NEHRU SMARAKA VIDYALAYA
BANGALORE

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

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

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

2.PROJECT OVERVIEW

Objective

This project targets the leading educational organizations in India which conduct competitive exams on national scale, modelled closely on the 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 batch that 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

3. 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).

4.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.

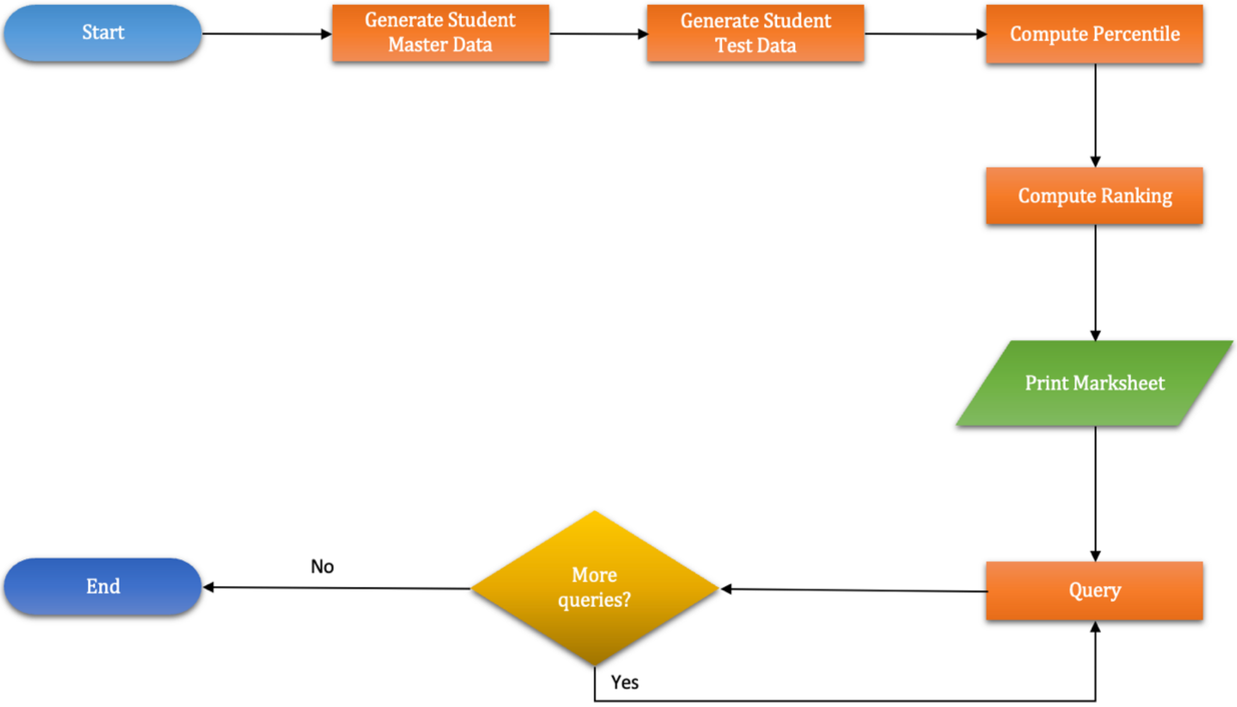
- **Compute Percentile and Rank:** This module uses the MySQL commands to compute the percentile and rank for all the students. Percentile is computed for each batch in the session separately. The best percentile across sessions for each student is considered for the final rank computation. The unique 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.

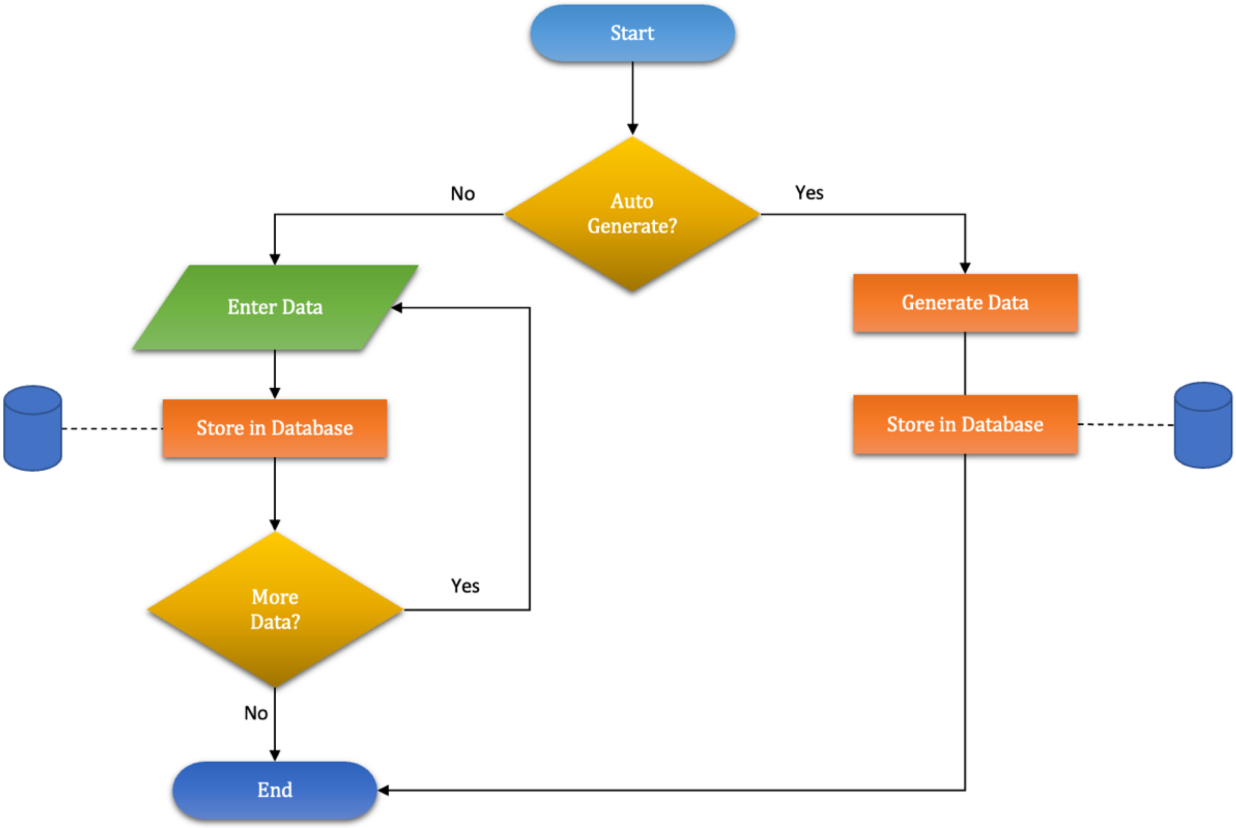
- **Query Module:** I have provided basic queries to obtain the list of students based on range of rank, marks in individual subjects etc. This module can be enhanced for advanced analytics and fraud detection in future.

