

FINDING THE FIRST FIVE HIGHEST IN EXCEL COLUMN

To find the **first five highest values** in an Excel column, you can use a few different methods depending on how you want to display the results. Here are the most effective approaches:

Method 1: Using the LARGE Function

If your data is in **Column A** (say, from A2 to A100), you can use the LARGE function like this:

excel

=LARGE(A\$2:A\$100, 1) → Highest

=LARGE(A\$2:A\$100, 2) → 2nd Highest

=LARGE(A\$2:A\$100, 3) → 3rd Highest

=LARGE(A\$2:A\$100, 4) → 4th Highest

=LARGE(A\$2:A\$100, 5) → 5th Highest

 You can place these formulas in cells B2 to B6 to list the top five values.

Method 2: Using SORT and INDEX (Excel 365 / Excel 2019+)

If you're using a newer version of Excel that supports dynamic arrays:

excel

=SORT(A2:A100, 1, -1)

This will sort the column in descending order. To get just the top five:

excel

=INDEX(SORT(A2:A100, 1, -1), SEQUENCE(5))

This returns the top 5 values as a vertical list.

Method 3: Advanced Filtering (Manual)

1. Select your column.
2. Go to **Data → Sort Z to A** (Descending).
3. Copy the first five rows manually.

 **Bonus Tip:** If you want to get the top 5 unique values, and your column has duplicates, you can use:

excel

```
=UNIQUE(SORT(A2:A100, 1, -1))
```

Then wrap it with INDEX(..., SEQUENCE(5)) to get the top 5 unique values.

Would you like to also extract the **corresponding row data** (like names or IDs next to the top values)? I can show you how to do that with XLOOKUP or FILTER.

What if I want to get the name also alongwith descending order of marks obtained

Great question, Jagdev! If you have a list of **names** in one column and their corresponding **marks** in another, and you want to display the **top 5 names along with their marks in descending order**, here's how to do it dynamically in Excel.

Let's assume:

- Column A: Names (A2:A100)
- Column B: Marks (B2:B100)

Method 1: Using SORT and INDEX (Excel 365 / Excel 2019+)

You can use the SORT function to sort both columns together:

excel

```
=SORT(A2:B100, 2, -1)
```

This sorts the entire range by **marks (column 2)** in descending order.

To get only the **top 5**, wrap it with INDEX and SEQUENCE:

excel

```
=INDEX(SORT(A2:B100, 2, -1), SEQUENCE(5), {1,2})
```

- ◆ This returns a 5-row, 2-column array with names and marks sorted by marks descending.

Method 2: Using FILTER and SORT (if you want more control)

excel

```
=FILTER(SORT(A2:B100, 2, -1), ROW(A2:A100)<=5)
```

This filters the top 5 rows after sorting. You can adjust the logic if your data has gaps or headers.

Method 3: Manual Helper Column (for older Excel versions)

1. In column C, use =LARGE(B\$2:B\$100, ROW()-1) in C2 to get top marks.

2. In column D, use =INDEX(A\$2:A\$100, MATCH(C2, B\$2:B\$100, 0)) to get the corresponding name.

Drag both formulas down for 5 rows.

Would you like to also **highlight** the top 5 in the original table or create a **dashboard-style view**? I can help with that too!

What if I want Names using formula =LARGE(A\$2:A\$100, 1) → Highest =LARGE(A\$2:A\$100, 2) → 2nd Highest =LARGE(A\$2:A\$100, 3) → 3rd Highest =LARGE(A\$2:A\$100, 4) → 4th Highest =LARGE(A\$2:A\$100, 5) → 5th Highest

Perfect! If you're using the LARGE function to extract the top 5 **marks**, and you want to get the **corresponding names**, you can pair it with INDEX and MATCH.

Let's assume:

- Column A: Names (A2:A100)
- Column B: Marks (B2:B100)

Step-by-Step Formula Setup

In column C, you extract the top 5 marks:

excel

C2: =LARGE(B\$2:B\$100, 1)

C3: =LARGE(B\$2:B\$100, 2)

C4: =LARGE(B\$2:B\$100, 3)

C5: =LARGE(B\$2:B\$100, 4)

C6: =LARGE(B\$2:B\$100, 5)

Then in column D, use this to get the corresponding name:

excel

D2: =INDEX(A\$2:A\$100, MATCH(C2, B\$2:B\$100, 0))

D3: =INDEX(A\$2:A\$100, MATCH(C3, B\$2:B\$100, 0))

D4: =INDEX(A\$2:A\$100, MATCH(C4, B\$2:B\$100, 0))

D5: =INDEX(A\$2:A\$100, MATCH(C5, B\$2:B\$100, 0))

D6: =INDEX(A\$2:A\$100, MATCH(C6, B\$2:B\$100, 0))

⚠ Handling Duplicate Marks

If two students have the same marks, MATCH will always return the **first match**. To handle duplicates properly, you can use a more advanced array formula or switch to FILTER and SORT if you're on Excel 365.

Would you like a version that handles **duplicate marks** and returns **all matching names**? I can show you how to do that too.

Using this I got Top Ten Students 1 106 SANDEEP KAUR 2 101 KULJIT KAUR 3 98 RAJWINDER KAUR 4 97 KOMALPREET KAUR 5 96.5 MANJOT KAUR 6 95 JASPREET KAUR 7 94 ANMOLPREET KAUR 8 91 SIMRANJEET KAUR 9 88 NEELAM KAUR 10 86 SIMARJEET KAUR 11 85 MUSKANDEEP KAUR 12 84 VANSHDEEP SINGH 13 83 AJANBIR SINGH 14 80 ARMANDEEP KAUR 15 80 ARMANDEEP KAUR 16 79.5 KHUSHPREET KAUR that is Armandeep Kaur 2 times while the second student with same marks=80 is the last student in the list?

