Banca d'Italia

```
import requests, pandas as pd, os, sqlalchemy, sqlite3, sys, duckdb
from io import BytesIO
from sqlalchemy import create_engine

sqlite  = create_engine('sqlite:///D:/files/Bankit.sqlite')
```

ddb = duckdb.connect('D:/Bankit.duckdb') ddb.close()

```
# Banche e moneta: serie nazionali #BAM mese https://www.bancaditalia.it/pubblicazioni/moneta
STAMEN =['AGGM0100','AGGM0200','AGGM0300','AGGM0400','AGGM0500','ATECO100','BMON0100','BSIBO
         'BSIO0200','CARB0100','CARB0200','CARB0300','CE00100','MID0100','MIR0100','MIR0200'
# Banche e istituzioni finanziarie: finanziamenti e raccolta per settori e territori  fine t
STAFINRA = ['TDB10290','TDB10221','TDB10224','TDB10226','TDB10295','TDB20224','TDB20226','TD
    'TFR20269','TFR20267','TFR40082','TFR40087','TFR40500','TFR40300','TFR40400','TFR30980',
    'TFR10163','TFR10269','TFR10267','TFR40100','TFR10288','TFR10289','TFR20236']

# Banche e istituzioni finanziarie: CONDIZIONI e RISCHIOSITA' del credito per settori e terri
STACORIS = ['TRI30266','TRI30267','TRI30265','TRI30271','TRI30601','TRI30602','TRI30603','TRI
            'TRI30631','TRI30632','TRI30634','TRI30635','TRI30636','TRI30431','TRI30446','TRI
            'TRI30190','TRI30136','TRI30166','TRI30871','TRI30881','TRI30890','TRI30900','TRI
            'TRI30269','TRI30206','TRI30031','TRI30231','TRI30226','TRI30033','TRI30211','TRI
# 'TRI30633',
AdHoc = ['TRI30634','TRI30635','TRI30636','TRI30431','TRI30446','TRI30466','TRI30476','TRI303
            'TRI30190','TRI30136','TRI30166','TRI30871','TRI30881','TRI30890','TRI30900','TRI
            'TRI30269','TRI30206','TRI30031','TRI30231','TRI30226','TRI30033','TRI30211','TRI

STAATER = ['TDB20207','TDB20212','TDB20220','TDB10222','TDB10225','TDB10227','TDB20210','TDB2
tabelle = ['TDB20207','TDB10219','TDB10224','TDB10226','TDB10232','TDB10295','TDB20212','TDB2
           'TFR20281','TFR30309','TFR30315','TFR40100','TRI30021','TRI30171','TRI30211','TRI3
singola = ['TFR40020']
vecchie = ['TDB10194','TDB10255']
```

aggiornamento periodico tabelle

```
def carica_dati_in_sql(tabella):
    file_url = f"https://a2a.bancaditalia.it/infostat/dataservices/export/IT/CSV/DATA/CUBE/B
    result = requests.get(file_url)
    date_column = ['DATA_OSS']
    df = pd.read_csv(BytesIO(result.content),
                compression='zip',
                header=0,
                sep=';',
                quotechar='"',
                encoding='utf-8',
                decimal=',',  # Gestisce numeri come "-5,8"
                dtype={'ENTE_SEGN': 'str', 'FENEC': 'str', 'VALORE': 'float64', 'LOC_SPORT'
                parse_dates=date_column,
                dayfirst=False)
    dtypes = {
        "DATA_OSS": sqlalchemy.types.DATE(),
        "DESINV": sqlalchemy.types.INTEGER(),
        "DURORI": sqlalchemy.types.INTEGER(),
        "TIPTASSO": sqlalchemy.types.INTEGER(),
        "VALORE": sqlalchemy.types.INTEGER(),
        "CLASSE_ACCORD":sqlalchemy.types.TEXT()
    }
    DataMax =df['DATA_OSS'].max()
    print(f"  ok tabella: {tabella}...DataMax: {DataMax}")
    df.to_sql(tabella, sqlite, if_exists='replace', index=False, dtype=dtypes)
    existing_tables = [row[0] for row in ddb.execute("SHOW TABLES").fetchall()]
    if tabella in existing_tables:
        ddb.execute(f"DROP TABLE {tabella}")
    ddb.execute(f"CREATE TABLE {tabella} AS SELECT * FROM df LIMIT 0")  # Crea una tabella v
    ddb.execute(f"INSERT INTO {tabella} SELECT * FROM df")
    return df
```

```
for tabella in STAFINRA:  # STACORIS
    df = carica_dati_in_sql(tabella)
```

```
  ok tabella: TDB10290...DataMax: 2025-04-30 00:00:00
  ok tabella: TDB10221...DataMax: 2025-04-30 00:00:00
  ok tabella: TDB10224...DataMax: 2025-04-30 00:00:00
  ok tabella: TDB10226...DataMax: 2025-04-30 00:00:00
```

