In [1]:

```
import pandas as pd
import psycopg2
import os, io
from dotenv import load_dotenv
```

In [2]:

```
load_dotenv()
```

Out[2]:

```
True
```

In [3]:

```
def create_db_conn():
    try:
        conn = psycopg2.connect(host=os.getenv('DB_HOST'), database=os.getenv('D
B_NAME'),
                                                user=os.getenv('DB_USER'), password
=os.getenv('DB_PASSWORD'),
                                        port=int(os.getenv('DB_PORT')))
        return conn
    except psycopg2.DatabaseError as e:
        print(f'database connection {e}')
        return None
    except Exception as e:
        print(f'unknown error {e}')
        return None


def read_sql_iostream(query: str, block_mergejoin=False, block_hashjoin=False, b
lock_seqscan=False) -> pd.DataFrame:
    """
    More effective way of loading content of database table to dataframe using i
o stream - StringIO.
    :param str query: Query select for accessing data in table.
    :param con: Connection to concrete database.
    :return pd.Dataframe: Output dataframe loaded from database.
    """
    try:
        con = create_db_conn()
        cur = con.cursor()
        copy_sql = f"COPY ({query.strip().rstrip(';')}) TO STDOUT WITH CSV HEADE
R"
        store = io.StringIO()
        cur.copy_expert(copy_sql, store)
        store.seek(0)
        df = pd.read_csv(store, na_values=['NULL', 'NaN', 'nan', 'null', ''], ke
ep_default_na=False)
    except Exception as e:
        raise e
    finally:
        try:
            cur.close()
            con.close()
        except Exception as e:
            print(f'error- {e}')
            pass
    return df
```

In [4]:

```
# Check whether the data folder exists or not
if not os.path.exists('./data'):
    os.makedirs('./data')
```

In [5]:

```
def offset_query(limit, value_offset):
    return  f"""
    SELECT
        so.price_without_vat AS order_price_without_vat,
        so.price_with_vat AS order_price_with_vat,
        so.bill_country,
        so.setting_currency_id,
        so.created_at,
        so.shop_basket_id,
        so.doc_date,
        so.exchange_currency_rate,
        so.source_type AS source,
        so.canceled_date,

        sc.code AS currency_code,
        sc.currency_symbol,
        sc.price_round_system,



        sb.total_price_before_discount_with_vat AS basket_total_price_before_dis
count_with_vat,
        sb.total_price_with_vat AS basket_total_price_with_vat,
        sb.count_basket_items,
        sb.count_products AS basket_count_products,
        sb.basket_type,

        sbi.quantity AS item_quantity,
        sbi.item_type,
        sbi.unit_price_with_vat AS item_unit_price_with_vat,
        sbi.unit_price_without_vat AS item_unit_price_without_vat,
        sbi.total_discount_with_vat AS item_total_discount_with_vat,



        cp.id as product_id,
        cp.code AS product_code,
        cp.catalog_category_id,
        cp.catalog_brand_id,
        cp.name AS product_name,
        cp.status AS product_status,
        cp.reviews_count,
        cp.reviews_average_score_price,
        cp.reviews_average_score_quality,
        cp.reviews_average_score_properties,
        cp.reviews_average_score_overall,
        cp.reviews_average_score,
        cp.is_in_stock,
        cp.is_ended,
        cp.is_new,
        cp.is_boosted,
        cp.purchase_price AS product_purchase_price,
        cp.eshop_stock_count,
        cp.is_fifo,
        cp.name_parameterize AS product_name_parameterize,
        cp.created_at AS product_since,

        cc.name AS category,
        cc.tree_path,
        cc.name_parameterize AS category_name_parameterized,
        cc.status AS category_status,
        cc.catalog_segment_id,
        cc.ancestor_ids AS categories_ancestor_ids,
        cc.descendant_ids AS categories_descendant_ids,
        cc.full_name_path AS category_full_name_path,
        cc.default_warranty_period,

        cb.name AS brand_name,
        cb.name_parameterize AS brand_parameterized,

        cs.name AS segment_name,
        cs.name_parameterize AS segment_parameterized,
        cs.status AS segment_status


    FROM shop_orders so
    LEFT JOIN setting_currencies sc ON so.setting_currency_id = sc.id
    INNER JOIN shop_baskets sb ON sb.id = so.shop_basket_id
    LEFT JOIN shop_basket_items sbi ON  sb.id = sbi.shop_basket_id
    INNER JOIN catalog_products cp ON cp.id = sbi.catalog_product_id
    LEFT JOIN catalog_categories cc ON cp.catalog_category_id = cc.id
    LEFT JOIN catalog_brands cb ON cp.catalog_brand_id = cb.id
    LEFT JOIN catalog_segments cs ON cs.id = cp.catalog_segment_id
    LIMIT {limit}
    OFFSET {value_offset}
    """
```

In [6]:

```
## Approximately 3,650 mil rows, if done differently it crashes pandas
data_0 = (read_sql_iostream(offset_query(500000, 500000*0)))
```

In [7]:

```
data_1 = (read_sql_iostream(offset_query(500000, 500000*1)))
```

In [8]:

```
data_2 = (read_sql_iostream(offset_query(500000, 500000*2)))
```

In [10]:

```
data_3 = (read_sql_iostream(offset_query(500000, 500000*3)))
```

In [11]:

```
data_4 = (read_sql_iostream(offset_query(500000, 500000*4)))
```

In [13]:

```
data_5 = (read_sql_iostream(offset_query(500000, 500000*5)))
```

In [14]:

```
data_6 = (read_sql_iostream(offset_query(500000, 500000*6)))
```

In [15]:

```
data_7 = (read_sql_iostream(offset_query(500000, 500000*7)))
```

In [16]:

```
full_orders = pd.concat([data_0, data_1])
full_orders = pd.concat([full_orders, data_2])
full_orders = pd.concat([full_orders, data_3])
full_orders = pd.concat([full_orders, data_4])
full_orders = pd.concat([full_orders, data_5])
full_orders = pd.concat([full_orders, data_6])
full_orders = pd.concat([full_orders, data_7])
```

In [18]:

```
full_orders.to_csv('data/data.csv', index=False)
```

In [19]:

```
full_orders
```

Out[19]:

	order_price_without_vat	order_price_with_vat	bill_country	setting_currency_id	
0	562.29000	674.75	BG	1	19:2
1	562.29000	674.75	BG	1	19:2
2	562.29000	674.75	BG	1	19:2
3	562.29000	674.75	BG	1	19:2
4	562.29000	674.75	BG	1	19:2
...	...	...	...	...	
180641	111.98000	129.90	DE	6	23:5
180642	2966.94000	3590.00	CZ	4	13:0
180643	241.90083	292.70	BE	6	01:1
180644	241.90083	292.70	BE	6	01:1
180645	1752.06000	2120.00	CZ	4	17:3

3680646 rows × 58 columns