  
plt.xlabel('Period')
  
plt.ylabel('CHE per capta in USD-2022')
  
plt.xticks(rotation=45)
  
plt.legend(loc='center left', bbox\_to\_anchor=(1, 0.5))
  
  
plt.show()

{"execution\_finish\_time":"2025-03-20T23:42:33.0441556Z","execution\_start\_time":"2025-03-20T23:42:25.6470152Z","livy\_statement\_state":"available","normalized\_state":"finished","parent\_msg\_id":"ae332b6d-288f-4f98-b06a-988e553dcff3","queued\_time":"2025-03-20T23:36:00.43522Z","session\_id":"47","session\_start\_time":null,"spark\_jobs":null,"spark\_pool":"ghparkcluster","state":"finished","statement\_id":6,"statement\_ids":[6]}

+--------------------+-------------------+--------------------+------+-------+--------------------+  
| Indicator|SpatialDimValueCode| Location|Period| Value| DateModified|  
+--------------------+-------------------+--------------------+------+-------+--------------------+  
|Current health ex...| GBR|United Kingdom of...| 2000|2027.64|2024-12-10T00:00:...|  
|Current health ex...| GBR|United Kingdom of...| 2001|2083.99|2024-12-10T00:00:...|  
|Current health ex...| GBR|United Kingdom of...| 2002|2374.82|2024-12-10T00:00:...|  
|Current health ex...| GBR|United Kingdom of...| 2003|2789.35|2024-12-10T00:00:...|  
|Current health ex...| GBR|United Kingdom of...| 2004|3377.12|2024-12-10T00:00:...|  
|Current health ex...| GBR|United Kingdom of...| 2005|3547.29|2024-12-10T00:00:...|  
|Current health ex...| GBR|United Kingdom of...| 2006|3817.99|2024-12-10T00:00:...|  
|Current health ex...| GBR|United Kingdom of...| 2007|4431.64|2024-12-10T00:00:...|  
|Current health ex...| GBR|United Kingdom of...| 2008|4301.42|2024-12-10T00:00:...|  
|Current health ex...| GBR|United Kingdom of...| 2009|3829.68|2024-12-10T00:00:...|  
|Current health ex...| GBR|United Kingdom of...| 2010|3892.38|2024-12-10T00:00:...|  
|Current health ex...| GBR|United Kingdom of...| 2011|4140.36|2024-12-10T00:00:...|  
|Current health ex...| GBR|United Kingdom of...| 2012|4217.25|2024-12-10T00:00:...|  
|Current health ex...| GBR|United Kingdom of...| 2013|4348.29|2024-12-10T00:00:...|  
|Current health ex...| GBR|United Kingdom of...| 2014| 4716.3|2024-12-10T00:00:...|  
|Current health ex...| GBR|United Kingdom of...| 2015|4440.74|2024-12-10T00:00:...|  
|Current health ex...| GBR|United Kingdom of...| 2016|4016.64|2024-12-10T00:00:...|  
|Current health ex...| GBR|United Kingdom of...| 2017|3907.82|2024-12-10T00:00:...|  
|Current health ex...| GBR|United Kingdom of...| 2018|4219.49|2024-12-10T00:00:...|  
|Current health ex...| GBR|United Kingdom of...| 2019|4258.63|2024-12-10T00:00:...|  
|Current health ex...| GBR|United Kingdom of...| 2020|4816.15|2024-12-10T00:00:...|  
|Current health ex...| GBR|United Kingdom of...| 2021|5612.05|2024-12-10T00:00:...|  
|Current health ex...| GBR|United Kingdom of...| 2022|5035.62|2024-12-10T00:00:...|  
+--------------------+-------------------+--------------------+------+-------+--------------------+  
  
+------+-------+  
|Period| Value|  
+------+-------+  
| 2000|2027.64|  
| 2001|2083.99|  
| 2002|2374.82|  
| 2003|2789.35|  
| 2004|3377.12|  
| 2005|3547.29|  
| 2006|3817.99|  
| 2007|4431.64|  
| 2008|4301.42|  
| 2009|3829.68|  
| 2010|3892.38|  
| 2011|4140.36|  
| 2012|4217.25|  
| 2013|4348.29|  
| 2014| 4716.3|  
| 2015|4440.74|  
| 2016|4016.64|  
| 2017|3907.82|  
| 2018|4219.49|  
| 2019|4258.63|  
| 2020|4816.15|  
| 2021|5612.05|  
| 2022|5035.62|  
+------+-------+

