People Analytic Project - Colacho Company, Inc

1. Import Libraries

In [ ]:
```
import pandas as pd
import numpy as np
import random
import string
from faker import Faker
from datetime import datetime

fake = Faker()
```

2. Create the data frame

We are going to simulate a database composed be hundred employees

In [ ]:
```
# Let's create a list with the names and other list with their surnames
names = [
    'John', 'Emily', 'Michael', 'Sarah', 'David', 'Jessica', 'Daniel', 'Olivia', 'C
    'Matthew', 'Ava', 'Andrew', 'Emma', 'James', 'Isabella', 'Joseph', 'Mia', 'Anth
    'Robert', 'Charlotte', 'William', 'Amelia', 'David', 'Evelyn', 'Alexander', 'Ha
    'Ethan', 'Chloe', 'Steven', 'Ella', 'Richard', 'Victoria', 'Ryan', 'Avery', 'Ch
    'Edward', 'Scarlett', 'George', 'Madison', 'Samuel', 'Lily', 'Jackson', 'Aria',
    'Owen', 'Addison', 'Sebastian', 'Aubrey', 'Benjamin', 'Natalie', 'Nicholas', 'Z
    'Aiden', 'Brooklyn', 'Adam', 'Stella', 'Dylan', 'Nora', 'Wyatt', 'Leah', 'Caleb
    'Nathan', 'Claire', 'Jayden', 'Ellie', 'Zachary', 'Savannah', 'Carter', 'Skylar
    'Isaac', 'Aaliyah', 'Christian', 'Penelope', 'Liam', 'Camila', 'Julian', 'Arian
    'Levi', 'Maya', 'Jaxon', 'Gabriella', 'Aaron', 'Madelyn', 'Brayden', 'Elena', '
]

surnames = [
    'Smith', 'Johnson', 'Williams', 'Jones', 'Brown', 'Davis', 'Miller', 'Wilson',
    'Anderson', 'Thomas', 'Jackson', 'White', 'Harris', 'Martin', 'Thompson', 'Garc
    'Clark', 'Rodriguez', 'Lewis', 'Lee', 'Walker', 'Hall', 'Allen', 'Young', 'Hern
    'Wright', 'Lopez', 'Hill', 'Scott', 'Green', 'Adams', 'Baker', 'Gonzalez', 'Nel
    'Mitchell', 'Perez', 'Roberts', 'Turner', 'Phillips', 'Campbell', 'Parker', 'Ev
    'Stewart', 'Sanchez', 'Morris', 'Rogers', 'Reed', 'Cook', 'Morgan', 'Bell', 'Mu
    'Rivera', 'Cooper', 'Richardson', 'Cox', 'Howard', 'Ward', 'Torres', 'Peterson'
    'James', 'Watson', 'Brooks', 'Kelly', 'Sanders', 'Price', 'Bennett', 'Wood', 'B
    'Henderson', 'Coleman', 'Jenkins', 'Perry', 'Powell', 'Long', 'Patterson', 'Hug
    'Butler', 'Simmons', 'Foster', 'Gonzales', 'Bryant', 'Alexander', 'Russell', 'G
]

sex = [
    'male','female','male','female','male','female','male','female','male','female'
]


# Create our data frame
colacho_df = pd.DataFrame({
    'Name': names,
    'Surname': surnames,
    'Gender': sex
```

```
})

# Checking if everything is ok
print(colacho_df.head())
```

```
      Name   Surname  Gender
0     John     Smith    male
1    Emily   Johnson  female
2  Michael  Williams    male
3    Sarah     Jones  female
4    David     Brown    male
```

2.1 Create new columns

Now we are going to create new columns, work department, id, address, birthdate, and age

2.1.a Work Departments Column

Here we are going to create a random sequence of values from 1 to 5, and then we are going to replace those values with the department's names

In [ ]:
```
# Random values to replace later
work_department = np.random.randint(1, 6, size=100)
colacho_df['WorkDepartment'] = work_department

# Its time to replace the values with the areas where people are from
department_map = {
    1: 'HHRR',
    2: 'Sales',
    3: 'Account',
    4: 'Marketing',
    5: 'Operations'
}

colacho_df['WorkDepartment'] = colacho_df['WorkDepartment'].map(department_map)

# Let's see if everthing is OK
print(colacho_df.head(10))
```

```
          Name   Surname  Gender WorkDepartment
0         John     Smith    male        Account
1        Emily   Johnson  female        Account
2      Michael  Williams    male          Sales
3        Sarah     Jones  female           HHRR
4        David     Brown    male     Operations
5      Jessica     Davis  female           HHRR
6       Daniel    Miller    male          Sales
7       Olivia    Wilson  female          Sales
8  Christopher     Moore    male      Marketing
9       Sophia    Taylor  female           HHRR
```

2.1.b ID column

For this column, we are going to create a random ID that will combine numbers with letters, we will use libraries random and string

In [ ]:
```
# Let's create to function to get the Id
def generate_id():
    letters = string.ascii_letters.upper()
    numbers = string.digits
    random_letters = ''.join(random.choice(letters) for i in range(2))
```

```
    random_numbers = ''.join(random.choice(numbers) for i in range(4))
    return random_letters + random_numbers

# Our ID column with their random values
colacho_df['ID'] = colacho_df.apply(lambda row: generate_id(), axis=1)

# Checking if works fine
# print(colacho_df.head(10))
colacho_df
```

Out[ ]:

	Name	Surname	Gender	WorkDepartment	ID
0	John	Smith	male	Account	SZ2174
1	Emily	Johnson	female	Account	ZG8243
2	Michael	Williams	male	Sales	LB2444
3	Sarah	Jones	female	HHRR	ZG2248
4	David	Brown	male	Operations	JE3473
...	...	...	...	...	...
95	Madelyn	Alexander	female	Operations	SU8337
96	Brayden	Russell	male	Account	NU0471
97	Elena	Griffin	female	Operations	HT8585
98	Oliver	Diaz	male	HHRR	QH3190
99	Anna	Hayes	female	HHRR	JM2926

