Natural Language Query Parser

User Manual

Data &
Analytics

Building a better
working world

Table of Contents

Data & Analytics

EY

Required Programs

In order to download, install and run the project, there is one program that is required and another that is recommended for users.

Docker Desktop is a program that allows users to download Docker containers, these containers consist of the project itself as well as any other necessary data to ensure that the project can run on virtually any machine and operating system. Thus, it is imperative that users download this program. A link to Docker Desktop can be found here: https://www.docker.com/get-started.

As the project is being uploaded to a GitHub repository, it is also recommended that users download GitHub Desktop to ensure that they are working with the most recent version of the project. This program also allows users to push any changes made to the project to the central repository. A link to GitHub Desktop can be found here: https://desktop.github.com/.

Lastly, for those users that wish to view the source code of the project or make changes, a code editor is recommended. There are many to choose from, however it is recommended that users download Microsoft Visual Studio code due to its ease of use and native support within Docker. A link to Microsoft Visual Studio can be found here: https://code.visualstudio.com/.

Required **Recommended** **Recommended (for developers)**

Installation and Setup

To install the programs mentioned in the previous slide, navigate to the respective program websites and follow the prompts provided to download and install the software. Moving forward, this guide will assume you have installed all these programs or are aware of alternate options for the recommended programs.

Importing the Project with GitHub Desktop

1. Once all the programs have been downloaded and installed, begin by opening the Docker Desktop program, keeping it minimised for now.
2. Open the GitHub Desktop program, on the initial screen you will see a 'Create new repository' button. (Figure 1).
3. Click that button, it will open a dialog box. Navigate to the URL tab within that box and copy the following link into the provided space: https://github.com/murphce/pace-2020-s2-group-1. This is the link to the GitHub repository that the project is stored on.
4. Below where the URL was entered, specify where you would like the project to be saved to the PC.
5. Once the repo has been successfully cloned, find the 'Fetch origin' tab at the top of the screen and click that button. (Figure 2). This will ensure that the version installed on your computer is the most recent build of the project. This is also how new versions of the project are downloaded so it is important to check for changes on occasion.

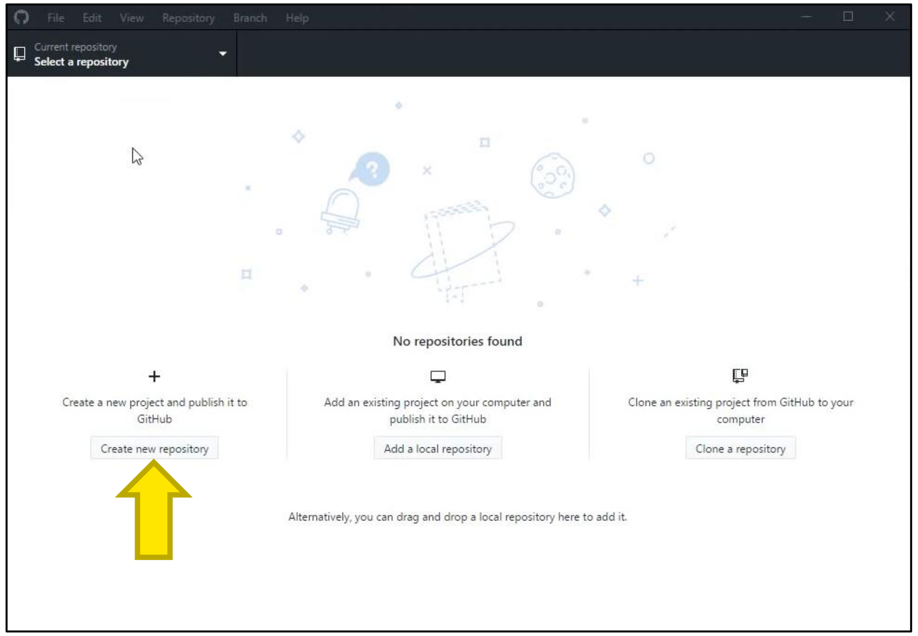

File
Edit
View
Repository
Branch
Help
Current repository
Select a repository
No repositories found
Create a new project and publish it to GitHub
Create new repository
Add an existing project on your computer and publish it to GitHub
Add a local repository
Clone an existing project from GitHub to your computer
Clone a repository
Alternatively, you can drag and drop a local repository here to add it.

Figure 1

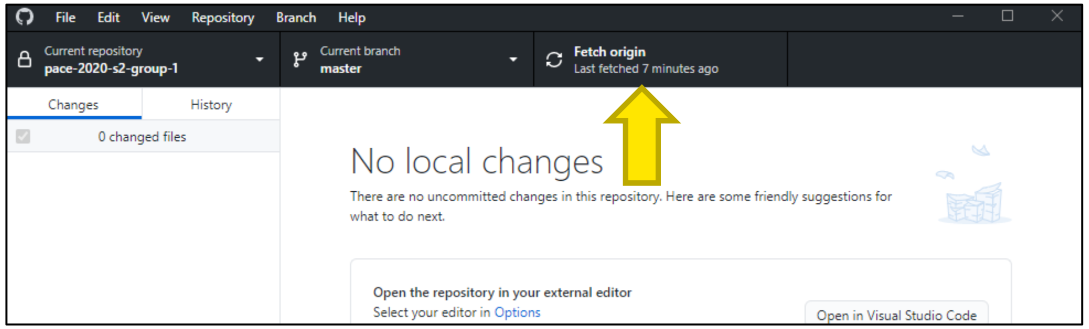

File
Edit
View
Repository
Branch
Help
Current repository
pace-2020-s2-group-1
Current branch
master
Fetch origin
Last fetched 7 minutes ago
Changes
History
0 changed files
No local changes
There are no uncommitted changes in this repository. Here are some friendly suggestions for what to do next.
Open the repository in your external editor
Select your editor in Options
Open in Visual Studio Code