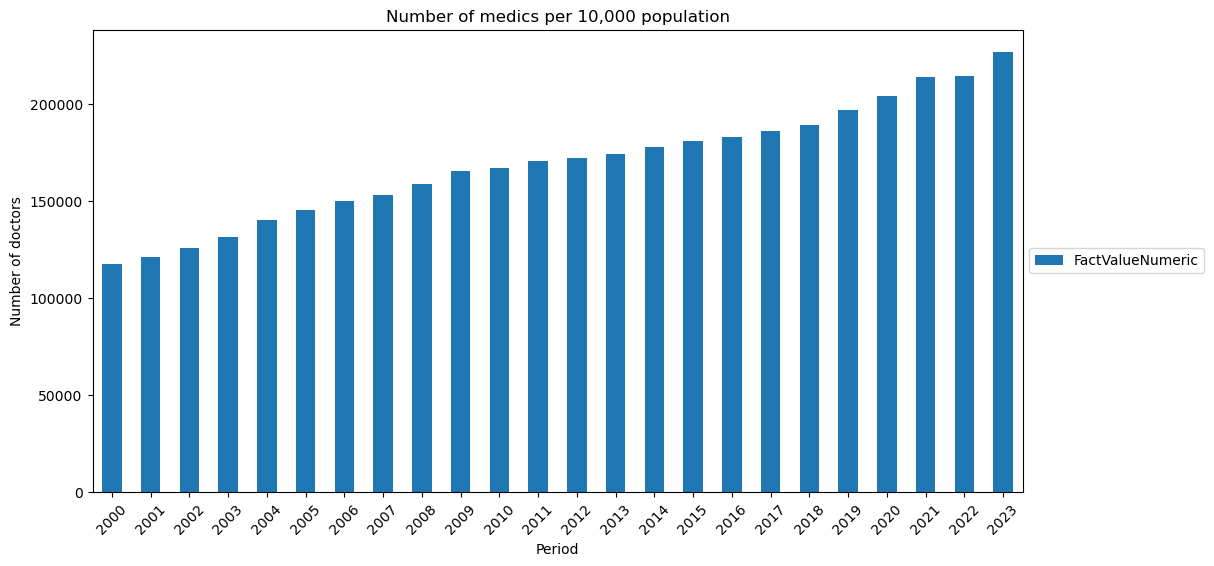


## Doctors in UK - <https://www.who.int/data/gho/data/indicators>

%%pyspark
  
import pandas as pd
  
import matplotlib.pyplot as plt
  
  
df = spark.sql("SELECT \* FROM `default`.`doctorsuk` where SpatialDimValueCode = 'GBR' and IndicatorCode = 'HWF\_0002' and Period > 1999 order by Period")
  
  
  
  
# select un-necessary columns to remove
  
cols\_to\_drop = [df.columns[0], df.columns[1], df.columns[2],df.columns[3],df.columns[4],df.columns[6],df.columns[8],df.columns[9]]
  
  
# Drop the specified columns
  
df = df.drop(\*cols\_to\_drop)
  
  
df.show(100)
  
  
# convery spark df to pandas
  
pandas\_df = df.toPandas()
  
  
# as it is loaded from csv,Identify the columns that need to be converted to numeric
  
columns\_to\_convert = list(pandas\_df.columns)[1:]
  
  
  
# Convert the specified columns to numeric
  
for col in columns\_to\_convert:
  
 pandas\_df[col] = pd.to\_numeric(pandas\_df[col], errors='coerce')
  
  
#df\_transposed = pandas\_df.set\_index('Indicators').T
  
##
  
  
pandas\_df = pandas\_df.set\_index('Period')
  
  
pandas\_df.plot(kind='bar', stacked=True, figsize=(12, 6))
  
  
  
# Customize plot
  
plt.title('Number of medics per 10,000 population')
  
plt.xlabel('Period')
  
plt.ylabel('Number of doctors')
  
plt.xticks(rotation=45)
  
plt.legend(loc='center left', bbox\_to\_anchor=(1, 0.5))
  
  
plt.show()

{"execution\_finish\_time":"2025-03-20T23:42:35.0105421Z","execution\_start\_time":"2025-03-20T23:42:33.0592306Z","livy\_statement\_state":"available","normalized\_state":"finished","parent\_msg\_id":"3f9708d9-d8ed-4ec6-a39a-e5558dafbe54","queued\_time":"2025-03-20T23:36:00.4357949Z","session\_id":"47","session\_start\_time":null,"spark\_jobs":null,"spark\_pool":"ghparkcluster","state":"finished","statement\_id":7,"statement\_ids":[7]}

+------+----------------+  
|Period|FactValueNumeric|  
+------+----------------+  
| 2000| 117332.00|  
| 2001| 120709.00|  
| 2002| 125451.00|  
| 2003| 131272.00|  
| 2004| 139846.00|  
| 2005| 145413.00|  
| 2006| 149558.00|  
| 2007| 152679.00|  
| 2008| 158736.00|  
| 2009| 165109.00|  
| 2010| 166674.00|  
| 2011| 170317.00|  
| 2012| 172069.00|  
| 2013| 173985.00|  
| 2014| 177620.00|  
| 2015| 180888.00|  
| 2016| 182737.00|  
| 2017| 185921.00|  
| 2018| 188783.00|  
| 2019| 196784.00|  
| 2020| 203907.00|  
| 2021| 213839.00|  
| 2022| 214287.00|  
| 2023| 226725.00|  
+------+----------------+

