In [1]:

```
from sqlalchemy import create_engine
import pandas as pd
import platform
import socket
from datetime import datetime
```

In [202]:

```
df_old = pd.DataFrame(
    [['a', 'x', 0, 0, 0], ['a', 'y', 0, 0, 0], ['a', 'z', 0, 0, 0],
     ['b', 'x', 0, 0, 0], ['b', 'y', 0, 0, 0], ['b', 'z', 0, 0, 0],
    ],
    columns=['i1', 'i2', 'c1', 'c2', 'c3']).set_index(['i1', 'i2'])

df_new = pd.DataFrame([['x', 1, 2], ['y', 3, 4]],
                      columns=['i2', 'c1', 'c2']).set_index('i2')

df_old
```

Out[202]:

		c1	c2	c3
i1	**i2**			
a	**x**	0	0	0
	y	0	0	0
	z	0	0	0
b	**x**	0	0	0
	y	0	0	0
	z	0	0	0

In [209]:

```
df_new.index
```

Out[209]:

```
Index(['x', 'y'], dtype='object', name='i2')
```

In [208]:

```
df_old.loc[('a', df_new.index), df_new.columns]
```

Out[208]:

		c1	c2
i1	**i2**		
a	**x**	1	2
	y	3	4

In [207]:

```
df_old.loc[('a', df_new.index), df_new.columns] = df_new.values
df_old
```

Out[207]:

		c1	c2	c3
i1	**i2**			
a	**x**	1	2	0
	y	3	4	0
	z	0	0	0
b	**x**	0	0	0
	y	0	0	0
	z	0	0	0

In [266]:

```
import pandas as pd

# Sample dataframes
df1 = pd.DataFrame({'Name': ['John', 'Alice', 'Bob'],
                    'Age': [25, 30, 35],
                    'City': ['New York', 'Los Angeles', 'Chicago']})

df2 = pd.DataFrame({'Name': ['John', 'Alice', 'Bob', 'Mike'],
                    'Age': [25, 30, 35, 40],
                    'City': ['New York', 'Los Angeles', 'Chicago', 'Boston']})

print(df1)
print(df2)
```

```
    Name  Age         City
0   John   25     New York
1  Alice   30  Los Angeles
2    Bob   35      Chicago
    Name  Age         City
0   John   25     New York
1  Alice   30  Los Angeles
2    Bob   35      Chicago
3   Mike   40       Boston
```

In [267]:

```
# Find new rows in df2
new_rows = df2[~df2.isin(df1)].dropna()

# Update rows in df1 with new values from df2
df1.update(new_rows)

print("Updated df1:")
print(df1)
```

```
Updated df1:
    Name  Age         City
0   John   25     New York
1  Alice   30  Los Angeles
2    Bob   35      Chicago
```

In [301]:

```
import pandas as pd

# Main dataframe (df1)
df1 = pd.DataFrame({'ID': [1, 2, 3, 4],
                    'Name': ['John', 'Alice', 'Bob', 'Mike'],
                    'Age': [25, 30, 35, 40]})

# DataFrame to compare (df2)
df2 = pd.DataFrame({'ID': [3, 4, 5],
                    'Name': ['Bob', 'Mike', 'Sarah'],
                    'Age': [35, 40, 28]})

# Compare dataframes
compared_df = pd.merge(df1, df2, on=['ID', 'Name', 'Age'], how='inner')

# Display compared dataframe
print(compared_df)
```

```
   ID  Name  Age
0   3   Bob   35
1   4  Mike   40
```

In [10]:

```
df1['price_2'] = df2['price_2']
```

In [11]:

```
df1['prices_match'] = np.where(df1['price_1'] == df2['price_2'], 'True', 'False')
```

In [12]:

```
df1
```

Out[12]:

	product_1	price_1	price_2	prices_match
0	computer	1200	900	False
1	monitor	800	800	True
2	printer	200	300	False
3	desk	350	350	True

