**Automated Data Pipeline**

*Internship Project Report submitted to*

# AITHENT INC.

*For Fulfillment of the successful completion*

*of*

# Internship Program

# *at*

# Aithent Inc.

# *by*

# Kaashvi Jain

# *Under the guidance of*

|  |  |
| --- | --- |
| Mr. Pravin Kumar Shrivastava  **(Reporting Manager)** | Miss Priya Jain  **(Project Guide)**  Mr. Mohit Kumar Malik  **(Project Guide)** |
| Aithent: Case Management Solutions for Financial Institutions |  |

**ACKNOWLEDGMENTS**

I would like to express my sincere gratitude to Mr. Pravin Kumar Shrivastava, my Reporting Manager, for his leadership and for facilitating the opportunity to undertake this internship.

I am deeply thankful to my Project Guides, Miss Priya Jain and Mr. Mohit Kumar Malik, for their expert mentorship, timely assistance, and constructive suggestions, which greatly contributed to the successful execution of my tasks.

This internship has been an enriching experience that allowed me to apply my skills in a real-world setting while learning from experienced professionals. I am truly grateful for the trust, support, and guidance extended to me during this journey.

**August 2025**

# Table of Contents

Content Page No.

1. [Introduction](#_bookmark0) 4
2. [Technologies Used 6](#_bookmark3)
3. [Problem Definition 7](#_bookmark5)
4. [Objectives 8](#_bookmark6)
5. [Methodology](#_bookmark7) 10
6. [Work done So Far 13](#_bookmark8)
7. Results 19
8. Challenges Faced 21
9. [Future Work 22](#_bookmark9)

10 Conclusion 23

References 24

Annexure A 25

Annexure B 27

Annexure C 30

# Introduction

## Background and Project Overview

In today’s data-driven organizations, managing large volumes of data such as employee records or project data demands efficient and reliable automation. Manual handling of database exports, report generation, and email communication often leads to errors, data inconsistency, and wasted time. Automating these processes is essential to enhance accuracy, save time and streamline data workflows. During my internship at Aithent Inc., I developed an end-to-end automated workflow to simplify data management, encompassing record storage, export, distribution, and analysis.

## Area of Work

My work focused on creating a seamless automation pipeline involving multiple technologies. This included designing an Oracle SQL Developer students table with automated unique ID generation using sequences and triggers; developing PL/SQL procedures to export data into dated CSV files; building Java applications with reusable methods for database connectivity, procedure execution, and automated emailing of reports; and implementing a Python script to convert exported data into Excel format and generate pivot charts for insightful analysis. The integration of these components created a unified system that minimizes manual intervention.

## Purpose of the Project

The primary goal of this project was to eliminate manual, error-prone processes by automating data export and reporting workflows. By enabling the execution of the entire process through a single Java program that handles database interaction, data export, and email delivery, the solution improves operational efficiency, ensures data integrity, and provides timely, actionable insights through automatic chart generation. This automation helps the organization reduce workload, improve accuracy, and make informed decisions based on reliable data.

# Technologies Used

Java During the internship project, several key technologies and tools were leveraged to design, develop, and automate the data export, reporting, and emailing workflow.

### Oracle SQL Developer:

This tool was used for comprehensive database design, writing SQL queries, and developing PL/SQL programs. Oracle SQL Developer provided a powerful environment for managing the student database and implementing complex database logic. Oracle sequences were employed to automate the generation of unique student IDs, ensuring data integrity and removing the need for manual ID entry. This automation streamlined record management within the database. Stored procedures in PL/SQL were developed to export data from the students table into CSV files. The procedures dynamically generate filenames incorporating the export date, which facilitates organized storage and easy data tracking.

### Java (JDBC, Mail API):

Java was used to develop multiple applications that integrated various workflow steps. JDBC (Java Database Connectivity) enabled reusable, secure database connections. Using JavaMail API, emails were programmatically sent with the exported CSV files attached, automating report distribution. The Java code was structured around reusable methods, allowing clean, modular, and maintainable code. Method chaining was applied so that running the email-sending Java program automatically triggered the database connection and export procedure execution, creating a seamless workflow.

### Python

Python scripts leveraged libraries such as pandas for data manipulation, openpyxl for Excel file creation, and matplotlib for chart generation. This enabled the conversion of exported CSV files to Excel format and the automatic generation of pivot charts, delivering visual insights into the data.

# Problem Definition

The data management process at Aithent Inc. currently involves multiple manual steps for exporting records, generating reports, and sending emails, which is both inefficient and prone to errors. The organization requires a comprehensive automated system that streamlines database exports, automates unique ID generation, and facilitates the seamless emailing of exported data files. This manual approach introduces several challenges:

* **Multiple Manual Steps:** Exporting database records requires executing several discrete steps, increasing the likelihood of human error and omissions.
* **ID Duplication Risk:** Without automation, assigning unique identifiers to records manually can lead to duplication, compromising data integrity.
* **Repetitive Emailing Tasks:** Sending files manually via email is time-consuming and prone to delays or mistakes.