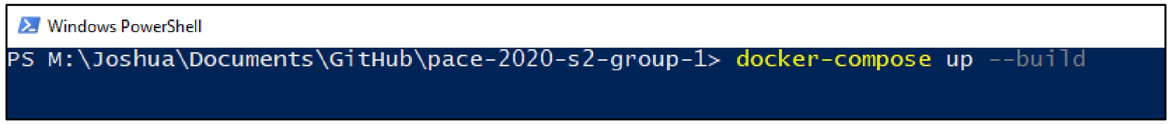
Figure 2

Installation and Setup

Importing the project into Docker and running the project

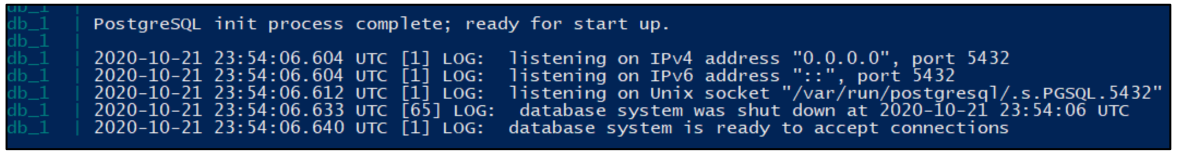
1. Once the project has been successfully cloned from the GitHub repository, navigate to the folder containing the project that was specified in step 3 of the previous slide. (The folder should be called ‘pace-2020-s2-group-1’)
2. Shift-right click on that folder and select ‘Open Powershell window here’
3. In the window that is opened, enter the following command: docker-compose up --build (Figure 3)
4. It will take a few minutes to install, but once the ‘database system is ready to accept connections’ message appears at the bottom, the project is installed. (Figure 4)
5. To open the project, navigate to the Docker Desktop screen. You will notice a tab named ‘pace-2020-s2-group-1’.
6. On the left side of that tab is a small arrow, click the arrow to expand the tab.
7. You will notice two more tabs appear underneath. The second tab allows you to open the project through the ‘open in browser’ button. Click on that and the project will open in your internet browser. (Figure 5)

Note: After step 4, the project can also be opened by navigating to an internet browser and entering the following: http://localhost:5000



```
PS M:\Joshua\Documents\GitHub\pace-2020-s2-group-1> docker-compose up --build
```

Figure 3



```
db_1  | PostgreSQL init process complete; ready for start up.
db_1  |
db_1  | 2020-10-21 23:54:06.604 UTC [1] LOG:  listening on IPv4 address "0.0.0.0", port 5432
db_1  | 2020-10-21 23:54:06.604 UTC [1] LOG:  listening on IPv6 address "::", port 5432
db_1  | 2020-10-21 23:54:06.612 UTC [1] LOG:  listening on Unix socket "/var/run/postgresql/.s.PGSQL.5432"
db_1  | 2020-10-21 23:54:06.633 UTC [65] LOG:  database system was shut down at 2020-10-21 23:54:06 UTC
db_1  | 2020-10-21 23:54:06.640 UTC [1] LOG:  database system is ready to accept connections
```

Figure 4

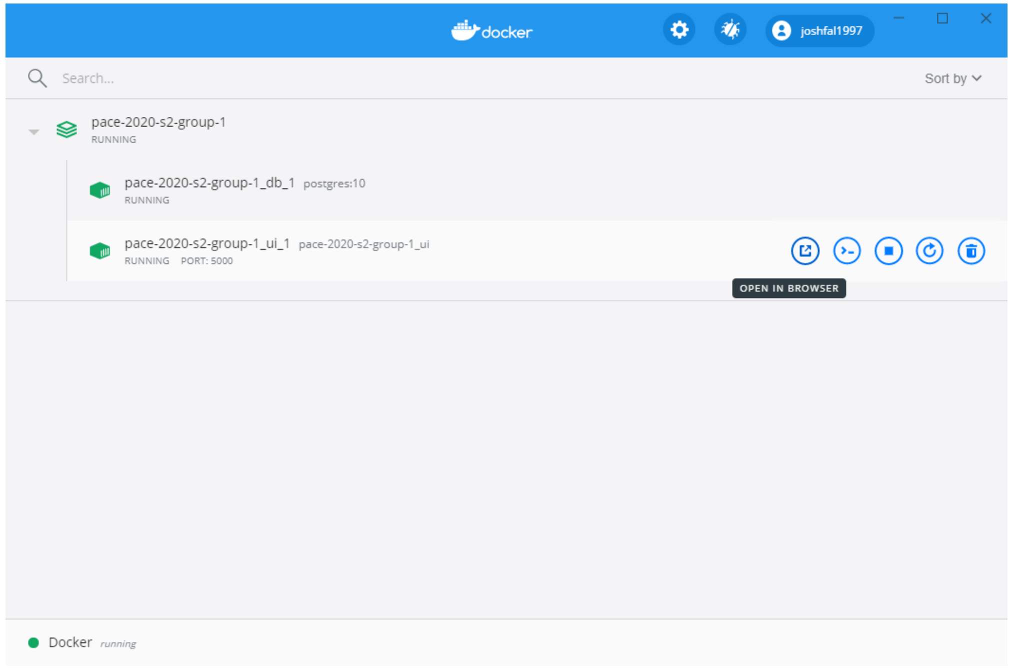
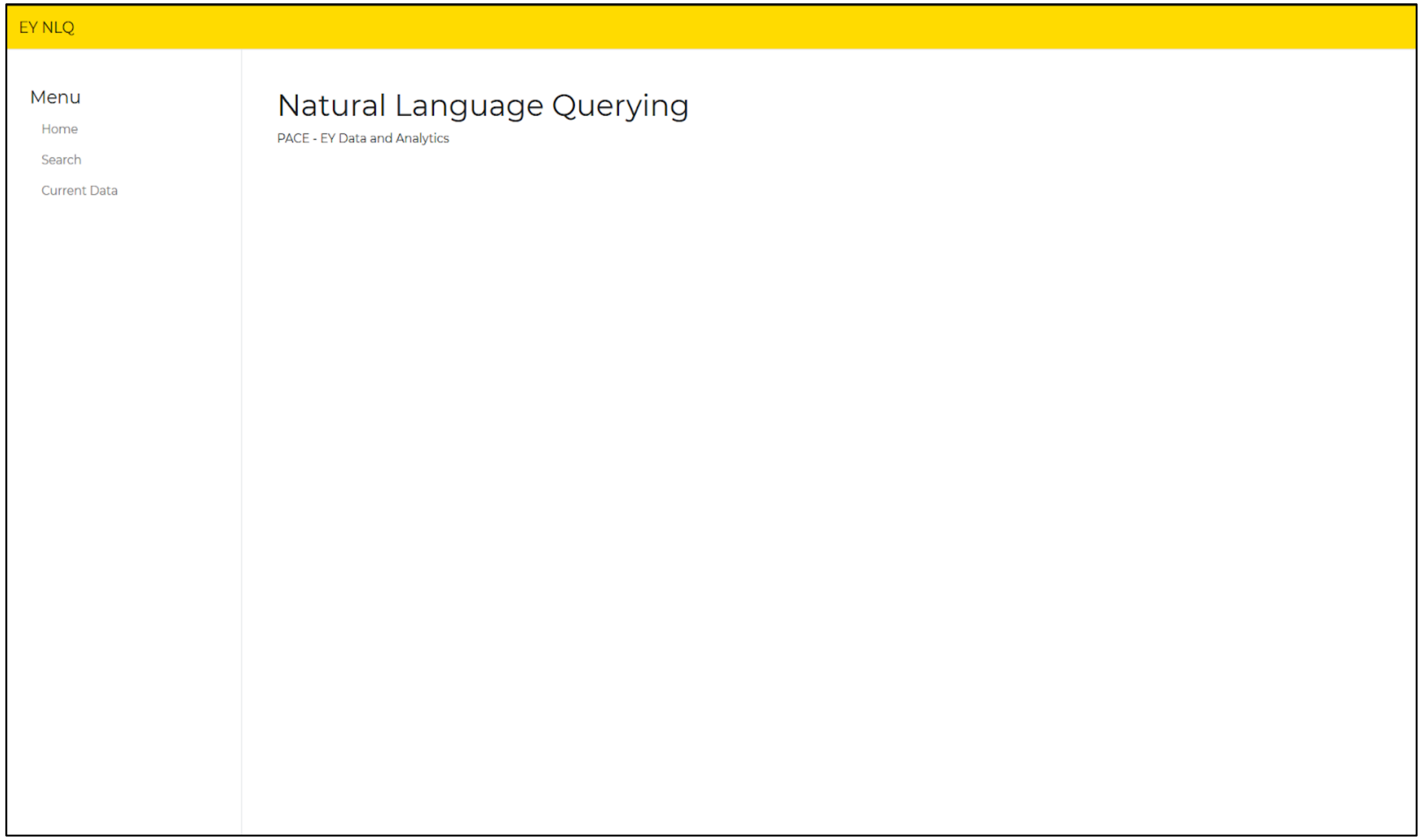