100 rows × 5 columns

2.1.c Address Column

Let's use faker to insert the values for the address column

In [ ]:

```
# Generate the fake address for the employees
Faker.seed(0)
address = [fake.street_address() for i in range(100)]
print(address)

colacho_df['Address'] = address
colacho_df['Address']
```

```
['0487 Hull Village Suite 759', '242 Christine Glen', '1157 Michael Island', '778
Brown Plaza', '60975 Jessica Squares', '93328 Davis Island', '48418 Olsen Plains A
pt. 989', '1965 Kelly Field Apt. 094', '12201 Massey Pine Suite 833', '477 Miller
Ridge Apt. 795', '04135 Marvin Via', '23098 Anthony Roads', '916 Mitchell Crescen
t', '032 Timothy Stream', '086 Mary Cliff', '45620 Williams Courts', '45792 Tammy
Centers Apt. 258', '97207 Mccullough Well Suite 564', '071 Stevenson Plains', '375
94 Garza Roads Apt. 466', '93523 Dana Lane', '69602 Brown Squares Apt. 787', '7547
0 Lopez Roads', '206 Stewart Forest', '089 James Rest', '4217 Burton Brooks', '142
8 Wilson Drives Suite 000', '485 Mays Vista', '977 Watson Plain', '69402 Joseph Ju
nction', '159 Blake Estates Apt. 945', '241 Cross Causeway Suite 281', '5461 Ann O
rchard Suite 551', '76045 Samantha Road Suite 111', '306 Corey Point', '7936 Miche
al Green Apt. 635', '08731 Sanders Fords', '92137 Ward Views', '97296 Rich Park',
'7389 Alec Squares Suite 508', '9269 Robbins Valley', '18013 Billy Bridge Suite 52
2', '8688 Audrey Springs Apt. 634', '6993 Diane Alley Apt. 489', '1744 Cruz Lights
Apt. 223', '500 Butler Overpass Apt. 256', '767 Craig Tunnel', '09925 Isabel Run',
'67109 Mendez Junction Apt. 942', '1830 Wallace Throughway Apt. 751', '74089 Jerry
Trail', '412 Snow Manors Apt. 161', '2076 Johnson Way', '415 David Square Apt. 55
2', '51923 Jamie Spring', '045 Sarah Fort', '71754 Anthony Fords', '7448 White Com
mon Apt. 894', '2235 Joel Ferry', '6029 Phillip Squares Apt. 264', '296 Walsh Corn
er Apt. 758', '6065 Harris Hill', '21418 Laura Mission Suite 266', '75825 Welch Co
rners', '40797 Jeffery Crescent Suite 892', '717 Amy Lodge Suite 014', '46896 Garc
ia Glen', '3910 Laura Inlet Suite 183', '64482 Amanda Loop', '447 Carroll Dam Apt.
116', '728 Gomez Mountains Suite 377', '0742 Williams Road Apt. 057', '81160 Willi
ams River Apt. 443', '431 Murray Isle', '56627 Alexandria Curve Apt. 275', '7633 B
entley Radial Apt. 603', '50239 Salazar Squares', '6708 Carpenter Overpass Suite 7
35', '00250 Pena Dam Apt. 639', '373 Franklin Rest Apt. 558', '4080 Miguel Fords S
uite 334', '8647 Wiggins Garden Apt. 481', '7706 Stevens Crest', '31263 Salazar Vi
aduct', '602 Tracy Crossroad Suite 556', '498 Jennifer Tunnel', '978 Nelson Brook
Apt. 912', '34088 Trevino Crossing Suite 419', '84331 Leonard Fort Suite 749', '08
855 Lisa Wells', '28123 Hudson Square Apt. 323', '9927 Christina Burg Suite 774',
'82778 Padilla Common', '96354 Acevedo Fords Apt. 535', '345 Kevin Knolls Apt. 25
0', '003 Alexander Shoal Suite 105', '37625 Thompson Isle Suite 606', '870 Robert
Loaf Apt. 082', '53230 Julia Villages', '169 Christine Mount']
```

Out[ ]:

```
0       0487 Hull Village Suite 759
1                242 Christine Glen
2               1157 Michael Island
3                    778 Brown Plaza
4             60975 Jessica Squares
                      ...
95    003 Alexander Shoal Suite 105
96    37625 Thompson Isle Suite 606
97          870 Robert Loaf Apt. 082
98              53230 Julia Villages
99               169 Christine Mount
Name: Address, Length: 100, dtype: object
```

2.1.d Birthday Column

With faker let's create the birthday column from our employees

In [ ]:

```
# Let's put the values into a variable
birthday = [fake.date_of_birth(minimum_age=20, maximum_age=68) for i in range(100)]
print(birthday)

colacho_df['Birthday'] = birthday
colacho_df
```

```
[datetime.date(2003, 4, 8), datetime.date(1965, 2, 27), datetime.date(1996, 10, 1
6), datetime.date(1995, 11, 29), datetime.date(1957, 9, 2), datetime.date(1972, 2,
16), datetime.date(1978, 2, 5), datetime.date(1980, 7, 26), datetime.date(1956, 1
0, 19), datetime.date(1997, 7, 12), datetime.date(1957, 6, 1), datetime.date(1988,
9, 20), datetime.date(1961, 1, 16), datetime.date(1979, 5, 2), datetime.date(1974,
11, 12), datetime.date(2000, 8, 25), datetime.date(1965, 3, 17), datetime.date(198
6, 2, 13), datetime.date(1971, 1, 6), datetime.date(1989, 6, 23), datetime.date(19
79, 2, 27), datetime.date(1965, 12, 27), datetime.date(1982, 6, 27), datetime.date
(1977, 12, 3), datetime.date(1973, 4, 22), datetime.date(1988, 6, 12), datetime.da
te(1981, 8, 29), datetime.date(1955, 12, 14), datetime.date(1976, 2, 4), datetime.
date(1991, 1, 13), datetime.date(1974, 7, 23), datetime.date(1992, 5, 27), datetim
e.date(1986, 11, 9), datetime.date(1957, 4, 28), datetime.date(1991, 2, 18), datet
ime.date(1993, 10, 7), datetime.date(1992, 7, 7), datetime.date(1972, 10, 21), dat
etime.date(2001, 9, 7), datetime.date(1957, 7, 20), datetime.date(1985, 12, 20), d
atetime.date(1981, 10, 30), datetime.date(1963, 2, 11), datetime.date(1982, 6, 3),
datetime.date(1978, 7, 13), datetime.date(1988, 9, 12), datetime.date(1958, 6, 5),
datetime.date(1956, 5, 9), datetime.date(1973, 7, 11), datetime.date(1957, 4, 11),
datetime.date(1972, 3, 23), datetime.date(2001, 4, 8), datetime.date(2001, 3, 31),
datetime.date(1994, 7, 20), datetime.date(2002, 10, 6), datetime.date(1974, 8, 3
1), datetime.date(2001, 8, 20), datetime.date(1969, 1, 14), datetime.date(1990, 1
2, 18), datetime.date(1990, 1, 28), datetime.date(1978, 2, 6), datetime.date(1981,
3, 20), datetime.date(1972, 1, 1), datetime.date(1969, 3, 1), datetime.date(1962,
11, 3), datetime.date(1993, 6, 18), datetime.date(1977, 5, 13), datetime.date(197
1, 6, 17), datetime.date(1994, 10, 31), datetime.date(2000, 10, 19), datetime.date
(1991, 3, 15), datetime.date(2001, 4, 23), datetime.date(1978, 12, 28), datetime.d
ate(1966, 1, 3), datetime.date(1965, 5, 12), datetime.date(1977, 6, 12), datetime.
date(1955, 7, 25), datetime.date(1994, 10, 1), datetime.date(1958, 6, 7), datetim
e.date(1993, 5, 20), datetime.date(1965, 7, 28), datetime.date(1978, 5, 24), datet
ime.date(1979, 8, 30), datetime.date(1974, 10, 5), datetime.date(1997, 7, 11), dat
etime.date(1974, 11, 30), datetime.date(1976, 12, 4), datetime.date(1988, 8, 14),
datetime.date(1982, 5, 31), datetime.date(1995, 10, 28), datetime.date(1984, 4, 1
0), datetime.date(1966, 7, 8), datetime.date(1955, 1, 6), datetime.date(1964, 7, 1
0), datetime.date(1993, 8, 21), datetime.date(1957, 11, 29), datetime.date(1985,
1, 6), datetime.date(1963, 7, 5), datetime.date(1978, 3, 5), datetime.date(1955,
7, 31)]
```

