NEHRU SMARAKA VIDYALAYA

BANGALORE

Logo, company name

Description automatically generated

Investigatory Project in **Computer Science**

**Topic:**

**Percentile and Rank Computation for**

**Competitive Entrance Exams**

**Submitted to:**

**Ms. Chandana**

**Submitted by:**

**Tejas Sharma**

Grade **12 ‘C’**

;

NEHRU SMARAKA VIDYALAYA

BANGALORE

Logo, company name

Description automatically generated

CERTIFICATE

This is to certify that **Tejas Sharma**, bona fide student of class XII C , has successfully completed the project titled “Percentile and Rank Computation for Competitive Entrance Exams” prescribed by the Central Board of Secondary Education for the year 2021-22.

|  |  |  |
| --- | --- | --- |
| **Teacher In-Charge** | **Head of the Department** | **Principal** |

# ACKNOWLEDGEMENT

I wish to express my gratitude to my teacher **Ms Chandana C L** whose invaluable guidance has been instrumental in helping me completing the project.

I also wish to express my sincere thanks to our principal **Mrs. Geetha Nagesh** and the management of Nehru Smaraka Vidyalaya for having changed our focus from exam-based learning to knowledge-based learning, which will prepare us for future challenges in academics and beyond.

I wish to thank my parents, friends and all those who have directly or indirectly contributed towards the completion of this project successfully.

# Table of Contents

[1. About Python 5](#_Toc95419584)

[History of Python 5](#_Toc95419585)

[Advantages of Python 5](#_Toc95419586)

[2. Project overview 7](#_Toc95419587)

[Objective 7](#_Toc95419588)

[Background 7](#_Toc95419589)

[Requirements 8](#_Toc95419590)

[Functional Requirements 8](#_Toc95419591)

[Non-Functional Requirements 8](#_Toc95419592)

[3. Python Features used 9](#_Toc95419593)

[Execution Environment 9](#_Toc95419594)

[Python Features used 9](#_Toc95419595)

[4. High level design 10](#_Toc95419596)

[Program components 10](#_Toc95419597)

[Flow Chart 11](#_Toc95419598)

[High Level Flow 11](#_Toc95419599)

[Generate Master Data 11](#_Toc95419600)

[Generate Test Session Data 12](#_Toc95419601)

[12](#_Toc95419602)

[Generate Percentile and Rank Data 12](#_Toc95419603)

[5. Source Code – Complete 13](#_Toc95419604)

[The Main Program 13](#_Toc95419605)

[Generate Test Data 15](#_Toc95419606)

[Compute Percentiles and Rank 17](#_Toc95419607)

[Print Marksheets and Query Module 21](#_Toc95419608)

[Names Used to generate Student Master Data 24](#_Toc95419609)

[6. Output 25](#_Toc95419610)

[1) Generate Student Master Data 25](#_Toc95419611)

[2) Generate Test Sessions Data 26](#_Toc95419612)

[3) Percentile and Rank Allotment 27](#_Toc95419613)

[4) Marks Sheet generation and Query module 28](#_Toc95419614)

[7. Conclusions 31](#_Toc95419615)

[APPLICATIONS AND ADVANTAGES OF THIS PROJECT 31](#_Toc95419616)

[FUTURE SCOPE 31](#_Toc95419617)

[Bibliography 32](#_Toc95419618)

# About Python

Python is a high level programming language which is clear and powerful [1]. Python is an interpreted, general purpose language that supports multiple programming paradigms including structured programming, object oriented programming and functional programming. Python interpreter and standard library is available under open source license making it freely available and distributable.

## History of Python

Python was developed in late 1980s by Guido Van Rossum as a successor to the ABC programming language[2]. Python 2.0 was released in 2000 with new features such as list comprehension and garbage collection. Python 3.0 was released in 2008 as a major revision which is not backward compatible with Python 2.x. Major changes in Python 3.0 include Unicode support, faster execution, better integer division and a vastly improved library for machine learning(ML) and data science. [3]

## Advantages of Python

Being an interpreted language, Python has a very fast development cycle. It is relatively easy to develop and debug Python programs compared to most other widely used programming languages. The edit-test-debug cycle is incredibly fast, thereby having a big impact on the productivity of programmers. The rich standard library and high level built-in data structures of Python makes it the de-facto choice of programmers for a. wide range of applications. Main advantages of Python include

* **Availability:** Python is freely available in both source and binary forms. It is open source under a GPL compatible license [4].
* **Portability:** Python is available for all major platforms including Windows, Mac and all flavours of Unix. The source code requires no change when migrating across various operating systems.
* **Rich standard library:** The standard library accompanying Python makes it suitable for a wide variety of applications including ML, data analytics, web applications, mathematical and scientific applications, graphics, information security etc [5] [6].
* **Ease of use:** Python has rich built-in data structures such as dictionary, list etc. besides having utilities for file handling, parsing regular expressions, connecting to databases, web servers etc. In this project we interface with MySQL using the Python DB interface.
* **Automatic memory management:** Python manages memory automatically making it easier to maintain large software.
* **Object oriented programming:** Python supports object oriented programming such as abstraction, encapsulation, inheritance etc. needed to write maintainable software for large systems.
* **Modularity:** Python supports modules and packages which makes it easy for code reuse and maintainability.
* **Machine Learning and Data Science:** Python libraries such as SciKit Learn [7], NumPy [8] etc. are the best in class for machine learning and data science applications
* **Graphical User Interface:** Python modules such as PyQt5 [9], wxPython [10] etc. provide rich GUI support for applications

# Project overview

## Objective

This project targets the leading educational organizations in India which conduct competitive exams on national scale, specifically management of JEE Mains exam. The software computes percentile and rank for all the students participating in competitive exams such as JEE mains, JEE advanced, NEET etc. and is designed to be easily extensible to other competitive exams both at national level and beyond. Customizable tie breaker algorithm will enable us to adapt this software to the needs of any competitive exam. The software also handles generation of data to match the real world scale, thereby facilitating validation of the tie breaker algorithm and overall testability of the application before deployment.

## Background

JEE Mains is one of the most popular exams in India written by more than 1 million students each year. It is the gateway to admission in most engineering colleges including National Institute of Technology (NITs), Indian Institute of Information Technology (IIITs) etc. and serves as the eligibility criteria for appearing in JEE advanced for admission in various Indian Institute of Technology (IITs). Students are assessed in 3 subjects viz. Mathematics, Physics and Chemistry. Being a competitive exam, students are allocated All India Ranks (AIR). Recently, the exam conducting authority National Testing Agency(NTA) has come up with the scheme of assigning *Percentiles* before assigning AIR. Percentile is an indicator of the number of students whose performances are exceeded by the mentioned student. It is usually expressed as a percentage of total students. For example, if a student obtains 95th percentile, it means that in every group of 100 students, the said candidate’s score is better than 95 students, i.e. his score is in the top 5% of the candidates.

The exam is conducted in 2 to 4 batches every year and each batch has between 4 to 8 sessions i.e. exam dates. A student has option to appear in any or all of the batches and is allotted one session (i.e. exam date and 1 particular exam paper) per batch by the NTA. For each session, percentiles are computed based on total marks (out of 300), Mathematics score, Physics score and Chemistry score separately. In case the student chooses to appear in multiple batches, the best of his percentiles computed for each batch (the sessions she appeared in) separately, becomes her overall percentile. Then on comparing total percentile, mathematics percentile, physics percentile and chemistry percentile, the final ranks are allotted. NTA used to apply date of birth as a last resort for tie breaker (the older candidate gets higher rank). That practice has been discontinued now and candidates with identical percentiles in all subjects get identical rank.

For the purpose of this project, we intend to work with the premise of two batches, with 4 sessions in each batch.

