Group A: Corrine Villany, Steven Prokop, Matthew Reese
Advanced Python For Business
Final Report
May 3, 2024

Final Report Group A

Introduction:

Composed of 16 universities that offer both undergraduate and graduate opportunities with varying majors across the board, the University of North Carolina System is one of the most well-respected public higher education systems in the United States. According to CollegeBoard, a well-known organization for high school students, most applicants should have “about five to eight applications” for different colleges. With most students applying to numerous universities, we thought it would be best to build something that helps incoming first-year students make the right enrollment decision for them. With over 500,000 students applying to the UNC System every year and most of them applying to multiple schools within the system, our team further believed there was room to build a program supporting UNC-specific applicants. The program intends to allow these students to view the trends in school-specific majors over time and assist them in their college decision-making process. Our group has created an application that will enable students to visualize trends of enrollment data across all schools and both degree levels, truly optimizing the speed and personalization of trends in enrollment data.

Statement of Problem:

It is common for prospective students to be shown outdated or vague numbers when going onto an undergraduate university’s enrollment information page. The data on some school’s pages, such as our undergraduate program at Fordham University, may only show information from the previous class’ demographics and acceptance rate. Students seeking historical data to assist them in decision-making may be out of luck if they only browse a school's webpage. Additionally, prospective students may have to worry about a paywall or

restricted access when trying to view enrollment information. The blockage of data can hurt students by not allowing for proper research of schools and impact the schools by potentially being ruled out by a student before they even check the numbers. If a school's enrollment data cannot be accessed by a student beforehand, the student may not even consider applying, destroying an opportunity before it is born. However, our group is looking to improve this conflict by allowing students to read multiple years of enrollment data to understand the bigger picture of the UNC School System.

Approach to Problem:

To start, our group needed to understand the domain of our sample data. We looked to answer three main questions: how can we create a comparable data visualization for multiple schools, what pieces of our data frame do we want to incorporate in our work, and what is the best format for our application? Our group has been researching undergraduate and graduate level degrees in our sample of the UNC system awarded over time to analyze trending information that students can use strategically when applying to various colleges. Taking in data from more than just the past year will allow college applicants to truly understand the direction a school is heading towards. For example, a school may have more than half of its students accepting a Bachelor of Science degree. Still, multiple years of data may suggest that the total percentage and enrollment of students receiving a Bachelor of Science degree is declining while a Bachelor of Arts degree is rising. The direction of this project has emphasized helping users understand the trends for UNCs' degree statistics. While we have the baseline code set up and ready for use, our group is moving towards putting this code into an application that can allow students to use it at any time for any university. The code allows for previous years of UNC system statistics to be analyzed and allows for easy access to input updated data as the years go

on. Overall, our approach to the problem was to take advantage of existing, publicly available information and capitalize on a large amount of data by creating a sample application to build for students.

Data

The project required a vast amount of data to work with so our answers could appeal to a mass market rather than a niche group of students. Fortunately for us, the University of North Carolina System has a file directory of numbers for previous years with enrollment data in categories such as majors, degree levels, and specific universities. The data comes from a public CSV file that was accessible through the UNC system website. The file includes 30 columns and over 16,000 data entries that are unique to review and compile for our program. With this extensive amount of data, our group could select specific columns we wanted to work with in our application. The data organization is the columns “Institution Code,” “Student Type,” “Level 1 Field of Study”, “2018-2019 Enrollment”, and “2019-2020 Enrollment.” The columns were carefully selected to ensure our model captured the most essential information. The data used in our application will help reflect multiple years of enrollment numbers and 16 different schools to choose from when measuring and comparing information. In summary, our data is designed to be easy to interpret and stay relevant for the user.

Model:

Our group decided for the model to be an application, as stated before. The application will include the results from our code implemented in the files for our model. There are two viewpoints for our model. First is the developed side we worked on, including the Python code and streamlit functions. Next is a local host, which contains the user interface that applicants will see. The code we produced is similar to any Python scripts we have done in class. The code

comprises imported packages, code lines with statements, and comments for readability if anyone needs to review the development side. Some of the key calculations and analysis used in the Python script include making input statements for the user to select schools, degree levels, and majors by following keywords. From the answers that users create, similar to a flowchart, the code then runs through if-statements to determine which graphs will be displayed for a user from all answers given to the model. For the graphs and charts, our group developed Matplotlib.pyplot codes to create charts that will develop quickly so the user can easily understand them and gain insights. With all the code our group created, we then ran the implementation phase through Streamlit to create a local host site.