In [26]:

```
import pandas as pd

technologies = ({
    'Courses':["Spark", "NumPY", "pandas", "Java", "PySpark"],
    'Fee' :[20000,25000,30000,22000,26000],
    'Duration':['30days','40days','35days','60days','50days'],
    'Discount':[1000,2500,1500,1200,3000]
               })
technologies1 = ({
    'Courses':["Spark", "Hadoop", "pandas", "Java", "PySpark"],
    'Fee' :[20000,24000,30000,22000,21000],
    'Duration':['30days','40days','35days','60days','50days'],
    'Discount':[1000,2500,1500,1200,3000]
               })
df1 = pd.DataFrame(technologies)
print("DataFrame1:\n", df1)
df2 = pd.DataFrame(technologies1)
print("DataFrame2:\n", df2)
```

```
DataFrame1:
    Courses    Fee Duration  Discount
0    Spark  20000   30days      1000
1    NumPY  25000   40days      2500
2   pandas  30000   35days      1500
3     Java  22000   60days      1200
4  PySpark  26000   50days      3000
DataFrame2:
    Courses    Fee Duration  Discount
0    Spark  20000   30days      1000
1   Hadoop  24000   40days      2500
2   pandas  30000   35days      1500
3     Java  22000   60days      1200
4  PySpark  21000   50days      3000
```

In [27]:

```
df1
```

Out[27]:

	Courses	Fee	Duration	Discount
0	Spark	20000	30days	1000
1	NumPY	25000	40days	2500
2	pandas	30000	35days	1500
3	Java	22000	60days	1200
4	PySpark	26000	50days	3000

In [28]:

```
df2
```

Out[28]:

	Courses	Fee	Duration	Discount
0	Spark	20000	30days	1000
1	Hadoop	24000	40days	2500
2	pandas	30000	35days	1500
3	Java	22000	60days	1200
4	PySpark	21000	50days	3000

In [64]:

```
df1[df1 == df2]
```

Out[64]:

	Courses	Fee	Duration	Discount
0	Spark	20000.0	30days	1000
1	NaN	NaN	40days	2500
2	pandas	30000.0	35days	1500
3	Java	22000.0	60days	1200
4	PySpark	NaN	50days	3000

In [61]:

```
new_df = df1.merge(df2, indicator=True, how="left")[lambda x: x._merge=='left_only'].dr
op('_merge',1)
new_df
```

Out[61]:

	Courses	Fee	Duration	Discount
1	NumPY	25000	40days	2500
4	PySpark	26000	50days	3000

In [76]:

```
df1.update(df2,join='left', overwrite=False, filter_func=None, errors='ignore')
```

In [78]:

```
df1
```

Out[78]:

	Courses	Fee	Duration	Discount
0	Spark	20000	30days	1000
1	NumPY	25000	40days	2500
2	pandas	30000	35days	1500
3	Java	22000	60days	1200
4	PySpark	26000	50days	3000

In [40]:

```
from sqlalchemy.sql import text
from sqlalchemy import create_engine
import pandas as pd
from datetime import datetime
```

In [65]:

```
data1 = {'Investment Name':['Canada Guaranty','Canada Guaranty','Canada Guaranty','Cana
da Guaranty','Canada Guaranty','Canada Guaranty'],
        'Metric':['Equity Val','Equity Val','Equity Val','Equity Val','Equity Val','Equ
ity Val'],
        'Order':[3,3,3,3,3,3],
        'Period':['Actual','Actual','Actual','Entry','F3','F4'],
        'As of Date':['2020-12-31','2021-12-31','2022-12-31','2010-12-31','2022-12-3
1','2022-12-31'],
        'Unit':[1000000,1000000,1000000,1000000,1000000,1000000],
        'Value':["2.542144e+09", "3.307770e+09", "3.404744e+09", "1.022000e+08", "3.081
153e+09", "3.404744e+09"]}
val_df = pd.DataFrame(data1)
val_df
```