Flow Chart

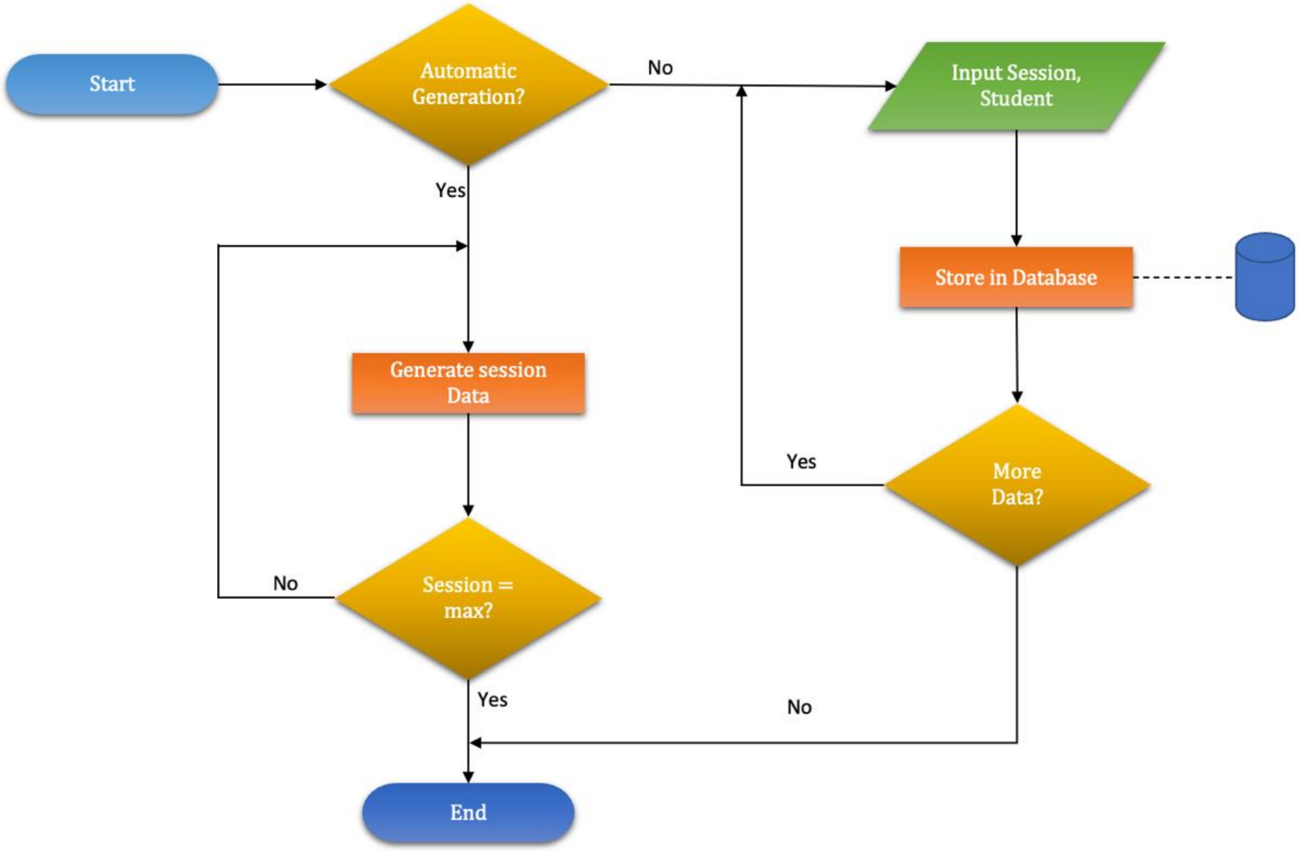
High Level Flow



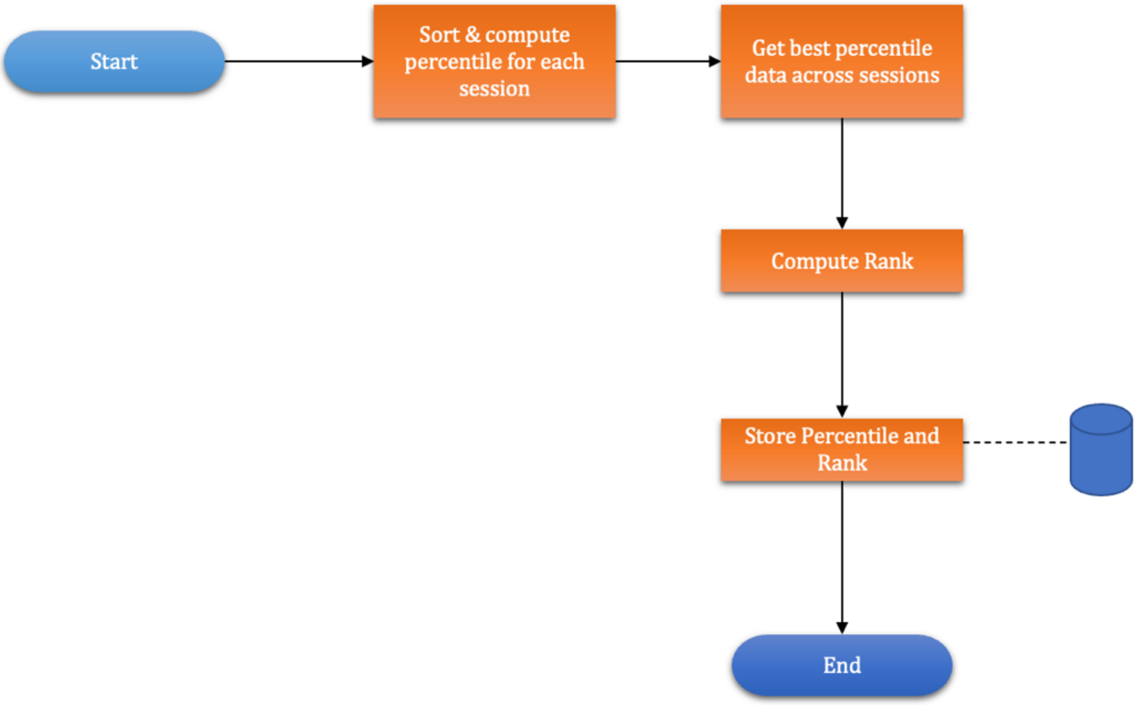
Generate Master Data



Generate Test Session Data



Generate Percentile and Rank Data


Start
Sort & compute percentile for each session
Get best percentile data across sessions
Compute Rank
Store Percentile and Rank
End

5.SOURCE CODE – COMPLETE

The Main Program

```
'''
This Program generates candidates for JEE MAINS exam
Registration Prefix will be used for all Registration ID's, for example 2022
Number of students for which scores ranks computed is to be inputed
Names, emails and mobile numbers will be auto-generated so will Registration ID's
This needs the file names.yaml to be in same folder as this program.
'''

import random
import sys
import yaml
import pandas as pd
import numpy as np
import mysql.connector as msql
import getpass

# names YAML file containing FirstName, LastName combinations
# n number of records
# prefix used in registration number
def gen_student_data(names, n, prefix):
    with open(names, "r") as inp:
        name_dict = yaml.safe_load(inp)
    name_sets = name_dict.keys()

    name_set_lengths = {}
    total = 0

    for i in range(1, len(name_sets) + 1):
        k = "Set" + str(i)
        first_names_len = len(name_dict[k]['FirstName'])
        last_names_len = len(name_dict[k]['LastName'])
        v = first_names_len * last_names_len
        name_set_lengths[k] = v
        total += v

    sets = []
    probs = []

    for k, v in name_set_lengths.items():
        sets.append(k)
        probs.append(v / total)

    email_ids = set()
    mobile_list = set()

    password = getpass.getpass('Enter Password for MySQL: ')
    conect = msql.connect(host = 'localhost', user = 'root', passwd = password)
    cursor = conect.cursor()
    cursor.execute('DROP DATABASE IF EXISTS JEE_Mains')
    cursor.execute('CREATE DATABASE JEE_Mains;')
    cursor.execute('USE JEE_Mains;')
    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);')

    for i in range(n):
        key = prefix + str(i).zfill(7)
        name_set = np.random.choice(sets, p = probs)
        first_name = random.choice(name_dict[name_set]['FirstName'])
        last_name = random.choice(name_dict[name_set]['LastName'])
        email_id = get_email_id(first_name, last_name, email_ids)
        contact_no = get_mobile_no(mobile_list)
        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))
        conect.commit()

    cursor.execute('SELECT * FROM Student_Master')
    a = cursor.fetchall()
    conect.close()

def get_email_id(first_name, last_name, email_ids):
    email_id = first_name + '.' + last_name
    ctr = 1

    while (email_id  in email_ids):
        email_id = first_name + '.' + last_name + '_' + str(ctr)
        ctr += 1
    email_ids.add(email_id)
    email_id += '@nta.com'
    return email_id

def get_mobile_no(mobile_list):
    n = 9000000000 + random.randint(1, 999999999)
    while n in mobile_list:
        n = 9000000000 + random.randint(1, 999999999)
    mobile_list.add(n)
    return n


#__main__
names = "names.yaml"
n = int(input('Enter the number of students (< 10 million) to generate percentile and rank: '))
prefix = input('Enter year or any 4 character prefix of your choice: ')
gen_student_data(names, n, prefix)

```

Generate Test Data

```python
'''
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.
'''

import numpy as np
import random
import mysql.connector as msql
import getpass

def get_student_marks():
    password = getpass.getpass('Enter Password for MySQL: ')
    conect = msql.connect(host = 'localhost', user = 'root', passwd = password, database = 'JEE_Mains')
    cursor = conect.cursor()
    cursor.execute("SELECT Reg_ID FROM Student_Master;")
    lis = cursor.fetchall()

    for i in range(8):
        cursor.execute('DROP TABLE IF EXISTS Session%s' %(str(i + 1)))
        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)))
    conect.commit()

    bands = []
    probs = []
    band1 = (-20, -10, 0.1)
    band2 = (-10, 0, 0.1)
    band3 = (0, 10, 0.2)
    band4 = (10, 20, 0.2)
    band5 = (20, 30, 0.1)
    band6 = (30, 40, 0.1)
    band7 = (40, 50, 0.1)
    band8 = (50, 60, 0.05)
    band9 = (60, 70, 0.02)
    band10 = (70, 77, 0.01)
    band11 = (77, 83, 0.01)
    band12 = (83, 88, 0.0055)
    band13 = (88, 92, 0.0025)
    band14 = (92, 95, 0.0013)
    band15 = (95, 97, 0.0005)
    band16 = (97, 99, 0.00018)
    band17 = (99, 100, 0.00002)

    for i in range(1, 18):
        a = 'band' + str(i)
        bands.append(a)
        a = eval(a)
        probs.append(a[2])

    for i in range (len(lis)):
        marks_band = eval(np.random.choice(bands, p = probs))
        math_marks = random.randint(marks_band[0], marks_band[1])
        phy_marks = random.randint(marks_band[0], marks_band[1])
        chem_marks = random.randint(marks_band[0], marks_band[1])
        attempts = np.random.choice([1,2,3], p = [0.25, 0.25, 0.5])

        if attempts == 1:
            num = random.choice([1,2,3,4])
            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)))
            conect.commit()
            cursor.execute('UPDATE Student_Master SET Attempt1 = %s WHERE Reg_ID = "%s";' %(str(num), str(lis[i][0])))
            conect.commit()

        elif attempts == 2:
            num = random.choice([5,6,7,8])
            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)))
            conect.commit()
            cursor.execute('UPDATE Student_Master SET Attempt2= %s WHERE Reg_ID = "%s";' %(str(num), str(lis[i][0])))
            conect.commit()

        elif attempts == 3:
            num = random.choice([1,2,3,4])
            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)))
            conect.commit()
            cursor.execute('UPDATE Student_Master SET Attempt1 = %s WHERE Reg_ID = "%s";' %(str(num), str(lis[i][0])))
            conect.commit()

            math_marks = random.randint(marks_band[0], marks_band[1])
            phy_marks = random.randint(marks_band[0], marks_band[1])
            chem_marks = random.randint(marks_band[0], marks_band[1])

            num = random.choice([5,6,7,8])
            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)))
            conect.commit()
            cursor.execute('UPDATE Student_Master SET Attempt2 = %s WHERE Reg_ID = "%s";' %(str(num), str(lis[i][0])))
            conect.commit()

    for i in range(1,9):
        cursor.execute('UPDATE Session%s SET Total = Math_Marks + Phy_Marks + Chem_Marks;' %(str(i)))
        conect.commit()

    conect.close()
#__main__
get_student_marks()
```

