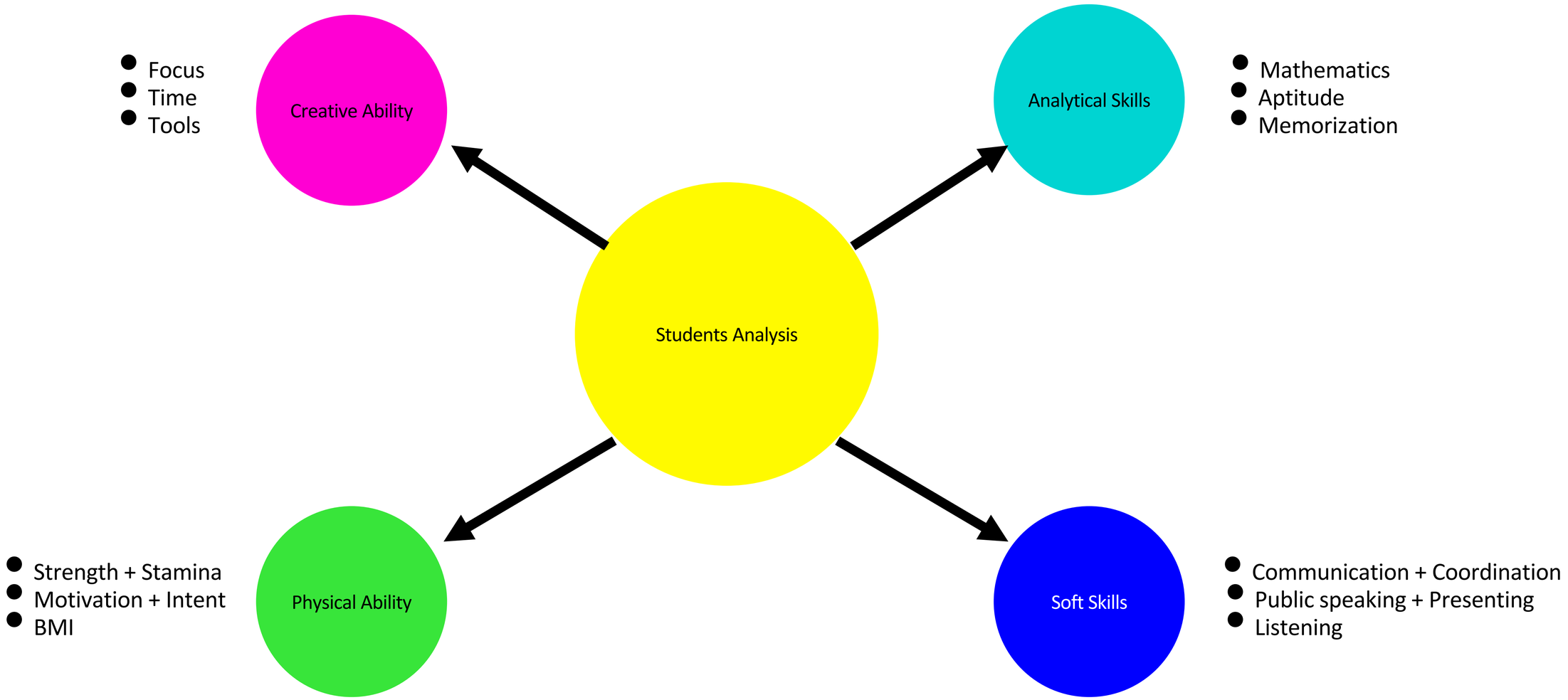
# EDUCATION SYSTEM IN INDIA!

ENABLING OVERALL GROWTH OF STUDENTS FROM AN EARLY AGE.

# ELIMINATING CONSERVATIVE GROUPISM

FOCUSING ON OVERALL GROWTH RATHER THAN SINGLE DOMAIN FOCUS!