Out[ ]:

	Name	Surname	Gender	WorkDepartment	ID	Address	Birthday
0	John	Smith	male	Account	SZ2174	0487 Hull Village Suite 759	2003-04-08
1	Emily	Johnson	female	Account	ZG8243	242 Christine Glen	1965-02-27
2	Michael	Williams	male	Sales	LB2444	1157 Michael Island	1996-10-16
3	Sarah	Jones	female	HHRR	ZG2248	778 Brown Plaza	1995-11-29
4	David	Brown	male	Operations	JE3473	60975 Jessica Squares	1957-09-02
...	...	...	...	...	...	...	...
95	Madelyn	Alexander	female	Operations	SU8337	003 Alexander Shoal Suite 105	1957-11-29
96	Brayden	Russell	male	Account	NU0471	37625 Thompson Isle Suite 606	1985-01-06
97	Elena	Griffin	female	Operations	HT8585	870 Robert Loaf Apt. 082	1963-07-05
98	Oliver	Diaz	male	HHRR	QH3190	53230 Julia Villages	1978-03-05
99	Anna	Hayes	female	HHRR	JM2926	169 Christine Mount	1955-07-31

100 rows × 7 columns

2.1.e Age column

For this task, we are going to take the dates from the Birthday column and using datetime.now() we are going to calculate the Age of our employees Then add the values to the database in the column "Age".

In [ ]:

```
# Convert 'birthday' column to datetime
colacho_df['Birthday'] = pd.to_datetime(colacho_df['Birthday'])

# Calculate the age based on today's date
today = datetime.now()
# age = np.random.randint(20, 68, size=100)
colacho_df['Age'] = today.year - colacho_df['Birthday'].dt.year - ((today.month < c

# Checking if everything is OK
colacho_df
```

Out[ ]:

	Name	Surname	Gender	WorkDepartment	ID	Address	Birthday	Age
0	John	Smith	male	Account	SZ2174	0487 Hull Village Suite 759	2003-04-08	20
1	Emily	Johnson	female	Account	ZG8243	242 Christine Glen	1965-02-27	58
2	Michael	Williams	male	Sales	LB2444	1157 Michael Island	1996-10-16	27
3	Sarah	Jones	female	HHRR	ZG2248	778 Brown Plaza	1995-11-29	28
4	David	Brown	male	Operations	JE3473	60975 Jessica Squares	1957-09-02	66
...	...	...	...	...	...	...	...	...
95	Madelyn	Alexander	female	Operations	SU8337	003 Alexander Shoal Suite 105	1957-11-29	66
96	Brayden	Russell	male	Account	NU0471	37625 Thompson Isle Suite 606	1985-01-06	38
97	Elena	Griffin	female	Operations	HT8585	870 Robert Loaf Apt. 082	1963-07-05	60
98	Oliver	Diaz	male	HHRR	QH3190	53230 Julia Villages	1978-03-05	45
99	Anna	Hayes	female	HHRR	JM2926	169 Christine Mount	1955-07-31	68

100 rows × 8 columns

2.1.f Saving our dataset

Because we are going to use our data for other purposes, we need to save our dataset, because some columns have random values and we don't want to change the data every time we restart the program.

In [ ]:
```
# Before saving our data, let's check our data
colacho_df.info()
```

```
<class 'pandas.core.frame.DataFrame'>
RangeIndex: 100 entries, 0 to 99
Data columns (total 8 columns):
 #   Column          Non-Null Count  Dtype
---  ------          --------------  -----
 0   Name            100 non-null    object
 1   Surname         100 non-null    object
 2   Gender          100 non-null    object
 3   WorkDepartment  100 non-null    object
 4   ID              100 non-null    object
 5   Address         100 non-null    object
 6   Birthday        100 non-null    datetime64[ns]
 7   Age             100 non-null    int64
dtypes: datetime64[ns](1), int64(1), object(6)
memory usage: 6.4+ KB
```

In [ ]:
```
# Checking for missing values
colacho_df.isna()
```

Out[ ]:

	Name	Surname	Gender	WorkDepartment	ID	Address	Birthday	Age
0	False	False	False	False	False	False	False	False
1	False	False	False	False	False	False	False	False
2	False	False	False	False	False	False	False	False
3	False	False	False	False	False	False	False	False
4	False	False	False	False	False	False	False	False
...	...	...	...	...	...	...	...	...
95	False	False	False	False	False	False	False	False
96	False	False	False	False	False	False	False	False
97	False	False	False	False	False	False	False	False
98	False	False	False	False	False	False	False	False
99	False	False	False	False	False	False	False	False

100 rows × 8 columns

In [ ]:

```
# Let's save the data in a .csv file
colacho_df.to_csv('colacho_database.csv', index=False)
```

People Analytic Project - Colacho Company, Inc (Part 2)

Salary Database

3. Import libraries

In [ ]:
```
import pandas as pd
import numpy as np
import matplotlib.pyplot as plt
```

4. Import our database

We are going to import our database and use it to create a new data frame for the salaries. We will use column names, surnames, work department, and ID. Also we will add some new columns too.

In [ ]:
```
# Let's import our data frame
cdf_rawdata = pd.read_csv('colacho_database.csv')
cdf_rawdata
```

Out[ ]:

	Name	Surname	Gender	WorkDepartment	ID	Address	Birthday	Age
0	John	Smith	male	Account	SZ2174	0487 Hull Village Suite 759	2003-04-08	20
1	Emily	Johnson	female	Account	ZG8243	242 Christine Glen	1965-02-27	58
2	Michael	Williams	male	Sales	LB2444	1157 Michael Island	1996-10-16	27
3	Sarah	Jones	female	HHRR	ZG2248	778 Brown Plaza	1995-11-29	28
4	David	Brown	male	Operations	JE3473	60975 Jessica Squares	1957-09-02	66
...	...	...	...	...	...	...	...	...
95	Madelyn	Alexander	female	Operations	SU8337	003 Alexander Shoal Suite 105	1957-11-29	66
96	Brayden	Russell	male	Account	NU0471	37625 Thompson Isle Suite 606	1985-01-06	38
97	Elena	Griffin	female	Operations	HT8585	870 Robert Loaf Apt. 082	1963-07-05	60
98	Oliver	Diaz	male	HHRR	QH3190	53230 Julia Villages	1978-03-05	45
99	Anna	Hayes	female	HHRR	JM2926	169 Christine Mount	1955-07-31	68