Implementation:

The user interface is a more cosmetically pleasing website containing the NC system logo, drop-down bars for users to input schools and degree levels, and information on majors, which users can select from to display customized charts. There are numerous charts on the local host site, including a custom bar chart showing enrollments for a specific school, degree level, and major the user selects. The chart is followed by two plots, including a scatterplot with a multi-school analysis of enrollment and a collective bar chart that shows comparative data across the 16 universities. We believe that these charts will be able to help students understand and read data at a speed faster than the Internet can provide them currently. Overall, our setup of the local host site is our final implementation, and it can take all of our information and make it ready for users to obtain insights on school enrollment data.

Results:

The data, Python script, and local host have all been successful for us to work with. Over time, we decided on various universities to use for our data and found that the UNC system

provided a great variety of school majors and degree levels for data to be stored publicly. We found from our model and implementation that it is much easier to decide which data can be used based on whether it is public information or not. Our model was successful with the UNC student summary CSV and worked with other previous files that were public. Private information is challenging to access and potentially has a monetary value compared to all public info, which can be stored in our model for future enhancements. Another result that followed our model and implementation is a great UI design for our website. The early drafts of our project successfully displayed charts and data but from the perspective of an .ipynb file through Jupyter Notebook. Once we moved to Streamlit, our user interface was more like an actual website, giving the user a cleaner view for their experience. The following is a link to a screen-recorded video of how the program works on the localhost. Lastly, following the link are screenshots of our designs in code.

https://www.loom.com/share/1bc8233a8fed46fd9a6d94227e188998?sid=29aa3327-af8f-4ed8-8806-8801e5468610

	A	B	C	D	E	F	G	H	I
1	Institution	Institution Code	Degree Level	Student Ty	Residency	Level 1 Field of Study	Level 2 Fie	2018-201	2019-202
2	All UNC System Institutions	All UNC System	Undergraduate	All Underg	All	Agricultural/Animal/Plant/Veterinary Science And Related Fields	All	2564	2565
3	All UNC System Institutions	All UNC System	Undergraduate	All Underg	All	Agricultural/Animal/Plant/Veterinary Science And Related Fields	Agribusine	368	375
4	All UNC System Institutions	All UNC System	Undergraduate	All Underg	All	Agricultural/Animal/Plant/Veterinary Science And Related Fields	Agricultura	65	69
5	All UNC System Institutions	All UNC System	Undergraduate	All Underg	All	Agricultural/Animal/Plant/Veterinary Science And Related Fields	Agriculture	271	248
6	All UNC System Institutions	All UNC System	Undergraduate	All Underg	All	Agricultural/Animal/Plant/Veterinary Science And Related Fields	Agronomy	106	111
7	All UNC System Institutions	All UNC System	Undergraduate	All Underg	All	Agricultural/Animal/Plant/Veterinary Science And Related Fields	Animal Sci	1276	1281
8	All UNC System Institutions	All UNC System	Undergraduate	All Underg	All	Agricultural/Animal/Plant/Veterinary Science And Related Fields	Animal Sci	66	64
9	All UNC System Institutions	All UNC System	Undergraduate	All Underg	All	Agricultural/Animal/Plant/Veterinary Science And Related Fields	Food Scier	72	64
10	All UNC System Institutions	All UNC System	Undergraduate	All Underg	All	Agricultural/Animal/Plant/Veterinary Science And Related Fields	Food Scier	104	80
11	All UNC System Institutions	All UNC System	Undergraduate	All Underg	All	Agricultural/Animal/Plant/Veterinary Science And Related Fields	Horticultu	121	138
12	All UNC System Institutions	All UNC System	Undergraduate	All Underg	All	Agricultural/Animal/Plant/Veterinary Science And Related Fields	Poultry Sci	94	87
13	All UNC System Institutions	All UNC System	Undergraduate	All Underg	All	Agricultural/Animal/Plant/Veterinary Science and Related Fields	Agricultura	11	9
14	All UNC System Institutions	All UNC System	Undergraduate	All Underg	All	Agricultural/Animal/Plant/Veterinary Science and Related Fields	Agroecolo	25	51
15	All UNC System Institutions	All UNC System	Undergraduate	All Underg	All	Agricultural/Animal/Plant/Veterinary Science and Related Fields	Soil Scienc	5	2
16	All UNC System Institutions	All UNC System	Undergraduate	All Underg	All	Agricultural/Animal/Plant/Veterinary Science and Related Fields	Turf and Tu	23	26
17	All UNC System Institutions	All UNC System	Undergraduate	All Underg	All	All Other Programs (Degree Seeking)	All	1478	1367
18	All UNC System Institutions	All UNC System	Undergraduate	All Underg	Out-of-Sta	Agricultural/Animal/Plant/Veterinary Science And Related Fields	Agricultura	0	1
19	All UNC System Institutions	All UNC System	Undergraduate	All Underg	All	Architecture And Related Services	All	511	530
20	All UNC System Institutions	All UNC System	Undergraduate	All Underg	All	Architecture And Related Services	Architectu	246	254
21	All UNC System Institutions	All UNC System	Undergraduate	All Underg	All	Architecture And Related Services	City/Urbar	73	77
22	All UNC System Institutions	All UNC System	Undergraduate	All Underg	All	Architecture And Related Services	Environme	138	150
23	All UNC System Institutions	All UNC System	Undergraduate	All Underg	All	Architecture and Related Services	Landscape	54	49
24	All UNC System Institutions	All UNC System	Undergraduate	All Underg	All	Area, Ethnic, Cultural, Gender, And Group Studies	All	816	795

