I want to query a oracle table using python library 'jaydebeapi' and display the table data in dashboard

Assume the oracle table has columns like 'id', 'business\_date' ,'status' ,'start\_time', 'end\_time', 'excution\_time' and has some rows like

(001, 2024-11-01, Success ,2024-11-01, 2024-11-02, 2)

(002, 2024-11-02, Success ,2024-11-01, 2024-11-01, 1)

(003, 2024-11-03, FAILED ,2024-11-01, 2024-11-04, 4)

(004, 2024-11-04, Success ,2024-11-01, 2024-11-03, 3)

The dashboard should have the options to run 4 different types of queries

Below are my requriements:

1. Query the table and fetch the data for given 'business\_date' . There should be an field to pass this 'business\_date' from the dashboard at runtime.

2. Query the table and fetch the data for given 'business\_date' and 'id' . There should be an field to pass both 'business\_date' & 'id' from the dashboard at runtime.

3. Query the table and fetch the data for given range of 'business\_date'. That is it should have a field to pass 'from' and 'to' business\_date. Once the results are displayed on to the dashboard , the dashboard should have a filter/dropdown filter to see the specific 'id'

Consider the below

Use the best approach to query the oracle table library 'jaydebeapi'

Use the best technology to create the dashboard

**ChatGPT said:**

ChatGPT

To implement your solution, we need to:

1. Query the Oracle table using the jaydebeapi library to fetch data based on various conditions.
2. Build a dashboard that provides the ability to:
   * Filter results by business\_date, id, and date range.
   * Display data and allow users to filter the results interactively using dropdowns and text fields.

**Step-by-step Approach**

**1. Setup Oracle Connection using jaydebeapi**

jaydebeapi is a Python library that allows you to connect to databases through JDBC. You'll need to install this library and set up a connection to your Oracle database. You'll also need the appropriate JDBC driver (usually ojdbc8.jar) for Oracle.

**Install the necessary libraries:**

bash

Copy code

pip install jaydebeapi pandas

**JDBC connection setup:**

You’ll need to specify your Oracle JDBC driver and connection string. Here's a Python function to establish the connection:

python

Copy code

import jaydebeapi

def get\_oracle\_connection():

conn = jaydebeapi.connect(

"oracle.jdbc.driver.OracleDriver", # Oracle JDBC driver

"jdbc:oracle:thin:@your\_host:port:SID", # JDBC URL

["username", "password"], # DB credentials

"path\_to\_ojdbc8.jar" # Path to Oracle JDBC driver

)

return conn

**2. Query the Oracle Table**

You can define four functions to query the database based on the user input.

* **1. Query by business\_date:**

python

Copy code

import pandas as pd

def fetch\_data\_by\_business\_date(business\_date):

conn = get\_oracle\_connection()

query = f"""

SELECT \* FROM your\_table\_name

WHERE business\_date = TO\_DATE('{business\_date}', 'YYYY-MM-DD')

"""

df = pd.read\_sql(query, conn)

conn.close()

return df

* **2. Query by business\_date and id:**

python

Copy code

def fetch\_data\_by\_business\_date\_and\_id(business\_date, id):

conn = get\_oracle\_connection()

query = f"""

SELECT \* FROM your\_table\_name

WHERE business\_date = TO\_DATE('{business\_date}', 'YYYY-MM-DD')

AND id = '{id}'

"""

df = pd.read\_sql(query, conn)

conn.close()

return df

* **3. Query by a range of business\_date:**

python

Copy code

def fetch\_data\_by\_date\_range(start\_date, end\_date):

conn = get\_oracle\_connection()

query = f"""

SELECT \* FROM your\_table\_name

WHERE business\_date BETWEEN TO\_DATE('{start\_date}', 'YYYY-MM-DD')

AND TO\_DATE('{end\_date}', 'YYYY-MM-DD')

"""

df = pd.read\_sql(query, conn)

conn.close()

return df

**3. Build the Dashboard**

We will use **Streamlit** to create an interactive dashboard. Streamlit is a great framework for building quick and interactive web applications in Python. It integrates well with Pandas for data visualization and supports widgets like text inputs, date pickers, and dropdowns.

