midterm-1

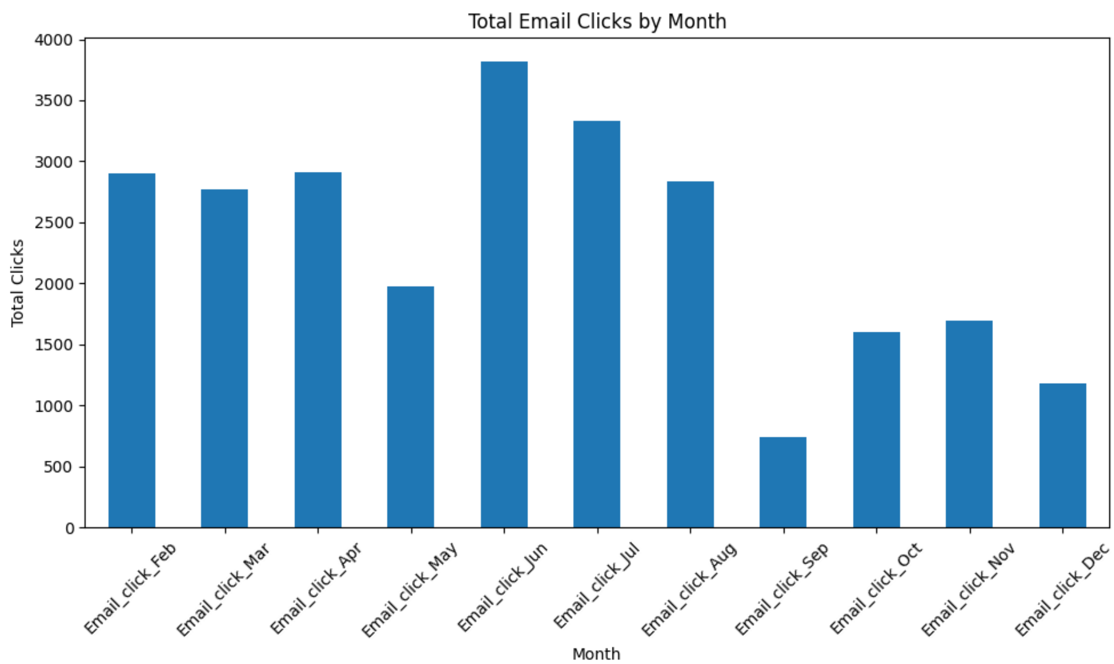
February 25, 2025

[37]:
```python
import pandas as pd
import numpy as np
import matplotlib.pyplot as plt
import seaborn as sns
df = pd.read_csv('student_enrollment_data.csv')
print(df.info())
print(df.describe())
print(df.columns)
```

```
<class 'pandas.core.frame.DataFrame'>
RangeIndex: 2575 entries, 0 to 2574
Data columns (total 23 columns):
 #   Column            Non-Null Count  Dtype
---  ------            --------------  -----
 0   Reference ID      2575 non-null   int64
 1   Sex               2575 non-null   object
 2   Student Type      2575 non-null   object
 3   Major 1           2561 non-null   object
 4   Major 2           2163 non-null   object
 5   IsPastorKid       2575 non-null   object
 6   IsAthlete         2575 non-null   object
 7   Race              2280 non-null   object
 8   Zip Code          2238 non-null   float64
 9   AddressState      2497 non-null   object
 10  Email_clicks_Jan  2575 non-null   int64
 11  Email_click_Feb   2575 non-null   int64
 12  Email_click_Mar   2575 non-null   int64
 13  Email_click_Apr   2575 non-null   int64
 14  Email_click_May   2575 non-null   int64
 15  Email_click_Jun   2575 non-null   int64
 16  Email_click_Jul   2575 non-null   int64
 17  Email_click_Aug   2575 non-null   int64
 18  Email_click_Sep   2575 non-null   int64
 19  Email_click_Oct   2575 non-null   int64
 20  Email_click_Nov   2575 non-null   int64
 21  Email_click_Dec   2575 non-null   int64
 22  Enrolled          2575 non-null   object
dtypes: float64(1), int64(13), object(9)
```

```
memory usage: 462.8+ KB
None
       Reference ID      Zip Code  Email_clicks_Jan  Email_click_Feb  \
count  2.575000e+03   2238.000000       2575.000000      2575.000000
mean   4.770369e+08  21705.662198          1.126602         1.126214
std    2.997064e+08  19710.784678          1.682758         1.651914
min    1.551550e+05    745.000000          0.000000         0.000000
25%    2.035259e+08  13750.750000          0.000000         0.000000
50%    4.717699e+08  14623.000000          0.000000         0.000000
75%    7.434801e+08  17884.250000          2.000000         2.000000
max    9.997468e+08  99577.000000         12.000000        14.000000

       Email_click_Mar  Email_click_Apr  Email_click_May  Email_click_Jun  \
count      2575.000000      2575.000000      2575.000000      2575.000000
mean          1.075340         1.131262         0.766214         1.483495
std           1.767947         1.954179         1.445630         2.240151
min           0.000000         0.000000         0.000000         0.000000
25%           0.000000         0.000000         0.000000         0.000000
50%           0.000000         0.000000         0.000000         0.000000
75%           2.000000         2.000000         1.000000         2.500000
max          15.000000        16.000000        13.000000        17.000000

       Email_click_Jul  Email_click_Aug  Email_click_Sep  Email_click_Oct  \
count      2575.000000      2575.000000      2575.000000      2575.000000
mean          1.293592         1.099806         0.286214         0.622913
std           1.864280         1.551417         0.709591         1.231049
min           0.000000         0.000000         0.000000         0.000000
25%           0.000000         0.000000         0.000000         0.000000
50%           0.000000         0.000000         0.000000         0.000000
75%           2.000000         2.000000         0.000000         1.000000
max          13.000000        11.000000        10.000000        15.000000

       Email_click_Nov  Email_click_Dec
count      2575.000000      2575.000000
mean          0.659417         0.458641
std           1.197366         0.977872
min           0.000000         0.000000
25%           0.000000         0.000000
50%           0.000000         0.000000
75%           1.000000         1.000000
max           9.000000        11.000000
Index(['Reference ID', 'Sex', 'Student Type', 'Major 1', 'Major 2',
       'IsPastorKid', 'IsAthlete', 'Race', 'Zip Code', 'AddressState',
       'Email_clicks_Jan', 'Email_click_Feb', 'Email_click_Mar',
       'Email_click_Apr', 'Email_click_May', 'Email_click_Jun',
       'Email_click_Jul', 'Email_click_Aug', 'Email_click_Sep',
       'Email_click_Oct', 'Email_click_Nov', 'Email_click_Dec', 'Enrolled'],
      dtype='object')
```

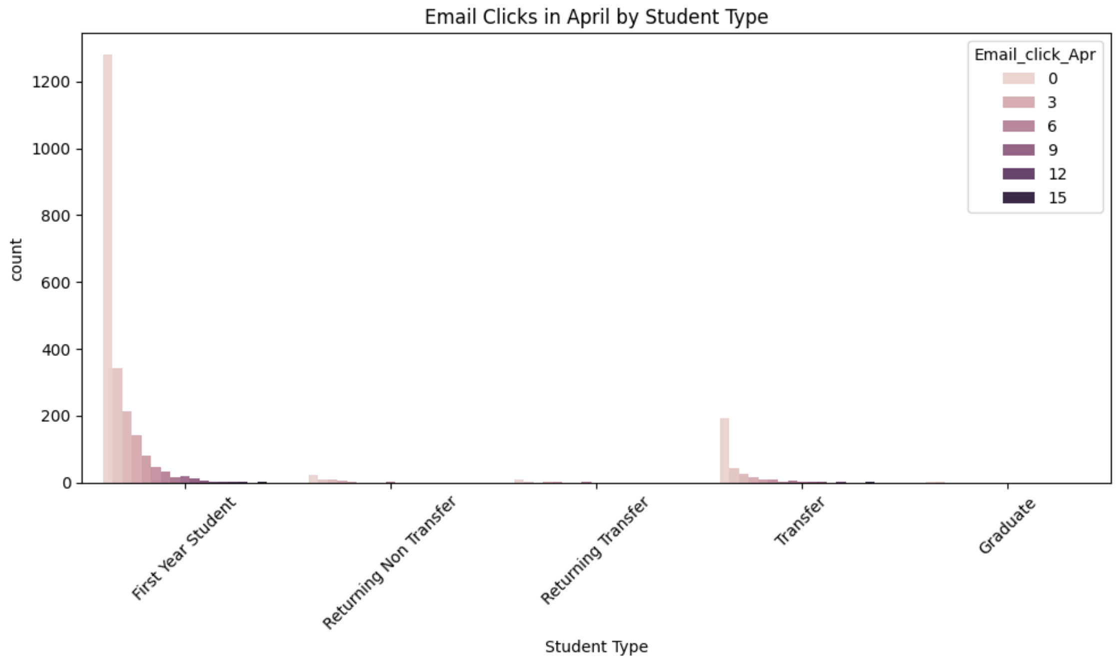
[38]:
```
email_columns = [col for col in df.columns if col.startswith('Email_click_')]
total_clicks = df[email_columns].sum()

plt.figure(figsize=(10,6))
total_clicks.plot(kind='bar')
plt.title('Total Email Clicks by Month')
plt.xlabel('Month')
plt.ylabel('Total Clicks')
plt.xticks(rotation=45)
plt.tight_layout()
plt.show()

print(f"Month with highest total clicks: {total_clicks.idxmax()}")
```

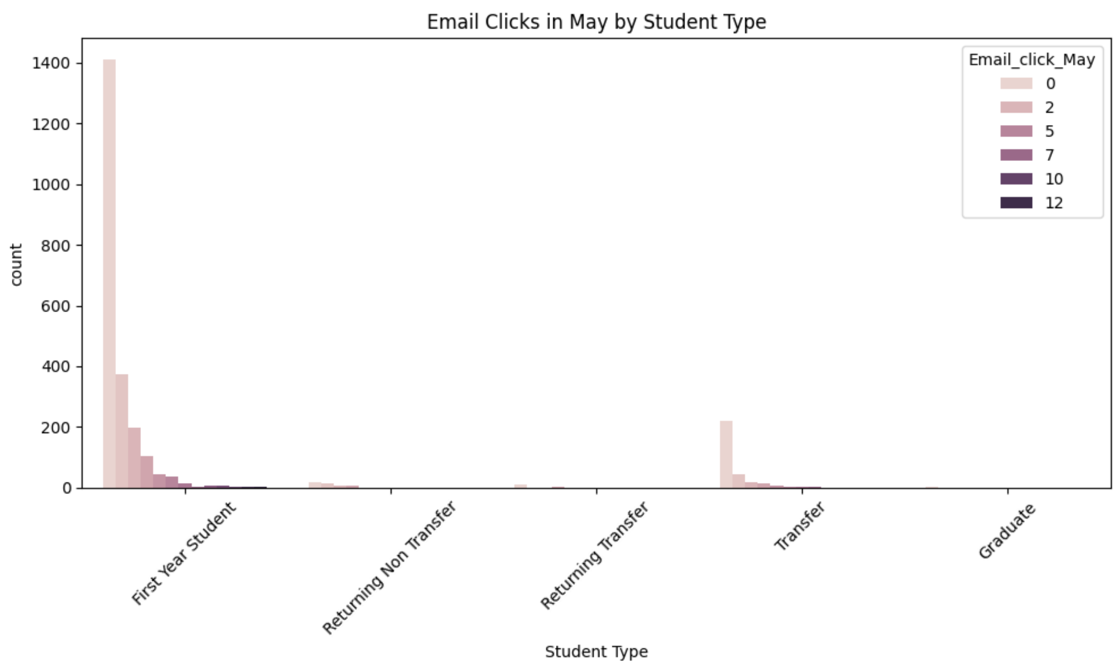



```
Month with highest total clicks: Email_click_Jun
```

[39]:
```
plt.figure(figsize=(10,6))
sns.countplot(x='Student Type', hue='Email_click_Apr', data=df)
plt.title('Email Clicks in April by Student Type')
plt.xticks(rotation=45)
plt.tight_layout()
plt.show()

plt.figure(figsize=(10,6))
sns.countplot(x='Student Type', hue='Email_click_May', data=df)
```

```
plt.title('Email Clicks in May by Student Type')
plt.xticks(rotation=45)
plt.tight_layout()
plt.show()
```





FInd percentage of students who clicked on emails

[40]:
```
def calculate_email_click_percentage(df, column):
    total_students = df.shape[0]
    email_clicks = df[column].sum()
    email_click_percentage = (email_clicks / total_students) * 100
    return email_click_percentage

email_click_percentage_apr = calculate_email_click_percentage(df,␣
 ↪'Email_click_Apr')
email_click_percentage_may = calculate_email_click_percentage(df,␣
 ↪'Email_click_May')

print(f'Percentage of students who clicked on the email in April:␣
 ↪{email_click_percentage_apr:.2f}%')
print(f'Percentage of students who clicked on the email in May:␣
 ↪{email_click_percentage_may:.2f}%')
```

```
Percentage of students who clicked on the email in April: 113.13%
Percentage of students who clicked on the email in May: 76.62%
```

Email clicks: The most email clicks were by first year students, while transfer students were second, but nowhere near the amount of clicks as the first years. I also discovered that April showed the highest average clicks per student. The story could be that first year students are more likely to engage with emails due to their unfamiliarity with college, while transfer students are less likely to engage with emails due to the fact they are more familiar with college processes.

Support for the Story:

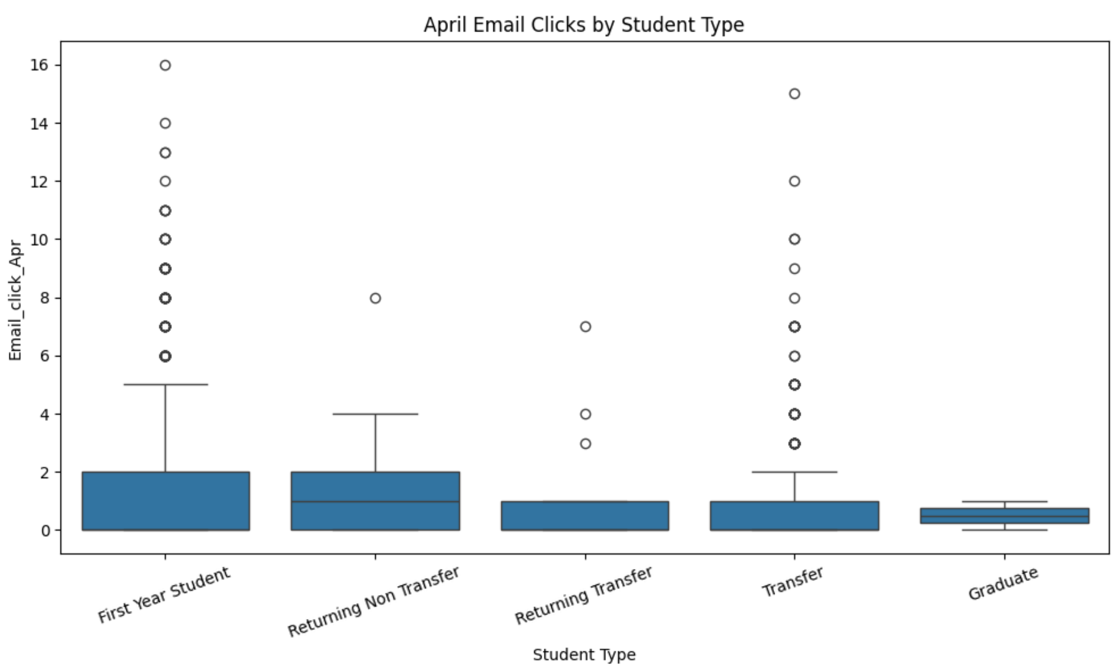
[41]:
```
df_first_year = df[df['Student Type'] == 'First Year Student']
df_transfer = df[df['Student Type'].str.contains('Transfer', na=False)]

email_click_percentage_first_year =␣
 ↪calculate_email_click_percentage(df_first_year, 'Email_click_Apr')
email_click_percentage_transfer = calculate_email_click_percentage(df_transfer,␣
 ↪'Email_click_Apr')

print(f'Percentage of first year students who clicked on the email in April:␣
 ↪{email_click_percentage_first_year:.2f}%')
print(f'Percentage of transfer students who clicked on the email in April:␣
 ↪{email_click_percentage_transfer:.2f}%')
```

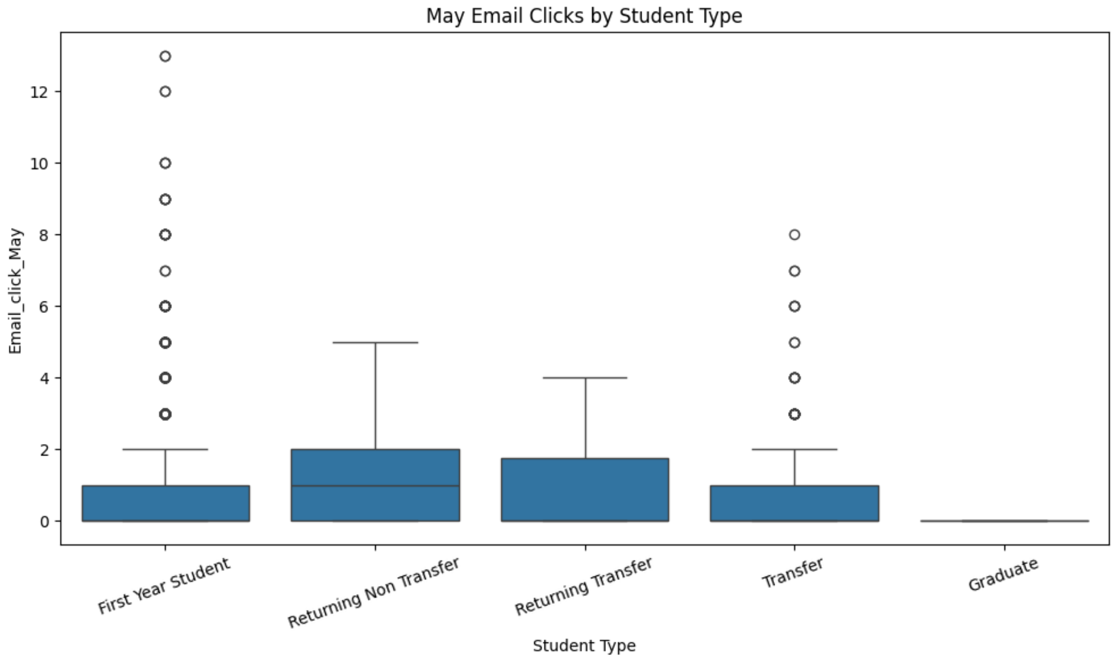
```
Percentage of first year students who clicked on the email in April: 113.55%
Percentage of transfer students who clicked on the email in April: 110.99%
```

Visuals:

[42]:
```
plt.figure(figsize=(10,6))
sns.boxplot(x='Student Type', y='Email_click_Apr', data=df)
```

```python
plt.title('April Email Clicks by Student Type')

plt.xticks(rotation=20)

plt.tight_layout()

plt.show()

plt.figure(figsize=(10,6))

sns.boxplot(x='Student Type', y='Email_click_May', data=df)

plt.title('May Email Clicks by Student Type')

plt.xticks(rotation=20)

plt.tight_layout()

plt.show()
```





Step 2:

steps I would take to clean the dataset before analysis: 1. Handle missing values 2. Handle outliers 3. Normalize or standardize numerical data 4. categorical data 6. Feature selection

correlation between emails clicked and enrollment

[43]:
```
df['Total_Clicks'] = df[['Email_click_Apr', 'Email_click_May',␣
↪'Email_click_Jun', 'Email_click_Jul', 'Email_click_Aug', 'Email_click_Sep',␣
↪'Email_click_Oct', 'Email_click_Nov', 'Email_click_Dec']].sum(axis=1)
df['Enrolled_Numeric'] = df['Enrolled'].map({'Yes': 1, 'No': 0})
correlation = df['Total_Clicks'].corr(df['Enrolled_Numeric'])
print(f"Correlation: {correlation}")
```

```
Correlation: 0.6308883699869055
```

This correlation shows the strength of the relationship between email engagement and enrollment. A positive correlation implies that higher email engagement is connected with a higher chance of enrollment.

[44]:
```
df['Total_Engagement'] = df['Total_Clicks'] / df[['Email_click_Apr',␣
↪'Email_click_May', 'Email_click_Jun', 'Email_click_Jul', 'Email_click_Aug',␣
↪'Email_click_Sep', 'Email_click_Oct', 'Email_click_Nov', 'Email_click_Dec']].
↪count(axis=1)



student_type_columns = [col for col in df.columns if col.startswith('Student␣
↪Type_')]
```

```
selected_features = student_type_columns + ['IsPastorKid', 'IsAthlete',␣
↪'Total_Clicks', 'Total_Engagement']
X = df[selected_features]
y = df['Enrolled_Numeric']
print("Available Types :", student_type_columns)
print("Selected features:", selected_features)
```

```
Available Types : []
Selected features: ['IsPastorKid', 'IsAthlete', 'Total_Clicks',
'Total_Engagement']
```

Which machine learning algorithms would I consider for this data set: - Logistic Regression: This algorithm is suitable for binary classification and it can handle both linear and non-linear relationships between features and the target. - K-nearest neighbors regression algorithm: Using this algorithm, we can find the nearest neighbors of a data point and make predictions based on the majority class or average value of those neighbors.

The model I chose is the K-nearest neighbor algorithm first

[45]:
```
import numpy as np
import pandas as pd
from sklearn.model_selection import train_test_split
from sklearn.preprocessing import StandardScaler
from sklearn.neighbors import KNeighborsRegressor

email_click_columns = [col for col in df.columns if col.
↪startswith('Email_click_')]
selected_features = ['IsPastorKid', 'IsAthlete'] + email_click_columns
X = df[selected_features].values
y = df['Enrolled'].map({'Yes': 1, 'No': 0}).values.astype(float)

X[:, 0] = (X[:, 0] == 'Yes').astype(int)
X[:, 1] = (X[:, 1] == 'Yes').astype(int)

X_train, X_test, y_train, y_test = train_test_split(X, y, test_size=0.2,␣
↪random_state=42)

scaler = StandardScaler()
X_train_norm = scaler.fit_transform(X_train)
X_test_norm = scaler.transform(X_test)
k = 5
knn_regressor = KNeighborsRegressor(n_neighbors=k)
knn_regressor.fit(X_train_norm, y_train)
y_pred = knn_regressor.predict(X_test_norm)

mse = np.mean((y_pred - y_test) ** 2)
print(f"Mean Squared Error: {mse:.4f}")
```

```
r_squared = 1 - (np.sum((y_test - y_pred) ** 2) / np.sum((y_test - np.
 ↪mean(y_test)) ** 2))
print(f"R-squared: {r_squared:.4f}")
```

```
---------------------------------------------------------------------------
ModuleNotFoundError                       Traceback (most recent call last)
Cell In[45], line 3
      1 import numpy as np
      2 import pandas as pd
----> 3 from sklearn.model_selection import train_test_split
      4 from sklearn.preprocessing import StandardScaler
      5 from sklearn.neighbors import KNeighborsRegressor

ModuleNotFoundError: No module named 'sklearn'
```

The K-nearest neighbor algorithm performed well with a Mean Squared Error of approximately 0.0929 and an R-squared value of approximately 0.6265. The model can accurately predict enrollment based on the given features.

Now I will test a Logistic Regression Algorithm

[ ]:
```
from sklearn.model_selection import train_test_split

X_train, X_test, y_train, y_test = train_test_split(X, y, test_size=0.2,␣
 ↪random_state=42)

from sklearn.linear_model import LogisticRegression

log_reg = LogisticRegression()

log_reg.fit(X_train, y_train)

y_pred = log_reg.predict(X_test)

from sklearn.metrics import accuracy_score

accuracy = accuracy_score(y_test, y_pred)

print(f"Accuracy: {accuracy:.4f}")

from sklearn.metrics import precision_score, recall_score, f1_score

precision = precision_score(y_test, y_pred)

recall = recall_score(y_test, y_pred)

f1 = f1_score(y_test, y_pred)
```

```python
print(f"Precision: {precision:.4f}")
```

The Logistic Regression algorithm performed well with an accuracy of 0.9029. The precision was also high, showing that the model was able to correctly classify the enrollment status of students.

To improve the model, I would consider adding more features such as age, grade, and other demographic information.

Discuss potential biases in the dataset that could affect the model's predictions. - Selection bias: The dataset may not represent the diverse range of students in the college.

Ethical considerations when using this dataset include:

1. Bias in data collection: Ensure that the dataset is representative of the diverse range of students in the college.
2. Bias in data interpretation: Analyze the data carefully to ensure that it is not biased towards any specific group.
3. Bias in data analysis: Perform analysis on the dataset without making assumptions.

Privacy concerns when using this dataset include:

1. Data: Make sure that the dataset is securely stored and protected from unauthorized access.
2. Data misuse: dont share sensitive information without consent.
3. Data privacy: Make sure that the students' privacy is protected.