Figure 5

Installation and Setup

If the project has been successfully installed, you should see the following screen:



Natural Language Querying

PACE - EY Data and Analytics

Project Home Screen

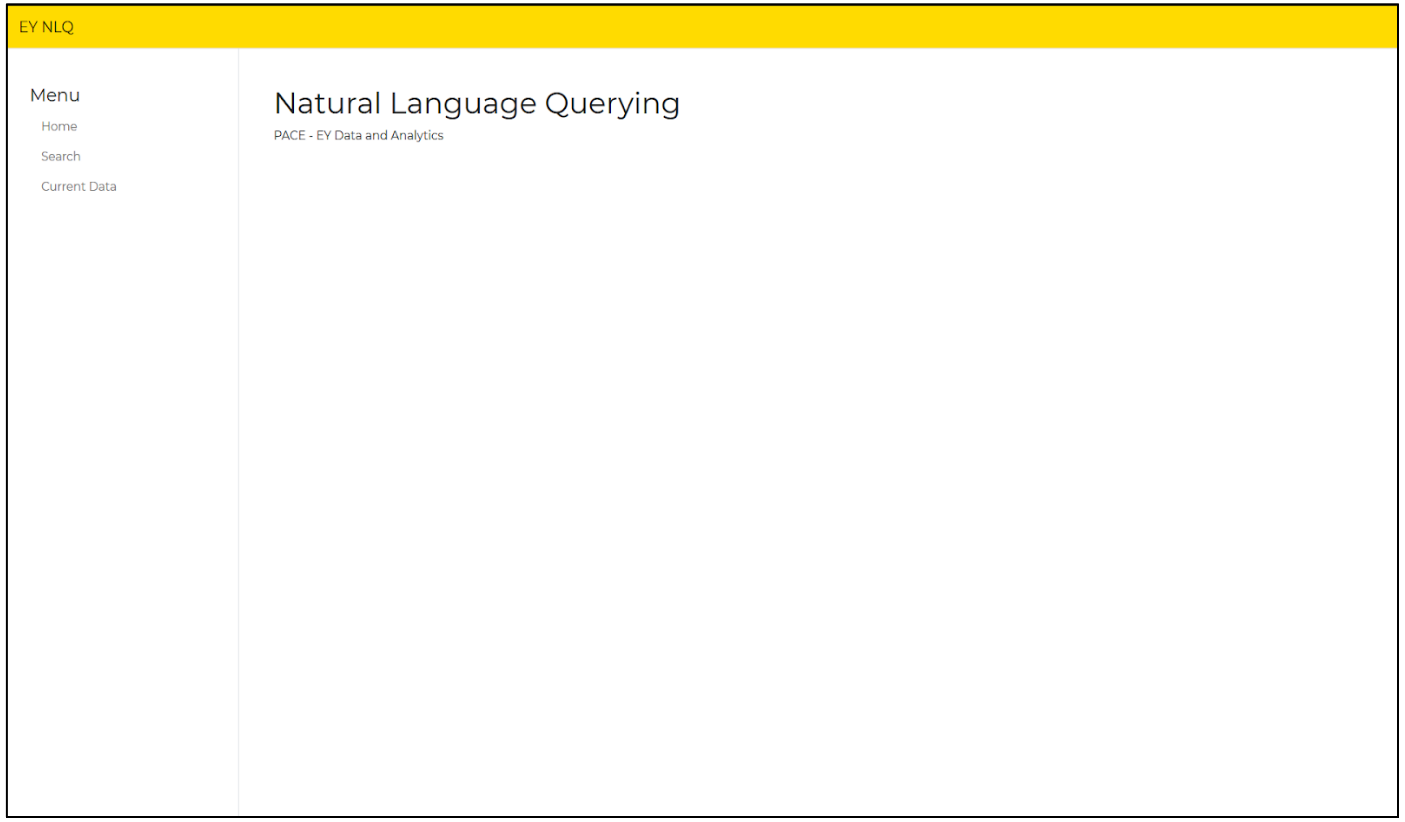
From the Home Screen, you will notice a menu on the left-hand side of the screen. This menu allows users to navigate the project.

