HR ANALYSIS REPORT

By

OJONUGWA WADA

Introduction

For this task, we were needed to perform a number of statistical tasks, ranging from central tendency and dispersion to confidence intervals and hypothesis testing, using one of the three datasets provided. I picked the HR dataset because it has a well-balanced mix of qualitative and numerical parameters relevant to study, such as employee engagement, performance score, and salary.

I began by cleaning and prepping the dataset, which involved altering data types, resolving missing information, adding variables such as tenure and age, and removing duplicate or extraneous entries. This prepared the data for accurate and meaningful analysis in each of the four jobs.

Data Description

The dataset contains precise information about employees in a firm, showcasing various elements of their employment, performance, and demographics. Understanding labor dynamics, employee engagement, compensation trends, and organizational effectiveness are all dependent on these traits. A full overview of the dataset's structure, major attributes, and potential uses is provided below.

The dataset has thirty-five columns (attributes) and numerous rows (records). While each row represents a unique individual, each column contains information on the employee's role, biography, and performance indicators. The dataset's combination of quantitative and qualitative attributes allows for a variety of statistical analyses, including measures of central tendency, dispersion, confidence intervals, and hypothesis testing.

Column Name	Data Type	Description	Quantitative/Qualitative
EmpID	Numerical	Unique identifier for each employee	Quantitative
MarriedID	Binary	Binary indicator for marital status (0 = Not Married, 1 = Married)	Qualitative
MaritalStatusID	Numerical	Numeric code for marital status	Qualitative
GenderID	Binary	Binary indicator for gender (0 = Female, 1 = Male)	Qualitative
EmpStatusID	Numerical	Numeric code for employment status	Qualitative
DeptID	Numerical	Numeric code for department	Qualitative
PerfScoreID	Numerical	Numeric code for performance score	Qualitative
FromDiversityJobFairID	Binary	Recruited from diversity job fair (0 = No, 1 = Yes)	Qualitative
Salary	Numerical	Employee's annual salary	Quantitative
Termd	Binary	Termination indicator (0 = Not Terminated, 1 = Terminated)	Qualitative
PositionID	Numerical	Numeric code for job position	Qualitative
Position	Categorical	Job title	Qualitative
State	Categorical	State where the employee works	Qualitative
Zip	Numerical	ZIP code of employee's location	Qualitative
DOB	Date	Date of birth	Qualitative
Sex	Categorical	Biological sex	Qualitative
MaritalDesc	Categorical	Description of marital status (e.g., Single)	Qualitative
CitizenDesc	Categorical	Citizenship status	Qualitative
HispanicLatino	Binary	Hispanic/Latino ethnicity indicator (Yes/No)	Qualitative
RaceDesc	Categorical	Race/ethnicity description	Qualitative
DateofHire	Date	Hiring date	Qualitative
DateofTermination	Date	Termination date (if applicable)	Qualitative
TermReason	Categorical	Reason for termination	Qualitative

EmploymentStatus	Categorical	Current employment status	Qualitative
Department	Categorical	Department where the employee works	Qualitative
ManagerName	Categorical	Name of the employee's manager	Qualitative
ManagerID	Numerical	Unique identifier for the manager	Qualitative
RecruitmentSource	Categorical	Source through which the employee was recruited	Qualitative
PerformanceScore	Categorical	Employee performance rating (e.g., Fully Meets, Exceeds)	Qualitative
EngagementSurvey	Numerical	Employee engagement survey score	Quantitative
EmpSatisfaction	Numerical	Self-reported satisfaction score	Quantitative
SpecialProjectsCount	Numerical	Number of special projects handled	Quantitative
LastPerformanceReview_Date	Date	Date of last performance review	Qualitative
DaysLateLast30	Numerical	Number of days late in the last 30 days	Quantitative
Absences	Numerical	Total recorded absences	Quantitative

This dataset offers a rich source of information for conducting data-driven analyses to support HR decision-making and organizational strategy.

Data Preprocessing

To ensure accuracy and consistency, the HR dataset was cleaned before analysis. While numerical columns such as EngagementSurvey and EmpSatisfaction were imputed using the median, missing values in important variables such as ManagerName, Position, and Salary were fixed by eliminating the affected rows. Categorical variables were assigned to suitable categories, and date data were transformed to datetime format for feature engineering. Two more factors, Age (derived from DOB) and Tenure (derived from Date of Hire), were included to improve the study. After deleting duplicates and extraneous columns, the dataset was simplified for statistical tasks including central tendency, dispersion, confidence intervals, and hypothesis testing. The completed dataset may now be evaluated.

