1.0 INTRODUCTION

Every year there are over 2,000+ matches played in europe. Many factors determine the outcome of these matches. Some of these factors include players, teams, infrastructure, strategy etc. Making a prediction on a match outcome is quite difficult due to many factors to be put into consideration.

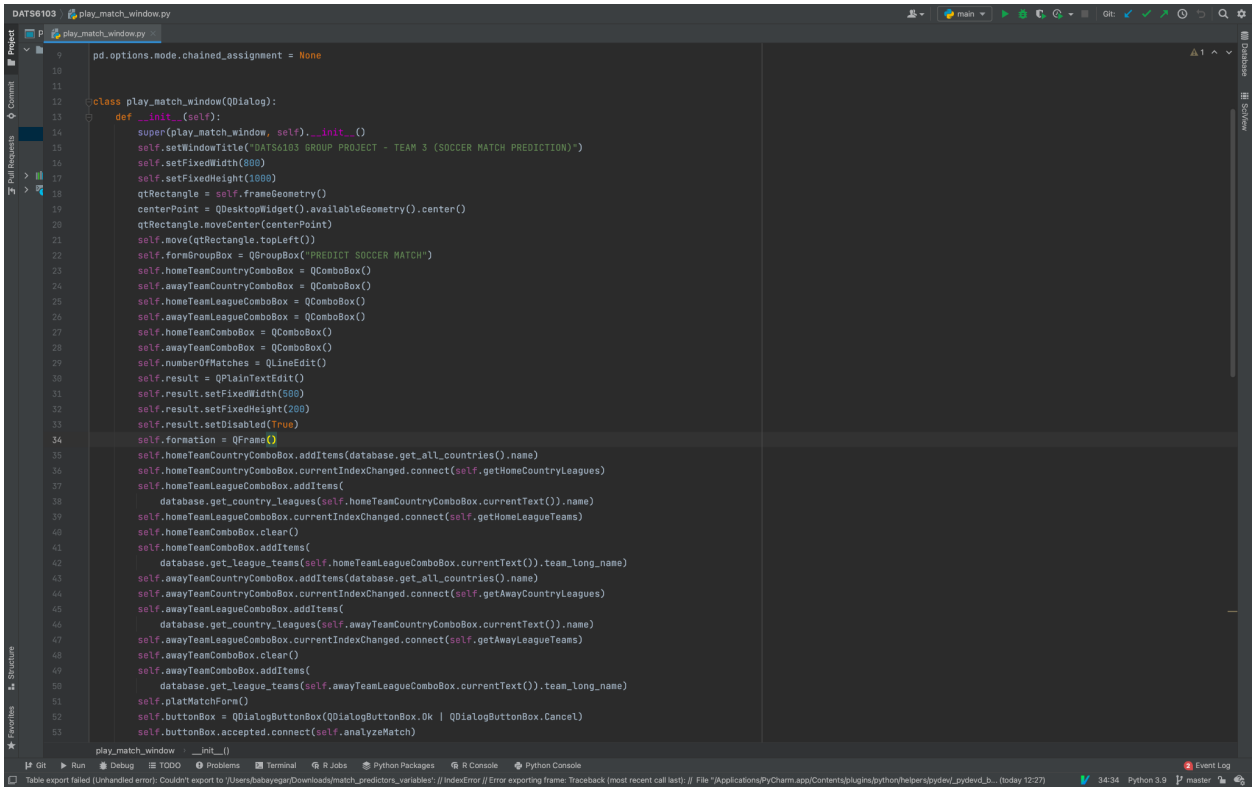
The project is supposed to analyze previous matches played and predict future matches between two teams.Since the teams are the target, we will build our predictors based on the teams statistics and features.We will design a GUI that will enable a user selected both the home and away team and generate the predictors based on the selected teams to predict the match outcome.

I will be designing the GUI with the required controls we need to select the home and away teams for prediction. Also I will be adding controls to display the results of each match outcome.The GUI will be designed using pyQt5 and the GUI will be populated using pandas.Some additional results like the team formations will be displayed to give the soccer fans a more related and genuine experience.

Lastly the decision tree will be used to predict the match outcome based on analysis and training of past matches.The match outcome will either be a Home Team Win, Draw or a Away Team Win.