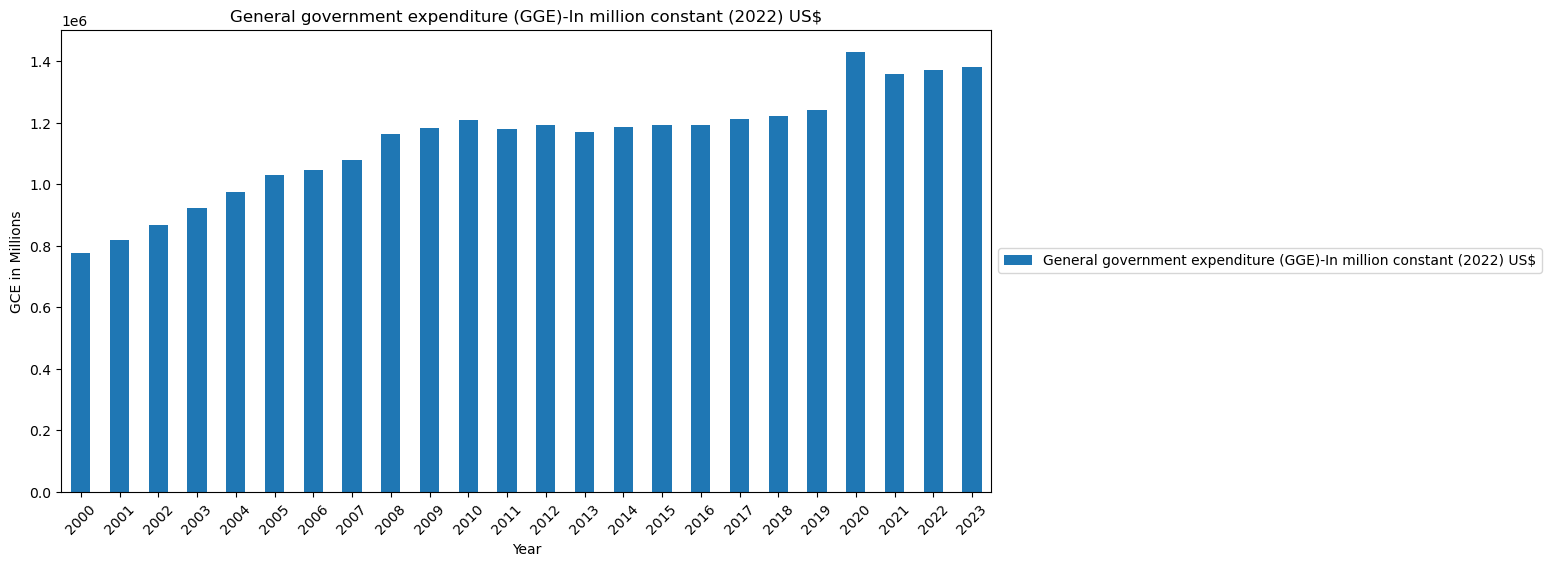


## General government expenditure (GGE)-In million constant (2022) US$

%%pyspark
  
  
import pandas as pd
  
import matplotlib.pyplot as plt
  
  
df = spark.sql("SELECT \* FROM `default`.`gceinmillionconstant2022usd`")
  
# convery spark df to pandas
  
pandas\_df = df.toPandas()
  
  
# as it is loaded from csv,Identify the columns that need to be converted to numeric
  
columns\_to\_convert = list(pandas\_df.columns)[1:]
  
  
  
# Convert the specified columns to numeric
  
for col in columns\_to\_convert:
  
 pandas\_df[col] = pd.to\_numeric(pandas\_df[col], errors='coerce')
  
  
df\_transposed = pandas\_df.set\_index('Indicators').T
  
  
display(df\_transposed)
  
  
df\_transposed.plot(kind='bar', stacked=True, figsize=(12, 6))
  
  
  
# Customize plot
  
plt.title('General government expenditure (GGE)-In million constant (2022) US$')
  
plt.xlabel('Year')
  
plt.ylabel('GCE in Millions')
  
plt.xticks(rotation=45)
  
plt.legend(loc='center left', bbox\_to\_anchor=(1, 0.5))
  
  
plt.show()

{"execution\_finish\_time":"2025-03-20T23:42:36.6344624Z","execution\_start\_time":"2025-03-20T23:42:35.0249215Z","livy\_statement\_state":"available","normalized\_state":"finished","parent\_msg\_id":"d4dd8540-433d-4d46-a4bc-bf03906a5d65","queued\_time":"2025-03-20T23:36:00.436338Z","session\_id":"47","session\_start\_time":null,"spark\_jobs":null,"spark\_pool":"ghparkcluster","state":"finished","statement\_id":8,"statement\_ids":[8]}

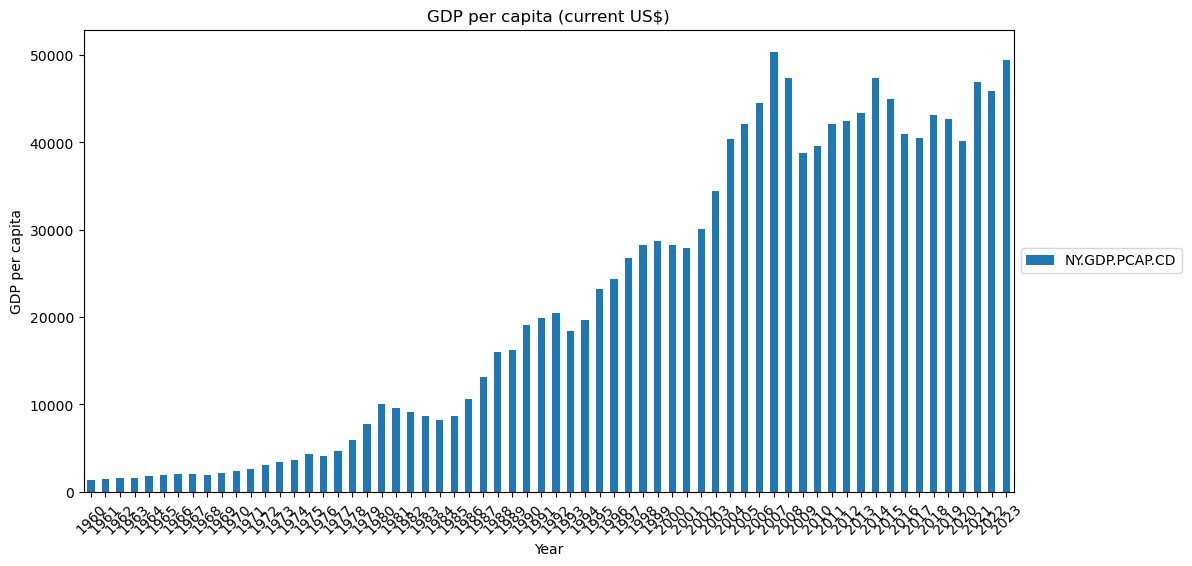
{"widget\_id":"7d3a2dd4-8a0d-4746-b7ba-614afb73106c","widget\_type":"Synapse.DataFrame"}