The primary goal of this project was to develop an end-to-end automated workflow that integrates all these steps. By running a single Java program, the system automatically establishes a database connection, executes the export procedure to generate a CSV file, attaches the file to an email, and sends it to designated recipients. This automation eliminates manual intervention, reduces errors, and accelerates the reporting process, enabling more reliable and timely data management.

# Objectives

To address the challenges of manual data management and reporting, this project is dedicated to developing an automated workflow that streamlines database operations, export processes, and report distribution.

By leveraging technologies such as Oracle SQL Developer, PL/SQL, Java, and Python, the goal is to build a unified system that seamlessly integrates unique ID generation, data export, email notification, and analytical reporting. This automation not only improves data accuracy and reliability but also simplifies complex tasks, making the process more efficient and accessible. The primary objectives of this project include:

1. **Automate ID Generation:** Implement Oracle sequences to automatically generate unique IDs, ensuring data integrity and eliminating manual ID assignment errors.
2. **PL/SQL Procedure for Export:** Develop a robust PL/SQL stored procedure to export database records into CSV files with dynamically generated filenames that include the export data for clear organization and tracking.
3. **Reusable Java methods:** Create a modular and reusable Java methods for establishing secure database connections and executing the export procedure efficiently.
4. **Automated Email Sending:** Develop a Java program that not only sends emails with the exported CSV file as an attachment but also automatically triggers the database export process, enabling seamless integration and automation.
5. **Python-based Pivot Chart Creation:** Implement a Python script to convert exported CSV files into Excel (.xlsx) format and generate pivot charts, providing visual analytics and insights from the data.
6. **Single-Click Secure Execution:** Ensure the entire workflow from database connection, data export, to email sending, is executed securely, error-free, and efficiently through a single Java file, simplifying user interaction and improving productivity.

# Methodology

To achieve the goals set forth by the Automation project, we've embraced a methodical and collaborative approach, encompassing the following key strategies:

**Step 1: Database Design**

The project began with designing the core database structure in Oracle SQL Developer. A students table was created to store student records, including relevant columns such as Student ID, name, marks, and other necessary attributes. To ensure automatic unique ID assignment and maintain data integrity, an Oracle sequence was implemented to generate incremental IDs. This sequence was linked through a trigger on the students table so that every new record automatically receives a unique student ID without manual intervention. Example: The sequence generator increments with each inserted row, and the trigger assigns this generated value to the student's ID column automatically upon insertion.

**Step 2: PL/SQL Procedure for Export**

To automate data extraction, a PL/SQL stored procedure was developed that exports the contents of the students table into a CSV file. This procedure dynamically creates the export file name by embedding the current export date in the filename, facilitating organized and chronological data storage. This process replaces manual export operations, ensuring consistency and reducing user errors. Overview: The procedure selects all student data, formats it for CSV output, and saves it to a designated directory with a filename like students\_YYYYMMDD.csv.

**Step 3: Java Development**

Integration and automation of the export and emailing processes were implemented using Java, divided into three modular files with reusable methods:

* **DBConn.java:**  
  Contains a reusable method to establish a secure and efficient JDBC connection to the Oracle database. This method standardizes connection handling for use across other Java files.
* **ExecuteProcedure.java:**  
  Calls the reusable database connection method and executes the PL/SQL export procedure. This file encapsulates procedure execution in a reusable method, enabling easy triggering of the data export from other Java programs.
* **SendEmail.java:**  
  This file builds on the previous two modules by calling the procedure execution method in ExecuteProcedure.java, thereby chaining the export process. It also configures email settings using the JavaMail API with SMTP authentication to send an email with the dynamically generated CSV file attached. Running this file triggers the full sequence—connect to the DB, export data, and send the email, via a single command.

**Step 4: Python Script for Analytics**

To enable data visualization, a Python script was implemented which performs the following tasks:

* Reads the exported CSV file.
* Converts the CSV data into an Excel workbook (.xlsx format) using libraries such as pandas and openpyxl.
* Generates pivot charts from the processed data using matplotlib or similar visualization libraries, automatically producing graphical reports that highlight key patterns and insights.

This step adds value by transforming raw data into actionable visual analytics.

**Step 5: Integration and Automation**

The final system integrates all above components into a cohesive automated workflow, centered around the “chain-call mechanism.” By executing the SendEmail.java file alone, the entire process (database connection, PL/SQL export execution, email dispatch with attachment) is triggered automatically. This reduces all manual effort to running a single Java program, streamlining operations and improving efficiency.

# Work done So Far

### Database Design

The foundational step was designing the students table in Oracle SQL Developer. The table was structured to store all necessary student information with appropriate data types. A key design feature was the automation of the primary key generation through a sequence and a trigger to ensure unique and incremental student IDs. Finished the design process, guaranteeing a clear development roadmap with component diagrams, data flow diagrams, and user interface prototypes.