ok tabella: TDB10295...DataMax: 2025-04-30 00:00:00
ok tabella: TDB20224...DataMax: 2025-04-30 00:00:00
ok tabella: TDB20226...DataMax: 2025-04-30 00:00:00
ok tabella: TDB20290...DataMax: 2025-04-30 00:00:00
ok tabella: TDB20295...DataMax: 2025-04-30 00:00:00
ok tabella: TFR40020...DataMax: 2025-03-31 00:00:00
ok tabella: TFR20232...DataMax: 2025-03-31 00:00:00
ok tabella: TFR20255...DataMax: 2025-03-31 00:00:00
ok tabella: TFR20231...DataMax: 2025-03-31 00:00:00
ok tabella: TFR30274...DataMax: 2025-03-31 00:00:00
ok tabella: TFR10425...DataMax: 2025-03-31 00:00:00
ok tabella: TFR10435...DataMax: 2025-03-31 00:00:00
ok tabella: TFR20281...DataMax: 2025-03-31 00:00:00
ok tabella: TFR10254...DataMax: 2025-03-31 00:00:00
ok tabella: TFR30309...DataMax: 2025-03-31 00:00:00
ok tabella: TFR30315...DataMax: 2025-03-31 00:00:00
ok tabella: TFR20163...DataMax: 2025-03-31 00:00:00
ok tabella: TFR20269...DataMax: 2025-03-31 00:00:00
ok tabella: TFR20267...DataMax: 2025-03-31 00:00:00
ok tabella: TFR40082...DataMax: 2025-03-31 00:00:00
ok tabella: TFR40087...DataMax: 2025-03-31 00:00:00
ok tabella: TFR40500...DataMax: 2025-03-31 00:00:00
ok tabella: TFR40300...DataMax: 2025-03-31 00:00:00
ok tabella: TFR40400...DataMax: 2025-03-31 00:00:00
ok tabella: TFR30980...DataMax: 2025-03-31 00:00:00
ok tabella: TFR30970...DataMax: 2025-03-31 00:00:00
ok tabella: TFR10194...DataMax: 2024-12-31 00:00:00
ok tabella: TFR10286...DataMax: 2024-12-31 00:00:00
ok tabella: TFR10241...DataMax: 2024-12-31 00:00:00
ok tabella: TFR10232...DataMax: 2025-03-31 00:00:00
ok tabella: TFR10255...DataMax: 2025-03-31 00:00:00
ok tabella: TFR10281...DataMax: 2025-03-31 00:00:00
ok tabella: TFR10236...DataMax: 2025-03-31 00:00:00
ok tabella: TFR10420...DataMax: 2025-03-31 00:00:00
ok tabella: TFR10430...DataMax: 2025-03-31 00:00:00
ok tabella: TFR10460...DataMax: 2025-03-31 00:00:00
ok tabella: TFR10194...DataMax: 2024-12-31 00:00:00
ok tabella: TFR10283...DataMax: 2024-12-31 00:00:00
ok tabella: TFR10287...DataMax: 2024-12-31 00:00:00
ok tabella: TFR10163...DataMax: 2025-03-31 00:00:00
ok tabella: TFR10269...DataMax: 2025-03-31 00:00:00
ok tabella: TFR10267...DataMax: 2025-03-31 00:00:00
ok tabella: TFR40100...DataMax: 2025-03-31 00:00:00

```
  ok tabella: TFR10288...DataMax: 2025-03-31 00:00:00
  ok tabella: TFR10289...DataMax: 2025-03-31 00:00:00
  ok tabella: TFR20236...DataMax: 2025-03-31 00:00:00
```

```
sqlite.close()
```

```
AttributeError: 'Engine' object has no attribute 'close'
---------------------------------------------------------------------------
AttributeError                            Traceback (most recent call last)
Cell In[10], line 1
----> 1 sqlite.close()
AttributeError: 'Engine' object has no attribute 'close'
```

