

PROJECT REPORT ON
“EMPLOYEE AND IT INFRASTRUCTURE, DATA
ANALYSIS USING PYTHON BASED DESKTOP
APPLICATION”

(FOR DMSRDE)



DEFENCE MATERIALS & STORES RESEARCH&DEVELOPMENT
ESTABLISHMENT, KANPUR

PROJECT GUIDE:

Mr. SANJEEV KUMAR (Sc. 'E')

SUBMITTED BY:

IRTEZA ALI

B. Tech IIIrd Year

(REG NO.1605210021)

DEPARTMENT OF COMPUTER SCIENCE AND TECHNOLOGY

INSTITUTE OF ENGINEERING AND
TECHNOLOGY, LUCKNOW- 226021

TABLE OF CONTENTS:

<u>Index</u>	2
<u>Preface</u>	3
<u>Acknowledgement</u>	4
<u>Certificate</u>	5
<u>Organizational profile</u>	6
<u>Introduction</u>	7
<u>Software tools and platform</u>	8
<u>Python</u>	9
Advantages.....	10
<u>System perquisites</u>	11
PyCharm.....	13
SQLite3.....	14
Pandas.....	15
PyQt.....	16

17	Matplotlib.....
19	<u>Functional windows</u>
20	Welcome window.....
21	<i>Functionalities</i>
24	Employee database window.....
25	<i>Functionalities</i>
27	Authentication login window.....
28	<i>Functionalities</i>
30	Sign up window.....
31	<i>Functionalities</i>
34	Device window.....
35	<i>Functionalities</i>
36	Desktop database window.....
37	<i>Graphing</i>
38	<i>Example plots generated</i>
40	<i>Functionalities</i>

42	Printers database window.....
	<i>Graphing</i>
43	
	<i>Example plots generated</i>
44	
	<i>Functionalities</i>
46	
	<u>Methods used</u>
48	
	Data visualization and techniques used.....
49	
	<u>Conclusion</u>
62	

Preface

The innovation of computers has drastically effected organization of all kinds, be it large or small. The volume of data processed in all type of industries, medical services, financial institutes and Agencies expanding geometrically seems unimaginable to function without computers. I have been assigned the project topic “EMPLOYEE AND IT INFRASTRUCTURE DATA ANALYSIS USING PYTHON BASED DESKTOP APPLICATION” in this Organization.

This is a collection of efficient implementations of various functions on database. Descriptions are brief and intuitive. This requires interpreted high-level scripting language, such as Python, and skills in programming concepts including pandas, matplotlib, python data analysis libraries, along with GUI development libraries.

The first section introduces basic python language and its advantages. . The next section presents the GUI functionality and development and use of SQLite database for storing user information. The next section presents some important python libraries and packages. This is followed by an implementation that allows efficient reading, and plotting graph etc. operations. The last section describes exporting the result to database and saving the results.

.

ACKNOWLEDGEMENT

I feel greatly indebted to all those who helped me during the project work. I avail this opportunity for thanking them. I sincerely express my gratitude to Director **Dr N Eswara Prasad**, D.M.S.R.D.E Kanpur & **Mr. S.B. Yadaw** Directorate Head DTS-I DMSRDE, for giving me an opportunity to enhance my skills and allowing me to join this esteemed organization for 6 weeks project training.

This acknowledgement would be incomplete if I fail to express deep gratitude to Mr. **Sanjeev Kumar** for his valuable guidance, patience and consummate support. Finally, I would like to thank Tec coordination section and whole staff of Administration centre D.M.S.R.D.E., Kanpur for the valuable support they rendered and for providing such a stimulating environment in which the project development was done with style and verve.

ORGANIZATION PROFILE

The origins of the Defence Materials & Stores Research & Development Establishment (DMSRDE) can be traced back to 1929, when it was known as the Inspectorate of General Stores in the Harness & Saddlery Factor in Cawnpore (present day Kanpur). After 1929, the establishment underwent a number of reorganizations. Finally, in 1976, the Defence Materials & Stores Research & Development Establishment was established through the amalgamation of three separate establishments.

The DMSRDE is involved in interdisciplinary research and development of materials for the military services, including various types of chemical protective clothing and equipment. Work at the DMSRDE laboratory in Kanpur includes the synthesis and development of polymers and composite materials, as well as research and development of protective clothing and equipment against hazardous and toxic chemicals. In partnership with the Defence Research and Development Establishment (DRDE), the DMSRDE has produced five types of protective systems and equipment that have been introduced into the services. These include nuclear, biological, and chemical (NBC) individual protection equipment, NBC collective protection system, NBC medical protection equipment, NBC detection equipment, and a NBC decontamination system.

The DMSRDE Kanpur laboratory interacts with the three military services, as well as a number of academic institutions, including the Indian Institute of Technology, Kanpur, and the Indian Institute of Technology, Mumbai.

INTRODUCTION

Organization requires the services of a large number of personnel. D.M.S.R.D.E involves in R&D, is a DRDO lab working in non-metallic materials, textiles and critical scientific and technological area. It has strong manpower consisting of around 200 officers and 600 staffs. These personnel occupy the various position created through the process of organizing. Each position of the organization has certain specific contribution to achieve organization objectives.

Now to provide various facilities to these employees is the responsibility of Government of India and management of the former is the duty of the organization. Organization has been authorized to hold its Employee Record.

D.M.S.R.D.E. also manages the records of offered courses, Higher Degree, Junior Research Fellowship, Senior Research Fellowship etc.

Database management, the repository of information is the foundation of the entire computerized information management system for an enterprise and has evolved from a specialized computer application to a central component of a modern computing environment.

Database systems are designed to manage large bodies of information. The management of data involves both the definition of structures for the storage of information and provision of mechanism for the information.

Python programming language is used to access the specific database which is directly related to D.M.S.R.D.E or to the employees and their facilities.

SOFTWARE TOOLS AND PLATFORM: -

COMPILER/INTERPRETER	WEBSITE	PLATFORM
CPython	www.python.org/download/	Windows, Mac, Linux
PyPy – Python implementation	pypy.org/download.html	Windows, Mac
Jupyter Notebook	Jupyter.org	Windows, Mac, Linux
PyCharm	Pycharm.io	Linux, Windows, OS X

IDLE	WEBSITE	PLATFORM
IDLE – A GUI based IDE that comes pre-installed with the reference implementation (CPython)	www.python.org/download/	Windows, Mac, Linux

Some Attractive features of project

- Quick Exploratory Data Analysis (EDA)
- Login – Signup based application
- Plotting attractive graphs on data
- Feeding data into machine learning tools like scikit-learn
- Interactive GUI models for analyzing your data
- Supports large amount of Data.

PYTHON

Python is a widely used high-level, general-purpose, interpreted, dynamic programming language. Its design philosophy emphasizes code readability, and its syntax allows programmers to express concepts in fewer lines of code than would be possible in languages such as C++ or Java. The language provides constructs intended to enable clear programs on both a small and large scale.

Python supports multiple programming paradigms, including object-oriented, imperative and functional programming or procedural styles. It features a dynamic type system and automatic memory management and has a large and comprehensive standard library. Python interpreters are available for Installation on many operating systems, allowing Python code execution on a wide variety of systems.

History

Python was conceived in the late 1980s, and its implementation was started in December 1989 by Guido van Rossum at CWI in the Netherlands as a successor to the ABC language (itself inspired by SETL) capable of exception handling and interfacing with the Amoeba operating system. Van Rossum is Python's principal author, and his continuing central role in deciding the direction of Python is reflected in the title given to him by the Python community, benevolent dictator for life (BDFL).

Versions

Python has two versions 2.x version and 3.x version. The 3.x version is a backward incompatible release was released to fix many design issues which plagued the 2.x series. The latest in the 2.x series is 2.7.15 and the latest in 3.x series is 3.6.5.

Advantages of Python

I believe that it is both possible and desirable to use Python:

- It is Free (as in both cost and source code).
- It is trivial to install on a Windows PC allowing students to take their interest further. For many the hurdle of installing a Pascal or C compiler on a Windows machine is either too expensive or too complicated.
- It is a flexible tool that allows both the teaching of traditional procedural programming and modern OOP. It can be used to teach a large number of transferable skills.
- It is a real-world programming language that can be and is used in academia and the commercial world.
- It appears to be quicker to learn and, in combination with its many libraries, this offers the possibility of more rapid student development allowing the course to be made more challenging and varied.
- Most importantly, its clean syntax offers increased understanding and enjoyment for students.

System prerequisites:

We will use [Python 3](#) and [PyCharm](#) to demonstrate the code in this project. In addition to Python and PyCharm, we will need the following Python modules:

- [matplotlib](#) - data visualization
- [PyQT](#) – GUI development
- [NumPy](#) - numerical data functionality
- [SQLite3](#) – Backend storage of data
- [pandas](#) - data import, clean-up, exploration, and analysis
- [xlrd](#) - read Excel data
- [xlwt](#) - write to Excel
- [XlsxWriter](#) - write to Excel (xlsx) files

There are multiple ways to get set up with all the modules. We cover three of the most common scenarios below.

I. If you have Python installed via Anaconda package manager, you can install the required modules using the command `conda install`. For example, to install pandas, you would execute the command `-conda install pandas`.

II. If you already have a regular, non-Anaconda Python installed on the computer, you can install the required modules using `pip`.

Open your command line program and execute command `pip install <module name>` to install a module. You should replace `<module name>` with the actual name of the module you are trying to install. For example, to install pandas, you would execute command `- pip install pandas`.

III. If you don't have Python already installed, you should get it through the Anaconda package manager. Anaconda provides installers for Windows, Mac, and Linux Computers.

If you choose the full installer, you will get all the modules you need, along with Python and pandas within a single package. This is the easiest and fastest way to get started.

PyCharm:

PyCharm is an integrated development environment (IDE) used in computer programming, specifically for the Python language. It is developed by the Czech company JetBrains. It provides code analysis, a graphical debugger, an integrated unit tester, integration with version control systems (VCSes), and supports web development with Django.

PyCharm is cross-platform, with Windows, macOS and Linux versions. The Community Edition is released under the Apache License, and there is also Professional Edition released under a proprietary license - this has extra features.

- Coding assistance and analysis, with `code_completion`, syntax and error highlighting, linter integration, and quick fixes
- Project and code navigation: specialized project views, file structure views and quick jumping between files, classes, methods and usages
- Python [refactoring](#): including rename, extract method, introduce variable, introduce constant, pull up, push down and others
- Support for web frameworks: [Django](#), [web2py](#) and [Flask](#)
- Integrated Python [debugger](#)
- Integrated [unit testing](#), with line-by-line [code coverage](#)
- [Google App Engine](#) Python development

- Version control integration: unified user interface for [Mercurial](#), [Git](#), [Subversion](#), [Perforce](#) and [CVS](#) with changelists and merge

SQLite3

SQLite is an in-process library that implements a self-contained, serverless, zero-configuration, transactional SQL database engine. The code for SQLite is in the public domain and is thus free for use for any purpose, commercial or private. SQLite is the most widely deployed database in the world with more applications than we can count, including several high-profile projects.