Compute Percentiles and Rank

```python
"""
Program acts on data already generated and updatees tables to set percentile and rank in a final created table.
Run after generate_test_data, no input except SQL Password
"""

import mysql.connector as msql
import math
import numpy as np
import random
import getpass
import csv

def compute_percentile_rank():
    password = getpass.getpass('Enter Password for MySQL: ')
    conect = msql.connect(user = 'root', host = 'localhost', passwd = password, database = 'JEE_Mains')
    cursor = conect.cursor()

    for i in range (1, 9):
        cursor.execute("DROP VIEW IF EXISTS Session%sP;" %(str(i)))
        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)));
        conect.commit()

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

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

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

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

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

    cursor.execute('DROP TABLE IF EXISTS Student_FinalScores;')
    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);")
    cursor.execute('SELECT Reg_ID, Attempt1, Attempt2, First_Name, Last_Name FROM Student_Master;')
    lis0 = cursor.fetchall()
    conect.commit()

    for i in range(len(lis0)):
        regid = lis0[i][0]
        attempt1 = lis0[i][1]
        attempt2 = lis0[i][2]
        S_name = str(lis0[i][3]) + ' ' + str(lis0[i][4])
        epsilon = 0.0000001
        tp1 = mp1 = cp1 = pp1 = tp2 = mp2 = pp2 = cp2 = 0.00
        t1 = m1 = p1 = c1 = t2 = m2 = c2 = p2 = 'NULL'
        if attempt1 != None:
            cursor.execute("SELECT * FROM Session%s WHERE Reg_ID = '%s';" %(str(attempt1), str(regid)))
            lis1 = cursor.fetchone()
            tp1 = lis1[8]
            mp1 = lis1[5]
            pp1 = lis1[6]
            cp1 = lis1[7]
            t1 = lis1[4]
            m1 = lis1[1]
            p1 = lis1[2]
            c1 = lis1[3]
        if attempt2 != None:
            cursor.execute("SELECT * FROM Session%s WHERE Reg_ID = '%s';" %(str(attempt2), str(regid)))
            lis2 = cursor.fetchone()
            tp2 = lis2[8]
            mp2 = lis2[5]
            pp2 = lis2[6]
            cp2 = lis2[7]
            t2 = lis2[4]
            m2 = lis2[1]
            p2 = lis2[2]
            c2 = lis2[3]

        tp = max(tp1, tp2)
        if abs(tp - tp2) > epsilon:
            mp, pp, cp, t = mp1, pp1, cp1, t1
        elif abs(tp - tp1) > epsilon:
            mp, pp, cp, t = mp2, pp2, cp2, t2
        else:
            mp = max(mp1, mp2)
            if abs(mp - mp2) > epsilon:
                pp, cp, t = pp1, cp1, t1
            elif abs(mp - mp1) > epsilon:
                pp, cp, t = pp2, cp2, t2
            else:
                pp = max(pp1, pp2)
                if abs(pp - pp2) > epsilon:
                    cp, t = cp1, t1
                elif abs(pp - pp1) > epsilon:
                    cp, t = cp2, t2
                else:
                    cp = max(cp1, cp2)
                    if abs(cp - cp2) > epsilon:
                        t = t1
                    elif abs(cp - cp1) > epsilon:
                        t = t2
                    else:
                        t = max(t1, t2)

        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)))
        conect.commit()

    cursor.execute('DROP VIEW IF EXISTS TempRank;')
    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);')
    cursor.execute('UPDATE Student_FinalScores T1 INNER JOIN TempRank T2 ON T1.Reg_ID = T2.Reg_ID SET T1.Final_Rank = T2.Final_Rank;')
    cursor.execute('DROP VIEW IF EXISTS TempRank;')
    conect.commit()

    conect.close()

#__main__
compute_percentile_rank()
```

Print Marksheets and Query Module

```
"""
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.
Run after generate_percentile_rank_data, needs SQL Password and other inputs from user.
"""

import mysql.connector as msql
import csv
import getpass

password = getpass.getpass('Enter Password for MySQL: ')
conect = msql.connect(user = 'root', host = 'localhost', passwd = password, database = 'JEE_Mains')
cursor = conect.cursor()

while True:
          print()
          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: ")

          if n == '0':
              print()
              print('Thank you!')
              print()
              conect.close()
              break

          elif n == '1':
              print()
              regid = input('Enter Registration ID of student: ')
              print()
              cursor.execute("SELECT * FROM Student_FinalScores WHERE Reg_ID = '%s'" %(regid))
              list3 = cursor.fetchall()

              if not list3:
                  print('InvalidRegIDError: Entered Registration ID of Nonexistant Student.')
                  continue

              else:
                  list3 = list3[0]
                  print('PRINTING MARKS CARD OF STUDENT', list3[1])
                  print('Registration ID: ', list3[0])
                  print()
                  print('ATTEMPT 1 MARKS:')
                  print('Maths: ', list3[3], '\nPhysics: ', list3[4], '\nChemistry: ', list3[5],     '\nTotal Score: ', list3[2])
                  print()
```

```
                print('ATTEMPT 2 MARKS:')
                print('Maths: ', list3[7], '\nPhysics: ', list3[8], '\nChemistry: ', list3[9],     '\nTotal Score: ', list3[6])
                print()
                print('SUBJECT-WISE PERCENTILES:')
                print('Maths percentile: ', list3[19])
                print('Physics percentile: ', list3[20])
                print('Chemistry percentile: ', list3[21])
                print()
                print('ONERALL PERCENTILE: ', list3[18])
                print('FINAL RANK: ', list3[22])

        elif n == '2':
           no = (input('Enter how many top rankers you want to obtain: '))
           print()
           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))
           list4 = cursor.fetchall()
           header = ('Registration ID', 'Student Name', 'Final Rank', 'Attempt 1 Total', 'Attempt 2 Total', 'Overall Percentile', 'Maths Percentile', 'Physics Percentile', 'Chemistry Percentile')

           with open ('Toppers.csv', 'w') as F:
               writer = csv.writer(F)
               writer.writerow(header)
               for row in list4:
                   writer.writerow(row)
           print('Successfully created CSV File Toppers.csv of Toppers.')
        elif n == '3':
          print()
          Lr = int(input('Enter lower rank limit: '))
          Hr = int(input('Enter upper rank limit: '))
          print()
          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)))
          list4 = cursor.fetchall()
          header = ('Registration ID', 'Student Name', 'Final Rank', 'Attempt 1 Total', 'Attempt 2 Total', 'Overall Percentile', 'Maths Percentile', 'Physics Percentile', 'Chemistry Percentile')
          with open ('Ranklist.csv', 'w') as F:
               writer = csv.writer(F)
               writer.writerow(header)
               for row in list4:
                   writer.writerow(row)
          print('Successfully created CSV File Ranklist.csv of Rankers between', Lr, 'and', Hr, '.')
        elif n == '4':
            print()
            Lm = int(input('Enter least value of marks for filter: '))
            Hm = int(input('Enter highest value of marks for filter: '))
            print()
```

```python
            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)))
            list4 = cursor.fetchall()
            header = ('Registration ID', 'Student Name', 'Final Rank', 'Attempt 1 Total', 'Attempt 2 Total', 'Overall Percentile', 'Maths Percentile', 'Physics Percentile', 'Chemistry Percentile')
            with open ('Markbandlist.csv', 'w') as F:
                writer = csv.writer(F)
                writer.writerow(header)
                for row in list4:
                    writer.writerow(row)
            print('Successfully created CSV File Markbandlist of Scorers between', Lm, 'and', Hm, '.')
        else:
            continue
```

