PYTHON PROGRAMMING FOR BUSINESS ANALYTICS

NAME: MANIBALAN AMARANATHAN

DATASET : EMPLOYEE DATA FROM KAGGLE

Basic Python concepts

Variable: Store a sample employee's details

In [38]:
```
employee_sample = {
    "Employee ID": "E001",
    "Full Name": "John Doe",
    "Job Title": "Software Engineer",
    "Annual Salary": 120000,
}
print("Sample Employee:", employee_sample)
```

```
Sample Employee: {'Employee ID': 'E001', 'Full Name': 'John Doe', 'Job Title': 'S
oftware Engineer', 'Annual Salary': 120000}
```

List: Extract unique job titles

In [3]:
```
job_titles = ["Software Engineer", "Data Analyst", "HR Manager", "Product Manage
print("Total Unique Job Titles:", len(job_titles))
```

```
Total Unique Job Titles: 5
```

Dictionary: Store Employee ID as key and Full Name as value

In [5]:
```
employee_dict = {
    "E001": "John Doe",
    "E002": "Jane Smith",
    "E003": "Alice Johnson",
    "E004": "Bob Brown",
    "E005": "Charlie Davis"
}
print("Total Employees in Dictionary:", len(employee_dict))
```

```
Total Employees in Dictionary: 5
```

Loop: Print first 5 employees' names

In [7]:
```
print("First 5 Employees:")
for emp_id, name in employee_dict.items():
    print(f"{emp_id}: {name}")
```

```
First 5 Employees:
E001: John Doe
E002: Jane Smith
E003: Alice Johnson
E004: Bob Brown
E005: Charlie Davis
```

Conditional Statements: Filter employees earning above $100,000

In [9]:
```
employee_salaries = {
    "John Doe": 120000,
    "Jane Smith": 95000,
    "Alice Johnson": 110000,
    "Bob Brown": 87000,
    "Charlie Davis": 105000
}

high_earners = [name for name, salary in employee_salaries.items() if salary > 1
print("Employees earning above $100,000:", high_earners)
```

```
Employees earning above $100,000: ['John Doe', 'Alice Johnson', 'Charlie Davis']
```

Reusable function

In [11]:
```
def get_high_earners(salary_threshold):
    """Returns a list of employees earning above the given salary threshold."""
    return [name for name, salary in employee_salaries.items() if salary > salar

threshold = 100000
high_earners = get_high_earners(threshold)
print(f"Employees earning above ${threshold}:", high_earners)
```

```
Employees earning above $100000: ['John Doe', 'Alice Johnson', 'Charlie Davis']
```

Interpretation of Analysis Steps and Visualizations

Loading the CSV file using pandas.read_csv function

In [33]:
```
import pandas as pd

file_path = r"C:\Users\balas\Downloads\Employee Sample Data 1.csv"
df = pd.read_csv(file_path,encoding='latin1')
df.head()
```

Out[33]:

	Employee ID	Full Name	Job Title	Department	Business Unit	Gender	Ethnicity
0	E02002	Kai Le	Controls Engineer	Engineering	Manufacturing	Male	Asian
1	E02003	Robert Patel	Analyst	Sales	Corporate	Male	Asian
2	E02004	Cameron Lo	Network Administrator	IT	Research & Development	Male	Asian
3	E02005	Harper Castillo	IT Systems Architect	IT	Corporate	Female	Latino
4	E02006	Harper Dominguez	Director	Engineering	Corporate	Female	Latino

Checking if there is any NA in the dataset

In [42]:
```
df.isna().sum()
```

Out[42]:
```
Employee ID        19
Full Name          40
Job Title          58
Department         47
Business Unit      82
Gender             49
Ethnicity          42
Age                 6
Hire Date          35
Annual Salary      73
Bonus %            48
Country           106
City               55
Exit Date        1137
dtype: int64
```

Handling Missing Values:

Exit Dates**: Replacing missing `Exit Date` with "00/00/0000" assumes employees without an exit date are still active. This clarifies retention rates and workforce stability.