**Table Structure:**

CREATE SEQUENCE seq

START WITH 1

INCREMENT BY 1

NOCYCLE

NOCACHE;

CREATE OR REPLACE PROCEDURE calc\_result IS

BEGIN

UPDATE STUDENTS

SET RESULT = CASE

WHEN (maths + english + science) >= 150 THEN 'PASS'

ELSE 'FAIL'

END;

END;

Create table Students(

ID INT not null,

name varchar(20) not null,

maths int,

english int,

science int,

result varchar(10),

PRIMARY KEY (ID)

);

INSERT INTO STUDENTS (ID, NAME, MATHS, ENGLISH, SCIENCE)

VALUES (seq.nextval, 'A', 75, 78, 71);

BEGIN

calc\_result;

END;

/

1. **PL/SQL Procedure for Export**

A PL/SQL stored procedure was implemented to export student data into a CSV file with a dynamically generated filename including the current date for easy identification.

**Procedure:**

CREATE OR REPLACE PROCEDURE export\_students(p\_filename OUT VARCHAR2) IS

file\_handle UTL\_FILE.FILE\_TYPE;

v\_line VARCHAR2(4000);

v\_filename VARCHAR2(100);

v\_count NUMBER := 0;

v\_exported NUMBER := 0;

BEGIN

SELECT COUNT(\*) INTO v\_count FROM students;

v\_filename := 'students\_' || TO\_CHAR(SYSDATE, 'YYYYMMDD') || '.csv';

file\_handle := UTL\_FILE.FOPEN('MY\_DIR', v\_filename, 'W');

UTL\_FILE.PUT\_LINE(file\_handle, 'ID,NAME,MATHS,ENGLISH,SCIENCE,RESULT');

FOR rec IN (SELECT ID, NAME, MATHS, ENGLISH, SCIENCE, RESULT FROM STUDENTS) LOOP

v\_line := rec.ID || ',' || rec.NAME || ',' || rec.MATHS || ',' || rec.ENGLISH || ',' || rec.SCIENCE || ',' || rec.RESULT;

UTL\_FILE.PUT\_LINE(file\_handle, v\_line);

v\_exported := v\_exported + 1;

END LOOP;

UTL\_FILE.FCLOSE(file\_handle);

DBMS\_OUTPUT.PUT\_LINE(v\_exported || ' number of rows exported.');

IF v\_count = v\_exported THEN

DBMS\_OUTPUT.PUT\_LINE('All rows exported successfully.');

ELSE

DBMS\_OUTPUT.PUT\_LINE('Exported row count does NOT match table row count.');

END IF;

p\_filename := v\_filename;

END;

/

**Sample filename:** students\_ 20250810.csv

1. **Java Program Implementation**

The Java code was split into three main files for modularity and reusability:

* **DBConnection.java:** Contains reusable method to establish and manage database connections using JDBC.

public class DBConn {

// Resuable method

    public static Connection getConnection() throws ClassNotFoundException, SQLException {

        String jdbcUrl = "jdbc:oracle:thin:@localhost:1521:orcl";

        String username = "system";

        String password = "xxxx";

        Class.forName("oracle.jdbc.driver.OracleDriver");

        return DriverManager.getConnection(jdbcUrl, username, password);

    }

* + **ExecuteProcedure.java:** Utilizes DBConnection to connect to the database and calls the PL/SQL export procedure.

public class ExportProcedure {

    // Reusable method

    public static String ExportFile() throws Exception {

        Connection conn = null;

        CallableStatement stmt = null;

        try {

            conn = DBConn.getConnection(); // connecting to the db

            System.out.println("Connection successful!");

            stmt = conn.prepareCall("{call export\_students(?)}"); // calling the procedure

            stmt.registerOutParameter(1, Types.VARCHAR);

            stmt.execute(); // executing the procedure

            // Filename

            String filename = stmt.getString(1);

            String fullpath = "C:\\Users\\Kaashvi Jain\\SQLORACLE\\" + filename;

            System.out.println("File exported to: " + fullpath);

            return fullpath;

        } finally{

            if (stmt != null) stmt.close(); // closing the statement

            if (conn != null) conn.close(); // closing the connection

        }

    }

* + **SendEmail.java:** The main program which invokes ExecuteProcedure, retrieves the exported CSV file, and sends it as an email attachment using JavaMail API configured with SMTP settings.