SQLite is an embedded SQL database engine. Unlike most other SQL databases, SQLite does not have a separate server process. SQLite reads and writes directly to ordinary disk files. A complete SQL database with multiple tables, indices, triggers, and views, is contained in a single disk file. The database file format is cross-platform - you can freely copy a database between 32-bit and 64-bit systems

SQLite is a compact library. With all features enabled, the library size can be less than 500KiB, depending on the target platform and compiler optimization settings. (64-bit code is larger.

The features involve the following:

- Zero-Configuration

- Server less
- Stable Cross-Platform Database File
- Compact
- Variable-length records

Pandas

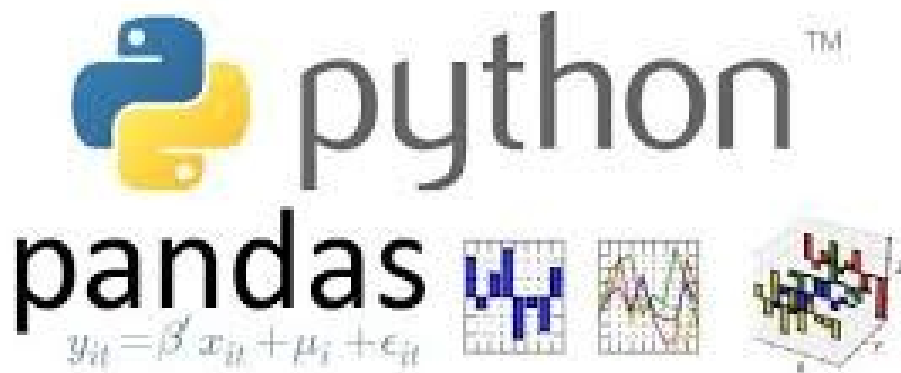
Pandas have excellent methods for reading all kinds of data from Excel files. You can also export your results from pandas back to Excel, if that's preferred by your intended audience. Pandas are great for other routine data analysis tasks

Pandas is better at automating data processing tasks than Excel, including processing Excel files.

In this tutorial, we are going to show you how to work with Excel files in pandas.

We will cover the following concepts.

- setting up our computer with the necessary software
- reading in data from Excel files into pandas
- data exploration in pandas
- visualizing data in pandas using the matplotlib visualization library
- manipulating and reshaping data in pandas
- moving data from pandas into Excel



Python for Data Analysis Training

PyQt

PyQt is a GUI widgets toolkit. It is a Python interface for **Qt**, one of the most powerful, and popular cross-platform GUI library.

PyQt is a blend of Python programming language and the Qt library. PyQt API is a set of modules containing a large number of classes and functions. While **QtCore** module contains non-GUI functionality for working with file and directory etc., **QtGui** module contains all the graphical controls. In addition, there are modules for working with XML (**QtXml**), SVG (**QtSvg**), and SQL (**QtSql**), etc.

Creating a simple GUI application using PyQt involves the following steps –

- Import QtGui module.
- Create an application object.
- A QWidget object creates top level window.
- Define the size and position of window by setGeometry() method.
- Connection to other windows using pushButtons()
- Enter the mainloop of application by **app.exec_()** method.

```
scratch_1.py x
1 from PyQt5 import QtCore, QtGui, QtWidgets
2 import pandas as pd
3 import matplotlib.pyplot as plt
4 import sqlite3
```

Matplotlib:

Matplotlib is a Python 2D plotting library which produces publication quality figures in a variety of hardcopy formats and interactive environments across platforms.

Some features provided by this python module are:

- python 2D plotting library which produces publication quality figures in a variety of hardcopy formats
- a set of functionalities similar to those of MATLAB.
- line plots, scatter plots, bar charts, histograms, pie charts etc.
- relatively low-level, some effort needed to create advanced visualization.

First, we import the matplotlib module and set matplotlib to display the plots right in the PyCharm window.

```
import matplotlib.pyplot as plt
%matplotlib inline
```

We will draw a bar plot where each bar will represent one of the top 10 data. We can do this by calling the plot method and setting the argument kind to barh. This tells matplotlib to draw a horizontal bar plot.

```
sorted_by_gross['Gross Earnings'].head(10).plot(kind="barh")  
plt. show()
```

Functional Windows:

1. **Welcome window:** This window is the introductory GUI window that appears on the screen first time when the python GUI is launched .
2. **Login Window:** This window is used to provide access to only people who are authorized.
3. **Sign Up Window:** This window is for creation of user accounts based on the information provided.
4. **Employee window:** This window is used to access database of employees and view them accordingly based on grouping.
5. **Device window:** This window is used to chose which database has to accessed whether printers or desktops.
6. **Desktop window:** This window is responsible for accessing and plotting databases of desktops available in organization.
7. **Printer window:** This window is responsible for accessing and plotting databases of printers available in organization.

Welcome Window:



This window is the introductory GUI window that appears on the screen first time when the python code is run . The window consists of 3 labels and two push buttons that are connected to other windows . It also contains a label with pixmap feature that holds the image in position. The login button takes us to login window whereas employee database can be accessed using employee push button. The code snippet can be found on next page.

Functionalities for welcome window:

1. Login Button:

```
self.pushButton.setFont(font)
self.pushButton.setCursor(QtGui.QCursor(QtCore.Qt.PointingHandCursor))
self.pushButton.setStyleSheet("QPushButton {\n"
    "    border: 2px solid #8f8f91;\n"
    "    border-radius: 6px;\n"
    "    background-color: glineargradient(spread:pad, x1:0.429599, y1:0.648, x2:0.440678, y2:0, stop: 0 #dadbdde, stop: 1 #f6f7fa);\n"
    "    min-width: 80px;\n"
    "}\n"
    "\n"
    "QPushButton:pressed {\n"
    "    background-color: glineargradient(x1: 0, y1: 0, x2: 0, y2: 1,\n"
    "                                     stop: 0 #dadbdde, stop: 1 #f6f7fa);\n"
    "}\n"
    "\n"
    "QPushButton:flat {\n"
    "    border: none; /* no border for a flat push button */\n"
    "}\n"
    "\n"
    "QPushButton:default {\n"
    "    border-color: navy; /* make the default button prominent */\n"
    "}\n"
    "\n"
    "QPushButton:hover { color: white }")
self.pushButton.setObjectName("pushButton")
```

2. Employees Button:

```
self.pushButton_2.setFont(font)
self.pushButton_2.setCursor(QtGui.QCursor(QtCore.Qt.PointingHandCursor))
self.pushButton_2.setStyleSheet("QPushButton {\n"
    "    border: 2px solid #8f8f91;\n"
    "    border-radius: 6px;\n"
    "    background-color: glineargradient(spread:pad, x1:0.429599, y1:0.648, x2:0.440678, y2:0, stop: 0 #dadbdde, stop: 1 #f6f7fa);\n"
    "    min-width: 80px;\n"
    "}\n"
    "\n"
    "QPushButton:pressed {\n"
    "    background-color: glineargradient(x1: 0, y1: 0, x2: 0, y2: 1,\n"
    "                                     stop: 0 #dadbdde, stop: 1 #f6f7fa);\n"
    "}\n"
    "\n"
    "QPushButton:flat {\n"
    "    border: none; /* no border for a flat push button */\n"
    "}\n"
    "\n"
    "QPushButton:default {\n"
    "    border-color: navy; /* make the default button prominent */\n"
    "}\n"
    "\n"
    "QPushButton:hover { color: white }")
self.pushButton_2.setObjectName("pushButton_2")
```

3. Connection to new Employees Window:

```
self.pushButton_2.setObjectName("pushButton_2")  
self.pushButton_2.clicked.connect(self.emp)  
self.pushButton_2.setObjectName("pushButton_2")
```

```
def emp(self):  
    self.object = QtWidgets.QWidget()  
    self.ui = Ui_Form()  
    self.ui.setupUi(self.object)  
    self.object.show()
```

4.Connection to new Login Window:

```
1518 def logged(self):  
1519     self.login = QtWidgets.QDialog()  
1520     self.ui = Ui_main0()  
1521     self.ui.setupUi(self.login)  
1522     self.login.show()  
1523  
1524 def log(self):  
1525     print("Please Enter Details to login")  
1526     self.logged()  
1527
```

```
1593 self.pushButton.setObjectName("pushButton")  
1594 self.pushButton.clicked.connect(self.log)  
1595 self.label_4 = QtWidgets.QLabel(Dialog)
```


5. Execution of welcome window:

```
2200 ▶ if __name__ == "__main__":  
2201     import sys  
2202  
2203     app = QtWidgets.QApplication(sys.argv)  
2204     Dialog = QtWidgets.QDialog()  
2205     ui = Ui_Dialog()  
2206     ui.setupUi(Dialog)  
2207     Dialog.show()  
2208     sys.exit(app.exec_())  
2209
```

6. The image:

```
self.label_2.setPixmap(QtGui.QPixmap("E:/Final/images.png"))
```

No database is accessed in the welcome window neither any data input is done its an introductory window that leads guests to employee window otherwise the authorization window can be opened through login button . The libraries used are PyQt and properties of inheritance are also used .

Employee database Window:

Scientists At DMSRDE

Category: Scientist F

5 Dr Trilok C Shami
6 Dr Ashok Ranjan
7 Dr Durgesh N Tripathi
8 Darshan Lal
9 Dr (Mrs) Tandra Nandi
10 Dr S M Abbas
11 Arati Kole
12 Dr R K Tiwari
13 J N Srivastava
14 A K Pandey
15 Sarvesh Kumar
16 Dr K Mukhopadhyay
17 Praveer Verma
18 Dr T H Goswami

Click to view Display

DRDO
Defence Research and Development Organisation

From the welcome window on the click of button employees it leads to employee database window in which by the use of pandas library csv file containing employee records is fed to the application and in an interactive manner the details can be viewed in the text box on the click of display button after choosing the category of employee record we want to see through the combo box.