The menu options are as follows:

Home: Returns users to the Home screen

Search: Allows users to enter queries on the current database

Current Data: Displays the entirety of the currently loaded database



The Home Screen

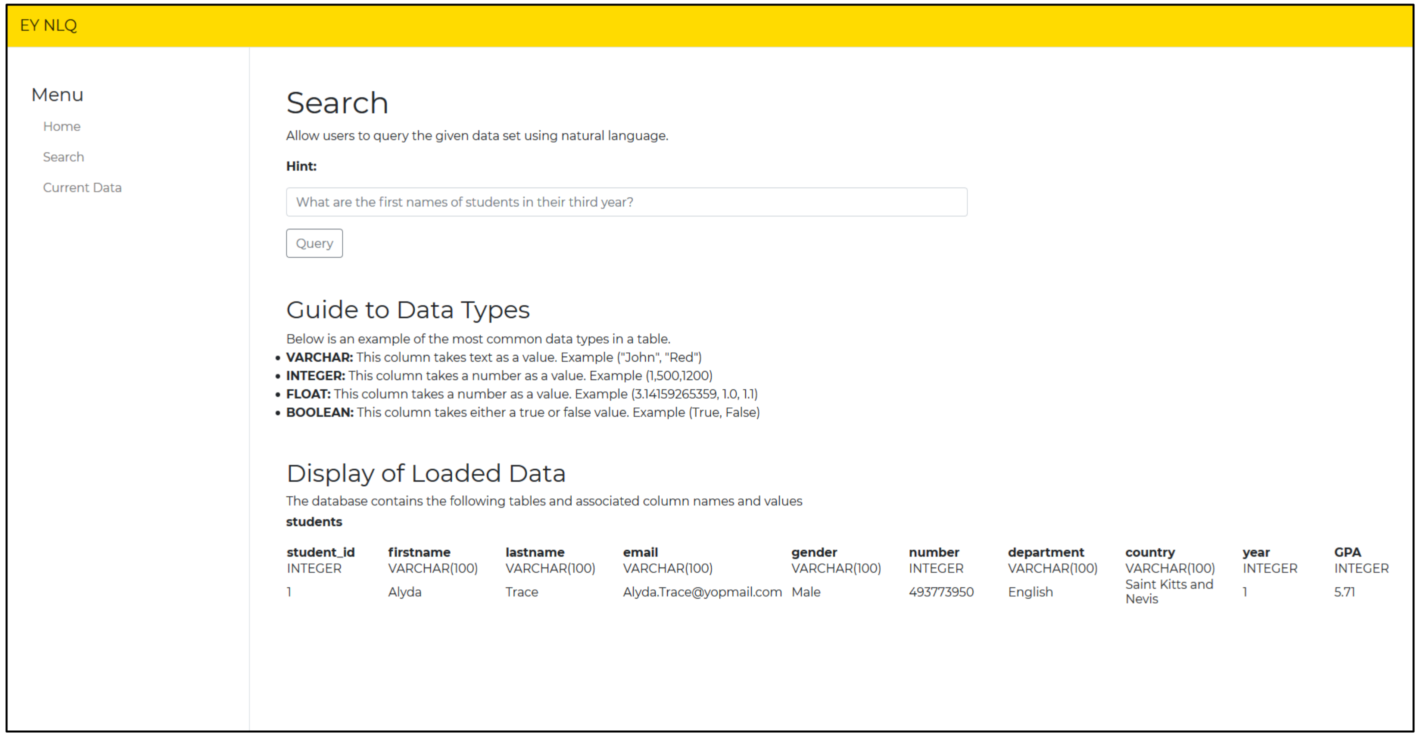
The Search Screen

By clicking the Search tab on the Menu bar, users are brought to the Search Screen. It is here that users can enter and run queries.

Queries are entered into the search bar and are run by pressing the Query button underneath. A hint is provided to the user within the search bar to give them an idea of a type of query they could run.

Underneath the search bar is the **Guide to Data Types** section. This provides a list of common datatypes that may be present in the currently loaded database. It is designed to give users who are unfamiliar with programming and data types a better understanding of the data they are querying.

This guide is also useful for the section underneath, **Display of Loaded Data.** This section provides a list of all the tables contained within the database. For each table present, the column names of that table are given. For every column, the data type stored within is also given. Finally an example record is given to the user.



Search

Allow users to query the given data set using natural language.

Hint:

What are the first names of students in their third year?

Guide to Data Types

Below is an example of the most common data types in a table.

- **VARCHAR:** This column takes text as a value. Example ("John", "Red")
- **INTEGER:** This column takes a number as a value. Example (1,500,1200)
- **FLOAT:** This column takes a number as a value. Example (3.14159265359, 1.0, 1.1)
- **BOOLEAN:** This column takes either a true or false value. Example (True, False)

Display of Loaded Data

The database contains the following tables and associated column names and values

students

student_id	**firstname**	**lastname**	**email**	**gender**	**number**	**department**	**country**	**year**	**GPA**
INTEGER	VARCHAR(100)	VARCHAR(100)	VARCHAR(100)	VARCHAR(100)	INTEGER	VARCHAR(100)	VARCHAR(100)	INTEGER	INTEGER
1	Alyda	Trace	Alyda.Trace@yopmail.com	Male	493773950	English	Saint Kitts and Nevis	1	5.71