## Requirements

### Functional Requirements

* The system should generate unique Id for each student
* The system should allow students to register for multiple batches
* The system should allot unique session in each of the batches, the student has registered for
* The percentile should be calculated up to 7 decimal places
* The rank should be absolute. If 2 candidates get AIR 97, we should skip AIR 98 and allot AIR 99 to the candidate with the next higher score
* The percentile should be computed for each session
* The best percentile of a student across all batches she has appeared for, should be used for ranking
* The system should support queries

### Non-Functional Requirements

* The system should support up to 1.5 million records
* The system should be portable
* The data should be persistent in a database such as MySQL

# Python Features used

## Execution Environment

|  |  |
| --- | --- |
| Language | Python 3.8 |
| Database | MySQL 8.0.23 |
| IDE | Spyder (Anaconda) |
| Platform | macOS (Monterey) |

## Python Features used

In this project I have used the following features of Python and Python library

* **SQL Connector:** Allows the user to interface python and MySQL [11], and type all SQL commands through python along with python modifications. In this program, all the SQL commands are executed through python, not directly in MySQL.
* **YAML Convertor:** Names are stored separately in a YAML file. The program uses a random algorithm to generate a name (first name + last name) for each candidate.
* **getpass module:** Allows the user to securely enter a password without it being displayed on the screen.
* **Lists:** Stores data (usually 1 record at a time) temporarily before writing onto MySQL and temporarily before displaying in the query output.
* **Functions:** Allows us to perform repetitive tasks with a different input parameter (or identical tasks) without writing code multiple times over.
* **csv Module:** Allows the user to read and write from CSV files (text files, entities separated by commas or delimiters, in rows that resemble those in MS Excel). This program (print marksheets and query module) writes data onto CSV files for viewing results in bulk.
* **numpy Module:** Allows the user to perform specific math tasks such as the probability distribution i.e. likelihood of a student having a particular mark range (important in the case of random data generators).

# High level design

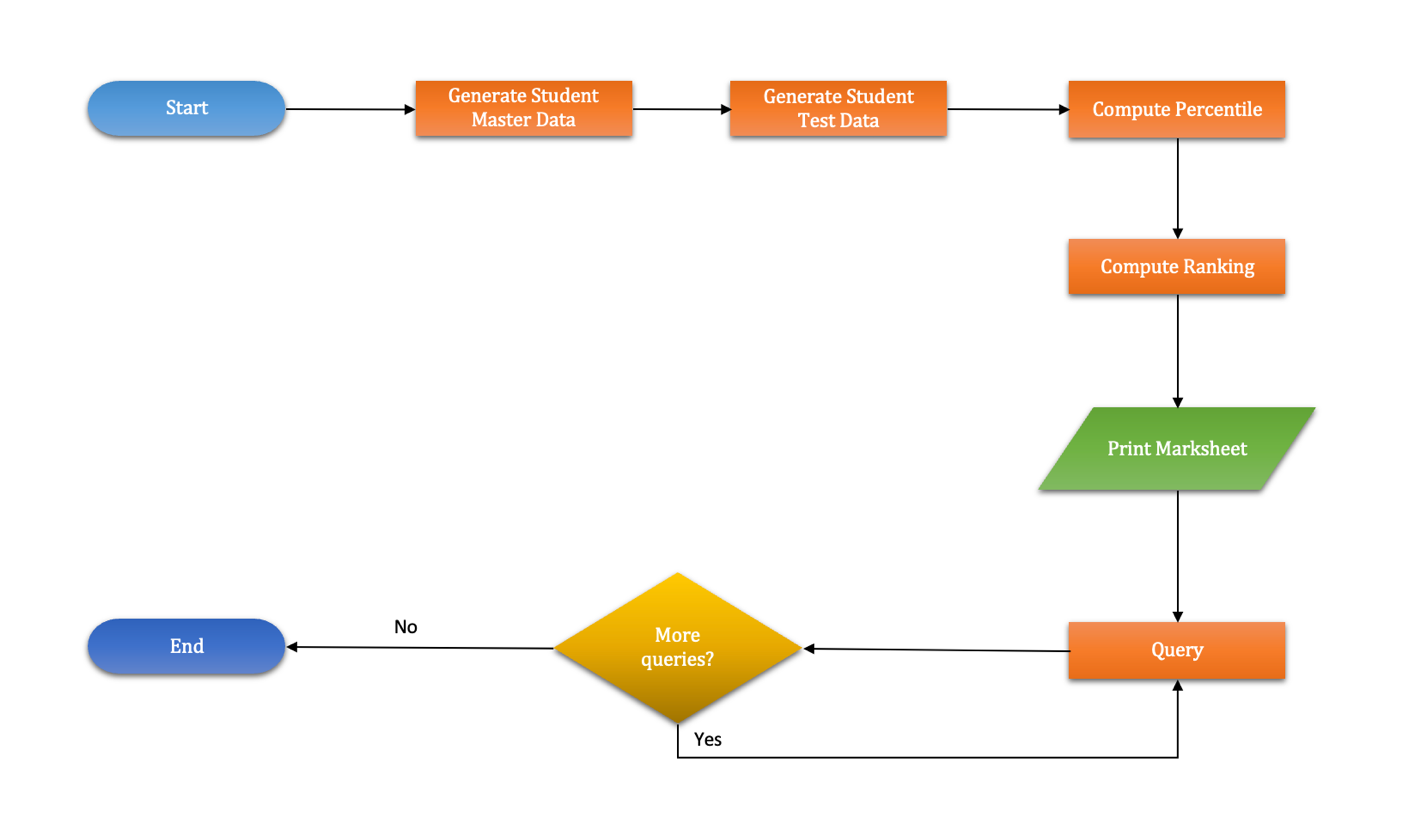
The Application is developed as a set of independent modules that will handle data entry, automatic test data generation, computation and output of results. There is also a query module for analytics.

## Program components

* **Generate Master Data:** This is responsible for generating the student master data that includes fields such as first name, last name, email address, mobile number etc. While it is possible to manually enter data, we recommend using our automatic test generator which uses Python’s random and numpy modules to generate randomly student records. I have tested the generation for 1.5 million students.
* **Generate Test Sessions Data:** This module can be used to register the students for the various test sessions and update the marks. Like the master data, we recommend using the automatic generator to obtain target sample quickly to test the program.
* **Compute Percentile and Rank:** This module uses the MySQL commands to compute the percentile and rank for all the students. Percentile is computed for each batch in the session separately. The best percentile across sessions for each student is considered for the final rank computation. The unique feature of this project is the ability to customize the tie-breaker algorithm.
* **Print Marksheets:** This module generates mark sheets for all the students and can be combined with notification module to email the score card to students.
* **Query Module:** I have provided basic queries to obtain the list of students based on range of rank, marks in individual subjects etc. This module can be enhanced for advanced analytics and fraud detection in future.

## Flow Chart

### High Level Flow



### 

### 

### Generate Master Data

### Generate Test Session Data

### 

### Generate Percentile and Rank Data

# Source Code – Complete

## The Main Program

1 '''

2 This Program generates candidates for JEE MAINS exam

3 Registration Prefix will be used for all Registration ID's, for example 2022

4 Number of students for which scores ranks computed is to be inputed

5 Names, emails and mobile numbers will be auto-generated so will Registration ID's

6 This needs the file names.yaml to be in same folder as this program.

7 '''

8

9 import random

10 import sys

11 import yaml

12 import pandas as pd

13 import numpy as np

14 import mysql.connector as msql

15 import getpass

16

17 # names YAML file containing FirstName, LastName combinations

18 # n number of records

19 # prefix used in registration number

20 def gen\_student\_data(names, n, prefix):

21 with open(names, "r") as inp:

22 name\_dict = yaml.safe\_load(inp)

23 name\_sets = name\_dict.keys()

24

25 name\_set\_lengths = {}

26 total = 0

27

28 for i in range(1, len(name\_sets) + 1):

29 k = "Set" + str(i)

30 first\_names\_len = len(name\_dict[k]['FirstName'])

31 last\_names\_len = len(name\_dict[k]['LastName'])

32 v = first\_names\_len \* last\_names\_len

33 name\_set\_lengths[k] = v

34 total += v

35

36 sets = []