Ah, excellent observation, Jagdev! You're running into a classic issue with MATCH: it always returns the **first occurrence** of a value. So when two students have the same marks (like both Armandeep Kaur with 80), MATCH will only find the first one—leading to duplicates in your top list and skipping the second student.

Let's fix that. 💡

✓ Solution: Use a Helper Column to Make Marks Unique

We'll slightly modify the marks so each one becomes unique by adding a tiny value based on row number. This way, LARGE will treat them as distinct, and MATCH will find different rows.

◆ Step 1: Create a helper column (say, Column C)

excel

C2: =B2 + ROW()/100000

Drag this down to C100. This adds a very small value to each mark based on its row number—enough to make duplicates unique without affecting sorting.

◆ Step 2: Use LARGE on the helper column

excel

D2: =LARGE(C\$2:C\$100, 1)

D3: =LARGE(C\$2:C\$100, 2)

...

D11: =LARGE(C\$2:C\$100, 10)

◆ **Step 3: Use MATCH on the helper column to get correct row**

excel

E2: =MATCH(D2, C\$2:C\$100, 0)

◆ **Step 4: Use INDEX to get the name and actual marks**

excel

F2: =INDEX(A\$2:A\$100, E2) → Name

G2: =INDEX(B\$2:B\$100, E2) → Actual Marks

Repeat for rows 2 to 11.

 **Result**

Now you'll get the **top 10 students**, even if some have identical marks, and each will appear only once with correct names.

Would you like me to help you turn this into a clean printable report or dashboard-style layout?

I have this list for Bimonthly Exam 20 marks for each subject: S.No Student ID Name FatherName MotherName Gender Class Punjabi Hindi English Math Science Social Science C.Sc Mechanical Drawing Total Percentage BI-MONTHLY EXAMINATION HELD IN JULY 2025 20 20 20 20 20 20 20 20 120 1 8792461 AJANBIR SINGH DAVINDER SINGH MANINDER KAUR Boy 10 13 15 14 12 13 16 8 15 83 69.17 2 9012827 ANMOLPREET KAUR SARWAN SINGH RANJIT KAUR Girl 10 12 18 17 16 12 19 9 16 94 78.33 3 7069347 ARMANDEEP SINGH JASWANT SINGH HARJINDER KAUR Boy 10 10 12 11 10 11 18 9 15 72 60.00 4 8614037 ARMANDEEP KAUR DALBIR SINGH RAMANDEEP KAUR Girl 10 14 17 12 12 7 18 6 15 80 66.67 5 8790511 ARMANDEEP SINGH NIRMAL SINGH BALJIT KAUR Boy 10 11 14 12 13 6 16 8 16 72 60.00 6 13890949 BALRAJ SINGH BIKRAMJIT SINGH LAKWINDER KAUR Boy 10 12 12 11 14 7 18 4 15 74 61.67 7 15773856 BIKRAMJIT SINGH JANTA MANROOP Boy 10 10 14 15 14 5 18 4 14 76 63.33 8 7077019 DILPREET KAUR NARINDER SINGH SHARANJIT KAUR Girl 10 11 12 13 12 5 18 6 15 71 59.17 9 7080980 GURPREET KAUR NARINDER SINGH KULWANT KAUR Girl 10 8 15 14 10 0 15 2 15 62 51.67 10 3958507 JARMANJEET SINGH SAHIB SINGH SUKHWINDER KAUR Boy 10 10 14 11 11 4 16 4 16 66 55.00 11 8789695 JASHANPREET SINGH SATNAM SINGH MADANPREET KAUR Boy 10 10 15 15 12 4 19 4 14 75 62.50 12 9062017 JASPREET KAUR GURJIT SINGH LAKHWINDER KAUR Girl 10 15 18 16 13 15 18 12 15 95 79.17 13 7069339 JIVAN SINGH GURDEEP SINGH SARABJIT KAUR Boy

10 10 13 10 12 0 15 0 16 60 50.00 14 8790429 KARANDEEP SINGH BALJIT SINGH JASPAL KAUR
Boy 10 9 12 15 10 4 18 5 14 68 56.67 15 8790588 KARANPREET SINGH RAJPAL SINGH PARAMJIT
KAUR Boy 10 9 13 17 12 4 18 5 15 73 60.83 16 9198115 KHUSHPREET KAUR NISHAN SINGH
SIMRANJIT KAUR Girl 10 11 14 14.5 12 12 16 11 15 79.5 66.25 17 8792320 KOMALPREET KAUR
DILBAG SINGH DALJIT KAUR Girl 10 14 18 16 16 14 19 14 14 97 80.83 18 8790676 KULJIT KAUR
MANGAL SINGH JASPINDER KAUR Girl 10 16 18 17 16 15 19 15 13 101 84.17 19 8790345
LOVEJOT SINGH SADHA SINGH JASWANT KAUR Boy 10 10 16 13 13 6 19 6 15 77 64.17 20
8806313 LOVEJOT SINGH JASWANT SINGH HARJINDER KAUR Boy 10 7 12 10 10 0 14 4 14 53
44.17 21 8797009 MANJOT KAUR GURPEET SINGH RAJBIR KAUR Girl 10 14 18 18.5 16 14 16 14
16 96.5 80.42 22 7080966 MANPREET KAUR NARINDER SINGH KULWANT KAUR Girl 10 9 16 13
14 0 13 0 15 65 54.17 23 8792146 MUSKANDEEP KAUR MADHA SINGH JASBIR KAUR Girl 10 11
18 14 16 8 18 10 12 85 70.83 24 8797216 MUSKANPREET KAUR BHAGWANT SINGH PALWINDER
KAUR Girl 10 12 14 11 15 7 19 10 14 78 65.00 25 8797160 NEELAM KAUR JASWANT SINGH
MANPREET KAUR Girl 10 14 16 12 16 12 18 14 12 88 73.33 26 8806296 PAWANDEEP KAUR
GURDEEP SINGH SARBJIT KAUR Girl 10 10 15 12 14 0 19 0 14 70 58.33 27 8797236 RAJVEER
SINGH KULDEEP SINGH MANPREET KAUR Boy 10 8 12 10 12 9 19 4 15 70 58.33 28 9777058
RAJWINDER KAUR MAJOR SINGH PREET KAUR Girl 10 14 17 16 19 13 19 15 12 98 81.67 29
8790228 SAHILPREET SINGH GURBINDER SINGH SARBJIT KAUR Boy 10 8 12 12 14 8 14 6 13 68
56.67 30 8792207 SANDEEP KAUR GURMEET SINGH SONIA Girl 10 17 18 18 18 17 18 17 17 106
88.33 31 8835723 SANDEEP SINGH JASPAL SINGH GURPREET KAUR Boy 10 7 12 12 14 0 14 0 2
59 49.17 32 8797179 SARTAJ SINGH MANGAL SINGH RAJWINDER KAUR Boy 10 8 14 13 14 4 14 4
4 67 55.83 33 9062159 SATBIR KAUR HARJINDER SINGH KARMJIT KAUR Girl 10 8 14 14 15 0 15 0
0 66 55.00 34 10352570 SIMARJEET KAUR AMARJIT SINGH RANJIT KAUR Girl 10 10 18 17 17 7
17 5 6 86 71.67 35 10118973 SIMRANJEET KAUR PARWINDER SINGH RANJIT KAUR Girl 10 10 18
15 19 10 19 6 7 91 75.83 36 8814734 VANSHDEEP SINGH TARANJIT SINGH KULWINDER KAUR
Boy 10 9 17 16 18 6 18 5 6 84 70.00 37 7407917 VANSHPREET SINGH GURBINDER SINGH
SARBJIT KAUR Boy 10 9 16 13 18 6 18 5 6 80 66.67 the sum excludes mark in C.Sc and
Mechanical Drawing? Can you help me design a python and streamlit project based on this
excel sheet?

Absolutely, Jagdev! This dataset is rich and perfect for a modular, scalable Streamlit project. Since you're already skilled in modular design and advanced features like visitor logging and authentication, we can build a robust educational dashboard that:

- Visualizes student performance
- Calculates accurate totals and percentages
- Filters by class, gender, or subject

- 🏆 Highlights top performers
- 🔒 Optionally includes authentication and visitor logging

❖ Project Overview: Student Performance Dashboard

◆ Core Features

Feature	Description
Data Upload	Upload Excel/CSV file with student marks
Subject-wise Analysis	View average, highest, and lowest marks per subject
Top Performers	Display top N students based on total or percentage
Filters	Filter by class, gender, or specific subjects

Custom Total Calculation Exclude optional subjects like C.Sc and Mechanical Drawing

Downloadable Reports Export filtered data or top performers

Authentication (Optional) Secure access using streamlit-authenticator

Visitor Logging (Optional) Track usage and interactions

❖ File Structure (Modularized)

```
student_dashboard/
|
├── app.py          # Main Streamlit app
├── data_loader.py  # Load and preprocess Excel data
├── analytics.py    # Compute totals, percentages, top performers
├── filters.py      # Filter logic by class, gender, etc.
├── visualizations.py # Charts and tables
├── config.yaml     # Authenticator config (optional)
├── visitor_log.py  # Visitor tracking (optional)
└── assets/
```

└── sample_data.xlsx # Your Excel file

Key Logic: Total & Percentage Calculation

python

```
# analytics.py
```

```
def calculate_scores(df):
```

```
    core_subjects = ['Punjabi', 'Hindi', 'English', 'Math', 'Science', 'Social Science']
```

```
    df['Total'] = df[core_subjects].sum(axis=1)
```

```
    df['Percentage'] = (df['Total'] / (len(core_subjects) * 20)) * 100
```

```
    return df
```

Streamlit App Skeleton

python

```
# app.py
```

```
import streamlit as st
```

```
from data_loader import load_data
```

```
from analytics import calculate_scores, get_top_students
```

```
from filters import apply_filters
```

```
from visualizations import show_summary, show_top_table
```

```
st.title("📊 Student Performance Dashboard")
```

```
uploaded_file = st.file_uploader("Upload Excel File", type=["xlsx", "csv"])
```

```
if uploaded_file:
```

```
    df = load_data(uploaded_file)
```

```
    df = calculate_scores(df)
```

```
st.sidebar.header("🔍 Filters")
```

```
filtered_df = apply_filters(df)
```

```
show_summary(filtered_df)
```

```
show_top_table(filtered_df)
```

Visualizations

- Bar chart: Subject-wise average
- Table: Top 10 students
- Pie chart: Gender distribution
- Line chart: Class-wise performance trend

Optional Add-ons

- **Authentication:** Use streamlit-authenticator to restrict access
- **Visitor Logging:** Log user actions with timestamps and filters used

Would you like me to generate the full code for each module, or start with one like `data_loader.py` or `analytics.py`? We can build it step-by-step and tailor it to your exact needs.