Functionalities for Employee database window:

1.Display Button:

```
553 self.Display = QtWidgets.QPushButton(Form)
554 self.Display.setGeometry(QtCore.QRect(180, 270, 131, 41))
555 self.Display.setObjectName("Display")
556 self.Display.clicked.connect(self.Exel)
```

2. Combo Box:

```
606 self.comboBox.setItemText(0, _translate("Form", "Select"))
607 self.comboBox.setItemText(1, _translate("Form", "Director"))
608 self.comboBox.setItemText(2, _translate("Form", "Scientist G"))
609 self.comboBox.setItemText(3, _translate("Form", "Scientist F"))
610 self.comboBox.setItemText(4, _translate("Form", "Scientist E"))
611 self.comboBox.setItemText(5, _translate("Form", "Scientist D"))
612 self.comboBox.setItemText(6, _translate("Form", "Scientist C"))
613 self.comboBox.setItemText(7, _translate("Form", "Scientist B"))
614 self.comboBox.setItemText(8, _translate("Form", "Scientist A"))
615 self.label_2.setText(_translate("Form", "Category"))
616 self.label_3.setText(_translate("Form", "Click to view " ))
```

3.Function to read and write data from excel based on combo box conditions:

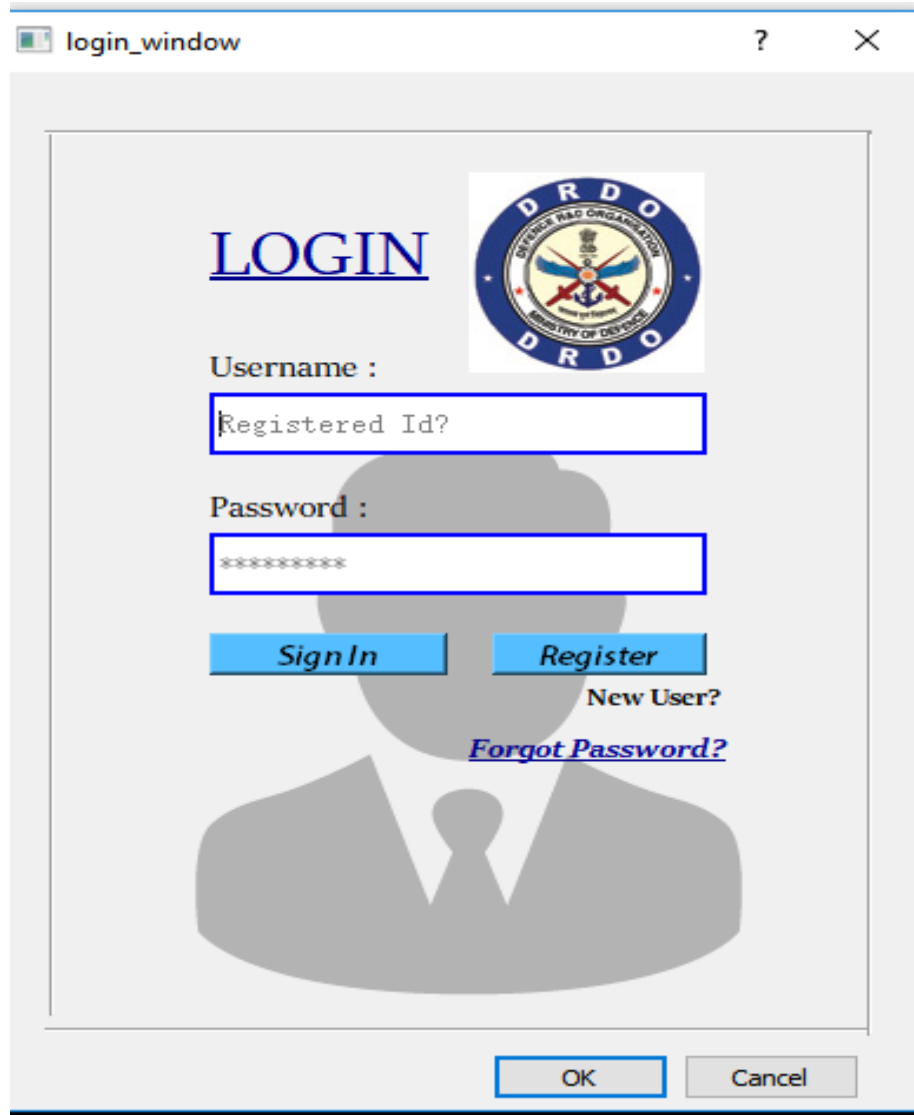
```
618 def Exel(self):
619     s = self.comboBox.currentText()
620
621     self.df = pd.read_csv("E:/names.csv")
622
623     if (s == "Director"):
624         l = []
625         l = 'Dr N Eswara Prasad.'
626         self.textEdit.setText(l)
627     elif (s == "Scientist G"):
628
629         y = "They are\n" + str(self.df.Names[0:5])
630         self.textEdit.setText(y)
631     elif (s == "Scientist F"):
632
633         y = "They are\n" + str(self.df.Names[5:21])
634         self.textEdit.setText(y)
635     elif (s == "Scientist E"):
636
637         y = "They are\n" + str(self.df.Names[22:38])
638         self.textEdit.setText(y)
639     elif (s == "Scientist D"):
```

Excel data :

O130 fx				
	A	B	C	D
1	Types		number	Names
2	Scientist 'G'	3	5	Dr Dipak Kumar Setua
3				Dr Sarfaraz Alam
4				S D Khattri
5				S B Yadav
6				Dr Anurag Srivastava
7	Scientist 'F'	14	17	Dr Trilok C Shami
8				Dr Ashok Ranjan
9				Dr Durgesh N Tripathi
10				Darshan Lal
11				Dr (Mrs) Tandra Nandi
12				Dr S M Abbas
13				Arati Kole
14				Dr R K Tiwari
15				J N Srivastava
16				A K Pandey
17				Sarvesh Kumar
18				Dr K Mukhopadhyay
19				Praveer Verma
20				Dr T H Goswami
21				Raj Kumar Jain
22				Shielendra Kumar
23				Dr Dibyendu S Bag
24	Scientist 'E'	13	16	Jitendra Yadav
25				Alok Kumar Dixit

The scientists in DMSRDE are divided from Scientist A to Scientist G based on their experience and seniority .

Authentication Login Window:



The screenshot shows a window titled "login_window" with a standard Windows-style title bar (minimize, maximize, close buttons). The window content has a light gray background. At the top left, the word "LOGIN" is displayed in a large, blue, serif font. To its right is the DRDO logo, which is a circular emblem with "DRDO" at the top and "MINISTRY OF DEFENCE" at the bottom. Below the logo, the text "Username :" is followed by a text input field containing the placeholder text "Registered Id?". Below this, the text "Password :" is followed by a text input field filled with asterisks. Under the password field are two blue buttons: "Sign In" and "Register". To the right of the "Register" button is the text "New User?". Below these buttons is a blue, underlined link that says "Forgot Password?". In the background, there is a faint, gray silhouette of a person in a suit. At the bottom right of the window, there are two buttons: "OK" and "Cancel".

From the welcome window on the click of button Login, it leads to window in which by the use of two text boxes the relevant details are taken as input. These details are cross checked on the push of sign in button if the user data exists in the already created SQLite database or not. If it does authentication successful message is launched and user is taken to device window or else Register message is launched. Registration can be done through click of Register button that takes user to SignUp window.

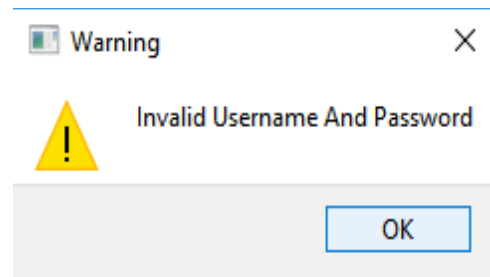
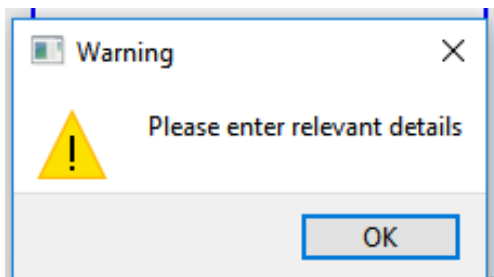
Functionalities for Login window:

1. Echo mode of password textBox :

```
self.lineEdit_5.setEchoMode(QtWidgets.QLineEdit.Password)
self.lineEdit_5.setObjectName("lineEdit_5")
```

2. Message boxes:

```
def showMessageBox(self, title, message):
    msgBox = QtWidgets.QMessageBox()
    msgBox.setIcon(QtWidgets.QMessageBox.Warning)
    msgBox.setWindowTitle(title)
    msgBox.setText(message)
    msgBox.setStandardButtons(QtWidgets.QMessageBox.Ok)
    msgBox.exec_()
```



Message boxes appear when the details are not filled or when the details are not found in sql database and authentication fails.

3. Register button:

```
def signUpShow(self):
    print("Welcome To Account Creation Panel\n")
    print("Please fill all the details")
    self.signUpWindow = QtWidgets.QDialog()
    self.ui = Ui_create()
    self.ui.setupUi(self.signUpWindow)
    self.signUpWindow.show()

def signUpCheck(self):
    print("Hi New User ")
    self.signUpShow()
```

clicked.connect(self.function) simply opens sign Up window

4. Sign in Button :

```
def loginCheck(self):
    username = self.lineEdit.text()
    password = self.lineEdit_2.text()

    connection = sqlite3.connect("login.db")
    result = connection.execute("SELECT * FROM USERS WHERE USERNAME = ? AND PASSWORD = ?", (username, password))
    if (username == "" and password == ""):
        msgBox = QtWidgets.QMessageBox()
        msgBox.setIcon(QtWidgets.QMessageBox.Warning)
        msgBox.setWindowTitle("Warning")
        msgBox.setText("Please enter relevant details")
        msgBox.setStandardButtons(QtWidgets.QMessageBox.Ok)
        msgBox.exec_()

    if (len(result.fetchall()) > 0 and username != "" and password != ""):
        print("User Found")
        self.deviceWindow = QtWidgets.QWidget()
        self.ui = Ui_device()
        self.ui.setupUi(self.deviceWindow)
        self.deviceWindow.show()
    else:
        self.showMessageBox('Warning', 'Invalid Username And Password')
        print("user not found")
```

It consists of a connection established between sqlite database and text boxes. The details input through textboxes are checked whether they exist in database or not if not message of user not found is displayed. On the contrary if data is matched and found in database then it takes us to new device window.

4. SQLite Database:

Database Structure		
Browse Data Edit Pragmas Execute SQL		
Create Table Create Index Modify Table Delete Table		
Name	Type	Schema
Tables (1)		
USERS		CREATE TABLE USERS(USERNAME TEXT NOT NULL,EMAIL TEXT,PHONE INTEGER,PASSWORD T
Indices (0)		
Views (0)		
Triggers (0)		

Sign UP Window:



Create Account

Username : irteza

Email Id : yo@gmail.com

Phone number : 9889622712

Password :

Re Enter Password :

Chose Designation

Director

Select

Intern

Trainee

Scientist

Technical officer

Security officer

Network analyst

Director

Assistant

Create

Login

Already registered?

DRDO

DEFENCE R&D ORGANISATION

MINISTRY OF DEFENCE

Like any basic sign up window the details that are asked to input can be found in any create account panel. The textboxes input the details and on click of create button the data is sent to sqlite database in a table wise manner. This data is further used in login window for authorization.