Out[65]:

	Investment Name	Metric	Order	Period	As of Date	Unit	Value
0	Canada Guaranty	Equity Val	3	Actual	2020-12-31	1000000	2.542144e+09
1	Canada Guaranty	Equity Val	3	Actual	2021-12-31	1000000	3.307770e+09
2	Canada Guaranty	Equity Val	3	Actual	2022-12-31	1000000	3.404744e+09
3	Canada Guaranty	Equity Val	3	Entry	2010-12-31	1000000	1.022000e+08
4	Canada Guaranty	Equity Val	3	F3	2022-12-31	1000000	3.081153e+09
5	Canada Guaranty	Equity Val	3	F4	2022-12-31	1000000	3.404744e+09

In [68]:

```
data2 = {'Investment_Name':['Canada Guaranty','Canada Guaranty','Canada Guaranty','Cana
da Guaranty','Canada Guaranty','Canada Guaranty','Canada Guaranty','Canada Guaranty','C
anada Guaranty','Canada Guaranty'],
        'Metric':['Equity Val','Equity Val','Equity Val','Equity Val','Equity Val','Equ
ity Val','Equity Val','Equity Val','Equity Val','Equity Val'],
         'Order_':[3,3,3,3,3,3,3,3,3,3],
        'Period':['Actual','Actual','Actual','Actual','Actual','Entry','F1','F2','F
3','F4'],
        'As_of_Date':['2019-12-31','2020-12-31','2021-06-30','2021-12-31','2022-12-3
1','2010-04-01','2021-12-31','2021-12-31','2022-12-31','2022-12-31'],
        'Unit':[1000000,1000000,1000000,1000000,1000000,1000000, 1000000,1000000,100000
0,1000000],
        'Value_':["1.973737e+09", "2.542144e+09", "2.610065e+09", "3.307770e+09", "3.40
4744e+09", "1.022000e+08","2.815691e+09","2.818039e+09","3.081183e+09","3.404744e+09"]}
df2 = pd.DataFrame(data2)

def dbConnection():
    server = 'PAVAN'
    dbname = 'company'
    engine = create_engine("mssql+pyodbc://"+server+"/"+dbname+"?driver=SQL+Server").ex
ecution_options(isolation_level='AUTOCOMMIT')
    return engine

eng = dbConnection()
con = eng.connect()
#con.execute(text("Truncate Table Canada").execution_options(autocommit=True))
df2.to_sql('Canada', con, schema=None, if_exists='append', index=False, index_label=Non
e, chunksize=None, dtype=None, method=None)
sql_df = pd.read_sql("select * from Canada", con)
sql_df
```

Out[68]:

	Investment_Name	Metric	Order_	Period	As_of_Date	Unit	Value_	Cre
0	Canada Guaranty	Equity Val	3	Actual	2019-12-31	1000000	1.973737e+09	Non
1	Canada Guaranty	Equity Val	3	Actual	2020-12-31	1000000	2.542144e+09	Non
2	Canada Guaranty	Equity Val	3	Actual	2021-06-30	1000000	2.610065e+09	Non
3	Canada Guaranty	Equity Val	3	Actual	2021-12-31	1000000	3.307770e+09	Non
4	Canada Guaranty	Equity Val	3	Actual	2022-12-31	1000000	3.404744e+09	Non
5	Canada Guaranty	Equity Val	3	Entry	2010-04-01	1000000	1.022000e+08	Non
6	Canada Guaranty	Equity Val	3	F1	2021-12-31	1000000	2.815691e+09	Non
7	Canada Guaranty	Equity Val	3	F2	2021-12-31	1000000	2.818039e+09	Non
8	Canada Guaranty	Equity Val	3	F3	2022-12-31	1000000	3.081183e+09	Non
9	Canada Guaranty	Equity Val	3	F4	2022-12-31	1000000	3.404744e+09	Non