Help me design this project in VS Code with file by file code. Also give me terraforming like command to generate directory structure mentioned above.

Perfect, Jagdev! Let's build this project step-by-step in **VS Code**, starting with a clean directory structure and modular files. I'll give you:

1.  **Terraforming-style command** to create folders and files
2.  **File-by-file code** for each module
3.  A sample Excel file structure reference
4.  Instructions to run the app

1. Directory Setup (Terraforming-style)

Open your terminal in VS Code and run this:

```
bash
```

```
mkdir student_dashboard && cd student_dashboard
```

```
# Create Python files  
touch app.py data_loader.py analytics.py filters.py visualizations.py visitor_log.py
```

```
# Create config file for authentication  
touch config.yaml
```

```
# Create assets folder and placeholder for Excel file  
mkdir assets && touch assets/sample_data.xlsx
```

2. data_loader.py – Load and Clean Excel Data

```
python  
import pandas as pd
```

```
def load_data(file):  
    df = pd.read_excel(file)  
  
    # Ensure consistent column names  
    df.columns = df.columns.str.strip()  
  
    return df
```

3. analytics.py – Calculate Totals, Percentages, Top Performers

```
python  
def calculate_scores(df):  
  
    core_subjects = ['Punjabi', 'Hindi', 'English', 'Math', 'Science', 'Social Science']  
  
    df['Total'] = df[core_subjects].sum(axis=1)  
  
    df['Percentage'] = (df['Total'] / (len(core_subjects) * 20)) * 100  
  
    return df
```

```
def get_top_students(df, top_n=10):  
    return df.sort_values(by='Percentage', ascending=False).head(top_n)
```

 **4. filters.py – Filter by Class, Gender, etc.**

python

```
import streamlit as st
```

```
def apply_filters(df):  
    gender = st.sidebar.selectbox("Gender", ["All"] + sorted(df['Gender'].unique()))  
    class_ = st.sidebar.selectbox("Class", ["All"] + sorted(df['Class'].unique()))
```

```
if gender != "All":
```

```
    df = df[df['Gender'] == gender]
```

```
if class_ != "All":
```

```
    df = df[df['Class'] == class_]
```

```
return df
```

 **5. visualizations.py – Charts and Tables**

python

```
import streamlit as st
```

```
import pandas as pd
```

```
def show_summary(df):  
    st.subheader("📊 Subject-wise Averages")  
    subjects = ['Punjabi', 'Hindi', 'English', 'Math', 'Science', 'Social Science']  
    avg_scores = df[subjects].mean().round(2)  
    st.bar_chart(avg_scores)
```

```
def show_top_table(df):  
    from analytics import get_top_students  
  
    st.subheader("🏆 Top Performers")  
  
    top_df = get_top_students(df)  
  
    st.dataframe(top_df[['Name', 'Class', 'Gender', 'Total', 'Percentage']])
```

💻 6. visitor_log.py – Optional Visitor Logging

```
python  
  
import datetime  
  
  
def log_visitor(user, action):  
    with open("visitor_log.txt", "a") as f:  
        f.write(f"{datetime.datetime.now()} - {user}: {action}\n")
```

🔒 7. config.yaml – Authentication Setup (Optional)

```
yaml  
  
credentials:  
  
    usernames:  
  
        jagdev:  
            email: jagdev@example.com  
            name: Jagdev Singh  
            password: "hashed_password_here" # Use hashed password from streamlit-authenticator
```

```
cookie:  
  
    expiry_days: 1  
    key: "some_signature_key"  
    name: "student_dashboard_cookie"
```

preauthorized:

emails:

- jagdev@example.com

8. app.py – Main Streamlit App

python

```
import streamlit as st

from data_loader import load_data

from analytics import calculate_scores

from filters import apply_filters

from visualizations import show_summary, show_top_table
```

```
st.set_page_config(page_title="Student Dashboard", layout="wide")

st.title("📊 Student Performance Dashboard")
```

```
uploaded_file = st.file_uploader("Upload Excel File", type=["xlsx"])

if uploaded_file:

    df = load_data(uploaded_file)

    df = calculate_scores(df)

    filtered_df = apply_filters(df)
```

```
show_summary(filtered_df)
```

```
show_top_table(filtered_df)
```

Sample Excel File Format (assets/sample_data.xlsx)

Make sure your Excel file has these columns:

Bi-Monthly Student Dashboard
jagdevsinghdosanjh@gmail.com

S.No | Student ID | Name | FatherName | MotherName | Gender | Class | Punjabi | Hindi | English | Math | Science | Social Science | C.Sc | Mechanical Drawing

 **Run the App**

In terminal:

bash

streamlit run app.py

Would you like me to help you integrate authentication next, or add download/export features for filtered data and top performers?

```
(base) PS C:\Users\jagdevsinghdosanjh.RHINO\biomnthly-student-dashboard> mkdir student_dashboard && cd student_dashboard >>> # Create Python files >> touch app.py data_loader.py analytics.py filters.py visualizations.py visitor_log.py >>>> # Create config file for authentication >> touch config.yaml >>>> # Create assets folder and placeholder for Excel file >> mkdir assets && touch assets/sample_data.xlsx >> At line:1 char:25 + mkdir student_dashboard && cd student_dashboard + ~~ The token '&&' is not a valid statement separator in this version. At line:10 char:14 + mkdir assets && touch assets/sample_data.xlsx + ~~ The token '&&' is not a valid statement separator in this version. + CategoryInfo : ParserError: (:) [], ParentContainsErrorRecordException + FullyQualifiedErrorId : InvalidEndOfLine (base) PS C:\Users\jagdevsinghdosanjh.RHINO\biomnthly-student-dashboard>
```

Ah, you're using **PowerShell**, which doesn't support `&&` like Bash does. No worries—we'll just break the commands into separate lines. Here's how to set up your project step-by-step in PowerShell:

 **PowerShell-Friendly Setup Commands**

powershell

Create the main project folder

mkdir student_dashboard

cd student_dashboard

Create Python files

New-Item app.py -ItemType File

New-Item data_loader.py -ItemType File

Bi-Monthly Student Dashboard
jagdevsinghdosanjh@gmail.com

New-Item analytics.py -ItemType File

New-Item filters.py -ItemType File

New-Item visualizations.py -ItemType File

New-Item visitor_log.py -ItemType File

Create config file

New-Item config.yaml -ItemType File

Create assets folder and sample Excel file

mkdir assets

New-Item assets\sample_data.xlsx -ItemType File

After running these, your folder structure will look like:

student_dashboard/

 └── app.py

 └── data_loader.py

 └── analytics.py

 └── filters.py

 └── visualizations.py

 └── visitor_log.py

 └── config.yaml

└── assets/

 └── sample_data.xlsx

Would you like me to help you populate each file with code again, or continue with authentication and export features next?

Yes

Awesome! Let's now **populate each file** with the code we discussed earlier so your project is fully functional in VS Code. I'll walk you through each file one by one.

 **data_loader.py – Load and Clean Excel Data**

python

```
import pandas as pd
```

```
def load_data(file):
```

```
    df = pd.read_excel(file)
```

```
    df.columns = df.columns.str.strip() # Clean column names
```

```
    return df
```

 **analytics.py – Calculate Totals, Percentages, Top Performers**

python

```
def calculate_scores(df):
```

```
    core_subjects = ['Punjabi', 'Hindi', 'English', 'Math', 'Science', 'Social Science']
```

```
    df['Total'] = df[core_subjects].sum(axis=1)
```

```
    df['Percentage'] = (df['Total'] / (len(core_subjects) * 20)) * 100
```

```
    return df
```

```
def get_top_students(df, top_n=10):
```

```
    return df.sort_values(by='Percentage', ascending=False).head(top_n)
```

 **filters.py – Sidebar Filters**

python

```
import streamlit as st
```

```
def apply_filters(df):
```

```
    gender = st.sidebar.selectbox("Filter by Gender", ["All"] + sorted(df['Gender'].unique()))
```

Bi-Monthly Student Dashboard
jagdevsinghdosanjh@gmail.com

```
class_ = st.sidebar.selectbox("Filter by Class", ["All"] + sorted(df['Class'].unique()))  
  
if gender != "All":  
    df = df[df['Gender'] == gender]  
  
if class_ != "All":  
    df = df[df['Class'] == class_]  
  
return df
```

 **visualizations.py – Charts and Top Table**

python

```
import streamlit as st
```

```
def show_summary(df):
```

```
    st.subheader("📊 Subject-wise Averages")  
  
    subjects = ['Punjabi', 'Hindi', 'English', 'Math', 'Science', 'Social Science']  
  
    avg_scores = df[subjects].mean().round(2)  
  
    st.bar_chart(avg_scores)
```

```
def show_top_table(df):
```

```
    from analytics import get_top_students  
  
    st.subheader("🏆 Top Performers")  
  
    top_df = get_top_students(df)  
  
    st.dataframe(top_df[['Name', 'Class', 'Gender', 'Total', 'Percentage']])
```

 **visitor_log.py – Optional Logging**

python

```
import datetime
```

```
def log_visitor(user, action):
    with open("visitor_log.txt", "a") as f:
        f.write(f"{datetime.datetime.now()} - {user}: {action}\n")
```

config.yaml – Authentication Setup (Optional)

yaml

credentials:

usernames:

jagdev:

email: jagdev@example.com

name: Jagdev Singh

password: "hashed_password_here" # Replace with hashed password

cookie:

expiry_days: 1

key: "some_signature_key"

name: "student_dashboard_cookie"

preauthorized:

emails:

- jagdev@example.com

 You can generate a hashed password using:

python

```
from streamlit_authenticator import Hasher
hashed = Hasher(['your_password']).generate()
print(hashed)
```

app.py – Main Streamlit App

python

```
import streamlit as st

from data_loader import load_data

from analytics import calculate_scores

from filters import apply_filters

from visualizations import show_summary, show_top_table
```

```
st.set_page_config(page_title="Student Dashboard", layout="wide")
```

```
st.title("📊 Student Performance Dashboard")
```

```
uploaded_file = st.file_uploader("Upload Excel File", type=["xlsx"])
```

```
if uploaded_file:
```

```
    df = load_data(uploaded_file)
```

```
    df = calculate_scores(df)
```

```
    filtered_df = apply_filters(df)
```

```
    show_summary(filtered_df)
```

```
    show_top_table(filtered_df)
```

To Run the App

In your terminal:

bash

streamlit run app.py

Would you like to add:

-  Authentication using streamlit-authenticator?
-  Export filtered data or top performers to Excel?

