```
import sqlite3
import pandas as pd

# Percorso al tuo file .sqlite
db_path = "D:/files/Bankit.sqlite"
pd.set_option('display.max_rows', None)
```

```
valore_cercato = 'deterioramento'

conn = sqlite3.connect(db_path)
cursor = conn.cursor()

# Otteniamo tutte le tabelle
cursor.execute("SELECT name FROM sqlite_master WHERE type='table'")
tabelle = [row[0] for row in cursor.fetchall()]

trovate = []

for tabella in tabelle:
    try:
        cursor.execute(f"PRAGMA table_info('{tabella}')")
        colonne = [r[1] for r in cursor.fetchall()]

        # Escludiamo 'DATA_OSS' e 'VALORE'
        colonne = [col for col in colonne if col.upper() not in ('DATA_OSS', 'VALORE', 'STATU

        for col in colonne:
            try:
                query = f'SELECT 1 FROM "{tabella}" WHERE "{col}" = ? LIMIT 1'
                cursor.execute(query, (valore_cercato,))
                if cursor.fetchone():
                    trovate.append((tabella, col))
                    break  # salta alla prossima tabella
            except sqlite3.OperationalError:
                continue
    except Exception as e:
        print(f"Errore con la tabella {tabella}: {e}")

# Mapping tabella -> descrizione, usando solo la parte prima dell'underscore
descrizioni = {}
try:
    cursor.execute("SELECT tabella, descrizione FROM tabelle")
```

```
    for tabella_completa, descrizione in cursor.fetchall():
        base_tabella = tabella_completa.split('_')[0].upper()
        if base_tabella not in descrizioni:
            descrizioni[base_tabella] = (tabella_completa, descrizione)
except Exception as e:
    print(f"Errore leggendo la tabella 'tabelle': {e}")

conn.close()

# Stampiamo i risultati con descrizione
for tabella, colonna in sorted(trovate):
    match = descrizioni.get(tabella.upper())
    if match:
        tabella_completa, desc = match
        print(f"  Trovato in: '{tabella}' ({desc}, da '{tabella_completa}'), colonna '{colonn
    else:
        print(f"  Trovato in: '{tabella}' ((descrizione non trovata)), colonna '{colonna}'")
```

Errore con la tabella TRI30633: database disk image is malformed

```
valore_cercato = 'deterioramento'

conn = sqlite3.connect(db_path)
cursor = conn.cursor()

# Otteniamo tutte le tabelle
cursor.execute("SELECT name FROM sqlite_master WHERE type='table'")
tabelle = [row[0] for row in cursor.fetchall()]

trovate = []

for tabella in tabelle:
    try:
        cursor.execute(f"PRAGMA table_info('{tabella}')")
        colonne = [r[1] for r in cursor.fetchall()]

        for col in colonne:
            try:
                query = f'SELECT 1 FROM "{tabella}" WHERE "{col}" = ? LIMIT 1'
                cursor.execute(query, (valore_cercato,))
                if cursor.fetchone():
```

```
                    trovate.append((tabella, col))
                    break  # salta alla prossima tabella
            except sqlite3.OperationalError:
                continue
    except Exception as e:
        print(f"Errore con la tabella {tabella}: {e}")

# Mapping tabella -> descrizione, usando solo la parte prima dell'underscore
descrizioni = {}
try:
    cursor.execute("SELECT tabella, descrizione FROM tabelle")
    for tabella_completa, descrizione in cursor.fetchall():
        base_tabella = tabella_completa.split('_')[0].upper()
        if base_tabella not in descrizioni:
            descrizioni[base_tabella] = (tabella_completa, descrizione)
except Exception as e:
    print(f"Errore leggendo la tabella 'tabelle': {e}")


conn.close()