inquiry su tabelle/viste esistenti

```
ddb.execute(f"SELECT * from V_TDB10224_stafinra where LOC_CTP = 'ITG2E' order by DATA_OSS des
```

```
ddb.execute(f"SELECT LOC_CTP, AREA from V_TDB10224_stafinra where LOC_CTP like 'ITG2%' group
```

aggiornamento tabella

```
query_create = f""" UPDATE stafinra
SET Descrizione = CASE Elemento
    WHEN 'ITC2F' THEN 'Cagliari'
    ELSE Descrizione
END
WHERE Elemento IN ('ITC2F');"""
```

creazione una tantum di viste

```
query_create = f""" UPDATE stafinra
SET Descrizione = CASE Elemento
    WHEN 'ITC20' THEN 'Liguria'
    WHEN 'ITG2D' THEN 'Sardegna Sud'
    WHEN 'ITG2E' THEN 'Sardegna Cagliari'
    WHEN 'ITG2F' THEN 'Sardegna Nord'
```

```
    WHEN 'ITG2G' THEN 'Sardegna Est'
    ELSE Descrizione
END
WHERE Elemento IN ('ITC20', 'ITG2D', 'ITG2E', 'ITG2F', 'ITG2G');"""
```

```
<duckdb.duckdb.DuckDBPyConnection at 0x1964d1f18b0>
```

```
# script valido per 'TDB10224','TFR20232'
tabella = 'TDB10224'
dominio = 'stafinra'
vista = f"V_{tabella}_{dominio}"  # Qui usiamo f-string per creare il nome dinamico della vis
query_drop = f"DROP VIEW IF EXISTS {vista};"
ddb.execute(query_drop)
query_create = f""" CREATE VIEW {vista} AS SELECT  d.DATA_OSS,
        d.ENTE_SEGN, s1.Descrizione AS SEGNALANTE,
        d.LOC_CTP,   s2.Descrizione AS AREA,
        d.SET_CTP,   s3.Descrizione AS TARGET,
        d.FENEC,     s4.Descrizione AS FENOMENO,
        d.ATECO_CTP, s5.Descrizione AS ATECO,
        d.VALORE
FROM {tabella} d
LEFT JOIN {dominio} s1 ON d.ENTE_SEGN = s1.Elemento
LEFT JOIN {dominio} s2 ON d.LOC_CTP = s2.Elemento
LEFT JOIN {dominio} s3 ON d.SET_CTP = s3.Elemento
LEFT JOIN {dominio} s4 ON d.FENEC = s4.Elemento
LEFT JOIN {dominio} s5 ON d.ATECO_CTP = s5.Elemento; """
```

```
ddb.execute(query_create)
df = ddb.execute(f"SELECT * FROM {vista} limit 5").fetchdf()
df
```

```
CatalogException: Catalog Error: View with name "V_TDB10224_stafinra" already exists!
---------------------------------------------------------------------------
CatalogException                          Traceback (most recent call last)
Cell In[42], line 1
----> 1 ddb.execute(query_create)
      2 df = ddb.execute(f"SELECT * FROM {vista}limit 5").fetchdf()
      3 df
CatalogException: Catalog Error: View with name "V_TDB10224_stafinra" already exists!
```

```
# script valido per 'TDB10226','TDB10295'
tabella = 'TFR20232'
dominio = 'stafinra'
vista = f"V_{tabella}_{dominio}"  # Qui usiamo f-string per creare il nome dinamico della vis
query_drop = f"DROP VIEW IF EXISTS {vista};"
ddb.execute(query_drop)
query_create = f""" CREATE VIEW {vista} AS SELECT  d.DATA_OSS,
        d.ENTE_SEGN, s1.Descrizione AS SEGNALANTE,
        d.LOC_CTP,   s2.Descrizione AS AREA,
        d.SET_CTP,   s3.Descrizione AS TARGET,
        d.FENEC,     s4.Descrizione AS FENOMENO,
        d.VALORE
FROM {tabella} d
LEFT JOIN {dominio} s1 ON d.ENTE_SEGN = s1.Elemento
LEFT JOIN {dominio} s2 ON d.LOC_CTP = s2.Elemento
LEFT JOIN {dominio} s3 ON d.SET_CTP = s3.Elemento
LEFT JOIN {dominio} s4 ON d.FENEC = s4.Elemento; """
```

```
# script valido per 'TDB10224','TDB10226','TDB10295'
tabella = 'TFR20232'
dominio = 'stafinra'
vista = f"V_{tabella}_{dominio}"  # Qui usiamo f-string per creare il nome dinamico della vis
query_drop = f"DROP VIEW IF EXISTS {vista};"
ddb.execute(query_drop)
query_create = f"""CREATE VIEW {vista} AS SELECT d.DATA_OSS,
    d.ENTE_SEGN,    s1.Descrizione AS SEGNALANTE,
    d.LOC_SPORT,    s2.Descrizione AS AREA,
    d.FENEC,        s3.Descrizione AS FENOMENO,
    d.VALORE
FROM {tabella} d
LEFT JOIN {dominio} s1 ON d.ENTE_SEGN = s1.Elemento
LEFT JOIN {dominio} s2 ON d.LOC_SPORT = s2.Elemento
LEFT JOIN {dominio} s3 ON d.FENEC     = s3.Elemento;"""
```

```
# script valido per 'TRI30156',
tabella = 'TRI30156'
dominio = 'stacoris'
vista = f"V_{tabella}_{dominio}"  # Qui usiamo f-string per creare il nome dinamico della vis
query_drop = f"DROP VIEW IF EXISTS {vista};"
ddb.execute(query_drop)

query_create = f"""CREATE VIEW {vista} AS SELECT d.DATA_OSS,
```

```
    d.ENTE_SEGN,     s1.Descrizione AS SEGNALANTE,
    d.SET_CTP,     s2.Descrizione AS AREA,
    d.FENEC,     s3.Descrizione AS FENOMENO,
    d.ATECO_CTP,     s4.Descrizione AS SETTORE_ATECO,
    d.CLASSE_ACCORD,     s5.Descrizione AS CLASSE_ACCORDO,
    d.SEDELEG_SOGG,     s6.Descrizione AS SEDE_LEGALE,
    d.VALORE
FROM {tabella} d
LEFT JOIN {dominio} s1 ON d.ENTE_SEGN = s1.Elemento
LEFT JOIN {dominio} s2 ON d.SET_CTP   = s2.Elemento
LEFT JOIN {dominio} s3 ON d.FENEC     = s3.Elemento
LEFT JOIN {dominio} s4 ON d.ATECO_CTP = s4.Elemento
LEFT JOIN {dominio} s5 ON CAST(d.CLASSE_ACCORD AS VARCHAR) = s5.Elemento
LEFT JOIN {dominio} s6 ON d.SEDELEG_SOGG = s6.Elemento; """
ddb.execute(query_create)
print(f"  Vista '{vista}' ricreata con successo!")
```

```
  Vista 'V_TRI30156_stacoris' ricreata con successo!
```

```
# vedere le viste memorizzate
query_create = f"""SELECT table_name, table_type
FROM information_schema.tables
WHERE table_schema = 'main' AND table_type = 'VIEW';"""
ddb.execute(query_create).fetchdf()
```

	table_name	table_type
0	V_TDB10224_stafinra	VIEW
1	V_TDB10226_stafinra	VIEW
2	V_TDB10295_stafinra	VIEW
3	V_TFR10194_stafinra	VIEW
4	V_TFR10255_stafinra	VIEW
5	V_TFR20232_stafinra	VIEW
6	V_TRI30156_stacoris	VIEW

```
# vedere le colonne
query_create = f"""DESCRIBE {vista};"""
ddb.execute(query_create).fetchdf()
```

	column_name	column_type	null	key	default	extra
0	DATA_OSS	TIMESTAMP_NS	YES	None	None	None
1	ENTE_SEGN	VARCHAR	YES	None	None	None
2	SEGNALANTE	VARCHAR	YES	None	None	None
3	LOC_CTP	VARCHAR	YES	None	None	None
4	AREA	VARCHAR	YES	None	None	None
5	SET_CTP	VARCHAR	YES	None	None	None
6	TARGET	VARCHAR	YES	None	None	None
7	FENEC	VARCHAR	YES	None	None	None
8	FENOMENO	VARCHAR	YES	None	None	None
9	ATECO_CTP	VARCHAR	YES	None	None	None
10	ATECO	VARCHAR	YES	None	None	None
11	VALORE	DOUBLE	YES	None	None	None