-  More visualizations like pie charts or heatmaps?

Let's keep building!

where to add hashed generated password

Great question, Jagdev! Once you've generated the hashed password using streamlit-authenticator, you'll need to **paste it into your config.yaml file** under the password field for the corresponding user.

Step-by-Step: Adding Hashed Password to config.yaml

1. Generate the hash

In a Python script or interactive shell:

```
python
```

```
from streamlit_authenticator import Hasher

hashed_passwords = Hasher(['your_plaintext_password']).generate()

print(hashed_passwords)
```

This will output something like:

```
python
```

```
['$2b$12$K9z...yourHashedPasswordHere']
```

2. Paste it into config.yaml

Update your config.yaml like this:

```
yaml
```

```
credentials:
```

```
    usernames:
```

```
        jagdev:
```

```
            email: jagdev@example.com
```

```
            name: Jagdev Singh
```

```
            password: "$2b$12$K9z...yourHashedPasswordHere" # Paste the full hash here
```

cookie:

```
expiry_days: 1  
key: "some_signature_key"  
name: "student_dashboard_cookie"
```

preauthorized:

```
emails:  
- jagdev@example.com
```

Make sure:

- The hash is enclosed in **quotes**.
- You **don't** include the square brackets [] from the output—just the string itself.

Pro Tip

If you're managing multiple users, just repeat the structure under `usernames`:

yaml

`usernames:`

`jagdev:`

...

`another_user:`

`email: user2@example.com`

`name: User Two`

`password: "$2b$12$AnotherHashedPassword"`

Would you like help integrating this config into your `app.py` with login logic? I can walk you through that next.

I would like to add:  Authentication using streamlit-authenticator?  Export filtered data or top performers to Excel?  More visualizations like pie charts or heatmaps?

Perfect! You're about to level up your dashboard in a big way. Here's how we'll tackle each enhancement:

1. Authentication with streamlit-authenticator

Install the package

bash

```
pip install streamlit-authenticator
```

Update app.py to include login

Add this at the top of your app.py:

python

```
import streamlit_authenticator as stauth
```

```
import yaml
```

```
from yaml.loader import SafeLoader
```

```
# Load config.yaml
```

```
with open('config.yaml') as file:
```

```
    config = yaml.load(file, Loader=SafeLoader)
```

```
authenticator = stauth.Authenticate(
```

```
    config['credentials'],
```

```
    config['cookie']['name'],
```

```
    config['cookie']['key'],
```

```
    config['cookie']['expiry_days'],
```

```
    config['preauthorized'])
```

```
)
```

```
name, authentication_status, username = authenticator.login("Login", "main")
```

```
if authentication_status:
```

```
authenticator.logout("Logout", "sidebar")

st.sidebar.success(f"Welcome {name} 🙌")

# Place your dashboard code here

elif authentication_status is False:

    st.error("Username or password is incorrect")

elif authentication_status is None:

    st.warning("Please enter your username and password")
```

Then **indent your dashboard code** (file upload, filters, visualizations) under if authentication_status::

2. Export Filtered Data or Top Performers to Excel

Add this to the bottom of your dashboard code:

```
python
```

```
import io
```

```
st.subheader("📥 Download Filtered Data")

buffer = io.BytesIO()

filtered_df.to_excel(buffer, index=False)

st.download_button(
    label="Download Filtered Data as Excel",
    data=buffer,
    file_name="filtered_students.xlsx",
    mime="application/vnd.openxmlformats-officedocument.spreadsheetml.sheet"
)
```

```
st.subheader("📥 Download Top Performers")

top_df = get_top_students(filtered_df)
```

```
buffer_top = io.BytesIO()  
  
top_df.to_excel(buffer_top, index=False)  
  
st.download_button(  
    label="Download Top Performers as Excel",  
    data=buffer_top,  
    file_name="top_students.xlsx",  
    mime="application/vnd.openxmlformats-officedocument.spreadsheetml.sheet"  
)
```

3. Add Pie Charts and Heatmaps

Update visualizations.py:

Pie Chart – Gender Distribution

python

```
def show_gender_pie(df):  
  
    import plotly.express as px  
  
    gender_counts = df['Gender'].value_counts().reset_index()  
  
    gender_counts.columns = ['Gender', 'Count']  
  
    fig = px.pie(gender_counts, names='Gender', values='Count', title='Gender Distribution')  
  
    st.plotly_chart(fig)
```

Heatmap – Subject Correlation

python

```
def show_heatmap(df):  
  
    import seaborn as sns  
  
    import matplotlib.pyplot as plt  
  
  
    st.subheader("  Subject Score Correlation")  
    subjects = ['Punjabi', 'Hindi', 'English', 'Math', 'Science', 'Social Science']
```

```
corr = df[subjects].corr()

fig, ax = plt.subplots()
sns.heatmap(corr, annot=True, cmap='coolwarm', ax=ax)
st.pyplot(fig)
```

Then call these in app.py:

```
python
from visualizations import show_gender_pie, show_heatmap

show_gender_pie(filtered_df)
show_heatmap(filtered_df)
```

Would you like me to help modularize these additions into separate files or walk you through testing them in VS Code?

Today

Give me updated files for these additional code updates

Absolutely, Jagdev! Here's your updated modular setup with all the new features integrated: authentication, Excel export, and enhanced visualizations. Each file is cleanly structured to keep your project scalable and maintainable.

config.yaml – Authentication Setup

yaml

credentials:

usernames:

jagdev:

email: jagdev@example.com

name: Jagdev Singh

password: "\$2b\$12\$YourHashedPasswordHere" # Replace with actual hash

Bi-Monthly Student Dashboard
jagdevsinghdosanjh@gmail.com

cookie:

```
expiry_days: 1  
key: "some_signature_key"  
name: "student_dashboard_cookie"
```

preauthorized:

emails:
- jagdev@example.com



python

```
import pandas as pd
```

```
def load_data(file):  
    df = pd.read_excel(file)  
    df.columns = df.columns.str.strip()  
    return df
```



python

```
def calculate_scores(df):  
    core_subjects = ['Punjabi', 'Hindi', 'English', 'Math', 'Science', 'Social Science']  
    df['Total'] = df[core_subjects].sum(axis=1)  
    df['Percentage'] = (df['Total'] / (len(core_subjects) * 20)) * 100  
    return df
```

```
def get_top_students(df, top_n=10):  
    return df.sort_values(by='Percentage', ascending=False).head(top_n)
```

 filters.py

python

```
import streamlit as st
```

```
def apply_filters(df):
```

```
    gender = st.sidebar.selectbox("Filter by Gender", ["All"] + sorted(df['Gender'].unique()))
```

```
    class_ = st.sidebar.selectbox("Filter by Class", ["All"] + sorted(df['Class'].unique()))
```

```
    if gender != "All":
```

```
        df = df[df['Gender'] == gender]
```

```
    if class_ != "All":
```

```
        df = df[df['Class'] == class_]
```

```
    return df
```

 visualizations.py

python

```
import streamlit as st
```

```
def show_summary(df):
```

```
    st.subheader("📊 Subject-wise Averages")
```

```
    subjects = ['Punjabi', 'Hindi', 'English', 'Math', 'Science', 'Social Science']
```

```
    avg_scores = df[subjects].mean().round(2)
```

```
    st.bar_chart(avg_scores)
```

```
def show_top_table(df):
```

```
    from analytics import get_top_students
```

Bi-Monthly Student Dashboard
jagdevsinghdosanjh@gmail.com

```
st.subheader("🏆 Top Performers")

top_df = get_top_students(df)

st.dataframe(top_df[['Name', 'Class', 'Gender', 'Total', 'Percentage']])

def show_gender_pie(df):

    import plotly.express as px

    gender_counts = df['Gender'].value_counts().reset_index()

    gender_counts.columns = ['Gender', 'Count']

    fig = px.pie(gender_counts, names='Gender', values='Count', title='Gender Distribution')

    st.plotly_chart(fig)

def show_heatmap(df):

    import seaborn as sns

    import matplotlib.pyplot as plt

    st.subheader("📊 Subject Score Correlation")

    subjects = ['Punjabi', 'Hindi', 'English', 'Math', 'Science', 'Social Science']

    corr = df[subjects].corr()

    fig, ax = plt.subplots()

    sns.heatmap(corr, annot=True, cmap='coolwarm', ax=ax)

    st.pyplot(fig)
```

 **visitor_log.py (Optional)**

```
python

import datetime
```

```
def log_visitor(user, action):  
    with open("visitor_log.txt", "a") as f:  
        f.write(f"{datetime.datetime.now()} - {user}: {action}\n")
```

app.py – Main App with All Features

```
python  
  
import streamlit as st  
  
import streamlit_authenticator as stauth  
  
import yaml  
  
from yaml.loader import SafeLoader  
  
import io  
  
  
from data_loader import load_data  
  
from analytics import calculate_scores, get_top_students  
  
from filters import apply_filters  
  
from visualizations import show_summary, show_top_table, show_gender_pie, show_heatmap  
  
  
# Load authentication config  
with open('config.yaml') as file:  
    config = yaml.load(file, Loader=SafeLoader)  
  
  
authenticator = stauth.Authenticate(  
    config['credentials'],  
    config['cookie']['name'],  
    config['cookie']['key'],  
    config['cookie']['expiry_days'],  
    config['preauthorized'])
```

Bi-Monthly Student Dashboard
jagdevsinghdosanjh@gmail.com

)

```
name, authentication_status, username = authenticator.login("Login", "main")
```

```
if authentication_status:
```

```
    authenticator.logout("Logout", "sidebar")
```

```
    st.sidebar.success(f"Welcome {name} 🙌")
```

```
st.set_page_config(page_title="Student Dashboard", layout="wide")
```

```
st.title("📊 Student Performance Dashboard")
```

```
uploaded_file = st.file_uploader("Upload Excel File", type=["xlsx"])
```

```
if uploaded_file:
```

```
    df = load_data(uploaded_file)
```

```
    df = calculate_scores(df)
```

```
    filtered_df = apply_filters(df)
```

```
    show_summary(filtered_df)
```

```
    show_top_table(filtered_df)
```

```
    show_gender_pie(filtered_df)
```

```
    show_heatmap(filtered_df)
```

```
# Export filtered data
```

```
st.subheader("📥 Download Filtered Data")
```

```
buffer = io.BytesIO()
```

```
filtered_df.to_excel(buffer, index=False)
```

Bi-Monthly Student Dashboard
jagdevsinghdosanjh@gmail.com

```
st.download_button(  
    label="Download Filtered Data as Excel",  
    data=buffer,  
    file_name="filtered_students.xlsx",  
    mime="application/vnd.openxmlformats-officedocument.spreadsheetml.sheet"  
)  
  
# Export top performers  
st.subheader("📥 Download Top Performers")  
top_df = get_top_students(filtered_df)  
buffer_top = io.BytesIO()  
top_df.to_excel(buffer_top, index=False)  
st.download_button(  
    label="Download Top Performers as Excel",  
    data=buffer_top,  
    file_name="top_students.xlsx",  
    mime="application/vnd.openxmlformats-officedocument.spreadsheetml.sheet"  
)
```

elif authentication_status is False:

```
st.error("Username or password is incorrect")
```

elif authentication_status is None:

```
st.warning("Please enter your username and password")
```

Let me know if you'd like:

- A folder structure suggestion for organizing these files
- A sample Excel file format for testing

- Help deploying this app online via Streamlit Cloud or another platform

Ready to roll this out?

create venv command and requirements.txt

Absolutely! Here's how to set up your **virtual environment** and create a requirements.txt file for your student dashboard project.