The Search Screen

Query Results

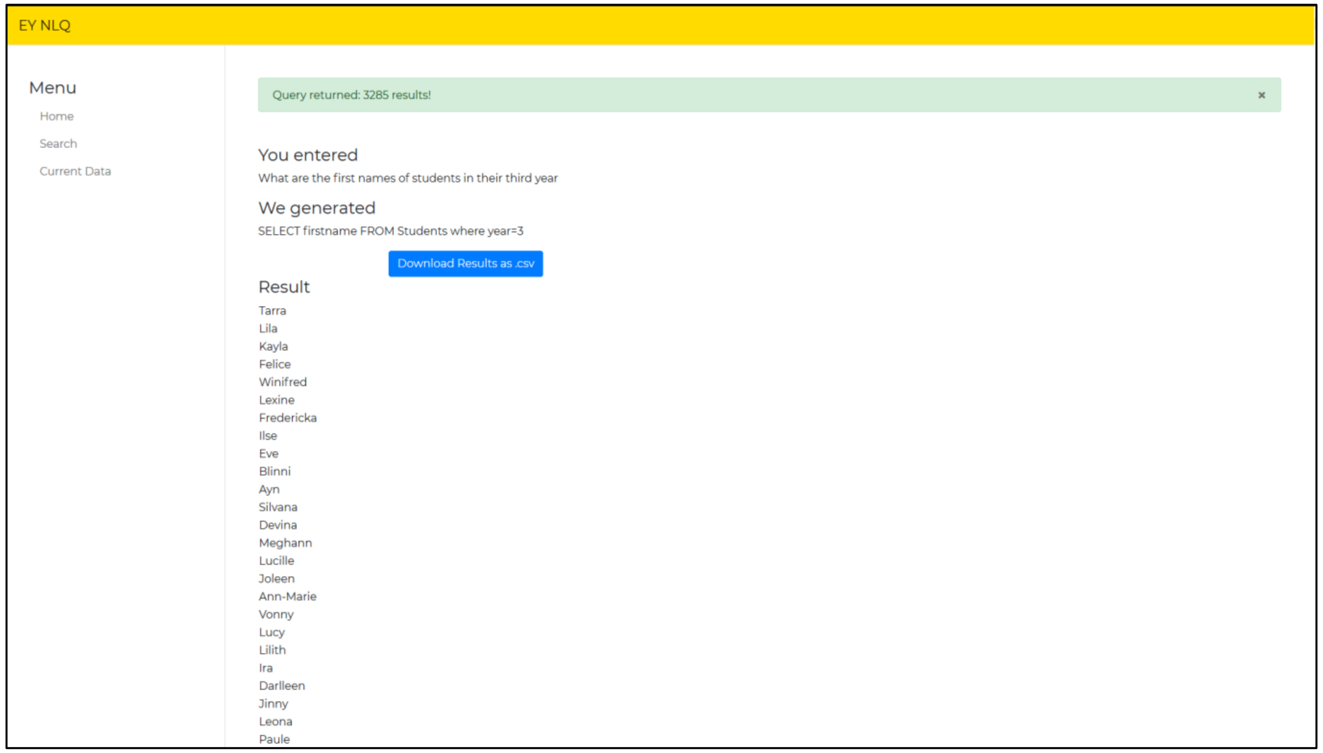
Upon entering a query, the user will be presented with one of two screens. This section will focus on the first of those screens – the Results screen.

The Results screen will be displayed when a user entered query has been successfully run.

At the top of the screen, the user will be presented with a success box detailing the number of results returned by the entered query.

Below that, the system will display what the user entered before displaying the SQL statement generated by the project. This is useful for developers and more advanced users to determine if the project is accurately displaying the correct results.

Under the **Result** heading is the information from the database that is being returned by the project. A blue button above the results allows users to save a copy of that query's output to a .csv file.



Query returned: 3285 results!

You entered

What are the first names of students in their third year

We generated

SELECT firstname FROM Students where year=3

Result

Tarra
Lila
Kayla
Felice
Winifred
Lexine
Fredericka
Ilse
Eve
Blinni
Ayn
Silvana
Devina
Meghann
Lucille
Joleen
Ann-Marie
Vonny
Lucy
Lilith
Ira
Darlleen
Jinny
Leona
Paule

The Results Screen

Data & Analytics

EY

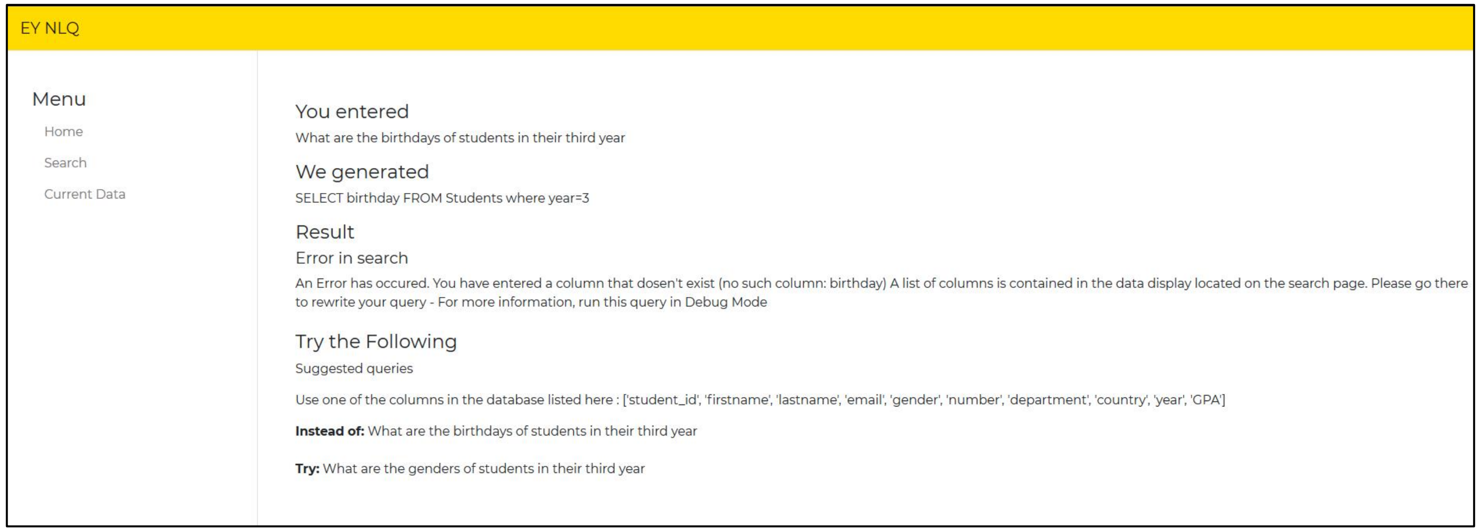
Errors in Queries

Upon entering a query, the user will be presented with one of two screens. This section will focus on the second of those screens – the Error screen.

The Error screen will be displayed if the user entered query is not successful. There are many reasons that a query may be unsuccessful (see the Troubleshooting section for more information).

Like the Results screen, the Error screen will return what the user entered as well as the SQL generated by the system.

Below this is an error message generated by the system that attempts to explain to the user the reason their query was unsuccessful. It then gives the user a suggested solution to their error.