//smtp server

        Properties props = new Properties();

        props.put("mail.smtp.host", "smtp.gmail.com");

        props.put("mail.smtp.port", "587");

        props.put("mail.smtp.auth", "true");

        props.put("mail.smtp.starttls.enable", "true");

        //login

        final String username = "xxx@gmail.com";

        final String password = "xxxx xxxx xxxx xxxx "; //app password

        //authenticate

        Session session = Session.getInstance(props, new Authenticator() {

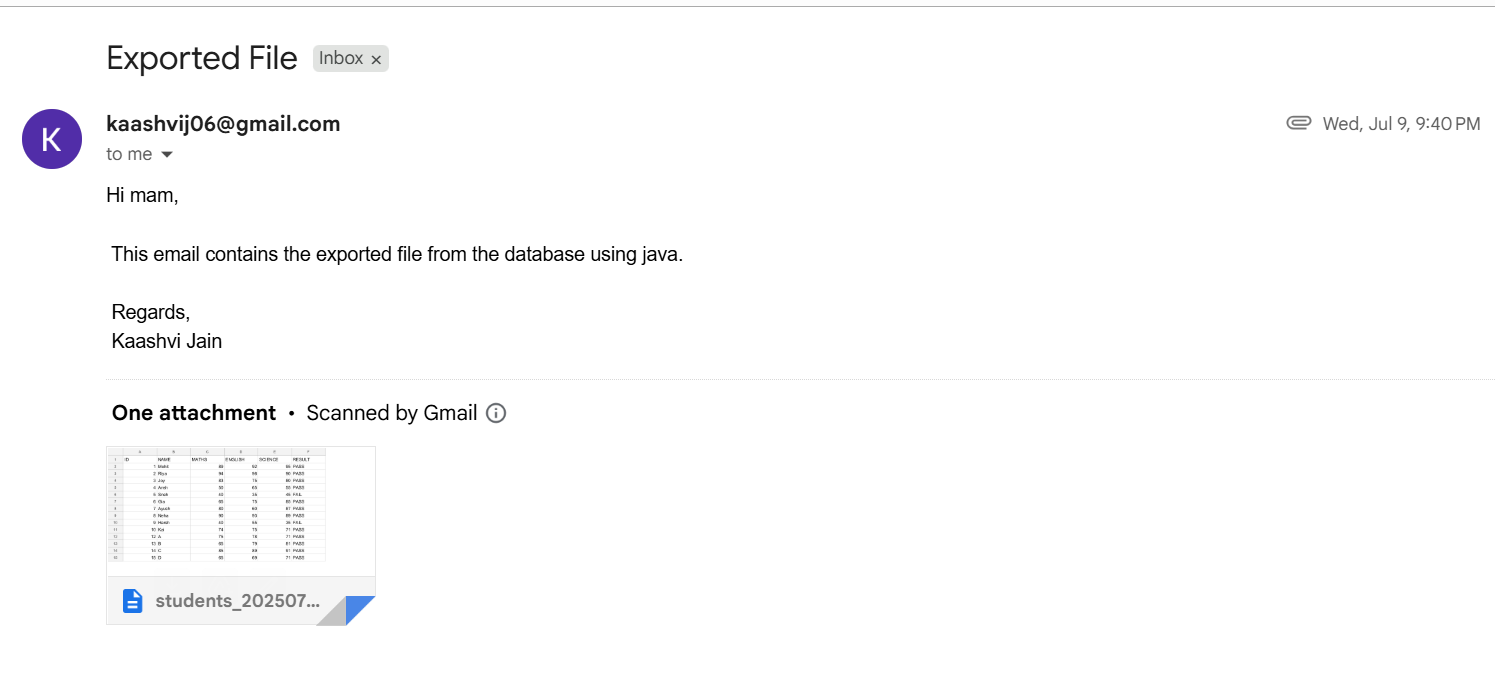
            protected PasswordAuthentication getPasswordAuthentication() {

                return new PasswordAuthentication(username, password);