Names Used to generate Student Master Data

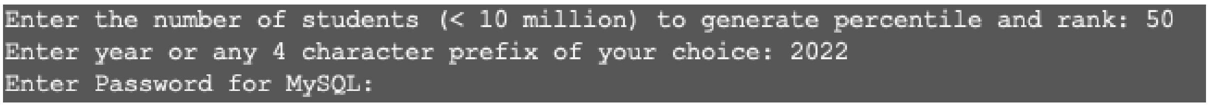
```
Set1:
    FirstName:
        - Tejas
        - Nikhil
        - Aryan
        - Siddharth
        - Balaji
        - Sundar
        - Ravi
        - Sudha
        - Shankar
        - Ravishankar
        - Suman
        - Nitin
        - Rajat
        - Swapnil
        - Vishal
        - Siddhi
        - 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
```

6.OUTPUT

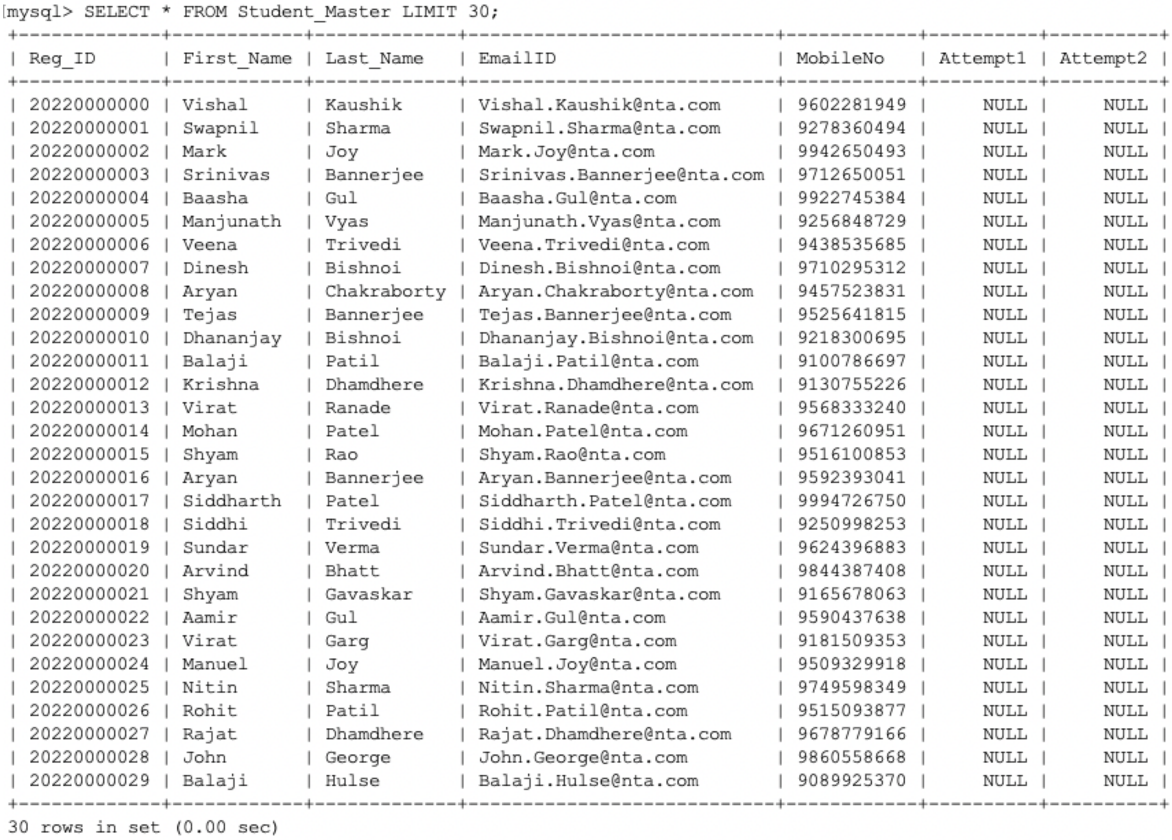
1) Generate Student Master Data

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.



```
Enter the number of students (< 10 million) to generate percentile and rank: 50
Enter year or any 4 character prefix of your choice: 2022
Enter Password for MySQL:
```

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



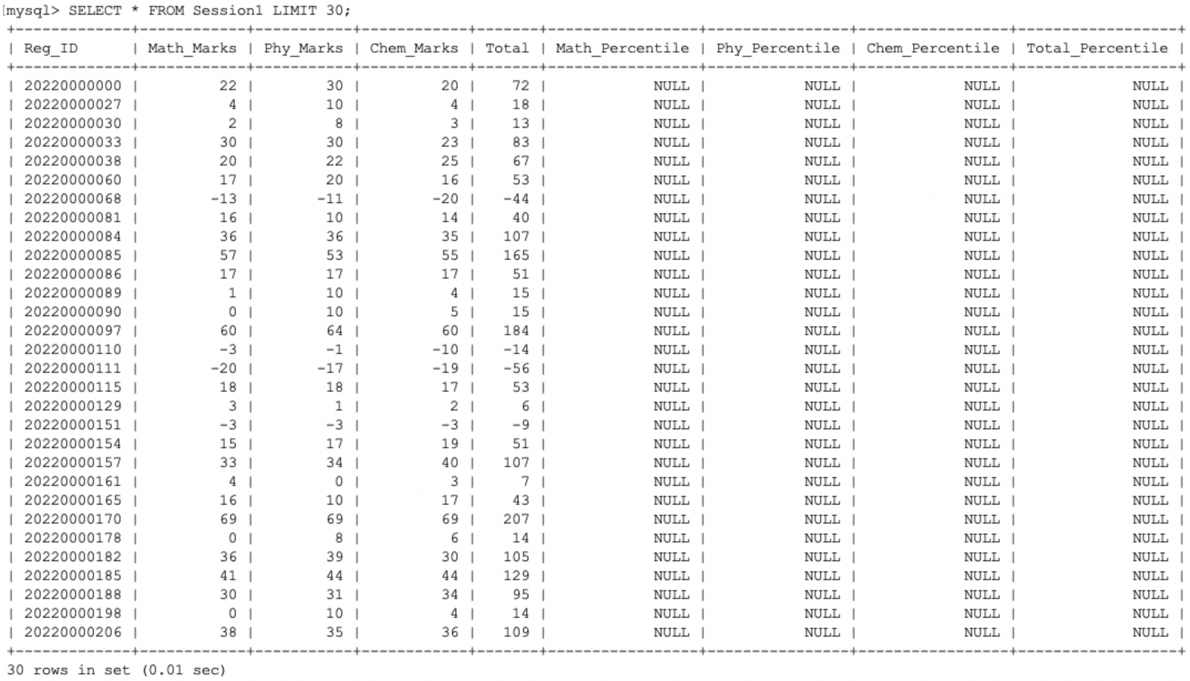
```
[mysql> SELECT * FROM Student_Master LIMIT 30;
+-------------+------------+-------------+----------------------------+------------+----------+----------+
| Reg_ID      | First_Name | Last_Name   | EmailID                    | MobileNo   | Attempt1 | Attempt2 |
+-------------+------------+-------------+----------------------------+------------+----------+----------+
| 20220000000 | Vishal     | Kaushik     | Vishal.Kaushik@nta.com     | 9602281949 |     NULL |     NULL |
| 20220000001 | Swapnil    | Sharma      | Swapnil.Sharma@nta.com     | 9278360494 |     NULL |     NULL |
| 20220000002 | Mark       | Joy         | Mark.Joy@nta.com           | 9942650493 |     NULL |     NULL |
| 20220000003 | Srinivas   | Bannerjee   | Srinivas.Bannerjee@nta.com | 9712650051 |     NULL |     NULL |
| 20220000004 | Baasha     | Gul         | Baasha.Gul@nta.com         | 9922745384 |     NULL |     NULL |
| 20220000005 | Manjunath  | Vyas        | Manjunath.Vyas@nta.com     | 9256848729 |     NULL |     NULL |
| 20220000006 | Veena      | Trivedi     | Veena.Trivedi@nta.com      | 9438535685 |     NULL |     NULL |
| 20220000007 | Dinesh     | Bishnoi     | Dinesh.Bishnoi@nta.com     | 9710295312 |     NULL |     NULL |
| 20220000008 | Aryan      | Chakraborty | Aryan.Chakraborty@nta.com  | 9457523831 |     NULL |     NULL |
| 20220000009 | Tejas      | Bannerjee   | Tejas.Bannerjee@nta.com    | 9525641815 |     NULL |     NULL |
| 20220000010 | Dhananjay  | Bishnoi     | Dhananjay.Bishnoi@nta.com  | 9218300695 |     NULL |     NULL |
| 20220000011 | Balaji     | Patil       | Balaji.Patil@nta.com       | 9100786697 |     NULL |     NULL |
| 20220000012 | Krishna    | Dhamdhere   | Krishna.Dhamdhere@nta.com  | 9130755226 |     NULL |     NULL |
| 20220000013 | Virat      | Ranade      | Virat.Ranade@nta.com       | 9568333240 |     NULL |     NULL |
| 20220000014 | Mohan      | Patel       | Mohan.Patel@nta.com        | 9671260951 |     NULL |     NULL |
| 20220000015 | Shyam      | Rao         | Shyam.Rao@nta.com          | 9516100853 |     NULL |     NULL |
| 20220000016 | Aryan      | Bannerjee   | Aryan.Bannerjee@nta.com    | 9592393041 |     NULL |     NULL |
| 20220000017 | Siddharth  | Patel       | Siddharth.Patel@nta.com    | 9994726750 |     NULL |     NULL |
| 20220000018 | Siddhi     | Trivedi     | Siddhi.Trivedi@nta.com     | 9250998253 |     NULL |     NULL |
| 20220000019 | Sundar     | Verma       | Sundar.Verma@nta.com       | 9624396883 |     NULL |     NULL |
| 20220000020 | Arvind     | Bhatt       | Arvind.Bhatt@nta.com       | 9844387408 |     NULL |     NULL |
| 20220000021 | Shyam      | Gavaskar    | Shyam.Gavaskar@nta.com     | 9165678063 |     NULL |     NULL |
| 20220000022 | Aamir      | Gul         | Aamir.Gul@nta.com          | 9590437638 |     NULL |     NULL |
| 20220000023 | Virat      | Garg        | Virat.Garg@nta.com         | 9181509353 |     NULL |     NULL |
| 20220000024 | Manuel     | Joy         | Manuel.Joy@nta.com         | 9509329918 |     NULL |     NULL |
| 20220000025 | Nitin      | Sharma      | Nitin.Sharma@nta.com       | 9749598349 |     NULL |     NULL |
| 20220000026 | Rohit      | Patil       | Rohit.Patil@nta.com        | 9515093877 |     NULL |     NULL |
| 20220000027 | Rajat      | Dhamdhere   | Rajat.Dhamdhere@nta.com    | 9678779166 |     NULL |     NULL |
| 20220000028 | John       | George      | John.George@nta.com        | 9860558668 |     NULL |     NULL |
| 20220000029 | Balaji     | Hulse       | Balaji.Hulse@nta.com       | 9089925370 |     NULL |     NULL |
+-------------+------------+-------------+----------------------------+------------+----------+----------+
30 rows in set (0.00 sec)
```