2.0 WORK DESCRIPTION

2.1 GUI Code 1



```python
pd.options.mode.chained_assignment = None


class play_match_window(QDialog):
    def __init__(self):
        super(play_match_window, self).__init__()
        self.setWindowTitle("DATS6103 GROUP PROJECT - TEAM 3 (SOCCER MATCH PREDICTION)")
        self.setFixedWidth(800)
        self.setFixedHeight(1000)
        qtRectangle = self.frameGeometry()
        centerPoint = QDesktopWidget().availableGeometry().center()
        qtRectangle.moveCenter(centerPoint)
        self.move(qtRectangle.topLeft())
        self.formGroupBox = QGroupBox("PREDICT SOCCER MATCH")
        self.homeTeamCountryComboBox = QComboBox()
        self.awayTeamCountryComboBox = QComboBox()
        self.homeTeamLeagueComboBox = QComboBox()
        self.awayTeamLeagueComboBox = QComboBox()
        self.homeTeamComboBox = QComboBox()
        self.awayTeamComboBox = QComboBox()
        self.numberOfMatches = QLineEdit()
        self.result = QPlainTextEdit()
        self.result.setFixedWidth(500)
        self.result.setFixedHeight(200)
        self.result.setDisabled(True)
        self.formation = QFrame()
        self.homeTeamCountryComboBox.addItems(database.get_all_countries().name)
        self.homeTeamCountryComboBox.currentIndexChanged.connect(self.getHomeCountryLeagues)
        self.homeTeamLeagueComboBox.addItems(
            database.get_country_leagues(self.homeTeamCountryComboBox.currentText()).name)
        self.homeTeamLeagueComboBox.currentIndexChanged.connect(self.getHomeLeagueTeams)
        self.homeTeamComboBox.clear()
        self.homeTeamComboBox.addItems(
            database.get_league_teams(self.homeTeamLeagueComboBox.currentText()).team_long_name)
        self.awayTeamCountryComboBox.addItems(database.get_all_countries().name)
        self.awayTeamCountryComboBox.currentIndexChanged.connect(self.getAwayCountryLeagues)
        self.awayTeamLeagueComboBox.addItems(
            database.get_country_leagues(self.awayTeamCountryComboBox.currentText()).name)
        self.awayTeamLeagueComboBox.currentIndexChanged.connect(self.getAwayLeagueTeams)
        self.awayTeamComboBox.clear()
        self.awayTeamComboBox.addItems(
            database.get_league_teams(self.awayTeamLeagueComboBox.currentText()).team_long_name)
        self.platMatchForm()
        self.buttonBox = QDialogButtonBox(QDialogButtonBox.Ok | QDialogButtonBox.Cancel)
        self.buttonBox.accepted.connect(self.analyzeMatch)
```

a. I first declared the class and its constructor. I also defined the window title, fixed height and fixed width to accommodate the form controls appropriately. I used the frame geometry to centralize the form when executed on any computer. I declared a group box control to embed all other controls inside it.

b. I used the combo box control to create the following dropdowns

- Home Team Country Combo
- Home Team League Combo
- Home Team Combo
- Away Team Country Combo
- Away Team League Combo

- Away Team Combo

I also created a plan edit field to display the results of the predicted model and I also sized it appropriately and disabled it to make it readonly.

I added a frame control to display the team's formation plot.

c. After creating all the controls, populated all the country, league and team combo with data from the dataset using sql queries in the database.py file. When a country is selected it triggers and populates the league combo and when a league is selected it triggers and populates the team combo

d. I added a button group of default pyqt5 buttons which consists of the cancel and ok buttons. The cancel button closes the form and the ok button submits the form to perform the model analysis and prediction.