In [43]:

```
db_df = sql_df.iloc[:,:-2]
db_df
```

Out[43]:

	Investment_Name	Metric	Order_	Period	As_of_Date	Unit	Value_
0	Canada Guaranty	Equity Val	3	Actual	2019-12-31	1000000	1.973737e+09
1	Canada Guaranty	Equity Val	3	Actual	2020-12-31	1000000	2.542144e+09
2	Canada Guaranty	Equity Val	3	Actual	2021-06-30	1000000	2.610065e+09
3	Canada Guaranty	Equity Val	3	Actual	2021-12-31	1000000	3.307770e+09
4	Canada Guaranty	Equity Val	3	Actual	2022-12-31	1000000	3.404744e+09
5	Canada Guaranty	Equity Val	3	Entry	2010-04-01	1000000	1.022000e+08
6	Canada Guaranty	Equity Val	3	F1	2021-12-31	1000000	2.815691e+09
7	Canada Guaranty	Equity Val	3	F2	2021-12-31	1000000	2.818039e+09
8	Canada Guaranty	Equity Val	3	F3	2022-12-31	1000000	3.081153e+09
9	Canada Guaranty	Equity Val	3	F4	2022-12-31	1000000	3.404744e+09

In [44]:

```
old_col_names = list(val_df.columns)
old_col_names
```

Out[44]:

```
['Investment Name', 'Metric', 'Order', 'Period', 'As of Date', 'Unit', 'Va
lue']
```

In [45]:

```
new_col_names = list(db_df.columns)
new_col_names
```

Out[45]:

```
['Investment_Name',
 'Metric',
 'Order_',
 'Period',
 'As_of_Date',
 'Unit',
 'Value_']
```

In [46]:

```
val_df.rename(columns=dict(zip(old_col_names, new_col_names)), inplace=True)
```

In [47]:

```
val_df
```

Out[47]:

	Investment_Name	Metric	Order_	Period	As_of_Date	Unit	Value_
0	Canada Guaranty	Equity Val	3	Actual	2020-12-31	1000000	2.542144e+09
1	Canada Guaranty	Equity Val	3	Actual	2021-12-31	1000000	3.307770e+09
2	Canada Guaranty	Equity Val	3	Actual	2022-12-31	1000000	3.404744e+09
3	Canada Guaranty	Equity Val	3	Entry	2010-12-31	1000000	1.022000e+08
4	Canada Guaranty	Equity Val	3	F3	2022-12-31	1000000	3.081153e+09
5	Canada Guaranty	Equity Val	3	F4	2022-12-31	1000000	3.404744e+09

In [48]:

```
list(val_df.columns)[:5]
```

Out[48]:

```
['Investment_Name', 'Metric', 'Order_', 'Period', 'As_of_Date']
```

In [49]:

```
val_df1 = val_df.sort_values(by=list(val_df.columns)[:5]).reset_index(drop=True)
val_df1.head()
```

Out[49]:

	Investment_Name	Metric	Order_	Period	As_of_Date	Unit	Value_
0	Canada Guaranty	Equity Val	3	Actual	2020-12-31	1000000	2.542144e+09
1	Canada Guaranty	Equity Val	3	Actual	2021-12-31	1000000	3.307770e+09
2	Canada Guaranty	Equity Val	3	Actual	2022-12-31	1000000	3.404744e+09
3	Canada Guaranty	Equity Val	3	Entry	2010-12-31	1000000	1.022000e+08
4	Canada Guaranty	Equity Val	3	F3	2022-12-31	1000000	3.081153e+09

In [63]:

```
db_df2 = db_df.sort_values(by=list(db_df.columns)[:5]).reset_index(drop=True)
db_df2['As_of_Date'] = pd.to_datetime(db_df2['As_of_Date'])
db_df2.head()
```

Out[63]:

	Investment_Name	Metric	Order_	Period	As_of_Date	Unit	Value_
0	Canada Guaranty	Equity Val	3	Actual	2019-12-31	1000000	1.973737e+09
1	Canada Guaranty	Equity Val	3	Actual	2020-12-31	1000000	2.542144e+09
2	Canada Guaranty	Equity Val	3	Actual	2021-06-30	1000000	2.610065e+09
3	Canada Guaranty	Equity Val	3	Actual	2021-12-31	1000000	3.307770e+09
4	Canada Guaranty	Equity Val	3	Actual	2022-12-31	1000000	3.404744e+09

In [51]:

```
load_df =  val_df1.merge(db_df2, indicator=True, how="left")[lambda x: x._merge=='left_
only'].drop('_merge',1)
load_df
```

Out[51]:

	Investment_Name	Metric	Order_	Period	As_of_Date	Unit	Value_
3	Canada Guaranty	Equity Val	3	Entry	2010-12-31	1000000	1.022000e+08

In [52]:

```
if len(load_df) == 0:
    current_date = datetime.now().strftime("%Y-%m-%d %H:%M:%S")
    for index, row in df1.iterrows():
        update_query = "update Valuation set Update_Dt='{0}' where Investment_Name
='{1}' and Metric='{2}' and Order_='{3}' and Period='{4}' and As_of_Date ='{5}'".format
(current_date,row[0],row[1],row[2],row[3],row[4],row[5])
        print(update_query)
        #cursor = engine.raw_connection().cursor()
        #cursor.execute(update_query)
        #cursor.commit()
        print('updated Date successfully ')
else:
    pass
```

In [53]:

```
#compared_df = pd.merge(db_df2, load_df, on=['Investment_Name', 'Metric', 'Order_', 'Period', 'As_of_Date'], how='right')
compared_df = pd.merge(db_df2, load_df, on=new_col_names[:-2], how='right')
compared_df
```

Out[53]:

	Investment_Name	Metric	Order_	Period	As_of_Date	Unit_x	Value__x	Unit_y
0	Canada Guaranty	Equity Val	3	Entry	2010-12-31	NaN	NaN	1000000

In [54]:

```
x_cols = list(compared_df.filter(regex=("x$")).columns)
y_cols = list(compared_df.filter(regex=("y$")).columns)
#x_cols.extends(y_cols)
```

In [55]:

```
#updating_df = compared_df[(compared_df['Unit_x'] == compared_df['Unit_y']) | (compared_df['Value__x'] == compared_df['Value__y'])]
updating_df = compared_df.dropna(axis=0, how='any', thresh=None, subset=None, inplace=False)
updating_df
```

Out[55]:

	Investment_Name	Metric	Order_	Period	As_of_Date	Unit_x	Value__x	Unit_y	Valu

In [56]:

```
#updating_df=updating_df.iloc[:,[0,1,2,3,4,7,8]]
updating_df=updating_df.iloc[:,[0,1,2,3,4,7,8]]
updating_df
```

Out[56]:

	Investment_Name	Metric	Order_	Period	As_of_Date	Unit_y	Value__y

In [57]:

```
if len(updating_df) == 0:
    pass
else:
    current_date = datetime.now().strftime("%Y-%m-%d %H:%M:%S")
    for index,row in updating_df.iterrows():
        print(index)
        print(row)
        update_query = "update Canada set Unit='{0}', Value_='{1}', Update_Dt='{2}' whe
re Investment_Name='{3}' and Metric='{4}' and Order_='{5}'  and Period='{6}' and As_of_
Date ='{7}'".format(row[5],row[6],current_date,row[0],row[1],row[2],row[3],row[4])
        print(update_query)
        engine=dbConnection()
        cursor = engine.raw_connection().cursor()
        cursor.execute(update_query)
        cursor.commit()
        print('updated successfully ')
```