Functionalities for SignUp window:

1.SQLite Database creation:




```
def insertData(self):
    self.showMessage('Success', 'Account created please proceed to login window')

    username = self.lineEdit.text()
    email = self.lineEdit_2.text()
    phone = self.lineEdit_3.text()
    password = self.lineEdit_4.text()
    category = self.comboBox.currentText()
    if (username != "" and email != "" and password != "" and phone != "" and category != ""):
        connection = sqlite3.connect("login.db")
        connection.execute("INSERT INTO USERS VALUES(?, ?, ?, ?, ?)", (username, email, phone, password, category))
        connection.commit()
        connection.close()
```

Creation of table named users.

2. Login.db:

The use of DB Browser. exe is done in order to view files with a .db(extension)

Table:  USERS  

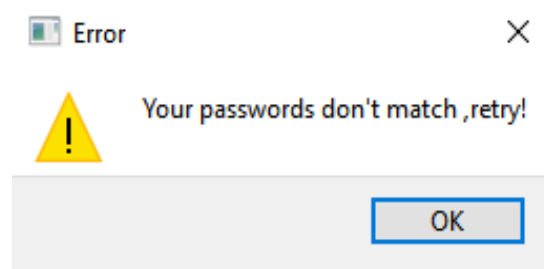
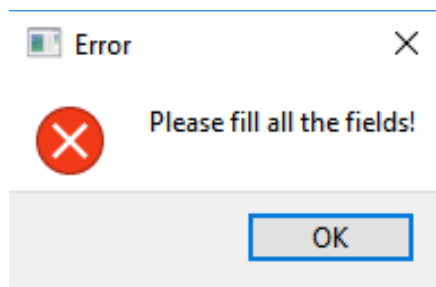
	USERNAME	EMAIL	PHONE	PASSWORD	CATEGORY
	Filter	Filter	Filter	Filter	Filter
8					Select
9				mm	Select
10	rizvi	rizvi@gmail.c...	9839424987	as	Select
11	rizvi	rizvi@gmail.c...	9839424987	as	Select
12	kk	ss	9786521212	asw	Selectkk
13	aa	aa	98771826162	qw	Security officer
14	ali99	ali99@gmail.c...	987657544	klk	Trainee
15	ali99	ali99@gmail.c...	987657544	klk	Trainee
16	san	san@gmail.com	98893772221	askl	Network analyst
17	san	san@gmail.com	98893772221	askl	Network analyst
18	sanjeevsir	sanj@gmail.c...	9889622731	askl	Select
19	sanjeevsir	sanj@gmail.c...	9889622731	askl	Network analyst
20	drsma	abbas93@red...	9838427151	dr@sma	Select
21	drsma	abbas93@red...	9838427151	dr@sma	Scientist
22	ali	ali@gmail.com	6393381264	ali	Intern
23	Anjali	anjali@gmail....	9889622645	qwerty	Security officer
24	Priyanka	Priyanka@gm...	9837564231	manimani	Scientist
25	Raiesh	raiesh@gmail	978765243	rai123	Trainee

3.Message boxes:

```

def showMessage(self, title, message):
    u = self.lineEdit.text()
    e = self.lineEdit_2.text()
    ph = self.lineEdit_3.text()
    p = self.lineEdit_4.text()
    r = self.lineEdit_5.text()
    c = self.comboBox.currentText()
    if (u == "" or e == "" or ph == "" or p == "" or u == "" or c == ""):
        msgBox = QtWidgets.QMessageBox()
        msgBox.setIcon(QtWidgets.QMessageBox.Critical)
        msgBox.setWindowTitle("Error")
        msgBox.setText("Please fill all the fields!")
        msgBox.setStandardButtons(QtWidgets.QMessageBox.Ok)
        msgBox.exec_()
    else:
        if (p != r):
            msgBox = QtWidgets.QMessageBox()
            msgBox.setIcon(QtWidgets.QMessageBox.Warning)
            msgBox.setWindowTitle("Error")
            msgBox.setText("Your passwords don't match ,retry!")
            msgBox.setStandardButtons(QtWidgets.QMessageBox.Ok)
            msgBox.exec_()
        else:
            msgBox = QtWidgets.QMessageBox()
            msgBox.setIcon(QtWidgets.QMessageBox.Information)
            msgBox.setWindowTitle(title)
            msgBox.setText(message)
            msgBox.setStandardButtons(QtWidgets.QMessageBox.Ok)
            msgBox.exec_()

```



Message box 1 appears when some field is left blank , the database is not created in this case.

Message box 2 appears when the password and retype password fields do not match.

4.Login button:

```
def loginShow(self):  
    self.loginWindow = QtWidgets.QDialog()  
    self.ui = Ui_main0()  
    self.ui.setupUi(self.loginWindow)  
    self.loginWindow.show()  
  
def loginCheck(self):  
    print("PLease Enter Details to login")  
    self.loginShow()
```

The creation class holds functions that are connected to in memory database sqlite3 . It has five columns in the table named username, password, phone number, email and designation. The table is named users and this table is further used for authentication purposes in the login window. Which can be accessed using login button provided in this window. Use of inheritance is done in order to do so.

Device Window:



Once the authentication of the user is successful . The device window which is named IT infrastructure of DRDO Kanpur as it leads to database access windows of desktops installed in the organization and also the printers. Which can be chosen easily with a click of a button.

Functionalities for SignUp window:

1. Desktops button and Printers button:

```
def open2(self):  
    self.win = QtWidgets.QDialog()  
    self.ui = Ui_desktop2()  
    self.ui.setupUi(self.win)  
    self.win.show()  
  
def open(self):  
    self.printer = QtWidgets.QDialog()  
    self.ui = Ui_desktop()  
    self.ui.setupUi(self.printer)  
    self.printer.show()
```

Connection between buttons and functions:

```
self.pushButton_3.setObjectName("pushButton_3")  
self.pushButton_3.clicked.connect(self.open2)
```

```
self.pushButton_4.setObjectName("pushButton_4")  
self.pushButton_4.clicked.connect(self.open)
```

Desktops database Window:

Printers ? X

Desktops Database Accessed

Number of devices: **View** 252 Add or append the data : **ADD**

Chose the desktop from make : DELL-SJSXFY1 RAM ☐ 512 MB ☐ 1 GB ☐ 2 GB ☒ 4 GB
☐ 256 MB ☐ DDR3 ☐ 1526 MB

Enter Serial number: 12

Click to view data : **View**

Sl. No.	12
Make	HP
Processor	Pentium-4
Ram	2 GB
DMS No.	7302
Remarks	Kamlesh Kumar
Group Name	DTT
Name: 11, dtype: object	

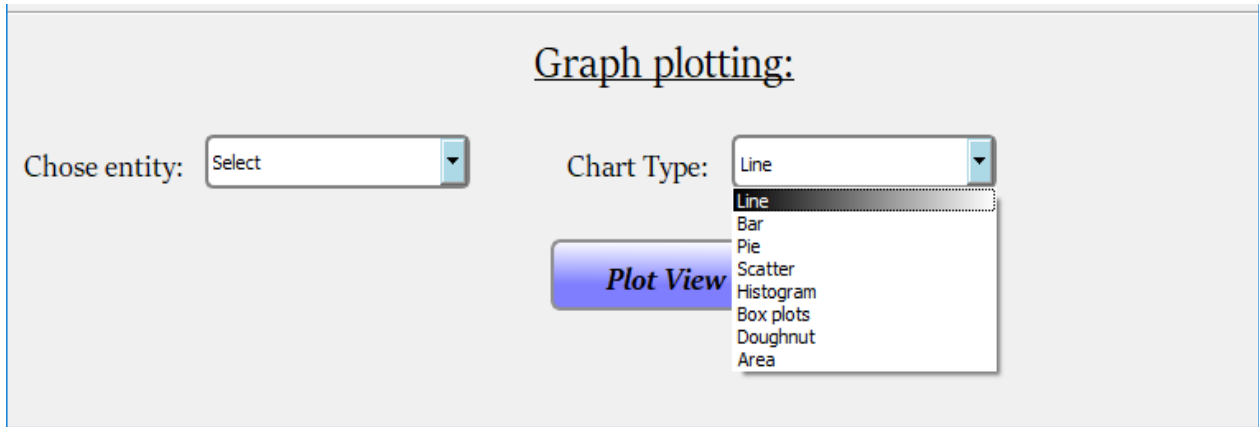
Graph plotting:

Chose entity: Select Chart Type: Line

Plot View

This window is connected to the excel database provided by the organization pertaining to installation of desktops. All the relevant information regarding group, model, processor, ram can be accessed easily. The serial number is fetched and all information pertaining to it is viwed in the text editing box on the press of button view.

Graphing Desktops Database



The image shows a web-based interface for graph plotting. At the top, the title "Graph plotting:" is centered. Below it, there are two dropdown menus. The first is labeled "Chose entity:" and has "Select" as its current value. The second is labeled "Chart Type:" and has "Line" as its current value. A blue button labeled "Plot View" is positioned between the two dropdowns. The "Chart Type:" dropdown menu is open, showing a list of options: Line, Bar, Pie, Scatter, Histogram, Box plots, Doughnut, and Area.

The bottom part is responsible for plotting of graphs on the excel data , the entity or category or column name can be chosen from first combo box and the chart type from the second combo box , on the click of button plot view a figure window appears with the plotted graphs the API used are pandas and matplotlib these are python libraries for data analysis.