set	20250331
STAATER	x
STAFINRA	
STACORIS	

```
len(STAFINRA)
```

```
50
```

```
ddb.close()
```

```
query = """SELECT * FROM TDB10295 limit 10;"""
ddb.execute(query).fetchdf()
```

```
query = """SELECT * FROM STACORIS limit 10;"""
ddb.execute(query).fetchdf()
```

legend, domain, structure

```
# merge di file informativi
def merge_csv_files(folder_path, output_file, file):
    all_files = [f for f in os.listdir(folder_path) if f.endswith(".csv") and f.startswith(fi
```

```
    df_list = []

    for file_name in all_files:
        file_path = os.path.join(folder_path, file_name)
        df = pd.read_csv(file_path, delimiter=';', dtype=str)  # Legge i CSV come stringhe
        df['File_Origine'] = file_name  # Aggiunge la colonna con il nome del file
        df_list.append(df)

    merged_df = pd.concat(df_list, ignore_index=True)
    merged_df.to_csv(output_file, index=False, sep=';')
    print(f"File unito salvato in: {output_file}")
```

```
file = 'LEGEND'  # Cambiare
```

```
folder_path = f"D:\\files\\csv\\Bankit"
output_file = f"D:\\files\\csv\\Bankit\\{file}.csv"
merge_csv_files(folder_path, output_file, file)
```

```
df = pd.read_csv(file+'.csv', sep =';')
df.to_sql(file, sqlite, if_exists='replace')
```

1203

```
df = pd.read_csv('io_tavole.tsv', sep ='\t')
df.to_sql('tabelle', sqlite, if_exists='replace')
```

89

BAM

```
tabella = 'MIR0300' # ATECO100,BSIB0600,BSIB0700,BSIB1010,MIR0100
file = f'https://a2a.bancaditalia.it/infostat/dataservices/export/IT/CSV/DATA/CUBE/BANKITALI
result = requests.get(file)
date_column = ['DATA_OSS']
data = pd.read_csv(BytesIO(result.content),compression='zip', header=0, sep=';',
                   quotechar='"', encoding='utf-8',parse_dates=date_column, dayfirst=False)
df = data.melt(id_vars=['DATA_OSS'], var_name='variabile', value_name='valore')
df.to_sql(tabella, sqlite, if_exists='replace')
```

2640

```
data.columns = data.columns.str.replace(r'BAM_ATECO.M.1070001.52000700.101.IT.', '')
data.dropna()
melted_df = data.melt(id_vars=['DATA_OSS'], var_name='Elemento', value_name='VALORE')

melted_df[['target','settore']] = melted_df['Elemento'].str.split('.', n=1, expand=True)
melted_df['VALORE'] = melted_df['VALORE']*1000000
```

```
dtypes = {"DIVISA1": sqlalchemy.types.INTEGER(), "DURORI": sqlalchemy.types.INTEGER(),"LOC_SE
          "VALORE": sqlalchemy.types.INTEGER()}
melted_df[['DATA_OSS', 'target', 'settore', 'VALORE']].to_sql('ATECO100', sqlite, if_exists=
```

21372

BSIB0800 - Prestiti ai residenti in Italia, per durata e tipologia

verifica punti di rottura

```
SBI33 = data.query('SET_CTP == "SBI33"')[['DATA_OSS','LOC_CTP','VALORE']]
# SBI33['DATA_OSS'] = pd.to_datetime(SBI33['DATA_OSS'])
SBI33.set_index(SBI33['DATA_OSS'], inplace = True)
ts = SBI33['VALORE']
plt.plot(ts, linewidth=3, color='red')
plt.title('')
plt.grid()
plt.show()
```

```
import ruptures as rpt
model = rpt.Dynp(model="l1")
model.fit(y)
result = model.predict()
rpt.display(SBI33, result)
plt.show()
```

b = df[‘DATA_OSS’].max() df = df.query(‘DATA_OSS == @b’)

data = data[data[‘DATA_OSS’] == data[‘DATA_OSS’].max()] data[‘DATA_OSS’] = data[‘DATA_OSS’].dt.date data = data[data[‘LOC_CTP’].isin(nuts1)]