2.2 GUI Code 2

```python
        self.buttonBox = QDialogButtonBox(QDialogButtonBox.Ok | QDialogButtonBox.Cancel)
        self.buttonBox.accepted.connect(self.analyzeMatch)
        self.buttonBox.rejected.connect(self.reject)
        self.figure = plt.figure()
        self.canvas = FigureCanvas(self.figure)
        mainLayout = QVBoxLayout()
        mainLayout.addWidget(self.formGroupBox)
        mainLayout.addWidget(self.buttonBox)
        mainLayout.addWidget(self.canvas)
        self.setLayout(mainLayout)

    def getHomeCountryLeagues(self):
        self.homeTeamLeagueComboBox.clear()
        self.homeTeamComboBox.clear()
        self.homeTeamLeagueComboBox.addItems(
            database.get_country_leagues(self.homeTeamCountryComboBox.currentText()).name)

    def getAwayCountryLeagues(self):
        self.awayTeamLeagueComboBox.clear()
        self.awayTeamComboBox.clear()
        self.awayTeamLeagueComboBox.addItems(
            database.get_country_leagues(self.awayTeamCountryComboBox.currentText()).name)

    def getHomeLeagueTeams(self):
        self.homeTeamComboBox.clear()
        self.homeTeamComboBox.addItems(
            database.get_league_teams(self.homeTeamLeagueComboBox.currentText()).team_long_name)

    def getAwayLeagueTeams(self):
        self.awayTeamComboBox.clear()
        self.awayTeamComboBox.addItems(
            database.get_league_teams(self.awayTeamLeagueComboBox.currentText()).team_long_name)

    def analyzeMatch(self):
        home_team = str(self.homeTeamComboBox.currentText())
        away_team = str(self.awayTeamComboBox.currentText())
        number_of_matches = int(self.numberOfMatches.text())
        matches = database.get_home_away_teams_matches(home_team, away_team, number_of_matches)
        self.figure.clear()
        team_formations.team_formation(home_team, away_team)
        self.canvas.draw()
        match_predictors_variables = match_predictors.create_match_predictors(matches,number_of_matches)
        match_predictors_variables = match_predictors_variables.drop('match_api_id', axis=1)
        print(match_predictors_variables)
```

a. I created box layout as the main form layout and i added the form group containing the controls, the button box containing the ok and cancel buttons and the canvas to display the team formation as widgets to the main layout

b. getHomeCountryLeagues(): This method is triggered whenever a home team country is changed. It returns a dataframe of the list of leagues for the home team.

c. getAwayCountryLeagues(): This method is triggered whenever a away team country is changed. It returns a dataframe of the list of leagues for the away team.

d. getHomeLeagueTeams: This method is triggered whenever a home team league is changed. It returns a dataframe of the list of teams in that league for the home team.

e. getAwayLeagueTeams(): This method is triggered whenever a away team league is changed. It returns a dataframe of the list of teams in that league for the away team.

f. analyzeMatch() : This method is triggered when the ok button is clicked. In this method the home and away team selected are collected from the form and passed to the get_home_away_team_matches which is also a method that gets the last n number of matches of both the home and away team played in their respective leagues. Next, the team formations plot is displayed in the canvas created earlier. And based on the teams selected the target variables and predictor variables are generated and passed to the prediction model.

2.3 GUI Code 3

```python
        self.awayTeamLeagueComboBox.clear()
        self.awayTeamComboBox.clear()
        self.awayTeamLeagueComboBox.addItems(
            database.get_country_leagues(self.awayTeamCountryComboBox.currentText()).name)

    def getHomeLeagueTeams(self):
        self.homeTeamComboBox.clear()
        self.homeTeamComboBox.addItems(
            database.get_league_teams(self.homeTeamLeagueComboBox.currentText()).team_long_name)

    def getAwayLeagueTeams(self):
        self.awayTeamComboBox.clear()
        self.awayTeamComboBox.addItems(
            database.get_league_teams(self.awayTeamLeagueComboBox.currentText()).team_long_name)

    def analyzeMatch(self):
        home_team = str(self.homeTeamComboBox.currentText())
        away_team = str(self.awayTeamComboBox.currentText())
        number_of_matches = int(self.numberOfMatches.text())
        matches = database.get_home_away_teams_matches(home_team, away_team, number_of_matches)
        self.figure.clear()
        team_formations.team_formation(home_team, away_team)
        self.canvas.draw()
        match_predictors_variables = match_predictors.create_match_predictors(matches,number_of_matches)
        match_predictors_variables = match_predictors_variables.drop('match_api_id', axis=1)
        print(match_predictors_variables)

    def platMatchForm(self):
        layout = QFormLayout()
        layout.addRow(QLabel("Home Team Country"), self.homeTeamCountryComboBox)
        layout.addRow(QLabel("Home Team League"), self.homeTeamLeagueComboBox)
        layout.addRow(QLabel("Home Team"), self.homeTeamComboBox)
        layout.addRow(QLabel("VS"))
        layout.addRow(QLabel("Away Team Country"), self.awayTeamCountryComboBox)
        layout.addRow(QLabel("Away Team League"), self.awayTeamLeagueComboBox)
        layout.addRow(QLabel("Away Team"), self.awayTeamComboBox)
        layout.addRow(QLabel("Number of Matches to Analyze"), self.numberOfMatches)
        layout.addRow(QLabel("Prediction Results"), self.result)
        self.formGroupBox.setLayout(layout)
```

a. The play_match_form() is the method that finally displays the form(box layout).To make the GUI more readable I added labels for each control. I used the add row also to make the controls uniform and easy to navigate. When the labels are called, it also calls its respective text edit or combo defined earlier.