You entered

What are the birthdays of students in their third year

We generated

SELECT birthday FROM Students where year=3

Result

Error in search

An Error has occured. You have entered a column that dosen't exist (no such column: birthday) A list of columns is contained in the data display located on the search page. Please go there to rewrite your query - For more information, run this query in Debug Mode

Try the Following

Suggested queries

Use one of the columns in the database listed here : ['student_id', 'firstname', 'lastname', 'email', 'gender', 'number', 'department', 'country', 'year', 'GPA']

Instead of: What are the birthdays of students in their third year

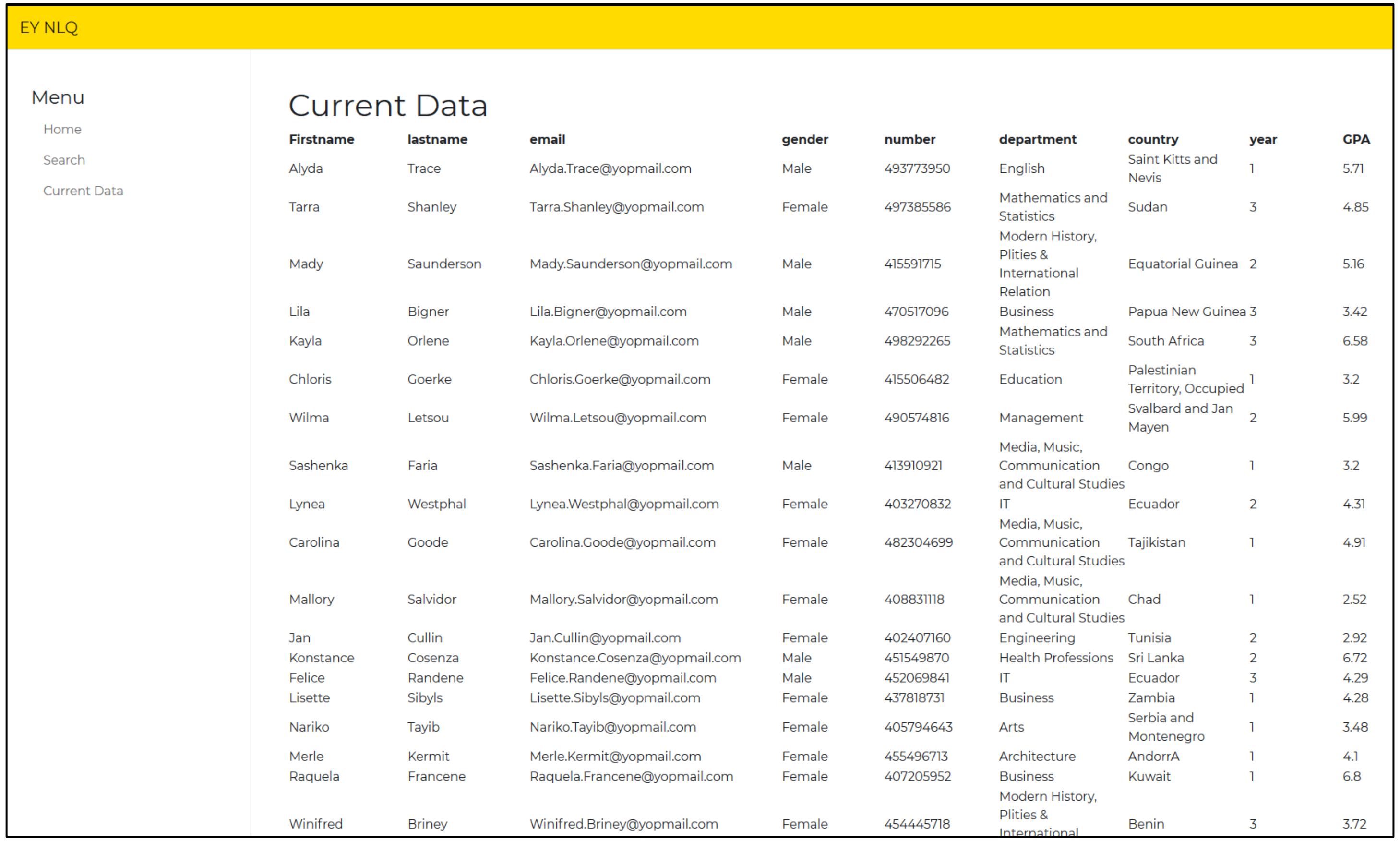
Try: What are the genders of students in their third year

The Error Screen

Data & Analytics

The Current Data Screen

The **Current Data** Screen simply displays all records that make up the database connected to the system. It also shows the names of the columns contained within each of the tables.



Current Data

Firstname	lastname	email	gender	number	department	country	year	GPA
Alyda	Trace	Alyda.Trace@yopmail.com	Male	493773950	English	Saint Kitts and Nevis	1	5.71
Tarra	Shanley	Tarra.Shanley@yopmail.com	Female	497385586	Mathematics and Statistics	Sudan	3	4.85
Mady	Saunderson	Mady.Saunderson@yopmail.com	Male	415591715	Modern History, Plities & International Relation	Equatorial Guinea	2	5.16
Lila	Bigner	Lila.Bigner@yopmail.com	Male	470517096	Business	Papua New Guinea	3	3.42
Kayla	Orlene	Kayla.Orlene@yopmail.com	Male	498292265	Mathematics and Statistics	South Africa	3	6.58
Chloris	Goerke	Chloris.Goerke@yopmail.com	Female	415506482	Education	Palestinian Territory, Occupied	1	3.2
Wilma	Letsou	Wilma.Letsou@yopmail.com	Female	490574816	Management	Svalbard and Jan Mayen	2	5.99
Sashenka	Faria	Sashenka.Faria@yopmail.com	Male	413910921	Media, Music, Communication and Cultural Studies	Congo	1	3.2
Lynea	Westphal	Lynea.Westphal@yopmail.com	Female	403270832	IT	Ecuador	2	4.31
Carolina	Goode	Carolina.Goode@yopmail.com	Female	482304699	Media, Music, Communication and Cultural Studies	Tajikistan	1	4.91
Mallory	Salvidor	Mallory.Salvidor@yopmail.com	Female	408831118	Media, Music, Communication and Cultural Studies	Chad	1	2.52
Jan	Cullin	Jan.Cullin@yopmail.com	Female	402407160	Engineering	Tunisia	2	2.92
Konstance	Cosenza	Konstance.Cosenza@yopmail.com	Male	451549870	Health Professions	Sri Lanka	2	6.72
Felice	Randene	Felice.Randene@yopmail.com	Male	452069841	IT	Ecuador	3	4.29
Lisette	Sibyls	Lisette.Sibyls@yopmail.com	Female	437818731	Business	Zambia	1	4.28
Nariko	Tayib	Nariko.Tayib@yopmail.com	Female	405794643	Arts	Serbia and Montenegro	1	3.48
Merle	Kermit	Merle.Kermit@yopmail.com	Female	455496713	Architecture	AndorrA	1	4.1
Raquela	Francene	Raquela.Francene@yopmail.com	Female	407205952	Business	Kuwait	1	6.8
Winifred	Briney	Winifred.Briney@yopmail.com	Female	454445718	Modern History, Plities & International	Benin	3	3.72