37 probs = []

38

39 for k, v in name\_set\_lengths.items():

40 sets.append(k)

41 probs.append(v / total)

42

43 email\_ids = set()

44 mobile\_list = set()

45

46 password = getpass.getpass('Enter Password for MySQL: ')

47 conect = msql.connect(host = 'localhost', user = 'root', passwd = password)

48 cursor = conect.cursor()

49 cursor.execute('DROP DATABASE IF EXISTS JEE\_Mains')

50 cursor.execute('CREATE DATABASE JEE\_Mains;')

51 cursor.execute('USE JEE\_Mains;')

52 cursor.execute('CREATE TABLE Student\_Master (Reg\_ID Char(11) NOT NULL PRIMARY KEY, First\_Name Varchar(20) NOT NULL, Last\_Name Varchar(20) NOT NULL, EmailID Varchar(50), MobileNo Char(10), Attempt1 Integer, Attempt2 Integer);')

53

54 for i in range(n):

55 key = prefix + str(i).zfill(7)

56 name\_set = np.random.choice(sets, p = probs)

57 first\_name = random.choice(name\_dict[name\_set]['FirstName'])

58 last\_name = random.choice(name\_dict[name\_set]['LastName'])

59 email\_id = get\_email\_id(first\_name, last\_name, email\_ids)

60 contact\_no = get\_mobile\_no(mobile\_list)

61 cursor.execute("INSERT INTO Student\_Master (Reg\_ID, First\_Name, Last\_Name, EmailID, MobileNo) VALUES ('%s', '%s', '%s', '%s', '%s');" %(key, first\_name, last\_name, email\_id, contact\_no))

62 conect.commit()

63

64 cursor.execute('SELECT \* FROM Student\_Master')

65 a = cursor.fetchall()

66 conect.close()

67

68 def get\_email\_id(first\_name, last\_name, email\_ids):

69 email\_id = first\_name + '.' + last\_name

70 ctr = 1

71

72 while (email\_id in email\_ids):

73 email\_id = first\_name + '.' + last\_name + '\_' + str(ctr)

74 ctr += 1

75 email\_ids.add(email\_id)

76 email\_id += '@nta.com'

77 return email\_id

78

79 def get\_mobile\_no(mobile\_list):

80 n = 9000000000 + random.randint(1, 999999999)

81 while n in mobile\_list:

82 n = 9000000000 + random.randint(1, 999999999)

83 mobile\_list.add(n)

84 return n

85

86

87 #\_\_main\_\_

88 names = "names.yaml"

89 n = int(input('Enter the number of students (< 10 million) to generate percentile and rank: '))

90 prefix = input('Enter year or any 4 character prefix of your choice: ')

91 gen\_student\_data(names, n, prefix)

92

## Generate Test Data

1 '''

2 This program generates random marks for the students whose ID's and names were already created in generate\_master\_data program and stored in MySQL. It Stores them in SQL Database.

3 '''

4

5 import numpy as np

6 import random

7 import mysql.connector as msql

8 import getpass

9

10 def get\_student\_marks():

11 password = getpass.getpass('Enter Password for MySQL: ')

12 conect = msql.connect(host = 'localhost', user = 'root', passwd = password, database = 'JEE\_Mains')

13 cursor = conect.cursor()

14 cursor.execute("SELECT Reg\_ID FROM Student\_Master;")

15 lis = cursor.fetchall()

16

17 for i in range(8):

18 cursor.execute('DROP TABLE IF EXISTS Session%s' %(str(i + 1)))

19 cursor.execute('CREATE TABLE Session%s (Reg\_ID CHAR(11) NOT NULL PRIMARY KEY, Math\_Marks Integer, Phy\_Marks Integer, Chem\_Marks Integer, Total Integer, Math\_Percentile FLOAT(10, 7), Phy\_Percentile FLOAT(10, 7), Chem\_Percentile FLOAT(10, 7), Total\_Percentile FLOAT(10, 7));' %(str(i + 1)))

20 conect.commit()

21

22 bands = []

23 probs = []

24 band1 = (-20, -10, 0.1)

25 band2 = (-10, 0, 0.1)

26 band3 = (0, 10, 0.2)

27 band4 = (10, 20, 0.2)

28 band5 = (20, 30, 0.1)

29 band6 = (30, 40, 0.1)

30 band7 = (40, 50, 0.1)

31 band8 = (50, 60, 0.05)

32 band9 = (60, 70, 0.02)

33 band10 = (70, 77, 0.01)

34 band11 = (77, 83, 0.01)

35 band12 = (83, 88, 0.0055)

36 band13 = (88, 92, 0.0025)

37 band14 = (92, 95, 0.0013)

38 band15 = (95, 97, 0.0005)

39 band16 = (97, 99, 0.00018)

40 band17 = (99, 100, 0.00002)

41

42 for i in range(1, 18):

43 a = 'band' + str(i)

44 bands.append(a)

45 a = eval(a)

46 probs.append(a[2])

47

48 for i in range (len(lis)):

49 marks\_band = eval(np.random.choice(bands, p = probs))

50 math\_marks = random.randint(marks\_band[0], marks\_band[1])

51 phy\_marks = random.randint(marks\_band[0], marks\_band[1])

52 chem\_marks = random.randint(marks\_band[0], marks\_band[1])

53 attempts = np.random.choice([1,2,3], p = [0.25, 0.25, 0.5])

54

55 if attempts == 1:

56 num = random.choice([1,2,3,4])

57 cursor.execute("INSERT INTO Session%s (Reg\_ID, Math\_Marks, Phy\_Marks, Chem\_Marks) VALUES ('%s', '%s', '%s', '%s')" %(str(num), str(lis[i][0]), str(math\_marks), str(phy\_marks), str(chem\_marks)))

58 conect.commit()

59 cursor.execute('UPDATE Student\_Master SET Attempt1 = %s WHERE Reg\_ID = "%s";' %(str(num), str(lis[i][0])))

60 conect.commit()

61

62 elif attempts == 2:

63 num = random.choice([5,6,7,8])

64 cursor.execute("INSERT INTO Session%s (Reg\_ID, Math\_Marks, Phy\_Marks, Chem\_Marks) VALUES ('%s', '%s', '%s', '%s')" %(str(num), str(lis[i][0]), str(math\_marks), str(phy\_marks), str(chem\_marks)))

65 conect.commit()

66 cursor.execute('UPDATE Student\_Master SET Attempt2= %s WHERE Reg\_ID = "%s";' %(str(num), str(lis[i][0])))

67 conect.commit()

68

69 elif attempts == 3:

70 num = random.choice([1,2,3,4])

71 cursor.execute("INSERT INTO Session%s (Reg\_ID, Math\_Marks, Phy\_Marks, Chem\_Marks) VALUES ('%s', '%s', '%s', '%s')" %(str(num), str(lis[i][0]), str(math\_marks), str(phy\_marks), str(chem\_marks)))

72 conect.commit()

73 cursor.execute('UPDATE Student\_Master SET Attempt1 = %s WHERE Reg\_ID = "%s";' %(str(num), str(lis[i][0])))

74 conect.commit()

75

76 math\_marks = random.randint(marks\_band[0], marks\_band[1])

77 phy\_marks = random.randint(marks\_band[0], marks\_band[1])

78 chem\_marks = random.randint(marks\_band[0], marks\_band[1])

79

80 num = random.choice([5,6,7,8])

81 cursor.execute("INSERT INTO Session%s (Reg\_ID, Math\_Marks, Phy\_Marks, Chem\_Marks) VALUES ('%s', '%s', '%s', '%s')" %(str(num), str(lis[i][0]), str(math\_marks), str(phy\_marks), str(chem\_marks)))

82 conect.commit()

83 cursor.execute('UPDATE Student\_Master SET Attempt2 = %s WHERE Reg\_ID = "%s";' %(str(num), str(lis[i][0])))

84 conect.commit()

85

86 for i in range(1,9):

