TASK_1 - HR DATA ANALYTICS

September 30, 2024

1 TASK 1 : HR DATA ANALYTICS

2 1. Connect Notebook with MySQL Database

[1]:
```
# Import required libraries
import pandas as pd
import mysql.connector
```

[2]:
```
# MySQL database connection details
host = 'YOUR_HOST_NAME'
user = 'YOUR_USER_NAME'
password = 'YOUR_PASSWORD'
```

[3]:
```
# CSV file path
csv_file = 'hr_data.csv'  # Replace 'your_file.csv' with your CSV file path

# Read the first row of the CSV to extract column names
with open(csv_file, 'r') as file:
    first_line = file.readline().strip()  # Read the first line
    column_names = first_line.split(',')  # Assuming columns are comma-separated
    file.close()
```

[4]:
```
# Create MySQL connection
conn = mysql.connector.connect(host=host, user=user, password=password)
cursor = conn.cursor()
```

[6]:
```
# Name of database
database = str(input("Enter database name: "))

# Create the database if it doesn't exist
create_db_query = f"CREATE DATABASE IF NOT EXISTS {database}"
cursor.execute(create_db_query)

# Close the connection as the database is created
conn.close()
```

```
Enter database name: hranalytics
```

[7]:
```
# Reconnect to the newly created or existing database
conn = mysql.connector.connect(host=host, user=user, password=password,␣
 ↪database=database)
cursor = conn.cursor()
```

[7]:
```
# Name of table
table_name = str(input("Enter table name: "))

# Create table with extracted column names
create_table_query = f"CREATE TABLE IF NOT EXISTS {table_name} ({', '.
 ↪join([f'{col} TEXT' for col in column_names])})"
cursor.execute(create_table_query)
```

```
Enter table name: hr_table
```

[8]:
```
# Read the CSV file into a pandas DataFrame
df = pd.read_csv(csv_file)
df.head()
```

[8]:

	EmpName	Age	Attrition	BusinessTravel	Department	\
0	ALBERTO PEDRUCO	51	No	Travel_Rarely	Sales	
1	LAWRENCE LEE	31	Yes	Travel_Frequently	Research & Development	
2	DWAYNE CURRY	32	No	Travel_Frequently	Research & Development	
3	RENEE MARQUARDT	38	No	Non-Travel	Research & Development	
4	HARVEY ELWIN	32	No	Travel_Rarely	Research & Development	

	DistanceFromHome	Education	EducationField	EmployeeCount	EmployeeID	…	\
0	6	2	Life Sciences	1	1	…	
1	10	1	Life Sciences	1	2	…	
2	17	4	Other	1	3	…	
3	2	5	Life Sciences	1	4	…	
4	10	1	Medical	1	5	…	

	NumCompaniesWorked	Over18	PercentSalaryHike	StandardHours	\
0	1.0	Y	11	8	
1	0.0	Y	23	8	
2	1.0	Y	15	8	
3	3.0	Y	11	8	
4	4.0	Y	12	8	

	StockOptionLevel	TotalWorkingYears	TrainingTimesLastYear	YearsAtCompany	\
0	0	1.0	6	1	
1	1	6.0	3	5	
2	3	5.0	2	5	
3	3	13.0	5	8	
4	2	9.0	2	6	

```
   YearsSinceLastPromotion  YearsWithCurrManager
0                        0                     0
1                        1                     4
2                        0                     3
3                        7                     5
4                        0                     4

[5 rows x 25 columns]
```

[9]:
```
# Checking shape of dataset
df.shape
```

[9]:
```
(4410, 25)
```

[10]:
```
# Checking column names
df.columns
```

[10]:
```
Index(['EmpName', 'Age', 'Attrition', 'BusinessTravel', 'Department',
       'DistanceFromHome', 'Education', 'EducationField', 'EmployeeCount',
       'EmployeeID', 'Gender', 'JobLevel', 'JobRole', 'MaritalStatus',
       'MonthlyIncome', 'NumCompaniesWorked', 'Over18', 'PercentSalaryHike',
       'StandardHours', 'StockOptionLevel', 'TotalWorkingYears',
       'TrainingTimesLastYear', 'YearsAtCompany', 'YearsSinceLastPromotion',
       'YearsWithCurrManager'],
      dtype='object')
```

