📊 Engineering Faculty Attrition Analysis

Author: Syed Haider Saleem
Date: May 2025

🎯 Objective

This notebook analyzes attrition trends in the Faculty of Engineering, exploring relationships across departments, job types, age, and engagement-related factors.

❓ Key Questions

> 🧩 Over the last few years, staff in the Faculty of Engineering have been regularly leaving.
> 🧩 How has attrition trended over the years?
> 🧩 Are there any underlying patterns or predictors of attrition?
> 🧩 How does this compare with other faculties?

🛠️ Approach

We break this down into the following steps:

1️⃣ Data Exploration and Preparation

- Clean and understand the dataset
- Engineer new features required for the analysis

2️⃣ Focus on Faculty of Engineering

- Visualize key trends in attrition
- Use statistical and machine learning models to uncover patterns
- Directly answer the posed questions

3️⃣ Compare Across Faculties

- Extend the analysis to other faculties for benchmarking

🎁 Bonus: External Comparison

- Compare internal attrition rates with an external dataset (e.g. QS Rankings)
- Investigate if external factors may explain attrition trends

---

In [651…
```
# Loading Librarires

import pandas as pd
pd.set_option('display.max_columns', None)
import seaborn as sns
import matplotlib.pyplot as plt
import warnings
warnings.filterwarnings('ignore')
import numpy as np
from statsmodels.multivariate.manova import MANOVA
```

In [652…
```
# Loading data provided

df = pd.read_csv("../Data/test_dataset.csv")
```

Step 1: Data Exploration and Preparation

In [653…
```
print("Number of Rows : ", df.shape[0])
print("Number of Columns : ", df.shape[1])
```

```
Number of Rows :  3000
Number of Columns :  21
```

In [654…
```
# Lets see how our data looks

df.head(2)
```

Out[654…

	FirstName	LastName	StartDate	ExitDate	PositionTitle	Supervisor	ADEmail	Faculty	EmployeeStatus	EmployeeT
0	Aaden	Mercer	26-Jul-23	NaN	Senior Research Fellow	Victoria Jacobs	aaden.mercer@bilearner.com	Faculty of Medicine	Active	Part-T
1	Aaliyah	Watts	9-May-20	NaN	Senior Lecturer	Jared Whitehead	aaliyah.watts@bilearner.com	Faculty of Medicine	Active	Cont

Creating nessary columns

In [655…
```
# calculate column Year of service

# Convert to datetime (if not already)
df['StartDate'] = pd.to_datetime(df['StartDate'], errors='coerce', dayfirst=True)
df['ExitDate'] = pd.to_datetime(df['ExitDate'], errors='coerce', dayfirst=True)

# Calculate difference in days and convert to years
df['YearsWorked'] = (df['ExitDate'] - df['StartDate']).dt.total_seconds() / (365.25 * 24 * 60 * 60)

# Optional: Round to 2 decimal places
df['YearsWorked'] = df['YearsWorked'].round(2)

# Extract start year into a new column
df['StartYear'] = df['StartDate'].dt.year

# Extract extract year into a new column
```

```
df['ExitYear'] = df['ExitDate'].dt.year

df['AverageWellbeingScore'] = df[['Engagement Score', 'Satisfaction Score', 'Work-Life Balance Score']].mean(axis=1)
df['AverageWellbeingScore'] = df['AverageWellbeingScore'].round(2)

df['isLeaving'] = df['ExitDate'].notna()

# Convert DOB column to datetime if not already
df['DateOfBirth'] = pd.to_datetime(df['DOB'], errors='coerce')

# Extract the birth year
df['BirthYear'] = df['DateOfBirth'].dt.year
```

Faculty distribution

In [656…

```
print("Number of Unique Faculties : " , df['Faculty'].unique())

# How is our data distributed among faculties?

df.groupby('Faculty').count()['Employee ID']
```

```
Number of Unique Faculties :  ['Faculty of Medicine' 'Faculty of Arts' 'Faculty of Engineering']
```

Out[656…

```
Faculty
Faculty of Arts            591
Faculty of Engineering     904
Faculty of Medicine       1505
Name: Employee ID, dtype: int64
```

In [657…

```
# How many staff have exit our data?

print("Number of staff having exit date : ",df[df['ExitDate'].notna()].shape[0])
```

```
Number of staff having exit date :  1533
```

Key Findings

- We have 3000 rows of data with 21 columns
- We have 3 distict faculities in our dataset including (Faculty of Arts : 591), (Faculty of Engineering : 904), (Faculty of Medicine : 1505)

- Out of 3000 rows of data, we have exit dates for 1533 number of rows meaning these are the people who have left

Focus on Faculty of Engineering

Over the last few years, it appears that staff in Faculty of Engineering have been departing the Faculty regularly. How has attrition trended over the last few years?

We are going to use our data to check if the annual attrition rate for faculty is high. According to AHRI Annual employee turnover report for December 2023, the average turnover rate is 14% in australian workplaces. In universities its expected to be a bit higher. We cant find the exact number so we will take anything above 20% as high for our reference.

We will plot attrition rate for Engineering faculty for each year so we can compare the rate with whats national average to comment if its high or not.

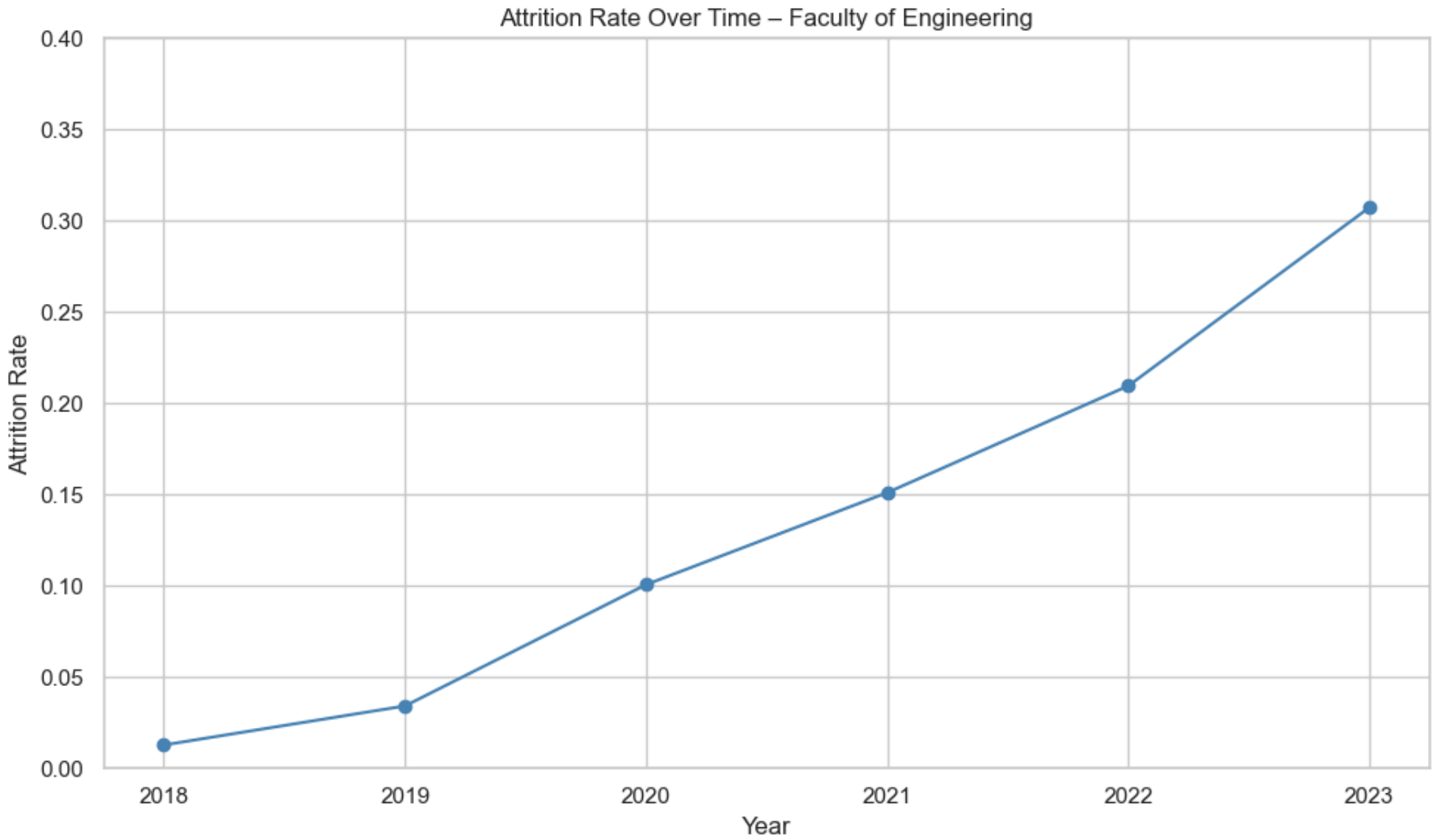
Attrition Rate = Number of people who left during the year / Average number of people during the year

Source: https://www.ahri.com.au/wp-content/uploads/AHRI-WorkOutlook-Report-Q1.pdf?_ga=2.60139561.548465833.1710108234-984794703.1663306512#:~:text=07-,6,19%25)%20%E2%80%93%20when%20recruiting%20staff.

In [658…

```
years = range(int(df['StartYear'].min()), int(df['ExitYear'].max()) + 1)

rows = []

for _, row in df.iterrows():
    if pd.notnull(row['StartYear']):
        start = int(row['StartYear'])
        end = int(row['ExitYear']) if pd.notnull(row['ExitYear']) else max(years)
        for y in range(start, end + 1):
            rows.append({
                'Faculty': row['Faculty'],
                'Year': y,
                'is_leaving_year': y == row['ExitYear']
            })

expanded_df = pd.DataFrame(rows)
```

```
# Total active people per year per faculty
total_by_year = expanded_df.groupby(['Faculty', 'Year']).size().rename('total')

# Leavers per year per faculty
leavers_by_year = expanded_df[expanded_df['is_leaving_year']].groupby(['Faculty', 'Year']).size().rename('leavers')

# Combine
combined = pd.concat([total_by_year, leavers_by_year], axis=1).fillna(0)
combined['attrition_rate'] = combined['leavers'] / combined['total']


# Filter for Faculty of Engineering
eng_df = combined.loc['Faculty of Engineering']

plt.figure(figsize=(10, 6))
plt.plot(eng_df.index, eng_df['attrition_rate'], marker='o', linestyle='-', color='steelblue')
plt.title('Attrition Rate Over Time – Faculty of Engineering')
plt.xlabel('Year')
plt.ylabel('Attrition Rate')
plt.ylim(0, 0.4)
plt.grid(True)
plt.tight_layout()
plt.show()
```



Conclusion:

This line chart confirms that the attrition rate has increased over time for the Faculty of Engineering.