**Install Streamlit:**

bash

Copy code

pip install streamlit

**4. Streamlit Dashboard Code**

Below is a Streamlit dashboard that satisfies your requirements:

python

Copy code

import streamlit as st

import pandas as pd

from datetime import datetime

# Import the database query functions

from your\_query\_functions import fetch\_data\_by\_business\_date, fetch\_data\_by\_business\_date\_and\_id, fetch\_data\_by\_date\_range

# Function to display the dashboard

def display\_dashboard():

st.title("Oracle Table Dashboard")

# 1. Query by Business Date

st.subheader("Query by Business Date")

business\_date = st.date\_input("Select Business Date", datetime.today())

if st.button('Fetch Data for Business Date'):

df = fetch\_data\_by\_business\_date(business\_date.strftime('%Y-%m-%d'))

st.dataframe(df)

# 2. Query by Business Date and ID

st.subheader("Query by Business Date and ID")

business\_date\_id = st.date\_input("Select Business Date", datetime.today())

id\_input = st.text\_input("Enter ID")

if st.button('Fetch Data for Business Date and ID'):

df = fetch\_data\_by\_business\_date\_and\_id(business\_date\_id.strftime('%Y-%m-%d'), id\_input)

st.dataframe(df)

# 3. Query by Date Range

st.subheader("Query by Date Range")

start\_date = st.date\_input("Start Date", datetime.today())

end\_date = st.date\_input("End Date", datetime.today())

if st.button('Fetch Data for Date Range'):

df = fetch\_data\_by\_date\_range(start\_date.strftime('%Y-%m-%d'), end\_date.strftime('%Y-%m-%d'))

st.dataframe(df)

# Dropdown to filter by ID in the result set

if not df.empty:

id\_options = df['id'].unique().tolist()

selected\_id = st.selectbox("Filter by ID", id\_options)

filtered\_df = df[df['id'] == selected\_id]

st.dataframe(filtered\_df)

# Run the dashboard app

if \_\_name\_\_ == "\_\_main\_\_":

display\_dashboard()

**Explanation:**

1. **st.title() and st.subheader()**: These create headings on the dashboard.
2. **st.date\_input()**: This widget allows the user to select a date for querying by business date.
3. **st.text\_input()**: This widget allows the user to input an ID.
4. **st.button()**: When clicked, the respective query is executed.
5. **st.dataframe()**: Displays the queried results in a tabular format.
6. **Date Range and ID filter**: For the date range query, once results are displayed, a dropdown allows the user to filter the data by id.

**5. Running the Dashboard**

1. Save the Python code for the dashboard in a file, e.g., dashboard.py.
2. Run the Streamlit app by executing the following command in your terminal:

bash

Copy code

streamlit run dashboard.py

This will launch the app in your default web browser.

**Considerations:**

1. **Security**: Ensure your database credentials are handled securely, possibly using environment variables or configuration files that are not committed to source control.
2. **JDBC Driver**: Make sure that the Oracle JDBC driver (ojdbc8.jar) is correctly installed and available to your Python environment.
3. **Scalability**: If the dataset grows large, consider implementing pagination in Streamlit to avoid performance bottlenecks.
4. **Error Handling**: Add exception handling in your query functions to handle any database connectivity or SQL issues gracefully.

By using **Streamlit**, you create an intuitive, easy-to-deploy dashboard for querying and displaying data from an Oracle database.

4o mini

**You said:**

show me how to create a python project by using the above. and consider pass the credentails via a config file. Add the pagination and error handling

**ChatGPT said:**

ChatGPT

Creating a Python project for querying an Oracle database using the jaydebeapi library and displaying the results in an interactive dashboard via Streamlit involves several steps. Below is a comprehensive guide to setting up the project, including passing credentials via a configuration file, implementing pagination, and adding error handling.