Data & Analytics

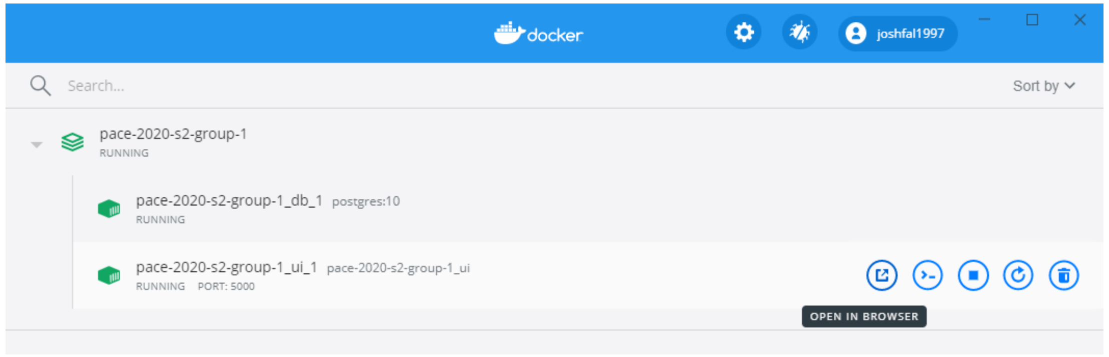
EY

Troubleshooting

Project doesn't open

There may be cases where the project has been installed successfully, however the project is not able to be opened. Users may be presented with a server error when attempting the enter the address: http://localhost:5000 in their internet browser.

The first step is to ensure that the project is running in Docker. Ensure that all modules of the Docker container are green and running (See Figure 6). If instead, the Docker modules are grey and the message Exited is shown underneath, navigate to the Start button and press it to start the program (See Figure 7). The project should start, and all modules should run – see the next page if this is not the case.

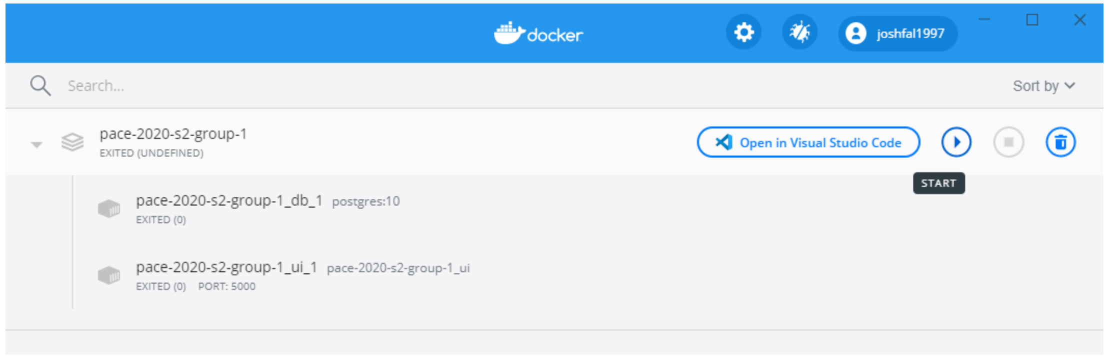


pace-2020-s2-group-1
RUNNING

pace-2020-s2-group-1_db_1 postgres:10
RUNNING

pace-2020-s2-group-1_ui_1 pace-2020-s2-group-1_ui
RUNNING PORT: 5000

Figure 6



pace-2020-s2-group-1
EXITED (UNDEFINED)

pace-2020-s2-group-1_db_1 postgres:10
EXITED (0)

pace-2020-s2-group-1_ui_1 pace-2020-s2-group-1_ui
EXITED (0) PORT: 5000

Figure 7

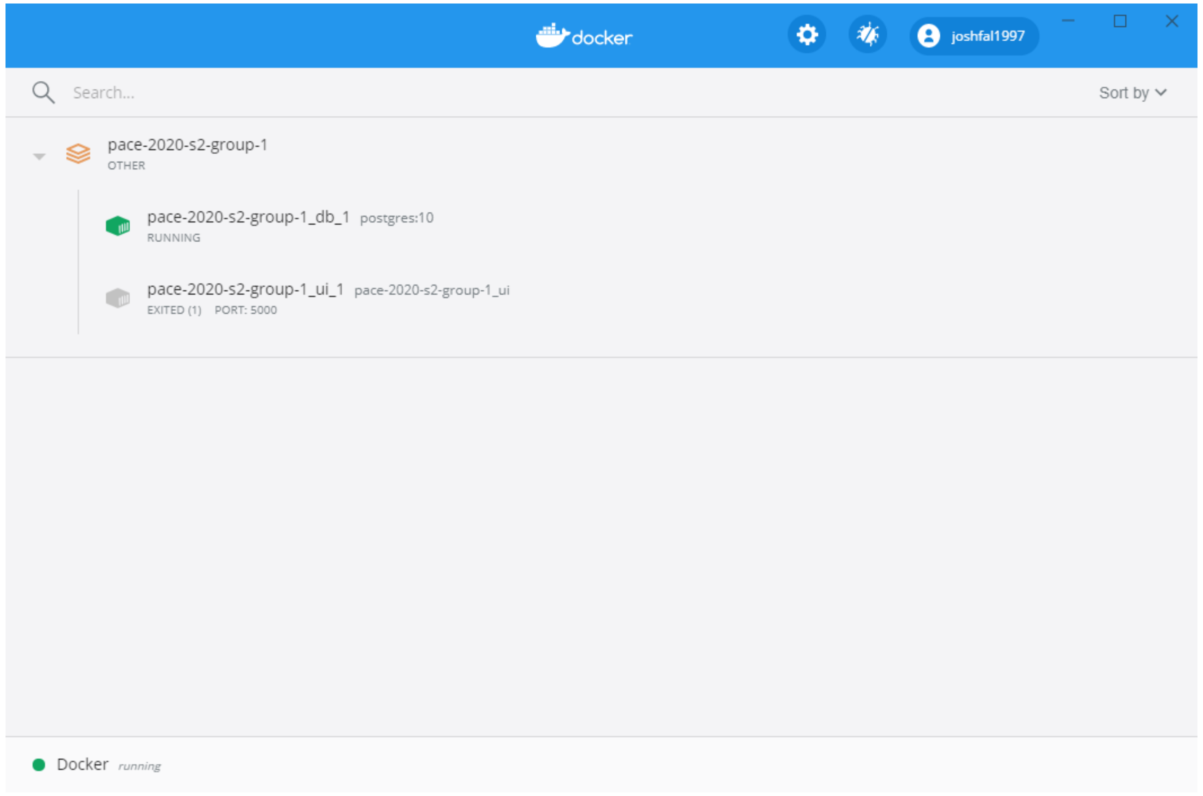
Troubleshooting

Project doesn't open (cont.)