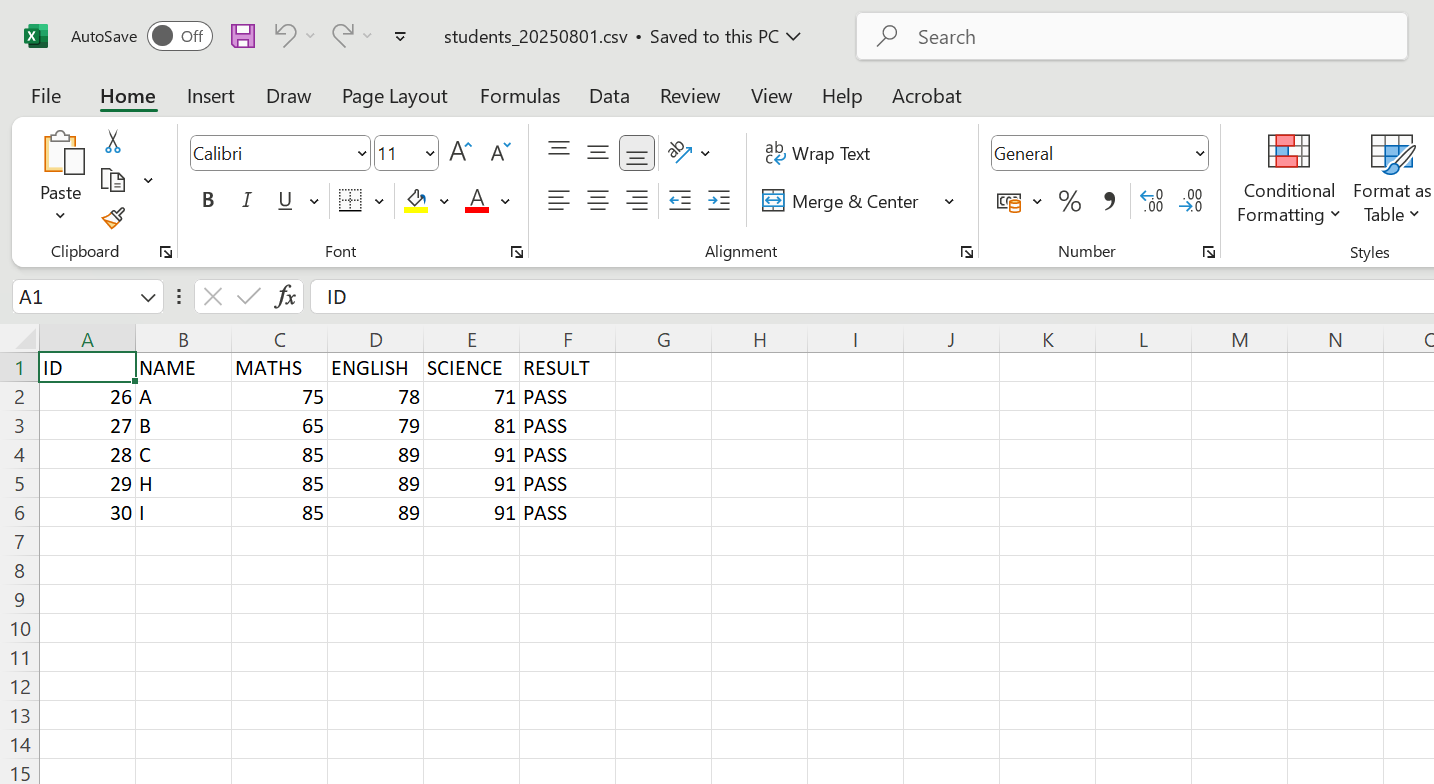
            }

        });

An email sent using **SendEmail.java:**



A sample exported **CSV file:**



1. **Python Script for Pivot Charts**

A Python script converts the exported CSV to Excel format (.xlsx) and generates pivot charts for data visualization. This script uses libraries like pandas for data processing, openpyxl for Excel file handling, and matplotlib for plotting charts.

Code snippet:

df = pd.read\_csv('students\_20250801.csv')

df.to\_excel(output\_excel, index=False)

wb = load\_workbook(output\_excel)

# chart info

def add\_subject\_chart(subject, col\_index, chart\_pos):

    chart = BarChart()

    chart.title = f"{subject} Marks by Student"

    chart.y\_axis.title = "Marks"

    chart.x\_axis.title = "Student"

    data = Reference(ws, min\_col=col\_index, min\_row=1, max\_row=ws.max\_row)