**Project Directory Structure:**

Here’s how your project directory structure could look:

graphql

Copy code

oracle-dashboard/

├── config.json # Configuration file for credentials

├── query\_functions.py # Python functions for querying the Oracle database

├── dashboard.py # Main Streamlit dashboard application

├── requirements.txt # List of dependencies

└── ojdbc8.jar # Oracle JDBC driver (make sure it's in your project folder)

**1. Creating the Configuration File (config.json)**

You will store your Oracle database credentials and other configuration settings in a JSON file for security and convenience.

**config.json:**

json

Copy code

{

"db\_url": "jdbc:oracle:thin:@your\_host:port:SID",

"username": "your\_username",

"password": "your\_password",

"jdbc\_driver\_path": "ojdbc8.jar"

}

* db\_url: The JDBC URL for connecting to your Oracle database.
* username and password: Your database login credentials.
* jdbc\_driver\_path: Path to your ojdbc8.jar file (make sure this file is downloaded and placed in the project directory).

**2. Creating Query Functions (query\_functions.py)**

In this file, we'll write the functions to connect to the Oracle database and fetch data. The functions will handle pagination, error handling, and reading credentials from the configuration file.

**query\_functions.py:**

python

Copy code

import jaydebeapi

import pandas as pd

import json

from datetime import datetime

# Function to load database credentials from config.json

def load\_db\_config():

with open('config.json', 'r') as f:

config = json.load(f)

return config

# Function to create a database connection

def get\_oracle\_connection():

config = load\_db\_config()

try:

conn = jaydebeapi.connect(

"oracle.jdbc.driver.OracleDriver", # Oracle JDBC driver

config["db\_url"], # JDBC URL

[config["username"], config["password"]], # DB credentials

config["jdbc\_driver\_path"] # Path to the Oracle JDBC driver

)

return conn

except Exception as e:

print(f"Error connecting to the database: {e}")

return None

# Function to fetch data by business date

def fetch\_data\_by\_business\_date(business\_date, page=1, page\_size=10):

conn = get\_oracle\_connection()

if conn is None:

return pd.DataFrame()

offset = (page - 1) \* page\_size

query = f"""

SELECT \* FROM your\_table\_name

WHERE business\_date = TO\_DATE('{business\_date}', 'YYYY-MM-DD')

ORDER BY business\_date

OFFSET {offset} ROWS FETCH NEXT {page\_size} ROWS ONLY

"""

try:

df = pd.read\_sql(query, conn)

conn.close()

return df

except Exception as e:

print(f"Error executing query: {e}")

return pd.DataFrame()

# Function to fetch data by business date and id

def fetch\_data\_by\_business\_date\_and\_id(business\_date, id, page=1, page\_size=10):

conn = get\_oracle\_connection()

if conn is None:

return pd.DataFrame()

offset = (page - 1) \* page\_size

query = f"""

SELECT \* FROM your\_table\_name

WHERE business\_date = TO\_DATE('{business\_date}', 'YYYY-MM-DD')

AND id = '{id}'

ORDER BY business\_date

OFFSET {offset} ROWS FETCH NEXT {page\_size} ROWS ONLY

"""

try:

df = pd.read\_sql(query, conn)

conn.close()

return df

except Exception as e:

print(f"Error executing query: {e}")

return pd.DataFrame()

# Function to fetch data by a range of business dates

def fetch\_data\_by\_date\_range(start\_date, end\_date, page=1, page\_size=10):

conn = get\_oracle\_connection()

if conn is None:

return pd.DataFrame()

offset = (page - 1) \* page\_size

query = f"""

SELECT \* FROM your\_table\_name

WHERE business\_date BETWEEN TO\_DATE('{start\_date}', 'YYYY-MM-DD')

AND TO\_DATE('{end\_date}', 'YYYY-MM-DD')

ORDER BY business\_date

OFFSET {offset} ROWS FETCH NEXT {page\_size} ROWS ONLY

"""

try:

df = pd.read\_sql(query, conn)

conn.close()

return df

except Exception as e:

print(f"Error executing query: {e}")

return pd.DataFrame()

**Explanation:**

* The get\_oracle\_connection function reads database credentials from config.json and attempts to establish a connection using jaydebeapi.
* The fetch\_data\_by\_\* functions are responsible for querying the Oracle database. They include pagination by using SQL OFFSET and FETCH NEXT clauses to limit the number of results per page.
* Each query function accepts page and page\_size arguments to implement pagination.