[11]:
```
# Insert data into the created table
for _, row in df.iterrows():
    # Handle NaN values by converting them to None
    row = [None if pd.isna(value) else value for value in row]

    # Prepare and execute the insert query
    insert_query = f"INSERT INTO {table_name} ({', '.join(column_names)})␣
 ↪VALUES ({', '.join(['%s'] * len(column_names))})"
    cursor.execute(insert_query, tuple(row))

# Commit changes and close the connection
conn.commit()
conn.close()

print(f"Table '{table_name}' created and data loaded successfully.")
```

```
Table 'hr_table' created and data loaded successfully.
```

3 2. Excess Databse Information as per your Requirement

[11]:
```
# Pass the query through this function ant get your results
from IPython.display import display

def execute_query(query):

    # MySQL database connection details
    host = 'YOUR_HOST_NAME'
    user = 'YOUR_USER_NAME'
    password = 'YOUR_PASSWORD'
    database = 'hranalytics'  # Replace with your database name

    # Connect to MySQL
    conn = mysql.connector.connect(host=host, user=user, password=password,␣
 ↪database=database)

    # Create a cursor
    cursor = conn.cursor()

    try:
        # Execute the query
        cursor.execute(query)

        # Fetch all results
        results = cursor.fetchall()


        # Display results in a table format
        if results:
            header = [desc[0] for desc in cursor.description]
            data = list(results)
            df = pd.DataFrame(data, columns=header)
            display(df)
        else:
            print("No results found.")

    except mysql.connector.Error as error:
        print(f"Error: {error.msg}")

    finally:
        # Close the cursor and connection
        cursor.close()
        conn.close()
```

3.1 1. Retrieve the total number of employees in the dataset.

[12]:
```
# Define the query
query = "SELECT COUNT(*) AS Total_Employees FROM hr_table"
# Call the function with the query
execute_query(query)
```

```
   Total_Employees
0             4410
```

3.2 2. List all unique job roles in the dataset.

[13]:
```
# Define the query
query = "SELECT DISTINCT JobRole AS Job_Role FROM hr_table;"
# Call the function with the query
execute_query(query)
```

```
                    Job_Role
0  Healthcare Representative
1          Research Scientist
2             Sales Executive
3             Human Resources
4           Research Director
5       Laboratory Technician
6      Manufacturing Director
7         Sales Representative
8                     Manager
```

3.3 3. Find the average age of employees.

[14]:
```
# Define the query
query = "SELECT AVG(Age) AS Average_Age FROM hr_table"
# Call the function with the query
execute_query(query)
```

```
   Average_Age
0     36.92381
```

3.4 4. Retrieve the names and ages of employees who have worked at the company for more than 5 years

[15]:
```
# Define the query
query = "SELECT EmpName, Age FROM hr_table WHERE YearsAtCompany > 5;"
# Call the function with the query
execute_query(query)
```

```
                    EmpName Age
0              RENEE MARQUARDT  38
```

```
1                  HARVEY ELWIN  32
2                    LEON WHITE  46
3                  NATHAN HARDY  31
4              SUSAN BUCHBINDER  25
…                            …  ..
2077             GLACIER YBANEZ  34
2078  ISABELLE ALLOUKO FIANKAN  39
2079              STEVEN PONDER  29
2080              CHRISTINE MAY  42
2081              CLAIRE WHALEY  40

[2082 rows x 2 columns]
```

3.5 5. Get a count of employees grouped by their department.

[16]:
```
# Define the query
query = "SELECT Department, COUNT(*) AS employee_count FROM hr_table GROUP BY␣
 ↪Department;"
# Call the function with the query
execute_query(query)
```

```
               Department  employee_count
0                   Sales            1338
1  Research & Development            2883
2         Human Resources             189
```

3.6 6. List employees who have 'High' Job Satisfaction.

[17]:
```
df['JobLevel'].value_counts()
```

[17]:
```
1    1629
2    1602
3     654
4     318
5     207
Name: JobLevel, dtype: int64
```

[18]:
```
# Define the query
query = "SELECT EmployeeID, EmpName AS EMP_NAME_HIGH FROM hr_table WHERE␣
 ↪JobLevel = 3;"
# Call the function with the query
execute_query(query)
```

```
    EmployeeID     EMP_NAME_HIGH
0            4   RENEE MARQUARDT
1            9      NATHAN HARDY
2           31  ROSELYN JEQUINTO
3           34    EUGENE GALEANO
```

```
4            38   JAMES VANNUCCHI
..          …               …
649        4377       JAMES TERRY
650        4378        GRACE KWAK
651        4391    ANDREA KOZIMOR
652        4392          JOE TONG
653        4402     ABUBAKER AZAM

[654 rows x 2 columns]
```

3.7 7. Find the highest Monthly Income in the dataset.