2) 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.

```
Enter Password for MySQL:
```

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



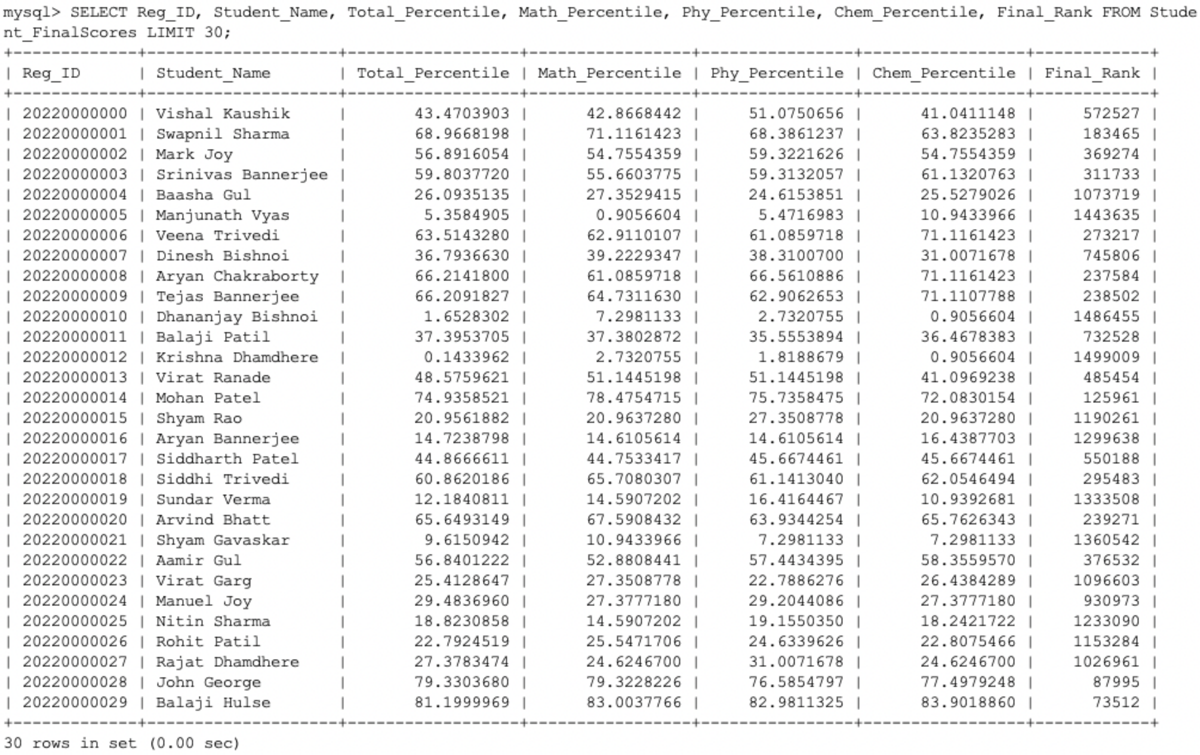
```
mysql> SELECT * FROM Session1 LIMIT 30;
+-------------+------------+-----------+------------+-------+-----------------+----------------+-----------------+------------------+
| Reg_ID      | Math_Marks | Phy_Marks | Chem_Marks | Total | Math_Percentile | Phy_Percentile | Chem_Percentile | Total_Percentile |
+-------------+------------+-----------+------------+-------+-----------------+----------------+-----------------+------------------+
| 20220000000 |         22 |        30 |         20 |    72 |            NULL |           NULL |            NULL |             NULL |
| 20220000027 |          4 |        10 |          4 |    18 |            NULL |           NULL |            NULL |             NULL |
| 20220000030 |          2 |         8 |          3 |    13 |            NULL |           NULL |            NULL |             NULL |
| 20220000033 |         30 |        30 |         23 |    83 |            NULL |           NULL |            NULL |             NULL |
| 20220000038 |         20 |        22 |         25 |    67 |            NULL |           NULL |            NULL |             NULL |
| 20220000060 |         17 |        20 |         16 |    53 |            NULL |           NULL |            NULL |             NULL |
| 20220000068 |        -13 |       -11 |        -20 |   -44 |            NULL |           NULL |            NULL |             NULL |
| 20220000081 |         16 |        10 |         14 |    40 |            NULL |           NULL |            NULL |             NULL |
| 20220000084 |         36 |        36 |         35 |   107 |            NULL |           NULL |            NULL |             NULL |
| 20220000085 |         57 |        53 |         55 |   165 |            NULL |           NULL |            NULL |             NULL |
| 20220000086 |         17 |        17 |         17 |    51 |            NULL |           NULL |            NULL |             NULL |
| 20220000089 |          1 |        10 |          4 |    15 |            NULL |           NULL |            NULL |             NULL |
| 20220000090 |          0 |        10 |          5 |    15 |            NULL |           NULL |            NULL |             NULL |
| 20220000097 |         60 |        64 |         60 |   184 |            NULL |           NULL |            NULL |             NULL |
| 20220000110 |         -3 |        -1 |        -10 |   -14 |            NULL |           NULL |            NULL |             NULL |
| 20220000111 |        -20 |       -17 |        -19 |   -56 |            NULL |           NULL |            NULL |             NULL |
| 20220000115 |         18 |        18 |         17 |    53 |            NULL |           NULL |            NULL |             NULL |
| 20220000129 |          3 |         1 |          2 |     6 |            NULL |           NULL |            NULL |             NULL |
| 20220000151 |         -3 |        -3 |         -3 |    -9 |            NULL |           NULL |            NULL |             NULL |
| 20220000154 |         15 |        17 |         19 |    51 |            NULL |           NULL |            NULL |             NULL |
| 20220000157 |         33 |        34 |         40 |   107 |            NULL |           NULL |            NULL |             NULL |
| 20220000161 |          4 |         0 |          3 |     7 |            NULL |           NULL |            NULL |             NULL |
| 20220000165 |         16 |        10 |         17 |    43 |            NULL |           NULL |            NULL |             NULL |
| 20220000170 |         69 |        69 |         69 |   207 |            NULL |           NULL |            NULL |             NULL |
| 20220000178 |          0 |         8 |          6 |    14 |            NULL |           NULL |            NULL |             NULL |
| 20220000182 |         36 |        39 |         30 |   105 |            NULL |           NULL |            NULL |             NULL |
| 20220000185 |         41 |        44 |         44 |   129 |            NULL |           NULL |            NULL |             NULL |
| 20220000188 |         30 |        31 |         34 |    95 |            NULL |           NULL |            NULL |             NULL |
| 20220000198 |          0 |        10 |          4 |    14 |            NULL |           NULL |            NULL |             NULL |
| 20220000206 |         38 |        35 |         36 |   109 |            NULL |           NULL |            NULL |             NULL |
+-------------+------------+-----------+------------+-------+-----------------+----------------+-----------------+------------------+
30 rows in set (0.01 sec)
```

3) 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.

```
Enter Password for MySQL:
```

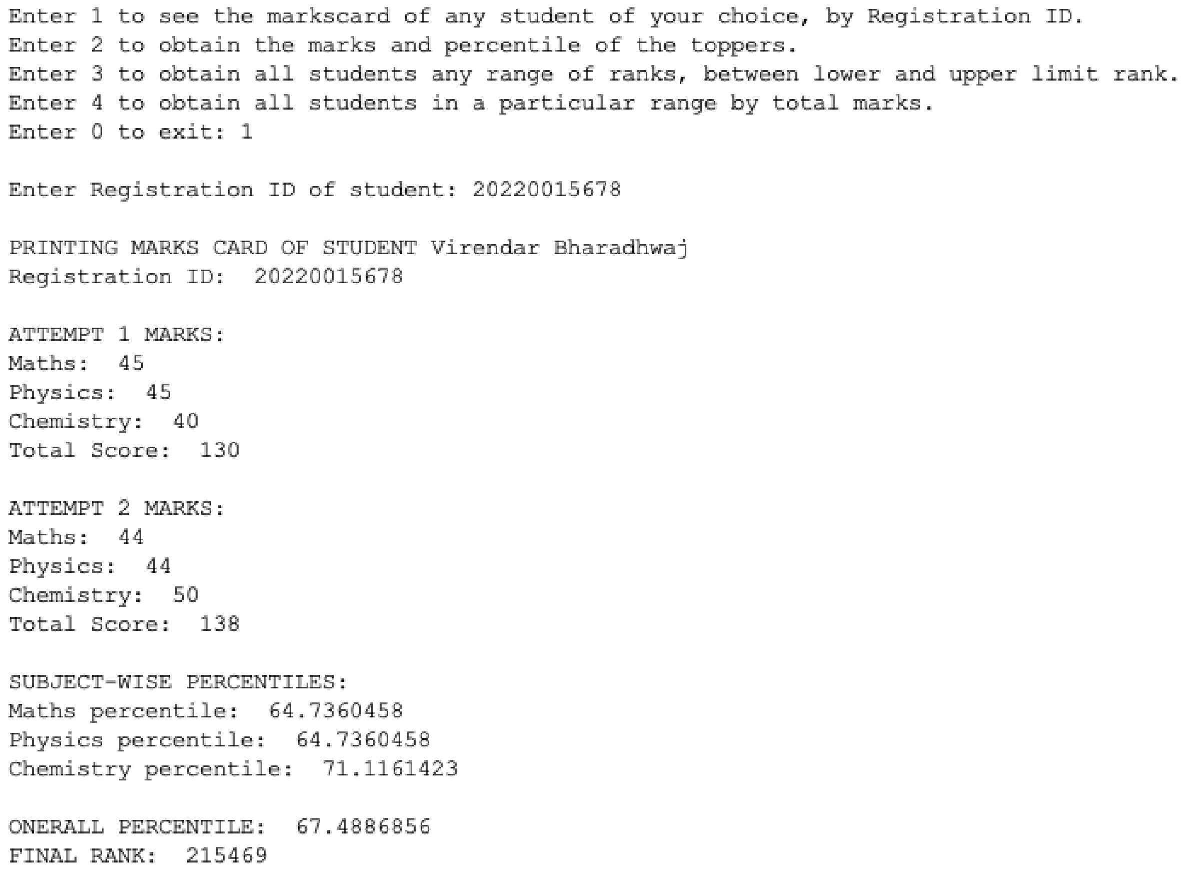
The percentile is computed for all the students and the best percentile for each student is considered for rank allotment. Sample rank list generated as a CSV file is given below.



```
mysql> SELECT Reg_ID, Student_Name, Total_Percentile, Math_Percentile, Phy_Percentile, Chem_Percentile, Final_Rank FROM Stude
nt_FinalScores LIMIT 30;
+-------------+--------------------+------------------+-----------------+----------------+-----------------+------------+
| Reg_ID      | Student_Name       | Total_Percentile | Math_Percentile | Phy_Percentile | Chem_Percentile | Final_Rank |
+-------------+--------------------+------------------+-----------------+----------------+-----------------+------------+
| 20220000000 | Vishal Kaushik     |       43.4703903 |      42.8668442 |     51.0750656 |      41.0411148 |     572527 |
| 20220000001 | Swapnil Sharma     |       68.9668198 |      71.1161423 |     68.3861237 |      63.8235283 |     183465 |
| 20220000002 | Mark Joy           |       56.8916054 |      54.7554359 |     59.3221626 |      54.7554359 |     369274 |
| 20220000003 | Srinivas Bannerjee |       59.8037720 |      55.6603775 |     59.3132057 |      61.1320763 |     311733 |
| 20220000004 | Baasha Gul         |       26.0935135 |      27.3529415 |     24.6153851 |      25.5279026 |    1073719 |
| 20220000005 | Manjunath Vyas     |        5.3584905 |       0.9056604 |      5.4716983 |      10.9433966 |    1443635 |
| 20220000006 | Veena Trivedi      |       63.5143280 |      62.9110107 |     61.0859718 |      71.1161423 |     273217 |
| 20220000007 | Dinesh Bishnoi     |       36.7936630 |      39.2229347 |     38.3100700 |      31.0071678 |     745806 |
| 20220000008 | Aryan Chakraborty  |       66.2141800 |      61.0859718 |     66.5610886 |      71.1161423 |     237584 |
| 20220000009 | Tejas Bannerjee    |       66.2091827 |      64.7311630 |     62.9062653 |      71.1107788 |     238502 |
| 20220000010 | Dhananjay Bishnoi  |        1.6528302 |       7.2981133 |      2.7320755 |       0.9056604 |    1486455 |
| 20220000011 | Balaji Patil       |       37.3953705 |      37.3802872 |     35.5553894 |      36.4678383 |     732528 |
| 20220000012 | Krishna Dhamdhere  |        0.1433962 |       2.7320755 |      1.8188679 |       0.9056604 |    1499009 |
| 20220000013 | Virat Ranade       |       48.5759621 |      51.1445198 |     51.1445198 |      41.0969238 |     485454 |
| 20220000014 | Mohan Patel        |       74.9358521 |      78.4754715 |     75.7358475 |      72.0830154 |     125961 |
| 20220000015 | Shyam Rao          |       20.9561882 |      20.9637280 |     27.3508778 |      20.9637280 |    1190261 |
| 20220000016 | Aryan Bannerjee    |       14.7238798 |      14.6105614 |     14.6105614 |      16.4387703 |    1299638 |
| 20220000017 | Siddharth Patel    |       44.8666611 |      44.7533417 |     45.6674461 |      45.6674461 |     550188 |
| 20220000018 | Siddhi Trivedi     |       60.8620186 |      65.7080307 |     61.1413040 |      62.0546494 |     295483 |
| 20220000019 | Sundar Verma       |       12.1840811 |      14.5907202 |     16.4164467 |      10.9392681 |    1333508 |
| 20220000020 | Arvind Bhatt       |       65.6493149 |      67.5908432 |     63.9344254 |      65.7626343 |     239271 |
| 20220000021 | Shyam Gavaskar     |        9.6150942 |      10.9433966 |      7.2981133 |       7.2981133 |    1360542 |
| 20220000022 | Aamir Gul          |       56.8401222 |      52.8808441 |     57.4434395 |      58.3559570 |     376532 |
| 20220000023 | Virat Garg         |       25.4128647 |      27.3508778 |     22.7886276 |      26.4384289 |    1096603 |
| 20220000024 | Manuel Joy         |       29.4836960 |      27.3777180 |     29.2044086 |      27.3777180 |     930973 |
| 20220000025 | Nitin Sharma       |       18.8230858 |      14.5907202 |     19.1550350 |      18.2421722 |    1233090 |
| 20220000026 | Rohit Patil        |       22.7924519 |      25.5471706 |     24.6339626 |      22.8075466 |    1153284 |
| 20220000027 | Rajat Dhamdhere    |       27.3783474 |      24.6246700 |     31.0071678 |      24.6246700 |    1026961 |
| 20220000028 | John George        |       79.3303680 |      79.3228226 |     76.5854797 |      77.4979248 |      87995 |
| 20220000029 | Balaji Hulse       |       81.1999969 |      83.0037766 |     82.9811325 |      83.9018860 |      73512 |
+-------------+--------------------+------------------+-----------------+----------------+-----------------+------------+
30 rows in set (0.00 sec)
```

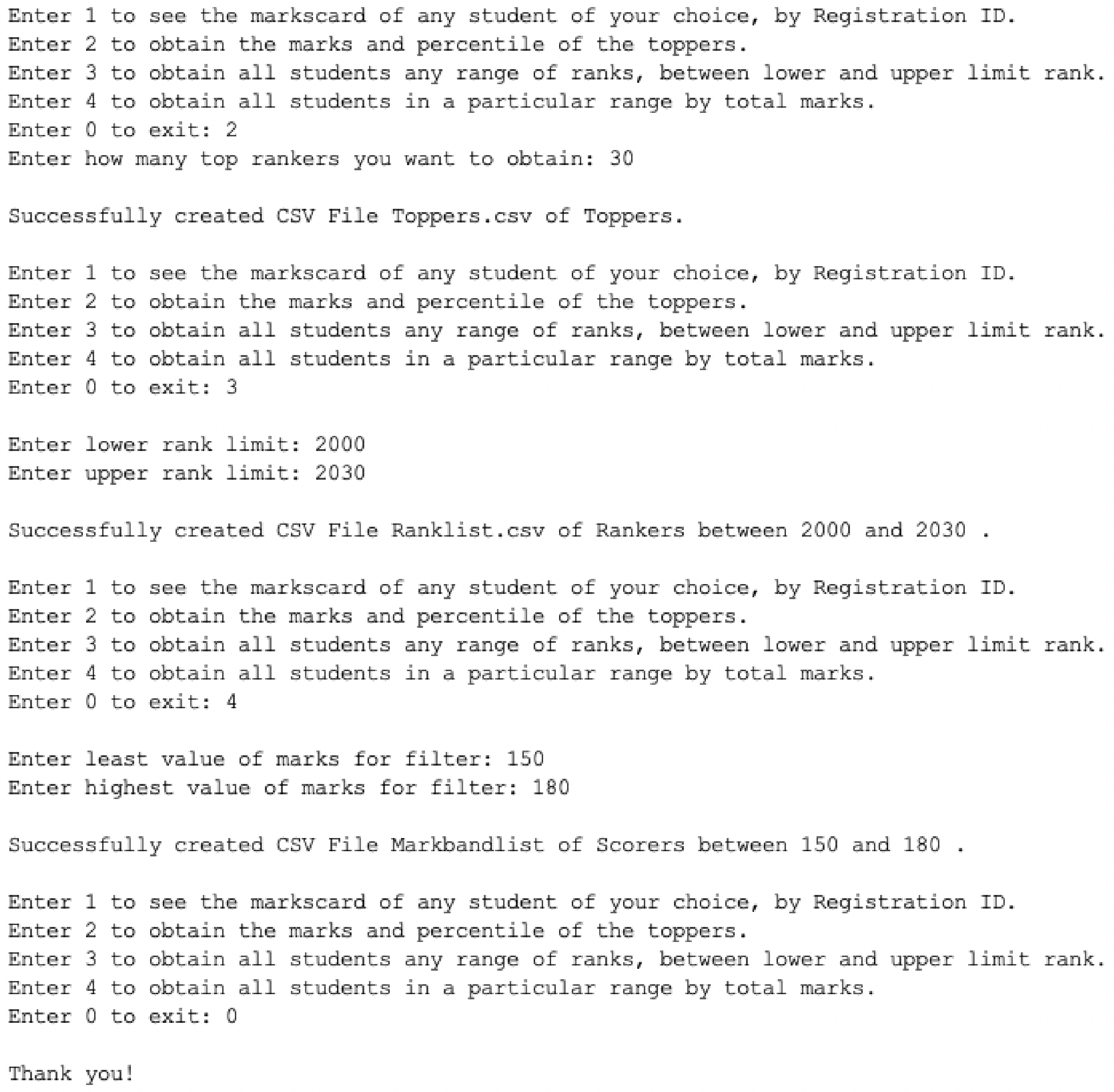
4) Marks Sheet generation and Query module

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.



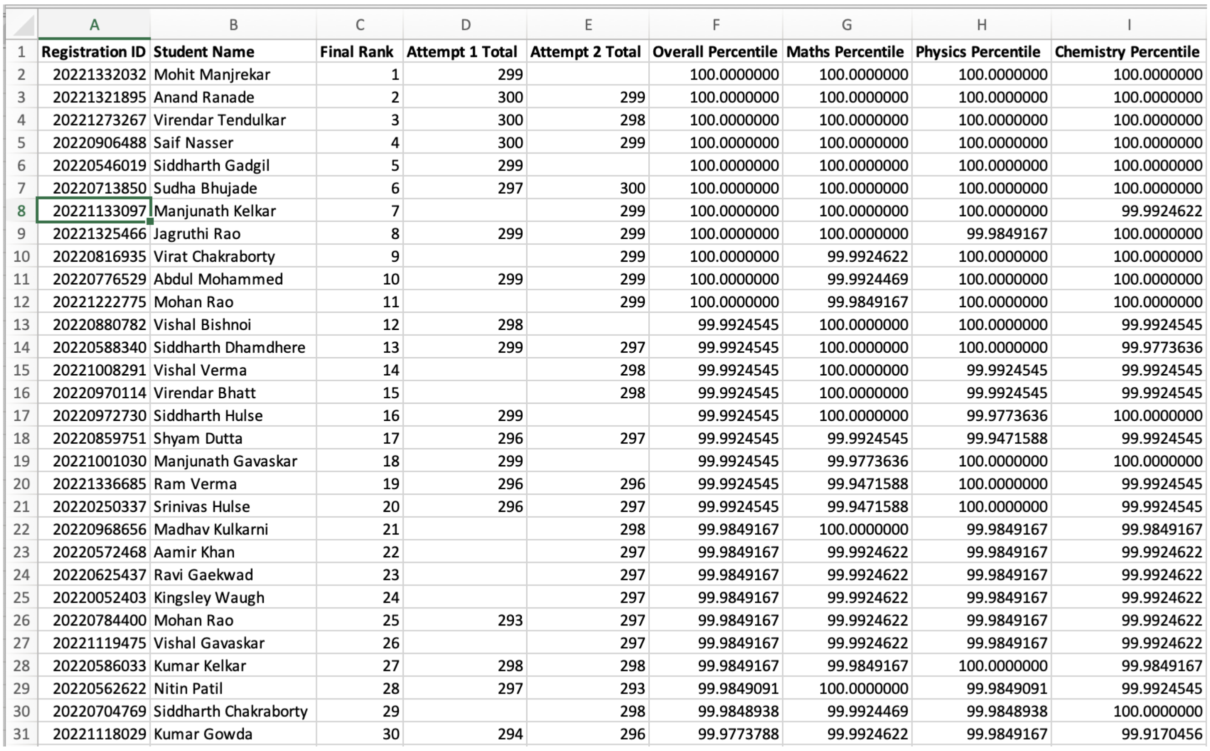
```
Enter 1 to see the markscard of any student of your choice, by Registration ID.
Enter 2 to obtain the marks and percentile of the toppers.
Enter 3 to obtain all students any range of ranks, between lower and upper limit rank.
Enter 4 to obtain all students in a particular range by total marks.
Enter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

Enter Registration ID of student: 20220015678

PRINTING MARKS CARD OF STUDENT Virendar Bharadhwaj
Registration ID:  20220015678

ATTEMPT 1 MARKS:
Maths:  45
Physics:  45
Chemistry:  40
Total Score:  130

ATTEMPT 2 MARKS:
Maths:  44
Physics:  44
Chemistry:  50
Total Score:  138

SUBJECT-WISE PERCENTILES:
Maths percentile:  64.7360458
Physics percentile:  64.7360458
Chemistry percentile:  71.1161423

ONERALL PERCENTILE:  67.4886856
FINAL RANK:  215469
```

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.



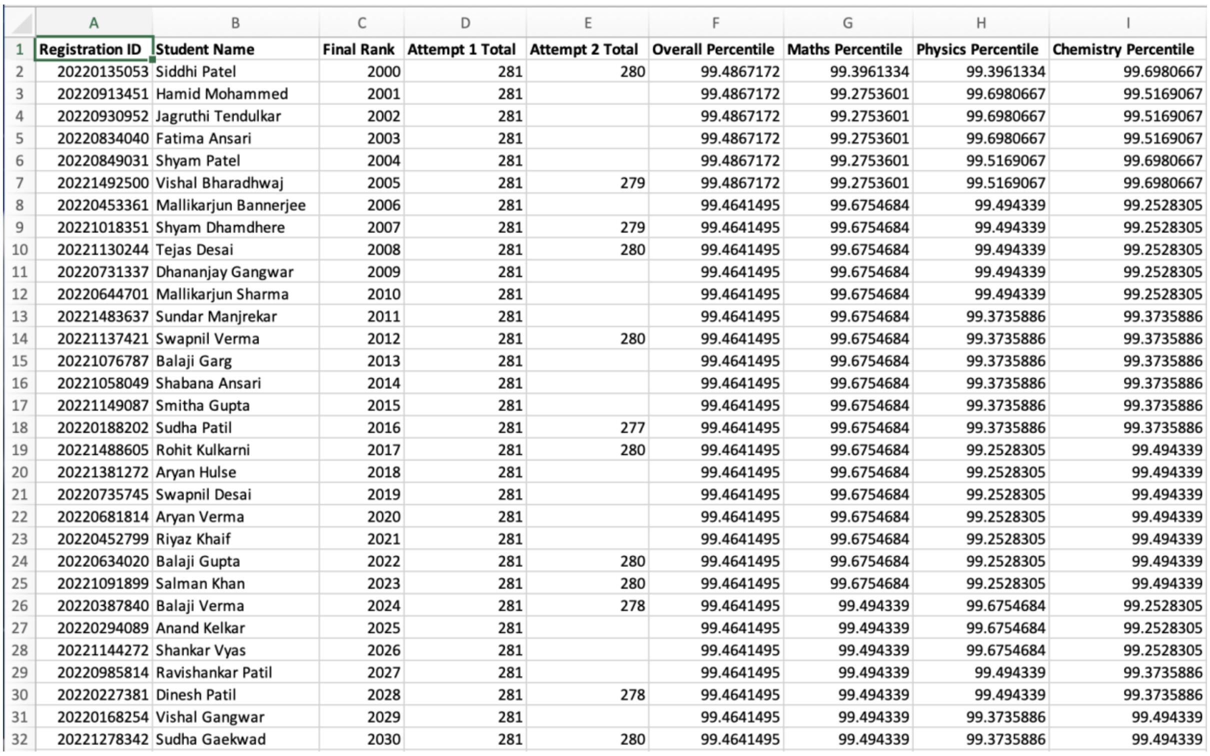
```
Enter 1 to see the markscard of any student of your choice, by Registration ID.
Enter 2 to obtain the marks and percentile of the toppers.
Enter 3 to obtain all students any range of ranks, between lower and upper limit rank.
Enter 4 to obtain all students in a particular range by total marks.
Enter 0 to exit: 2
Enter how many top rankers you want to obtain: 30

Successfully created CSV File Toppers.csv of Toppers.

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

Enter lower rank limit: 2000
Enter upper rank limit: 2030

Successfully created CSV File Ranklist.csv of Rankers between 2000 and 2030 .

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

Enter least value of marks for filter: 150
Enter highest value of marks for filter: 180

Successfully created CSV File Markbandlist of Scorers between 150 and 180 .

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

Thank you!
```