# Stampiamo i risultati con descrizione
for tabella, colonna in sorted(trovate):
    match = descrizioni.get(tabella.upper())
    if match:
        tabella_completa, desc = match
        print(f"  Trovato in: '{tabella}' ({desc}, da '{tabella_completa}'), colonna '{colonn
    else:
        print(f"  Trovato in: '{tabella}' ((descrizione non trovata)), colonna '{colonna}'")
```

```
Errore con la tabella TRI30633: database disk image is malformed
```

```
import sqlite3
import pandas as pd

valori_richiesti = ['S11', 'S14BI4']

conn = sqlite3.connect(db_path)
cursor = conn.cursor()

# Ottieni tutte le tabelle
```

```
cursor.execute("SELECT name FROM sqlite_master WHERE type='table'")
tabelle = [row[0] for row in cursor.fetchall()]

risultati = []
for tabella in tabelle:
    riga = {'tabella': tabella}
    try:
        cursor.execute(f'PRAGMA table_info("{tabella}")')
        colonne = [r[1] for r in cursor.fetchall()]

        for valore in valori_richiesti:
            trovato = None
            for col in colonne:
                try:
                    cursor.execute(f'SELECT 1 FROM "{tabella}" WHERE "{col}" = ? LIMIT 1', (v
                    if cursor.fetchone():
                        trovato = col
                        break
                except sqlite3.OperationalError:
                    continue
            riga[valore] = trovato
        risultati.append(riga)
    except Exception as e:
        print(f"Errore con la tabella {tabella}: {e}")

conn.close()

# Converte in DataFrame
df = pd.DataFrame(risultati)

# Conta solo i valori trovati nella colonna 'SET_CTP'
df["conteggio"] = df[valori_richiesti].apply(lambda row: sum(v == "SET_CTP" for v in row), ax

# Filtra solo tabelle con almeno un valore trovato in SET_CTP
df = df[df["conteggio"] > 0]

# Ricarica le descrizioni dalla tabella 'tabelle'
conn = sqlite3.connect(db_path)
cursor = conn.cursor()

mappa_descrizioni = {}
try:
```

```
    cursor.execute("SELECT tabella, descrizione FROM tabelle")
    for tabella_completa, descrizione in cursor.fetchall():
        base = tabella_completa.split("_")[0].upper()
        if base not in mappa_descrizioni:
            mappa_descrizioni[base] = (tabella_completa, descrizione)
except Exception as e:
    print(f"Errore leggendo la tabella 'tabelle': {e}")

conn.close()

# Aggiunge la descrizione basandosi sulla parte base della tabella
df["descrizione"] = df["tabella"].apply(lambda x: mappa_descrizioni.get(x.upper(), ("", "(des

# Riordina le colonne
col_riordinate = ["tabella", "descrizione"] + valori_richiesti + ["conteggio"]
df = df[col_riordinate]

# Ordina per conteggio decrescente
df = df.sort_values(by="conteggio", ascending=False)

# Visualizza
print(df.to_string(index=False))
# df.to_csv("valori_solo_in_SET_CTP.csv", index=False)
```

```
Errore con la tabella TRI30633: database disk image is malformed
 tabella
TDB30156
TRI30603                                                Tasso di deterioramento annuale dei pres
TDC30021
TRI30031                                               Sofferenze (al lordo delle svalutazioni e a
TRI30033                                                       Sofferenze (al lordo delle svaluta
TRI30211                                               Sofferenze (al lordo delle svalutazioni e a
TRI30251                                    Nuove sofferenze (al lordo delle svalutazioni e al net
TRI30271 Sofferenze rettificate (flusso): utilizzato in sofferenza rettificata all'inizio e r
TRI30601                                                Tasso di deterioramento annuale dei pres
