**Leisha Saanvi .M**

**Case Study: Employment Agreement Data Analysis**

**INDEX**

|  |  |
| --- | --- |
| **TITLE** | **PAGE NUMBER** |
| Importing Dataset | 2 |
| Data Cleaning and Pre - Processing | 2 - 3 |
| Exploratory Data Analysis | 4 - 5 |
| Technical Analysis (SQL commands and Python Scripts for Data Analysis & Visualization) | 6 - 8 |
| Cloud Service Integration | 9 |
| Machine Learning Model | 10 |

Initially, an employee dataset is downloaded in csv format, and exported to excel sheet.

The dataset includes the following fields:

* Employee\_ID
* Name
* Department
* Position
* Agreement\_Date
* Contract\_Term (in months)
* Base\_Salary
* Bonus
* Benefits
* Cloud\_Service\_Used (e.g., AWS, Azure, GCP)
* Tools\_Used (e.g., Python, R, SQL)

Link to the dataset:



**Step 1: Dataset is imported in colab.**

In Google Colab, data can be imported using the files.upload() function from the google.colab module, which prompts a file upload dialog. The uploaded file is then read into a pandas DataFrame using pd.read\_csv() combined with io.BytesIO to handle the byte stream.

**Step 2:** **Data Cleaning and Preprocessing:**

 **Checking and Handling Missing Values:**

The code first checks for missing values in the 'Agreement\_Date' column, and fills these missing values using current date as default.

 **Replacing Missing Values with Averages:**

The mean values of 'Contract\_Term', 'Base\_Salary', and 'Bonus' are calculated using .mean()

Missing values in these columns are replaced with their respective mean values using .fillna()

 **Replacing Missing Values with Mode:**

Mode (most frequent value) for 'Benefits', 'Cloud\_Service\_Used', and 'Tools\_Used' is calculated using .mode()[0]. Missing values in these categorical columns are filled with their respective modes using .fillna()

  **Date Standardization**:

The 'Agreement\_Date' column is converted to a datetime.

 **Numerical Standardization**:

'Contract\_Term' is converted to integer type, 'Base\_Salary' and 'Bonus' are converted to float type.

 **Saving Cleaned Data:**

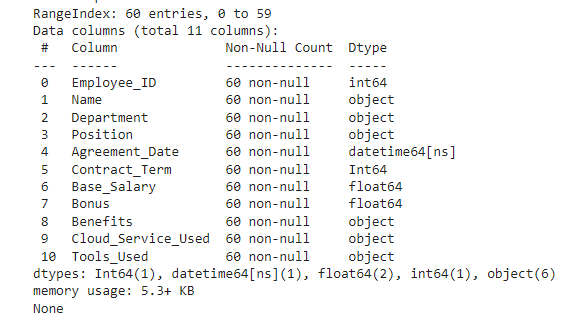
Finally, the cleaned DataFrame is saved to a CSV file named 'cleaned\_employee\_data.csv'.

Link to the cleaned dataset:

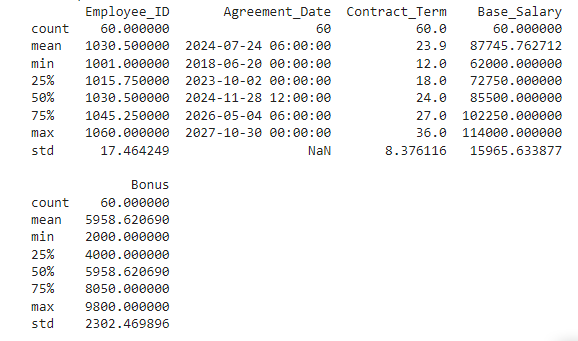


**Step 3: Exploratory Data Analysis**

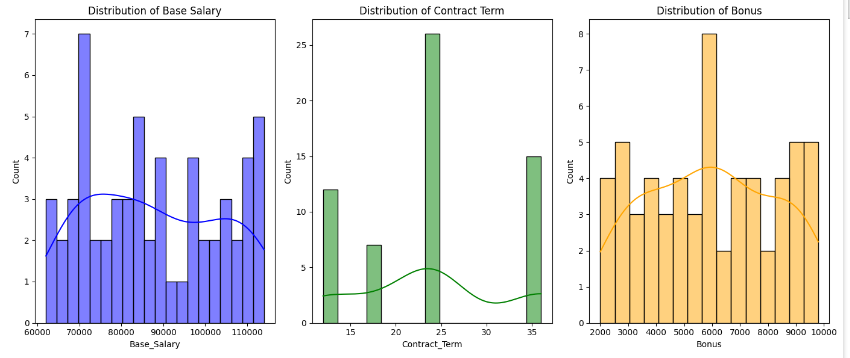
**Dataset Summary:**



**Descriptive Statistics for Numerical Columns:**



* **Visualizing the distribution of key numerical variables: Base\_Salary, Contract\_Term, and Bonus**

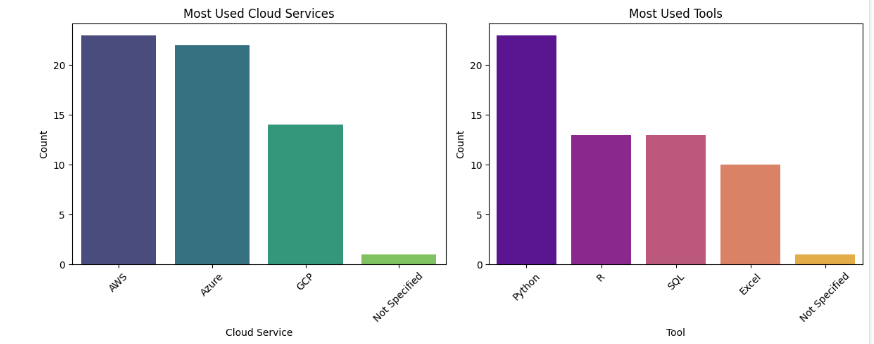


**Base Salary:** It could be observed that the highest count of base salary is around 70000, and the least count lies between 90000 – 97000.

**Contract Term:** Highest count of contract period is 24 months, meanwhile the least lies between 15 to 20 months.

**Bonus:** 6000 was the most frequently given bonus amount, meanwhile 65000 and 8000 is the least frequent.

* **Identify trends in the use of cloud services (Cloud\_Service\_Used) and tools (Tools\_Used)**



The most used **cloud service** is AWS, meanwhile the least used is Google Cloud Platform.

The most commonly used **tool** is Python, the least preferred is Excel.

**Step 4: Technical Analysis**

* SQL Commands – implemented in colab by importing the **pandasql** library.

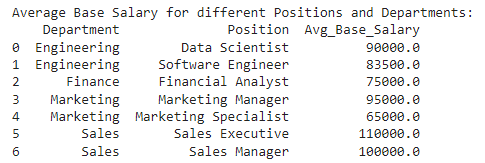
1. **Average Base\_Salary for different Positions and Departments**

SELECT Department, Position, AVG(Base\_Salary) AS Avg\_Base\_Salary

FROM df

GROUP BY Department, Position

ORDER BY Department, Avg\_Base\_Salary DESC;



**It could be observed that Sales Executive has the highest base salasry, meanwhile marketing manager has the least.**

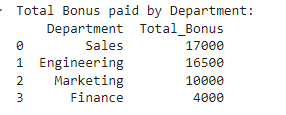
1. **Total Bonus paid by Department**

SELECT Department, SUM(Bonus) AS Total\_Bonus

FROM df

GROUP BY Department

ORDER BY Total\_Bonus DESC;



**It could be observed that Sales department is paid with the highest bonus of 17000, meanwhile Finance dept has the least bonus of 4000.**

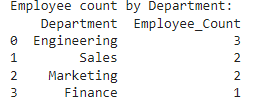
1. **Count of employees in each department**

SELECT Department, COUNT(Employee\_ID) AS Employee\_Count

FROM df

GROUP BY Department

ORDER BY Employee\_Count DESC;

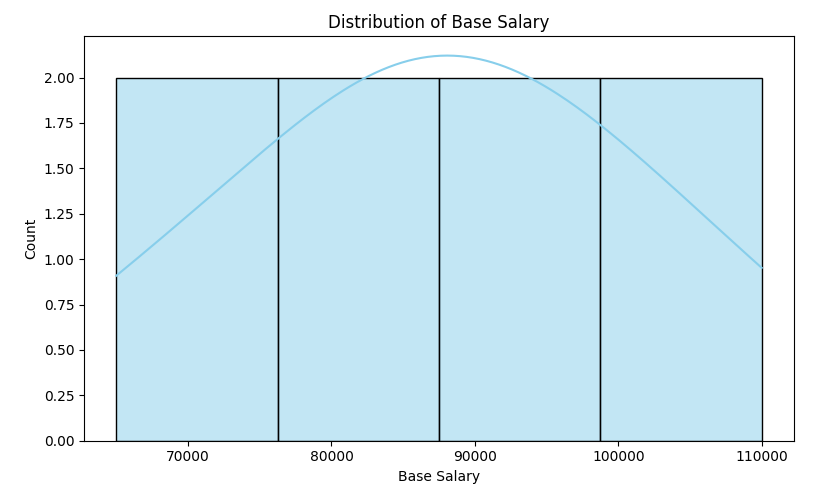
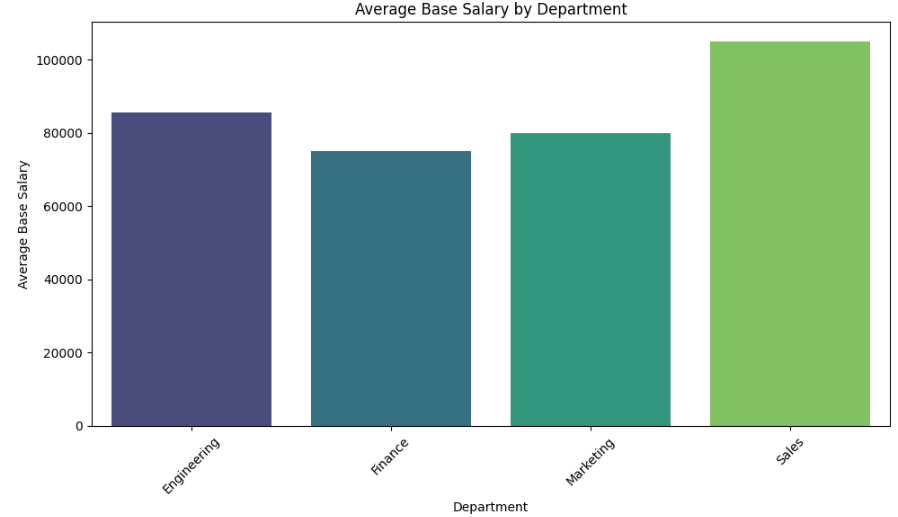


**It could be observed that Engineering department has the highest count of 3, meanwhile Finance has the lowest count of 1.**