data = pd.merge(data, stamen, how = ‘left’, left_on=‘ENTE_SEGN’, right_on=‘Elemento’).drop(columns=[‘E
‘segnalante’}) data = pd.merge(data, stamen, how = ‘left’, left_on=‘LOC_CTP’,
right_on=‘Elemento’).drop(columns=[‘index’,‘Dominio’, ‘Elemento’]).rename(columns={‘Descrizione’:
‘area’}) data = pd.merge(data, stamen, how = ‘left’, left_on=‘SET_CTP’, right_on=‘Elemento’).drop(columns
‘Elemento’]).rename(columns={‘Descrizione’: ‘target’}) data = pd.merge(data, stamen, how
= ‘left’, left_on = ‘FENEC’, right_on = ‘Elemento’).rename(columns={‘Descrizione’:
‘FENOMENO’})[[‘DATA_OSS’,‘SEGNALANTE’, ‘FENOMENO’,‘LOC_CTP’,‘SET_CTP’,
‘VALORE’]]

data = pd.merge(data, nuts, how=‘left’, left_on=‘SEDELEG_SOGG’, right_on=‘cod’) data
= pd.merge(data, EntiSegnalanti, how=‘left’, left_on=‘ENTE_SEGN’, right_on=‘ENTE_SEGN’)
# EntiSegnalanti = pd.DataFrame(data, columns=[‘ENTE_SEGN’, ‘Decod_Segnalante’])
data = pd.merge(data, FenomenoEconomico, how=‘left’, left_on=‘FENEC’, right_on=‘FENEC’)
# FenomenoEconomico = pd.DataFrame(data, columns=[‘FENEC’, ‘Decod_FenEco’]) data
= pd.merge(data, controparte, how=‘left’, left_on=‘SET_CTP’, right_on=‘SET_CTP’) #
controparte = pd.DataFrame(data, columns=[‘SET_CTP’, ‘controp’]) data = pd.merge(data,
attività, how=‘left’, left_on=‘ATECO_CTP’, right_on=‘ATECO_CTP’) # attività =
pd.DataFrame(data, columns=[‘ATECO_CTP’, ‘ATECO’])’

‘IT’
https://it.wikipedia.org/wiki/Nomenclatura_delle_unit%C3%A0_territoriali_pe

Nord Ovest = ‘ITC’, ‘ITC1’,‘ITC2’,‘ITC3’,‘ITC4’

NordEst = ‘ITH’,‘ITH3’,‘ITH4’,‘ITH5’

Centro = ‘ITI’,‘ITI1’,‘ITI2’,‘ITI3’,‘ITI4’

sud = ‘ITF’,‘ITF1’,‘ITF2’,‘ITF3’, ‘ITF4’,‘ITF5’,‘ITF6’

isole = ‘ITG’,‘ITG1’, ‘ITG2’

TRENTINO-ALTO ADIGE = ‘ITHBI12’

```
ddb.execute("DESCRIBE TDB10295").fetchdf()
```

	column_name	column_type	null	key	default	extra
0	DATA_OSS	TIMESTAMP_NS	YES	None	None	None
1	ENTE_SEGN	VARCHAR	YES	None	None	None
2	FENEC	VARCHAR	YES	None	None	None
3	LOC_CTP	VARCHAR	YES	None	None	None
4	SET_CTP	VARCHAR	YES	None	None	None
5	VALORE	DOUBLE	YES	None	None	None
6	STATUS	DOUBLE	YES	None	None	None

```
import duckdb

# Connessione al database DuckDB
ddb = duckdb.connect('D:/Bankit.duckdb')

# Creazione della vista
query = """
CREATE OR REPLACE VIEW V_TDB10295 AS
SELECT
*
FROM data d
LEFT JOIN stamen s1 ON d.ENTE_SEGN = s1.Elemento
LEFT JOIN stamen s2 ON d.LOC_CTP = s2.Elemento
LEFT JOIN stamen s3 ON d.SET_CTP = s3.Elemento
LEFT JOIN stamen s4 ON d.FENEC = s4.Elemento;
"""

# Esegui la query per creare la vista
ddb.execute(query)

# Verifica che la vista sia stata creata correttamente
print(ddb.execute("DESCRIBE data_enriched").fetchdf())  # Mostra la struttura della vista
```