100 rows × 8 columns

4.1 Salaries data frame

It's time to create our new data frame for the employees salaries

In [ ]:

```
# Create the new data frame in a new variable
cdf_salary = cdf_rawdata[['ID', 'Name', 'Surname', 'WorkDepartment']].copy()

# Checking if the data frame has the information needed
cdf_salary
```

Out[ ]:

	ID	Name	Surname	WorkDepartment
0	SZ2174	John	Smith	Account
1	ZG8243	Emily	Johnson	Account
2	LB2444	Michael	Williams	Sales
3	ZG2248	Sarah	Jones	HHRR
4	JE3473	David	Brown	Operations
...	...	...	...	...
95	SU8337	Madelyn	Alexander	Operations
96	NU0471	Brayden	Russell	Account
97	HT8585	Elena	Griffin	Operations
98	QH3190	Oliver	Diaz	HHRR
99	JM2926	Anna	Hayes	HHRR

100 rows × 4 columns

4.2 Create the new columns

With the new data frame, now we can add the new columns ***Salary*** and ***Experience***

4.2.a Salary

For this example we are not going to use a date to calculate the years the employees have in the company. Instead, we will generate the number by using a random function

In [ ]:

```
# Salaries
salary = np.random.randint(18000, 75000, size = 100)
cdf_salary['Salary'] = salary

cdf_salary
```

Out[ ]:

	ID	Name	Surname	WorkDepartment	Salary
0	SZ2174	John	Smith	Account	58694
1	ZG8243	Emily	Johnson	Account	68734
2	LB2444	Michael	Williams	Sales	72195
3	ZG2248	Sarah	Jones	HHRR	22344
4	JE3473	David	Brown	Operations	21531
...	...	...	...	...	...
95	SU8337	Madelyn	Alexander	Operations	19536
96	NU0471	Brayden	Russell	Account	63372
97	HT8585	Elena	Griffin	Operations	49178
98	QH3190	Oliver	Diaz	HHRR	54154
99	JM2926	Anna	Hayes	HHRR	53523

100 rows × 5 columns

4.2.b Experience

The same as in the Salary column, but for the years of experience, I will transform them into months to make more easier to understand the values avoiding later, dealing with float numbers because they won't represent correctly the time of experience from each employee.

In [ ]:

```
# Create the random number generator
exp_years = np.random.uniform(0, 8, size=100)

# Convert the years of experience into months
exp_months = exp_years * 12

# Round our numbers into one decimal
exp_months_rounded = np.round(exp_months)

# Put the values into the column 'Experience'
cdf_salary['Experience'] = exp_months_rounded

# Check our data frame
cdf_salary
```

Out[ ]:

	ID	Name	Surname	WorkDepartment	Salary	Experience
0	SZ2174	John	Smith	Account	58694	28.0
1	ZG8243	Emily	Johnson	Account	68734	46.0
2	LB2444	Michael	Williams	Sales	72195	58.0
3	ZG2248	Sarah	Jones	HHRR	22344	72.0
4	JE3473	David	Brown	Operations	21531	15.0
...	...	...	...	...	...	...
95	SU8337	Madelyn	Alexander	Operations	19536	47.0
96	NU0471	Brayden	Russell	Account	63372	36.0
97	HT8585	Elena	Griffin	Operations	49178	43.0
98	QH3190	Oliver	Diaz	HHRR	54154	56.0
99	JM2926	Anna	Hayes	HHRR	53523	62.0

100 rows × 6 columns

4.3 Some numbers

Let's use .describe() method to see some details about salary information and experience

In [ ]:

```
# Basic statistics from 'Salary' and 'Experience' columns
cdf_salary[['Salary', 'Experience']].describe()
```

Out[ ]:

	Salary	Experience
count	100.000000	100.000000
mean	47439.660000	50.560000
std	17124.951892	26.425928
min	18183.000000	3.000000
25%	32124.250000	30.500000
50%	49239.500000	49.000000
75%	60620.500000	74.000000
max	73589.000000	96.000000

4.3.a Correlation between Salary and Experience

Now with the ***Linear Regression*** I will try to analyze the correlation between Salary and Experience

In [ ]:

```
# Visualizing our data with a scatterplot
x = cdf_salary['Experience'];
y = cdf_salary['Salary'];
plt.scatter(x = x, y = y, color='#9467bd')
#obtain m (slope) and b(intercept) of linear regression line
m, b = np.polyfit(x, y, 1)
lreg = np.corrcoef(x, y)
plt.plot(x, m*x+b, color='red')
```

```
# Labels and title
plt.xlabel('Experience')
plt.ylabel('Salary')
plt.title('Correlation between Salary and Experience ')
print(lreg)
```

```
[[ 1.         -0.02052559]
 [-0.02052559  1.        ]]
```

4.3.b Import the statmodels formula

I import the Linear regression formula to work with it.

In [ ]:
```
import statsmodels.formula.api as smf
model = smf.ols('Salary ~ Experience', data = cdf_salary).fit()
model.summary()
```

Out[ ]:

OLS Regression Results

Dep. Variable:	Salary	**R-squared:**	0.000
Model:	OLS	**Adj. R-squared:**	-0.010
Method:	Least Squares	**F-statistic:**	0.04130
Date:	Wed, 13 Dec 2023	**Prob (F-statistic):**	0.839
Time:	14:42:02	**Log-Likelihood:**	-1116.2
No. Observations:	100	**AIC:**	2236.
Df Residuals:	98	**BIC:**	2242.
Df Model:	1		
Covariance Type:	nonrobust		

	coef	**std err**	**t**	**P>|t|**	**[0.025**	**0.975]**
Intercept	4.811e+04	3729.751	12.900	0.000	4.07e+04	5.55e+04
Experience	-13.3013	65.448	-0.203	0.839	-143.180	116.578

Omnibus:	41.015	**Durbin-Watson:**	1.868
Prob(Omnibus):	0.000	**Jarque-Bera (JB):**	7.046
Skew:	-0.198	**Prob(JB):**	0.0295
Kurtosis:	1.762	**Cond. No.**	124.

Notes:
[1] Standard Errors assume that the covariance matrix of the errors is correctly specified.

4.3.c Creating our prediction model

With *Machine Learning* I will create a prediction model

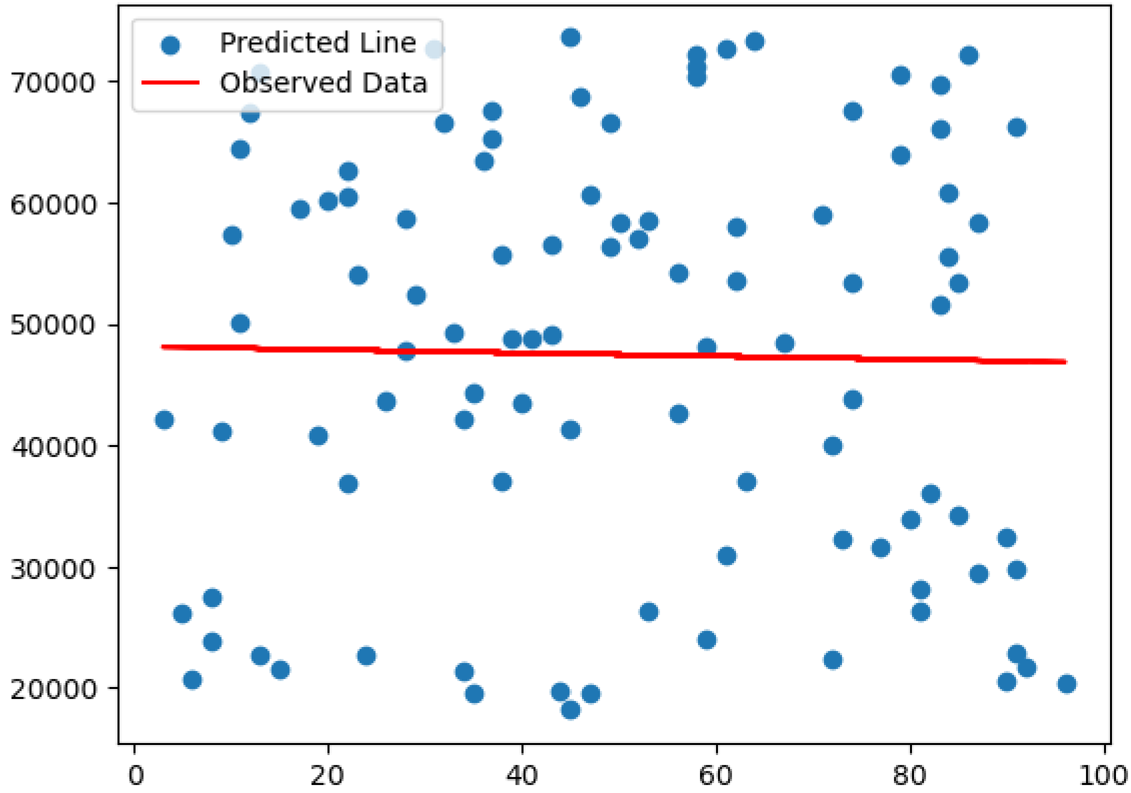
In [ ]:

```
# Let's predict our salaries for each experience years
pred1 = model.predict(pd.DataFrame(cdf_salary['Experience']))
print(pred1)
```

```
0     47739.737776
1     47500.314019
2     47340.698180
3     47154.479702
4     47912.654934
          ...
95    47487.012699
96    47633.327217
97    47540.217978
98    47367.300820
99    47287.492901
Length: 100, dtype: float64
```

In [ ]:

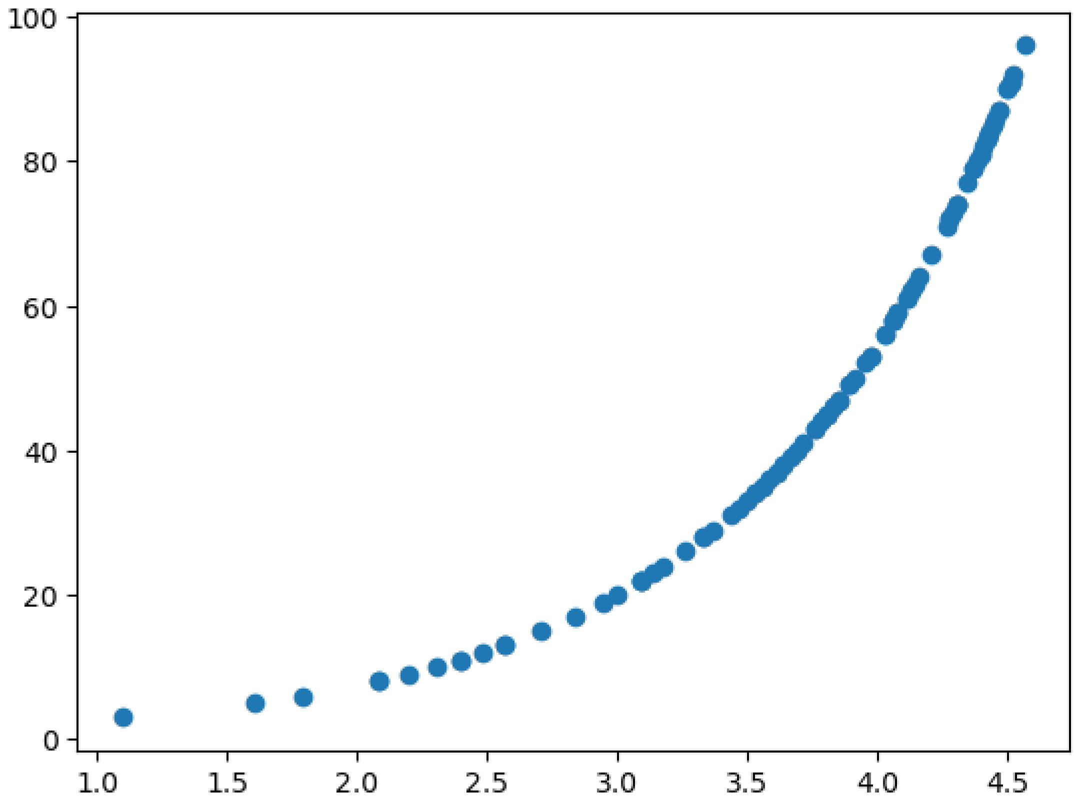
```
# Regression Line
plt.scatter(x, y)
plt.plot(x, pred1, 'r')
plt.legend(['Predicted Line', 'Observed Data'])
plt.show()
```



In [ ]:
```
# Error Calculation
res1 = y - pred1
res_sqr1 = res1 * res1
mse1 = np.mean(res_sqr1)
rmse1 = np.sqrt(mse1)
print(rmse1)
```

```
17035.522328550174
```

In [ ]:
```
# Transformed data
# Log Transformation
plt.scatter(x = np.log(x), y = x)
np.corrcoef(np.log(x), y)
model2 = smf.ols('Salary ~ Experience', data = cdf_salary)
```



4.4 Separating data

Let's work with the data in different ways. It's time to separate the data by department and analyze the correlation separately.