📝 Task 1 - Measures of Central Tendency Report

In this task, I used metrics of central tendency mean, median, and mode to three carefully chosen qualities from the HR dataset, each inside a distinct scenario that illustrates basic HR issues.

Scenario 1: Mean Salary by Department

I chose Salary to look at the average across departments. Salaries are a key economic indicator that determines employee retention and motivation. By analyzing average pay across departments, I was able to identify compensation trends and disparities. The data showed that the Executive Office had the greatest mean remuneration, while Production had the lowest. This knowledge is important in determining fair salary and budget alignment across roles.

SCENARIO 1: MEAN SALARY BY DEPARTMENT

Purpose:This analysis calculates the average (mean) salary across different departments. It helps HR and management evaluate how compensation is distributed and whether certain departments are underpaid or overpaid relative to others.

Rationale:Understanding salary distribution by department can uncover disparities, inform compensation strategies, and assist with budgeting and workforce planning.

In [13]:
```
# Group the dataset by 'Department' and compute the mean salary
mean_salary_by_dept = df.groupby('Department', as_index=False)['Salary'].mean()

# Rename the column for clarity
mean_salary_by_dept.rename(columns={'Salary': 'Mean Salary'}, inplace=True)

# Sort the result for better readability
mean_salary_by_dept.sort_values(by='Mean Salary', ascending=False, inplace=True)

# Print output
display(
    mean_salary_by_dept.style
    .set_caption("Mean Salary by Department")
    .format({'Mean Salary': '{:,.2f}'})
)
```

Mean Salary by Department

	Department	Mean Salary
1	Executive Office	250,000.00
2	IT/IS	97,064.64
5	Software Engineering	94,989.45
0	Admin Offices	71,791.89
4	Sales	69,061.26
3	Production	59,953.55

Scenario 2: Median Age by Performance Score

In this scenario, I used age to get the medians for each performance score category. Outliers do not alter the median age, which provides a more accurate portrayal of the average employee in each group. This helps to uncover age-related performance patterns. Individuals who "fully meet" or "exceed" expectations were frequently in their early forties, showing that experience and maturity may have a positive influence on performance.

SCENARIO 2: MEDIAN AGE BY PERFORMANCE SCORE

Purpose:This analysis calculates the median age of employees within each performance score category. It gives insight into generational trends across different performance levels. ¶

Rationale:Median is used (instead of mean) because it's resistant to outliers and better reflects the 'typical' employee age. Identifying age patterns in high or low performers can help with training, mentoring, and workforce planning.

In [14]:
```
# Group by performance score and calculate the median age
median_age_by_perf = df.groupby('PerformanceScore', as_index=False)['Age'].median()

# Rename the column for clarity
median_age_by_perf.rename(columns={'Age': 'Median Age'}, inplace=True)

# Sort by median age for clearer analysis
median_age_by_perf.sort_values(by='Median Age', ascending=False, inplace=True)

display(
    median_age_by_perf.style
    .set_caption("Median Age by Performance Score")
    .format({'Median Age': '{:.2f}'})
)
```

Median Age by Performance Score

	PerformanceScore	Median Age
0	Exceeds	42.00
2	Needs Improvement	42.00
1	Fully Meets	41.00
3	PIP	40.00

Scenario 3: Mode of Engagement Survey by Position

Finally, I used Engagement Survey Scores to determine the mode for each work role. Engagement levels reflect both employee happiness and organizational health. Identifying the most common engagement levels by job indicates which roles are frequently more satisfied or disengaged, allowing for more targeted HR interventions.