87 cursor.execute('UPDATE Session%s SET Total = Math\_Marks + Phy\_Marks + Chem\_Marks;' %(str(i)))

88 conect.commit()

89

90 conect.close()

91 #\_\_main\_\_

92 get\_student\_marks()

## Compute Percentiles and Rank

1 """

2 Program acts on data already generated and updatees tables to set percentile and rank in a final created table.

3 Run after generate\_test\_data, no input except SQL Password

4 """

5

6 import mysql.connector as msql

7 import math

8 import numpy as np

9 import random

10 import getpass

11 import csv

12

13 def compute\_percentile\_rank():

14 password = getpass.getpass('Enter Password for MySQL: ')

15 conect = msql.connect(user = 'root', host = 'localhost', passwd = password, database = 'JEE\_Mains')

16 cursor = conect.cursor()

17

18 for i in range (1, 9):

19 cursor.execute("DROP VIEW IF EXISTS Session%sP;" %(str(i)))

20 cursor.execute('CREATE VIEW Session%sP AS (WITH t AS (SELECT COUNT(Reg\_ID), Total, Math\_Marks, Phy\_Marks, Chem\_Marks FROM Session%s GROUP BY Total, Math\_Marks, Phy\_Marks, Chem\_Marks) SELECT Total, Math\_Marks, Phy\_Marks, Chem\_Marks, ROUND( 100 \* (1 - PERCENT\_RANK() OVER (ORDER BY Total DESC)), 7)Total\_Percentile, ROUND( 100 \* (1 - PERCENT\_RANK() OVER (ORDER BY Math\_Marks DESC)), 7)Math\_Percentile, ROUND( 100 \* (1 - PERCENT\_RANK() OVER (ORDER BY Phy\_Marks DESC)), 7)Phy\_Percentile, ROUND(100 \* (1 - PERCENT\_RANK() OVER (ORDER BY Chem\_Marks DESC)), 7) Chem\_Percentile FROM t)' %(str(i), str(i)));

21 conect.commit()

22

23 cursor.execute('CREATE VIEW Session%sJ AS (SELECT DISTINCT Session%s.Reg\_ID, Session%s.Total, Session%sP.Total\_Percentile FROM Session%s INNER JOIN Session%sP ON (Session%s.Total = Session%sP.Total) ORDER BY Session%sP.Total\_Percentile ASC);' %(str(i), str(i), str(i), str(i), str(i), str(i), str(i), str(i), str(i)));

24 conect.commit()

25 cursor.execute('UPDATE Session%s T1 INNER JOIN Session%sJ T2 ON T1.Reg\_ID = T2.Reg\_ID SET T1.Total\_Percentile = T2.Total\_Percentile;' %(str(i), str(i)))

26 conect.commit()

27 cursor.execute('DROP VIEW Session%sJ;' %(str(i)))

28 conect.commit()

29

30 cursor.execute('CREATE VIEW Session%sJ AS (SELECT DISTINCT Session%s.Reg\_ID, Session%s.Math\_Marks, Session%sP.Math\_Percentile FROM Session%s INNER JOIN Session%sP ON (Session%s.Math\_Marks = Session%sP.Math\_Marks) ORDER BY Session%sP.Math\_Percentile ASC);' %(str(i), str(i), str(i), str(i), str(i), str(i), str(i), str(i), str(i)));

31 conect.commit()

32 cursor.execute('UPDATE Session%s T1 INNER JOIN Session%sJ T2 ON T1.Reg\_ID = T2.Reg\_ID SET T1.Math\_Percentile = T2.Math\_Percentile;' %(str(i), str(i)))

33 conect.commit()

34 cursor.execute('DROP VIEW Session%sJ;' %(str(i)))

35 conect.commit()

36

37 cursor.execute('CREATE VIEW Session%sJ AS (SELECT DISTINCT Session%s.Reg\_ID, Session%s.Phy\_Marks, Session%sP.Phy\_Percentile FROM Session%s INNER JOIN Session%sP ON (Session%s.Phy\_Marks = Session%sP.Phy\_Marks) ORDER BY Session%sP.Phy\_Percentile ASC);' %(str(i), str(i), str(i), str(i), str(i), str(i), str(i), str(i), str(i)));

38 conect.commit()

39 cursor.execute('UPDATE Session%s T1 INNER JOIN Session%sJ T2 ON T1.Reg\_ID = T2.Reg\_ID SET T1.Phy\_Percentile = T2.Phy\_Percentile;' %(str(i), str(i)))

40 conect.commit()

41 cursor.execute('DROP VIEW Session%sJ;' %(str(i)))

42 conect.commit()

43

44 cursor.execute('CREATE VIEW Session%sJ AS (SELECT DISTINCT Session%s.Reg\_ID, Session%s.Chem\_Marks, Session%sP.Chem\_Percentile FROM Session%s INNER JOIN Session%sP ON (Session%s.Chem\_Marks = Session%sP.Chem\_Marks) ORDER BY Session%sP.Chem\_Percentile ASC);' %(str(i), str(i), str(i), str(i), str(i), str(i), str(i), str(i), str(i)));

45 conect.commit()

46 cursor.execute('UPDATE Session%s T1 INNER JOIN Session%sJ T2 ON T1.Reg\_ID = T2.Reg\_ID SET T1.Chem\_Percentile = T2.Chem\_Percentile;' %(str(i), str(i)))

47 conect.commit()

48 cursor.execute('DROP VIEW Session%sJ;' %(str(i)))

49 conect.commit()

50

51 cursor.execute("DROP VIEW IF EXISTS Session%sP;" %(str(i)))

52

53 cursor.execute('DROP TABLE IF EXISTS Student\_FinalScores;')

54 cursor.execute("CREATE TABLE Student\_FinalScores (Reg\_ID Char(11) NOT NULL PRIMARY KEY, Student\_Name VARCHAR(40), Total1 INTEGER, Math1 INTEGER, Phy1 INTEGER, Chem1 INTEGER, Total2 INTEGER, Math2 INTEGER, Phy2 INTEGER, Chem2 INTEGER, TP1 FLOAT(10, 7), TP2 FLOAT(10, 7), MP1 FLOAT(10, 7), MP2 FLOAT(10, 7), PP1 FLOAT(10, 7), PP2 FLOAT(10, 7), CP1 FLOAT(10, 7), CP2 FLOAT(10, 7), Total\_Percentile FLOAT(10, 7), Math\_Percentile FLOAT(10, 7), Phy\_Percentile FLOAT(10, 7), Chem\_Percentile FLOAT(10, 7), Final\_Rank INTEGER);")

55 cursor.execute('SELECT Reg\_ID, Attempt1, Attempt2, First\_Name, Last\_Name FROM Student\_Master;')

56 lis0 = cursor.fetchall()

57 conect.commit()

58

59 for i in range(len(lis0)):

60 regid = lis0[i][0]

61 attempt1 = lis0[i][1]

62 attempt2 = lis0[i][2]

63 S\_name = str(lis0[i][3]) + ' ' + str(lis0[i][4])

64 epsilon = 0.0000001

65 tp1 = mp1 = cp1 = pp1 = tp2 = mp2 = pp2 = cp2 = 0.00

66 t1 = m1 = p1 = c1 = t2 = m2 = c2 = p2 = 'NULL'

67 if attempt1 != None:

68 cursor.execute("SELECT \* FROM Session%s WHERE Reg\_ID = '%s';" %(str(attempt1), str(regid)))

69 lis1 = cursor.fetchone()

70 tp1 = lis1[8]

71 mp1 = lis1[5]

72 pp1 = lis1[6]

73 cp1 = lis1[7]

74 t1 = lis1[4]

75 m1 = lis1[1]

76 p1 = lis1[2]

77 c1 = lis1[3]

78 if attempt2 != None:

79 cursor.execute("SELECT \* FROM Session%s WHERE Reg\_ID = '%s';" %(str(attempt2), str(regid)))

80 lis2 = cursor.fetchone()

81 tp2 = lis2[8]

82 mp2 = lis2[5]

83 pp2 = lis2[6]

84 cp2 = lis2[7]

85 t2 = lis2[4]

86 m2 = lis2[1]

87 p2 = lis2[2]

88 c2 = lis2[3]

89

90 tp = max(tp1, tp2)

91 if abs(tp - tp2) > epsilon:

92 mp, pp, cp, t = mp1, pp1, cp1, t1

93 elif abs(tp - tp1) > epsilon:

94 mp, pp, cp, t = mp2, pp2, cp2, t2

95 else:

96 mp = max(mp1, mp2)

97 if abs(mp - mp2) > epsilon:

98 pp, cp, t = pp1, cp1, t1

99 elif abs(mp - mp1) > epsilon:

100 pp, cp, t = pp2, cp2, t2

101 else:

102 pp = max(pp1, pp2)

103 if abs(pp - pp2) > epsilon:

104 cp, t = cp1, t1

105 elif abs(pp - pp1) > epsilon:

106 cp, t = cp2, t2

107 else:

108 cp = max(cp1, cp2)

109 if abs(cp - cp2) > epsilon:

110 t = t1

111 elif abs(cp - cp1) > epsilon:

112 t = t2

113 else:

114 t = max(t1, t2)

115

116 cursor.execute("INSERT INTO Student\_FinalScores (Reg\_ID, Student\_Name, Total1, Math1, Phy1, Chem1, Total2, Math2, Phy2, Chem2, TP1, MP1, PP1, CP1, TP2, MP2, PP2, CP2, Total\_Percentile, Math\_Percentile, Phy\_Percentile, Chem\_Percentile) VALUES ('%s', '%s', %s, %s, %s, %s, %s, %s, %s, %s, %s, %s, %s, %s, %s, %s, %s, %s, %s, %s, %s, %s);" %(str(lis0[i][0]), S\_name, str(t1), str(m1), str(p1), str(c1), str(t2), str(m2), str(p2), str(c2), str(tp1), str(mp1), str(pp1), str(cp1), str(tp2), str(mp2), str(pp2), str(cp2), str(tp), str(mp), str(pp), str(cp)))

117 conect.commit()

118

119 cursor.execute('DROP VIEW IF EXISTS TempRank;')

120 cursor.execute('CREATE VIEW TempRank AS (SELECT Reg\_ID, Total\_Percentile, Math\_Percentile, Phy\_Percentile, Chem\_Percentile, ROW\_NUMBER() OVER (ORDER BY Total\_Percentile DESC, Math\_Percentile DESC, Phy\_Percentile DESC, Chem\_Percentile DESC) AS Final\_Rank FROM Student\_FinalScores ORDER BY Final\_Rank);')

121 cursor.execute('UPDATE Student\_FinalScores T1 INNER JOIN TempRank T2 ON T1.Reg\_ID = T2.Reg\_ID SET T1.Final\_Rank = T2.Final\_Rank;')

122 cursor.execute('DROP VIEW IF EXISTS TempRank;')

123 conect.commit()

124

125 conect.close()

126

127 #\_\_main\_\_

128 compute\_percentile\_rank()

## Print Marksheets and Query Module

1 """

2 Program allows user to enter student Reg ID and see his scores or alternatively, store data of toppers in a CSV file for users to read using excel or other tools; no. of toppers is decided by user.

3 Run after generate\_percentile\_rank\_data, needs SQL Password and other inputs from user.

4 """

5

6 import mysql.connector as msql

7 import csv

8 import getpass

9

10 password = getpass.getpass('Enter Password for MySQL: ')

11 conect = msql.connect(user = 'root', host = 'localhost', passwd = password, database = 'JEE\_Mains')

12 cursor = conect.cursor()

13

14 while True:

15 print()

16 n = input("Enter 1 to see the markscard of any student of your choice, by Registration ID. \nEnter 2 to obtain the marks and percentile of the toppers. \nEnter 3 to obtain all students any range of ranks, between lower and upper limit rank. \nEnter 4 to obtain all students in a particular range by total marks. \nEnter 0 to exit: ")

17

18 if n == '0':

19 print()

20 print('Thank you!')

21 print()

22 conect.close()

23 break

24

25 elif n == '1':

26 print()

27 regid = input('Enter Registration ID of student: ')

28 print()

29 cursor.execute("SELECT \* FROM Student\_FinalScores WHERE Reg\_ID = '%s'" %(regid))

30 list3 = cursor.fetchall()

31

32 if not list3:

33 print('InvalidRegIDError: Entered Registration ID of Nonexistant Student.')

34 continue

35

36 else:

37 list3 = list3[0]

38 print('PRINTING MARKS CARD OF STUDENT', list3[1])

39 print('Registration ID: ', list3[0])

40 print()

41 print('ATTEMPT 1 MARKS:')

42 print('Maths: ', list3[3], '\nPhysics: ', list3[4], '\nChemistry: ', list3[5], '\nTotal Score: ', list3[2])

43 print()

44 print('ATTEMPT 2 MARKS:')

45 print('Maths: ', list3[7], '\nPhysics: ', list3[8], '\nChemistry: ', list3[9], '\nTotal Score: ', list3[6])

46 print()

47 print('SUBJECT-WISE PERCENTILES:')

48 print('Maths percentile: ', list3[19])

49 print('Physics percentile: ', list3[20])

50 print('Chemistry percentile: ', list3[21])

51 print()

52 print('ONERALL PERCENTILE: ', list3[18])

53 print('FINAL RANK: ', list3[22])

54

55 elif n == '2':

56 no = (input('Enter how many top rankers you want to obtain: '))

57 print()

58 cursor.execute('SELECT Reg\_ID, Student\_Name, Final\_Rank, Total1, Total2, Total\_Percentile, Math\_Percentile, Phy\_Percentile, Chem\_Percentile FROM Student\_FinalScores ORDER BY Final\_Rank LIMIT %s;' %(no))

59 list4 = cursor.fetchall()

60 header = ('Registration ID', 'Student Name', 'Final Rank', 'Attempt 1 Total', 'Attempt 2 Total', 'Overall Percentile', 'Maths Percentile', 'Physics Percentile', 'Chemistry Percentile')

61

62 with open ('Toppers.csv', 'w') as F:

63 writer = csv.writer(F)

64 writer.writerow(header)

65 for row in list4:

66 writer.writerow(row)

67 print('Successfully created CSV File Toppers.csv of Toppers.')

68 elif n == '3':

69 print()

70 Lr = int(input('Enter lower rank limit: '))

71 Hr = int(input('Enter upper rank limit: '))

72 print()

73 cursor.execute('SELECT Reg\_ID, Student\_Name, Final\_Rank, Total1, Total2, Total\_Percentile, Math\_Percentile, Phy\_Percentile, Chem\_Percentile FROM Student\_FinalScores WHERE Final\_Rank BETWEEN %s AND %s ORDER BY Final\_Rank;' %(str(Lr), str(Hr)))

74 list4 = cursor.fetchall()

75 header = ('Registration ID', 'Student Name', 'Final Rank', 'Attempt 1 Total', 'Attempt 2 Total', 'Overall Percentile', 'Maths Percentile', 'Physics Percentile', 'Chemistry Percentile')

76 with open ('Ranklist.csv', 'w') as F:

77 writer = csv.writer(F)

78 writer.writerow(header)

79 for row in list4:

80 writer.writerow(row)

81 print('Successfully created CSV File Ranklist.csv of Rankers between', Lr, 'and', Hr, '.')

82 elif n == '4':

83 print()

84 Lm = int(input('Enter least value of marks for filter: '))

85 Hm = int(input('Enter highest value of marks for filter: '))

86 print()

87 cursor.execute('SELECT Reg\_ID, Student\_Name, Final\_Rank, Total1, Total2, Total\_Percentile, Math\_Percentile, Phy\_Percentile, Chem\_Percentile FROM Student\_FinalScores WHERE GREATEST(Total1, Total2) BETWEEN %s AND %s ORDER BY Final\_Rank;' %(str(Lm), str(Hm)))

88 list4 = cursor.fetchall()

89 header = ('Registration ID', 'Student Name', 'Final Rank', 'Attempt 1 Total', 'Attempt 2 Total', 'Overall Percentile', 'Maths Percentile', 'Physics Percentile', 'Chemistry Percentile')

90 with open ('Markbandlist.csv', 'w') as F:

91 writer = csv.writer(F)

92 writer.writerow(header)

93 for row in list4:

94 writer.writerow(row)

95 print('Successfully created CSV File Markbandlist of Scorers between', Lm, 'and', Hm, '.')

96 else:

97 continue

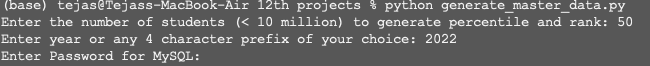
## Names Used to generate Student Master Data

|  |  |
| --- | --- |
| Set1:  FirstName:  - Tejas  - Nikhil  - Aryan  - Siddharth  - Balaji  - Sundar  - Ravi  - Sudha  - Shankar  - Ravishankar  - Suman  - Nitin  - Rajat  - Swapnil  - Vishal  - Siddhi  - Jagruthi  - Dinesh  - Ram  - Krishna  - Shyam  - Srinivas  - Veena  - Mohit  - Manjunath  - Mallikarjun  - Kumar  - Rohit  - Pranav  - Anand  - Arvind  - Sachin  - Virat  - Virendar  - Vijay  - Dhananjay  - Madhav  - Mohan  - Smitha  LastName:  - Sharma  - Bishnoi  - Gupta  - Verma  - Bharadhwaj  - Trivedi  - Tiwari  - Kulkarni  - Bhatt  - Rao  - Gowda  - Gangwar  - Garg  - Dutta  - Chakraborty  - Bannerjee  - Chatterjee  - Kaushik  - Hulse  - Tendulkar  - Kelkar  - Dhamdhere  - Desai  - Vyas  - Patel  - Ranade  - Bhujade  - Gaekwad  - Gadgil  - Patil  - Manjrekar  - Gavaskar | Set2:  FirstName:  - Aamir  - Salman  - Shahrukh  - Akmal  - Abdul  - Riyaz  - Fatima  - Shaheen  - Ayesha  - Zeenat  - Shabana  - Baasha  - Shabeer  - Hamid  - Saif  - Mohammed  LastName:  - Khan  - Ahmed  - Mohammed  - Ansari  - Gul  - Nasser  - Khaif  Set3:  FirstName:  - Joe  - Bijoe  - Steve  - Mark  - Manuel  - Immanuel  - Felix  - Kingsley  - John  - James  - David  - Matthew  - Jeffery  - Joseph  - Isaac  LastName:  - Thomas  - George  - Waugh  - Joy  - Sebastian  - Anthony  - Isaac  - Smith  - Jones |

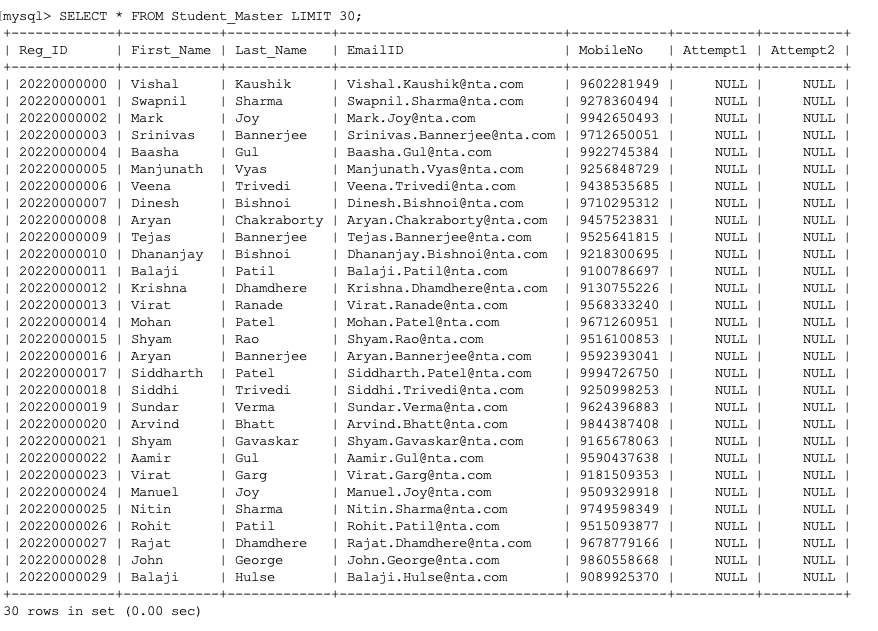
# Output

### Generate Student Master Data

The below screen shot depicts the process of generating the student master data. Our program randomly chooses first name, last name combination from a list to facilitate auto generation of master data.



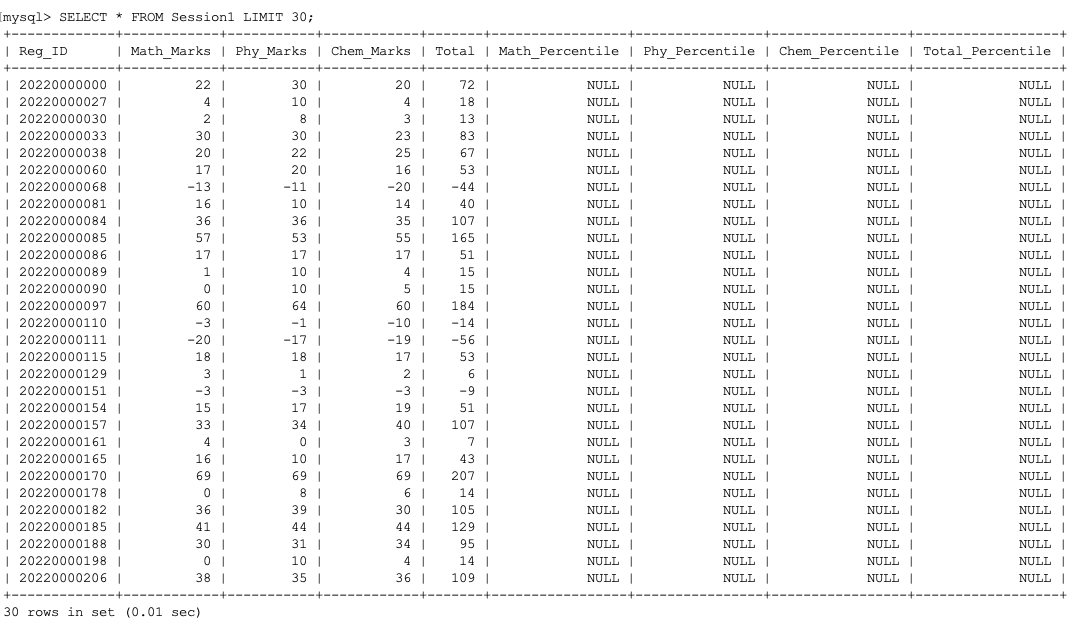
Student master data is generated and stored in MySQL in the student master table. A subset of the master data generated is given below.



### Generate Test Sessions Data

Once the master data for the students is created, we go ahead and generate the test data for the students. A student may register for either or both of the sessions.