If the project is still not running successfully, it could be that the current build of the project is not functioning as intended. The project within Docker may display an orange icon indicating an error (see Figure 8).

In this case it is best to contact the system administrators for assistance. Advanced users may revert the program to an earlier version through GitHub Desktop or examine the source code of the project through Microsoft Visual Studio to determine the issue.

However, these advanced alternatives are not recommended unless the user has a thorough understanding of the project.



pace-2020-s2-group-1
OTHER
pace-2020-s2-group-1_db_1 postgres:10
RUNNING
pace-2020-s2-group-1_ui_1 pace-2020-s2-group-1_ui
EXITED (1) PORT: 5000
Docker running

Figure 8

Troubleshooting

Generated SQL does not match user query

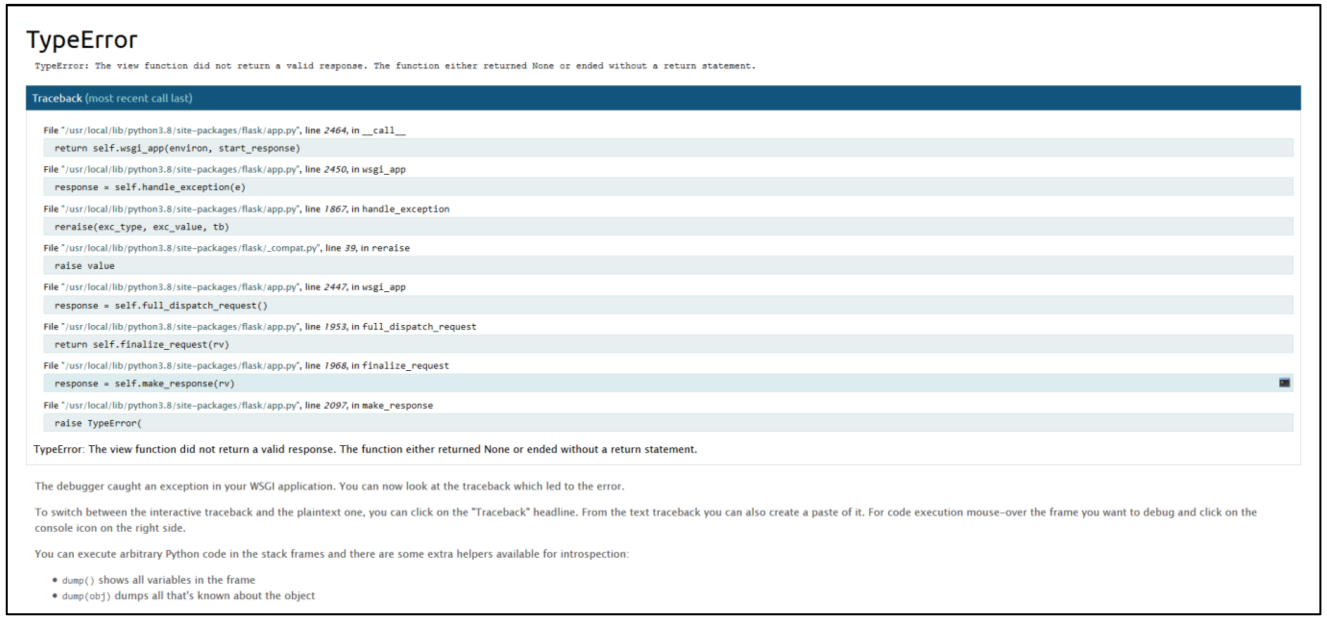
In some situations a user entered query may execute without raising an error, however the results displayed are incorrect. In this situation the system is incorrectly reading the user query and generating the incorrect SQL statement.

In this situation, users can attempt to reword their query. Alternatively users can contact the system administrators for assistance as it could be that the issue is within the system itself and not in the user query.

Fixing errors in queries

To address issues within queries, users should carefully read the error message displayed on the Error Screen. They should then try to implement the solutions given by the system. Users should also try rewording their queries.

For more advanced users, there is a Debug Mode built into the project that will allow the system to display the error messages directly from the database. This provides more useful information for troubleshooting but is less user-friendly for beginners (see Figure 9).
To toggle Debug Mode, users must access the project's source code and change the DebugMode variable within the App.py file to True.



```
TypeError
TypeError: The view function did not return a valid response. The function either returned None or ended without a return statement.

Traceback (most recent call last)

File "/usr/local/lib/python3.8/site-packages/flask/app.py", line 2464, in __call__
  return self.wsgi_app(environ, start_response)
File "/usr/local/lib/python3.8/site-packages/flask/app.py", line 2450, in wsgi_app
  response = self.handle_exception(e)
File "/usr/local/lib/python3.8/site-packages/flask/app.py", line 1867, in handle_exception
  reraise(exc_type, exc_value, tb)
File "/usr/local/lib/python3.8/site-packages/flask/_compat.py", line 39, in reraise
  raise value
File "/usr/local/lib/python3.8/site-packages/flask/app.py", line 2447, in wsgi_app
  response = self.full_dispatch_request()
File "/usr/local/lib/python3.8/site-packages/flask/app.py", line 1953, in full_dispatch_request
  return self.finalize_request(rv)
File "/usr/local/lib/python3.8/site-packages/flask/app.py", line 1968, in finalize_request
  response = self.make_response(rv)
File "/usr/local/lib/python3.8/site-packages/flask/app.py", line 2097, in make_response
  raise TypeError(
TypeError: The view function did not return a valid response. The function either returned None or ended without a return statement.
```

The debugger caught an exception in your WSGI application. You can now look at the traceback which led to the error.

To switch between the interactive traceback and the plaintext one, you can click on the "Traceback" headline. From the text traceback you can also create a paste of it. For code execution mouse-over the frame you want to debug and click on the console icon on the right side.

You can execute arbitrary Python code in the stack frames and there are some extra helpers available for introspection:

- `dump()` shows all variables in the frame
- `dump(obj)` dumps all that's known about the object

Figure 9