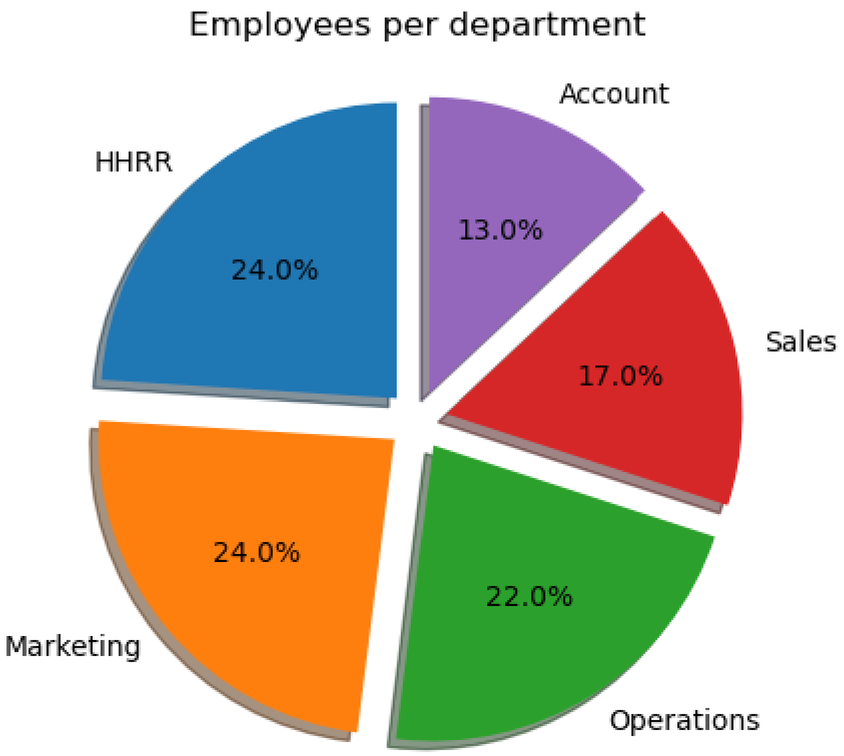
4.4.a Counting the number of employees per department

In [ ]:
```
# Counting employees per department
cdf_salary_count = cdf_salary['WorkDepartment'].value_counts()
cdf_salary_count
```

Out[ ]:
```
HHRR          24
Marketing     24
Operations    22
Sales         17
Account       13
Name: WorkDepartment, dtype: int64
```

In [ ]:
```
# Creating a plot to visualize the data
explode = [0.1] * len(cdf_salary_count)
cdf_salary['WorkDepartment'].value_counts().plot(
    kind='pie',
    explode=explode,
    title='Employees per department',
    autopct='%1.1f%%',
    shadow=True,
    startangle=90,
    ylabel=''
)
```

Out[ ]:
```
<Axes: title={'center': 'Employees per department'}>
```



4.4.b Sorting values by 'Experience' and 'Salary'

In [ ]:
```
cdf_salary.sort_values(['Experience', 'Salary'], ascending=False)
```

Out[ ]:

	ID	Name	Surname	WorkDepartment	Salary	Experience
19	RH0053	Abigail	Robinson	Account	20378	96.0
58	UG0788	Daniel	Murphy	Marketing	21697	92.0
40	KR3331	Edward	Mitchell	Marketing	66309	91.0
93	JI1684	Gabriella	Gonzales	Sales	29843	91.0
88	LO3156	Gabriel	Flores	HHRR	22806	91.0
...	...	...	...	...	...	...
80	FV7585	Isaac	Henderson	Sales	27565	8.0
56	ET9072	Nicholas	Morgan	Account	23835	8.0
75	BV1209	Savannah	Price	Operations	20713	6.0
12	EF3588	Andrew	Jackson	Sales	26255	5.0
13	MZ3212	Emma	White	HHRR	42158	3.0