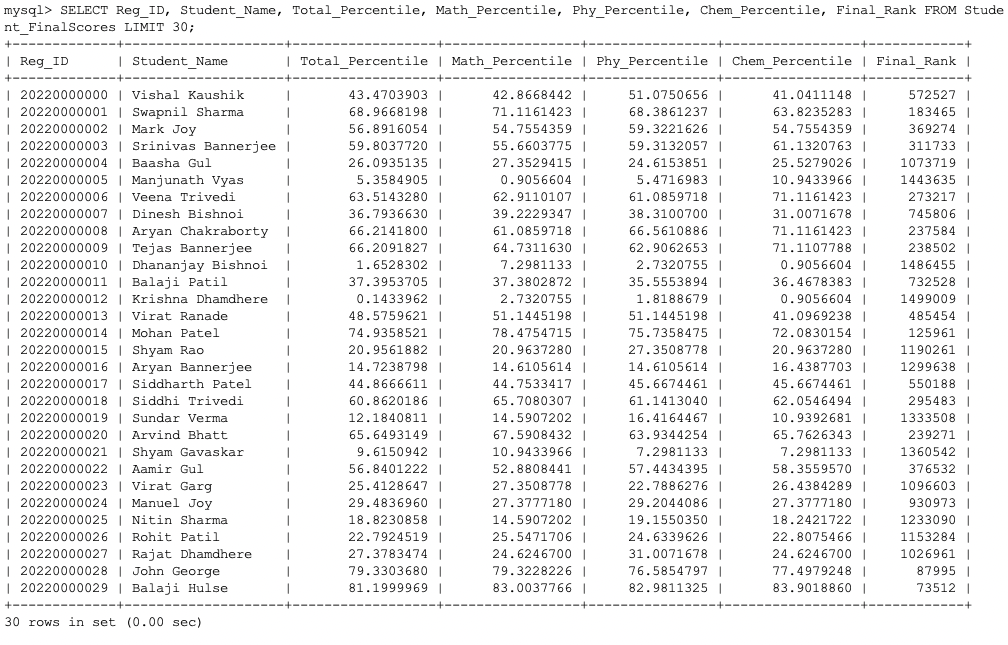
Output is stored in MySQL using a separate table for each test session. The below table gives the snapshot of the test data.



### Percentile and Rank Allotment

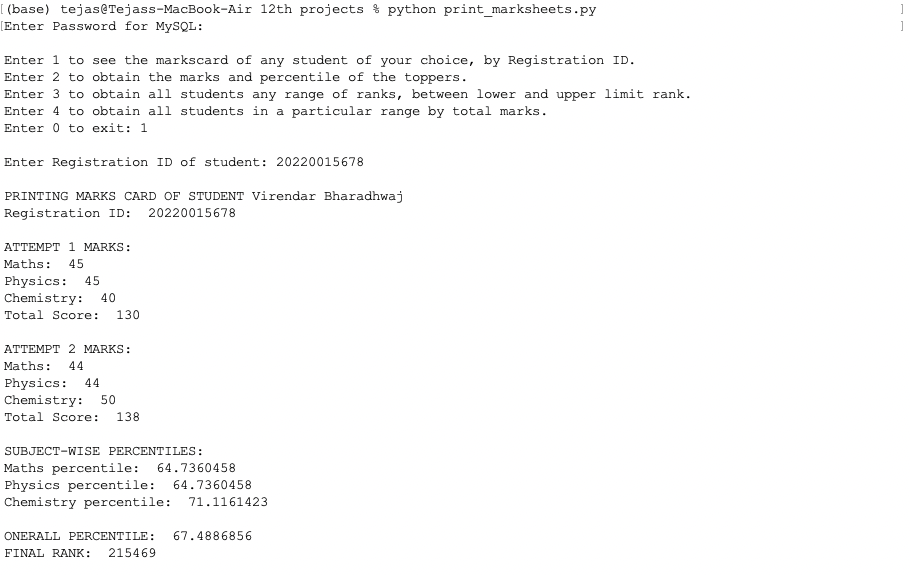
Once the tables for both the test sessions are generated, we are ready to run the main query in SQL to compute the percentile and rank for the students.

The percentile is computed for all the students and the best percentile for each student is considered for rank allotment. Sample rank list generated as a CSV file is given below.

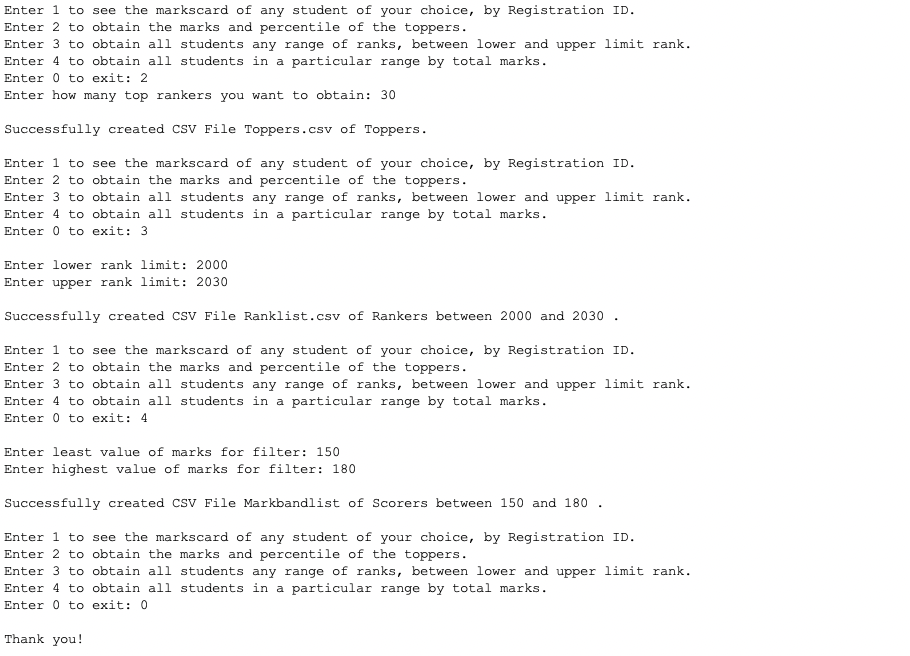


### Marks Sheet generation and Query module

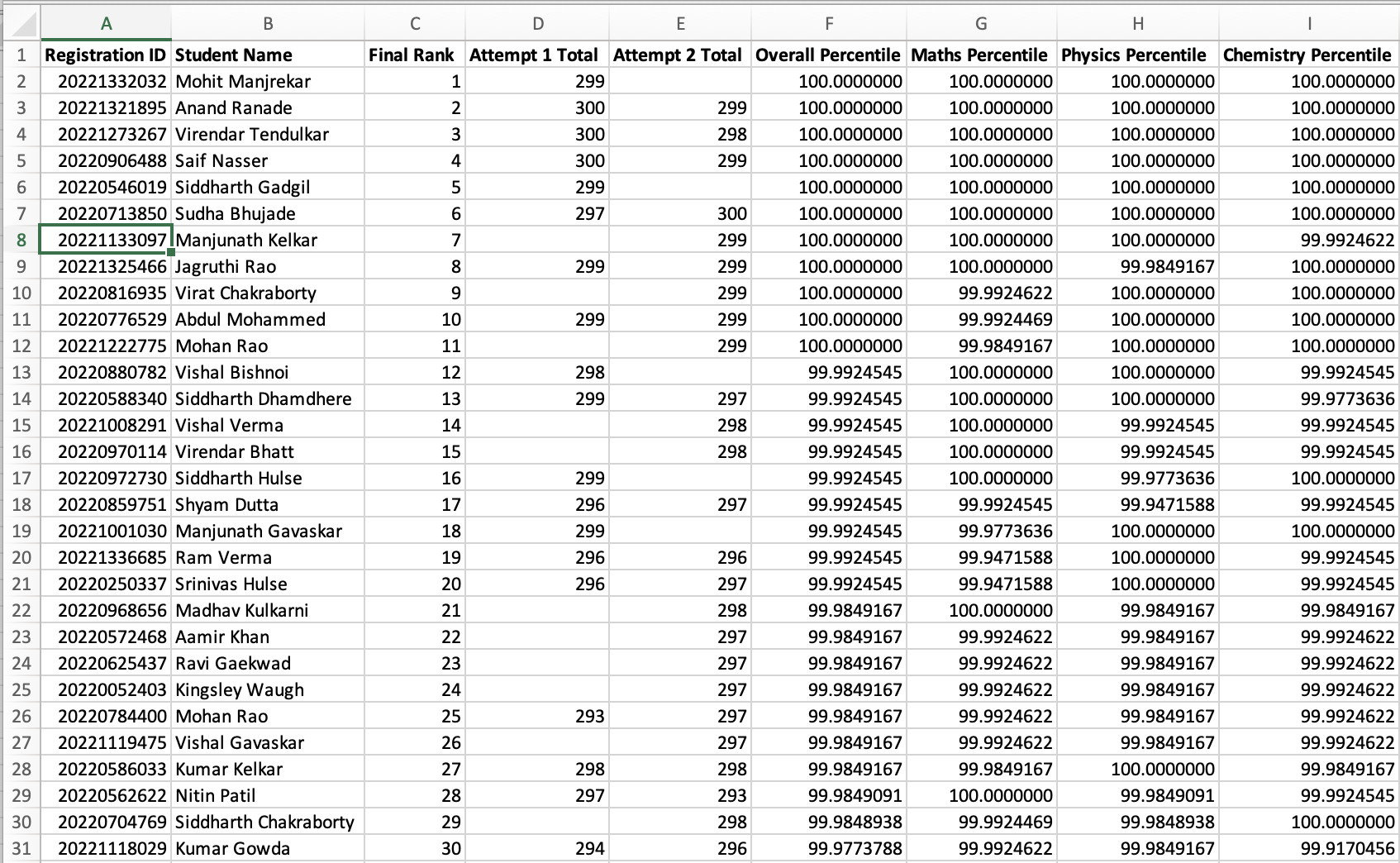
The marks sheet for each student is printed by this module which can subsequently be emailed to the registered email address of the student. A sample marksheet is displayed below.