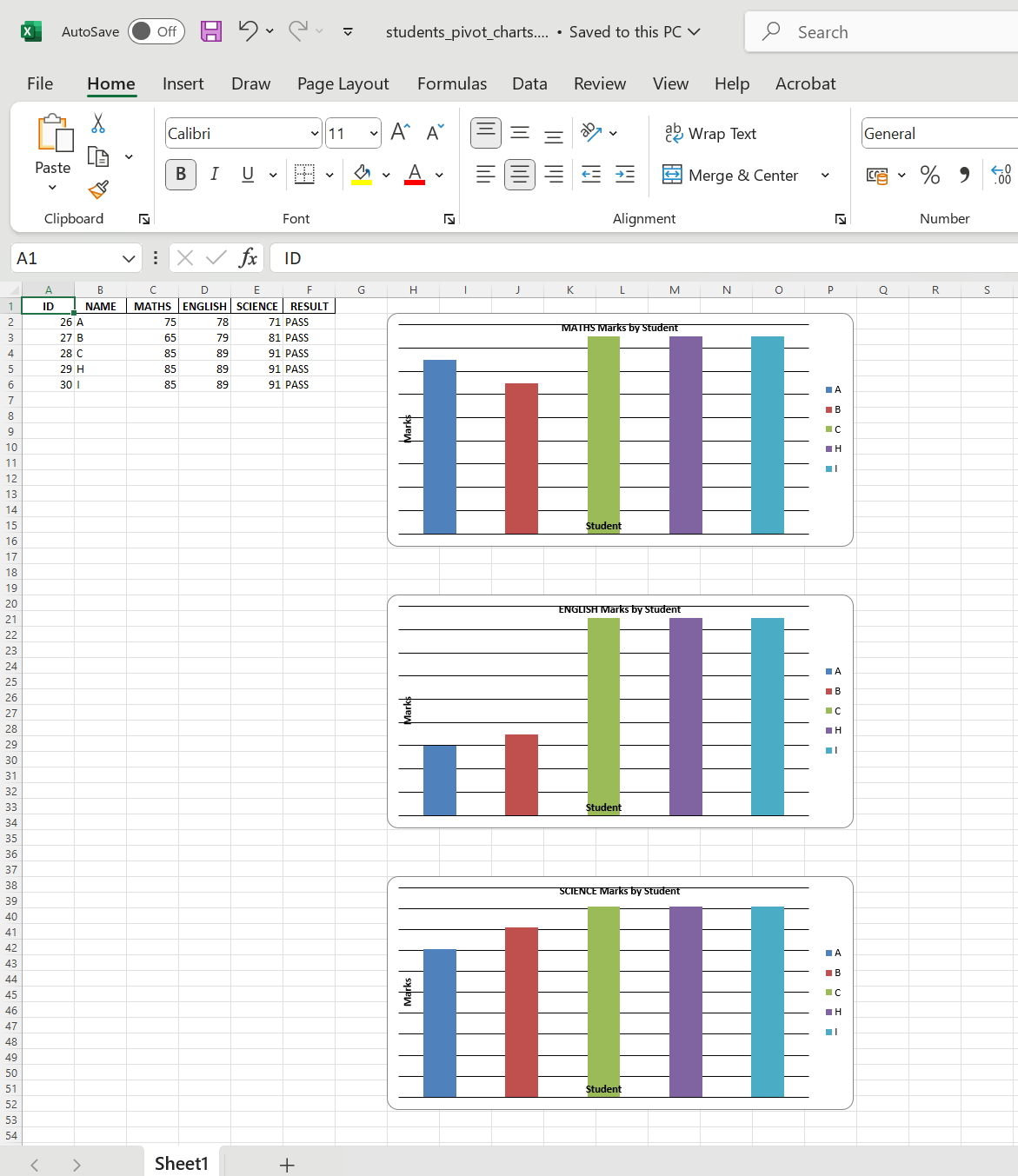
    cats = Reference(ws, min\_col=2, min\_row=2, max\_row=ws.max\_row)

    chart.add\_data(data, titles\_from\_data=True)

    chart.set\_categories(cats)

    ws.add\_chart(chart, chart\_pos)

**XLXS file** containing data visualization:



# Results

Throughout the internship, the project delivered significant improvements and demonstrated clear benefits over manual processing:

**Execution Time Improvement**

The automated workflow drastically reduced the time required for routine data management tasks. Whereas manual export, emailing, and report generation could take from 30 minutes to upto several hours involving multiple manual steps, the automated system accomplishes the entire process in just a few minutes or less. Running a single Java program triggers all steps, making the process highly efficient and freeing up valuable time for other tasks.

**Prevention of ID Duplication**

By utilizing Oracle sequences combined with triggers for ID generation, the system ensured that each student record receives a unique and sequential identifier automatically. This mechanism eliminates the risk of duplicate IDs that commonly occur with manual data entry. The sequence generator increments reliably with each insertion, and the trigger enforces assignment without manual input, maintaining data integrity across the database.

**One-File Execution Demonstration**

The core achievement of this project is the seamless integration of all components through the SendEmail.java file. Executing this single Java program initiates a chain reaction: it first establishes a secure database connection, then executes the PL/SQL procedure to export student data as a CSV file with a dynamic filename reflecting the export date, and finally sends the exported file as an email attachment.

This one-command execution not only eliminates multiple manual steps but also reduces human error and provides timely, visual insights into the data, as demonstrated by the successfully generated pivot charts and the emailed report attachments.

# Challenges Faced

During the internship project, several technical challenges were encountered that helped deepen my understanding and problem-solving skills:

* **Oracle SQL Developer Setup:** Initially, I faced difficulties in setting up Oracle SQL Developer on my computer. Configuring the development environment to connect with the Oracle database and ensuring smooth operation required troubleshooting installation steps and system compatibility issues.
* **Oracle SMTP Package Limitations:**I attempted to use the UTL & SMTP package within Oracle SQL Developer for sending emails but encountered setup and functional limitations. Due to these challenges, the project shifted to using the JavaMail API, which offered more flexibility and security for email handling.
* **JavaMail Security Settings:** Configuring JavaMail to handle secure SMTP connections, particularly with providers like Gmail, was challenging due to frequent changes in security protocols. This required adapting authentication strategies and enabling SSL/TLS to successfully send emails.
* **File Path and Permission Issues:** Exporting data as CSV files from the database was occasionally blocked by file path misconfigurations or insufficient write permissions. Careful configuration of directories and permissions was necessary to enable smooth and consistent file exports.