**3. Creating the Streamlit Dashboard (dashboard.py)**

This is where the user interacts with the dashboard. You’ll provide widgets like date pickers, text inputs, and buttons to filter and view the data.

**dashboard.py:**

python

Copy code

import streamlit as st

import pandas as pd

from datetime import datetime

from query\_functions import fetch\_data\_by\_business\_date, fetch\_data\_by\_business\_date\_and\_id, fetch\_data\_by\_date\_range

# Default pagination settings

PAGE\_SIZE = 10

# Function to display the dashboard

def display\_dashboard():

st.title("Oracle Table Dashboard")

# 1. Query by Business Date

st.subheader("Query by Business Date")

business\_date = st.date\_input("Select Business Date", datetime.today())

page\_num = st.number\_input("Page Number", min\_value=1, value=1)

if st.button('Fetch Data for Business Date'):

df = fetch\_data\_by\_business\_date(business\_date.strftime('%Y-%m-%d'), page=page\_num, page\_size=PAGE\_SIZE)

if not df.empty:

st.dataframe(df)

else:

st.write("No data found.")

# 2. Query by Business Date and ID

st.subheader("Query by Business Date and ID")

business\_date\_id = st.date\_input("Select Business Date", datetime.today())

id\_input = st.text\_input("Enter ID")

page\_num = st.number\_input("Page Number", min\_value=1, value=1)

if st.button('Fetch Data for Business Date and ID'):

df = fetch\_data\_by\_business\_date\_and\_id(business\_date\_id.strftime('%Y-%m-%d'), id\_input, page=page\_num, page\_size=PAGE\_SIZE)

if not df.empty:

st.dataframe(df)

else:

st.write("No data found.")

# 3. Query by Date Range

st.subheader("Query by Date Range")

start\_date = st.date\_input("Start Date", datetime.today())

end\_date = st.date\_input("End Date", datetime.today())

page\_num = st.number\_input("Page Number", min\_value=1, value=1)

if st.button('Fetch Data for Date Range'):

df = fetch\_data\_by\_date\_range(start\_date.strftime('%Y-%m-%d'), end\_date.strftime('%Y-%m-%d'), page=page\_num, page\_size=PAGE\_SIZE)

if not df.empty:

st.dataframe(df)

# Dropdown to filter by ID in the result set

id\_options = df['id'].unique().tolist()

selected\_id = st.selectbox("Filter by ID", id\_options)

filtered\_df = df[df['id'] == selected\_id]

st.dataframe(filtered\_df)

else:

st.write("No data found.")

# Run the dashboard app

if \_\_name\_\_ == "\_\_main\_\_":

display\_dashboard()

**Explanation:**

* Streamlit widgets are used to gather input from the user:
  + st.date\_input() for date pickers (business date, start date, and end date).
  + st.text\_input() for the ID input field.
  + st.number\_input() for pagination (page number).
* Once the user presses the "Fetch Data" button, the respective query is executed, and the results are displayed in a paginated format.
* If the user selects a date range, the results are displayed, and a dropdown (st.selectbox()) is provided to filter results by id.

**4. Dependencies (requirements.txt)**

To ensure your environment is set up with the necessary dependencies, create a requirements.txt file:

**requirements.txt:**

Copy code

jaydebeapi

pandas

streamlit

You can install all dependencies at once using the following command:

bash

Copy code

pip install -r requirements.txt

**5. Running the Project**

1. **Run the Streamlit Dashboard:**

Make sure you have your Oracle JDBC driver (ojdbc8.jar) placed in the project directory, and the config.json file contains the correct credentials.

To start the dashboard, run the following command:

bash

Copy code

streamlit run dashboard.py

1. **View the Dashboard:**

Streamlit will automatically open a local web server