[19]:
```
# Define the query
query = "SELECT MAX(MonthlyIncome) AS Highest_Income FROM hr_table;"
# Call the function with the query
execute_query(query)
```

```
  Highest_Income
0          99980
```

3.8 8. List employees who have 'Travel_Rarely' as their BusinessTravel type.

[20]:
```
# Define the query
query = "SELECT EmployeeID, EmpName FROM hr_table WHERE BusinessTravel =␣
 ↪'Travel_Rarely';"
# Call the function with the query
execute_query(query)
```

```
     EmployeeID          EmpName
0             1  ALBERTO PEDRUCO
1             5     HARVEY ELWIN
2             6       LEON WHITE
3             7   DENNIS HERRERA
4             8    DONALD BRYANT
…          …               …
3124       4406  MERRICK PASCUAL
3125       4407   JENNIFER CHING
3126       4408      KARTIK SHAH
3127       4409    CHRISTINE MAY
3128       4410    CLAIRE WHALEY

[3129 rows x 2 columns]
```

3.9 9. Retrieve the distinct MaritalStatus categories in the dataset.

[21]:
```
# Define the query
query = "SELECT DISTINCT MaritalStatus FROM hr_table;"
# Call the function with the query
execute_query(query)
```

```
  MaritalStatus
0       Married
1        Single
2      Divorced
```

3.10 10. Get a list of employees with more than 2 years of work experience but less than 4 years in their current role.

[22]:
```
df.columns
```

[22]:
```
Index(['EmpName', 'Age', 'Attrition', 'BusinessTravel', 'Department',
       'DistanceFromHome', 'Education', 'EducationField', 'EmployeeCount',
       'EmployeeID', 'Gender', 'JobLevel', 'JobRole', 'MaritalStatus',
       'MonthlyIncome', 'NumCompaniesWorked', 'Over18', 'PercentSalaryHike',
       'StandardHours', 'StockOptionLevel', 'TotalWorkingYears',
       'TrainingTimesLastYear', 'YearsAtCompany', 'YearsSinceLastPromotion',
       'YearsWithCurrManager'],
      dtype='object')
```

[23]:
```
# Define the query
query = "SELECT EmployeeID, EmpName FROM hr_table WHERE TotalWorkingYears > 2␣
 ↪AND YEARSATCOMPANY > 2 AND YearsWithCurrManager < 4 AND␣
 ↪YearsSinceLastPromotion <4;"
# Call the function with the query
execute_query(query)
```

```
     EmployeeID           EmpName
0             3       DWAYNE CURRY
1            16       LUIS HERRERA
2            17     GEORGE FOURAS
3            18    MARTIN LALOR JR
4            23        OLLIE BANKS
…           …                …
1317       4401  STEPHANIE JOHNSON
1318       4402     ABUBAKER AZAM
1319       4406    MERRICK PASCUAL
1320       4407     JENNIFER CHING
1321       4408        KARTIK SHAH

[1322 rows x 2 columns]
```

3.11 11. List employees who have changed their job roles within the company (JobLevel and JobRole differ from their previous job).

[24]:
```
# Define the query
query = "SELECT␣
 ↪EmployeeID,EmpName,CurrentJobRole,PreviousJobRole,CurrentJobLevel,PreviousJobLevel␣
 ↪FROM(SELECT EmployeeID, EmpName, JobRole AS CurrentJobRole, JobLevel AS␣
 ↪CurrentJobLevel, LAG(JobRole) OVER (PARTITION BY EmployeeID ORDER BY␣
 ↪YearsAtCompany) AS PreviousJobRole, LAG(JobLevel) OVER (PARTITION BY␣
 ↪EmployeeID ORDER BY YearsAtCompany) AS PreviousJobLevel FROM hr_table ) AS␣
 ↪JobChanges WHERE (CurrentJobRole <> PreviousJobRole) OR (CurrentJobLevel <>␣
 ↪PreviousJobLevel);"
# Call the function with the query
execute_query(query)
```

```
No results found.
```

3.12 12. Find the average distance from home for employees in each department.

[25]:
```
# Define the query
query = "SELECT Department, AVG(DistanceFromHome) AS Avg_Distance FROM hr_table␣
 ↪GROUP BY Department;"
# Call the function with the query
execute_query(query)
```

	Department	Avg_Distance
0	Sales	9.230942
1	Research & Development	9.236212
2	Human Resources	8.253968

3.13 13. Retrieve the top 5 employees with the highest MonthlyIncome.

[26]:
```
# Define the query
query = "SELECT EmployeeID, EmpName, MonthlyIncome FROM hr_table ORDER BY␣
 ↪MonthlyIncome DESC LIMIT 5;"
# Call the function with the query
execute_query(query)
```

	EmployeeID	EmpName	MonthlyIncome
0	4126	REX CALAUNAN	99980
1	2656	THEODORE UNAEGBU	99980
2	1186	JAMES RAMSEY	99980
3	1662	LAURA KELLY	99910
4	3132	JOHN ST CROIX	99910

3.14 14. Calculate the percentage of employees who have had a promotion in the last year.

[27]:
```
# Define the query
query = "SELECT (COUNT(CASE WHEN YearsSinceLastPromotion <= 1 THEN 1 END) /␣
 ↪COUNT(*)) * 100 AS Promotion_Percentage FROM hr_table;"
# Call the function with the query
execute_query(query)
```

```
  Promotion_Percentage
0              63.8095
```

3.15 15. List the employees with the highest and lowest EnvironmentSatisfaction.

[28]:
```
# MySQL database connection details
host = 'YOUR_HOST_NAME'
user = 'YOUR_USER_NAME'
password = 'YOUR_PASSWORD'
database = 'hranalytics'
```

[29]:
```
# Create MySQL connection
conn = mysql.connector.connect(host=host, user=user, password=password,␣
 ↪database=database)
cursor = conn.cursor()
```

[30]:
```
# CSV file path
csv_file = 'employee_survey_data.csv'  # Replace 'your_file.csv' with your CSV␣
 ↪file path

# Read the first row of the CSV to extract column names
with open(csv_file, 'r') as file:
    first_line = file.readline().strip()  # Read the first line
    column_names = first_line.split(',')  # Assuming columns are comma-separated
    file.close()
```

[59]:
```
# Name of table
table_name = str(input("Enter table name: "))

# Create table with extracted column names
create_table_query = f"CREATE TABLE IF NOT EXISTS {table_name} ({', '.
 ↪join([f'{col} TEXT' for col in column_names])})"
cursor.execute(create_table_query)
```

```
Enter table name: employee_survey_table
```

[31]:
```
# Read the CSV file into a pandas DataFrame
df = pd.read_csv(csv_file)
df.head()
```

[31]:

	EmployeeID	EnvironmentSatisfaction	JobSatisfaction	WorkLifeBalance
0	1	3.0	4.0	2.0
1	2	3.0	2.0	4.0
2	3	2.0	2.0	1.0
3	4	4.0	4.0	3.0
4	5	4.0	1.0	3.0

[32]:
```
# Checking shape of dataset
df.shape
```

[32]:
```
(4410, 4)
```

[33]:
```
# Checking column names
df.columns
```

[33]:
```
Index(['EmployeeID', 'EnvironmentSatisfaction', 'JobSatisfaction',
       'WorkLifeBalance'],
      dtype='object')
```

[63]:
```
# Insert data into the created table
for _, row in df.iterrows():
    # Handle NaN values by converting them to None
    row = [None if pd.isna(value) else value for value in row]

    # Prepare and execute the insert query
    insert_query = f"INSERT INTO {table_name} ({', '.join(column_names)})␣
 ↪VALUES ({', '.join(['%s'] * len(column_names))})"
    cursor.execute(insert_query, tuple(row))

# Commit changes and close the connection
conn.commit()
conn.close()

print(f"Table '{table_name}' created and data loaded successfully.")
```

```
Table 'employee_survey_table' created and data loaded successfully.
```

[34]:
```
# Define the query
query = "SELECT a.EmployeeID, a.EmpName,b.EnvironmentSatisfaction FROM hr_table␣
 ↪a JOIN employee_survey_table b ON a.EmployeeID = b.EmployeeID WHERE b.
 ↪EnvironmentSatisfaction IN (SELECT MAX(EnvironmentSatisfaction) FROM␣
 ↪employee_survey_table UNION SELECT MIN(EnvironmentSatisfaction) FROM␣
 ↪employee_survey_table)"
```

```
# Call the function with the query
execute_query(query)
```

```
No results found.
```

3.16 16. Find the employees who have the same JobRole and MaritalStatus.

[35]:
```
# Define the query
query = "SELECT EmployeeID, JobRole, MaritalStatus FROM hr_table e1 WHERE␣
 ↪EXISTS (SELECT 1 FROM hr_table e2 WHERE e1.EmployeeID <> e2.EmployeeID AND␣
 ↪e1.JobRole = e2.JobRole AND e1.MaritalStatus = e2.MaritalStatus) ORDER BY␣
 ↪JobRole, MaritalStatus, EmployeeID;"
# Call the function with the query
execute_query(query)
```

	EmployeeID	JobRole	MaritalStatus
0	1006	Healthcare Representative	Divorced
1	1035	Healthcare Representative	Divorced
2	1078	Healthcare Representative	Divorced
3	1191	Healthcare Representative	Divorced
4	1313	Healthcare Representative	Divorced
…	…	…	…
4405	791	Sales Representative	Single
4406	80	Sales Representative	Single
4407	865	Sales Representative	Single
4408	940	Sales Representative	Single
4409	996	Sales Representative	Single

```
[4410 rows x 3 columns]
```

3.17 17. List the employees with the highest TotalWorkingYears who also have a PerformanceRating of 4.

[36]:
```
# CSV file path
csv_file = 'manager_survey_data.csv'  # Replace 'your_file.csv' with your CSV␣
 ↪file path

# Read the first row of the CSV to extract column names
with open(csv_file, 'r') as file:
    first_line = file.readline().strip()  # Read the first line
    column_names = first_line.split(',')  # Assuming columns are comma-separated
    file.close()
```

[73]:
```
# Name of table
table_name = str(input("Enter table name: "))

# Create table with extracted column names
```

```
create_table_query = f"CREATE TABLE IF NOT EXISTS {table_name} ({', '.
 ↪join([f'{col} TEXT' for col in column_names])})"
cursor.execute(create_table_query)
```

```
Enter table name: manager_survey_table
```

[37]:
```
# Read the CSV file into a pandas DataFrame
df = pd.read_csv(csv_file)
df.head()
```

[37]:

	EmployeeID	JobInvolvement	PerformanceRating
0	1	3	3
1	2	2	4
2	3	3	3
3	4	2	3
4	5	3	3

[38]:
```
# Checking shape of dataset
df.shape
```

[38]:
```
(4410, 3)
```

[39]:
```
# Checking column names
df.columns
```

[39]:
```
Index(['EmployeeID', 'JobInvolvement', 'PerformanceRating'], dtype='object')
```

[77]:
```
# Insert data into the created table
for _, row in df.iterrows():
    # Handle NaN values by converting them to None
    row = [None if pd.isna(value) else value for value in row]

    # Prepare and execute the insert query
    insert_query = f"INSERT INTO {table_name} ({', '.join(column_names)})␣
 ↪VALUES ({', '.join(['%s'] * len(column_names))})"
    cursor.execute(insert_query, tuple(row))

# Commit changes and close the connection
conn.commit()
conn.close()

print(f"Table '{table_name}' created and data loaded successfully.")
```

```
Table 'manager_survey_table' created and data loaded successfully.
```

[40]:
```
# Define the query
```

```
query = "SELECT a.EmployeeID, a.EmpName, a.TotalWorkingYears, b.
 ↪PerformanceRating FROM hr_table a JOIN manager_survey_table b ON a.
 ↪EmployeeID = b.EmployeeID WHERE b.PerformanceRating = 4 AND a.
 ↪TotalWorkingYears = (SELECT MAX(TotalWorkingYears) FROM hr_table WHERE␣
 ↪EmployeeID IN (SELECT EmployeeID FROM manager_survey_table WHERE␣
 ↪PerformanceRating = 4));"
# Call the function with the query
execute_query(query)
```

	EmployeeID	EmpName	TotalWorkingYears	PerformanceRating
0	45	NOEL MORONEY	9.0	4
1	53	STEPHEN TACCHINI	9.0	4
2	121	MICHAEL HENNESSEY	9.0	4
3	195	DOMINIC CELAYA	9.0	4
4	267	RICHARD ZERCHER	9.0	4
5	273	ELAINE COLEMAN	9.0	4
6	353	CURTIS LUM	9.0	4
7	534	JONATHAN FUCHS	9.0	4
8	582	MICHAEL FAVETTI	9.0	4
9	632	MARC PEARSON	9.0	4
10	916	FRANK AGNOST	9.0	4
11	1093	VINCE CHHABRIA	9.0	4
12	1244	MICHAEL TEUPEL	9.0	4
13	1256	MICHAEL MOODY	9.0	4
14	1269	MICHELLE DURGY	9.0	4
15	1302	ALEX TAKAOKA	9.0	4
16	1354	DANTE GIOVANNELLI	9.0	4
17	1357	KEVIN BYRNE	9.0	4
18	1515	LEONARDO HARRIS	9.0	4
19	1523	JOSEPH CORDES	9.0	4
20	1591	JASON REICHARD	9.0	4
21	1665	CHANH PHUNG	9.0	4
22	1737	EDGARDO VERGARA	9.0	4
23	1743	PATRICK D'ARCY	9.0	4
24	1823	LORENZO DONATI	9.0	4
25	2004	BRENDA WALKER	9.0	4
26	2052	SIDNEY LAWS	9.0	4
27	2102	RYAN JONES	9.0	4
28	2386	JENNIFER CHON	9.0	4
29	2563	REY BUZON	9.0	4
30	2714	DANIEL COTTER	9.0	4
31	2726	JOHN DRAKE	9.0	4
32	2739	RAQUEL ALFONZO-YUMUL	9.0	4
33	2772	DAVID LOUSTALOT	9.0	4
34	2824	THOMAS WORTMAN	9.0	4
35	2827	ROSEMARIE SMITH	9.0	4
36	2985	TIMOTHY KELLY	9.0	4

37	2993	MARK MORENO	9.0	4
38	3061	TONY LEUNG	9.0	4
39	3135	MELONEE ALVAREZ	9.0	4
40	3207	PATRICIA CORREA	9.0	4
41	3213	JULIE LYNCH	9.0	4
42	3293	FRANCES WILLIAMS	9.0	4
43	3474	MARCO MAGALLON	9.0	4
44	3522	DIANE OSHIMA	9.0	4
45	3572	NELLIE SARTE	9.0	4
46	3856	PATRICK DUDY	9.0	4
47	4033	WAYLAND LEE	9.0	4
48	4184	BRADFORD BENSON	9.0	4
49	4196	BRIAN CHEU	9.0	4
50	4209	MATTHEW WAYNE	9.0	4
51	4242	JAROME WINESBERRY	9.0	4
52	4294	RAYMOND NG	9.0	4
53	4297	QINGWEN XI	9.0	4

3.18 18. Calculate the average Age and JobSatisfaction for each BusinessTravel type.

[79]:
```
# Define the query
query = ""
# Call the function with the query
execute_query(query)
```

```
No results found.
```

3.19 19. Retrieve the most common EducationField among employees.

[41]:
```
# Define the query
query = "SELECT EducationField, COUNT(*) AS Frequency FROM hr_table GROUP BY␣
 ↪EducationField ORDER BY COUNT(*) DESC LIMIT 1;"
# Call the function with the query
execute_query(query)
```

```
  EducationField  Frequency
0  Life Sciences       1818
```

3.20 20. List the employees who have worked for the company the longest but haven't had a promotion.

[42]:
```
# Define the query
query = "SELECT EmployeeID, EmpName, YearsAtCompany, YearsSinceLastPromotion␣
 ↪FROM hr_table WHERE YearsAtCompany = (SELECT MAX(YearsAtCompany) FROM␣
 ↪hr_table) AND YearsSinceLastPromotion = 0;"
# Call the function with the query
```

```
execute_query(query)
```

```
   EmployeeID            EmpName YearsAtCompany YearsSinceLastPromotion
0          48    MERCEDES GERMAN              9                       0
1          73        RUBY MARTIN              9                       0
2         273     ELAINE COLEMAN              9                       0
3         336  JULIE VAN NOSTERN              9                       0
4         353         CURTIS LUM              9                       0
..        …                   …            …                       …
61       4151          RUTH WANG              9                       0
62       4164       FRANCISCO HO              9                       0
63       4201        SHARON HITE              9                       0
64       4264          JUNG PARK              9                       0
65       4286      JASON MAXWELL              9                       0

[66 rows x 4 columns]
```