2.4 Data Preprocessing

```python
con = sqlite3.connect("soccer.sqlite")
cur = con.cursor()


def get_all_countries():
    return pd.read_sql('SELECT * FROM Country;', con)

def get_country_leagues(country_name):
    query = f'SELECT DISTINCT l.name FROM League l ' \
            f'JOIN Country c ON l.country_id = c.id ' \
            f'WHERE c.name =  "{country_name}" ORDER BY l.name ASC'
    return pd.read_sql(query, con)

def get_league_teams(league_name):
    query = f'SELECT DISTINCT l.name , t.team_long_name FROM Match m  JOIN League l ' \
            f'ON m.league_id = l.id  JOIN Team t  ON m.home_team_api_id = t.team_api_id WHERE' \
            f' l.name = "{league_name}"' \
            f'ORDER BY t.team_long_name ASC;'
    return pd.read_sql(query, con)
```

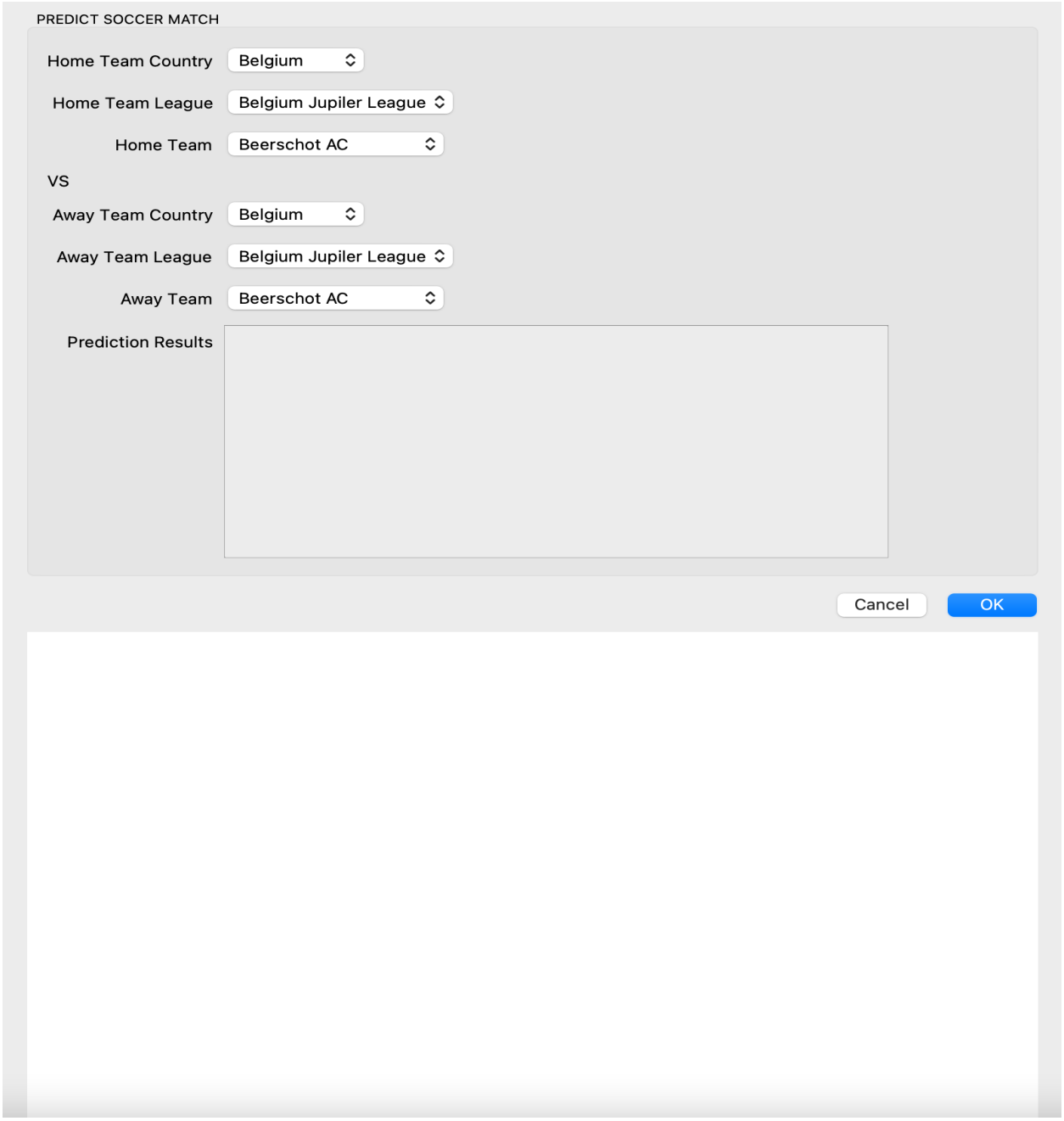
a. get_all_countries() : This fetches the list of countries from the sqlite dataset in a dataframe.No parameter is passed to this method.

b. get_country_leagues() : This fetches all the leagues in a country. The country name parameter is passed.

c. get_league_teams() : This fetches all the teams in a league by joining with the match table and removing duplicates.

3.0 RESULTS

3. 1 Play Match Window



The figure 3.1 above, shows the main GUI of the project. This is the first page that pops up when the program is executed. The view is a window where all controls are defined. The window has a title to make it more descriptive.

Since we are playing a soccer match i decided to have both team selections for the home and away team in different combo boxes. When a country is selected, triggers a method to populate the leagues for that country. When a league is selected it also triggers a method to populate the teams under the selected league. The data used to populate the country, league and teams are called from the methods in the database.py file as described in the code subroutines above.
The controls were added using the addItems() function and calling the variable to be displayed.

Lastly i added two action buttons, which are the cancel button and the Ok button. The cancel button is used to quit the whole form and the ok button is used to proceed with the model training and generating the results.Based on the team selection, the matches for both teams are generated while limiting the records to the number of matches specified.

3.2 Match Played Results

From the figure 3.2, when the home and away team is selected , the generated predictors are passed to the decision tree classifier for prediction. When the result is returned, I first added the match outcome label to the plain text edit using the set text function and I further appended the results from the classification report to the plain text edit using the append function so that it will not overwrite the match outcome displayed.

I also went ahead to display the team formation returned as an axis on the canvas we created earlier.It has to be an axis rather than a plot.