Step 1: Create a Virtual Environment

In PowerShell or your terminal inside the student_dashboard folder:

powershell

python -m venv venv

This creates a folder named venv containing your isolated Python environment.

Step 2: Activate the Virtual Environment

On Windows (PowerShell):

powershell

.\venv\Scripts\Activate

You should see (venv) appear in your terminal prompt.

Step 3: Install Required Packages

Run this to install all necessary packages:

bash

pip install streamlit pandas openpyxl plotly seaborn pyyaml streamlit-authenticator

Step 4: Create requirements.txt

Once everything is installed, generate the file:

bash

pip freeze > requirements.txt

This will include all the packages and versions used in your environment.

Sample requirements.txt (clean version)

If you want a manually curated version:

txt

streamlit

pandas

openpyxl

plotly

seaborn

pyyaml

streamlit-authenticator

Now you're ready to share or deploy your project with confidence. Would you like help setting up a `.gitignore` file or preparing this for deployment on Streamlit Cloud?

Yes

Perfect! Here's how to set up your project for clean version control and easy deployment on **Streamlit Cloud** or any Git-based platform.

Step 1: Create a `.gitignore` File

This file tells Git which files/folders to ignore (like your virtual environment and cache files).

Recommended `.gitignore` for your project:

`gitignore`

`# Ignore virtual environment`

`venv/`

`# Ignore Python cache files`

`__pycache__/`

`*.py[cod]`

`# Ignore Streamlit config and logs`

`.streamlit/`

*.log

Ignore OS-specific files

.DS_Store

Thumbs.db

Save this as .gitignore in your project root.

Step 2: Prepare for Streamlit Cloud Deployment

Streamlit Cloud reads your repo and looks for:

- requirements.txt — for dependencies
- A Python script (e.g., app.py) — your main Streamlit app
- Optional: .streamlit/config.toml — for custom settings

Example folder structure:

```
student_dashboard/
|
├── app.py
├── requirements.txt
├── .gitignore
├── venv/      # ignored by Git
└── .streamlit/
    └── config.toml  # optional
```

Optional config.toml (for wide layout, dark theme, etc.)

toml

[theme]

base = "dark"

primaryColor = "#4CAF50"

```
backgroundColor = "#0E1117"  
secondaryBackgroundColor = "#262730"  
textColor = "#FAFAFA"  
font = "sans serif"
```

[server]

```
headless = true  
enableCORS = false  
port = 8501
```

Final Tips

- Push your code to GitHub.
- Go to Streamlit Cloud and link your repo.
- It will auto-detect app.py and install dependencies from requirements.txt.

Would you like help writing a README.md to make your repo more professional and user-friendly?

```
streamlit_authenticator.utilities.exceptions.DeprecationError: This app has encountered an error. The original error message is redacted to prevent data leaks. Full error details have been recorded in the logs (if you're on Streamlit Cloud, click on 'Manage app' in the lower right of your app). Traceback: File "/mount/src/biomnthalystudentdashboard/app.py", line 16, in <module> authenticator = stauth Authenticate( config['credentials'], ...<3 lines>... config['preauthorized'] ) File "/home/adminuser/venv/lib/python3.13/site-packages/streamlit_authenticator/views/authentication_view.py", line 72, in __init__ raise DeprecationError(f"""Please note that the 'pre_authorized' parameter has been ...<2 lines>... {params.REGISTER_USER_LINK}."""")
```

Ah, I see what's happening! The error you're encountering is due to a **deprecated parameter** in the streamlit-authenticator library. Specifically, the preauthorized parameter in your config.yaml and Authenticate() call is no longer supported in the latest version.

Fix: Remove preauthorized from both places

Step 1: Update config.yaml

Remove this block entirely:

yaml

preauthorized:

emails:

- jagdev@example.com

Your updated config.yaml should look like:

yaml

credentials:

usernames:

jagdev:

email: jagdev@example.com

name: Jagdev Singh

password: "\$2b\$12\$YourHashedPasswordHere"

cookie:

expiry_days: 1

key: "some_signature_key"

name: "student_dashboard_cookie"

Step 2: Update app.py

Remove the config['preauthorized'] argument from the Authenticate() call:

python

```
authenticator = stauth.Authenticate(
```

```
    config['credentials'],
```

```
    config['cookie']['name'],
```

```
    config['cookie']['key'],
```

```
    config['cookie']['expiry_days'])
```

)

 **Step 3: Optional – Add Registration Link (if needed)**

If you want to allow user registration, the new version supports a registration view. Let me know if you'd like to add that.

Once you make these changes, your app should run smoothly again—whether locally or on Streamlit Cloud. Want me to help you add a README.md next to make your repo look sharp and professional?