# Evaluation Metrics/ Student Attributes



# MODEL DESIGN

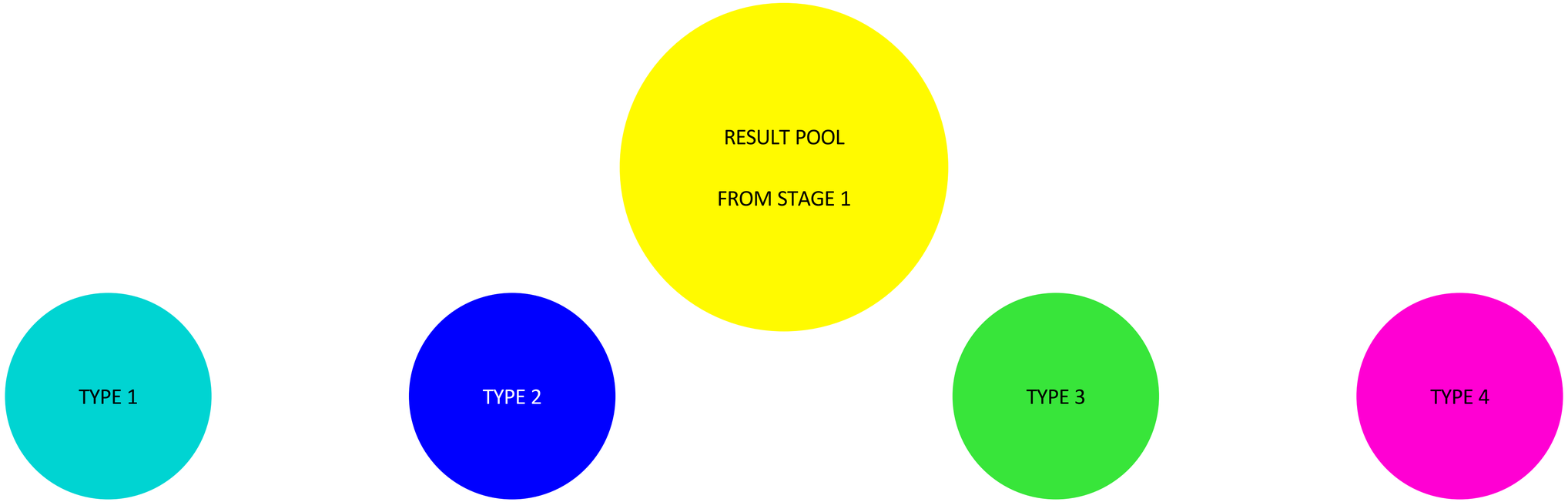
## STAGE 1: EVALUATION

| EVALUATION METRIC | RESULT |
|---|---|
| WRITTEN EXAM | ANALYTICAL SKILLS |
| COUNSELLING | SOFT SKILLS<br>AND INTERACTION |
| PHYSICAL TEST | PHYSICAL HEALTH |
| CREATIVITY TEST | |

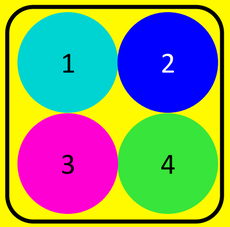
## STAGE 1: DELIVERABLE

***VISUALIZATION*** AND ***ANALYTICS*** OF A CHILD'S ABILITY!

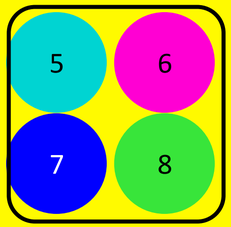
# STAGE 2: STATISTICAL MATCHING



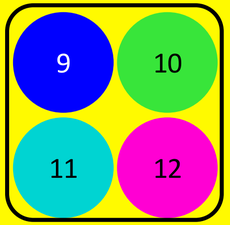

EACH BATCH



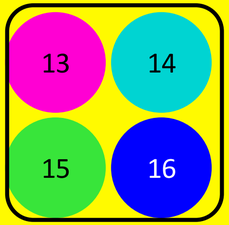

ACTIVITY GROUP 1




ACTIVITY GROUP 2




ACTIVITY GROUP 3




ACTIVITY GROUP 4

# ROLE SPECIFICATION

- SUSHMITHA: PYTHON FUNCTION FOR DECIDING THE TYPE OF STUDENT
- ASHISH: STATISTICAL ANALYSIS FOR MATCHING
- ABHISHEK: CREATING AND MAINTAINING THE DATABASE
- SNEHITH: DATA VISUALISATION
- YASHWANTH: BUSINESS MODEL AND SCHEMA BUILD