In [15]:
```
# Define a custom function to calculate the mode of a series
def calculate_mode(series):
    """Return the mode of a Series or NaN if the mode cannot be determined."""
    mode_result = stats.mode(series, keepdims=True)
    return mode_result.mode[0] if mode_result.count[0] > 0 else np.nan

# Apply the mode function for each Position
mode_engagement_by_pos = df.groupby('Position')['EngagementSurvey'].apply(calculate_mode).reset_index()
mode_engagement_by_pos.rename(columns={'EngagementSurvey': 'Mode of Engagement Survey'}, inplace=True)

# Sort for readability
mode_engagement_by_pos.sort_values(by='Mode of Engagement Survey', ascending=False, inplace=True)


display(
    mode_engagement_by_pos.style
    .set_caption("Mode of Engagement Survey by Position")
    .format({'Mode of Engagement Survey': '{:.2f}'})
)
```

Mode of Engagement Survey by Position

	Position	Mode of Engagement Survey
28	Sr. Accountant	5.00
22	Production Technician II	5.00
12	IT Director	5.00
21	Production Technician I	5.00
17	Network Engineer	5.00
29	Sr. DBA	4.96
7	Data Architect	4.94
18	President & CEO	4.83

These scenarios were chosen for their practical relevance and ability to highlight important workforce insights.

📑 Task 2 - Measure of Spread/Dispersion Report

In Task 2, I focused on key HR variables that best represented assessments of dispersion range, variance, and standard deviation. Understanding data distribution is crucial for detecting anomalies, outliers, and underlying trends in employee behavior and organizational structure.

Scenario 1: Salary Spread by Department

Salary was chosen again to see how remuneration differed by department. Salary variations may suggest inequity or ineffective compensation techniques. A large standard deviation in locations like administrative offices shows pay anomalies, but a low dispersion in the executive office indicates a consistent wage.

In [17]:
```
# Group by department and calculate salary spread metrics
stats = df.groupby('Department')['Salary'].agg(
    Range=lambda x: x.max() - x.min(),
    Variance='var',
    StdDev='std'
).reset_index()

# Rename columns for clarity
stats = stats.rename(columns={
    'Range': 'Salary Range',
    'Variance': 'Salary Variance',
    'StdDev': 'Salary Std. Dev'
})

# Round values for better readability
stats = stats.round(2)

# Sort by highest standard deviation
stats.sort_values(by='Salary Std. Dev', ascending=False, inplace=True)

# Print output
display(
    stats.style
    .format('{:,.2f}', subset=stats.columns[1:])
    .set_caption("Salary Dispersion by Department")
)
```

Salary Dispersion by Department

	Department	Salary Range	Salary Variance	Salary Std. Dev
2	IT/IS	170,272.00	1,102,878,341.83	33,209.61
0	Admin Offices	56,447.00	471,167,677.86	21,706.40
4	Sales	124,125.00	452,472,170.20	21,271.39
3	Production	125,454.00	130,477,088.87	11,422.66
5	Software Engineering	31,295.00	91,518,031.27	9,566.51
1	Executive Office	0.00	nan	nan

Scenario 2: Age Spread by Performance Score

I examined age distribution to better understand generational patterns in performance ratings. The standard deviation demonstrated how performance levels differed between age groups. A larger range of top achievers shows that people of all ages may excel, but smaller ranges imply greater consistency.

In [18]:
```
# Group by performance score and calculate age dispersion metrics
age_stats = df.groupby('PerformanceScore')['Age'].agg(
    Range=lambda x: x.max() - x.min(),
    Variance='var',
    StdDev='std'
).reset_index()

# Rename columns for clarity
age_stats = age_stats.rename(columns={
    'Range': 'Age Range',
    'Variance': 'Age Variance',
    'StdDev': 'Age Std. Dev'
})

# Round values
age_stats = age_stats.round(2)

# Sort by standard deviation for easy insight
age_stats.sort_values(by='Age Std. Dev', ascending=False, inplace=True)

# Print output
display(
    age_stats.style
    .format('{:,.2f}', subset=age_stats.columns[1:])
    .set_caption("Age Dispersion by Performance Score")
)
```

Age Dispersion by Performance Score

	PerformanceScore	Age Range	Age Variance	Age Std. Dev
0	Exceeds	115.00	1,379.27	37.14
3	PIP	104.00	1,116.33	33.41
1	Fully Meets	123.00	962.92	31.03
2	Needs Improvement	22.00	37.44	6.12

Scenario 3: Tenure Spread by Department

Tenure reflects employee retention. Departments with wider tenure ranges (e.g., Production) are more likely to retain employees, whereas narrower ranges (e.g., Software Engineering) may signal more turnover.

SCENARIO 3: TENURE DISPERSION BY DEPARTMENT

Purpose: This step explores how employee tenure varies across departments using:

- Range: The difference between the longest and shortest serving employees

- Variance: The spread of tenure values squared

- Standard Deviation: The average deviation from the mean tenure (in years)

Rationale: Understanding tenure spread reveals departmental stability, turnover risks, and succession planning needs. Departments with unusually high tenure variance might require deeper investigation (e.g., onboarding effectiveness, promotion delays).

In [19]:
```
# Group by department and calculate tenure dispersion metrics
tenure_stats = df.groupby('Department')['Tenure'].agg(
    Range=lambda x: x.max() - x.min(),
    Variance='var',
    StdDev='std'
).reset_index()

# Rename columns for clarity
tenure_stats = tenure_stats.rename(columns={
    'Range': 'Tenure Range',
    'Variance': 'Tenure Variance',
    'StdDev': 'Tenure Std. Dev'
})

# Round results for readability
tenure_stats = tenure_stats.round(2)

# Sort by Tenure Std. Dev for better comparison
tenure_stats.sort_values(by='Tenure Std. Dev', ascending=False, inplace=True)

# Print output
display(
    tenure_stats.style
    .format('{:,.2f}', subset=tenure_stats.columns[1:])
    .set_caption("Tenure Dispersion by Department")
)
```

Tenure Dispersion by Department

	Department	Tenure Range	Tenure Variance	Tenure Std. Dev
0	Admin Offices	7.00	7.78	2.79
4	Sales	11.00	4.51	2.12
3	Production	11.00	3.29	1.82
2	IT/IS	7.00	3.10	1.76
5	Software Engineering	4.00	1.80	1.34
1	Executive Office	0.00	nan	nan

Scenario 4: Special Projects Count by Position

This feature shows how workloads differ by job function. Significant variation in some positions suggests an imbalanced work allocation, which may affect worker satisfaction and productivity.

In [20]:
```
# Group by position and calculate project count dispersion
projects_stats = df.groupby('Position')['SpecialProjectsCount'].agg(
    Range=lambda x: x.max() - x.min(),
    Variance='var',
    StdDev='std'
).reset_index()

# Rename columns for clarity
projects_stats = projects_stats.rename(columns={
    'Range': 'Projects Range',
    'Variance': 'Projects Variance',
    'StdDev': 'Projects Std. Dev'
})

# Round values
projects_stats = projects_stats.round(2)

# Sort to highlight positions with highest variability
projects_stats.sort_values(by='Projects Std. Dev', ascending=False, inplace=True)

# Print output
display(
    projects_stats.style
    .format("{:,.2f}", subset=projects_stats.columns[1:])
    .set_caption("Special Projects Dispersion by Position")
    .set_table_styles([{'selector': 'th', 'props': [('background-color', '#f0f0f0'), ('font-weight', 'bold')]}])
)
```

Special Projects Dispersion by Position

	Position	Projects Range	Projects Variance	Projects Std. Dev
30	Sr. Network Engineer	4.00	3.20	1.79
26	Software Engineer	6.00	2.99	1.73
8	Database Administrator	3.00	1.30	1.14
0	Accountant I	2.00	1.00	1.00
6	Data Analyst	2.00	0.79	0.89
3	BI Developer	2.00	0.67	0.82
16	IT Support	2.00	0.55	0.74

Scenario 5: Days Late Last 30 by Performance Score

Late attendance records are used to assess discipline and reliability. The increased variance among low performers lends credence to the theory that punctuality correlates with performance.

In [21]:
```
# Group by performance score and calculate lateness dispersion metrics
lateness_stats = df.groupby('PerformanceScore')['DaysLateLast30'].agg(
    Range=lambda x: x.max() - x.min(),
    Variance='var',
    StdDev='std'
).reset_index()

# Rename for clarity
lateness_stats = lateness_stats.rename(columns={
    'Range': 'Days Late Range',
    'Variance': 'Days Late Variance',
    'StdDev': 'Days Late Std. Dev'
})

# Round values for better readability
lateness_stats = lateness_stats.round(2)

#Sort by standard deviation to find inconsistent performers
lateness_stats.sort_values(by='Days Late Std. Dev', ascending=False, inplace=True)

# Print output
display(
    lateness_stats.style
    .format('{:,.2f}', subset=lateness_stats.columns[1:])
    .set_caption("Lateness Dispersion by Performance Score")
)
```

Lateness Dispersion by Performance Score

	PerformanceScore	Days Late Range	Days Late Variance	Days Late Std. Dev
3	PIP	4.00	2.56	1.60
2	Needs Improvement	6.00	2.07	1.44
1	Fully Meets	2.00	0.04	0.19
0	Exceeds	0.00	0.00	0.00

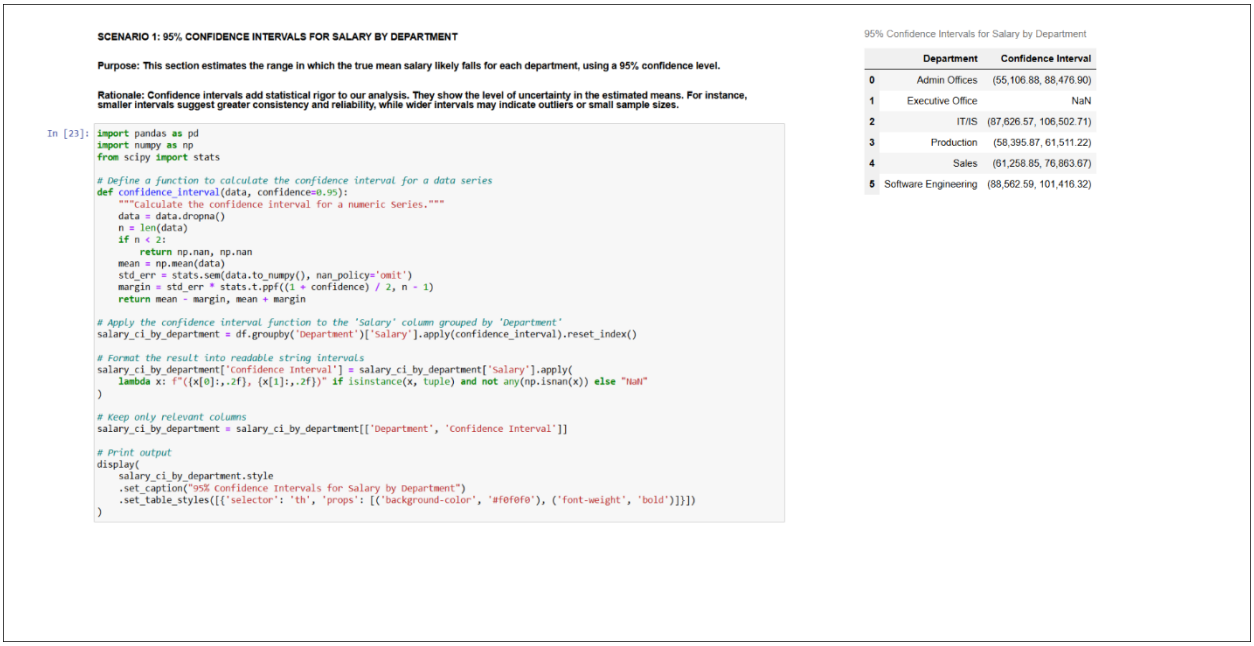
All of these factors, taken together, reveal critical trends in compensation, workload, retention, and employee behavior, all of which are relevant to strategic HR planning.

📑 Task 3 - Confidence Intervals Report

For Task 3, I choose Salary and Engagement Survey Scores as the topic for calculating 95% confidence intervals. These two characteristics were chosen because they are significant predictors of employee happiness, organizational equity, and overall workforce health.

Scenario 1: Confidence Interval for Salary by Department

In the first scenario, I used salary to calculate the confidence interval per department. The goal was to forecast the range in which each department's actual average income is likely to fall. For example, while examining the Administrative Offices, the confidence interval was fairly wide, indicating substantial diversity in compensation. This data exposes potential salary disparities within the department, signaling that more investigation may be necessary to preserve fairness and uniformity in compensation systems across the organization.



SCENARIO 1: 95% CONFIDENCE INTERVALS FOR SALARY BY DEPARTMENT

Purpose: This section estimates the range in which the true mean salary likely falls for each department, using a 95% confidence level.