Possible things to consider: Our dataset is currently limited, bigger data might reveal more consistent trends

Are there any underlying patterns to our attrition?

In order to answer this we need to consider few cases.

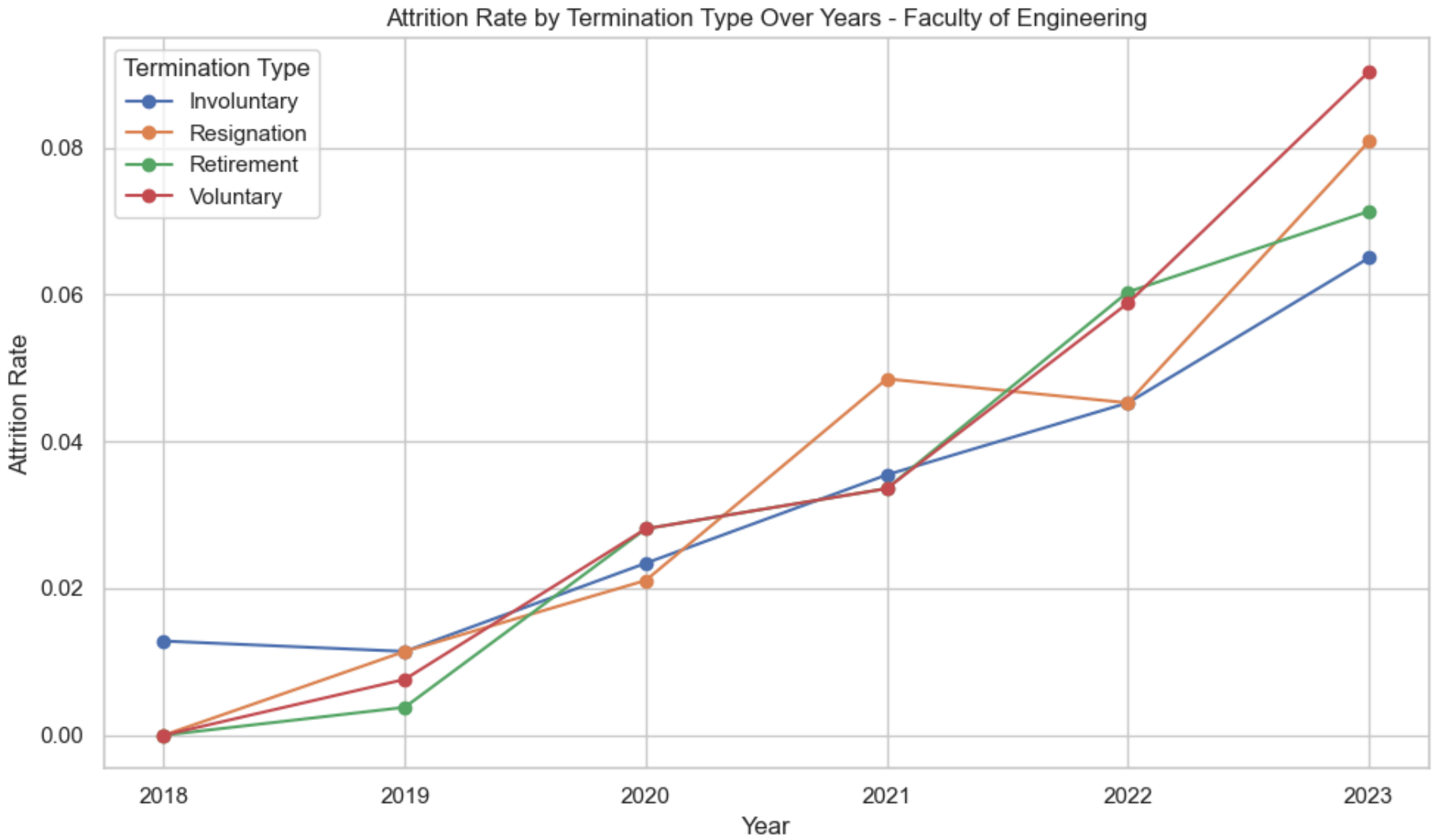
- Can we see a pattern if we see the actual Termination Types?
- Does employee contract type affects the behaviour?
- Does age at all affects the attrition rate?
- Are there particular job positions that have higher or lower attrition rate?
- How does employee rating effects the attrition rate?
- Most importantly, could engagement, satisfaction and work-life balance scores align with attrition trend? i.e people giving low scores should have more probably to leave

The code below looks at termination types

In [659…

```
years = range(int(df['StartYear'].min()), int(df['ExitYear'].max()) + 1)

rows = []

for _, row in df.iterrows():
    if pd.notnull(row['StartYear']):
        start = int(row['StartYear'])
        end = int(row['ExitYear']) if pd.notnull(row['ExitYear']) else max(years)
        for y in range(start, end + 1):
            rows.append({
                'Faculty': row['Faculty'],
                'Year': y,
                'is_leaving_year': y == row['ExitYear'],
                'TerminationType': row['TerminationType'] if pd.notnull(row['ExitYear']) and y == row['ExitYear'] else None
            })

expanded_df = pd.DataFrame(rows)

# Filter only the rows where the person actually left that year
leavers = expanded_df[expanded_df['is_leaving_year'] == True]

# Group by Year and TerminationType (only in Faculty of Engineering)
eng_leavers = leavers[leavers['Faculty'] == 'Faculty of Engineering']
```

```
# Group and count
leavers_by_type = eng_leavers.groupby(['Year', 'TerminationType']).size().reset_index(name='leavers')

# Total staff per year (from expanded_df — includes everyone working that year)
total_by_year = expanded_df[expanded_df['Faculty'] == 'Faculty of Engineering'].groupby('Year').size().rename('total')

# Merge and calculate attrition rate
result = leavers_by_type.merge(total_by_year, on='Year')
result['attrition_rate'] = result['leavers'] / result['total']

# Pivot for line chart
pivot_df = result.pivot(index='Year', columns='TerminationType', values='attrition_rate').fillna(0)

# Plot
pivot_df.plot(kind='line', marker='o', figsize=(10, 6))
plt.title('Attrition Rate by Termination Type Over Years - Faculty of Engineering')
plt.ylabel('Attrition Rate')
plt.xlabel('Year')
plt.grid(True)
plt.legend(title='Termination Type')
plt.tight_layout()
plt.show()
```

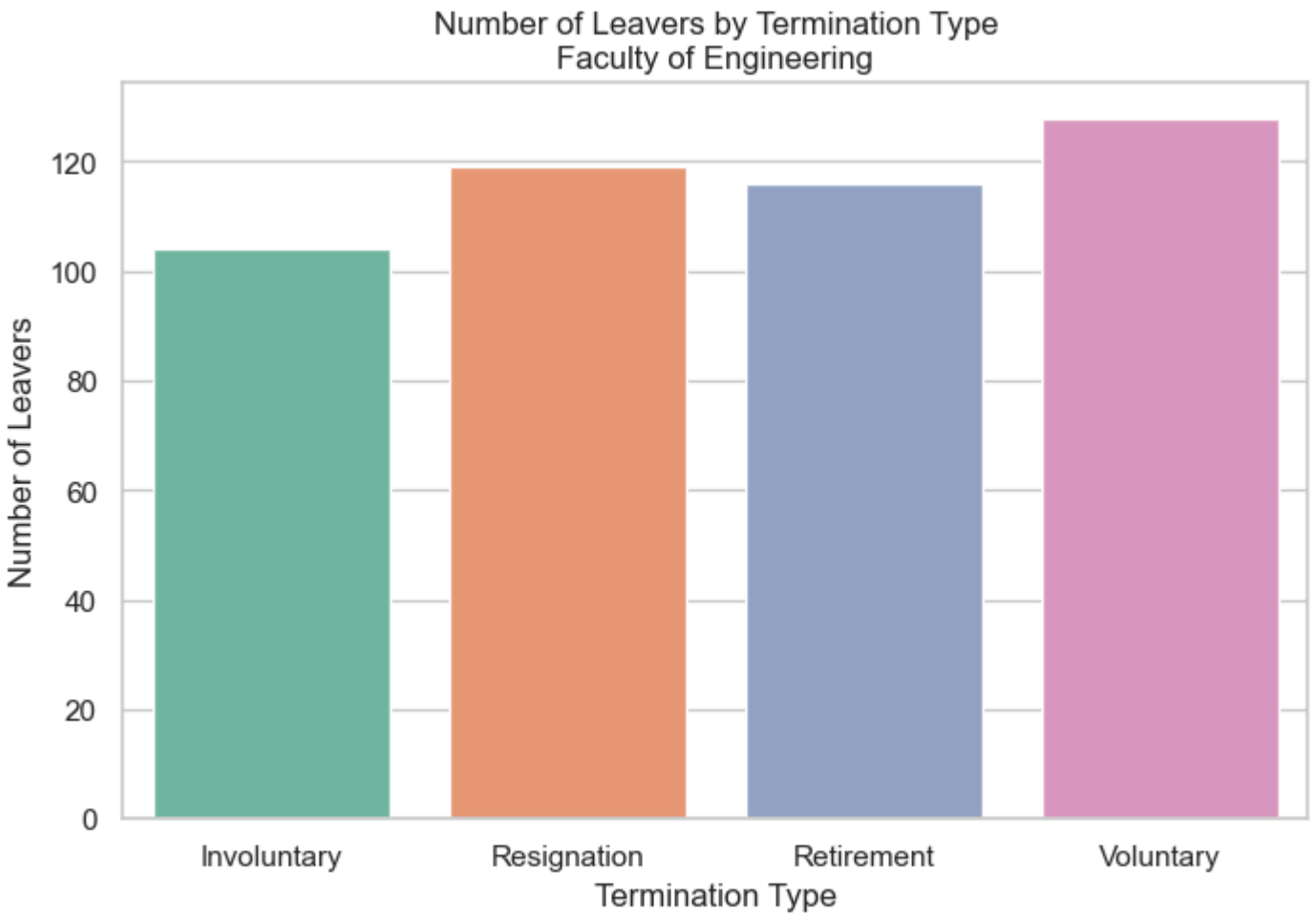


In [660…

```
# Step 1: Filter only Faculty of Engineering
eng_df = df[df['Faculty'] == 'Faculty of Engineering']

# Step 2: Total number of unique employees ever in Engineering
total_engineering_employees = eng_df['Employee ID'].nunique()

# Step 3: Get only rows with an exit year and termination type (i.e., people who left)
```

```
leavers = eng_df[pd.notnull(eng_df['ExitYear']) & pd.notnull(eng_df['TerminationType'])]

# Step 4: Count number of leavers by TerminationType
leavers_by_type = leavers.groupby('TerminationType')['Employee ID'].nunique().reset_index()
leavers_by_type = leavers_by_type.rename(columns={'Employee ID': 'leavers'})

print(leavers_by_type)

plt.figure(figsize=(8,5))
sns.barplot(data=leavers_by_type, x='TerminationType', y='leavers', palette='Set2')

plt.title('Number of Leavers by Termination Type\nFaculty of Engineering')
plt.ylabel('Number of Leavers')
plt.xlabel('Termination Type')

plt.show()
```

```
  TerminationType  leavers
0     Involuntary      104
1     Resignation      119
2      Retirement      116
3       Voluntary      128
```



Results: There seems to be higher voluntary terminations. Could be because of incentives for voluntary departures or type of employee

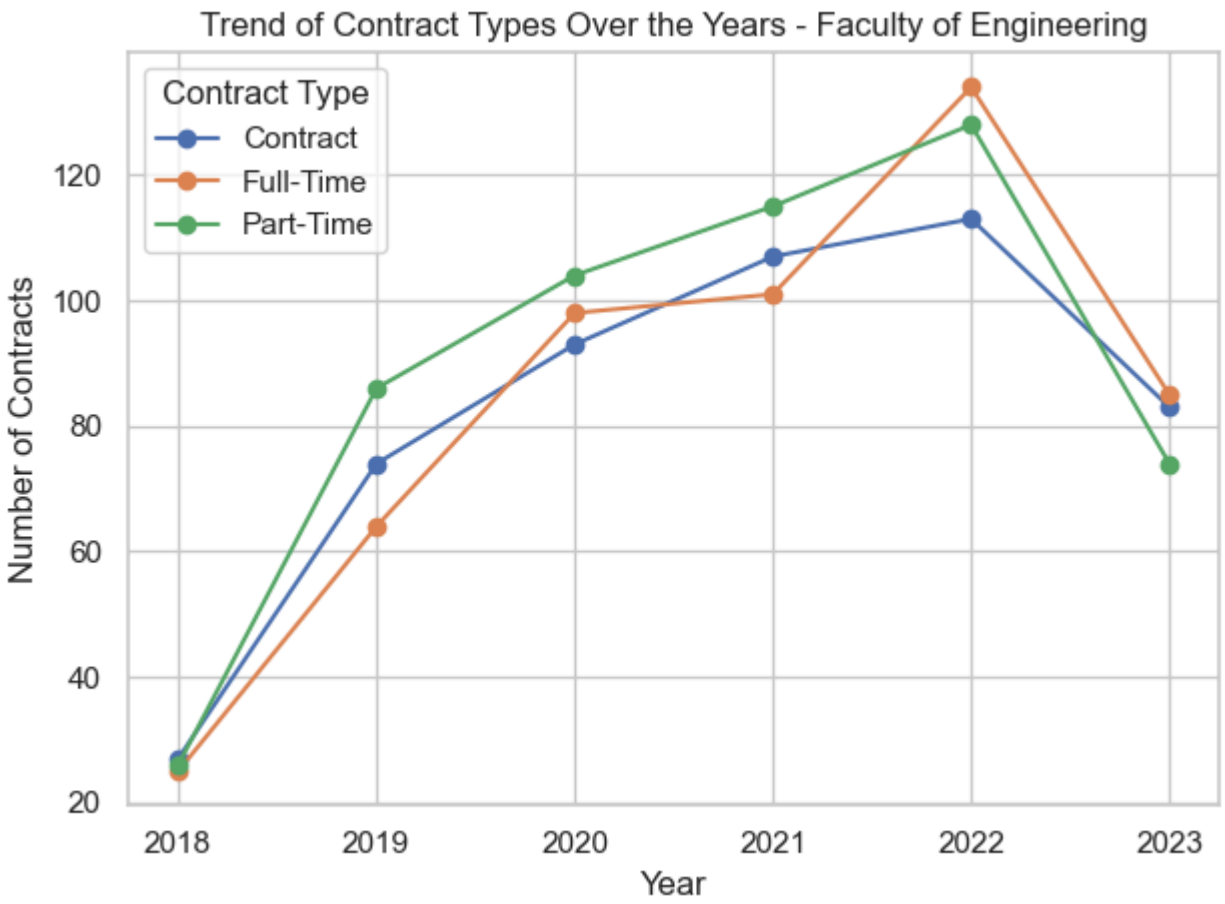
Now lets try to see if Employee Type (contract, full time) could give us information on why we have more voluntary terminations!

In [661…

```
# Filter for Faculty of Engineering before expanding
eng_df = df[df['Faculty'] == 'Faculty of Engineering']

rows = []
```

```
for _, row in eng_df.iterrows():  # use eng_df here instead of df
    if pd.notnull(row['StartYear']):
        start = int(row['StartYear'])
        end = int(row['ExitYear']) if pd.notnull(row['ExitYear']) else start  # or use latest year
        for y in range(start, end + 1):
            rows.append({
                'Year': y,
                'EmployeeType': row['EmployeeType']
            })

expanded_df = pd.DataFrame(rows)

# Group by Year and EmployeeType (assuming EmployeeType is contract type)
contract_trends = expanded_df.groupby(['Year', 'EmployeeType']).size().reset_index(name='count')

# Pivot for plotting
pivot_df = contract_trends.pivot(index='Year', columns='EmployeeType', values='count').fillna(0)

# Plot
plt.figure(figsize=(10,6))
pivot_df.plot(kind='line', marker='o')

plt.title('Trend of Contract Types Over the Years - Faculty of Engineering')
plt.ylabel('Number of Contracts')
plt.xlabel('Year')
plt.grid(True)
plt.legend(title='Contract Type')
plt.tight_layout()
plt.show()
```

```
<Figure size 1000x600 with 0 Axes>
```



no particular trend except that the number has decreased for all types in 2023

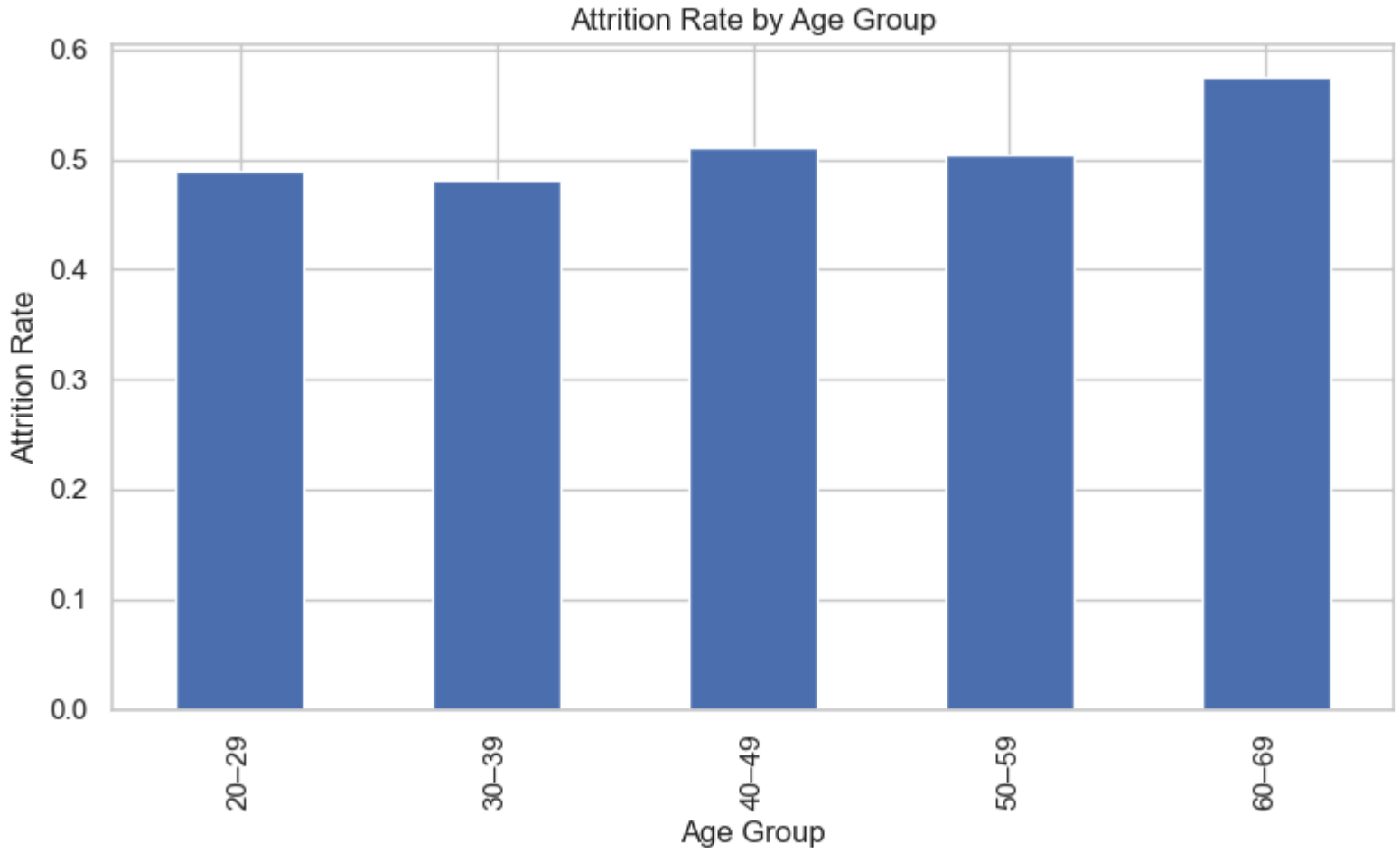
Next Step: Can we use age factor to uncover some hidden trend?

In [662…
```
# Age factor
```

In [663…
```
# Filter for Faculty of Engineering first
eng_df = df[df['Faculty'] == 'Faculty of Engineering'].copy()

reference_year = 2023
```

```
# Calculate BirthYear and AgeAtExitOrNow
eng_df['BirthYear'] = pd.to_datetime(eng_df['DateOfBirth']).dt.year
eng_df['AgeAtExitOrNow'] = np.where(
    pd.notnull(eng_df['ExitYear']),
    eng_df['ExitYear'] - eng_df['BirthYear'],
    reference_year - eng_df['BirthYear']
)

# Define age bins and labels
bins = [20, 30, 40, 50, 60, 70]
labels = ['20-29', '30-39', '40-49', '50-59', '60-69']

eng_df['AgeGroup'] = pd.cut(eng_df['AgeAtExitOrNow'], bins=bins, labels=labels, right=False)

# Total per age group
total_by_age = eng_df.groupby('AgeGroup').size()

# Leavers per age group (ExitYear not null)
leavers_by_age = eng_df[eng_df['ExitYear'].notnull()].groupby('AgeGroup').size()

# Combine and calculate attrition rate
attrition_by_age = pd.DataFrame({
    'total': total_by_age,
    'leavers': leavers_by_age
}).fillna(0)

attrition_by_age['attrition_rate'] = attrition_by_age['leavers'] / attrition_by_age['total']

print(attrition_by_age)

attrition_by_age['attrition_rate'].plot(kind='bar', figsize=(8,5), title='Attrition Rate by Age Group')
plt.ylabel('Attrition Rate')
plt.xlabel('Age Group')
plt.grid(True)
plt.tight_layout()
plt.show()
```

AgeGroup	total	leavers	attrition_rate
20–29	143	70	0.489510
30–39	154	74	0.480519
40–49	133	68	0.511278
50–59	123	62	0.504065
60–69	160	92	0.575000



Conclusion: Seems like a consistent trend across all age brackets, group having highest attrition rate is from 60-69 which could potentially be leaving due to retirements

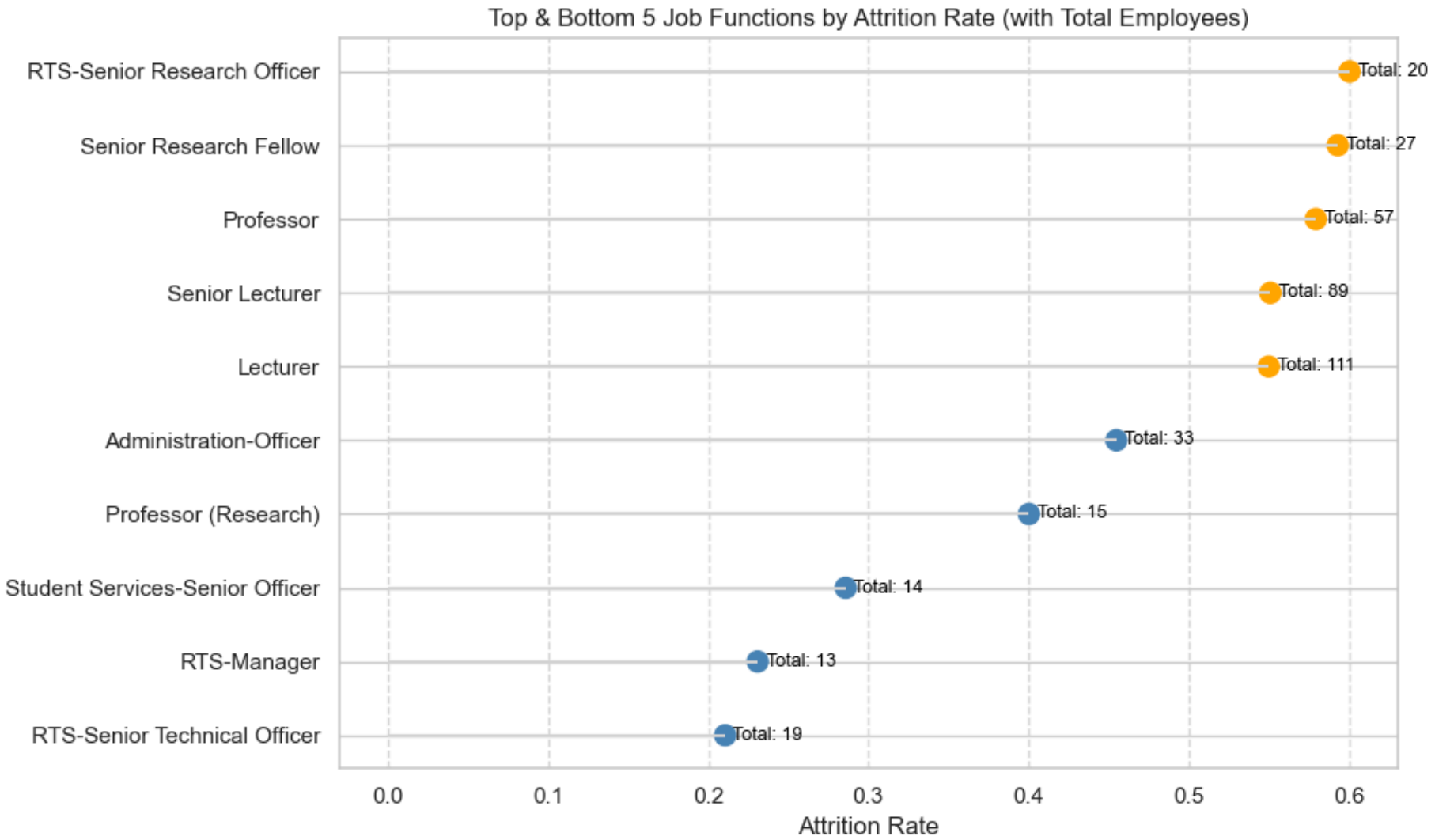
Lets try to find the Job Functions that have the highest and Lowest Attrition rate.

In [664…

```
# Filter for Engineering faculty only
eng_df = df[df['Faculty'] == 'Faculty of Engineering'].copy()

# Total employees per JobFunction (Engineering only)
total_per_job = eng_df.groupby('JobFunction').size().rename('total')

# Leavers per JobFunction (where ExitYear is not null)
leavers_per_job = eng_df[eng_df['ExitYear'].notnull()].groupby('JobFunction').size().rename('leavers')

# Combine total and leavers
job_attrition = pd.concat([total_per_job, leavers_per_job], axis=1).fillna(0)

# Filter to include only JobFunctions with total > 10
job_attrition = job_attrition[job_attrition['total'] > 10]

# Calculate attrition rate
job_attrition['attrition_rate'] = job_attrition['leavers'] / job_attrition['total']

# Get top 5 job functions by attrition rate
top_5_attrition_jobs = job_attrition.sort_values('attrition_rate', ascending=False).head(5)

# Get bottom 5 job functions by attrition rate
bottom_5_attrition_jobs = job_attrition.sort_values('attrition_rate', ascending=True).head(5)

# Combine top and bottom 5, remove duplicates, sort
combined_attrition_jobs = pd.concat([top_5_attrition_jobs, bottom_5_attrition_jobs])
combined_attrition_jobs = combined_attrition_jobs.drop_duplicates()
combined_attrition_jobs = combined_attrition_jobs.sort_values('attrition_rate')

print(combined_attrition_jobs)

# Assign colors
color_map = {
    job: 'orange' if job in top_5_attrition_jobs.index else 'steelblue'
    for job in combined_attrition_jobs.index
}
colors = [color_map[job] for job in combined_attrition_jobs.index]
```

```
# Plot: Lollipop chart
plt.figure(figsize=(10, 6))
plt.hlines(
    y=combined_attrition_jobs.index,
    xmin=0,
    xmax=combined_attrition_jobs['attrition_rate'],
    color='lightgray'
)
plt.scatter(
    combined_attrition_jobs['attrition_rate'],
    combined_attrition_jobs.index,
    c=colors,
    s=100
)

# Annotate total employee count
for i, (attrition, total) in enumerate(zip(
    combined_attrition_jobs['attrition_rate'],
    combined_attrition_jobs['total']
)):
    plt.text(attrition + 0.005, combined_attrition_jobs.index[i], f'Total: {int(total)}',
             va='center', fontsize=9, color='black')

# Labels and layout
plt.title('Top & Bottom 5 Job Functions by Attrition Rate (with Total Employees)')
plt.xlabel('Attrition Rate')
plt.grid(axis='x', linestyle='--', alpha=0.7)
plt.tight_layout()
plt.show()
```

JobFunction	total	leavers	attrition_rate
RTS-Senior Technical Officer	19	4.0	0.210526
RTS-Manager	13	3.0	0.230769
Student Services-Senior Officer	14	4.0	0.285714
Professor (Research)	15	6.0	0.400000
Administration-Officer	33	15.0	0.454545
Lecturer	111	61.0	0.549550
Senior Lecturer	89	49.0	0.550562
Professor	57	33.0	0.578947
Senior Research Fellow	27	16.0	0.592593
RTS-Senior Research Officer	20	12.0	0.600000



This chart highlights which job functions are at the highest risk (RTS-Senior Reseach Officer is at the top with 60% attrition rate)!

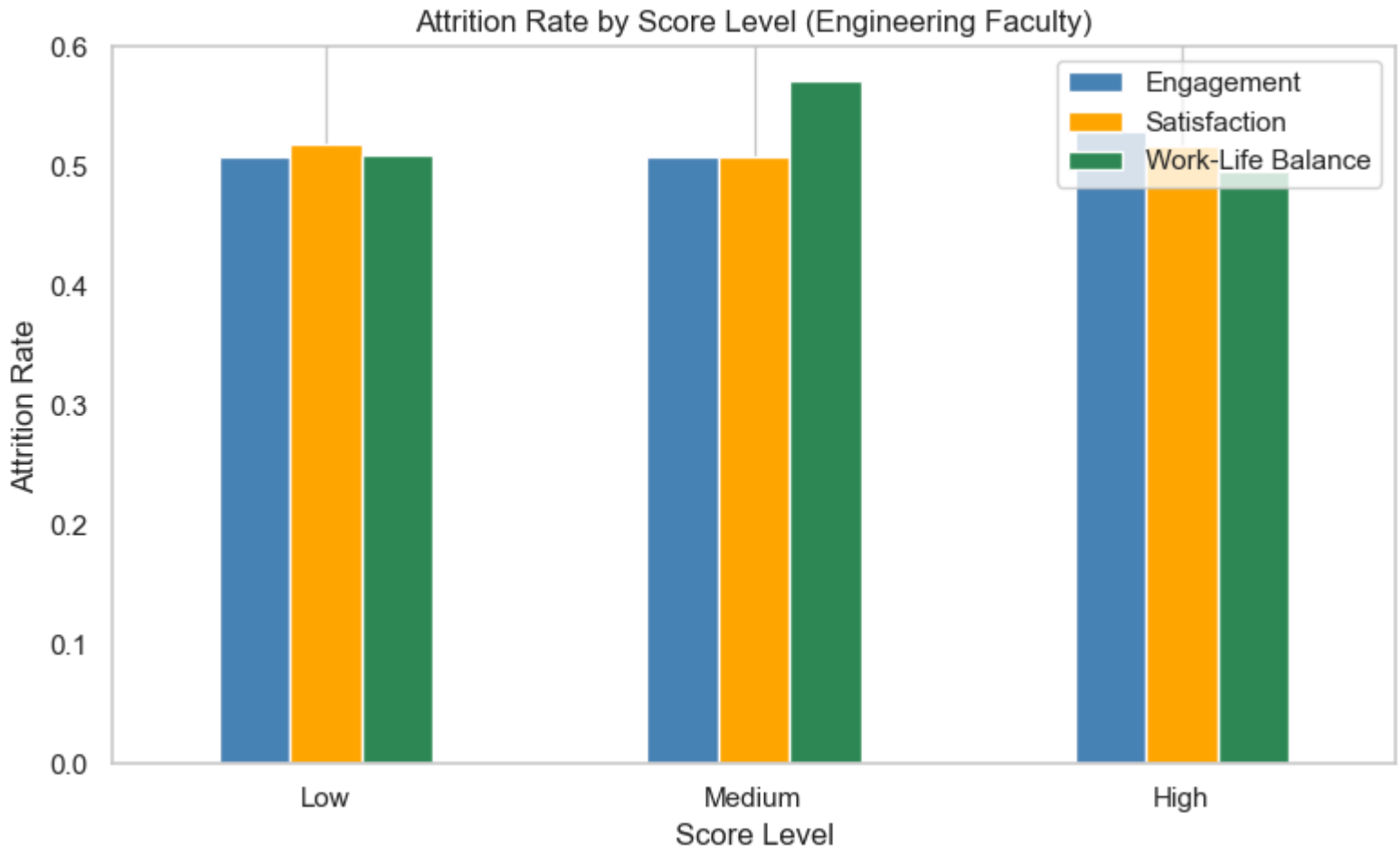
Lets use Scores (Satisfaction, Engagement and Work life balance) to see if there is a correlation of these with attrition levels

In [665…

```
# Filter for Faculty of Engineering
eng_df = df[df['Faculty'] == 'Faculty of Engineering'].copy()
```

```
# Ensure 'Leaver' column exists
eng_df['Leaver'] = eng_df['ExitYear'].notnull().astype(int)

# Define binning
bins = [0, 2, 3.5, 5]
labels = ['Low', 'Medium', 'High']

# Bin all three scores
eng_df['EngagementLevel'] = pd.cut(eng_df['Engagement Score'], bins=bins, labels=labels)
eng_df['SatisfactionLevel'] = pd.cut(eng_df['Satisfaction Score'], bins=bins, labels=labels)
eng_df['WorkLifeLevel'] = pd.cut(eng_df['Work-Life Balance Score'], bins=bins, labels=labels)

# Function to compute attrition rate per category
def compute_attrition(level_column):
    grouped = eng_df.groupby(level_column)['Leaver'].agg(['count', 'sum'])
    grouped.columns = ['total', 'leavers']
    grouped['attrition_rate'] = grouped['leavers'] / grouped['total']
    return grouped['attrition_rate']

# Compute attrition for all three
eng_attr = compute_attrition('EngagementLevel')
sat_attr = compute_attrition('SatisfactionLevel')
wlb_attr = compute_attrition('WorkLifeLevel')

# Combine into one DataFrame
combined = pd.DataFrame({
    'Engagement': eng_attr,
    'Satisfaction': sat_attr,
    'Work-Life Balance': wlb_attr
})

# Plot as grouped bar chart
combined.plot(kind='bar', figsize=(8, 5), color=['steelblue', 'orange', 'seagreen'])
plt.title('Attrition Rate by Score Level (Engineering Faculty)')
plt.ylabel('Attrition Rate')
plt.xlabel('Score Level')
plt.xticks(rotation=0)
plt.grid(axis='y')
```

```
plt.tight_layout()
plt.show()
```



In [666…

```
combined
```

Out[666...

	Engagement	Satisfaction	Work-Life Balance
Low	0.508108	0.519444	0.510086
Medium	0.508571	0.508475	0.571429
High	0.529248	0.517711	0.496000

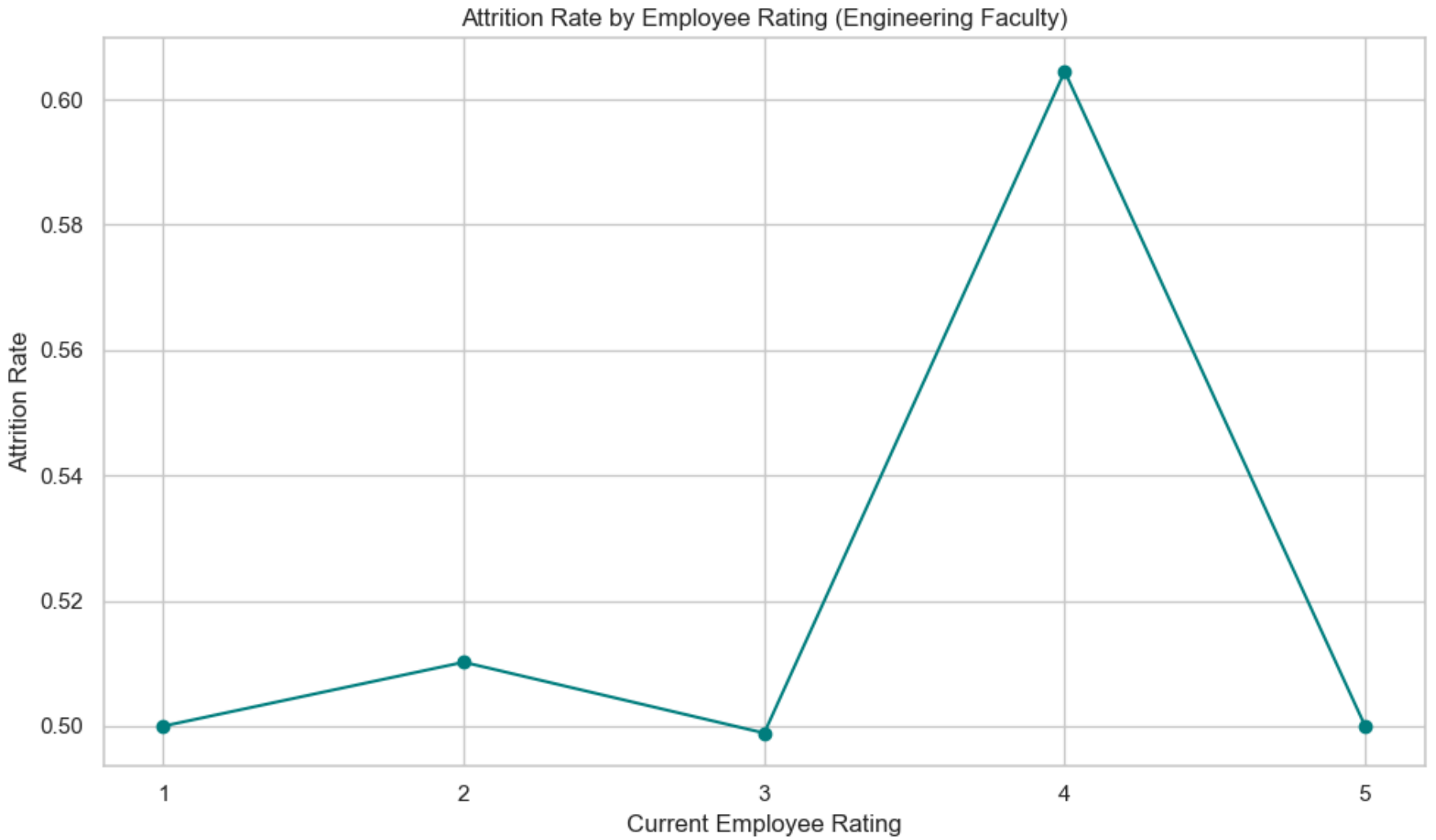
These measures does not seem to be deciding factor for attrition rate, but appears random in general. The only relaible information we see is, people giving high scores (3.5-5) on work-life balance tends to have a lower attrition rate then the rest

Lets now check how Current Employee Rating affects their attrition rate!