Figure 1

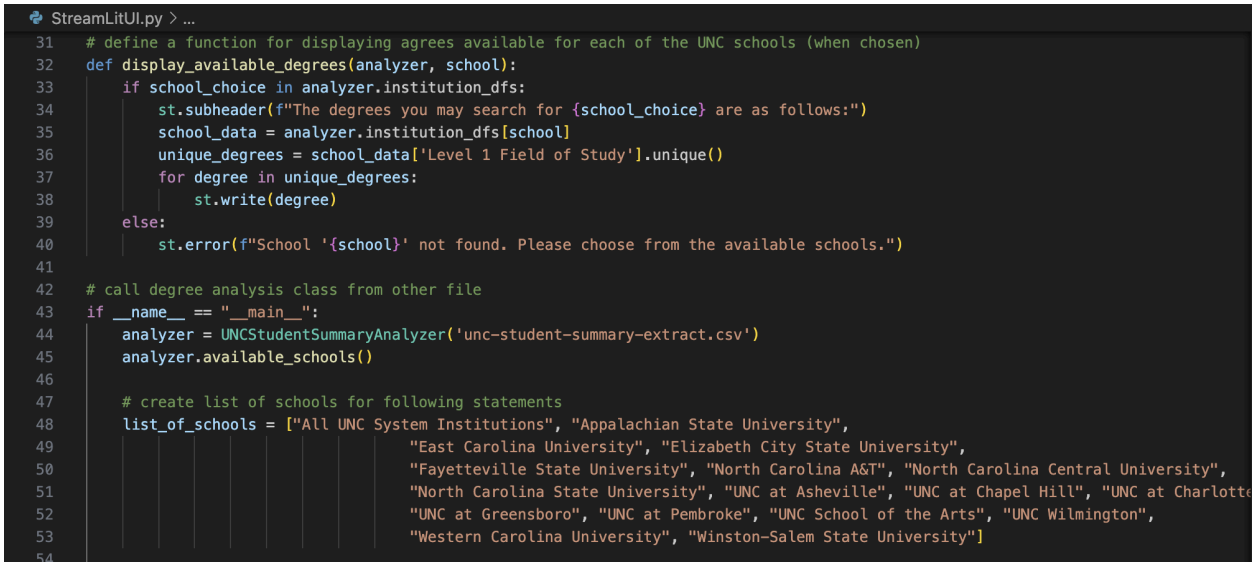
Figure 1 displays the first 24 of over 16,000 rows of data in our student summary extract CSV.

StreamLitUI.py > ...

```python
import streamlit as st
import matplotlib.pyplot as plt
from UNCStudentSummaryAnalyzerClass import UNCStudentSummaryAnalyzer
from UNCtextanalysis import DataAnalyzer

# display image logo
st.image('UNC-System-Logo-2.png')

# display page title
st.title("UNC School System Degree Analyzer")

#display page subheader
st.subheader("Welcome to the UNC School System Degree Analyzer!")

# display description paragraph part 1
st.write('''This application provides a comprehensive analysis of student completion data
         for various degrees across UNC system institutions. Users can explore the number
         of completers for specific degrees over different academic years and
         compare the completion rates across different schools within the UNC system.''')

# display other description statements
st.write('Key features of the application include:')
st.write('1) Available Degrees: Users can view a list of available degrees offered by a selected UNC institution.')
st.write('2) Completion Analysis: Users can visualize the number of completers for a chosen degree at a specific institution over
st.write('3) Cross-Institution Comparison: Users can compare the completion rates of a particular degree across different UNC sys
st.write('''With its user-friendly interface and powerful analytical capabilities,
         our UNC Student Summary Analyzer empowers users to gain valuable insights
         into student completion trends, aiding academic decision-making and institutional planning.''')
st.write('Start exploring below to unlock data-driven insights into student completions across the UNC system!')
```

Figure 2

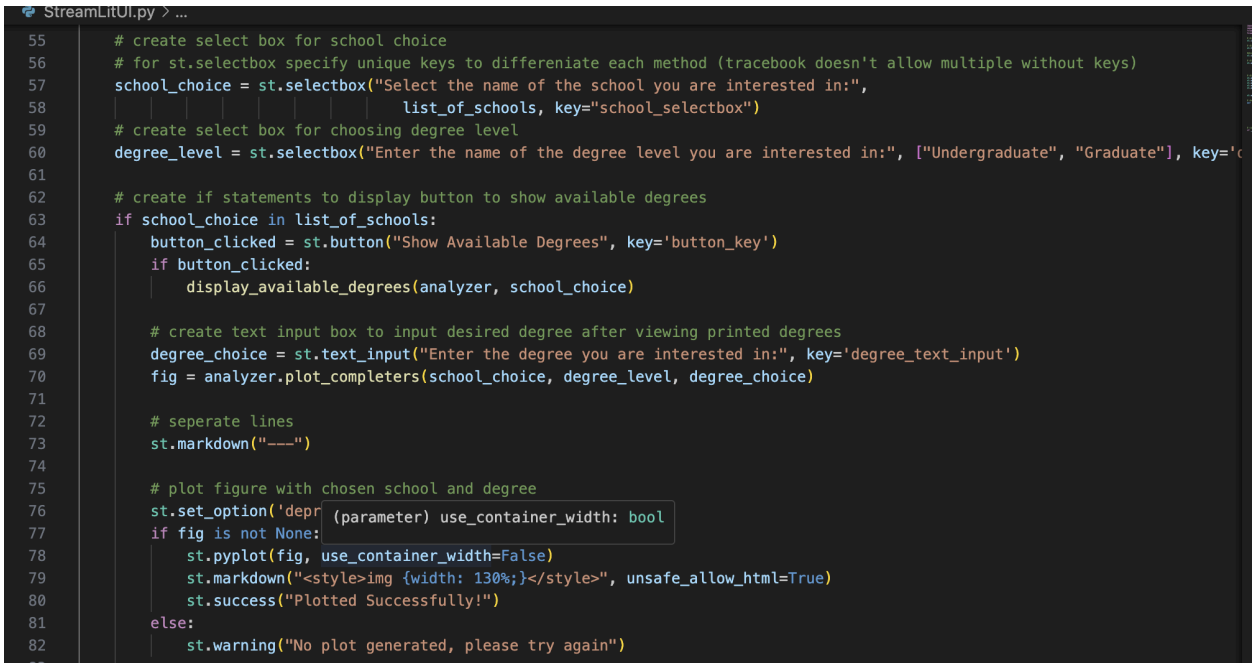
Figure 2 displays our first 29 lines of code with the steps of importing packages, displaying the page title, subheaders, and description statements necessary for the user to understand what the application can do, setting up the steps to call the main class.



```python
# define a function for displaying agrees available for each of the UNC schools (when chosen)
def display_available_degrees(analyzer, school):
    if school_choice in analyzer.institution_dfs:
        st.subheader(f"The degrees you may search for {school_choice} are as follows:")
        school_data = analyzer.institution_dfs[school]
        unique_degrees = school_data['Level 1 Field of Study'].unique()
        for degree in unique_degrees:
            st.write(degree)
    else:
        st.error(f"School '{school}' not found. Please choose from the available schools.")

# call degree analysis class from other file
if __name__ == "__main__":
    analyzer = UNCStudentSummaryAnalyzer('unc-student-summary-extract.csv')
    analyzer.available_schools()

    # create list of schools for following statements
    list_of_schools = ["All UNC System Institutions", "Appalachian State University",
                       "East Carolina University", "Elizabeth City State University",
                       "Fayetteville State University", "North Carolina A&T", "North Carolina Central University",
                       "North Carolina State University", "UNC at Asheville", "UNC at Chapel Hill", "UNC at Charlotte
                       "UNC at Greensboro", "UNC at Pembroke", "UNC School of the Arts", "UNC Wilmington",
                       "Western Carolina University", "Winston-Salem State University"]
```

Figure 3

Figure 3 continues with the previous code by calling the main class. In addition, it sets up parameters to help the user see the select and text input boxes on the application.



```python
    # create select box for school choice
    # for st.selectbox specify unique keys to differeniate each method (tracebook doesn't allow multiple without keys)
    school_choice = st.selectbox("Select the name of the school you are interested in:",
                                 list_of_schools, key="school_selectbox")
    # create select box for choosing degree level
    degree_level = st.selectbox("Enter the name of the degree level you are interested in:", ["Undergraduate", "Graduate"], key='

    # create if statements to display button to show available degrees
    if school_choice in list_of_schools:
        button_clicked = st.button("Show Available Degrees", key='button_key')
        if button_clicked:
            display_available_degrees(analyzer, school_choice)

        # create text input box to input desired degree after viewing printed degrees
        degree_choice = st.text_input("Enter the degree you are interested in:", key='degree_text_input')
        fig = analyzer.plot_completers(school_choice, degree_level, degree_choice)

        # seperate lines
        st.markdown("---")

        # plot figure with chosen school and degree
        st.set_option('depr (parameter) use_container_width: bool
        if fig is not None:
            st.pyplot(fig, use_container_width=False)
            st.markdown("<style>img {width: 130%;}</style>", unsafe_allow_html=True)
            st.success("Plotted Successfully!")
        else:
            st.warning("No plot generated, please try again")
```

Figure 4

Figure 4 continues on by displaying the web page's dropdown boxes/choices, creating a loop to show the available degrees (per school chosen) after clicking a button, and graphing this degree if the figure has data to show.

```
        # create button to allow user to see comparison chart
        # call get comparison data and write if statement setting up the graph
        show_comparison = st.button("Show Comparison Chart", key='comparison_button')
        if show_comparison:
            compare_df = analyzer.get_comparison_data(school_choice, degree_level, degree_choice)
            if not compare_df.empty:
                fig_compare, ax_compare = plt.subplots(figsize=(10, 8))
                compare_df.set_index('School').plot(kind='barh', ax=ax_compare, legend=True, color=['blue', 'orange'])
                ax_compare.set_ylabel('School')
                ax_compare.set_xlabel('Completers')
                ax_compare.set_title(f'Comparison of {degree_choice} Completers in {degree_level} Across Schools')
                ax_compare.tick_params(axis='y', which='both', left=False, right=True)  # Show ticks only on the right side
                plt.xticks(rotation=45, ha='right')
                plt.tight_layout()
                st.pyplot(fig_compare)
                st.markdown("<style>img {width: 150%;}</style>", unsafe_allow_html=True)
    else:
        st.warning('No comparison data available')

        st.warning("No plot generated. Please check your inputs and try again.")

    # seperate lines
    st.markdown("---")

    # Display Stagnant Chart
    analyzer2 = DataAnalyzer('unc-student-summary-extract.csv')
    fig, ax = plt.subplots()
    fig2 = analyzer2.plot_completers_scatter_comparison()
    st.pyplot(fig2, use_container_width=False)
    st.markdown("<style>img {width: 160%;}</style>", unsafe_allow_html=True)
```

Figure 5

Lastly, Figure 5 shows the call to get_data_comparison for the user to see a comparison of the specified degree throughout all UNC schools that offer it. In addition, we wanted to display a stagnant scatter plot chart at the bottom so the user could see changes over time.

Lessons Learned:

The project was very interesting, and our group enjoyed figuring out how our project could work best/what was needed for improvement. Our project works best with a file that stores information about numerous schools instead of developing multiple Python notebooks for every individual school. From this, we learned that when the time comes to implement this application for more schools, we will need to adjust the code based on file sizes, column names, and data that users want to look at most. Our project also had many visualizations to pick from, and our

final application used only the ones displayed on the local host site. For example, while we were able to use the “plt” method to show visualizations on Juptyer Notebook, running visualizations through Streamlit required designing graphs from scratch using the “ax” method. In addition, we had previously included an additional multi-school scatterplot and a text analysis line chart for majors in the school system. Still, we decided that such charts were not ready for display. In a newer version of the application, there is a possibility of bringing these back to visualization and not just code. Lastly, we learned that our subject of enrollment data and school fields of study information is particular and varies a ton by school. Therefore, we created in our application the order of picking a school first and then having school-specific majors appear on the screen. To summarize, the project taught many lessons and will allow us to adapt when updates need to be made.