In [9]:
```
# 02-05-2022
# group project
# team D - SAASY

#team mates

# Sushmita Dogga
# Ashish Das
# Abhishek Koundle
# Nukala Snehith
# Chamarthi Yashwanth

# Title: Education System in INDIA
# about - Analysing Each student individually on parameters like Analytical s
# And matching the students with different skilled children in a team for eac
# interest from a very early point and can be multi skilled

import matplotlib.pyplot as plt
import numpy as np
import pandas as pd
```

In [2]:
```
df = pd.read_csv(r"C:\Users\Yash\exam.csv") # importing the file that contain
```

In [3]:
```
# grouping for each student

def stu_type_individual(n):
    n = n-1
    AS =  (df['AS_mth'][n]+df['AS_apt'][n]+df['AS_mem'][n])/3 # average of ar
    SS =  (df['SS_cc'][n]+df['SS_pp'][n]+df['SS_lis'][n])/3   # average of sc
    PA =  (df['PA_ss'][n]+df['PA_mi'][n]+df['PA_bmi'][n])/3    # average of p
    CA =  (df['CA_foc'][n]+df['CA_time'][n]+df['CA_tools'][n])/3  # average c
    avg = round((AS+SS+PA+CA)/4) # finding the average of all the categories
    AS_max =  max([df['AS_mth'][n],df['AS_apt'][n],df['AS_mem'][n]])
    SS_max =  max([df['SS_cc'][n],df['SS_pp'][n],df['SS_lis'][n]])
    PA_max =  max([df['PA_ss'][n],df['PA_mi'][n],df['PA_bmi'][n]])
    CA_max =  max([df['CA_foc'][n],df['CA_time'][n],df['CA_tools'][n]])
    as_x = abs(avg - AS_max)  # finding the parameter which is close or equal
    ss_x = abs(avg - SS_max)
    pa_x = abs(avg - PA_max)
    ca_x = abs(avg - CA_max)
    x = min([as_x,ss_x,pa_x,ca_x])
    if x == as_x:
        return 'Type A'  # classsifying into types
    elif x == ss_x:
        return 'Type B'
    elif x == pa_x:
        return 'Type C'
    elif x == ca_x:
        return 'Type D'
```

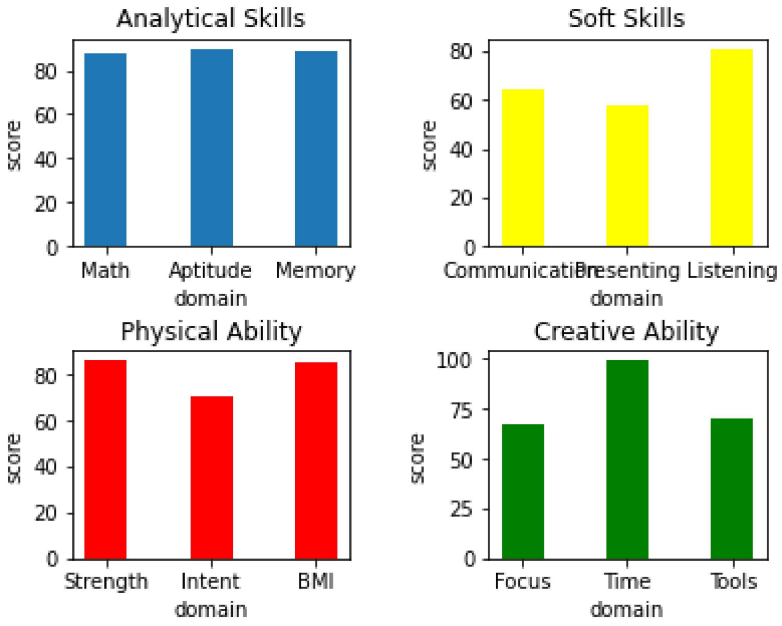
In [4]:
```
a = []
for i in df['ID']:
    a.append(stu_type_individual(i))
df['Type'] = a
df
```