Top 30 students (generated as a CSV)



	A	B	C	D	E	F	G	H	I
1	Registration ID	Student Name	Final Rank	Attempt 1 Total	Attempt 2 Total	Overall Percentile	Maths Percentile	Physics Percentile	Chemistry Percentile
2	20221332032	Mohit Manjrekar	1	299		100.0000000	100.0000000	100.0000000	100.0000000
3	20221321895	Anand Ranade	2	300	299	100.0000000	100.0000000	100.0000000	100.0000000
4	20221273267	Virendar Tendulkar	3	300	298	100.0000000	100.0000000	100.0000000	100.0000000
5	20220906488	Saif Nasser	4	300	299	100.0000000	100.0000000	100.0000000	100.0000000
6	20220546019	Siddharth Gadgil	5	299		100.0000000	100.0000000	100.0000000	100.0000000
7	20220713850	Sudha Bhujade	6	297	300	100.0000000	100.0000000	100.0000000	100.0000000
8	20221133097	Manjunath Kelkar	7		299	100.0000000	100.0000000	100.0000000	99.9924622
9	20221325466	Jagruthi Rao	8	299	299	100.0000000	100.0000000	99.9849167	100.0000000
10	20220816935	Virat Chakraborty	9		299	100.0000000	99.9924622	100.0000000	100.0000000
11	20220776529	Abdul Mohammed	10	299	299	100.0000000	99.9924469	100.0000000	100.0000000
12	20221222775	Mohan Rao	11		299	100.0000000	99.9849167	100.0000000	100.0000000
13	20220880782	Vishal Bishnoi	12	298		99.9924545	100.0000000	100.0000000	99.9924545
14	20220588340	Siddharth Dhamdhere	13	299	297	99.9924545	100.0000000	100.0000000	99.9773636
15	20221008291	Vishal Verma	14		298	99.9924545	100.0000000	99.9924545	99.9924545
16	20220970114	Virendar Bhatt	15		298	99.9924545	100.0000000	99.9924545	99.9924545
17	20220972730	Siddharth Hulse	16	299		99.9924545	100.0000000	99.9773636	100.0000000
18	20220859751	Shyam Dutta	17	296	297	99.9924545	99.9924545	99.9471588	99.9924545
19	20221001030	Manjunath Gavaskar	18	299		99.9924545	99.9773636	100.0000000	100.0000000
20	20221336685	Ram Verma	19	296	296	99.9924545	99.9471588	100.0000000	99.9924545
21	20220250337	Srinivas Hulse	20	296	297	99.9924545	99.9471588	100.0000000	99.9924545
22	20220968656	Madhav Kulkarni	21		298	99.9849167	100.0000000	99.9849167	99.9849167
23	20220572468	Aamir Khan	22		297	99.9849167	99.9924622	99.9849167	99.9924622
24	20220625437	Ravi Gaekwad	23		297	99.9849167	99.9924622	99.9849167	99.9924622
25	20220052403	Kingsley Waugh	24		297	99.9849167	99.9924622	99.9849167	99.9924622
26	20220784400	Mohan Rao	25	293	297	99.9849167	99.9924622	99.9849167	99.9924622
27	20221119475	Vishal Gavaskar	26		297	99.9849167	99.9924622	99.9849167	99.9924622
28	20220586033	Kumar Kelkar	27	298	298	99.9849167	99.9849167	100.0000000	99.9849167
29	20220562622	Nitin Patil	28	297	293	99.9849091	100.0000000	99.9849091	99.9924545
30	20220704769	Siddharth Chakraborty	29		298	99.9848938	99.9924469	99.9848938	100.0000000
31	20221118029	Kumar Gowda	30	294	296	99.9773788	99.9924622	99.9849167	99.9170456

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



	A	B	C	D	E	F	G	H	I
1	Registration ID	Student Name	Final Rank	Attempt 1 Total	Attempt 2 Total	Overall Percentile	Maths Percentile	Physics Percentile	Chemistry Percentile
2	20220135053	Siddhi Patel	2000	281	280	99.4867172	99.3961334	99.3961334	99.6980667
3	20220913451	Hamid Mohammed	2001	281		99.4867172	99.2753601	99.6980667	99.5169067
4	20220930952	Jagruthi Tendulkar	2002	281		99.4867172	99.2753601	99.6980667	99.5169067
5	20220834040	Fatima Ansari	2003	281		99.4867172	99.2753601	99.6980667	99.5169067
6	20220849031	Shyam Patel	2004	281		99.4867172	99.2753601	99.5169067	99.6980667
7	20221492500	Vishal Bharadhwaj	2005	281	279	99.4867172	99.2753601	99.5169067	99.6980667
8	20220453361	Mallikarjun Bannerjee	2006	281		99.4641495	99.6754684	99.494339	99.2528305
9	20221018351	Shyam Dhamdhere	2007	281	279	99.4641495	99.6754684	99.494339	99.2528305
10	20221130244	Tejas Desai	2008	281	280	99.4641495	99.6754684	99.494339	99.2528305
11	20220731337	Dhananjay Gangwar	2009	281		99.4641495	99.6754684	99.494339	99.2528305
12	20220644701	Mallikarjun Sharma	2010	281		99.4641495	99.6754684	99.494339	99.2528305
13	20221483637	Sundar Manjrekar	2011	281		99.4641495	99.6754684	99.3735886	99.3735886
14	20221137421	Swapnil Verma	2012	281	280	99.4641495	99.6754684	99.3735886	99.3735886
15	20221076787	Balaji Garg	2013	281		99.4641495	99.6754684	99.3735886	99.3735886
16	20221058049	Shabana Ansari	2014	281		99.4641495	99.6754684	99.3735886	99.3735886
17	20221149087	Smitha Gupta	2015	281		99.4641495	99.6754684	99.3735886	99.3735886
18	20220188202	Sudha Patil	2016	281	277	99.4641495	99.6754684	99.3735886	99.3735886
19	20221488605	Rohit Kulkarni	2017	281	280	99.4641495	99.6754684	99.2528305	99.494339
20	20221381272	Aryan Hulse	2018	281		99.4641495	99.6754684	99.2528305	99.494339
21	20220735745	Swapnil Desai	2019	281		99.4641495	99.6754684	99.2528305	99.494339
22	20220681814	Aryan Verma	2020	281		99.4641495	99.6754684	99.2528305	99.494339
23	20220452799	Riyaz Khaif	2021	281		99.4641495	99.6754684	99.2528305	99.494339
24	20220634020	Balaji Gupta	2022	281	280	99.4641495	99.6754684	99.2528305	99.494339
25	20221091899	Salman Khan	2023	281	280	99.4641495	99.6754684	99.2528305	99.494339
26	20220387840	Balaji Verma	2024	281	278	99.4641495	99.494339	99.6754684	99.2528305
27	20220294089	Anand Kelkar	2025	281		99.4641495	99.494339	99.6754684	99.2528305
28	20221144272	Shankar Vyas	2026	281		99.4641495	99.494339	99.6754684	99.2528305
29	20220985814	Ravishankar Patil	2027	281		99.4641495	99.494339	99.494339	99.3735886
30	20220227381	Dinesh Patil	2028	281	278	99.4641495	99.494339	99.494339	99.3735886
31	20220168254	Vishal Gangwar	2029	281		99.4641495	99.494339	99.3735886	99.494339
32	20221278342	Sudha Gaekwad	2030	281	280	99.4641495	99.494339	99.3735886	99.494339

7. CONCLUSIONS

APPLICATIONS AND ADVANTAGES OF THIS PROJECT

- Ranking of candidates in various competitive exams to increase the 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 closely modelled on 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/.