### Worldbank data - gDP per capita

%%pyspark
  
  
import pandas as pd
  
import matplotlib.pyplot as plt
  
  
df = spark.sql("SELECT \* FROM `default`.`gdppercapitaworldbank` where `Country Code` = 'GBR'")
  
  
# select un-necessary columns to remove
  
cols\_to\_drop = [df.columns[0], df.columns[1], df.columns[2]]
  
  
# Drop the specified columns
  
df = df.drop(\*cols\_to\_drop)
  
  
# convery spark df to pandas
  
pandas\_df = df.toPandas()
  
  
# as it is loaded from csv,Identify the columns that need to be converted to numeric
  
columns\_to\_convert = list(pandas\_df.columns)[1:]
  
  
# Convert the specified columns to numeric
  
for col in columns\_to\_convert:
  
 pandas\_df[col] = pd.to\_numeric(pandas\_df[col], errors='coerce')
  
  
pandas\_df = pandas\_df.rename(columns={'Indicator Code': 'Year'})
  
  
df\_transposed = pandas\_df.set\_index('Year').T
  
  
  
df\_transposed.plot(kind='bar', stacked=True, figsize=(12, 6))
  
  
  
# Customize plot
  
plt.title('GDP per capita (current US$)')
  
plt.xlabel('Year')
  
plt.ylabel('GDP per capita')
  
plt.xticks(rotation=45)
  
plt.legend(loc='center left', bbox\_to\_anchor=(1, 0.5))
  
  
plt.show()

{"execution\_finish\_time":"2025-03-20T23:42:39.5808086Z","execution\_start\_time":"2025-03-20T23:42:36.649377Z","livy\_statement\_state":"available","normalized\_state":"finished","parent\_msg\_id":"c07c2f4c-af88-4231-bda7-75cbfacec357","queued\_time":"2025-03-20T23:36:00.4368938Z","session\_id":"47","session\_start\_time":null,"spark\_jobs":null,"spark\_pool":"ghparkcluster","state":"finished","statement\_id":9,"statement\_ids":[9]}

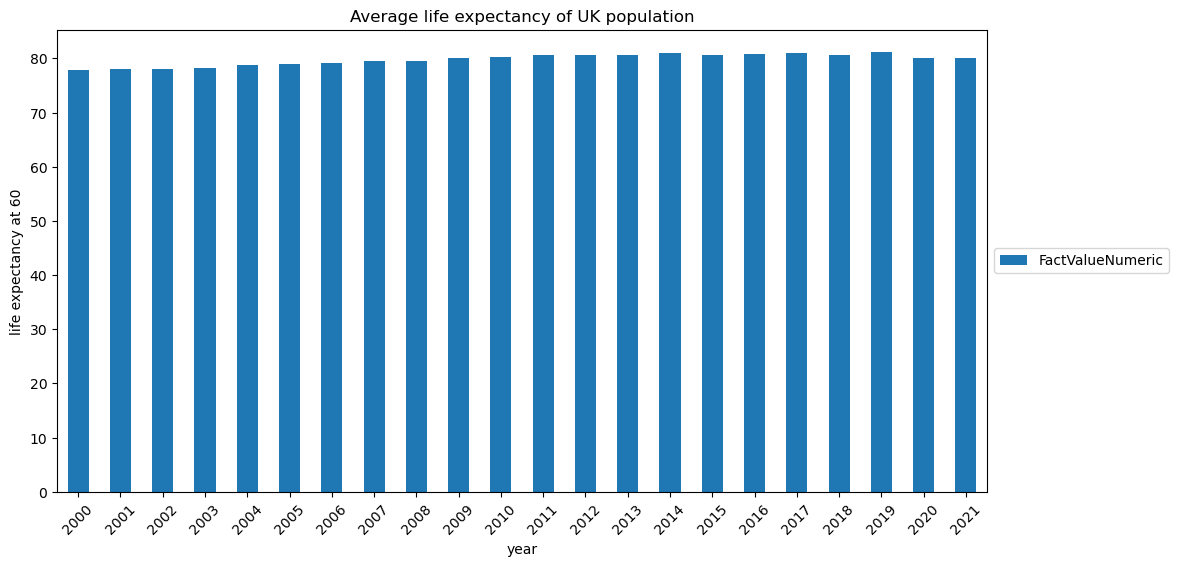


## Life expectancy at 60 based on male, female and both sex

%%pyspark
  
import pandas as pd
  
import matplotlib.pyplot as plt
  
df = spark.sql("SELECT \* FROM `default`.`lifeexpectancybirthand60` where `Country Code` = 'GBR' and FactValueNumeric >= 60 and Period >= 1999 and Sex5 = 'SEX\_BTSX' order by Period")
  
  
# select un-necessary columns to remove
  
cols\_to\_drop = [df.columns[0], df.columns[1], df.columns[3], df.columns[4], df.columns[5]]
  
  
# Drop the specified columns
  
df = df.drop(\*cols\_to\_drop)
  
  
df.show(23, False)
  
  
pandas\_df = df.toPandas()
  
# as it is loaded from csv,Identify the columns that need to be converted to numeric
  
columns\_to\_convert = list(pandas\_df.columns)[1:]
  
  
# Convert the specified columns to numeric
  
for col in columns\_to\_convert:
  
 pandas\_df[col] = pd.to\_numeric(pandas\_df[col], errors='coerce')
  
  
pandas\_df = pandas\_df.set\_index('Period')
  
  
pandas\_df.plot(kind='bar', stacked=True, figsize=(12, 6))
  
  
  
# Customize plot
  
plt.title('Average life expectancy of UK population')
  
plt.xlabel('year')
  
plt.ylabel('life expectancy at 60')
  
plt.xticks(rotation=45)
  
plt.legend(loc='center left', bbox\_to\_anchor=(1, 0.5))
  
  
plt.show()

{"execution\_finish\_time":"2025-03-20T23:42:42.536102Z","execution\_start\_time":"2025-03-20T23:42:39.5991502Z","livy\_statement\_state":"available","normalized\_state":"finished","parent\_msg\_id":"0eaf9c1f-1e0c-4695-b5d7-8ad5b037eb77","queued\_time":"2025-03-20T23:36:00.4374543Z","session\_id":"47","session\_start\_time":null,"spark\_jobs":null,"spark\_pool":"ghparkcluster","state":"finished","statement\_id":10,"statement\_ids":[10]}

+------+----------------+  
|Period|FactValueNumeric|  
+------+----------------+  
|2000 |77.86 |  
|2001 |77.95 |  
|2002 |78.08 |  
|2003 |78.15 |  
|2004 |78.68 |  
|2005 |78.91 |  
|2006 |79.23 |  
|2007 |79.42 |  
|2008 |79.54 |  
|2009 |80.03 |  
|2010 |80.25 |  
|2011 |80.64 |  
|2012 |80.64 |  
|2013 |80.69 |  
|2014 |80.94 |  
|2015 |80.66 |  
|2016 |80.81 |  
|2017 |80.92 |  
|2018 |80.62 |  
|2019 |81.22 |  
|2020 |80.14 |  
|2021 |80.1 |  
+------+----------------+



## UK Primary Health Expenses

%%pyspark
  
from pyspark.sql import functions as F
  
df = spark.sql('SELECT \* FROM `default`.`phcexpendituretrends` where Countries like "%United%"')
  
df.show(10)
  
# select un-necessary columns to remove
  
cols\_to\_drop = [df.columns[1]]
  
# Drop the specified columns
  
df = df.drop(\*cols\_to\_drop)
  
df.show(10)
  
  
# Get the year columns dynamically
  
year\_columns = [col for col in df.columns if col != "Countries"]
  
  
# Create the stack expression dynamically
  
stack\_expr = "stack(" + str(len(year\_columns)) + ", "
  
  
for year\_col in year\_columns:
  
 stack\_expr += f"'{year\_col}', `{year\_col}`, "
  
stack\_expr = stack\_expr[:-2] + ") as (Year, Value)" #Remove trailing comma and space and close stack expression
  
  
# Transpose the DataFrame
  
transposed\_df = df.select("Countries", F.expr(stack\_expr)).where("Value is not null")
  
  
# Rename the "Year" column
  
transposed\_df = transposed\_df.withColumnRenamed("Countries", "Category")
  
  
# Transpose the DataFrame
  
transposed\_df = df.select("Countries", F.expr(stack\_expr)).where("Value is not null")
  
  
transposed\_df = transposed\_df.drop(\*[transposed\_df.columns[0]] )
  
  
transposed\_df.show(10)

{"execution\_finish\_time":"2025-03-20T23:42:44.4946324Z","execution\_start\_time":"2025-03-20T23:42:42.5512371Z","livy\_statement\_state":"available","normalized\_state":"finished","parent\_msg\_id":"aa40075a-40cd-4195-902a-e873410a366b","queued\_time":"2025-03-20T23:36:00.4379981Z","session\_id":"47","session\_start\_time":null,"spark\_jobs":null,"spark\_pool":"ghparkcluster","state":"finished","statement\_id":11,"statement\_ids":[11]}

+--------------------+--------------------+-----+-----+-----+-----+-----+-----+-----+----+  
| Countries| Indicators| 2016| 2017| 2018| 2019| 2020| 2021| 2022|2023|  
+--------------------+--------------------+-----+-----+-----+-----+-----+-----+-----+----+  
|United Kingdom of...|Primary Health Ca...|1,499|1,457|1,597|1,587|1,795|2,348|1,906|null|  
+--------------------+--------------------+-----+-----+-----+-----+-----+-----+-----+----+  
  
+--------------------+-----+-----+-----+-----+-----+-----+-----+----+  
| Countries| 2016| 2017| 2018| 2019| 2020| 2021| 2022|2023|  
+--------------------+-----+-----+-----+-----+-----+-----+-----+----+  
|United Kingdom of...|1,499|1,457|1,597|1,587|1,795|2,348|1,906|null|  
+--------------------+-----+-----+-----+-----+-----+-----+-----+----+  
  
+----+-----+  
|Year|Value|  
+----+-----+  
|2016|1,499|  
|2017|1,457|  
|2018|1,597|  
|2019|1,587|  
|2020|1,795|  
|2021|2,348|  
|2022|1,906|  
+----+-----+

## UK GDP price index (100 in 2022)

%%pyspark
  
df = spark.sql("SELECT \* FROM `default`.`ukgdppriceindex2022is100` order by Indicators")
  
df.show(10)
  
  
# Get the year columns dynamically
  
year\_columns = [col for col in df.columns if col != "Indicators"]
  
  
# Create the stack expression dynamically
  
stack\_expr = "stack(" + str(len(year\_columns)) + ", "
  
  
for year\_col in year\_columns:
  
 stack\_expr += f"'{year\_col}', `{year\_col}`, "
  
stack\_expr = stack\_expr[:-2] + ") as (Year, Value)" #Remove trailing comma and space and close stack expression
  
  
# Transpose the DataFrame
  
transposed\_df = df.select("Indicators", F.expr(stack\_expr)).where("Value is not null")
  
  
# Rename the "Year" column
  
transposed\_df = transposed\_df.withColumnRenamed("CouIndicatorsntries", "Category")
  
  
# Transpose the DataFrame
  
transposed\_df = df.select("Indicators", F.expr(stack\_expr)).where("Value is not null")
  
  
transposed\_df = transposed\_df.drop(\*[transposed\_df.columns[0]] )
  
  
transposed\_df.show(24)

{"execution\_finish\_time":"2025-03-20T23:42:46.4655716Z","execution\_start\_time":"2025-03-20T23:42:44.508493Z","livy\_statement\_state":"available","normalized\_state":"finished","parent\_msg\_id":"868984ab-bb36-41ed-b8b9-97102396c65d","queued\_time":"2025-03-20T23:36:00.438558Z","session\_id":"47","session\_start\_time":null,"spark\_jobs":null,"spark\_pool":"ghparkcluster","state":"finished","statement\_id":12,"statement\_ids":[12]}

+--------------------+----+----+----+----+----+----+----+----+----+----+----+----+----+----+----+----+----+----+----+----+----+----+----+----+  
| Indicators|2000|2001|2002|2003|2004|2005|2006|2007|2008|2009|2010|2011|2012|2013|2014|2015|2016|2017|2018|2019|2020|2021|2022|2023|  
+--------------------+----+----+----+----+----+----+----+----+----+----+----+----+----+----+----+----+----+----+----+----+----+----+----+----+  
|Gross domestic pr...| 62| 62| 64| 65| 67| 69| 71| 72| 75| 76| 78| 79| 80| 82| 83| 84| 85| 87| 89| 91| 95| 95| 100| 107|  
+--------------------+----+----+----+----+----+----+----+----+----+----+----+----+----+----+----+----+----+----+----+----+----+----+----+----+  
  
+----+-----+  
|Year|Value|  
+----+-----+  
|2000| 62|  
|2001| 62|  
|2002| 64|  
|2003| 65|  
|2004| 67|  
|2005| 69|  
|2006| 71|  
|2007| 72|  
|2008| 75|  
|2009| 76|  
|2010| 78|  
|2011| 79|  
|2012| 80|  
|2013| 82|  
|2014| 83|  
|2015| 84|  
|2016| 85|  
|2017| 87|  
|2018| 89|  
|2019| 91|  
|2020| 95|  
|2021| 95|  
|2022| 100|  
|2023| 107|  
+----+-----+

## UK NET Migration

%%pyspark
  
df = spark.sql("SELECT \* FROM `default`.`uknetmigration` where Time > 1999")
  
df = df.drop(\*[df.columns[0]] )
  
df.show(24)

{"execution\_finish\_time":"2025-03-20T23:42:47.6334807Z","execution\_start\_time":"2025-03-20T23:42:46.4792523Z","livy\_statement\_state":"available","normalized\_state":"finished","parent\_msg\_id":"d40a8285-ecc7-4084-a94e-6844de87b919","queued\_time":"2025-03-20T23:36:00.4390972Z","session\_id":"47","session\_start\_time":null,"spark\_jobs":null,"spark\_pool":"ghparkcluster","state":"finished","statement\_id":13,"statement\_ids":[13]}

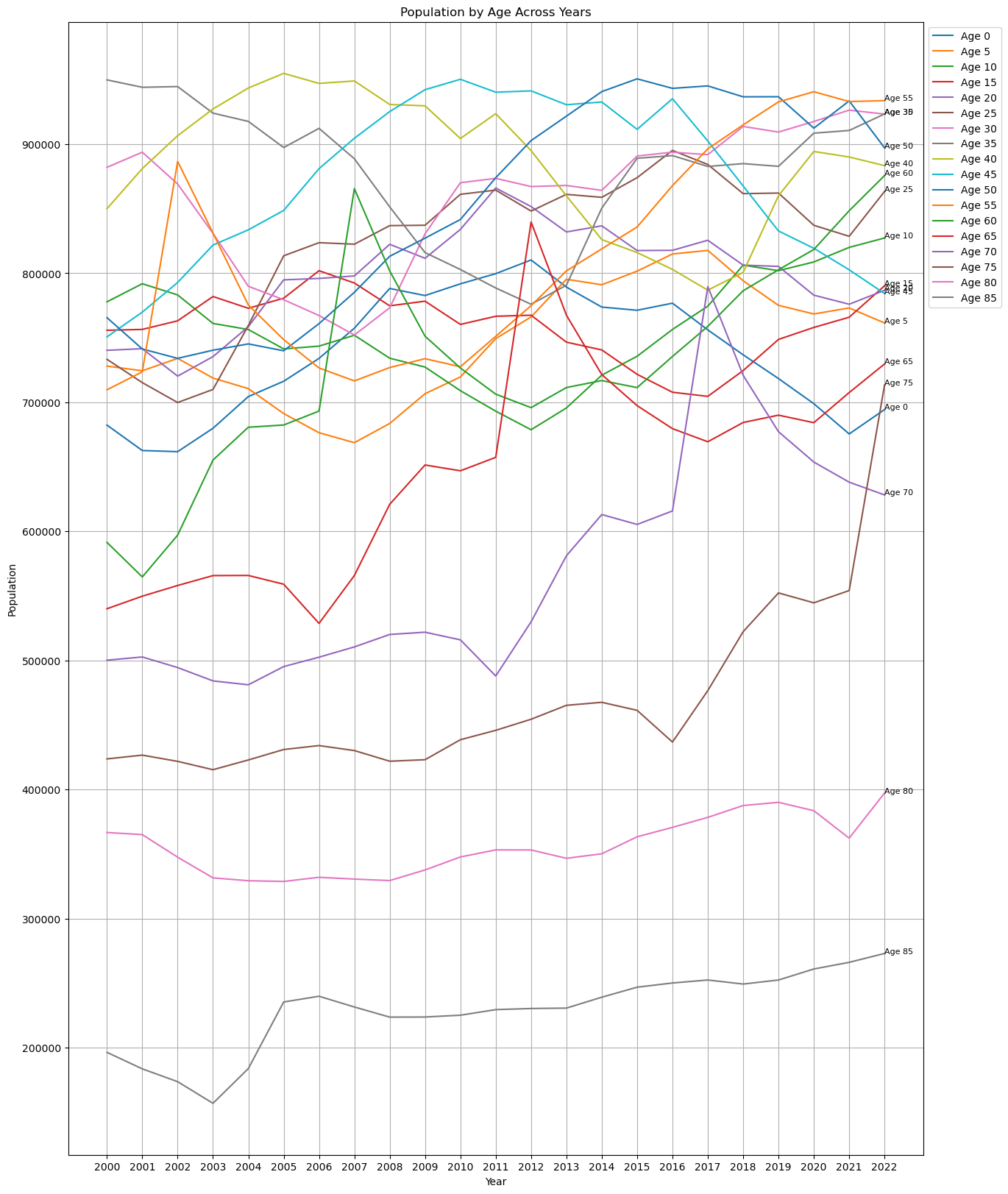
+----+------+  
|Time| Value|  
+----+------+  
|2000|153917|  
|2001|175133|  
|2002|203577|  
|2003|211846|  
|2004|276077|  
|2005|308279|  
|2006|288923|  
|2007|309385|  
|2008|263390|  
|2009|249639|  
|2010|274811|  
|2011|228773|  
|2012|172088|  
|2013|237272|  
|2014|306091|  
|2015|347839|  
|2016|324649|  
|2017|272311|  
|2018|259744|  
|2019|232727|  
|2020|144221|  
|2021|462967|  
|2022|487029|  
|2023|445523|  
+----+------+  
only showing top 24 rows

## UK population by age

%%pyspark
  
  
import matplotlib.pyplot as plt
  
import pandas as pd
  
  
# SQL query to select the data
  
sql = "SELECT `Age`, `2000`, `2001`, `2002`, `2003`, `2004`, `2005`, `2006`, `2007`, `2008`, `2009`, `2010`, `2011`, `2012`, `2013`, `2014`, `2015`, `2016`, `2017`, `2018`, `2019`, `2020`, `2021`, `2022` FROM ukpopulation\_byage\_total WHERE Age < 90"
  
  
# Execute the query and get the DataFrame
  
df = spark.sql(sql)
  
  
# Show the DataFrame
  
df.show(10, False)
  
  
# Convert the Spark DataFrame to a Pandas DataFrame
  
pdf = df.toPandas()
  
  
# Set the Age column as the index
  
pdf.set\_index('Age', inplace=True)
  
  
# Transpose the DataFrame to have years as rows and ages as columns
  
df\_transposed = pdf.T
  
  
# Convert all column values to numeric (float or int)
  
df\_transposed = df\_transposed.apply(lambda x: pd.to\_numeric(x.str.replace(',', ''), errors='coerce'))
  
  
  
# Plot the graph
  
plt.figure(figsize=(15, 20))
  
# Plot a subset of the data (e.g., every 5th age group)
  
for age in df\_transposed.columns[::5]:
  
 plt.plot(df\_transposed.index.astype(str), df\_transposed[age], label=f'Age {age}')
  
 # Add text annotations for each line (legend on the graph itself)
  
 plt.annotate(f'Age {age}', (df\_transposed.index[-1], df\_transposed[age].values[-1]), fontsize=8)
  
  
# Add grid lines
  
plt.grid(True)
  
  
# Add title and labels
  
plt.title('Population by Age Across Years')
  
plt.xlabel('Year')
  
plt.ylabel('Population')
  
# Show legend over the graph itself
  
plt.legend(loc='upper left', bbox\_to\_anchor=(1, 1))
  
plt.show()

{"execution\_finish\_time":"2025-03-20T23:42:49.6463874Z","execution\_start\_time":"2025-03-20T23:42:47.6472407Z","livy\_statement\_state":"available","normalized\_state":"finished","parent\_msg\_id":"5a46b54c-1b8f-43df-80c1-399adc174db3","queued\_time":"2025-03-20T23:36:00.4396539Z","session\_id":"47","session\_start\_time":null,"spark\_jobs":null,"spark\_pool":"ghparkcluster","state":"finished","statement\_id":14,"statement\_ids":[14]}

+---+------+------+------+------+------+------+------+------+------+------+------+------+------+------+------+------+------+------+------+------+------+------+------+  
|Age|2000 |2001 |2002 |2003 |2004 |2005 |2006 |2007 |2008 |2009 |2010 |2011 |2012 |2013 |2014 |2015 |2016 |2017 |2018 |2019 |2020 |2021 |2022 |  
+---+------+------+------+------+------+------+------+------+------+------+------+------+------+------+------+------+------+------+------+------+------+------+------+  
|0 |682385|662662|661786|679910|704450|716331|734035|757429|788225|782726|791801|799686|810270|789278|773792|771325|776812|756135|736831|718410|698956|675503|694401|  
|1 |701642|679455|664800|662639|679406|704755|716092|737641|758643|787962|781631|789910|804353|813213|792690|776857|774080|777784|755272|736596|719804|702771|690758|  
|2 |712124|698903|682505|667850|663651|680794|704383|717749|740782|759942|788659|782209|793396|807232|815304|794490|778370|773200|776393|753572|733178|719432|716737|  
|3 |731484|710986|702372|685822|670805|665589|681323|705686|718402|743661|761633|790453|785327|796108|809997|816833|795593|777281|771230|775597|750732|730634|732823|  
|4 |723535|730156|714737|706437|689289|674460|666792|683009|705931|718630|746833|764623|792931|787955|798836|812416|817806|794858|775628|770377|773574|749403|742836|  
|5 |728103|724436|734046|718816|710619|691326|676448|668737|683652|706675|719662|749378|766166|795348|791076|801633|814920|817718|794185|775133|768443|773122|761499|  
|6 |747575|727937|728146|738183|722477|712945|693670|680008|670158|684756|707338|719588|750418|767679|798043|793824|804040|815878|816989|794340|773899|768118|785056|  
|7 |755711|745817|731009|732731|742354|724826|715647|697650|683270|672111|685914|706879|723299|751615|769533|800681|796576|805402|816478|817312|793911|773893|779908|  
|8 |780381|755780|749690|734751|737099|744563|727362|719998|701198|686883|673953|686034|711217|727113|753056|771332|803011|798172|806239|817768|816567|794321|784975|  
|9 |791817|780454|758094|753773|738187|739928|747299|731213|723676|705020|690309|674821|690806|716032|731534|754804|773411|804828|799835|807966|818206|816921|805260|  
+---+------+------+------+------+------+------+------+------+------+------+------+------+------+------+------+------+------+------+------+------+------+------+------+  
only showing top 10 rows