In [667...

```
# Filter data for Faculty of Engineering
eng_df = df[df['Faculty'] == 'Faculty of Engineering'].copy()

# Ensure 'Current Employee Rating' is numeric
eng_df['Current Employee Rating'] = pd.to_numeric(eng_df['Current Employee Rating'], errors='coerce')

# Drop rows with missing ratings
eng_df = eng_df.dropna(subset=['Current Employee Rating'])

# Group by rating
rating_grouped = eng_df.groupby('Current Employee Rating').agg(
    total=('Employee ID', 'count'),
    leavers=('ExitYear', lambda x: x.notna().sum())
)

# Calculate attrition rate
rating_grouped['attrition_rate'] = rating_grouped['leavers'] / rating_grouped['total']

# Reset index for plotting
rating_grouped = rating_grouped.reset_index()

# Plot
plt.figure(figsize=(10, 6))
plt.plot(rating_grouped['Current Employee Rating'], rating_grouped['attrition_rate'],
         marker='o', linestyle='-', color='teal')
plt.title('Attrition Rate by Employee Rating (Engineering Faculty)')
```

```
plt.xlabel('Current Employee Rating')
plt.ylabel('Attrition Rate')
plt.grid(True)
plt.xticks(rating_grouped['Current Employee Rating'])
plt.tight_layout()
plt.show()
```



Logistic Regression

Now that we have seen visualisations, lets confirm our results by ususing a machine learning model. What this does is give us a more systematic approach to test if any variable effects the outcome (A person leaving). We are going to use Logistic regression for it as it is a good baseline model for initial analysis

In [668…

```
import pandas as pd
import statsmodels.api as sm

# 1. Filter for Faculty of Engineering only
eng_df = df[df['Faculty'] == 'Faculty of Engineering'].copy()

# 2. Create binary target 'Leaver': 1 if ExitYear not null, else 0
eng_df['Leaver'] = eng_df['ExitYear'].notnull().astype(int)

# 3. Select predictors and target
predictors = ['GenderCode', 'EmployeeType', 'StaffType', 'BirthYear', 'Satisfaction Score', 'Engagement Score',
'Work-Life Balance Score', 'Current Employee Rating']
target = 'Leaver'

# 4. Prepare predictor dataframe with dummies for categorical variables
X = eng_df[predictors]
X = pd.get_dummies(X, drop_first=True)  # Convert categoricals to dummy vars

# 5. Convert all predictor columns to numeric (just in case)
X = X.apply(pd.to_numeric, errors='coerce')

# 6. Drop rows with any missing values in predictors or target
data = pd.concat([X, eng_df[target]], axis=1).dropna()

# 7. Separate cleaned predictors and target
X_clean = data.drop(columns=[target])
y_clean = data[target].astype(int)

# 8. Add constant for intercept
X_clean = sm.add_constant(X_clean)

# Convert bool columns to int
```

```
for col in X_clean.columns:
    if X_clean[col].dtype == 'bool':
        X_clean[col] = X_clean[col].astype(int)

# 9. Fit logistic regression model
model = sm.Logit(y_clean, X_clean).fit()

# 10. Print model summary
print(model.summary())
```

```
Optimization terminated successfully.
         Current function value: 0.686031
         Iterations 4
                           Logit Regression Results
==============================================================================
Dep. Variable:                 Leaver   No. Observations:                  904
Model:                          Logit   Df Residuals:                      894
Method:                           MLE   Df Model:                            9
Date:                Mon, 19 May 2025   Pseudo R-squ.:                0.009479
Time:                        15:05:32   Log-Likelihood:                -620.17
converged:                       True   LL-Null:                       -626.11
Covariance Type:            nonrobust   LLR p-value:                    0.2207
================================================================================================
                                 coef    std err          z      P>|z|      [0.025      0.975]
------------------------------------------------------------------------------------------------
const                         13.0538      7.368      1.772      0.076      -1.387      27.495
BirthYear                     -0.0067      0.004     -1.792      0.073      -0.014       0.001
Satisfaction Score             0.0128      0.048      0.267      0.790      -0.081       0.107
Engagement Score               0.0448      0.047      0.958      0.338      -0.047       0.136
Work-Life Balance Score       -0.0048      0.047     -0.102      0.919      -0.098       0.088
Current Employee Rating        0.0538      0.065      0.827      0.408      -0.074       0.181
GenderCode_Male               -0.1434      0.136     -1.058      0.290      -0.409       0.122
EmployeeType_Full-Time         0.0122      0.165      0.074      0.941      -0.311       0.335
EmployeeType_Part-Time         0.1599      0.166      0.966      0.334      -0.165       0.484
StaffType_Professional        -0.2844      0.143     -1.994      0.046      -0.564      -0.005
================================================================================================
```

We fitted a logistic regression model to predict employee attrition (leavers vs non-leavers) within the Faculty of Engineering using demographics, employment type, and engagement-related scores as predictors. Among the predictors, only being classified as a professional

staff type was associated with a statistically significant reduced likelihood of leaving as it has a negative coefficient. Other factors such as satisfaction, engagement, work-life balance, gender, and employee type did not have a significant effect in this sample.

Key Insights

- Attrition rate has increased over the years, atleast in the faculty of Engineering.
- Engineering had a slightly higher voluntary attrition rate.
- Age bracket does not seem to affect the attrition rate.
- The number of employee type (full time, part time and contract) seems to have been reduced in 2023
- RTS-Senior Research officer job function seems to have the highest attrition rate. Our data is limited so it could just be because of small data adding bias.
- Work Life balance seems to be slightly reliable indicator of wheather staff will stay longer, as people reporting high on work life balance have lower attrition rates.
- Employee's having 4 as Current Employee Rating tends to have much higher attrition rates.
- From our logistic regression model results, Professional staff has reduced likelihood of leaving

Part 3: Compare accross Faculties

In our dataset we have three different faculties, one being the Faculty of Engineering, we can only compare with two other Faculties that are : Faculty of Arts and Faculty of Medicine

Here are few things we would like to compare:

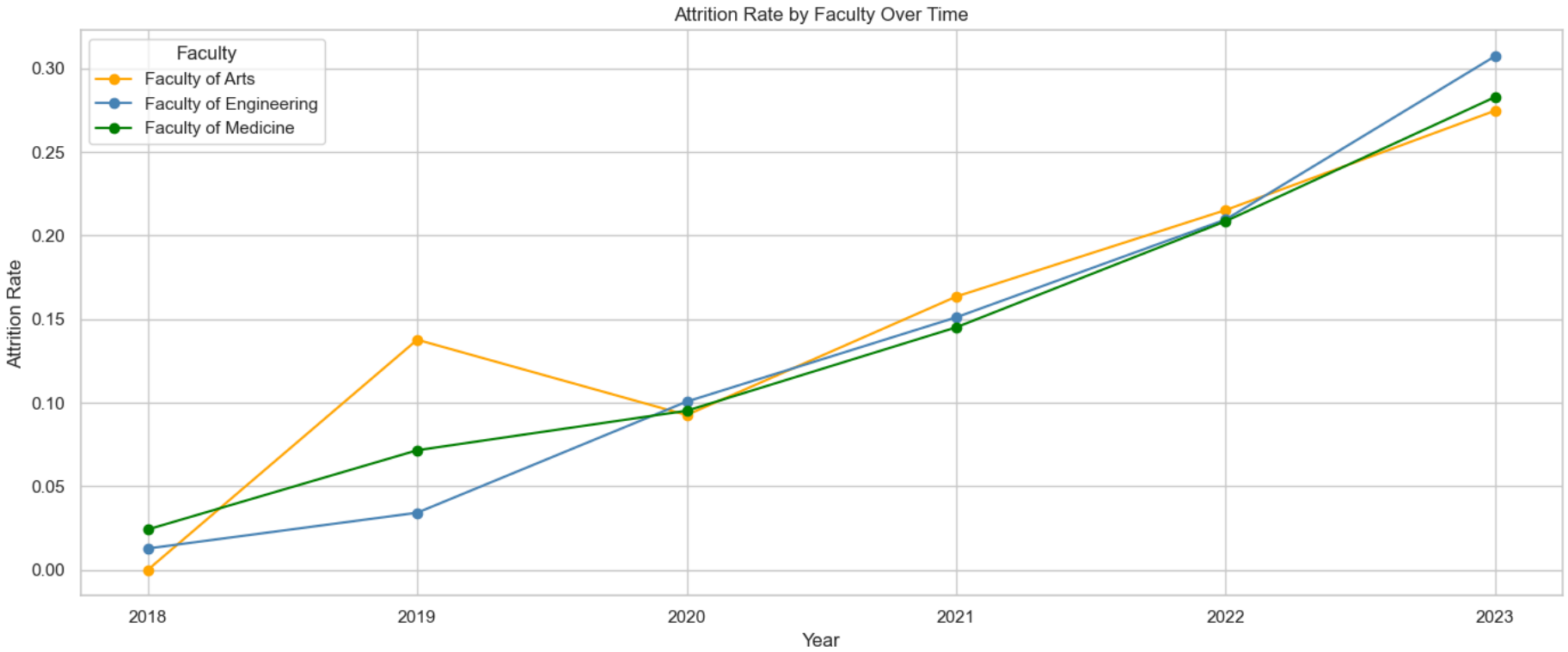
- Attrition rate comparision among faculties
- How does faculty of Engineering compare to others in terms of Engagement, Satisfaction and Worklife balance
- Lastly we will perform a test to see if results are significant or not

In [671…

```
years = range(int(df['StartYear'].min()), int(df['ExitYear'].max()) + 1)

rows = []

for _, row in df.iterrows():
```

```
    if pd.notnull(row['StartYear']):
        start = int(row['StartYear'])
        end = int(row['ExitYear']) if pd.notnull(row['ExitYear']) else max(years)
        for y in range(start, end + 1):
            rows.append({
                'Faculty': row['Faculty'],
                'Year': y,
                'is_leaving_year': y == row['ExitYear']
            })

expanded_df = pd.DataFrame(rows)

# Total active people per year per faculty
total_by_year = expanded_df.groupby(['Faculty', 'Year']).size().rename('total')

# Leavers per year per faculty
leavers_by_year = expanded_df[expanded_df['is_leaving_year']].groupby(['Faculty', 'Year']).size().rename('leavers')

# Combine
combined = pd.concat([total_by_year, leavers_by_year], axis=1).fillna(0)
combined['attrition_rate'] = combined['leavers'] / combined['total']


plot_df = combined.reset_index()

pivot_df = plot_df.pivot(index='Year', columns='Faculty', values='attrition_rate')

# Define custom colors, ensuring 'Faculty of Engineering' is blue
custom_colors = {
    'Faculty of Engineering': 'steelblue',  # or 'blue'
    'Faculty of Arts': 'orange',
    'Faculty of Medicine': 'green'
}

# Plot with consistent colors
pivot_df.plot(
    marker='o',
    figsize=(14, 6),
    color=[custom_colors[col] for col in pivot_df.columns]
)
```

```
plt.title('Attrition Rate by Faculty Over Time')
plt.xlabel('Year')
plt.ylabel('Attrition Rate')
plt.grid(True)
plt.legend(title='Faculty')
plt.tight_layout()
plt.show()
```