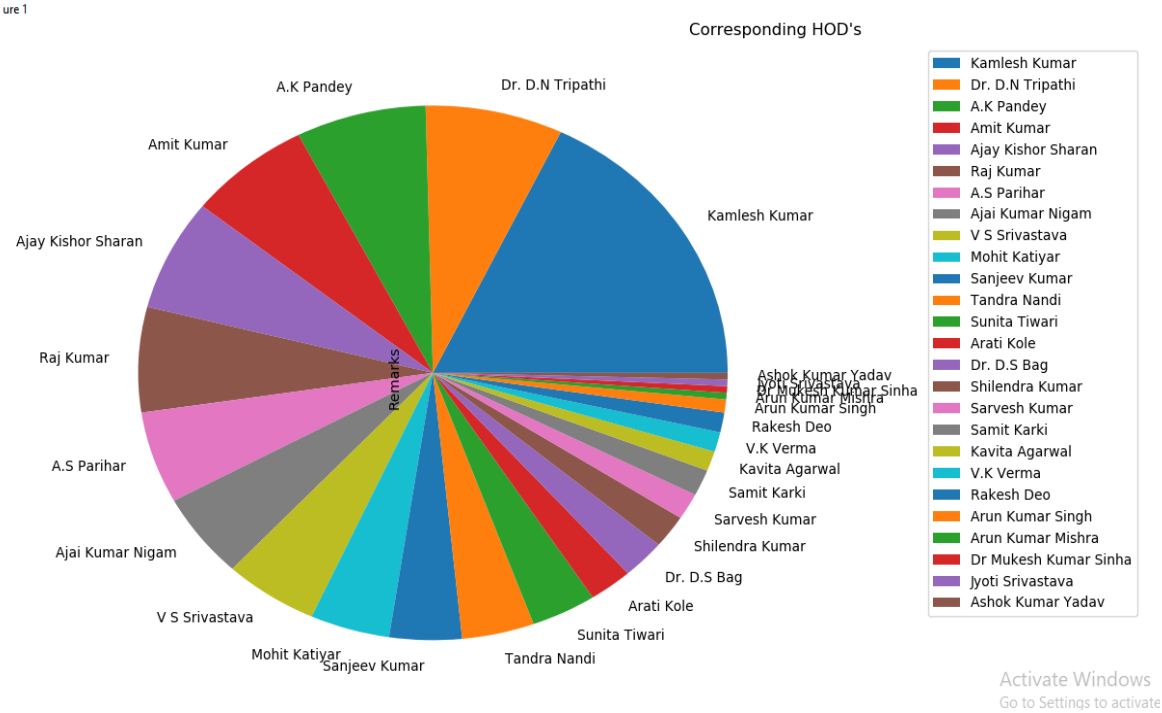
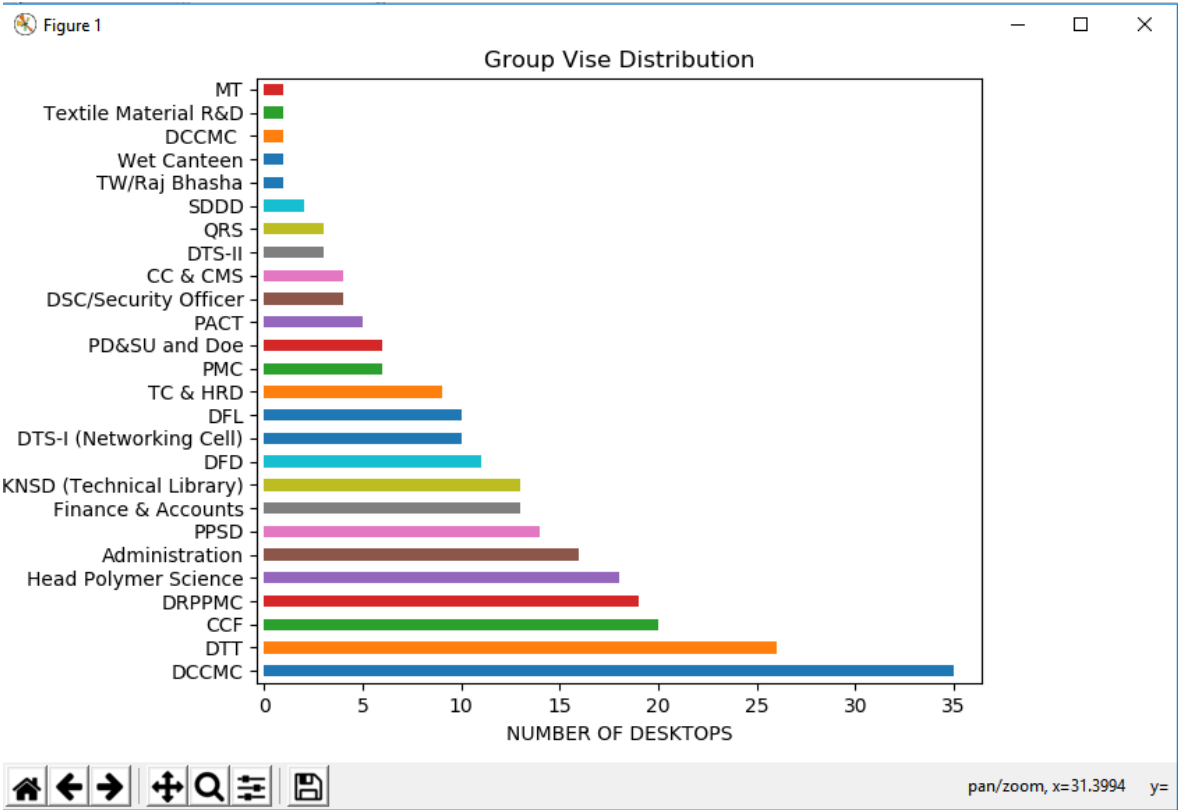
In the Entity the available categories are:

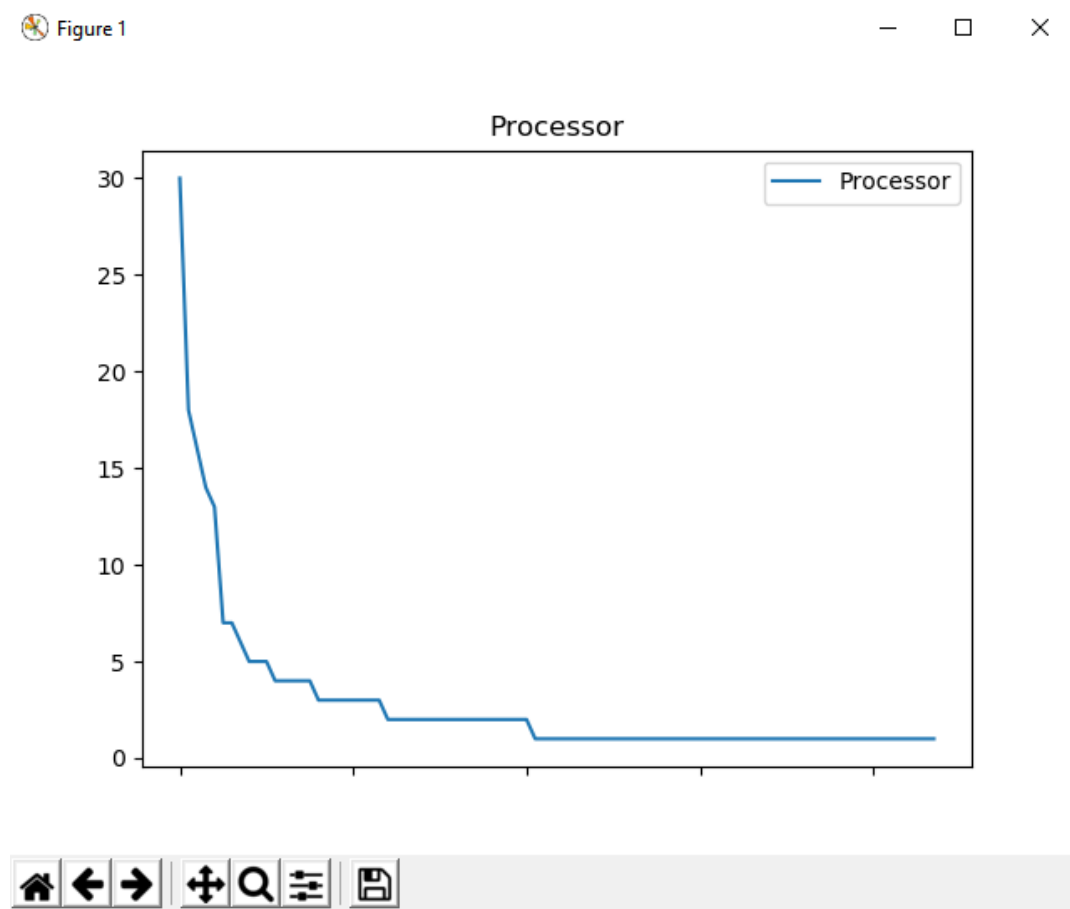
1. Make
2. Processor
3. Group
4. Remark
5. Ram

Chart types available are:

1. Line
2. Bar
3. Pie
4. Scatter
5. Box
6. Area

Example plots generated:





The bar chart was generated with the data of categorical distribution of desktop computers amongst the employees.

The pie chart represented the head of departments under whose jurisdiction the maintenance and installations of desktops is done.

The line chart shows that number of processors installed on the basis of same type.

Functionalities for Desktop database window:

1. Counting number of devices:

```
def counts(self):
    print("hi")
    url = "E:/desk.csv"
    self.df = pd.read_csv(url)

    ##it counts the total values in every column
    x = self.df.count(axis=0, level=None, numeric_only=False)
    ##y counts the length of inserted df
    y = len(self.df.index)
    self.spinBox.setValue(y)
```

2. Viewing details of devices from database:

```
def views(self):
    url = "E:/desk.csv"
    self.df = pd.read_csv(url)
    s = self.comboBox.currentText()
    m = int(self.lineEdit.text())
    a = m - 1
    l = str(self.df.loc[a])
    self.textEdit.setText(l)
```

3. Graphing:

```
def graph(self):
    url = "E:/desk.csv"
    self.df = pd.read_csv(url)
    s = self.comboBox_2.currentText()
    b = self.comboBox_3.currentText()
    if (s == 'Make' and b == 'Bar'):
        k = self.df['Make']
        pd.value_counts(k).plot(title="Desktops Installed", kind='barh')
        plt.xlabel("NUMBER OF Desktops")
        plt.ylabel("Make")
        plt.show()

    if (s == 'Processor' and b == 'Bar'):
        k = self.df['Processor']
        pd.value_counts(k).plot(title="Processors ", kind='barh')
        plt.xlabel("NUMBER OF Desktops")
        plt.ylabel("Make")
        plt.show()
```

Excel data :

	A	B	C	D	E	F	G	H	I	J
1	Sl. No.	Make	Processor	Ram	DMS No.	Remarks	Group Name			
2	1	LENOVO	Core2Duo	2 GB	6841	Kamlesh Kumar	DTT			
3	2	LENOVO	Core2Duo	2 GB	6842	Kamlesh Kumar	DTT			
4	3	HCL	Pentium-4	1 GB	6841	Kamlesh Kumar	DTT			
5	4	WIPRO	Core i-5	2 GB	N/A	Kamlesh Kumar	DTT			
6	5	ACER	Core i-3	4 GB	7403	Kamlesh Kumar	DTT			
7	6	ACER	Pentium-4	2 GB	7121	Kamlesh Kumar	DTT			
8	7	LENOVO	Pentium-4	2 GB	7146	Kamlesh Kumar	DTT			
9	8	COMPAQ	Pentium-4	2 GB	6693	Kamlesh Kumar	DTT			
10	9	HCL	N/A	512 MB	6273	Kamlesh Kumar	DTT			
11	10	HCL	Pentium-4	512 MB	6003	Kamlesh Kumar	DTT			
12	11	HP	Pentium-4	2 GB	7174	Kamlesh Kumar	DTT			
13	12	HP	Pentium-4	2 GB	7302	Kamlesh Kumar	DTT			
14	13	COMPAQ	Pentium-4	2 GB	N/A	Kamlesh Kumar	DTT			
15	14	WIPRO	Pentium-4	N/A	6245	Kamlesh Kumar	DTT			
16	15	HP	core i-3	N/A	7005	Kamlesh Kumar	DTT			
17	16	HP	intel vipro	N/A	6457	Kamlesh Kumar	DTT			
18	17	DELL	core i-7	N/A	N/A	Kamlesh Kumar	DTT			
19	18	HP	L1710	N/A	6459	Kamlesh Kumar	DTT			
20	19	DELL-90TXFY1	core-i7	N/A	N/A	Kamlesh Kumar	DTT			
21	20	DELL-5JSXFY1	core-i7	N/A	N/A	Kamlesh Kumar	DTT			
22	21	HP	core-i3	N/A	N/A	Kamlesh Kumar	DTT			
23	22	HP	intel vipro	N/A	N/A	Kamlesh Kumar	DTT			
24	23	DELL	core- i7	N/A	N/A	Kamlesh Kumar	DTT			
25	24	HP	core-i3	N/A	N/A	Kamlesh Kumar	DTT			

The data consists of 255 rows in which column wise distribution is done on the basis of make, group, remark, processors and ram.

Printers database Window:

The screenshot shows a window titled "Printers" with a subtitle "Printers Database Accessed". It contains several interactive elements:

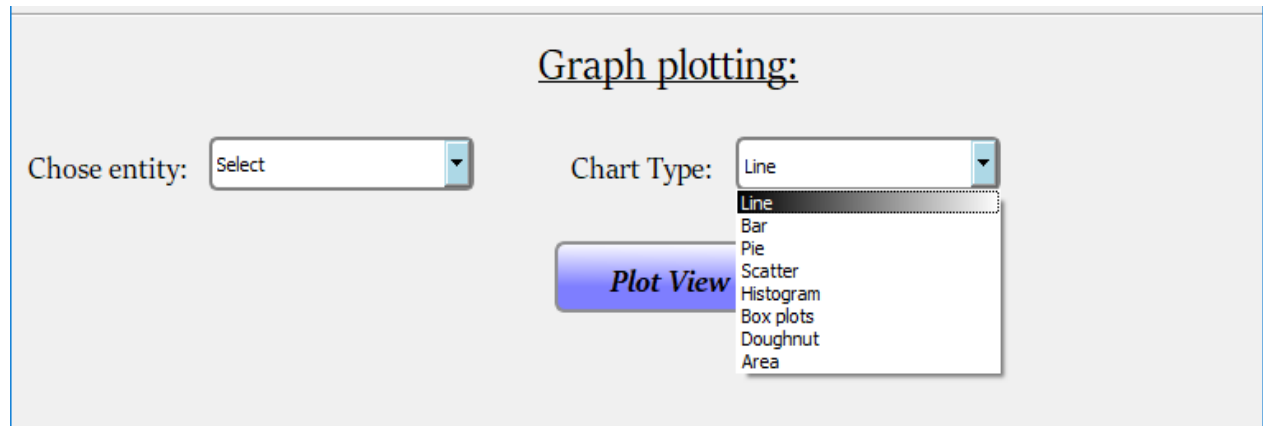
- Number of devices:** A label followed by a blue "View" button and a text box containing "162".
- Add or append the data:** A blue "ADD" button.
- Chose the printer from make :** A dropdown menu currently showing "Select".
- Type:** Two checkboxes labeled "Color" and "Black n White".
- Enter Serial number:** A text input field.
- Click to view data :** A blue "View" button.
- Graph plotting:** A section with two dropdown menus: "Chose entity:" (showing "Select") and "Chart Type:" (showing "Select").
- Plot View:** A blue button at the bottom.

The dropdown menu for "Chose the printer from make :" is open, displaying a list of printer models and brands: Select, HP-1007, HP-1505, HP-1022, HP Laser jet, Laser jet cp 1025, Samsung Scx-4300, HP Laser 1020, HP Laser jet P1505, HP Laser jet P 1007, Laser jet, DMP, HP, Canon, and Xerox.

This window is connected to the excel database provided by the organization pertaining to installation of printers. All the relevant information regarding group, model, type can be accessed easily. The serial number is fetched and all

information pertaining to it is viewed in the text editing box on the press of button view.

Graphing Printers Database



The image shows a web-based interface for graph plotting. At the top, the title 'Graph plotting:' is centered. Below it, there are two dropdown menus. The first is labeled 'Chose entity:' and has 'Select' as its current value. The second is labeled 'Chart Type:' and has 'Line' as its current value. A blue button labeled 'Plot View' is positioned between the two dropdowns. The 'Chart Type' dropdown menu is open, showing a list of options: Line, Bar, Pie, Scatter, Histogram, Box plots, Doughnut, and Area.

The bottom part is responsible for plotting of graphs on the excel data , the entity or category or column name can be chosen from first combo box and the chart type from the second combo box , on the click of button plot view a figure window appears with the plotted graphs the API used are pandas and matplotlib these are python libraries for data analysis.

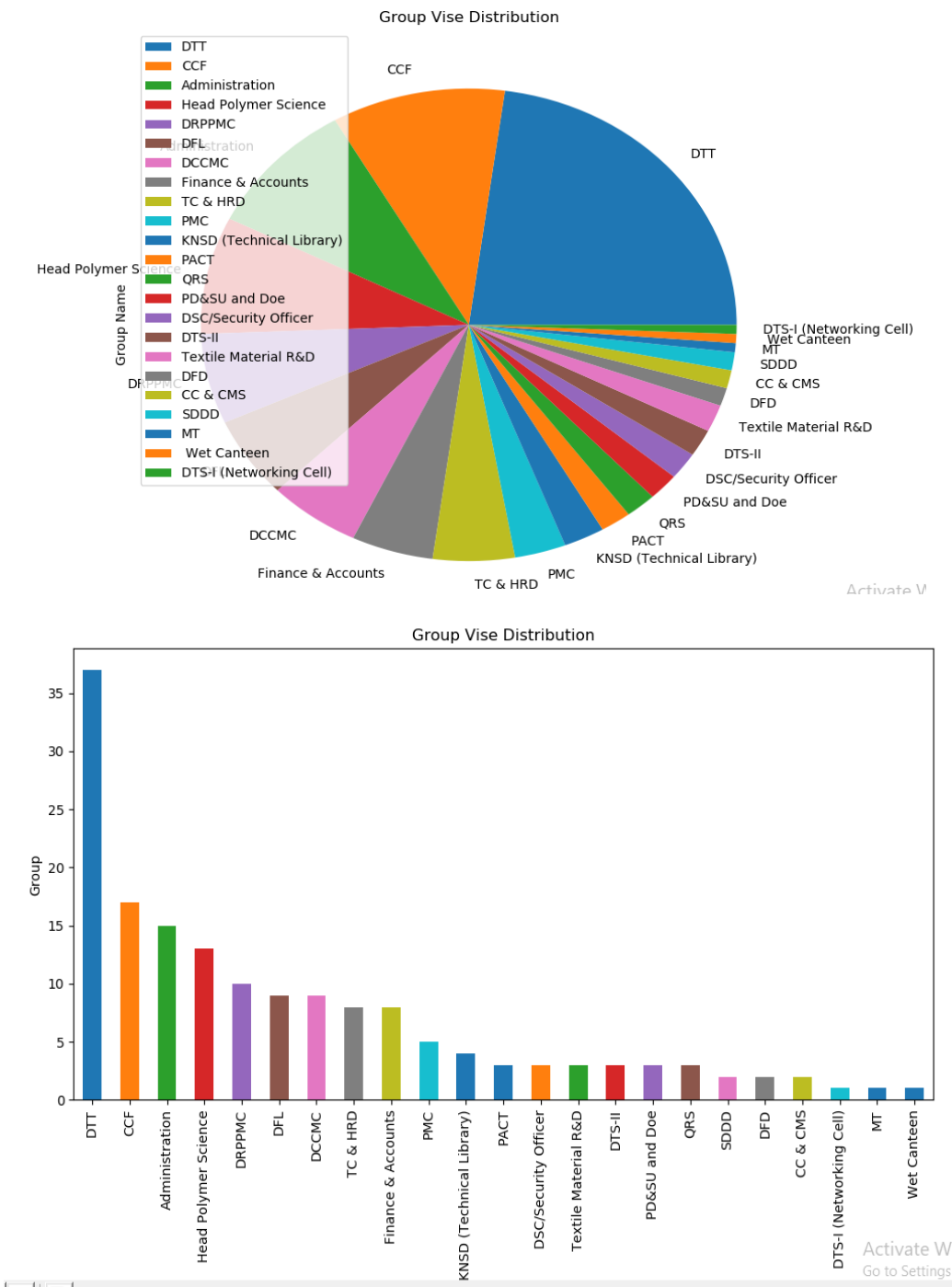
In the Entity the available categories are:

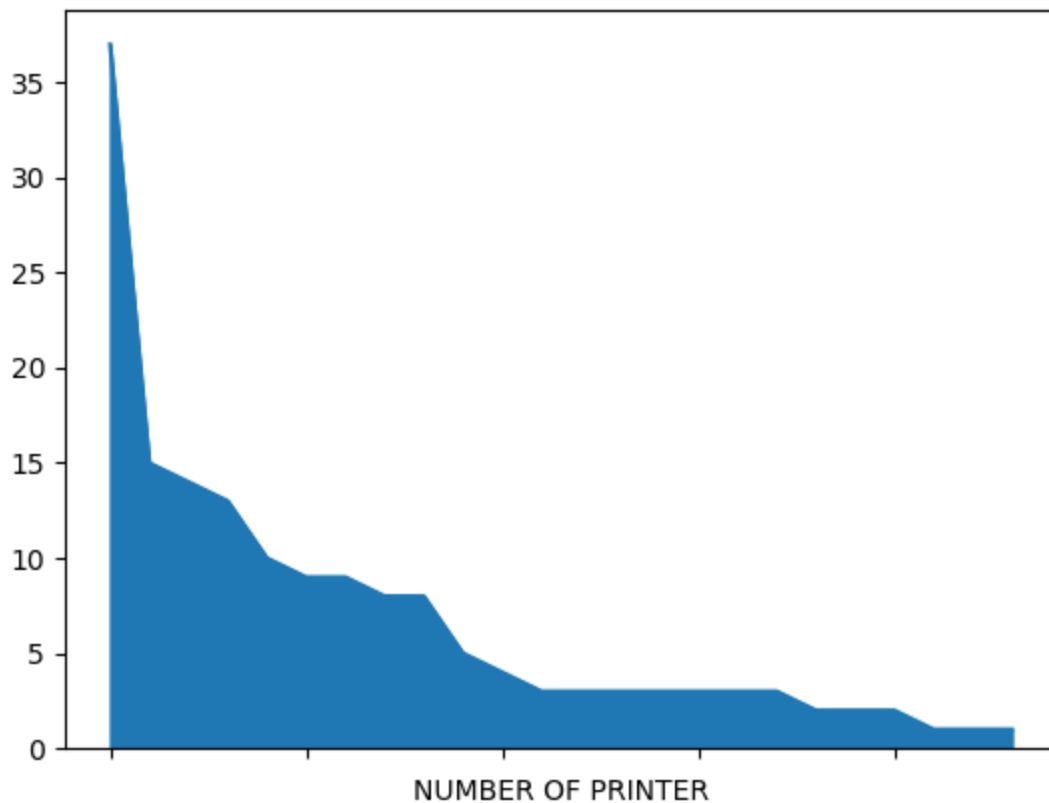
1. Make
2. Processor
3. Group
4. Remark
5. Type

Chart types available are:

1. Line
2. Bar
3. Pie
4. Scatter
5. Box
6. Area

Example plots generated:





The bar chart was generated with the data of company based distribution of printers amongst the employees.

The pie chart represented the departments under which installations of printers is done.

The area chart shows that number of processors installed on the basis of same type.

Functionalities for Printer database window:

1. Counting number of devices:

```
def counts(self):
    print("hi")
    url = "E:/desk.csv"
    self.df = pd.read_csv(url)

    ##it counts the total values in every column
    x = self.df.count(axis=0, level=None, numeric_only=False)
    ##y counts the length of inserted df
    y = len(self.df.index)
    self.spinBox.setValue(y)
```

2. Checkbox toggling on basis of colored or b/w:

```
def live(self):
    url = "E:/printer.csv"
    self.df = pd.read_csv(url)
    s = self.comboBox.currentText()
    if (s == "HP Laser Jet"):
        ##checkboxes are enabled or disabled with toggle
        self.checkBox.toggle()
    else:
        self.checkBox_2.toggle()
```

3. Graphing:

```
def graph(self):
    url = "E:/printer.csv"
    self.df = pd.read_csv(url)
    s = self.comboBox_2.currentText()
    b = self.comboBox_3.currentText()
    if (s == 'Make' and b == 'Bar'):
        k = self.df['Make']
        pd.value_counts(k).plot(title="Printers Installed", kind='barh')
        plt.xlabel("NUMBER OF PRINTERS")
        plt.ylabel("Make")
        plt.show()
    if (s == 'Type' and b == 'Bar'):
        k = self.df['Type']
        pd.value_counts(k).plot(title="B&W vs Colored Printers installed", kind='barh')
        plt.xlabel("NUMBER OF PRINTERS")
        plt.ylabel("Type")
        plt.show()
```

Excel data :

142	HP	LaserJet P1007	6796	B/W	A.K Pandey	Head Polymer Science
143	HP	Color LaserJet CP2025	6953	B/W	A.K Pandey	Head Polymer Science
144	HP	Color LaserJet 2600n	6484	B/W	A.K Pandey	Head Polymer Science
145	HP	Color LaserJet CP1025	Sl. No.-CNCH112652	B/W	A.K Pandey	Head Polymer Science
146	HP	Color LaserJet CP1515n	6894	B/W	A.K Pandey	Head Polymer Science
147	HP	Color LaserJet 2600n	6575	B/W	A.K Pandey	Head Polymer Science
148	HP	Color LaserJet CP1515n	6844	B/W	A.K Pandey	Head Polymer Science
149	Canon	LBP 2900B	Sl. No.-L11121E	B/W	A.K Pandey	Head Polymer Science
150	HP	LaserJet 1020 plus	6700	B/W	A.K Pandey	Head Polymer Science
151	HP	Color LaserJet 2600n	6384	B/W	A.K Pandey	Head Polymer Science
152	HP	LaserJet CP1025 color	7265	B/W	A.K Pandey	Head Polymer Science
153	HP	LaserJet p2015	6395	B/W	A.K Pandey	Head Polymer Science
154	HP	Color LaserJet CP1515n	6699	B/W	A.K Pandey	Head Polymer Science
155	HP	1020 CNC0170178		B/W	Ajai Kumar Nigam	Finance & Accounts
156	HP	1020 N/A		B/W	Ajai Kumar Nigam	Finance & Accounts
157	HP	1020 N/A		B/W	Ajai Kumar Nigam	Finance & Accounts
158	HP	1007 CC365A		B/W	Ajai Kumar Nigam	Finance & Accounts
159	HP	1007 CC365A		B/W	Ajai Kumar Nigam	Finance & Accounts
160	HP	1022 VNRJ7BP2V6		B/W	Ajai Kumar Nigam	Finance & Accounts
161	HP	1600DN CE749A		B/W	Ajai Kumar Nigam	Finance & Accounts
162	HP	1007 CC365A		B/W	Ajai Kumar Nigam	Finance & Accounts

The data consists of 162 rows in which column wise distribution is done on the basis of make, group, remark, and types

The Methods used:

- `read_excel ()` method
- `head ()` method
- `concat()` method
- `shape` method
- `tail()` method
- `sort_values()` method
- `describe` method

The detailed description is given in the pages up further.

The main data visualization techniques used:

- Reading data from the Excel file
- DataFrame Attributes
- Exploring the data
- Using the `ExcelFile` class to read multiple sheets
- Getting statistical information about the data
- Pivot Table in pandas
- Exporting the results to Excel

Reading data from the Excel file

We need to first import the data from the Excel file into pandas. To do that, we start by importing the pandas module.

```
import pandas as pd
```

read_excel () method

We then use the pandas' read_excel method to read in data from the Excel file. The easiest way to call this method is to pass the file name. If no sheet name is specified then it will read the first sheet in the index (as shown below).

```
excel_file = 'data.xls'  
data = pd.read_excel(excel_file)
```

Here, the read_excel method read the data from the Excel file into a pandas DataFrame object. Pandas defaults to storing data in DataFrames.

DataFrame Attributes

df.attribute	description
dtypes	list the types of the columns
columns	list the column names
axes	list the row labels and column names
ndim	number of dimensions
size	number of elements
shape	return a tuple representing the dimensionality
values	numpy representation of the data

We then stored this DataFrame into a variable called data.

head () method

Pandas has a built-in DataFrame.head() method that we can use to easily display the first few rows of our DataFrame. If no argument is passed, it will display first five rows. If a number is passed, it will display the equal number of rows from the top.

```
data.head()
```

Excel files having multiple sheets

Excel files quite often have multiple sheets and the ability to read a specific sheet or all of them is very important. To make this easy, the pandas read_excel method takes an argument called sheetname that tells pandas which sheet to read in the data from.

For this, you can either use the sheet name or the sheet number. Sheet numbers start with zero. If the sheetname argument is not given, it defaults to zero and pandas will import the first sheet.

By default, pandas will automatically assign a numeric index or row label starting with zero. You may want to leave the default index as such if your data doesn't have a column with unique values that can serve as a better index.

In case there is a column that you feel would serve as a better index, you can override the default behavior by setting index_col property to a column. It takes a numeric value for setting a single column as index or a list of numeric values for creating a multi-index.

In the below code, we are choosing the first column, 'Title', as index (index=0) by passing zero to the index_col argument.

```
data_sheet1 = pd.read_excel(excel_file, sheetname=0, index_col=0)
data_sheet1.head()
```

concat () method

The pandas concat method for this and pass in the names of the different DataFrames we just created and assign the results to a new DataFrame object, data. By keeping the DataFrame name same as before, we are overwriting the previously created DataFrame.

```
data = pd.concat([data_sheet1, data_sheet2, data_sheet3])
```

We can check if this concatenation by checking the number of rows in the combined DataFrame by calling the method shapes on it that will give us the number of rows and columns.

```
data.shape
```

```
(5042, 24)
```


Using the ExcelFile class to read multiple sheets

We can also use the ExcelFile class to work with multiple sheets from the same Excel file. We first wrap the Excel file using ExcelFile and then pass it to read_excel method.

```
xlsx = pd.ExcelFile(excel_file)
xlsx = pd.ExcelFile(excel_file)
data_sheets = []
for sheet in xlsx.sheet_names:
    data_sheets.append(xlsx.parse(sheet))
data = pd.concat(data_sheets)
```

If you are reading an Excel file with a lot of sheets and are creating a lot of DataFrames, ExcelFile is more convenient and efficient in comparison to read_excel. With ExcelFile, you only need to pass the Excel file once, and then you can use it to get the DataFrames. When using read_excel, you pass the Excel file every time and hence the file is loaded again for every sheet. This can be a huge performance drag if the Excel file has many sheets with a large number of rows.

Exploring the data

Now that we have read in the data set from our Excel file, we can start exploring it using pandas. A pandas DataFrame stores the data in a tabular format, just like the way Excel displays the data in a sheet. Pandas has a lot of built-in methods to explore the DataFrame we created from the Excel file we just read in.

We already introduced the method head in the previous section that displays few rows from the top from the DataFrame. Let's look at few more methods that come in handy while exploring the data set.

shape method

We can use this method to find out the number of rows and columns for the DataFrame.

```
data.shape
```

```
(5042, 25)
```

This tells us our Excel file has 5042 records and 25 columns or observations. This can be useful in reporting the number of records and columns and comparing that with the source data set.

tail() method

We can use the tail method to view the bottom rows. If no parameter is passed, only the bottom five rows are returned.

```
data.tail ()
```

sort_values() method

In Excel, you're able to sort a sheet based on the values in one (Since we have the data sorted by values in a column, we can do few interesting things with it. For example, we can display the top 10 data by Gross Earnings.) or more columns. In pandas, you can do the same thing with the sort_values method. For example, let's sort our data DataFrame based on the Gross Earnings column.

```
sorted_by_gross = data.sort_values(['Gross Earnings'], ascending=False)
```

Since we have the data sorted by values in a column, we can do few interesting things with it. For example, we can display the top 10 data by Gross Earnings

```
sorted_by_gross["Gross Earnings"].head(10)
```

```
1867    760505847.0
1027    658672302.0
1263    652177271.0
610     623279547.0
611     623279547.0
1774    533316061.0
1281    474544677.0
226     460935665.0
1183    458991599.0
618     448130642.0
```

```
Name: Gross Earnings, dtype: float64
```

Matplotlib:

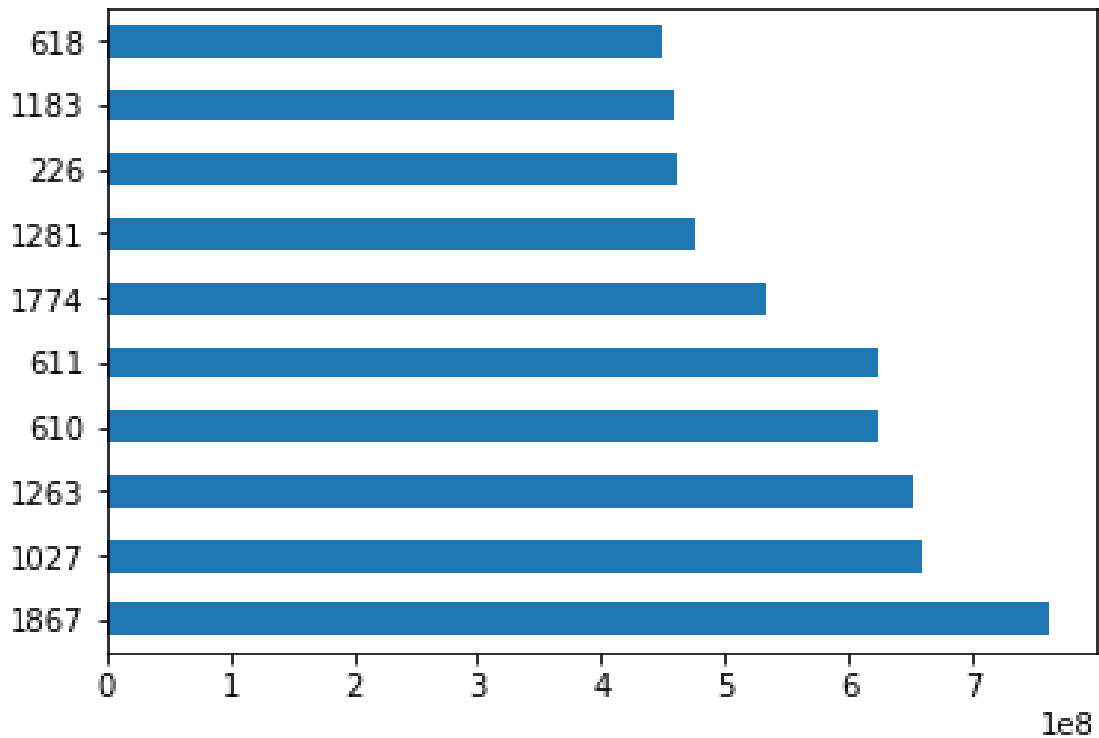
- python 2D plotting library which produces publication quality figures in a variety of hardcopy formats
- a set of functionalities similar to those of MATLAB.
- line plots, scatter plots, bar charts, histograms, pie charts etc.
- relatively low-level, some effort needed to create advanced visualization.

First, we import the matplotlib module and set matplotlib to display the plots right in the Jupyter Notebook.

```
import matplotlib.pyplot as plt
%matplotlib inline
```

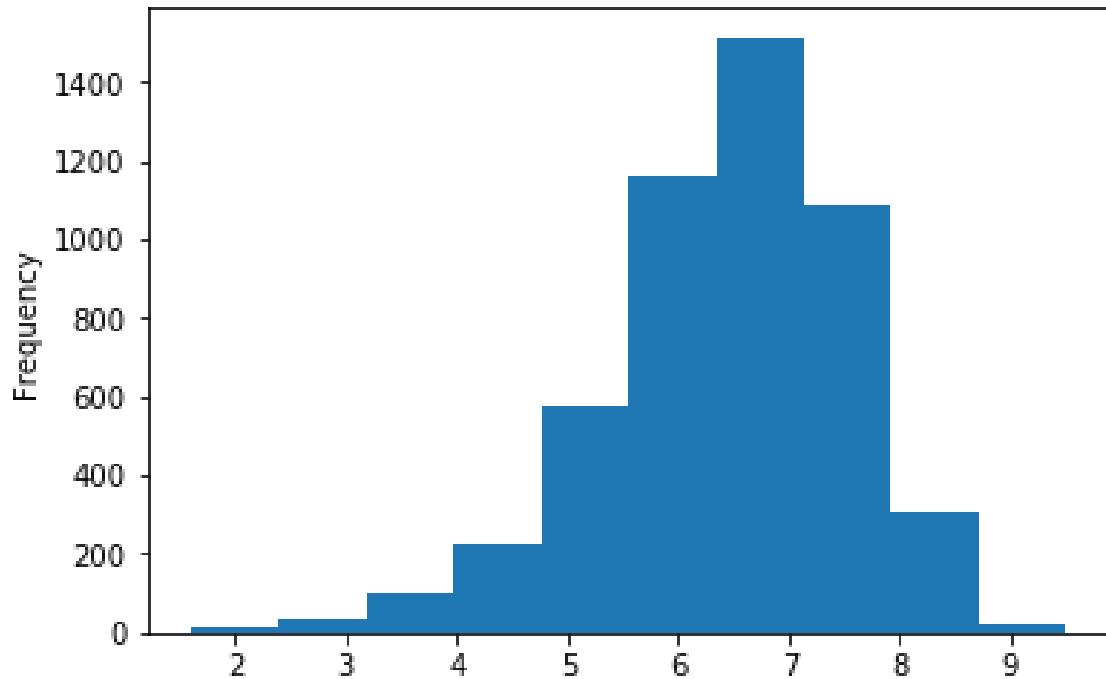
We will draw a bar plot where each bar will represent one of the top 10 data. We can do this by calling the plot method and setting the argument kind to barh. This tells matplotlib to draw a horizontal bar plot.

```
sorted_by_gross['Gross Earnings'].head(10).plot(kind="barh")
plt.show()
```



Let's create a histogram of Scores to check the distribution of Scores across all data. Histograms are a good way to visualize the distribution of a data set. We use the plot method on the Scores series from our data DataFrame and pass it the argument kind="hist".

```
data['Score'].plot(kind="hist")  
plt.show()
```



This data visualization suggests that most of the Scores fall between six and eight.

Getting statistical information about the data

Pandas has some very handy methods to look at the statistical data about our data set. For example, we can use the describe method to get a statistical summary of the data set.

```
data.describe()
```

describe method

The describe method displays below information for each of the columns.

- the count or number of values
- mean
- standard deviation
- minimum, maximum
- 25%, 50%, and 75% quantile

Please note that this information will be calculated only for the numeric values.

We can also use the corresponding method to access this information one at a time. For example, to get the mean of a particular column, you can use the mean method on that column.

```
data["Gross Earnings"].mean()  
48468407.526809327
```

Just like mean, there are methods available for each of the statistical information we want to access.

Pivot Table in pandas

Advanced Excel users also often use pivot tables. A pivot table summarizes the data of another table by grouping the data on an index and applying operations such as sorting, summing, or averaging. You can use this feature in pandas too.

We need to first identify the column or columns that will serve as the index, and the column(s) on which the summarizing formula will be applied. Let's start small, by choosing Year as the index column and Gross Earnings as the summarization column and creating a separate DataFrame from this data.

```
data_subset = data[['Year', 'Gross Earnings']]
data_subset.head()
```

	Year	Gross Earnings
0	1916.0	NaN
1	1920.0	3000000.0
2	1925.0	NaN
3	1927.0	26435.0
4	1929.0	9950.0

We now call `pivot_table` on this subset of data. The method `pivot_table` takes a parameter `index`. As mentioned, we want to use `Year` as the index.

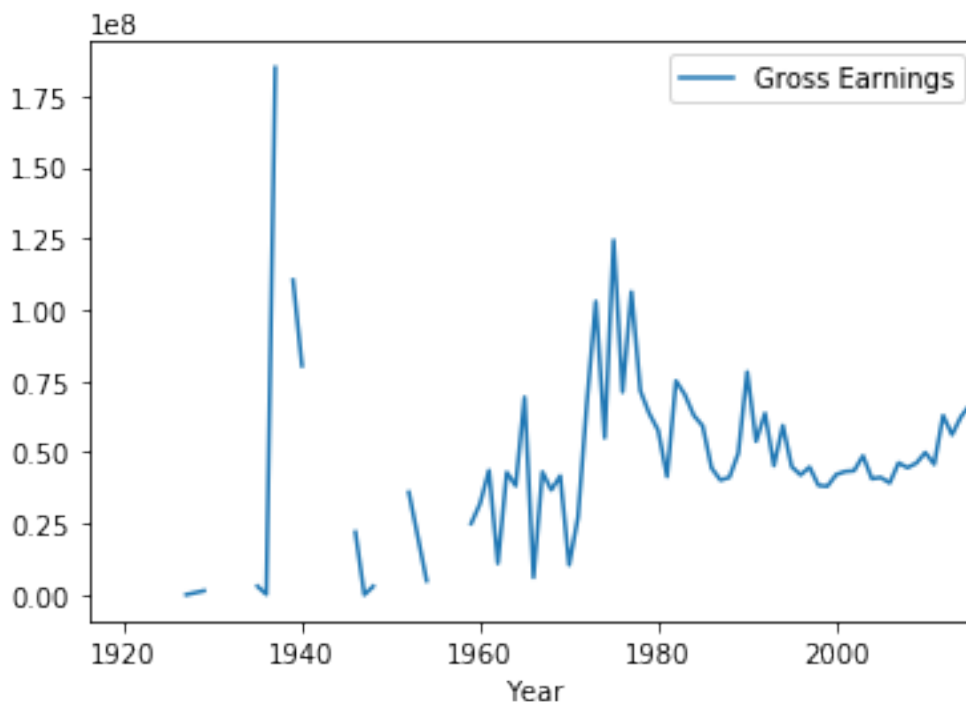
```
earnings_by_year = data_subset.pivot_table(index=['Year'])
earnings_by_year.head()
```

	Gross Earnings
Year	
1916.0	NaN
1920.0	3000000.0
1925.0	NaN
1927.0	26435.0
1929.0	1408975.0

This gave us a pivot table with grouping on `Year` and summarization on the sum of `Gross Earnings`. Notice, we didn't need to specify `Gross Earnings` column explicitly as pandas automatically identified it the values on which summarization should be applied.

We can use this pivot table to create some data visualizations. We can call the plot method on the DataFrame to create a line plot and call the show method to display the plot in the notebook.

```
earnings_by_year.plot()  
plt.show()
```



We saw how to pivot with the single column as the index.

Exporting the results to Excel

If you're going to be working with colleagues who use Excel, saving Excel files out of pandas is important. You can export or write a pandas DataFrame to an Excel file using pandas to_excel method. Pandas uses the xlwt Python module internally for writing to Excel files. The to_excel method is called

on the DataFrame we want to export. We also need to pass a filename to which this DataFrame will be written.

```
data.to_excel('output.xlsx')
```

Conclusion

- Python using pandas can do a lot of complex data analysis and manipulations, which depending on your need and expertise, can go beyond what we can achieve if we are just using Excel.
- One of the major benefits of using Python and pandas over Excel is that it helps us automate Excel file processing by writing scripts and integrating with our automated data workflow.
- Python also has excellent methods for reading all kinds of data from Excel files.
- Python libraries used for GUI development are easy to use and manipulate.
- We can export our results from pandas back to Excel too using python if that's preferred by our intended audience.