# Future Work

The current project establishes a strong foundation for automated data management; however, there are several opportunities to enhance and expand its capabilities further:

* **Scheduling Exports and Emails Automatically:** Automating the scheduling of export and email tasks using tools like cron jobs or Windows Task Scheduler can make the process completely hands-off, enabling regular, timely data processing without user intervention.
* **Adding a User Interface for Non-Technical Users:** Developing a simple, intuitive UI would allow non-technical users to trigger exports, configure email recipients, and view reports without interacting directly with code or command-line programs.
* **Integration with Cloud Storage Services:** Extending the system to upload exported files and generated reports directly to cloud platforms such as Google Drive or Dropbox would enhance accessibility and secure backup of critical data.

# Conclusion

This internship project enabled me to gain significant practical experience and technical skills:

* Mastered Oracle SQL automation, applying sequences, triggers, and stored procedures to streamline database operations.
* Developed proficiency in Java JDBC for database connectivity and JavaMail API for secure automated email dispatch.
* The project provided hands-on insight into integrating multiple technologies, bridging database management with Java programs and Python-based data analytics.
* Most importantly, I successfully fulfilled my mentor’s goal by creating a unified, automated workflow that reduces manual effort to running a single file, improving overall efficiency and reliability in data management.

This experience enhanced my understanding of real-world software development challenges and solutions, preparing me for future projects requiring integration across diverse technical stacks.

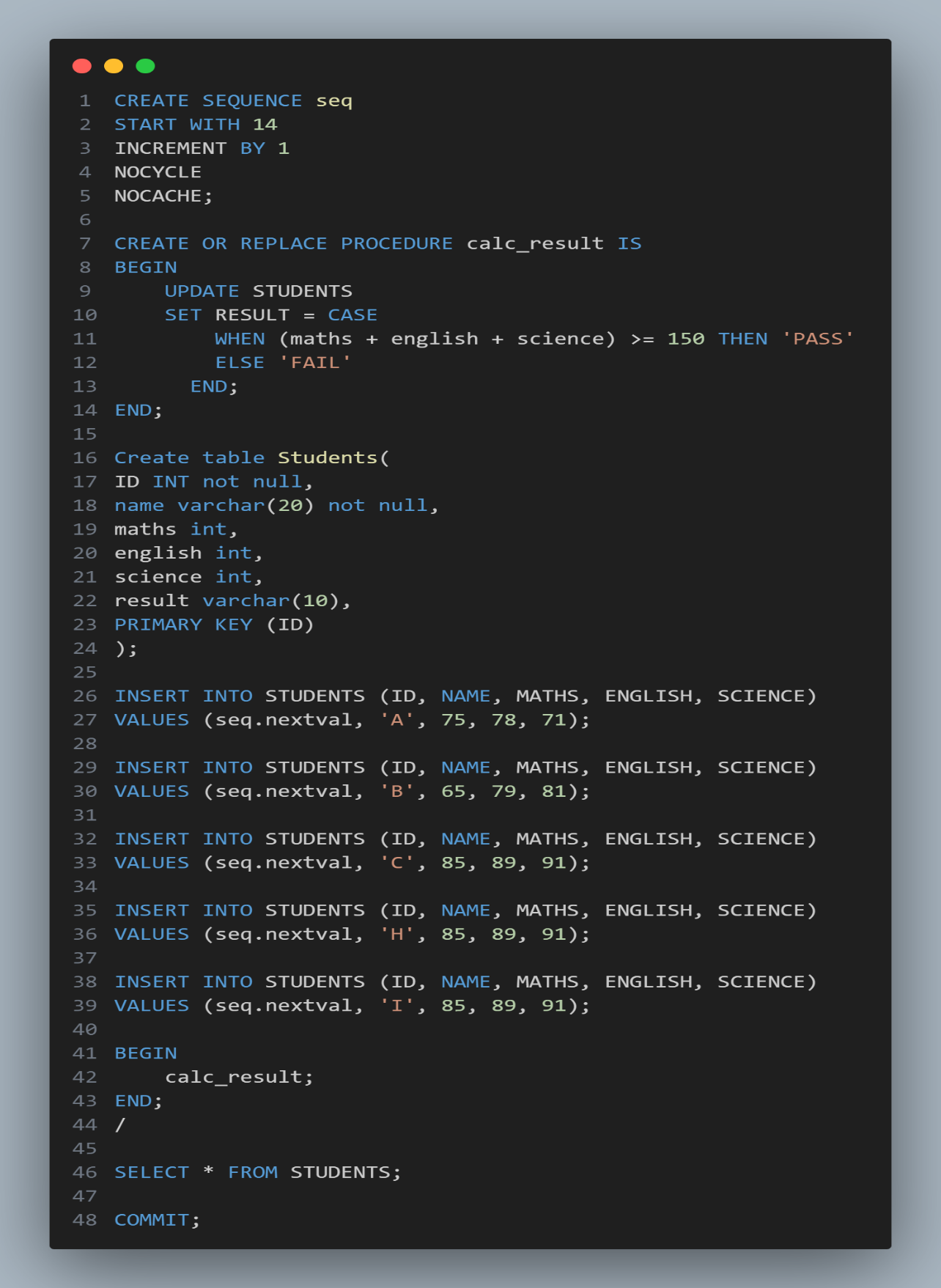
# References

1. Oracle SQL Documentation. <https://docs.oracle.com/en/database/oracle/oracle-database/19/sqlrf/>
2. PL/SQL Tutorial <https://www.geeksforgeeks.org/plsql/pl-sql-tutorial/>
3. Email though JavaMail API [https://www.geeksforgeeks.org/java/send-email-using-java-program/](https://www.javatpoint.com/)
4. Python pandas Official Documentation (Data Manipulation)  
   <https://pandas.pydata.org/docs/>
5. Python Pandas Pivot Table and Chart Tutorials  
   <https://jakevdp.github.io/PythonDataScienceHandbook/03.09-pivot-tables.html>