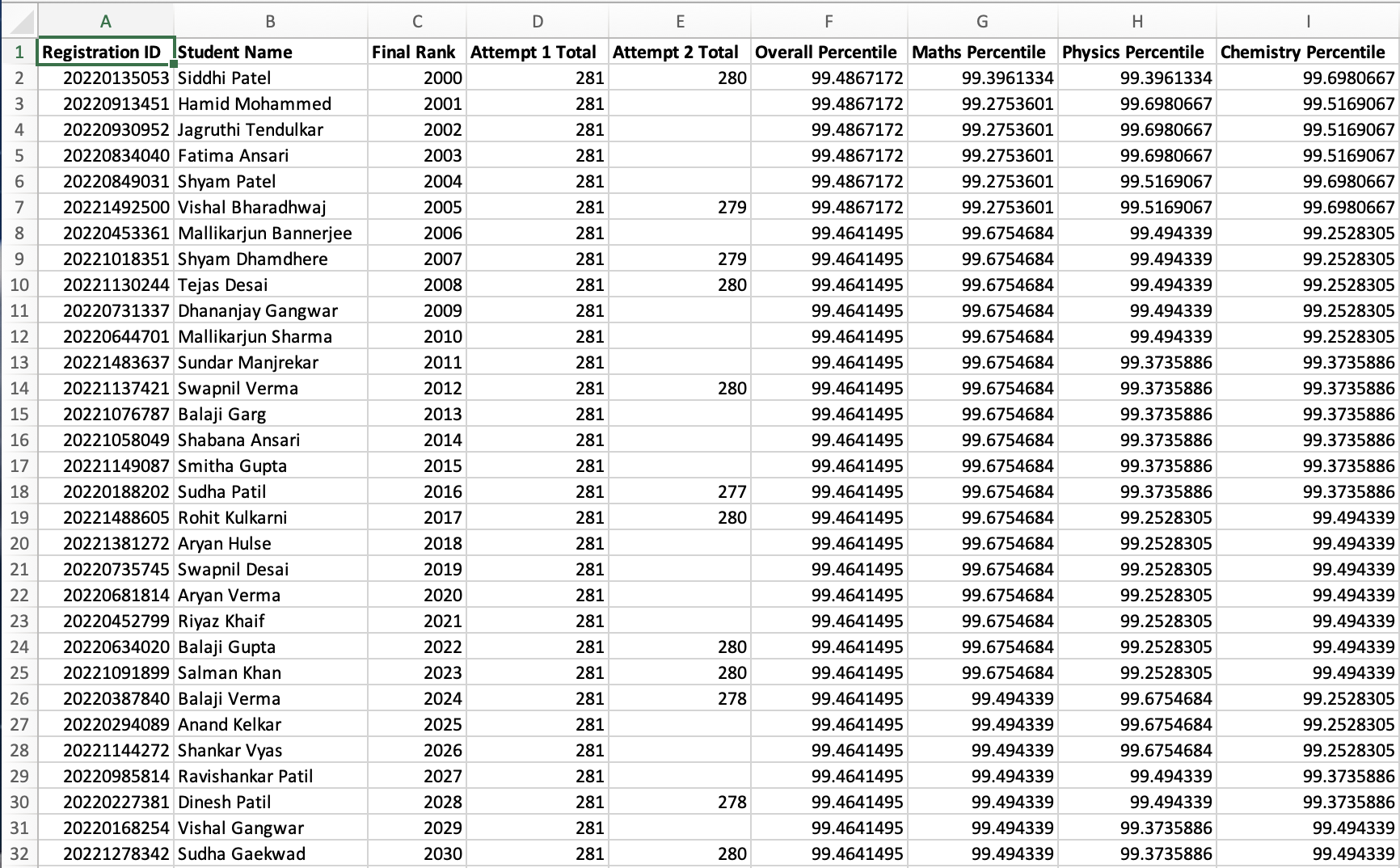
The query module can be used to obtain lot of interesting statistics about the test performance of the students. The output of top 30 students, students in the rank range 2000-2030 is given below.



Top 30 students (generated as a CSV)



Students in the rank range 2000 – 2030 (generated as a CSV)



# Conclusions

## APPLICATIONS AND ADVANTAGES OF THIS PROJECT

* Ranking of candidates in various competitive exams to increase speed of publishing results and minimize errors
* Customisable Tie breaker algorithms to meet the needs of each competitive exam
* Generate data at real world scale to enable testing and validation of algorithms such as Tie breaker
* Analytics to improve the quality of the exam

## FUTURE SCOPE

While this project is designed for the JEE mains exam, it can easily be extended to any competitive exam by configuring the exam mode, ranking algorithm etc. The following enhancements will further enhance the project:

* Form for registering in the exam
* Notification module to inform students by email, SMS etc.
* Mock exam simulator using random questions from the past exam to enable candidates get accustomed to the exam pattern
* College/ Course allotment based on candidate preference and score
* Inclusion of Date of Birth a last-resort tie breaking criteria in the event of a tie in Total, Maths, Physics and Chemistry Percentiles.
* Advanced analytics for fraud detection

# Bibliography

|  |  |
| --- | --- |
| [1] | Python Wiki, “Python Home Page,” [Online]. Available: https://wiki.python.org. |
| [2] | “Python Wiki,” [Online]. Available: https://en.wikipedia.org/wiki/Python\_(programming\_language). |
| [3] | “Python2 vs Python3,” [Online]. Available: https://www.guru99.com/python-2-vs-python-3.html. |
| [4] | Python Docs, “Python Licensing,” [Online]. Available: https://docs.python.org/3/license.html. |
| [5] | S. Arora, Computer Science with Python for Class XII, Dhanpat Rai and sons, 2021. |
| [6] | NCERT, Computer Science for class XII, NCERT, 2020. |
| [7] | Scikit, “Scikit Learn,” [Online]. Available: https://scikit-learn.org/. |
| [8] | NumPy, “NumPy,” [Online]. Available: https://numpy.org/. |
| [9] | PyQt5, “PyQt5,” [Online]. Available: https://pypi.org/project/PyQt5/. |
| [10] | WxPython, “WxPython,” [Online]. Available: https://www.wxpython.org/. |
| [11] | “MySQL Developer Guide,” [Online]. Available: https://dev.mysql.com/doc/. |