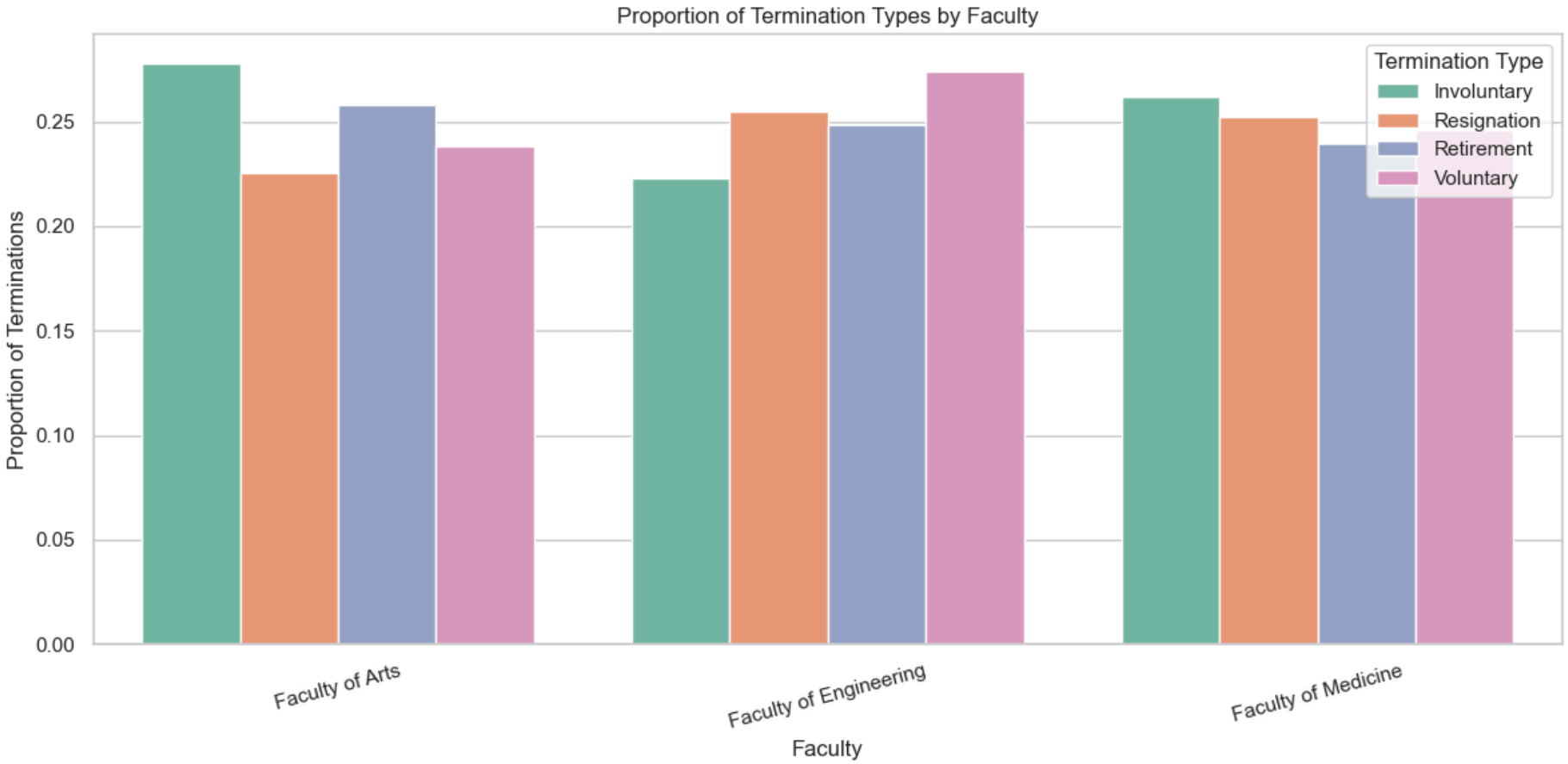
Analysis: It seems that attrition rate for Engineering was lower on average in 2018 and 2019 (colud be because of data bias) but it is slighly higher then other Faculties in 2023. The unusal attrition rate for all Faculties in 2019 could potentially be explained due to COVID 19

Now lets see which type of Terminations are more prevalent among Faculties

In [673…

```
# Filter to only people who exited
leavers_df = df[df['ExitYear'].notnull()]
```

```
# Count terminations per Faculty and TerminationType
termination_counts = leavers_df.groupby(['Faculty', 'TerminationType']).size().reset_index(name='count')

# Calculate total terminations per Faculty
faculty_totals = termination_counts.groupby('Faculty')['count'].transform('sum')

# Calculate percentage within each Faculty
termination_counts['percentage'] = termination_counts['count'] / faculty_totals

# Plot
plt.figure(figsize=(12, 6))
sns.barplot(
    data=termination_counts,
    x='Faculty',
    y='percentage',
    hue='TerminationType',
    palette='Set2'
)

plt.title('Proportion of Termination Types by Faculty')
plt.ylabel('Proportion of Terminations')
plt.xlabel('Faculty')
plt.xticks(rotation=15)
plt.legend(title='Termination Type')
plt.tight_layout()
plt.show()
```



Please Note : We are keeping in mind that there is uneven data for each faculty so we normalising data within each faculty and only displaying ratio of how much percent of particular termination type exists

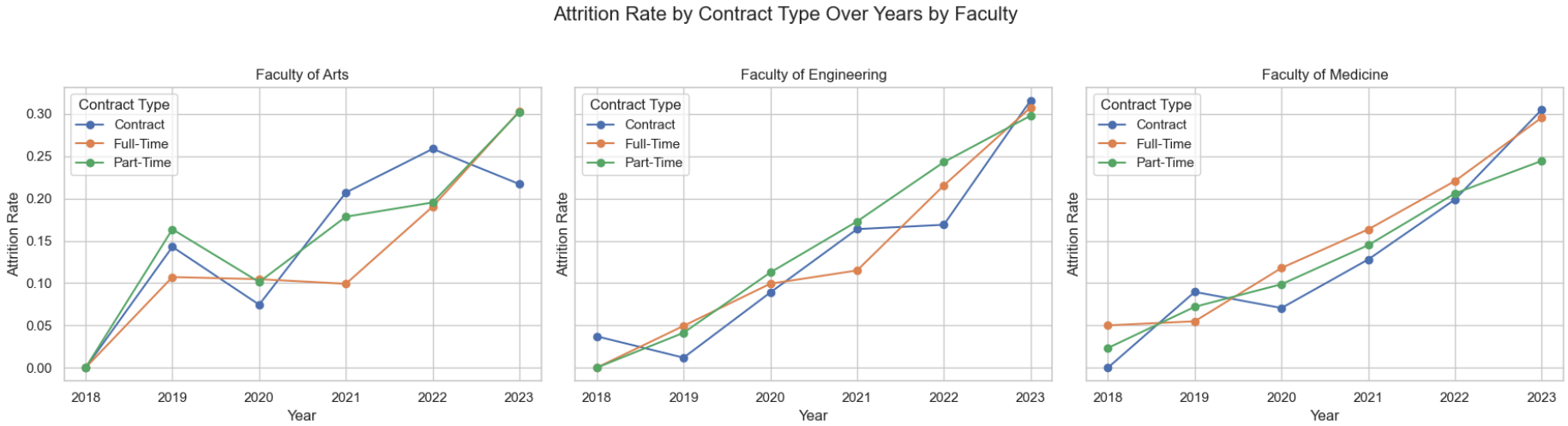
One Thing that stands out from the last charts is, for the other two facluties, involuntary type of termination is the highest occuring while in Faculty of Engineering Voluntary type of Termination is the highest. Could it be because of different types of contracts assigned in each faculty or it could just be that the Staisfaction, Engagement and Work life balance differ accorss faculties and could that be the reason for high attrition rate in Engineering

In [674…

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

```
# Assuming df has all faculties and these columns:
# 'Faculty', 'StartYear', 'ExitYear', 'EmployeeType'

# Expand the data to one row per employee per year employed
years = range(int(df['StartYear'].min()), int(df['ExitYear'].max()) + 1)

rows = []
for _, row in df.iterrows():
    if pd.notnull(row['StartYear']):
        start = int(row['StartYear'])
        end = int(row['ExitYear']) if pd.notnull(row['ExitYear']) else max(years)
        for y in range(start, end + 1):
            rows.append({
                'Faculty': row['Faculty'],
                'Year': y,
                'EmployeeType': row['EmployeeType'],
                'is_leaver': y == row['ExitYear']
            })

expanded_df = pd.DataFrame(rows)

# Calculate total employees per faculty, year, contract
total_per_group = expanded_df.groupby(['Faculty', 'Year', 'EmployeeType']).size().rename('total').reset_index()

# Calculate leavers per faculty, year, contract
leavers_per_group = expanded_df[expanded_df['is_leaver']].groupby(['Faculty', 'Year', 'EmployeeType']).size().rename('leavers'

# Merge totals and leavers
attrition_df = pd.merge(total_per_group, leavers_per_group, on=['Faculty', 'Year', 'EmployeeType'], how='left').fillna(0)

# Calculate attrition rate
attrition_df['attrition_rate'] = attrition_df['leavers'] / attrition_df['total']

# Faculties to plot
faculties = attrition_df['Faculty'].unique()

# Create subplots — one per faculty
fig, axes = plt.subplots(1, len(faculties), figsize=(6*len(faculties), 5), sharey=True)

if len(faculties) == 1:
    axes = [axes]  # make it iterable if just one plot
```

```
for ax, faculty in zip(axes, faculties):
    subset = attrition_df[attrition_df['Faculty'] == faculty]
    for contract_type in subset['EmployeeType'].unique():
        data = subset[subset['EmployeeType'] == contract_type]
        ax.plot(data['Year'], data['attrition_rate'], marker='o', label=contract_type)
    ax.set_title(faculty)
    ax.set_xlabel('Year')
    ax.set_ylabel('Attrition Rate')
    ax.grid(True)
    ax.legend(title='Contract Type')

plt.suptitle('Attrition Rate by Contract Type Over Years by Faculty', fontsize=16)
plt.tight_layout(rect=[0, 0, 1, 0.95])
plt.show()
```



These three sibplots for each faculty confirms that Employee type (i.e full time, part time or contract) does not have any significant corelation with attrition type

MANOVA Test

Now we would perform a test called MANOVA to check if Satisfaction score, Engagement Score and Work life balance scores differ signifcantly across different faculties. The purpose of performing this test is, if we find out that results are significant we can took a deep dive into

understanding if these factors can be the reason for different attrition levels across faculties

In [675…
```
df_manova = df.rename(columns={
    'Satisfaction Score': 'Satisfaction',
    'Work-Life Balance Score': 'WorkLifeBalance',
    'Engagement Score': 'Engagement'
})
```

In [676…
```
# Run MANOVA
maov = MANOVA.from_formula('Satisfaction + WorkLifeBalance + Engagement ~ Faculty', data=df_manova)
print(maov.mv_test())
```

```
                  Multivariate linear model
================================================================

----------------------------------------------------------------
       Intercept        Value  Num DF   Den DF    F Value  Pr > F
----------------------------------------------------------------
          Wilks' lambda 0.2777 3.0000 2995.0000 2597.1966 0.0000
         Pillai's trace 0.7223 3.0000 2995.0000 2597.1966 0.0000
 Hotelling-Lawley trace 2.6015 3.0000 2995.0000 2597.1966 0.0000
    Roy's greatest root 2.6015 3.0000 2995.0000 2597.1966 0.0000
----------------------------------------------------------------

----------------------------------------------------------------
          Faculty         Value  Num DF   Den DF  F Value Pr > F
----------------------------------------------------------------
            Wilks' lambda 0.9985 6.0000 5990.0000  0.7584 0.6027
           Pillai's trace 0.0015 6.0000 5992.0000  0.7586 0.6025
   Hotelling-Lawley trace 0.0015 6.0000 3991.5561  0.7583 0.6027
      Roy's greatest root 0.0010 3.0000 2996.0000  0.9995 0.3920
================================================================
```

There is no statistically significant difference in the combined levels of Satisfaction, Work-Life Balance, and Engagement across faculties (p = 0.60). This suggests that faculty membership does not meaningfully influence these employee experience metrics.

Analysis Result

Based on our tests performed, we can conclude that no particular feature (internally) available in data can explain the differences in attrition level among faculties. We might have to look at external features to try and understand this behaviour.

Bonus Part : Compare with Times Ranking data

In this part, we have taken small chunk of data manually from The times higher education Rankings, this is done to see if our university rankings for the department of Engineering are causing any effect on the attrition level of staff. This is a very basic level comparison only to show we can possiblly dive into more external factors to narrow down the cause of higher attrition. For now we are just looking at overall ranking points

In [677…
```
df_ranking = pd.read_excel("../Data/Engineering Times Ranking Monash.xlsx")
```

In [678…
```
df_ranking.head()
```

Out[678…

	Year	Overall Ranking Points
0	2018	63.9
1	2019	63.4
2	2020	60.4
3	2021	60.8
4	2022	64.9

In [679…
```
import pandas as pd

# Filter for Faculty of Engineering
eng_df = df[df['Faculty'] == 'Faculty of Engineering'].copy()

# Fill ExitYear for current employees with a future year (e.g., 2025)
eng_df['ExitYear'] = eng_df['ExitYear'].fillna(2025)

# Make sure EntryYear and ExitYear are integers
eng_df['EntryYear'] = pd.to_datetime(eng_df['StartDate']).dt.year
eng_df['EntryYear'] = eng_df['EntryYear'].astype(int)
```

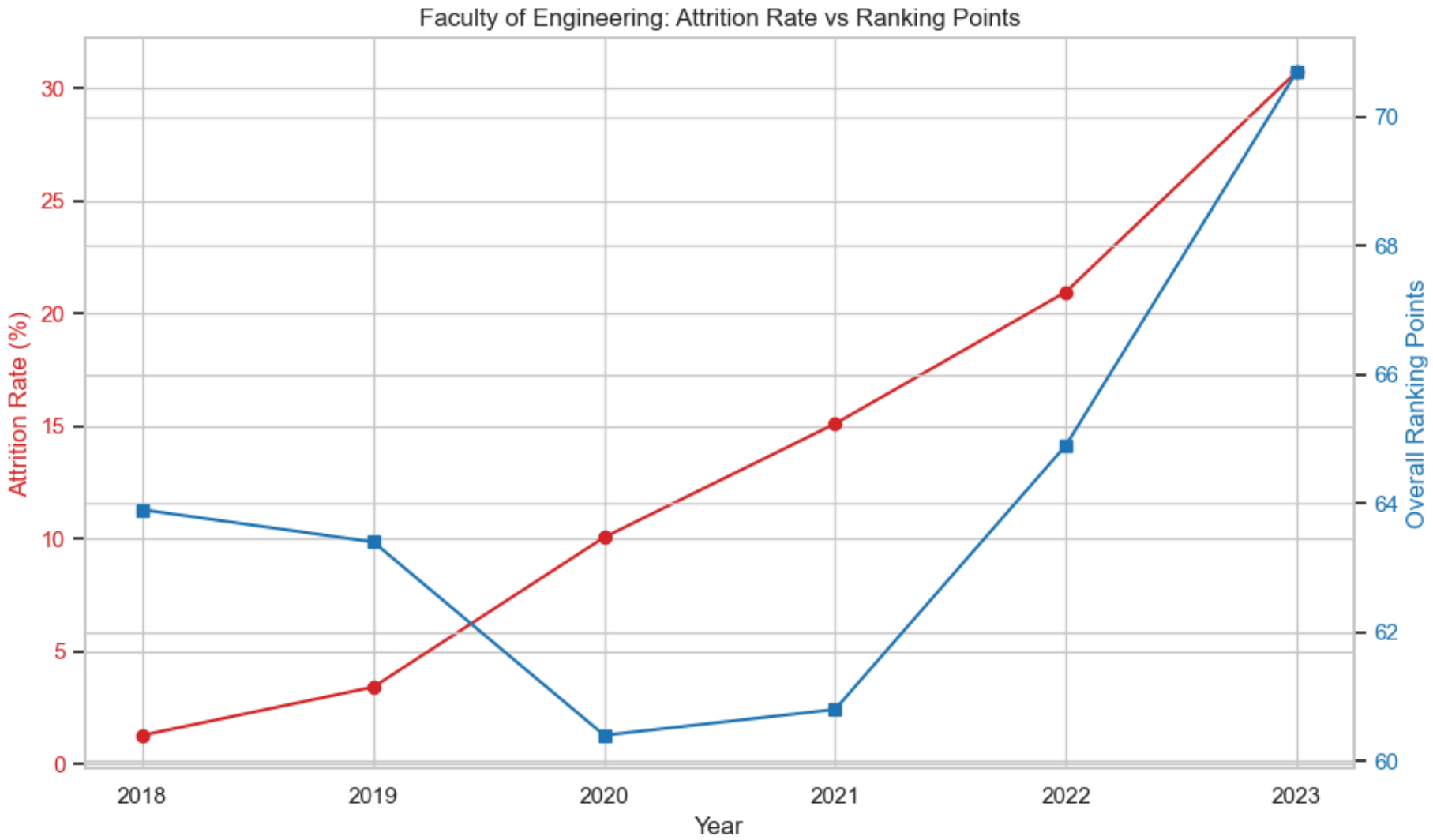
```
eng_df['ExitYear'] = eng_df['ExitYear'].astype(int)

# Define years to analyze (adjust as needed)
years = range(2018, 2024)

# Prepare list to collect results
rows = []

for year in years:
    # Employees who were working in this year
    total_staff = eng_df[(eng_df['EntryYear'] <= year) & (eng_df['ExitYear'] >= year)].shape[0]

    # Leavers in this year
    leavers = eng_df[eng_df['ExitYear'] == year].shape[0]

    # Calculate attrition rate
    attrition_rate = (leavers / total_staff) * 100 if total_staff > 0 else 0

    # Append result
    rows.append({'Year': year, 'Leavers': leavers, 'TotalStaff': total_staff, 'AttritionRate': attrition_rate})

# Create DataFrame
attrition_by_year = pd.DataFrame(rows)

# View result
print(attrition_by_year)
```

```
   Year  Leavers  TotalStaff  AttritionRate
0  2018        1          78       1.282051
1  2019        9         263       3.422053
2  2020       43         427      10.070258
3  2021       81         536      15.111940
4  2022      139         663      20.965309
5  2023      194         631      30.744849
```

In [680…

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

# Assuming you already have the two dataframes:
# 1. attrition_df with columns: ['Year', 'AttritionRate']
# 2. ranking_df with columns: ['Year', 'Overall Ranking Points']
```

```
# Merge both on 'Year'
merged_df = pd.merge(attrition_by_year[['Year', 'AttritionRate']], df_ranking, on='Year')

# Plot
fig, ax1 = plt.subplots(figsize=(10, 6))

# Plot Attrition Rate
color = 'tab:red'
ax1.set_xlabel('Year')
ax1.set_ylabel('Attrition Rate (%)', color=color)
ax1.plot(merged_df['Year'], merged_df['AttritionRate'], color=color, marker='o', label='Attrition Rate')
ax1.tick_params(axis='y', labelcolor=color)

# Create second y-axis
ax2 = ax1.twinx()
color = 'tab:blue'
ax2.set_ylabel('Overall Ranking Points', color=color)
ax2.plot(merged_df['Year'], merged_df['Overall Ranking Points'], color=color, marker='s', label='Ranking Points')
ax2.tick_params(axis='y', labelcolor=color)

# Title and grid
plt.title('Faculty of Engineering: Attrition Rate vs Ranking Points')
fig.tight_layout()
plt.grid(True)
plt.show()
```



Analysis: Again no particular trend could be established.

Conclusion Summary

- The attrition rate of Faculty of Engineering has been high over the last few years as compared to previous years (based on provided data)

- The attrition rate of Faculty of Engineering is higher than other two faculties (Arts, Medicine) in 2023. Another thing to notice is that Voluntary type of terminations have increased, so we might need to look why they are up
- From our detailed analysis we can conclude, varibales like age, or Staff Type (i.e Professional or Academic) or particular job functions does effect the attrition rate, but that does not explain the behaviour as of why the staff are leaving more frequently in the last few years. So we cannot conclude that there are any underlying patterns to our data that effects attrition rate.
- There is no major difference among faculties in terms of Scores for Satisfaction, Engagement or Work life balance which could be significant. However people reporting high on Work life balance tends to stay longer. This is something that can be leveraged.
- External factors could potentially impact attrition rates, the one factor we picked did not effect it however we need to explore more in this if we want to be certain of it.
- To get more reliable results we need to retest all our theories on bigger dataset.

In [ ]: