SQL and Machine learning Task

Done by: Dareen Fawwaz

Table of Contents

Table of Tables

Table of Figures

Part A: SQL:

1. Load the data:

I used sqlite as my database and sqlite browser as the interface:

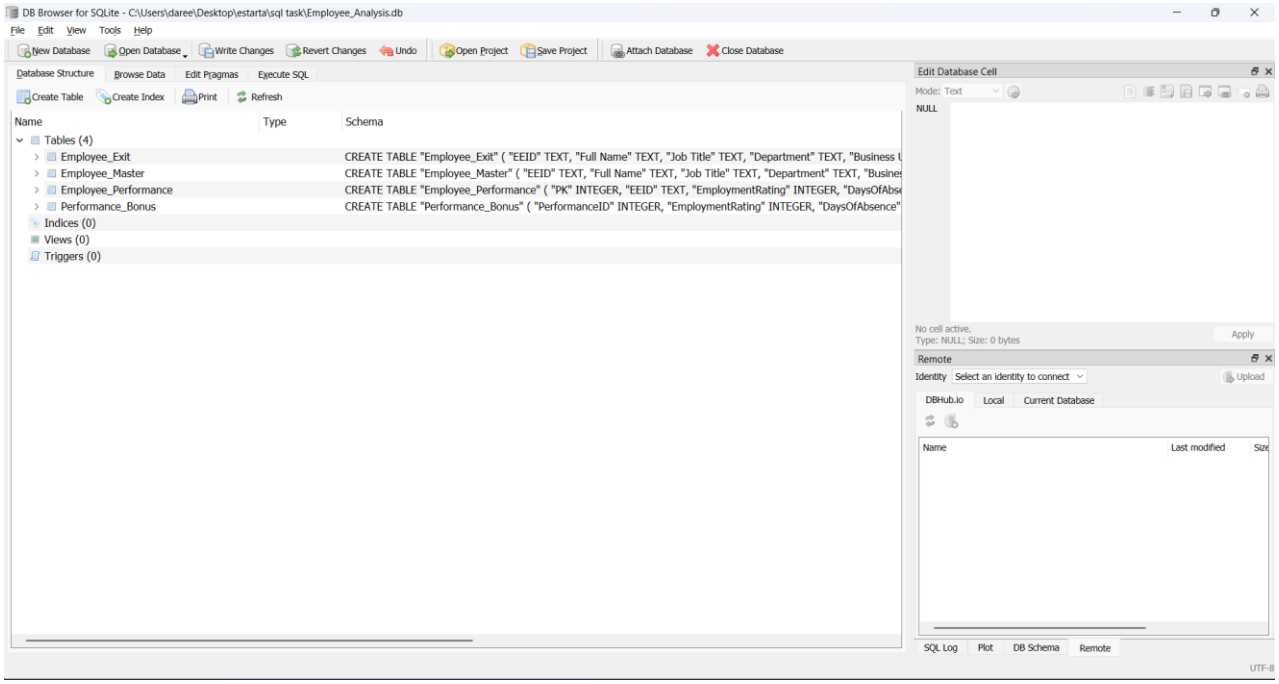


Figure 1: load data

2. Know your data:

a. Count total rows in employee:
TotalEmployeesNo = 1000

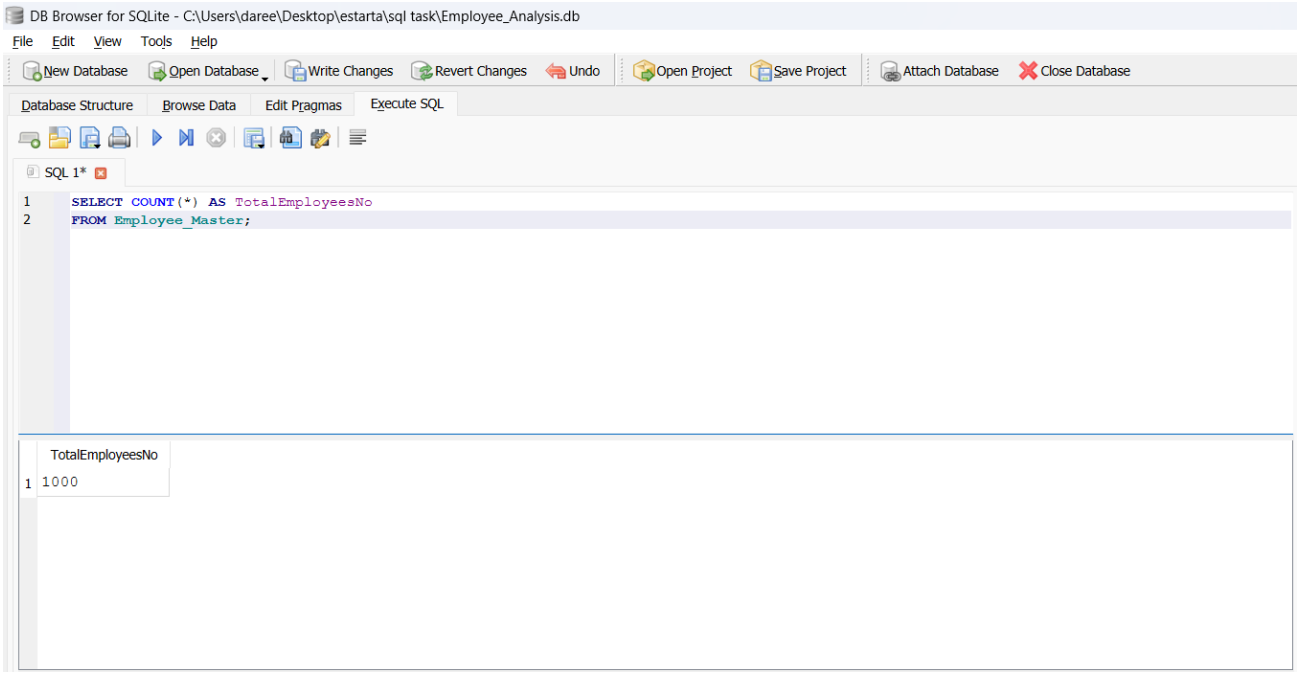


Figure 2: total employees no.

b. Count rows per Department:
I have found that 2 employees are not listed for a department and the highest number of employees are under the IT department

```
SELECT Department, COUNT(*) AS EmployeesCount
FROM Employee_Master
GROUP BY Department
ORDER BY EmployeesCount;
```

Table 1: rows count per dept

	2
Accounting	96
Finance	120
Marketing	120
Human Resources	124
Sales	140
Engineering	158
IT	240

c. Count how many nulls appear in ExitDate:
I found 915 null record that means that 915 employees are still working

```
SELECT COUNT(*) AS ActiveEmployees
FROM Employee_Master
WHERE "Exit Date" IS NULL;
```

Table 2: null count in exitdate

	WorkingEmployees
1	915

d. Age range of employees:

```
SELECT MIN(Age) AS MinAge, MAX(Age) AS MaxAge FROM Employee_Master;
```

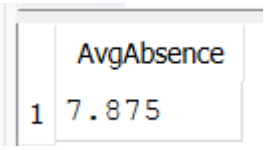
Table 3: age range

	MinAge	MaxAge
1	25	65

e. Days of Absence:

```
SELECT AVG(DaysOfAbsence) AS AvgAbsence FROM
Employee_Performance;
```

Table 4: avg absence



	AvgAbsence
1	7.875

3. Merge tables:

```
CREATE TABLE Employee_Profile AS
SELECT

table1.*,table2.EmploymentRating,table2.DaysOfAbsence,table2.EducationLevel,
table2.CertificationsEarned
FROM
   Employee_Master AS table1
INNER JOIN
   Employee_Performance AS table2 ON table1.EEID = table2.EEID;
```

- Here I have found that there is multiple duplicated EEID in both tables so I dropped from the employee_master duplicates and kept first occurrence and kept the highest rating from employee_preformance

4. Insert sample records:

```
INSERT INTO Employee_Profile (EEID, "Full Name", "Job Title", Department, Gender,
Age, "Annual Salary", "Bonus %", Country, City, "Exit Date", EmploymentRating,
DaysOfAbsence, EducationLevel, CertificationsEarned)
VALUES
('E1001', 'Dareen Fawwaz', 'Senior Developer', 'IT', 'Female', 20, 135000.00, 0.20,
'Jordan', 'Amman', NULL, 5, 2, 'Master', 3);
INSERT INTO Employee_Profile (EEID, "Full Name", "Job Title", Department, Gender,
Age, "Annual Salary", "Bonus %", Country, City, "Exit Date", EmploymentRating,
DaysOfAbsence, EducationLevel, CertificationsEarned)
VALUES
('E1002', 'Ahmad Ahmad', 'Financial Analyst', 'Finance', 'Male', 27, 85000.00, 0.08,
'Canada', 'Toronto', NULL, 3, 7, 'Bachelor', 1);
INSERT INTO Employee_Profile (EEID, "Full Name", "Job Title", Department, Gender,
Age, "Annual Salary", "Bonus %", Country, City, "Exit Date", EmploymentRating,
DaysOfAbsence, EducationLevel, CertificationsEarned)
VALUES
```

('E1003', 'Mohammad Mohammad', 'HR Assistant', 'HR', 'Male', 23, 55000.00, 0.00, 'USA', 'Phoenix', NULL, 3, 10, 'Associate', 0);
INSERT INTO Employee_Profile (EEID, "Full Name", "Job Title", Department, Gender, Age, "Annual Salary", "Bonus %", Country, City, "Exit Date", EmploymentRating, DaysOfAbsence, EducationLevel, CertificationsEarned)
VALUES
('E1004', 'Sara Ahmad', 'IT Support Specialist', 'IT', 'Female', 45, 95000.00, 0.10, 'USA', 'Phoenix', NULL, 4, 4, 'Master', 2);
INSERT INTO Employee_Profile (EEID, "Full Name", "Job Title", Department, Gender, Age, "Annual Salary", "Bonus %", Country, City, "Exit Date", EmploymentRating, DaysOfAbsence, EducationLevel, CertificationsEarned)
VALUES
('E1005', 'Maria Rami', 'Marketing Specialist', 'Marketing', 'Female', 30, 70000.00, 0.05, 'Mexico', 'Mexico City', NULL, 2, 20, 'Bachelor', 1);
INSERT INTO Employee_Profile (EEID, "Full Name", "Job Title", Department, Gender, Age, "Annual Salary", "Bonus %", Country, City, "Exit Date", EmploymentRating, DaysOfAbsence, EducationLevel, CertificationsEarned)
VALUES
('E1006', 'Dalia Ahmad', 'Sales Director', 'Sales', 'Male', 52, 160000.00, 0.30, 'UK', 'London', NULL, 4, 3, 'Master', 3);
INSERT INTO Employee_Profile (EEID, "Full Name", "Job Title", Department, Gender, Age, "Annual Salary", "Bonus %", Country, City, "Exit Date", EmploymentRating, DaysOfAbsence, EducationLevel, CertificationsEarned)
VALUES
('E1007', 'Tala Mohammad', 'Data Scientist', 'IT', 'Female', 29, 110000.00, 0.15, 'Germany', 'Berlin', '2025-01-15', 5, 0, 'PhD', 2);
INSERT INTO Employee_Profile (EEID, "Full Name", "Job Title", Department, Gender, Age, "Annual Salary", "Bonus %", Country, City, "Exit Date", EmploymentRating, DaysOfAbsence, EducationLevel, CertificationsEarned)
VALUES
('E1008', 'Rami Ahmad', 'Logistics Coordinator', 'Operations', 'Male', 48, 65000.00, 0.00, 'USA', 'Dallas', NULL, 3, 15, 'Bachelor', 1);
INSERT INTO Employee_Profile (EEID, "Full Name", "Job Title", Department, Gender, Age, "Annual Salary", "Bonus %", Country, City, "Exit Date", EmploymentRating, DaysOfAbsence, EducationLevel, CertificationsEarned)
VALUES
('E1009', 'Lina Rami', 'Treasury Manager', 'Finance', 'Female', 38, 105000.00, 0.12, 'France', 'Paris', NULL, 4, 6, 'Master', 0);
INSERT INTO Employee_Profile (EEID, "Full Name", "Job Title", Department, Gender, Age, "Annual Salary", "Bonus %", Country, City, "Exit Date", EmploymentRating, DaysOfAbsence, EducationLevel, CertificationsEarned)
VALUES
('E1010', 'Alia Ahmad', 'Junior DevOps', 'IT', 'Male', 24, 68000.00, 0.05, 'USA', 'Phoenix', NULL, 4, 5, 'Bachelor', 1);

5. Conditional update:

```
UPDATE Employee_Profile
SET City = 'Abu Dhabi'
WHERE
   City = 'Phoenix'
   AND Gender = 'Male'
   AND Department = 'IT';
```

Table 5: city equal "Abu Dhabi"

E00523	Daniel Jordan	Network Administrator	IT	Corporate	Male	Caucasian	58	7/26/1993	$69,260	0	United States	Abu Dhabi		8	15	Bachelor's	3
E01722	Asher Huynh	Manager	IT	Manufacturing	Male	Asian	45	1/22/2015	$101,288	0.1	United States	Abu Dhabi		10	9	Bachelor's	4
E1010	Alia Ahmad	Junior DevOps	IT		Male		24		68000.0	0.05	USA	Abu Dhabi		4	5	Bachelor	1

6. Export CSVs:

First check departments in table the result is equal 10 departments:

```
SELECT DISTINCT Department FROM Employee_Profile;
```

Table 6: departments

IT
Finance
Sales
Accounting
Human Resources
Engineering
Marketing
NULL
HR
Operations

then I proceeded with this query for every department and exported it into an CSV file and merged the HR and Human Resources and filled the null department with "Unassigned":

```
SELECT * FROM Employee_Profile WHERE Department = 'IT';
```

Part B: Python And ML:

1. Prepare the IT dataset:

- I imported these libraries to help with machine learning build:

```
import pandas as pd

import numpy as np

from sklearn.model_selection import train_test_split

from sklearn.preprocessing import StandardScaler, OneHotEncoder

from sklearn.compose import ColumnTransformer

from sklearn.pipeline import Pipeline

from sklearn.linear_model import LinearRegression

from sklearn.metrics import mean_absolute_error, mean_squared_error

from datetime import datetime

import re
```

- Clean data and drop exit column:

```
IT_DATA.columns = IT_DATA.columns.str.replace(r'[^\w\s]', '', regex=True) #clean column names
IT_DATA.rename(columns={'Bonus ': 'Cur_Bonus'}, inplace=True) # Cur_Bonus = Current_Bonus_Ratio
IT_DATA.rename(columns={'Exit Date': 'Init_Exit'}, inplace=True)
IT_DATA['Annual Salary'] = IT_DATA['Annual Salary'].replace({r'[$,]': ''}, regex=True).astype(float) #remove dollar sign
IT_DATA['Cur_Bonus'] = IT_DATA['Cur_Bonus'].astype(float)
IT_DATA.drop('Init_Exit', axis=1, inplace=True)
IT_DATA = IT_DATA[~IT_DATA['EEID'].str.contains('E100', na=False)].reset_index(drop=True) #filter employees
```

Figure 3: code 1

- Clean the columns names
- Renamed the bonus column to cur_bonus (current) and exit data to init_exit
- Removed the dollar sign from anuual salary column
- Changed bonus datatype into float
- Dropped the exit date column
- Filtered the employees by EEID

```
[27]: num_feats = ['Age', 'Annual Salary', 'Cur_Bonus', 'EmploymentRating', 'DaysOfAbsence', 'CertificationsEarned']
for feat in num_feats:
    if IT_DATA[feat].isnull().any():
        med_val = IT_DATA[feat].median() #because of outlires
        IT_DATA[feat].fillna(med_val, inplace=True)
for col in IT_DATA.columns:
    if IT_DATA[col].dtype == 'object' and IT_DATA[col].isnull().any():
        mode_val = IT_DATA[col].mode()[0]
        IT_DATA[col].fillna(mode_val, inplace=True)

print("Data Cleanup complete. IT dataset ready.")
print("Total IT records:", len(IT_DATA))
```

```
Data Cleanup complete. IT dataset ready.
Total IT records: 215
```

Figure 4: code 2

- Filled numerical columns with median to avoid outliers
- Filled object data type columns with mode value

2. Exit-Date Prediction:

- Load and clean data:

```
Exit_Data = pd.read_csv('Employee_Exit.csv')
```

```
Exit_Data.rename(columns={'Bonus %': 'Bonus_Train'}, inplace=True)
```

```
Exit_Data['Annual Salary'] = Exit_Data['Annual Salary'].replace({r'[$,]': ''}, regex=True).astype(float)
Exit_Data['Bonus_Train'] = Exit_Data['Bonus_Train'].replace({r'%': ''}, regex=True).astype(float) / 100
```

```
DT_FMT = '%m/%d/%Y'
Exit_Data['Hire Date'] = pd.to_datetime(Exit_Data['Hire Date'], format=DT_FMT, errors='coerce')
Exit_Data['Exit Date'] = pd.to_datetime(Exit_Data['Exit Date'], format=DT_FMT, errors='coerce')
Exit_Data['Tenure_Days'] = (Exit_Data['Exit Date'] - Exit_Data['Hire Date']).dt.days
```

```
Exit_Data.dropna(subset=['Tenure_Days', 'Annual Salary', 'Bonus_Train'], inplace=True)
```

Figure 5: code 3

- Removed dollar sign from annual salary and % sign from bonus
- Renamed bonus to bonus_train
- Reformatted the hire date and exit date into date format
- Added a column that calculates the difference between hire date and exit date
- Dropped null and nan values from features columns

```
X_COLS = ['Age', 'Annual Salary', 'Bonus_Train', 'Country', 'Gender',
          'Job Title', 'Department', 'Business Unit']
Y_TARG = 'Tenure_Days'
X_train = Exit_Data[X_COLS]
Y_train = Exit_Data[Y_TARG]
```

Figure 6: code 4

- The chosen features are:

X_COLS = ['Age', 'Annual Salary', 'Bonus_Train', 'Country', 'Gender', 'Job Title', 'Department', 'Business Unit']

- And the target variable is tenure_days(difference between hire and exit dates)

```
# apply different preprocessing steps to different subsets of columns
preproc = ColumnTransformer(
    transformers=[
        ('scale', StandardScaler(), ['Age', 'Annual Salary', 'Bonus_Train']), #scales the numerical features so that they have a mean of 0 and a standard
        ('encode', OneHotEncoder(handle_unknown='ignore'), ['Country', 'Gender', 'Job Title', 'Department', 'Business Unit']) #converts categorical text
    ]
)
```

```
X_T, X_V, Y_T, Y_V = train_test_split(X_train, Y_train, test_size=0.2, random_state=42)
```

Figure 7: code 5

- apply different pre-processing steps to different subsets of columns
- scales the numerical features so that they have a mean of 0 and a standard deviation of 1
- converts categorical text data into a numerical format that machine learning models can understand
- then split data into 20% validation and 80% training

```
#chain together multiple data processing and modeling steps so that they can be treated as a single unit.
model_lin = Pipeline(steps=[
    ('preproc', preproc),
    ('reg', LinearRegression()) ])
#model is trained using the preprocessed data outputted by the previous step (preproc) and the target variable ($Y$)
```

```
model_lin.fit(X_T, Y_T)
print("Model trained: Linear Regression for Tenure.")
```

```
Model trained: Linear Regression for Tenure.
```

Figure 8: code 6

- chained together multiple data processing and modelling steps so that they can be treated as a single unit

- train data using preproc defined function and linear regression

```
preds_V = model_lin.predict(X_V)
preds_V[preds_V < 0] = 0
```

Figure 9: code 7

- predict and if any negative prediction set it to zero

```
err_mae = mean_absolute_error(Y_V, preds_V) #measures the average magnitude of the errors in a set of predictions.
err_mse = mean_squared_error(Y_V, preds_V) #measures the average of the squares of the errors
err_rmse = np.sqrt(err_mse)
```

```
print(f"\nModel Performance:")
print(f"MAE: {err_mae:.2f} days")
print(f"RMSE: {err_rmse:.2f} days")
```

```
Model Performance:
MAE: 866.06 days
RMSE: 997.81 days
```

Figure 10: code 8

- calculate the mean absolute error and mean squared error and take the square root of it

```
IT_INPUT = IT_DATA.copy()
```

```
IT_INPUT.rename(columns={'Cur_Bonus': 'Bonus_Train'}, inplace=True)
```

```
pred_tenure = model_lin.predict(IT_INPUT[X_COLS])
pred_tenure[pred_tenure < 0] = 0
```

```
IT_INPUT['Hire Date'] = pd.to_datetime(IT_INPUT['Hire Date'], format=DT_FMT, errors='coerce')
tenure_td = pd.to_timedelta(pred_tenure, unit='D')
pred_exit_dt = IT_INPUT['Hire Date'] + tenure_td
```

```
IT_DATA['Predicted_ExitDate'] = pred_exit_dt.dt.strftime(DT_FMT)
```

```
print(f"\nPrediction Complete. Predicted Exit Dates appended to IT_DATA.")
print("--- Sample Predictions ---")
print(IT_DATA[['Full Name', 'Hire Date', 'Predicted_ExitDate']].head().to_string())
```

```
Prediction Complete. Predicted Exit Dates appended to IT_DATA.
--- Sample Predictions ---
          Full Name  Hire Date Predicted_ExitDate
0  Lillian Gonzales  3/13/2009         05/10/2014
1  Scarlett Jenkins  11/9/2011         06/21/2016
2  Brooklyn Salazar   3/1/2011         03/10/2016
3       Riley Rojas  1/21/2021         01/16/2026
4    Isabella Scott  4/26/2016         06/01/2021
```

Figure 11: code 9

- Predict exit dates and append them as a new column ExitDate to Employee_Profile _IT

3. Bonus Percentage Prediction:

- Load and clean data:

```
EMP_BONUS_DATA = pd.read_csv('Performance_Bonus.csv')
```

```
EMP_BONUS_DATA.rename(columns={'Bouns': 'Bonus_Amt'}, inplace=True) #bonus amount
EMP_BONUS_DATA['Bonus_Amt'] = EMP_BONUS_DATA['Bonus_Amt'].str.replace('%','').astype(float) / 100
```

```
level_map = {"Bachelor's": 0, "Master's": 1, "Doctorate": 2}
EMP_BONUS_DATA['EL_EN'] = EMP_BONUS_DATA['EducationLevel'].map(level_map)
```

```
Q1, Q3 = np.percentile(EMP_BONUS_DATA['Bonus_Amt'], [25, 75])
IQR = Q3 - Q1
Upper_Bound = Q3 + 1.5 * IQR
EMP_BONUS_DATA = EMP_BONUS_DATA[(EMP_BONUS_DATA['Bonus_Amt'] <= Upper_Bound)]
```

Figure 12: code 10

- Load the preformane_bonus data
- Renamed the bonus into bonus_amt (amount) and removed the dollar sign
- Mapped (encode) the education level into numerical values
- Remove outlies from bonus column

```
X_COLS = ['EmploymentRating', 'DaysOfAbsence', 'CertificationsEarned', 'EL_EN']
Y_TARG = 'Bonus_Amt'
```

```
X_train_bonus = EMP_BONUS_DATA[X_COLS]
Y_train_bonus = EMP_BONUS_DATA[Y_TARG]
```

```
X_T, X_V, Y_T, Y_V = train_test_split(X_train_bonus, Y_train_bonus, test_size=0.25, random_state=35)
```

Figure 13: code 11

- the chosen features are:

X_COLS = ['EmploymentRating', 'DaysOfAbsence', 'CertificationsEarned', 'EL_EN']

- And the target variable is bonus amount
- Split data into 25% validation and 75% training

```
model_bonus = LinearRegression()
model_bonus.fit(X_T, Y_T)
Y_pred_V = model_bonus.predict(X_V)
```

Figure 14: code 12

- Preform prediction functions

```
mae_bonus = mean_absolute_error(y_true=Y_V, y_pred=Y_pred_V)
mse_bonus = mean_squared_error(y_true=Y_V, y_pred=Y_pred_V)
rmse_bonus = np.sqrt(mse_bonus)
```

```
print(f"MAE: {mae_bonus:.4f} (Ratio)")
print(f"RMSE: {rmse_bonus:.4f} (Ratio)")
print(f"Training Mean Bonus: {(EMP_BONUS_DATA['Bonus_Amt']).mean():.4f} (Ratio)")
```

```
MAE: 0.1030 (Ratio)
RMSE: 0.1174 (Ratio)
Training Mean Bonus: 0.2019 (Ratio)
```

Figure 15: code 13

- calculate mean absolute error and mean squared error

```
IT_DATA['Predicted Bonus'] = model_bonus.predict(IT_DATA[X_COLS])
```

```
IT_DATA.drop(['EL_EN'], axis=1, inplace=True)
```

```
print("\n--- Predicted Bonus for IT Employees ---")
print(IT_DATA[['Full Name', 'Cur_Bonus', 'Predicted Bonus']].head().to_string())
```

```
--- Predicted Bonus for IT Employees ---
          Full Name  Cur_Bonus  Predicted Bonus
0  Lillian Gonzales     0.0000         0.199100
1  Scarlett Jenkins     0.0032         0.204686
2  Brooklyn Salazar     0.0000         0.206017
3       Riley Rojas     0.0000         0.207836
4    Isabella Scott     0.0000         0.197077
```

Figure 16: code 14

- Append predictions in a new column Predicted_bonus to IT dataset

```
count_under_rewarded = 0
count_zero_bonus = 0
```

```
for actual_bonus, predicted_bonus in zip(IT_DATA['Cur_Bonus'], IT_DATA['Predicted Bonus']):
    if actual_bonus == 0.0:
        count_zero_bonus += 1

    if predicted_bonus > actual_bonus:
        count_under_rewarded += 1

print(f"Employees who received less actual bonus than predicted: {count_under_rewarded} ({count_under_rewarded / len(IT_DATA) * 100:.2f}%)")
print(f"Employees with zero actual bonus: {count_zero_bonus} ({count_zero_bonus / len(IT_DATA) * 100:.2f}%)")
```

```
Employees who received less actual bonus than predicted: 215 (100.00%)
Employees with zero actual bonus: 147 (68.37%)
```

Figure 17: code 15

- Employees who received less actual bonus than predicted: 215 (100 .00%)
- Employees with zero actual bonus: 147 (68.37%)
- According to numbers and percentages all employees is predicted to receive a bonus amount greater than what they already have and most employees in original dataset have zero bonuses