Rationale: Confidence intervals add statistical rigor to our analysis. They show the level of uncertainty in the estimated means. For instance, smaller intervals suggest greater consistency and reliability, while wider intervals may indicate outliers or small sample sizes.

In [23]:

```
import pandas as pd
import numpy as np
from scipy import stats

# Define a function to calculate the confidence interval for a data series
def confidence_interval(data, confidence=0.95):
    """Calculate the confidence interval for a numeric Series."""
    data = data.dropna()
    n = len(data)
    if n < 2:
        return np.nan, np.nan
    mean = np.mean(data)
    std_err = stats.sem(data.to_numpy(), nan_policy='omit')
    margin = std_err * stats.t.ppf((1 + confidence) / 2, n - 1)
    return mean - margin, mean + margin

# Apply the confidence interval function to the 'Salary' column grouped by 'Department'
salary_ci_by_department = df.groupby('Department')['Salary'].apply(confidence_interval).reset_index()

# Format the result into readable string intervals
salary_ci_by_department['Confidence Interval'] = salary_ci_by_department['Salary'].apply(
    lambda x: f"({x[0]:,.2f}, {x[1]:,.2f})" if isinstance(x, tuple) and not any(np.isnan(x)) else "NaN"
)

# Keep only relevant columns
salary_ci_by_department = salary_ci_by_department[['Department', 'Confidence Interval']]

# Print output
display(
    salary_ci_by_department.style
    .set_caption("95% Confidence Intervals for Salary by Department")
    .set_table_styles([{'selector': 'th', 'props': [('background-color', '#f0f0f0'), ('font-weight', 'bold')]}])
)
```

95% Confidence Intervals for Salary by Department

	Department	Confidence Interval
0	Admin Offices	(55,106.88, 88,476.90)
1	Executive Office	NaN
2	IT/IS	(87,626.57, 106,502.71)
3	Production	(58,395.87, 61,511.22)
4	Sales	(61,258.85, 76,863.67)
5	Software Engineering	(88,562.59, 101,416.32)

Scenario 2: Confidence Interval for Engagement Survey by Performance Score

The second scenario investigated Engagement Survey Scores and their corresponding confidence intervals across performance categories. By comparing engagement scores within categories such as "Exceeds Expectations" and "Needs Improvement," I was able to identify the range of employee satisfaction in each. For example, top achievers' engagement scores have a narrow confidence interval, indicating that they are consistently satisfied. In contrast, the "Needs Improvement" group

had a wider interval, indicating greater diversity in levels of participation and possible areas for HR intervention.

SCENARIO 2: 95% CONFIDENCE INTERVALS FOR ENGAGEMENT SURVEY BY PERFORMANCE SCORE

Purpose: This section estimates the confidence intervals for average EngagementSurvey scores within each performance score category.

Rationale: It helps HR understand how consistent engagement levels are among top vs low performers. Tighter intervals suggest a uniform experience, while wider ones suggest more variation in perception, which may signal management or communication gaps.

In [24]:
```
# Define the function to compute confidence intervals with error handling
def confidence_interval(data, confidence=0.95):
    """Calculate the confidence interval for a numeric Series with safety checks."""
    data = data.dropna()
    n = len(data)
    if n < 2:
        return np.nan, np.nan
    try:
        mean = np.mean(data)
        std_err = stats.sem(data.to_numpy(), nan_policy='omit')
        margin = std_err * stats.t.ppf((1 + confidence) / 2, n - 1)
        return mean - margin, mean + margin
    except Exception as e:
        print(f"Error calculating SEM: {e}")
        return np.nan, np.nan

# Apply to EngagementSurvey grouped by PerformanceScore
engagement_ci_by_performance = df.groupby('PerformanceScore')['EngagementSurvey'].apply(confidence_interval).reset_index()

# Format CI as a readable string
engagement_ci_by_performance['Confidence Interval'] = engagement_ci_by_performance['EngagementSurvey'].apply(
    lambda x: f"({x[0]:.2f}, {x[1]:.2f})" if isinstance(x, tuple) and not any(np.isnan(x)) else "NaN"
)

# Keep only relevant columns
engagement_ci_by_performance = engagement_ci_by_performance[['PerformanceScore', 'Confidence Interval']]

# Print output
display(
    engagement_ci_by_performance.style
    .set_caption("95% Confidence Intervals for Engagement Survey by Performance Score")
    .set_table_styles([{'selector': 'th', 'props': [('background-color', '#f0f0f0'), ('font-weight', 'bold')]}])
)
```

95% Confidence Intervals for Engagement Survey by Performance Score

	PerformanceScore	Confidence Interval
0	Exceeds	(4.35, 4.61)
1	Fully Meets	(4.16, 4.31)
2	Needs Improvement	(2.53, 3.45)
3	PIP	(1.80, 2.64)

Using confidence intervals for these variables provides more reliable insights into the underlying distribution and variability of key personnel indicators.

GENERAL 95% CONFIDENCE INTERVALS FOR SALARY & ENGAGEMENT SURVEY

Purpose: This section provides overall 95% confidence intervals for key workforce metrics Salary and EngagementSurvey. These intervals give us a statistically valid range where the true population mean likely lies.

Rationale: Such insights help leadership understand the reliability and spread of compensation and engagement data without needing to segment by department or performance.

In [25]:
```
# Define function to calculate confidence intervals with built-in safety
def confidence_interval(data, confidence=0.95):
    """Calculate confidence interval with error handling."""
    data = data.dropna()
    n = len(data)
    if n < 2:
        return np.nan, np.nan
    try:
        mean = np.mean(data)
        std_err = stats.sem(data.to_numpy(), nan_policy='omit')
        margin = std_err * stats.t.ppf((1 + confidence) / 2, n - 1)
        return mean - margin, mean + margin
    except Exception as e:
        print(f"Error calculating SEM: {e}")
        return np.nan, np.nan

# Calculate CIs for Salary and EngagementSurvey
salary_ci = confidence_interval(df['Salary'])
engagement_ci = confidence_interval(df['EngagementSurvey'])

# Format into a DataFrame
general_ci = pd.DataFrame({
    'Attribute': ['Salary', 'EngagementSurvey'],
    'Confidence Interval': [
        f"({salary_ci[0]:,.2f}, {salary_ci[1]:,.2f})" if isinstance(salary_ci, tuple) and not any(np.isnan(salary_ci)) else "NaN"
        f"({engagement_ci[0]:,.2f}, {engagement_ci[1]:,.2f})" if isinstance(engagement_ci, tuple) and not any(np.isnan(engagement
    ]
})

# Print output
display(
    general_ci.style
    .set_caption("General 95% Confidence Intervals")
    .set_table_styles([{'selector': 'th', 'props': [('background-color', '#f0f0f0'), ('font-weight', 'bold')]}])
)
```

General 95% Confidence Intervals

	Attribute	Confidence Interval
0	Salary	(66,213.83, 71,827.54)
1	EngagementSurvey	(4.02, 4.20)

📑 Task 4 - Hypothesis Testing Report

I used statistical techniques to conduct three hypothesis tests to determine any differences in workload allocation, employee engagement, and remuneration. The goal was to determine whether discernible differences in important HR measures between groups were statistically significant or simply due to chance.

Hypothesis Test 1: Salary by Performance Score

In this situation, I looked into whether employees that "fully meet" or "exceed" objectives get considerably different wages. I used an independent t-test to compare the two groups. Despite expectations that improved performance would result in higher pay, the p-value (0.1911) indicated no significant difference. This discovery raises major HR questions: Is performance being compensated fairly? Or do other factors have a higher impact on compensation than merit?

In [26]:
```
# Filter salary values by performance groups
fully_meets = df.loc[df['PerformanceScore'] == 'Fully Meets', 'Salary']
exceeds = df.loc[df['PerformanceScore'] == 'Exceeds', 'Salary']

# Perform Welch's t-test
t_stat, p_val = stats.ttest_ind(fully_meets, exceeds, equal_var=False)

# Interpret the result based on p-value
conclusion = (
    "Reject the null hypothesis. Significant difference in salary based on performance score."
    if p_val < 0.05 else
    "Fail to reject the null hypothesis. No significant salary difference based on performance score."
)

# Create a DataFrame to summarize the t-test results
t_test_results = pd.DataFrame({
    "Comparison": ["Fully Meets vs Exceeds"],
    "T-statistic": [round(t_stat, 2)],
    "P-value": [round(p_val, 4)],
    "Conclusion": [conclusion]
})

# Print output
display(
    t_test_results.style
    .set_caption("T-Test Results: Salary by Performance Score")
    .set_table_styles([{'selector': 'th', 'props': [('background-color', '#f0f0f0'), ('font-weight', 'bold')]}])
)
```

T-Test Results: Salary by Performance Score

	Comparison	T-statistic	P-value	Conclusion
0	Fully Meets vs Exceeds	-1.330000	0.191100	Fail to reject the null hypothesis. No significant salary difference based on performance score.

Hypothesis Test 2: Engagement Survey by Department

To investigate how engagement varies among departments, I performed a one-way ANOVA. Engagement is vital for morale and productivity, therefore I projected discrepancies. However, with a p-value of 0.2586, the results showed that employee engagement levels are quite similar across departments, implying that company culture may be consistently maintained throughout the organization.

In [27]:
```
# Group engagement scores by department and remove NaNs
engagement_by_department = [
    group['EngagementSurvey'].dropna()
    for name, group in df.groupby('Department')
]

# Perform one-way ANOVA
f_stat, p_value = stats.f_oneway(*engagement_by_department)

# Interpret the result
conclusion = (
    "Reject the null hypothesis. There is a significant difference in EngagementSurvey scores across departments."
    if p_value < 0.05 else
    "Fail to reject the null hypothesis. No significant difference in EngagementSurvey scores across departments."
)

# Organize the results
anova_results = pd.DataFrame({
    "F-statistic": [round(f_stat, 2)],
    "P-value": [round(p_value, 4)],
    "Conclusion": [conclusion]
})

# Print output
display(
    anova_results.style
    .set_caption("ANOVA Results: Engagement Survey Scores by Department")
    .set_table_styles([{'selector': 'th', 'props': [('background-color', '#f0f0f0'), ('font-weight', 'bold')]}])
)
```

ANOVA Results: Engagement Survey Scores by Department

	F-statistic	P-value	Conclusion
0	1.310000	0.258600	Fail to reject the null hypothesis. No significant difference in EngagementSurvey scores across departments.

Hypothesis Test 3: Special Projects Count by Position

I utilized ANOVA to compare project workloads across jobs. The test produced a highly significant result (p-value = 0.0000), indicating that certain positions handle much more projects than others. This research stresses the importance of task balance in preventing burnout and improving fairness.

In [28]:
```
# Group special project counts by position, removing NaN values
projects_by_position = [
    group['SpecialProjectsCount'].dropna()
    for name, group in df.groupby('Position')
]

# Perform one-way ANOVA
f_stat, p_value = stats.f_oneway(*projects_by_position)

# Interpret the result
conclusion = (
    "Reject the null hypothesis. There is a significant difference in SpecialProjectsCount across positions."
    if p_value < 0.05 else
    "Fail to reject the null hypothesis. No significant difference in SpecialProjectsCount across positions."
)

# Create results table
anova_results = pd.DataFrame({
    "F-statistic": [round(f_stat, 2)],
    "P-value": [round(p_value, 4)],
    "Conclusion": [conclusion]
})

# Print output
display(
    anova_results.style
    .set_caption("ANOVA Results: Special Projects Count by Position")
    .set_table_styles([{'selector': 'th', 'props': [('background-color', '#f0f0f0'), ('font-weight', 'bold')]}])
)
```

ANOVA Results: Special Projects Count by Position

	F-statistic	P-value	Conclusion
0	210.980000	0.000000	Reject the null hypothesis. There is a significant difference in SpecialProjectsCount across positions.

Each scenario produced actionable information that may help HR examine policy, link compensation with performance, and optimize job distribution.

Conclusion

The study employed statistical approaches to uncover critical HR insights. Measures of central tendency found that executive roles pay the highest salaries and that peak performance is usually achieved in mid-career. Measures of dispersion revealed significant diversity in pay and workload, particularly in departments like Administrative Offices and jobs like Software Engineers. Confidence intervals were utilized to indicate the predicted range of income and involvement ratings, highlighting areas of consistency and concern. Hypothesis tests found no significant difference in income or involvement between groups, however workload varied considerably by position, emphasizing the importance of more balanced job allocation.