```python
from project_functions.connect import Connect
from project_functions.query import Query
import pandas as pd
from calendar import monthrange

class Transaction():
    """
    Wrapper around Connect and Query classes to query against the transactions database
    and validate the query results.
    """
    def __init__(self, connection:Connect):
        """
        Take in the database connection and set an internal attribute to the connection.
        """

        #store the connection object as an attribute
        self.connection = connection

        #Instantiate a query class
        self.query = Query(connection)

        #Ensure the user has access to this table and data can be read from it.
        self._validate_transaction_connection()

    def retreive_transactions(self, month:str, year:str):
        """
        Take in a month and year value in MM and YYYY format, respectively, and return a dataframe of all transactions in that time frame.

        Invalid queries should return a one row dataframe with -1 for all values.
        """
        try:
            #Identify the maximum day value for the day and month
            max_day = monthrange(int(year),int(month))[1]

            #Compose the min and max date strings for the query
            min_date = f'{year}-{month}-01 00:00:00'
            max_date = f'{year}-{month}-{max_day} 00:00:00'

            #Use the query manager to query the db with the min and max date parameters. Also, drop account id
            result = self.query.read_query(f"""
                                    SELECT *
                                    FROM transaction
                                    WHERE txn_date BETWEEN %s AND %s
                                """, params=(min_date, max_date)).drop("account_id", axis=1)

            #Assert that the result must have more than one value, and raise an error otherwise
            assert len(result) > 0, "The entered date is invalid."

            #Return the result
            return result

        #Capture the exception if the try block, above, fails
        except Exception as e:
            print(f'Error: {e}')
            # return the invalid df
            return self.invalid_df

    def _validate_transaction_connection(self):
        """
        Validate the connection to the transaction table.

        Also, construct the dataframe that will be returned if self.retreive_transactions encounters an
        invalid month and year.
        """
        try:
            #Attempt to query the transaction table
            result = self.query.read_query("""
                                    SELECT *
                                    FROM transaction
                                    LIMIT 1
                                """)

            # Construct the invalid dataframe from the result
            self.invalid_df = {}

            #Loop through the columns in result
            for col in result.columns:

                #We don't want to show account_id, so skip that
                if col != "account_id":
                    self.invalid_df[col]=[-1]

            #Store the invalid dataframe as an attribute to be returned if necessary.
            self.invalid_df = pd.DataFrame(self.invalid_df)

        #Capture the exception if the try block, above, fails
        except Exception as e:
            print(f'ERROR: {e}')
```