## World health data

%%pyspark
  
resultDF = spark.sql('SELECT year,health\_exp as health\_exp\_percentage\_gdp,life\_expect, maternal\_mortality,infant\_mortality,neonatal\_mortality, under\_5\_mortality FROM `default`.`worldhealthdata` where country\_code = "GBR" and year > 1999 and year < 2023')
  
resultDF.show(24)
  
  
schema\_name = 'warehouse'
  
table\_name = 'uk\_health\_data'
  
  
spark\_sql\_schema\_Create = f"CREATE SCHEMA IF NOT EXISTS {schema\_name};"
  
spark.sql(spark\_sql\_schema\_Create)
  
  
resultDF.write.format("delta").mode("ignore").saveAsTable(f"{schema\_name}.{table\_name}")

{"execution\_finish\_time":"2025-03-20T23:42:51.5562693Z","execution\_start\_time":"2025-03-20T23:42:49.6759957Z","livy\_statement\_state":"available","normalized\_state":"finished","parent\_msg\_id":"e200902b-2617-4166-abf8-df46c71fd547","queued\_time":"2025-03-20T23:36:00.4401856Z","session\_id":"47","session\_start\_time":null,"spark\_jobs":null,"spark\_pool":"ghparkcluster","state":"finished","statement\_id":15,"statement\_ids":[15]}

+----+-------------------------+-----------+------------------+----------------+------------------+-----------------+  
|year|health\_exp\_percentage\_gdp|life\_expect|maternal\_mortality|infant\_mortality|neonatal\_mortality|under\_5\_mortality|  
+----+-------------------------+-----------+------------------+----------------+------------------+-----------------+  
|2000| 7.12928009|77.74146341| 11| 5.6| 3.8| 6.5|  
|2001| 7.34416342|77.99268293| 11| 5.5| 3.7| 6.4|  
|2002| 7.65009499|78.14390244| 11| 5.4| 3.6| 6.3|  
|2003| 8.05579185|78.44634146| 11| 5.3| 3.6| 6.2|  
|2004| 8.30715179|78.74634146| 11| 5.2| 3.5| 6.1|  
|2005| 8.36301231|79.04878049| 11| 5.1| 3.5| 6|  
|2006| 8.62893772|79.24878049| 11| 5| 3.4| 5.9|  
|2007| 8.62399578|79.44878049| 11| 4.9| 3.3| 5.8|  
|2008| 9.02561569| 79.6| 11| 4.8| 3.2| 5.6|  
|2009| 10.04505444|80.05121951| 10| 4.6| 3.1| 5.4|  
|2010| 9.92673969|80.40243902| 10| 4.4| 3| 5.2|  
|2011| 9.92486763|80.95121951| 9| 4.3| 3| 5|  
|2012| 9.95296001|80.90487805| 8| 4.1| 2.9| 4.8|  
|2013| 9.87251186|81.00487805| 8| 4| 2.8| 4.6|  
|2014| 9.86246872|81.30487805| 8| 3.9| 2.7| 4.5|  
|2015| 9.79776669|80.95609756| 8| 3.9| 2.7| 4.5|  
|2016| 9.72899723|81.15609756| 9| 3.8| 2.8| 4.4|  
|2017| 9.5958786|81.25609756| 9| 3.8| 2.8| 4.4|  
|2018| 9.73079777|81.25609756| 9| 3.8| 2.8| 4.4|  
|2019| 9.95761967|81.40487805| 9| 3.8| 2.8| 4.3|  
|2020| 12.15863419|80.35121951| 10| 3.7| 2.8| 4.3|  
|2021| 12.36469936| 80.7| null| 3.6| 2.8| 4.2|  
|2022| 11.34454918|82.05853659| null| 3.6| 2.7| 4.1|  
+----+-------------------------+-----------+------------------+----------------+------------------+-----------------+

df = spark.sql("SHOW schemas;")
  
df.show()

{"execution\_finish\_time":"2025-03-20T23:42:52.1808123Z","execution\_start\_time":"2025-03-20T23:42:51.5707087Z","livy\_statement\_state":"available","normalized\_state":"finished","parent\_msg\_id":"e9699d9c-8fb4-4850-a69c-07ff9d66794b","queued\_time":"2025-03-20T23:36:00.4407073Z","session\_id":"47","session\_start\_time":null,"spark\_jobs":null,"spark\_pool":"ghparkcluster","state":"finished","statement\_id":16,"statement\_ids":[16]}

+---------+  
|namespace|  
+---------+  
| default|  
|warehouse|  
+---------+