3.3 Data Preprocessing Results

3.3.1 Country Data

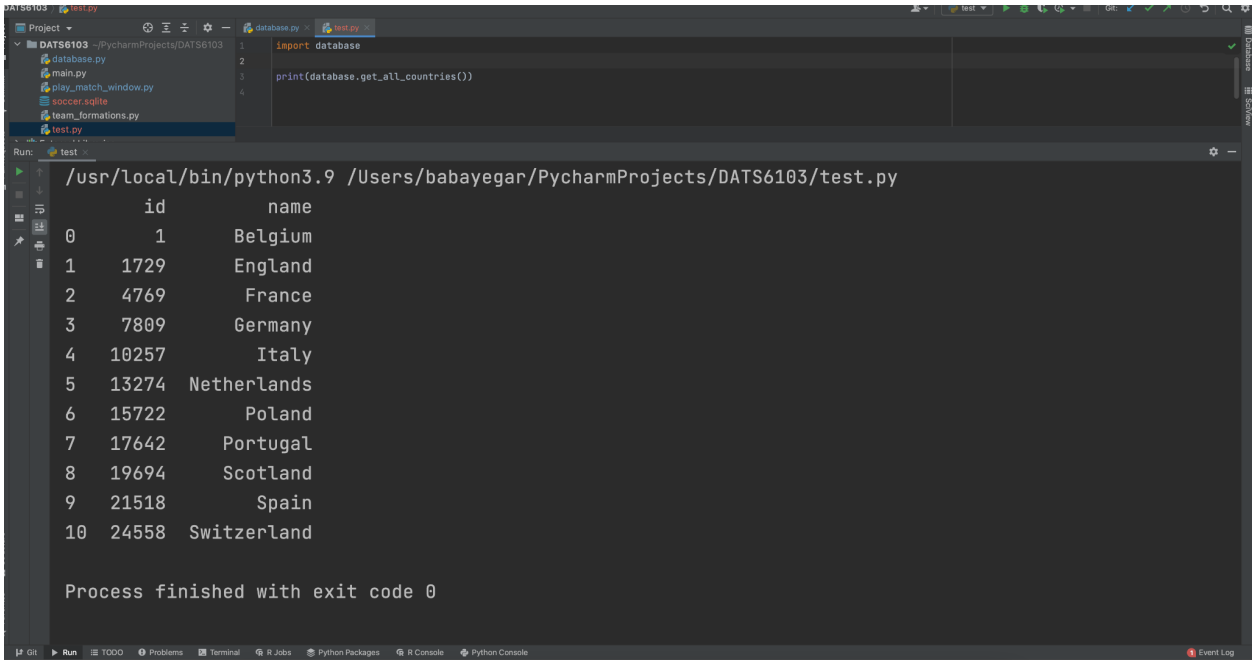


Figure 3.3.1 shows the list of countries returned from the get_all_counries() method is the database file. It returns both the country's id and name.When the results are obtained, it is returned in the form of a panda dataframe.This requires no parameter to be passed.

3.3.2 League Data

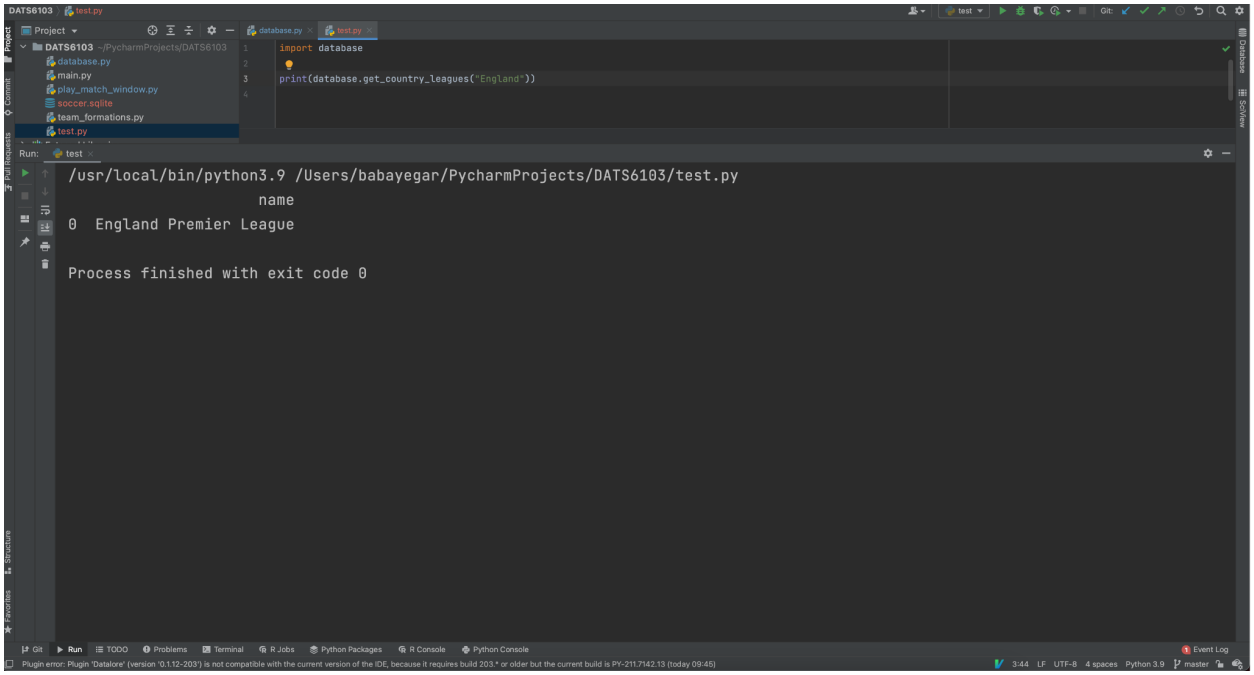


Figure 3.3.2 shows the list of leagues in a country returned from the get_country_leagues() method is the database file. It returns both the league name.When the results are obtained, it is returned in the form of a panda dataframe.This requires the country name parameter to be passed.

3.3.3 Team Data

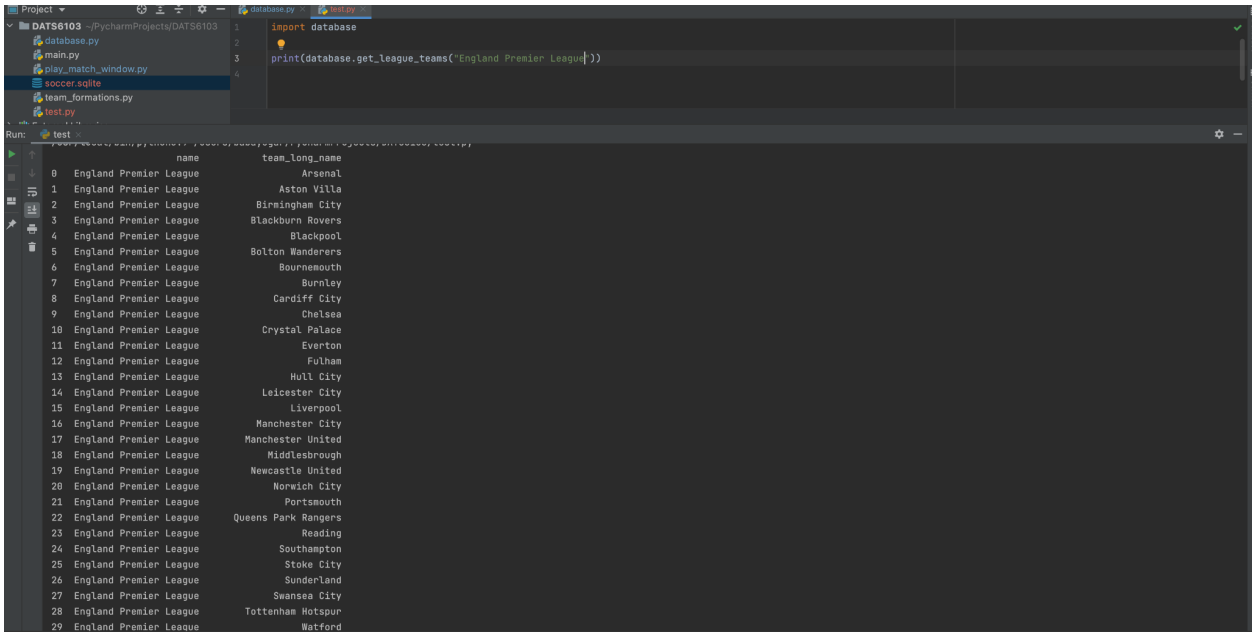


Figure 3.3.3 shows the list of leagues in a country returned from the get_league_teams() method is the database file. It returns both the league name and the team name.As said earlier, because there was no direct relationship on the team and league table, a join with the match table had to be used and duplicates dropped. When the results are obtained, it is returned in the form of a panda dataframe.This requires the league name parameter to be passed.

4.0 SUMMARY & CONCLUSIONS

This project was set out to analyze previous matches and predict a new match. Moreover it was designed to be interactive, where a user's input determines the outcome of the match which makes the result change on each game played.

My work on this project was building up the entire GUI and trying to preprocess data from the sqlite dataset file.From my results, that has been very successful. From the GUI section, using pyqt5 has been a very good learning experience. One thing i noticed in pyqt5 is that when designing layouts is that the code must be written properly or else controls might not even show up. You have to size controls properly and call them at the right time. On the data preprocessing section, it was to see that sql results can be converted to a panda dataframe. This made populating the GUI controls possible.

5.0 CODE STATISTICS

The codes i wrote are both from the database.py and the play_match_window.py.

I wrote a total of 113 lines of code.

Line of code copied from the Internet : 31
Lines of code modified : 13
Lines of added : 82

% of code from the internet is : 15.9%

6.0 REFERENCE

1. https://www.geeksforgeeks.org/pyqt5-set-fix-window-size-for-height-or-width/
2. https://pythonprogramminglanguage.com/pyqt5-center-window/#:~:text=To%20center%20a%20Python%20PyQt,the%20center%20of%20the%20screen.
3. https://www.tutorialspoint.com/pyqt5/pyqt5_tutorial.pdf
4. https://datacarpentry.org/python-ecology-lesson/09-working-with-sql/index.html
5. https://github.com/BABAYEGAR/Data-Mining/blob/master/Demo/PyQt5/Demo/Main.py