100 rows × 6 columns

4.4.c Group our data by Department

In [ ]:
```
# Creating a dictionary to store the data from 'WorkDepartment'
grouped_departments = {}

# Group the data by each department, we also need to copy the data
for department, data in cdf_salary.groupby('WorkDepartment'):
    grouped_departments[department] = data.copy()
```

```python
# Checking if the data is ok
print(grouped_departments)
```

```
{'Account':         ID      Name   Surname WorkDepartment  Salary  Experience
0   SZ2174      John     Smith        Account   58694        28.0
1   ZG8243     Emily   Johnson        Account   68734        46.0
19  RH0053   Abigail  Robinson        Account   20378        96.0
24  VI4934     David    Walker        Account   44257        35.0
53  FQ5081    Aubrey    Rogers        Account   24076        59.0
56  ET9072  Nicholas    Morgan        Account   23835         8.0
57  BH7650      Zoey      Bell        Account   19824        44.0
60  UM4299     Aiden    Rivera        Account   33860        80.0
61  CC6460  Brooklyn    Cooper        Account   42127        34.0
64  AM0081     Dylan    Howard        Account   30985        61.0
70  LD5111    Nathan     James        Account   72621        61.0
94  ID3572     Aaron    Bryant        Account   52413        29.0
96  NU0471   Brayden   Russell        Account   63372        36.0, 'HHRR':
ID      Name     Surname WorkDepartment  Salary  Experience
3   ZG2248     Sarah       Jones           HHRR   22344        72.0
5   BB3609   Jessica       Davis           HHRR   59021        71.0
9   PH5293    Sophia      Taylor           HHRR   73316        64.0
13  MZ3212      Emma       White           HHRR   42158         3.0
15  CF9510  Isabella      Martin           HHRR   36044        82.0
17  OE3003       Mia      Garcia           HHRR   41203         9.0
20  GL5268    Robert       Clark           HHRR   65214        37.0
25  SL6383    Evelyn        Hall           HHRR   57049        52.0
29  TG4843     Grace        King           HHRR   40898        19.0
30  QA5493     Ethan      Wright           HHRR   48410        67.0
31  JZ5680     Chloe       Lopez           HHRR   66649        32.0
32  ZR8804    Steven        Hill           HHRR   21436        34.0
43  WL5614   Madison      Turner           HHRR   20503        90.0
49  IT0440   Lillian     Collins           HHRR   73589        45.0
51  TZ8429   Addison     Sanchez           HHRR   58526        53.0
54  AB2695  Benjamin        Reed           HHRR   72186        86.0
62  VI0208      Adam  Richardson           HHRR   47708        28.0
68  AK0475     Caleb        Gray           HHRR   37027        38.0
74  LX8932   Zachary     Sanders           HHRR   19656        35.0
78  IO9348      Luke      Barnes           HHRR   41401        45.0
86  BD4756    Julian   Patterson           HHRR   56282        49.0
88  LO3156   Gabriel      Flores           HHRR   22806        91.0
98  QH3190    Oliver        Diaz           HHRR   54154        56.0
99  JM2926      Anna       Hayes           HHRR   53523        62.0, 'Marketing':
ID         Name    Surname WorkDepartment  Salary  Experience
8   KP5874  Christopher       Moore      Marketing   57277        10.0
10  AA4082      Matthew    Anderson      Marketing   36958        22.0
11  JL5409          Ava      Thomas      Marketing   37018        63.0
28  EL3126       Thomas   Hernandez      Marketing   60532        22.0
33  IO2124         Ella       Scott      Marketing   70426        58.0
36  TT6646         Ryan       Baker      Marketing   22786        13.0
40  KR3331       Edward    Mitchell      Marketing   66309        91.0
42  TR6921       George     Roberts      Marketing   53399        85.0
44  YQ4003       Samuel    Phillips      Marketing   32291        73.0
45  XT7996         Lily    Campbell      Marketing   67620        74.0
46  PK4672      Jackson      Parker      Marketing   62556        22.0
50  YN0851         Owen     Stewart      Marketing   60790        84.0
52  XQ0649    Sebastian      Morris      Marketing   56539        43.0
55  EL9384      Natalie        Cook      Marketing   43830        74.0
58  UG0788       Daniel      Murphy      Marketing   21697        92.0
59  OD1273       Hannah      Bailey      Marketing   43673        26.0
65  PA5532         Nora        Ward      Marketing   67424        12.0
69  SH1351       Audrey     Ramirez      Marketing   22675        24.0
71  TE5901       Claire      Watson      Marketing   51629        83.0
83  HV9147     Penelope       Perry      Marketing   42683        56.0
84  EO8653         Liam      Powell      Marketing   28088        81.0
90  VP4724         Levi      Butler      Marketing   66137        83.0
91  PL0038         Maya     Simmons      Marketing   43512        40.0
92  WV2546        Jaxon      Foster      Marketing   58383        87.0, 'Operation
```

```
s':          ID       Name     Surname  WorkDepartment  Salary  Experience
4    JE3473      David        Brown     Operations    21531        15.0
14   HI1614      James       Harris     Operations    31624        77.0
18   LB3872    Anthony     Martinez     Operations    53429        74.0
21   FW2732  Charlotte    Rodriguez     Operations    70769        13.0
23   JW8178     Amelia          Lee     Operations    54090        23.0
26   VH1780  Alexander        Allen     Operations    71166        58.0
27   GF2719     Harper        Young     Operations    50153        11.0
38   VY9916    Charles       Nelson     Operations    40021        72.0
41   BZ1996   Scarlett        Perez     Operations    55770        38.0
47   AM7483       Aria        Evans     Operations    49301        33.0
63   JN5295     Stella          Cox     Operations    63938        79.0
66   BV8467      Wyatt       Torres     Operations    26346        53.0
72   WL2565     Jayden       Brooks     Operations    57930        62.0
73   XQ3462      Ellie        Kelly     Operations    32464        90.0
75   BV1209   Savannah        Price     Operations    20713         6.0
76   NA2508     Carter      Bennett     Operations    48031        59.0
77   IW6082     Skylar         Wood     Operations    69755        83.0
85   MU4083     Camila         Long     Operations    48793        39.0
87   WG6354     Ariana       Hughes     Operations    60065        20.0
89   NW0984    Kennedy   Washington     Operations    26405        81.0
95   SU8337    Madelyn    Alexander     Operations    19536        47.0
97   HT8585      Elena      Griffin     Operations    49178        43.0, 'Sales':
ID        Name     Surname WorkDepartment  Salary  Experience
2    LB2444    Michael   Williams          Sales   72195        58.0
6    GD4300     Daniel     Miller          Sales   64451        11.0
7    NK9946     Olivia     Wilson          Sales   66589        49.0
12   EF3588     Andrew    Jackson          Sales   26255         5.0
16   QT6110     Joseph   Thompson          Sales   55506        84.0
22   ME7089    William      Lewis          Sales   67646        37.0
34   ER4294    Richard      Green          Sales   70584        79.0
35   WR3782   Victoria      Adams          Sales   48703        41.0
37   MW3541      Avery   Gonzalez          Sales   59561        17.0
39   BU9606      Sofia     Carter          Sales   58297        50.0
48   FY3880      Henry    Edwards          Sales   29514        87.0
67   HX0309       Leah   Peterson          Sales   18183        45.0
79   VC2960      Riley       Ross          Sales   72681        31.0
80   FV7585      Isaac  Henderson          Sales   27565         8.0
81   YB9747    Aaliyah    Coleman          Sales   34310        85.0
82   SB3766  Christian    Jenkins          Sales   60564        47.0
93   JI1684  Gabriella   Gonzales          Sales   29843        91.0}
```

In [ ]:

```
# Now we can create our data frame for each department by accessing our dictionary
sales_df = grouped_departments['Sales']
account_df = grouped_departments['Account']
hhrr_df = grouped_departments['HHRR']
operations_df = grouped_departments['Operations']
marketing_df = grouped_departments['Marketing']
sales_df
```

Out[ ]:

	ID	Name	Surname	WorkDepartment	Salary	Experience
2	LB2444	Michael	Williams	Sales	72195	58.0
6	GD4300	Daniel	Miller	Sales	64451	11.0
7	NK9946	Olivia	Wilson	Sales	66589	49.0
12	EF3588	Andrew	Jackson	Sales	26255	5.0
16	QT6110	Joseph	Thompson	Sales	55506	84.0
22	ME7089	William	Lewis	Sales	67646	37.0
34	ER4294	Richard	Green	Sales	70584	79.0
35	WR3782	Victoria	Adams	Sales	48703	41.0
37	MW3541	Avery	Gonzalez	Sales	59561	17.0
39	BU9606	Sofia	Carter	Sales	58297	50.0
48	FY3880	Henry	Edwards	Sales	29514	87.0
67	HX0309	Leah	Peterson	Sales	18183	45.0
79	VC2960	Riley	Ross	Sales	72681	31.0
80	FV7585	Isaac	Henderson	Sales	27565	8.0
81	YB9747	Aaliyah	Coleman	Sales	34310	85.0
82	SB3766	Christian	Jenkins	Sales	60564	47.0
93	JI1684	Gabriella	Gonzales	Sales	29843	91.0

4.4.d Number of employees per department

In [ ]:

```
# Let's count the employees per department using a function
def countEmployees(df):
    return df.shape[0]

sales_count = countEmployees(sales_df)
account_count = countEmployees(account_df)
hhrr_count = countEmployees(hhrr_df)
operations_count = countEmployees(operations_df)
marketing_count = countEmployees(marketing_df)

print(
    "Number of employees in Sales: ", sales_count, "\n",
    "Number of employees in HHRR: ", hhrr_count, "\n",
    "Number of employees in Account: ", account_count, "\n",
    "Number of employees in Marketing: ", marketing_count, "\n",
    "Number of employees in Operations: ", operations_count,
    )
```

```
Number of employees in Sales:  17
 Number of employees in HHRR:  24
 Number of employees in Account:  13
 Number of employees in Marketing:  24
 Number of employees in Operations:  22
```

4.5 Correlation between Salary and Experience per department

Now that we have separated the data into departments, we can analyze the correlation between Salary and Experience per department

-> For this calculations we are not going to use the prediction model <-

4.5.a Sales Department

In [ ]:
```
# Using a scatter plot to visualize the correlation between Salary and Experience
x = sales_df['Experience']
y = sales_df['Salary']
color = '#9ba64b'
plot = plt.scatter(x = x, y = y, color=color)
#obtain m (slope) and b(intercept) of linear regression line
m, b = np.polyfit(x, y, 1)
lreg = np.corrcoef(x, y)
plt.plot(x, m*x+b, color='red')
# Labels and title
plt.xlabel('Experience')
plt.ylabel('Salary')
plt.suptitle('Sales Department', style='italic', fontsize=14, color='red')
plt.title('Correlation between Salary and Experience ')
print(lreg)
```

```
[[ 1.         -0.06433463]
 [-0.06433463  1.        ]]
```

4.5.b HHRR Department

In [ ]:
```
# Using a scatter plot to visualize the correlation between Salary and Experience
x = hhrr_df['Experience']
y = hhrr_df['Salary']
color = '#9ba64b'
plot = plt.scatter(x = x, y = y, color=color)
#obtain m (slope) and b(intercept) of linear regression line
m, b = np.polyfit(x, y, 1)
lreg = np.corrcoef(x, y)
```

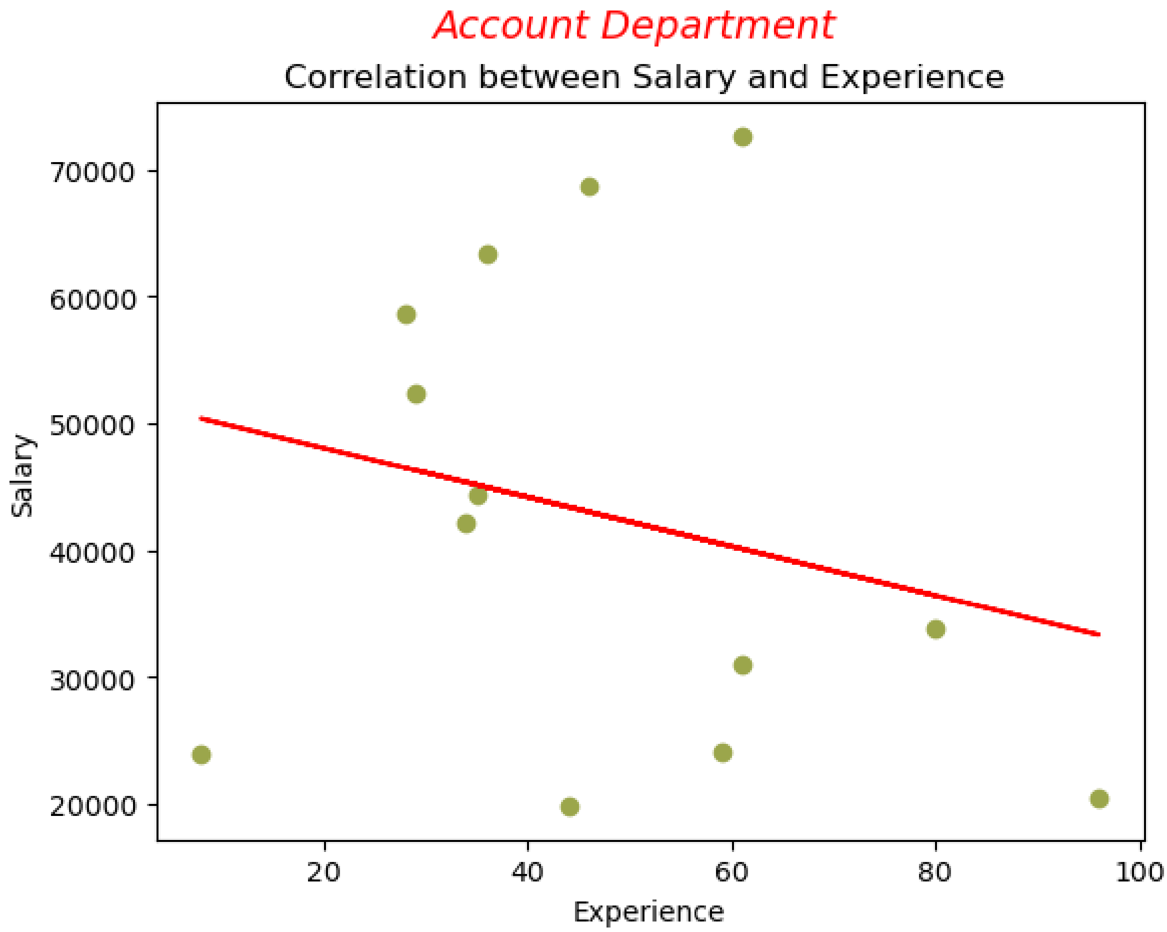
```
plt.plot(x, m*x+b, color='red')
# Labels and title
plt.xlabel('Experience')
plt.ylabel('Salary')
plt.suptitle('HHRR Department', style='italic', fontsize=14, color='red')
plt.title('Correlation between Salary and Experience ')
print(lreg)
```

```
[[ 1.         -0.03556674]
 [-0.03556674  1.        ]]
```

4.5.c Account Department

In [ ]:
```
x = account_df['Experience']
y = account_df['Salary']
color = '#9ba64b'
plot = plt.scatter(x = x, y = y, color=color)
#obtain m (slope) and b(intercept) of linear regression line
m, b = np.polyfit(x, y, 1)
lreg = np.corrcoef(x, y)
plt.plot(x, m*x+b, color='red')
# Labels and title
plt.xlabel('Experience')
plt.ylabel('Salary')
plt.suptitle('Account Department', style='italic', fontsize=14, color='red')
plt.title('Correlation between Salary and Experience ')
print(lreg)
```

```
[[ 1.         -0.24160341]
 [-0.24160341  1.        ]]
```



4.5.d Marketing Department

In [ ]:
```
x = marketing_df['Experience']
y = marketing_df['Salary']
color = '#9ba64b'
plot = plt.scatter(x = x, y = y, color=color)
#obtain m (slope) and b(intercept) of linear regression line
m, b = np.polyfit(x, y, 1)
lreg = np.corrcoef(x, y)
plt.plot(x, m*x+b, color='red')
# Labels and title
plt.xlabel('Experience')
plt.ylabel('Salary')
plt.suptitle('Marketing Department', style='italic', fontsize=14, color='red')
plt.title('Correlation between Salary and Experience ')
print(lreg)
```

```
[[1.         0.07352061]
 [0.07352061 1.        ]]
```

4.5.e Operations Department

In [ ]:
```
x = operations_df['Experience']
y = operations_df['Salary']
color = '#9ba64b'
plot = plt.scatter(x = x, y = y, color=color)
#obtain m (slope) and b(intercept) of linear regression line
m, b = np.polyfit(x, y, 1)
lreg = np.corrcoef(x, y)
plt.plot(x, m*x+b, color='red')
# Labels and title
plt.xlabel('Experience')
plt.ylabel('Salary')
plt.suptitle('Operations Department', style='italic', fontsize=14, color='red')
plt.title('Correlation between Salary and Experience ')
print(lreg)
```

```
[[1.         0.02472314]
 [0.02472314 1.        ]]
```


Operations Department
Correlation between Salary and Experience
70000
60000
50000
40000
30000
20000
Salary
20
40
60
80
Experience