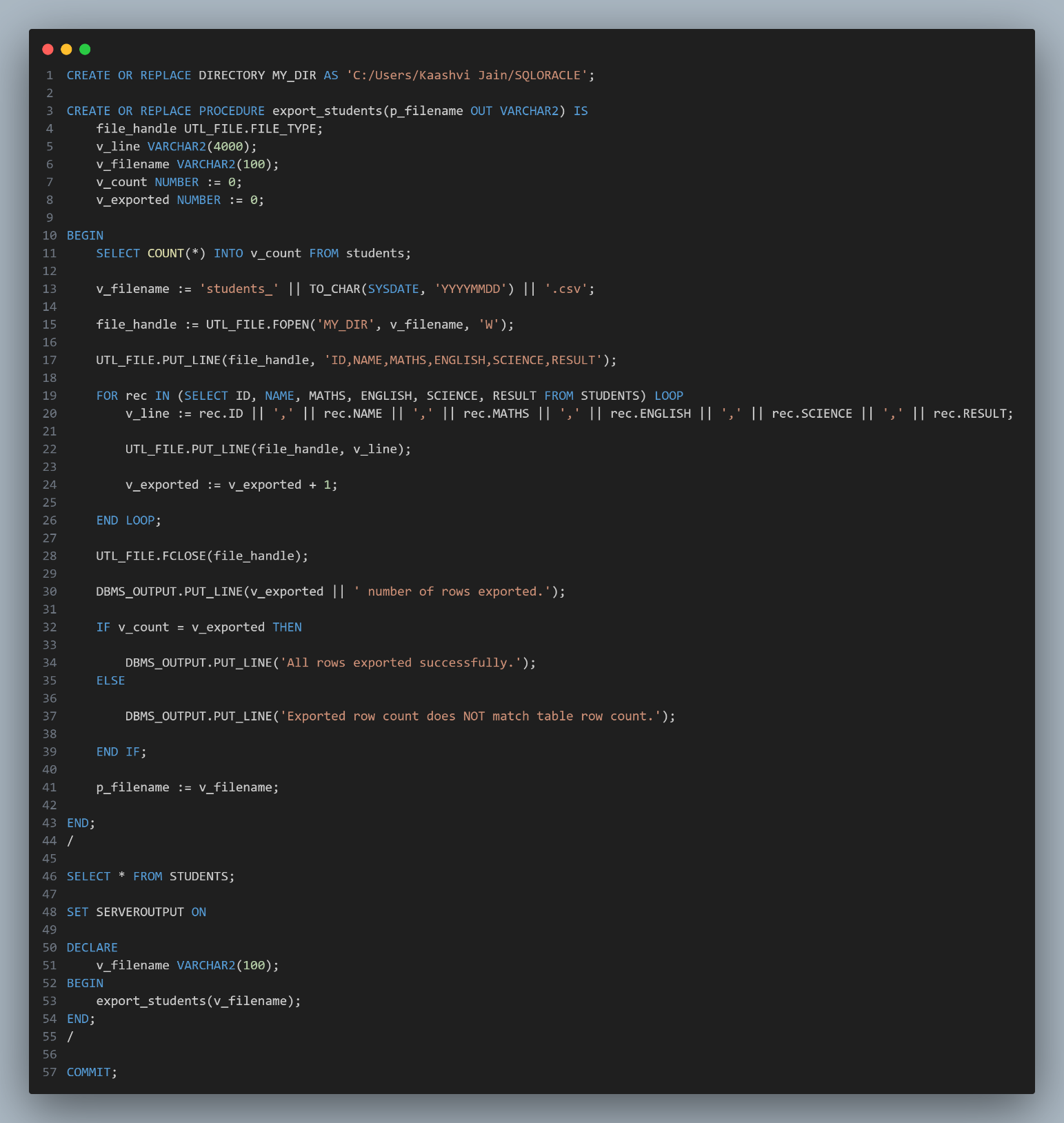
**Annexure A**

**SQL**

**students.sql:**



**Exportprocedure.sql:**



**Annexure B**

**Java Files**

**DBConn.java:**



**ExportProcedure.java:**



**SendEmail.java:**

**Annexure C**

**Python**

**Pivotchart.py:**

