```python
import pandas as pd

class Connect():
    """
    Wrapper for pandas SQL engine and query management.

    Additional functionality has been added for credential ingestion from a text file.
    """
    def __init__(self, host:str = 'localhost', port:str = '5432', db:str = 'bank'):

        #These can be customized, but the default values will be used for this project
        self.host = host
        self.port = port
        self.db = db

    def connect(self, path:str):
        """
        Read in the credentials from 'path' and test the DB connection.
        """
        try:
            #Create a dictionary for credentials
            creds = {}

            #Open the file from the supplied path
            with open(path) as f:

                #This is a very naive approach to this.
                #The vault file will be in a user\npass format per the assignment,
                # but this will fail in any other scenario.
                for i, line in enumerate(f.readlines()):
                    if i==0:
                        creds["user"]=line.strip()
                    else:
                        creds["pass"]=line.strip()

            #Use the private _test_connection method to verify the credentials.
            self.engine = self._test_connection(user=creds["user"], password=creds["pass"])

        # Catch an exception within the try block and print it to the user
        except Exception as e:
            print(f"Error: {e}")

    def _test_connection(self, user, password):
        """
        Verify the credentials are valid for the desired database.
        """
        try:
            #Build the engine string
            engine = f'postgresql://{user}:{password}@{self.host}:{self.port}/{self.db}'

            #Query information about the database schema.
            #This allows the test to be invariant to a table, making it more flexible
            pd.read_sql_query("""
                              SELECT *
                              FROM INFORMATION_SCHEMA.TABLES
                              """
                              ,engine)

            #Give the user some feedback
            print(f"User: {user} successfully connected to {self.db}!")
            return engine

        # Catch an exception within the try block and print it to the user
        except Exception as e:
            print(f"Error: {e}")
```