In [58]:

```
#insert_val = compared_df[(compared_df['Unit_x'] != compared_df['Unit_y']) & (compared_
df['Value__x'] != compared_df['Value__y'])]
insert_val = compared_df[compared_df.isna().sum(axis=1)>=1]
insert_val
```

Out[58]:

	Investment_Name	Metric	Order_	Period	As_of_Date	Unit_x	Value__x	Unit_y
0	Canada Guaranty	Equity Val	3	Entry	2010-12-31	NaN	NaN	1000000

In [59]:

```
#insert_df = insert_df.iloc[:,[0,1,2,3,4,7,8]]
insert_df = insert_val.loc[:,['Investment_Name', 'Metric', 'Order_', 'Period', 'As_of_D
ate','Unit_y', 'Value__y']]
insert_df
```

Out[59]:

	Investment_Name	Metric	Order_	Period	As_of_Date	Unit_y	Value__y
0	Canada Guaranty	Equity Val	3	Entry	2010-12-31	1000000	1.022000e+08

In [60]:

```
#insert_df.rename(columns={"Investment_Name":"Investment_Name","Metric":"Metric","Orde
r":"Order_","Period":"Period","As_of_Date":"As_of_Date","Unit_y":"Unit","Value__y":"Val
ue_"}, inplace=True)
insert_df.rename(columns=dict(zip(list(insert_df.columns), new_col_names)), inplace=Tru
e)
insert_df
```

Out[60]:

	Investment_Name	Metric	Order_	Period	As_of_Date	Unit	Value_
0	Canada Guaranty	Equity Val	3	Entry	2010-12-31	1000000	1.022000e+08

In [61]:

```
insert_df['Create_Dt'] =  datetime.now().strftime("%Y-%m-%d %H:%M:%S")
insert_df['Update_Dt'] =  datetime.now().strftime("%Y-%m-%d %H:%M:%S")
insert_df
```

Out[61]:

	Investment_Name	Metric	Order_	Period	As_of_Date	Unit	Value_	Cre
0	Canada Guaranty	Equity Val	3	Entry	2010-12-31	1000000	1.022000e+08	202 01 12:5

In [62]:

```
insert_df.info()
```

```
<class 'pandas.core.frame.DataFrame'>
Int64Index: 1 entries, 0 to 0
Data columns (total 9 columns):
 #   Column           Non-Null Count  Dtype
---  ------           --------------  -----
 0   Investment_Name  1 non-null      object
 1   Metric           1 non-null      object
 2   Order_           1 non-null      int64
 3   Period           1 non-null      object
 4   As_of_Date       1 non-null      object
 5   Unit             1 non-null      int64
 6   Value_           1 non-null      object
 7   Create_Dt        1 non-null      object
 8   Update_Dt        1 non-null      object
dtypes: int64(2), object(7)
memory usage: 80.0+ bytes
```

In [240]:

```
if len(insert_df)==0:
        pass
else:
    insert_df.to_sql('Canada', con=con, schema=None, if_exists='append', index=False, i
ndex_label=None, chunksize=None, dtype=None, method=None)
    print('Inserted data successfuly')
```

```
Inserted data successfuly
```

Stop

In [73]:

```
server = 'PAVAN'
dbname = 'company'

eng = create_engine("mssql+pyodbc://"+server+"/"+dbname+"?driver=SQL+Server")
con = eng.connect()

after_updated = pd.read_sql("select * from Canada", con)
after_updated
```

Out[73]:

	Investment_Name	Metric	Order_	Period	As_of_Date	Unit	Value_	Cr
0	Canada Guaranty	Equity Val	3	Actual	2019-12-31	1000000	1.973737e+09	Na
1	Canada Guaranty	Equity Val	3	Actual	2020-12-31	1000000	2.542144e+09	Na
2	Canada Guaranty	Equity Val	3	Actual	2021-06-30	1000000	2.610065e+09	Na
3	Canada Guaranty	Equity Val	3	Actual	2021-12-31	1000000	3.307770e+09	Na
4	Canada Guaranty	Equity Val	3	Actual	2022-12-31	1000000	3.404744e+09	Na
5	Canada Guaranty	Equity Val	3	Entry	2010-04-01	1000000	1.022000e+08	Na
6	Canada Guaranty	Equity Val	3	F1	2021-12-31	1000000	2.815691e+09	Na
7	Canada Guaranty	Equity Val	3	F2	2021-12-31	1000000	2.818039e+09	Na
8	Canada Guaranty	Equity Val	3	F3	2022-12-31	1000000	3.081153e+09	Na
9	Canada Guaranty	Equity Val	3	F4	2022-12-31	1000000	3.404744e+09	Na
10	Canada Guaranty	Equity Val	3	Entry	2010-12-31	1000000	1.022000e+08	20 30 21

In [231]:

```
#new_df1 = df1.set_index(['Investment_Name', 'Metric', 'Order_', 'Period', 'As_of_Dat
e'])
```

In [232]:

```
# new_df = new_df1.merge(db_df2, indicator=True, how="left")[lambda x: x._merge=='left_
only'].drop('_merge',1)
# new_df
```

In [233]:

```
data1 = {'Investment_Name':['Canada Guaranty','Canada Guaranty','Canada Guaranty','Cana
da Guaranty','Canada Guaranty','Canada Guaranty','Canada Guaranty'],
        'Metric':['Equity Val','Equity Val','Equity Val','Equity Val','Equity Val','Equ
ity Val','Equity Val'],
        'Order_':[3,3,3,3,3,3,3],
        'Period':['Actual','Actual','Actual','Entry','Entry','F3','F4'],
        'As_of_Date':['2020-12-31','2021-12-31','2022-12-31','2010-12-31','2023-06-2
3','2022-12-31','2022-12-31'],
        'Unit':[1000000,1000000,1000000,1000000,1000000,1000000,1000000],
        'Value_':["2.542144e+09", "3.307770e+09", "3.404744e+09", "1.022000e+08", "5.00
0000e+09", "3.081153e+09", "3.404744e+09"]}
df1 = pd.DataFrame(data1)
df1
```

Out[233]:

	Investment_Name	Metric	Order_	Period	As_of_Date	Unit	Value_
0	Canada Guaranty	Equity Val	3	Actual	2020-12-31	1000000	2.542144e+09
1	Canada Guaranty	Equity Val	3	Actual	2021-12-31	1000000	3.307770e+09
2	Canada Guaranty	Equity Val	3	Actual	2022-12-31	1000000	3.404744e+09
3	Canada Guaranty	Equity Val	3	Entry	2010-12-31	1000000	1.022000e+08
4	Canada Guaranty	Equity Val	3	Entry	2023-06-23	1000000	5.000000e+09
5	Canada Guaranty	Equity Val	3	F3	2022-12-31	1000000	3.081153e+09
6	Canada Guaranty	Equity Val	3	F4	2022-12-31	1000000	3.404744e+09

In [74]:

```
import os
import getpass
#from passwordretrieval import CyberArk
```

In [76]:

```
if os.name == 'nt':
    user = getpass.getuser()
    print(user)
else:
    os.system('kinit -k -t /opt/kerberos/svc_analytics')
    #user =
```

```
LENOVO
```

In [77]:

```
def getSecret():

    if os.name == 'nt':
        #from requests_negotiate_sspi import HttpNegotiateAuth
        #authentication = HttpNegotiateAuth()
        principal = getpass.getuser() + "@OTPP.COM"
    else:
        #from requests_kerberos import HTTPKerberosAuth,REQUIRED,OPTIONAL
        #authentication = HTTPKerberosAuth(force_preemptive=True,mutual_authentication=
OPTIONAL)
        os.system("kinit -k -t /opt/kerberos/svc_analytics_dev.keytab svc_analytics_dev
@OTPP.COM")
        principal = svc_account_name
```