TRI30602                                                Tasso di deterioramento annuale dei pres
TRI30604                                                Tasso di deterioramento annuale dei pres
TRI30466                                                                Prestiti (escluse soffe
TRI30605                                                Tasso di deterioramento annuale dei pres
TRI30486                                                            Flusso trimestrale nuovi ing
TRI30496                                                            Flusso trimestrale nuovi ing
TRI30507                                                            Flusso trimestrale nuovi ing
```

```
TRI30516                                                    Flusso trimestrale nuovi ing
TRI30524                                                    Flusso trimestrale nuovi ing
TRI30631                                                       Flusso trimestrale nuovi
TRI30632                                                       Flusso trimestrale nuovi
TRI30156                                                                    Prestit
TRI30021                                                                    Prestit
TRI30446                                                                        Pre
TFR40100
TFR30970                                                                     Dati c
TRI30634                                                       Flusso trimestrale nuovi
TRI30269
TFR20269
TDB10232
TDB10290                                              Depositi (esclusi PCT) - per provin
TDB10295                                              Prestiti (esclusi PCT) - per provin
TDB20290
TDB20295
TFR20232
TFR20281
TFR20163
TRI30171                                                                    Prestit
TRI30268
TFR10241
TFR10288
TFR10281
TRI30267
TFR10287
TFR10163
TFR10269
TRI30231                                         Sofferenze (al lordo delle svalutazioni e a
TFR10232
```

```python
tabella = 'TFR10420'
conn = sqlite3.connect(db_path)
cursor = conn.cursor()

# Corretto uso di PRAGMA
cursor.execute(f"PRAGMA table_info({tabella})")
tutte_le_colonne = [row[1] for row in cursor.fetchall()]

# Colonne da esplorare (escluse quelle tecniche)
colonne_da_esplorare = [col for col in tutte_le_colonne if col not in ("DATA_OSS", "VALORE",
```

```python
# Raccolta dei valori unici
risultati = {}
for col in colonne_da_esplorare:
    try:
        cursor.execute(f'SELECT DISTINCT "{col}" FROM {tabella} ORDER BY 1')
        risultati[col] = [row[0] if row[0] is not None else "" for row in cursor.fetchall()]
    except Exception as e:
        print(f"Errore nella colonna {col}: {e}")

conn.close()

# Conversione in DataFrame e sostituzione dei NaN con stringa vuota
df_unici = pd.DataFrame(dict([(k, pd.Series(v)) for k, v in risultati.items()])).fillna("")

# Opzioni display per Pandas
pd.set_option("display.max_columns", None)
pd.set_option("display.max_rows", None)
pd.set_option("display.max_colwidth", None)
pd.set_option("display.width", None)

print("\nValori unici per ciascuna colonna (escluse DATA_OSS e VALORE):\n")
print(df_unici)
```

```
Valori unici per ciascuna colonna (escluse DATA_OSS e VALORE):

    DESINV DURORI ENTE_SEGN    FENEC  LOC_CTP RESIDENZA1 TIPTASSO
0     10.0   18.0   1070001  1001530  9999999         IT     80.0
1     40.0          1100010                IT              800.0
2     70.0                                ITC            10000.0
3     80.0                               ITC1
4    110.0                               ITC11
5    120.0                               ITC12
6    902.0                               ITC13
7    905.0                               ITC14
8    909.0                               ITC15
9    912.0                               ITC16
10   914.0                               ITC17
11   916.0                               ITC18
12   997.0                                ITC2
13   998.0                               ITC20
```

14	ITC3
15	ITC31
16	ITC32
17	ITC33
18	ITC34
19	ITC4
20	ITC41
21	ITC42
22	ITC43
23	ITC44
24	ITC46
25	ITC47
26	ITC48
27	ITC49
28	ITC4A
29	ITC4B
30	ITC4C
31	ITC4D
32	ITF
33	ITF1
34	ITF11
35	ITF12
36	ITF13
37	ITF14
38	ITF2
39	ITF21
40	ITF22
41	ITF3
42	ITF31
43	ITF32
44	ITF33
45	ITF34
46	ITF35
47	ITF4
48	ITF43
49	ITF44
50	ITF45
51	ITF46
52	ITF47
53	ITF48
54	ITF5
55	ITF51
56	ITF52

57	ITF6
58	ITF61
59	ITF62
60	ITF63
61	ITF64
62	ITF65
63	ITG
64	ITG1
65	ITG11
66	ITG12
67	ITG13
68	ITG14
69	ITG15
70	ITG16
71	ITG17
72	ITG18
73	ITG19
74	ITG2
75	ITG25
76	ITG26
77	ITG27
78	ITG28
79	ITG29
80	ITG2A
81	ITG2B
82	ITG2C
83	ITG2D
84	ITG2E
85	ITG2F
86	ITG2G
87	ITG2H
88	ITH
89	ITH10
90	ITH20
91	ITH3
92	ITH31
93	ITH32
94	ITH33
95	ITH34
96	ITH35
97	ITH36
98	ITH37
99	ITH4

100	ITH41
101	ITH42
102	ITH43
103	ITH44
104	ITH5
105	ITH51
106	ITH52
107	ITH53
108	ITH54
109	ITH55
110	ITH56
111	ITH57
112	ITH58
113	ITH59
114	ITHBI12
115	ITI
116	ITI1
117	ITI11
118	ITI12
119	ITI13
120	ITI14
121	ITI15
122	ITI16
123	ITI17
124	ITI18
125	ITI19
126	ITI1A
127	ITI2
128	ITI21
129	ITI22
130	ITI3
131	ITI31
132	ITI32
133	ITI33
134	ITI34
135	ITI35
136	ITI4
137	ITI41
138	ITI42
139	ITI43
140	ITI44
141	ITI45

```
tabella = 'TFR10420'
domini = 'STACORIS'
conn = sqlite3.connect(db_path)
cursor = conn.cursor()
mappa_domini = {
    "ATECO_CTP": "ATECO",
    "SEDELEG_SOGG":"TERRITORIO",'LOC_CTP':"TERRITORIO",
    'DIVISA1':'VALUTISO',
    "CLASSE_ACCORD": "CLAGRAND",
    "LIVAFF":"PLURAFF",
    'DURORI':'DURATA',
    "ENTE_SEGN":"AZIENDA",
    "FENEC":"FENOMECON",
    "SET_CTP":"SETTORIST",
    'DESINV':'DESINV'
    # altri domini verranno aggiunti se identificati
}

# Elenco colonne da esplorare
cursor.execute(f"PRAGMA table_info({tabella})")
tutte_le_colonne = [row[1] for row in cursor.fetchall()]
colonne_da_esplorare = [col for col in tutte_le_colonne if col not in ("DATA_OSS", "VALORE",

righe_uniche = []

# Estrazione valori + descrizioni
for col in colonne_da_esplorare:
    dominio = mappa_domini.get(col)
    if dominio is None:
        print(f"  Nessun dominio disponibile per la colonna '{col}', verrà saltata.")
        continue

    try:
        query = f'''
        SELECT DISTINCT '{col}' AS colonna,
                        CAST(t."{col}" AS TEXT) AS codice,
                        COALESCE(d.Descrizione, '') AS descrizione
        FROM {tabella} t

        LEFT JOIN "DOMAIN-STAFINRA-MULTICUBE" d

            ON CAST(t."{col}" AS TEXT) = CAST(d.Elemento AS TEXT)
```

```
            AND d.Dominio = '{dominio}'
        WHERE t."{col}" IS NOT NULL
        ORDER BY codice
        '''
        cursor.execute(query)
        righe_uniche.extend(cursor.fetchall())
    except Exception as e:
        print(f"Errore nella colonna {col}: {e}")

conn.close()

pd.DataFrame(righe_uniche, columns=["Colonna", "Codice", "Descrizione"])
```

```
Nessun dominio disponibile per la colonna 'RESIDENZA1', verrà saltata.
Nessun dominio disponibile per la colonna 'TIPTASSO', verrà saltata.
```

	Colonna	Codice	Descrizione
0	DESINV	10	Investimenti non finanziari: costruzioni - abi...
1	DESINV	110	Altri investimenti: acquisto di beni durevoli ...
2	DESINV	120	Investimenti finanziari
3	DESINV	40	Investimenti non finanziari: costruzioni - ope...
4	DESINV	70	Investimenti: acquisto immobili - abitazioni f...
5	DESINV	80	Investimenti: acquisto immobili - abitazioni d...
6	DESINV	902	Investimenti non finanziari: costruzioni - fab...
7	DESINV	905	Investimenti non finanziari: macchine, attrezz...
8	DESINV	909	Investimenti: acquisto di immobili - altri imm...
9	DESINV	912	Investimenti non finanziari: investimenti in c...
10	DESINV	914	Investimenti: acquisto di immobili diversi da ...
11	DESINV	916	Investimenti: destinazioni diverse da acquisto...
12	DESINV	997	Tutte le finalità
13	DESINV	998	Invest.diversi dai non finanziari: destin. div...
14	DURORI	18	Oltre 1 anno
15	ENTE_SEGN	1070001	Banche e Cassa depositi e prestiti
16	ENTE_SEGN	1100010	Banche
17	FENEC	1001530	Prestiti oltre il breve termine (esclusi PCT e...
18	LOC_CTP	9999999	Non classificabile
19	LOC_CTP	IT	Italia
20	LOC_CTP	ITC	Italia nord-occidentale
21	LOC_CTP	ITC1	Piemonte
22	LOC_CTP	ITC11	Torino

	Colonna	Codice	Descrizione
23	LOC_CTP	ITC12	Vercelli
24	LOC_CTP	ITC13	Biella
25	LOC_CTP	ITC14	Verbano-Cusio-Ossola
26	LOC_CTP	ITC15	Novara
27	LOC_CTP	ITC16	Cuneo
28	LOC_CTP	ITC17	Asti
29	LOC_CTP	ITC18	Alessandria
30	LOC_CTP	ITC2	Valle d'Aosta/Vallée d'Aoste
31	LOC_CTP	ITC20	
32	LOC_CTP	ITC3	Liguria
33	LOC_CTP	ITC31	Imperia
34	LOC_CTP	ITC32	Savona
35	LOC_CTP	ITC33	Genova
36	LOC_CTP	ITC34	La Spezia
37	LOC_CTP	ITC4	Lombardia
38	LOC_CTP	ITC41	Varese
39	LOC_CTP	ITC42	Como
40	LOC_CTP	ITC43	Lecco
41	LOC_CTP	ITC44	Sondrio
42	LOC_CTP	ITC46	Bergamo
43	LOC_CTP	ITC47	Brescia
44	LOC_CTP	ITC48	Pavia
45	LOC_CTP	ITC49	Lodi
46	LOC_CTP	ITC4A	Cremona
47	LOC_CTP	ITC4B	Mantova
48	LOC_CTP	ITC4C	Milano
49	LOC_CTP	ITC4D	Monza e della Brianza
50	LOC_CTP	ITF	Italia meridionale
51	LOC_CTP	ITF1	Abruzzo
52	LOC_CTP	ITF11	L'Aquila
53	LOC_CTP	ITF12	Teramo
54	LOC_CTP	ITF13	Pescara
55	LOC_CTP	ITF14	Chieti
56	LOC_CTP	ITF2	Molise
57	LOC_CTP	ITF21	Isernia
58	LOC_CTP	ITF22	Campobasso
59	LOC_CTP	ITF3	Campania
60	LOC_CTP	ITF31	Caserta
61	LOC_CTP	ITF32	Benevento
62	LOC_CTP	ITF33	Napoli
63	LOC_CTP	ITF34	Avellino

	Colonna	Codice	Descrizione
64	LOC_CTP	ITF35	Salerno
65	LOC_CTP	ITF4	Puglia
66	LOC_CTP	ITF43	Taranto
67	LOC_CTP	ITF44	Brindisi
68	LOC_CTP	ITF45	Lecce
69	LOC_CTP	ITF46	Foggia
70	LOC_CTP	ITF47	Bari
71	LOC_CTP	ITF48	Barletta-Andria-Trani
72	LOC_CTP	ITF5	Basilicata
73	LOC_CTP	ITF51	Potenza
74	LOC_CTP	ITF52	Matera
75	LOC_CTP	ITF6	Calabria
76	LOC_CTP	ITF61	Cosenza
77	LOC_CTP	ITF62	Crotone
78	LOC_CTP	ITF63	Catanzaro
79	LOC_CTP	ITF64	Vibo Valentia
80	LOC_CTP	ITF65	Reggio di Calabria
81	LOC_CTP	ITG	Italia insulare
82	LOC_CTP	ITG1	Sicilia
83	LOC_CTP	ITG11	Trapani
84	LOC_CTP	ITG12	Palermo
85	LOC_CTP	ITG13	Messina
86	LOC_CTP	ITG14	Agrigento
87	LOC_CTP	ITG15	Caltanissetta
88	LOC_CTP	ITG16	Enna
89	LOC_CTP	ITG17	Catania
90	LOC_CTP	ITG18	Ragusa
91	LOC_CTP	ITG19	Siracusa
92	LOC_CTP	ITG2	Sardegna
93	LOC_CTP	ITG25	Sassari
94	LOC_CTP	ITG26	Nuoro
95	LOC_CTP	ITG27	Cagliari
96	LOC_CTP	ITG28	Oristano
97	LOC_CTP	ITG29	Olbia-Tempio
98	LOC_CTP	ITG2A	Ogliastra
99	LOC_CTP	ITG2B	Medio Campidano
100	LOC_CTP	ITG2C	Carbonia-Iglesias
101	LOC_CTP	ITG2D	
102	LOC_CTP	ITG2E	
103	LOC_CTP	ITG2F	
104	LOC_CTP	ITG2G	

	Colonna	Codice	Descrizione
105	LOC_CTP	ITG2H	Sud Sardegna
106	LOC_CTP	ITH	Italia nord-orientale
107	LOC_CTP	ITH10	Bolzano-Bozen
108	LOC_CTP	ITH20	Trento
109	LOC_CTP	ITH3	Veneto
110	LOC_CTP	ITH31	Verona
111	LOC_CTP	ITH32	Vicenza
112	LOC_CTP	ITH33	Belluno
113	LOC_CTP	ITH34	Treviso
114	LOC_CTP	ITH35	Venezia
115	LOC_CTP	ITH36	Padova
116	LOC_CTP	ITH37	Rovigo
117	LOC_CTP	ITH4	Friuli Venezia Giulia
118	LOC_CTP	ITH41	Pordenone
119	LOC_CTP	ITH42	Udine
120	LOC_CTP	ITH43	Gorizia
121	LOC_CTP	ITH44	Trieste
122	LOC_CTP	ITH5	Emilia Romagna
123	LOC_CTP	ITH51	Piacenza
124	LOC_CTP	ITH52	Parma
125	LOC_CTP	ITH53	Reggio Emilia
126	LOC_CTP	ITH54	Modena
127	LOC_CTP	ITH55	Bologna
128	LOC_CTP	ITH56	Ferrara
129	LOC_CTP	ITH57	Ravenna
130	LOC_CTP	ITH58	Forlì Cesena
131	LOC_CTP	ITH59	Rimini
132	LOC_CTP	ITHBI12	Trentino-Alto Adige
133	LOC_CTP	ITI	Italia centrale
134	LOC_CTP	ITI1	Toscana
135	LOC_CTP	ITI11	Massa Carrara
136	LOC_CTP	ITI12	Lucca
137	LOC_CTP	ITI13	Pistoia
138	LOC_CTP	ITI14	Firenze
139	LOC_CTP	ITI15	Prato
140	LOC_CTP	ITI16	Livorno
141	LOC_CTP	ITI17	Pisa
142	LOC_CTP	ITI18	Arezzo
143	LOC_CTP	ITI19	Siena
144	LOC_CTP	ITI1A	Grosseto
145	LOC_CTP	ITI2	Umbria

	Colonna	Codice	Descrizione
146	LOC_CTP	ITI21	Perugia
147	LOC_CTP	ITI22	Terni
148	LOC_CTP	ITI3	Marche
149	LOC_CTP	ITI31	Pesaro Urbino
150	LOC_CTP	ITI32	Ancona
151	LOC_CTP	ITI33	Macerata
152	LOC_CTP	ITI34	Ascoli Piceno
153	LOC_CTP	ITI35	Fermo
154	LOC_CTP	ITI4	Lazio
155	LOC_CTP	ITI41	Viterbo
156	LOC_CTP	ITI42	Rieti
157	LOC_CTP	ITI43	Roma
158	LOC_CTP	ITI44	Latina
159	LOC_CTP	ITI45	Frosinone