**Results of Data analysis and visualization using Python scripts.**

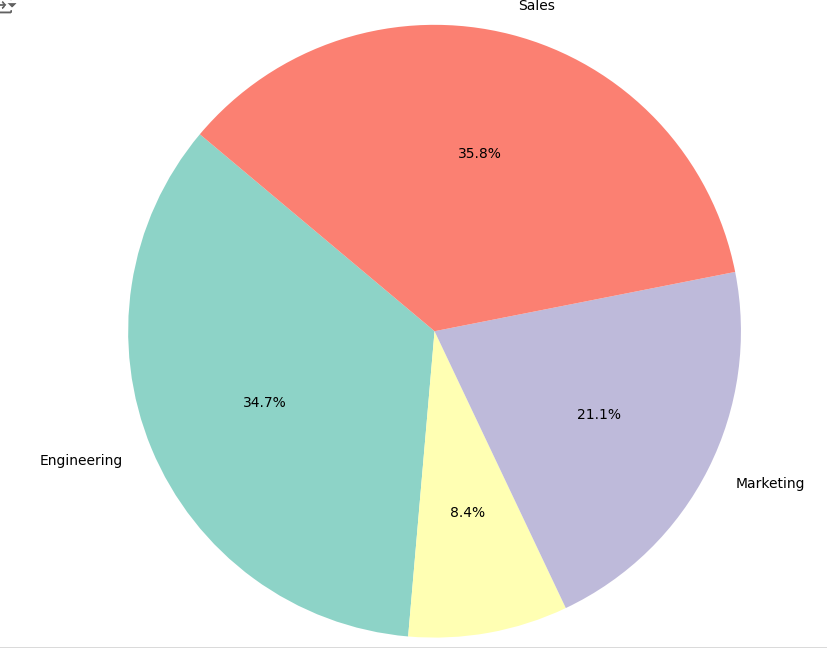
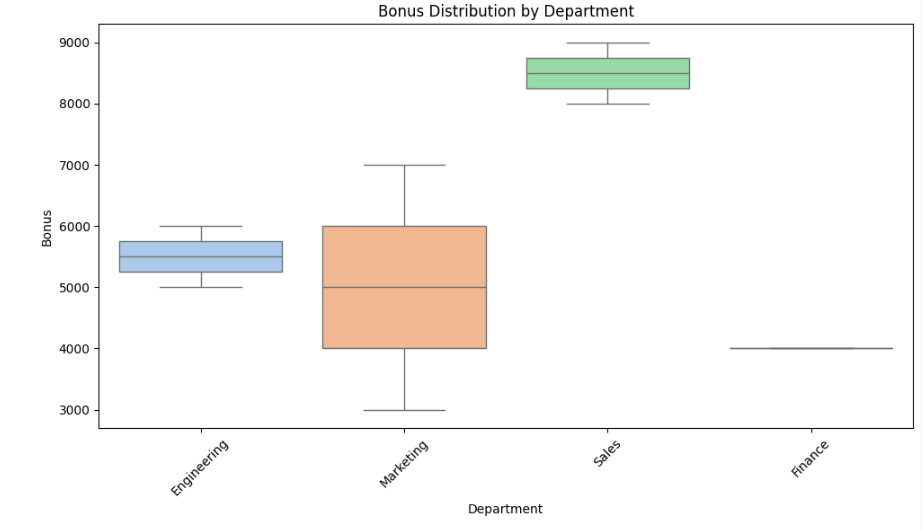
( Two graphs or plotted for each script )

**Calculate average base salary by department**



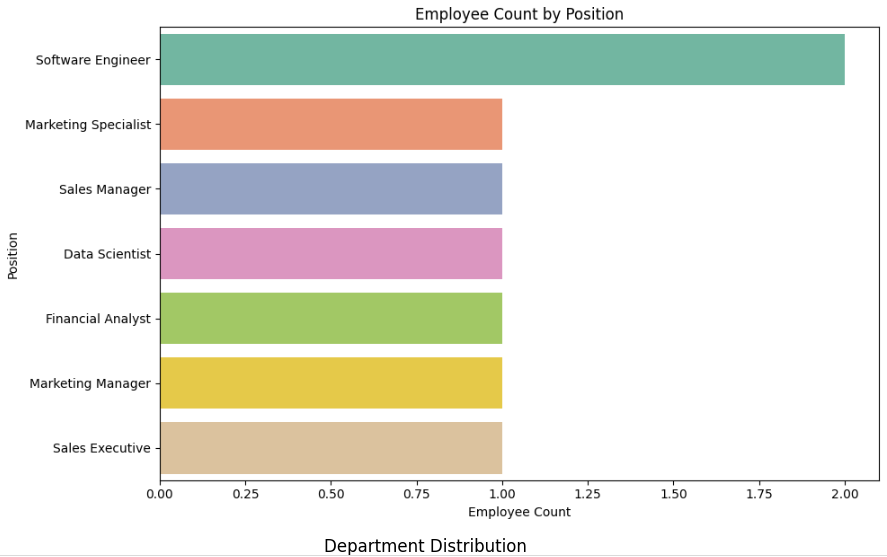
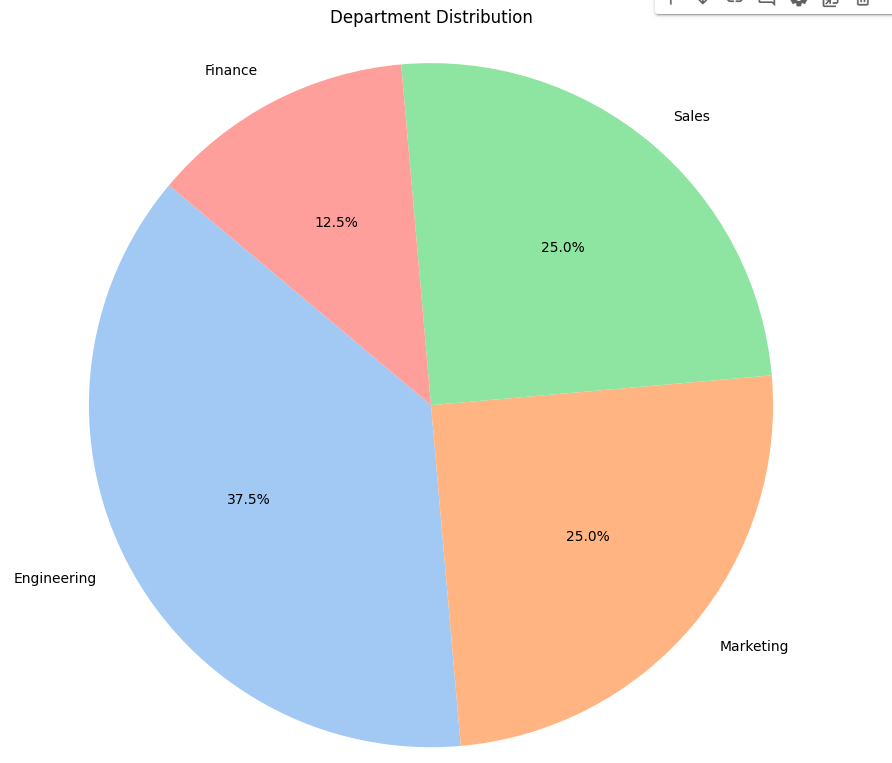
(Bar Plots)

**Calculate total bonus by department**

(Pie Chart) (Box Plot)

**Count of employees by position**

(Bar Plot) (Pie Chart)

**Step 5 : Cloud Service Integration**

Cloud service integration typically involves connecting various cloud services within a provider's ecosystem (e.g., AWS, Azure, GCP) to create cohesive workflows and data pipelines. For instance, in AWS, integration could start with data ingestion into Amazon S3 from sources like IoT devices or web applications. AWS Glue can then automate data preparation and transformation tasks, sending processed data to Amazon Redshift or Amazon Athena for analysis. Visualizing insights can be done using Amazon QuickSight, which directly queries data stored in S3 or data warehouses. This seamless integration streamlines the end-to-end data processing lifecycle, from ingestion to visualization, leveraging the strengths of each service for scalable and efficient analytics.

Several Benefits include:

Using cloud services like AWS for data analysis offers several benefits. Firstly, it provides scalable infrastructure, allowing you to handle varying data volumes efficiently without upfront investments in hardware. Services like AWS Glue simplify data processing with managed ETL capabilities, automating tasks like data transformation and loading. Storage solutions such as Amazon S3 ensure durability and accessibility for your data, while services like Amazon QuickSight enable quick and interactive visualization of insights. Overall, leveraging cloud services reduces operational overhead, enhances flexibility, and accelerates time-to-insight for data-driven decision making.

**Step 6: Machine Learning Model**

The **Random Forest Regressor** was chosen for this model due to its ability to handle high-dimensional data and capture complex non-linear relationships. It is robust against overfitting through the aggregation of multiple decision trees, each trained on different random subsets of the data. This method also manages missing values effectively and provides insights into feature importance. Furthermore, random forests are scalable and versatile, suitable for both classification and regression tasks, making them an ideal choice for creating accurate and reliable predictive models on diverse datasets.

RESULTS OF THE MODEL:

R – squared: 0.9677119223425846

Mean square error: 8926756.581229301