In [44]:
```
df["Exit Date"].fillna("00/00/0000", inplace=True)
#count of NA in exit date column
df["Exit Date"].isna().sum()
```

```
C:\Users\balas\AppData\Local\Temp\ipykernel_7840\1364157816.py:2: FutureWarning:
A value is trying to be set on a copy of a DataFrame or Series through chained as
signment using an inplace method.
The behavior will change in pandas 3.0. This inplace method will never work becau
se the intermediate object on which we are setting values always behaves as a cop
y.

For example, when doing 'df[col].method(value, inplace=True)', try using 'df.meth
od({col: value}, inplace=True)' or df[col] = df[col].method(value) instead, to pe
rform the operation inplace on the original object.


  df["Exit Date"].fillna("00/00/0000", inplace=True)
```

Out[44]:
```
0
```

Forward-Filling:

Ensures continuity in other columns (e.g., filling missing job titles or departments based on previous entries).

In [46]:
```
df.fillna(method='ffill', inplace=True)
#count of NA's in column wise
df.isna().sum()
```

```
C:\Users\balas\AppData\Local\Temp\ipykernel_7840\4102867782.py:2: FutureWarning:
DataFrame.fillna with 'method' is deprecated and will raise in a future version.
Use obj.ffill() or obj.bfill() instead.
  df.fillna(method='ffill', inplace=True)
```

Out[46]:
```
Employee ID      0
Full Name        0
Job Title        0
Department       0
Business Unit    0
Gender           0
Ethnicity        0
Age              0
Hire Date        0
Annual Salary    0
Bonus %          0
Country          0
City             0
Exit Date        0
dtype: int64
```

Removing Duplicates:

Ensures data accuracy by eliminating redundant entries.

In [48]:
```
df.drop_duplicates(inplace=True)
```

Data Type Conversions:

Salary & Bonus: Numeric conversions enable mathematical operations (e.g., calculating total compensation).
Date Parsing: Facilitates tenure analysis and time-based metrics.

In [52]:
```
df['Annual Salary'] = df['Annual Salary'].replace("[\$,]", '', regex=True).astyp
df['Bonus %'] = df['Bonus %'].replace('%', '', regex=True).astype(float)
df['Hire Date'] = pd.to_datetime(df['Hire Date'], errors='coerce')
df['Exit Date'] = pd.to_datetime(df['Exit Date'], errors='coerce')
df.info()
```

```
<class 'pandas.core.frame.DataFrame'>
Index: 1113 entries, 0 to 1196
Data columns (total 14 columns):
 #   Column         Non-Null Count  Dtype
---  ------         --------------  -----
 0   Employee ID    1113 non-null   object
 1   Full Name      1113 non-null   object
 2   Job Title      1113 non-null   object
 3   Department     1113 non-null   object
 4   Business Unit  1113 non-null   object
 5   Gender         1113 non-null   object
 6   Ethnicity      1113 non-null   object
 7   Age            1113 non-null   float64
 8   Hire Date      1113 non-null   datetime64[ns]
 9   Annual Salary  1113 non-null   float64
 10  Bonus %        1113 non-null   float64
 11  Country        1113 non-null   object
 12  City           1113 non-null   object
 13  Exit Date      113 non-null    datetime64[ns]
dtypes: datetime64[ns](2), float64(3), object(9)
memory usage: 130.4+ KB
```

```
<>:1: SyntaxWarning: invalid escape sequence '\$'
<>:1: SyntaxWarning: invalid escape sequence '\$'
C:\Users\balas\AppData\Local\Temp\ipykernel_7840\975286643.py:1: SyntaxWarning: i
nvalid escape sequence '\$'
  df['Annual Salary'] = df['Annual Salary'].replace("[\$,]", '', regex=True).asty
pe(float)
```

In [60]:
```
df["Annual Salary"].head()
```

Out[60]:
```
0     92368.0
1     45703.0
2     83576.0
3     98062.0
4    175391.0
Name: Annual Salary, dtype: float64
```

Department-Specific Insights

IT Department Filter:

Identifies the size of the IT workforce (e.g., 250 employees). This could reflect the company's reliance on technology or need for tech talent.

In [62]:
```
it_employees = df[df['Department'] == 'IT']
print("Total IT Employees:", len(it_employees))
```

```
Total IT Employees: 312
```

Employees with highest salaries

Sorting the employees with salaries to get highest earning employee in the organization. Reveals where the company allocates its budget.

In [160…
```
df_sorted = df.sort_values(by="Annual Salary", ascending=False)
print("\nTop 5 highest earners:")
print(df_sorted.head())
```

```
Top 5 highest earners:
    Employee ID        Full Name       Job Title       Department  \
555      E02557    Robert Rogers  Vice President      Engineering
947      E02949    Kinsley Huynh  Vice President  Human Resources
331      E02333  Christopher Luu  Vice President      Engineering
254      E02256  Eloise Williams  Vice President            Sales
182      E02184    Ariana Sharma  Vice President            Sales

              Business Unit  Gender  Ethnicity   Age  Hire Date  \
555           Manufacturing    Male  Caucasian  53.0 2011-08-26
947      Specialty Products  Female  Caucasian  53.0 2019-01-13
331  Research & Development    Male      Asian  61.0 2005-10-27
254      Specialty Products  Female      Black  42.0 2019-02-12
182  Research & Development  Female      Asian  45.0 2014-01-24

     Annual Salary  Bonus %        Country     City  Exit Date  \
555       258734.0     34.0  United States  Seattle        NaT
947       258722.0     36.0  United States  Phoenix        NaT
331       258700.0     32.0  United States   Austin        NaT
254       258115.0     36.0  United States   Austin        NaT
182       257725.0     34.0  United States  Seattle 2022-06-29

     Total Compensation
555           346703.56
947           351861.92
331           341484.00
254           351036.40
182           345351.50
```

Average Salary by Department:

Engineering or Executive roles may show higher averages, signaling investment in specialized skills. Lower averages in departments like HR or Sales might indicate junior roles or cost-saving strategies.

In [66]:
```
department_salary_avg = df.groupby("Department")["Annual Salary"].mean()
print("Average Salary by Department:")
print(department_salary_avg)
```

```
Average Salary by Department:
Department
Accounting          124939.158730
Engineering         110020.625000
Finance             119486.495575
Human Resources     122965.938053
IT                   91156.150641
Marketing           122716.937500
Sales               100539.674556
Name: Annual Salary, dtype: float64
```

Mathametical Operations

Calculating Total Compensation

Combines salary and bonuses to show the true cost of employees. High bonuses in Sales/Finance might reflect performance-driven cultures.

In [120…
```
df["Total Compensation"] = df["Annual Salary"] + (df["Annual Salary"] * (df["Bon
print("Total Compensation Column Added:")
print(df[['Employee ID', 'Annual Salary', 'Bonus %', 'Total Compensation']])
```

```
Total Compensation Column Added:
     Employee ID  Annual Salary  Bonus %  Total Compensation
0         E02002        92368.0      0.0            92368.00
1         E02003        45703.0      0.0            45703.00
2         E02004        83576.0      0.0            83576.00
3         E02005        98062.0      0.0            98062.00
4         E02006       175391.0     24.0           217484.84
...          ...            ...      ...                 ...
1192      E02021       102649.0      6.0           108807.94
1193      E02022       122875.0     12.0           137620.00
1194      E02023        83323.0      0.0            83323.00
1195      E02024        66721.0      0.0            66721.00
1196      E02025       246400.0     36.0           335104.00

[1113 rows x 4 columns]
```

Bonus Metrics:

Average Salary for an employees with a bonus

In [123…
```
avg_salary_with_bonus = df[df['Bonus %'] > 0]['Annual Salary'].mean()
print(f"Average Salary for Employees with a Bonus: ${avg_salary_with_bonus:.2f}"
```

```
Average Salary for Employees with a Bonus: $153584.68
```

Average Bonus:

A low average (e.g., 5%) suggests conservative bonus policies.

Total Bonus Expenditure:

Critical for financial planning (e.g., $2M spent on bonuses annually).

In [72]:
```
avg_bonus_pct = df["Bonus %"].mean()
print(f"Average Bonus Percentage: {avg_bonus_pct:.2f}%")

total_bonus = (df["Annual Salary"] * df["Bonus %"] / 100).sum()
print(f"Total Bonus Expenditure: ${total_bonus:.2f}")
```

```
Average Bonus Percentage: 8.00%
Total Bonus Expenditure: $15800790.41
```

Diversity & Inclusion

Gender Distribution:

A skewed ratio (e.g., 70% Male, 30% Female) could indicate a need for diversity initiatives.

In [131…
```
gender_percentage = df['Gender'].value_counts(normalize=True) * 100
print("Percentage of Employees by Gender:")
print(gender_percentage)
```

```
Percentage of Employees by Gender:
Gender
Female    52.201258
Male      47.798742
Name: proportion, dtype: float64
```

Ethnicity Distribution:

High Asian/Latino representation in tech roles might reflect hiring trends or regional demographics.

In [133…
```
ethnicity_percentage = df['Ethnicity'].value_counts(normalize=True) * 100
print("Percentage of Employees by Ethnicity:")
print(ethnicity_percentage)
```

```
Percentage of Employees by Ethnicity:
Ethnicity
Asian        43.935310
Latino       27.672956
Caucasian    21.922731
Black         6.469003
Name: proportion, dtype: float64
```

Visualizations

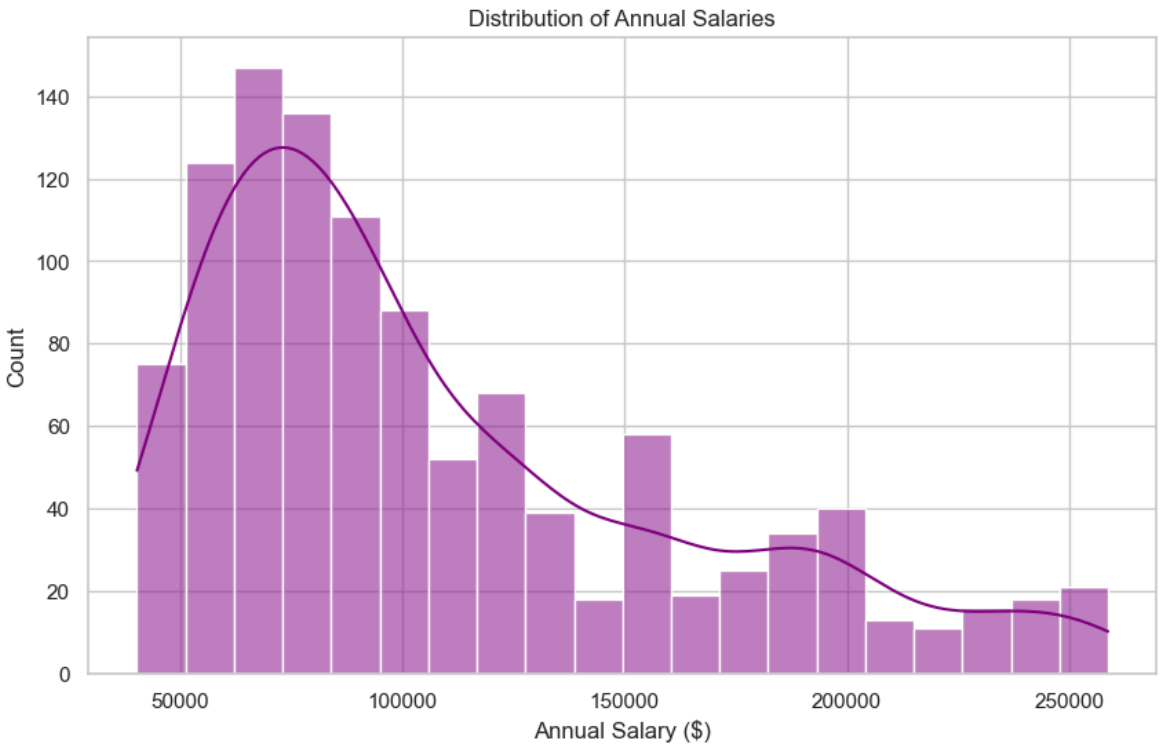
In [76]:
```
import matplotlib.pyplot as plt
import seaborn as sns

# Set the style
sns.set_theme(style="whitegrid")
```

Salary Histogram:

A right-skewed distribution indicates most employees earn below $100k, with a few high earners.

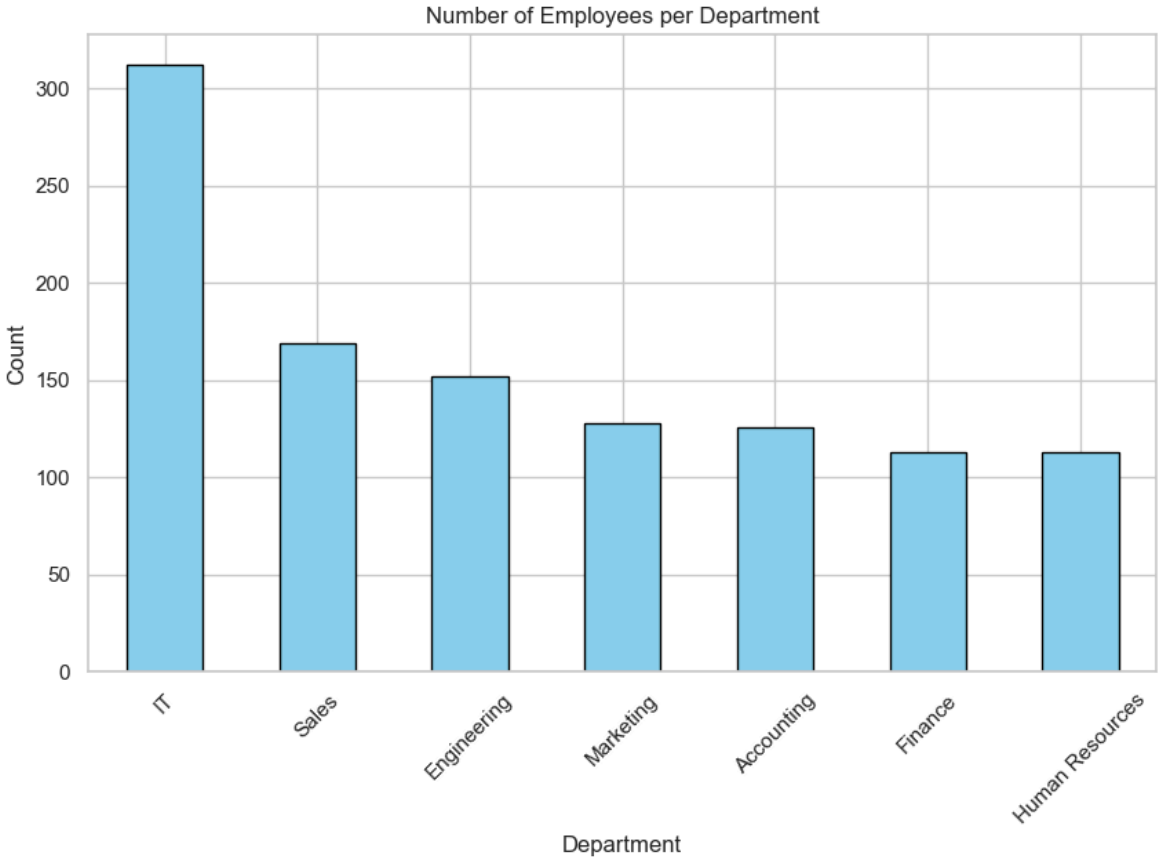
In [78]:
```
plt.figure(figsize=(10, 6))
sns.histplot(df["Annual Salary"], bins=20, kde=True, color="purple")
plt.title("Distribution of Annual Salaries")
plt.xlabel("Annual Salary ($)")
plt.show()
```



Employees per Department:

Larger departments (e.g., Sales, IT) may require more resources or have higher turnover.

In [80]:
```
plt.figure(figsize=(10, 6))
df["Department"].value_counts().plot(kind="bar", color="skyblue", edgecolor="bla
plt.title("Number of Employees per Department")
plt.ylabel("Count")
plt.xlabel("Department")
plt.xticks(rotation=45)
plt.show()
```



Salary Boxplot by Department:

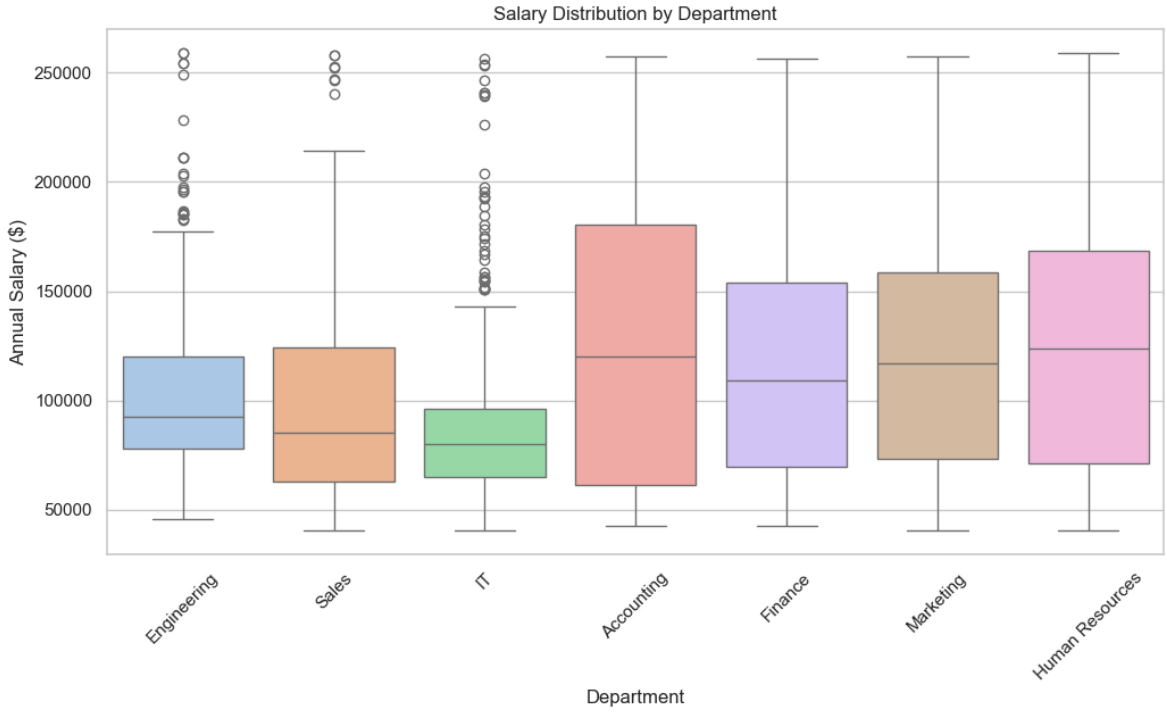
Outliers in Engineering/Finance suggest highly paid specialists. Narrow ranges in HR indicate standardized pay.

In [82]:
```
# Boxplot: Salary Distribution by Department
plt.figure(figsize=(12, 6))
sns.boxplot(x="Department", y="Annual Salary", data=df, palette="pastel")
plt.title("Salary Distribution by Department")
plt.xlabel("Department")
plt.ylabel("Annual Salary ($)")
plt.xticks(rotation=45)
plt.show()
```

```
C:\Users\balas\AppData\Local\Temp\ipykernel_7840\2438496701.py:3: FutureWarning:

Passing `palette` without assigning `hue` is deprecated and will be removed in v
0.14.0. Assign the `x` variable to `hue` and set `legend=False` for the same effe
ct.

  sns.boxplot(x="Department", y="Annual Salary", data=df, palette="pastel")
```



SQL & Advanced Queries

SQL Alchemy

Create an SQLite database and Store the dataset in the database

In [84]:
```
from sqlalchemy import create_engine

engine = create_engine("sqlite:///employee_data.db", echo=True)
df.to_sql("employees", con=engine, if_exists="replace", index=False)
engine.dispose()
```

```
2025-01-31 14:59:13,143 INFO sqlalchemy.engine.Engine BEGIN (implicit)
2025-01-31 14:59:13,159 INFO sqlalchemy.engine.Engine PRAGMA main.table_info("emp
loyees")
2025-01-31 14:59:13,159 INFO sqlalchemy.engine.Engine [raw sql] ()
2025-01-31 14:59:13,164 INFO sqlalchemy.engine.Engine PRAGMA temp.table_info("emp
loyees")
2025-01-31 14:59:13,164 INFO sqlalchemy.engine.Engine [raw sql] ()
2025-01-31 14:59:13,164 INFO sqlalchemy.engine.Engine
CREATE TABLE employees (
        "Employee ID" TEXT,
        "Full Name" TEXT,
        "Job Title" TEXT,
        "Department" TEXT,
        "Business Unit" TEXT,
        "Gender" TEXT,
        "Ethnicity" TEXT,
        "Age" FLOAT,
        "Hire Date" DATETIME,
        "Annual Salary" FLOAT,
        "Bonus %" FLOAT,
        "Country" TEXT,
        "City" TEXT,
        "Exit Date" DATETIME,
        "Total Compensation" FLOAT
)


2025-01-31 14:59:13,175 INFO sqlalchemy.engine.Engine [no key 0.00336s] ()
2025-01-31 14:59:13,253 INFO sqlalchemy.engine.Engine INSERT INTO employees ("Emp
loyee ID", "Full Name", "Job Title", "Department", "Business Unit", "Gender", "Et
hnicity", "Age", "Hire Date", "Annual Salary", "Bonus %", "Country", "City", "Exi
t Date", "Total Compensation") VALUES (?, ?, ?, ?, ?, ?, ?, ?, ?, ?, ?, ?, ?, ?,
?)
2025-01-31 14:59:13,269 INFO sqlalchemy.engine.Engine [generated in 0.03377s]
[('E02002', 'Kai Le', 'Controls Engineer', 'Engineering', 'Manufacturing', 'Mal
e', 'Asian', 47.0, '2022-02-05 00:00:00.000000', 92368.0, 0.0, 'United States',
'Columbus', None, 92368.0), ('E02003', 'Robert Patel', 'Analyst', 'Sales', 'Corpo
rate', 'Male', 'Asian', 58.0, '2013-10-23 00:00:00.000000', 45703.0, 0.0, 'United
States', 'Chicago', None, 45703.0), ('E02004', 'Cameron Lo', 'Network Administrat
or', 'IT', 'Research & Development', 'Male', 'Asian', 34.0, '2019-03-24 00:00:00.
000000', 83576.0, 0.0, 'China', 'Shanghai', None, 83576.0), ('E02005', 'Harper Ca
stillo', 'IT Systems Architect', 'IT', 'Corporate', 'Female', 'Latino', 39.0, '20
18-04-07 00:00:00.000000', 98062.0, 0.0, 'United States', 'Seattle', None, 98062.
0), ('E02006', 'Harper Dominguez', 'Director', 'Engineering', 'Corporate', 'Femal
e', 'Latino', 42.0, '2005-06-18 00:00:00.000000', 175391.0, 24.0, 'United State
s', 'Austin', None, 217484.84), ('E02007', 'Ezra Vu', 'Network Administrator', 'I
T', 'Manufacturing', 'Male', 'Asian', 62.0, '2004-04-22 00:00:00.000000', 66227.
0, 0.0, 'United States', 'Phoenix', '2014-02-14 00:00:00.000000', 66227.0), ('E02
008', 'Jade Hu', 'Sr. Analyst', 'Accounting', 'Specialty Products', 'Female', 'As
ian', 58.0, '2009-06-27 00:00:00.000000', 89744.0, 0.0, 'China', 'Chongqing', Non
e, 89744.0), ('E02009', 'Miles Chang', 'Analyst II', 'Finance', 'Corporate', 'Mal
e', 'Asian', 62.0, '1999-02-19 00:00:00.000000', 69674.0, 0.0, 'China', 'Chengd
u', None, 69674.0)  ... displaying 10 of 1113 total bound parameter sets ...  ('E
02024', 'Serenity Cao', 'Account Representative', 'Sales', 'Manufacturing', 'Fema
le', 'Asian', 31.0, '2018-10-21 00:00:00.000000', 66721.0, 0.0, 'United States',
'Miami', None, 66721.0), ('E02025', 'Parker Lai', 'Vice President', 'Accounting',
'Specialty Products', 'Male', 'Asian', 48.0, '2006-11-29 00:00:00.000000', 24640
0.0, 36.0, 'United States', 'Miami', None, 335104.0)]
2025-01-31 14:59:13,269 INFO sqlalchemy.engine.Engine COMMIT
```

Engineering Employees Query:

Extracts all engineers, useful for project staffing or skill gap analysis.

In [ ]:
```
from sqlalchemy.orm import sessionmaker
from sqlalchemy import text

# Create a session
Session = sessionmaker(bind=engine)
session = Session()

# Query
engineering_employees = session.execute(text((
    """SELECT * FROM employees
    WHERE Department = 'Engineering'"""
    ))).fetchall()

# Close the session
session.close()

engineering_employees
```

Gender Diversity per Department:

Reveals imbalances (e.g., Male-dominated Engineering vs. balanced HR), guiding diversity strategies. Break down the number of male and female employees by department.

In [96]:
```
session = Session()

gender_diversity = session.execute(text("""
    SELECT Department, Gender, COUNT(*) AS Employee_Count
    FROM employees
    GROUP BY Department, Gender
    ORDER BY Department
""")).fetchall()

session.close()

gender_diversity
```

```
2025-01-31 15:06:32,817 INFO sqlalchemy.engine.Engine BEGIN (implicit)
2025-01-31 15:06:32,823 INFO sqlalchemy.engine.Engine
    SELECT Department, Gender, COUNT(*) AS Employee_Count
    FROM employees
    GROUP BY Department, Gender
    ORDER BY Department

2025-01-31 15:06:32,826 INFO sqlalchemy.engine.Engine [generated in 0.00203s] ()
2025-01-31 15:06:32,832 INFO sqlalchemy.engine.Engine ROLLBACK
```

Out[96]:
```
[('Accounting', 'Female', 66),
 ('Accounting', 'Male', 60),
 ('Engineering', 'Female', 76),
 ('Engineering', 'Male', 76),
 ('Finance', 'Female', 58),
 ('Finance', 'Male', 55),
 ('Human Resources', 'Female', 63),
 ('Human Resources', 'Male', 50),
 ('IT', 'Female', 158),
 ('IT', 'Male', 154),
 ('Marketing', 'Female', 71),
 ('Marketing', 'Male', 57),
 ('Sales', 'Female', 89),
 ('Sales', 'Male', 80)]
```

Employee Tenure:

Tenure >5 years suggests strong retention.
Short tenure (<1 year) in Sales may indicate high turnover or hiring sprees.

In [ ]:
```
session = Session()

emp_tenure = session.execute(text("""
    SELECT `Employee ID`, `Full Name`, (
        CASE
            WHEN "Exit Date" IS NOT NULL THEN
                JULIANDAY("Exit Date") - JULIANDAY("Hire Date")
            ELSE
                JULIANDAY(CURRENT_DATE) - JULIANDAY("Hire Date")
        END
    ) / 365 AS Tenure_Years
    FROM employees
""")).fetchall()

session.close()

emp_tenure
```

Key Business Implications

Compensation Strategy:

High salaries in Engineering/Finance align with competitive talent markets.

Diversity Gaps:

Underrepresentation of certain groups in leadership roles could impact company culture.

Budget Allocation:

High bonus expenditures in Sales/Finance reflect performance-driven incentives.

Retention:

Long tenure in Corporate roles vs. short tenure in Manufacturing may signal job satisfaction issues.

Operational Focus:

Large IT/Engineering teams suggest a tech-centric business model.