Here's the code with the filled-in blocks for connecting to DuckDB and appending tables:

Python

```python
import pyarrow as pa
import pyarrow.parquet as pq
import requests
import fsspec
import duckdb

def load_data_from_api(*args, **kwargs):
    # setup the vars
    months = range(1, 13)
    year = 2022
    colour = 'green'  # 'yellow'
    base_url = "https://d37ci6vzurychx.cloudfront.net/trip-data"

    # Connect to local DuckDB
    conn = duckdb.connect()

    # Create empty list to store DataFrames (not needed for DuckDB)
    # dataframes = []

    # Iterate through months and download data
    for month in months:
        print(month)
        filename = f"{colour}_tripdata_{year}-{month:02d}.parquet"  # .csv.gz / parquet
        print(filename)
        url = f"{base_url}/{filename}"
        print(url)

        response = requests.get(url, stream=True)
        if response.status_code == 200:
            fs = fsspec.filesystem("http")
            with fs.open(url) as f:
                table = pq.read_table(f)

                # Append each month's table to DuckDB
                arrow.duckdb.write(table, connection=conn, table_name="your_table_name")

        else:
            print(f"Failed to download {filename}. Status code: {response.status_code}")
```

```
    # Return the DuckDB connection
    return conn
```

Key changes:

1. **Connecting to DuckDB:**
   - Added conn = duckdb.connect() to establish a connection.
2. **Appending tables:**
   - Removed the dataframes list as it's not necessary for DuckDB.
   - Replaced dataframes.append(table) with arrow.duckdb.write(table, connection=conn, table_name="your_table_name") to directly write the PyArrow table to DuckDB.
3. **Returning the connection:**
   - The function now returns the DuckDB connection object (conn) for further use.

Remember:

- Replace "your_table_name" with the actual table name you want to create in DuckDB.
- Close the DuckDB connection (conn.close()) when you're finished with it.