Out[4]:

| | ID | AS_mth | AS_apt | AS_mem | SS_cc | SS_pp | SS_lis | PA_ss | PA_mi | PA_bmi | CA_foc |
|---|---|---|---|---|---|---|---|---|---|---|---|
| **0** | 1 | 88 | 90 | 89 | 64 | 58 | 81 | 86 | 70 | 85 | 67 |
| **1** | 2 | 91 | 86 | 88 | 87 | 90 | 93 | 35 | 84 | 43 | 87 |
| **2** | 3 | 94 | 88 | 91 | 81 | 84 | 86 | 48 | 46 | 54 | 77 |
| **3** | 4 | 91 | 93 | 92 | 62 | 55 | 82 | 82 | 81 | 83 | 94 |
| **4** | 5 | 83 | 83 | 83 | 55 | 53 | 71 | 90 | 90 | 91 | 95 |
| **5** | 6 | 75 | 79 | 77 | 61 | 52 | 45 | 85 | 89 | 88 | 95 |
| **6** | 7 | 80 | 80 | 80 | 88 | 84 | 90 | 69 | 58 | 67 | 65 |
| **7** | 8 | 84 | 70 | 76 | 78 | 73 | 85 | 87 | 75 | 86 | 83 |
| **8** | 9 | 57 | 54 | 55 | 82 | 78 | 77 | 81 | 59 | 72 | 33 |
| **9** | 10 | 70 | 62 | 66 | 62 | 61 | 79 | 86 | 58 | 80 | 69 |
| **10** | 11 | 99 | 99 | 99 | 91 | 96 | 95 | 41 | 78 | 39 | 85 |
| **11** | 12 | 69 | 78 | 74 | 93 | 96 | 96 | 46 | 90 | 49 | 81 |
| **12** | 13 | 79 | 77 | 78 | 89 | 93 | 93 | 41 | 94 | 41 | 82 |
| **13** | 14 | 79 | 79 | 79 | 80 | 80 | 84 | 59 | 79 | 60 | 82 |
| **14** | 15 | 81 | 83 | 82 | 70 | 73 | 78 | 83 | 73 | 78 | 78 |
| **15** | 16 | 73 | 71 | 72 | 71 | 66 | 71 | 88 | 93 | 89 | 95 |

In [21]:

```
# visualising exam student performance

def vis_func(n):
    n = n-1
    as_vis = np.array([df['AS_mth'][n],df['AS_apt'][n],df['AS_mem'][n]])
    ss_vis = np.array([df['SS_cc'][n],df['SS_pp'][n],df['SS_lis'][n]])
    pa_vis = np.array([df['PA_ss'][n],df['PA_mi'][n],df['PA_bmi'][n]])
    ca_vis = np.array([df['CA_foc'][n],df['CA_time'][n],df['CA_tools'][n]])

    x_as = np.array(['Math','Aptitude','Memory'])
    x_ss = np.array(['Communication','Presenting','Listening'])
    x_pa = np.array(['Strength','Intent','BMI'])
    x_ca = np.array(['Focus','Time','Tools'])

    # plot 1
    plt.subplot(2,2,1)
    plt.bar(x_as,as_vis,width=0.4)
    plt.title('Analytical Skills')
    plt.xlabel('domain')
    plt.ylabel('score')

    # plot 2
    plt.subplot(2,2,2)
    plt.bar(x_ss,ss_vis,color='yellow',width=0.4)
    plt.title('Soft Skills')
    plt.xlabel('domain')
    plt.ylabel('score')

    # plot 3
    plt.subplot(2,2,3)
    plt.bar(x_pa,pa_vis,color='red',width=0.4)
    plt.title('Physical Ability')
    plt.xlabel('domain')
    plt.ylabel('score')

    # plot 4
    plt.subplot(2,2,4)
    plt.bar(x_ca,ca_vis,color='green',width=0.4)
    plt.title('Creative Ability')
    plt.xlabel('domain')
    plt.ylabel('score')
    plt.subplots_adjust(left=0.2,
                        bottom=0.3,
                        right=1.0,
                        top=1.2,
                        wspace=0.5,
                        hspace=0.5)
    plt.show()

vis_func(1) # takes student id as input
```


Analytical Skills
score
0
20
40
60
80
Math
Aptitude
Memory
domain
Soft Skills
score
0
20
40
60
80
Communication
Presenting
Listening
domain
Physical Ability
score
0
20
40
60
80
Strength
Intent
BMI
domain
Creative Ability
score
0
25
50
75
100
Focus
Time
Tools
domain

In [ ]: