RRHH - Proyecto de People Analytics

Organizando la informaciñon y resolviendo Analíticas de RRHH.

La base de datos es de sitio Kaggle:

Usuario: *PAVANSUBHASH*

Título: *IBM HR Analytics Employee Attrition & Performance*

Link: https://www.kaggle.com/datasets/pavansubhasht/ibm-hr-analytics-attrition-dataset

El motivo de este proyecto es para practicar mis habilidades analíticas, utilizando una base real de RRHH. Para llevar a cabo el análisis usaré mis conocimientos en ***Python, Excel y Power BI***. En lo posible voy a intentar aplicar modelos de Machine Learning.

Analizaré los datos para obtener insights valiosos que permitan tomar medidas relacionadas con los RRHH. Una vez finalizada la exploración y logrados los insights, intentaré añadir datos para crear una comparación de las métricas, simulando el paso de los años pasados.

Trabajaré con el archivo '***WA_Fn-UseC_-HR-Employee-Attrition***', descargado de una de las bases del sitio web Kaggle.

Me plantearé hipótesis que deberán ser confirmadas o rechazadas por los datos. Luego diseñaré los dashboards necesarios para visualizar los resultados de las hipótesis planteadas. Por último, se detallarán las conclusiones alcanzadas.

1. Importando las librerias

In [ ]:
```
import pandas as pd
import numpy as np

# Este comando es para ingorar las advertencias. Fué una recomendación de mi amigo
import warnings
warnings.filterwarnings('ignore')
```

2. Importando nuestro base de datos

Importamos nuestra base de datos del archivo'***WA_Fn-UseC_-HR-Employee-Attrition***' para poder trabajar con ella. La misma es un archivo .csv. Primero voy a explorar los datos, limpiarlos y remover todos aquelos valores que no sean necesarios para nuestro análisis.

In [ ]:
```
df_rawdata = pd.read_csv('WA_Fn_UseC_HR_Employee_Attrition.csv')
df_rawdata.head(5)
```

Out[ ]:

	Age	Attrition	BusinessTravel	DailyRate	Department	DistanceFromHome	Education	Educa
0	41	Yes	Travel_Rarely	1102	Sales	1	2	Life
1	49	No	Travel_Frequently	279	Research & Development	8	1	Life
2	37	Yes	Travel_Rarely	1373	Research & Development	2	2	
3	33	No	Travel_Frequently	1392	Research & Development	3	4	Life
4	27	No	Travel_Rarely	591	Research & Development	2	1	

5 rows × 35 columns

In [ ]:
```
df_rawdata.columns
```

Out[ ]:
```
Index(['Age', 'Attrition', 'BusinessTravel', 'DailyRate', 'Department',
       'DistanceFromHome', 'Education', 'EducationField', 'EmployeeCount',
       'EmployeeNumber', 'EnvironmentSatisfaction', 'Gender', 'HourlyRate',
       'JobInvolvement', 'JobLevel', 'JobRole', 'JobSatisfaction',
       'MaritalStatus', 'MonthlyIncome', 'MonthlyRate', 'NumCompaniesWorked',
       'Over18', 'OverTime', 'PercentSalaryHike', 'PerformanceRating',
       'RelationshipSatisfaction', 'StandardHours', 'StockOptionLevel',
       'TotalWorkingYears', 'TrainingTimesLastYear', 'WorkLifeBalance',
       'YearsAtCompany', 'YearsInCurrentRole', 'YearsSinceLastPromotion',
       'YearsWithCurrManager'],
      dtype='object')
```

In [ ]:
```
df_rawdata.info()
```

```
<class 'pandas.core.frame.DataFrame'>
RangeIndex: 1470 entries, 0 to 1469
Data columns (total 35 columns):
 #   Column                    Non-Null Count  Dtype 
---  ------                    --------------  ----- 
 0   Age                       1470 non-null   int64 
 1   Attrition                 1470 non-null   object
 2   BusinessTravel            1470 non-null   object
 3   DailyRate                 1470 non-null   int64 
 4   Department                1470 non-null   object
 5   DistanceFromHome          1470 non-null   int64 
 6   Education                 1470 non-null   int64 
 7   EducationField            1470 non-null   object
 8   EmployeeCount             1470 non-null   int64 
 9   EmployeeNumber            1470 non-null   int64 
 10  EnvironmentSatisfaction   1470 non-null   int64 
 11  Gender                    1470 non-null   object
 12  HourlyRate                1470 non-null   int64 
 13  JobInvolvement            1470 non-null   int64 
 14  JobLevel                  1470 non-null   int64 
 15  JobRole                   1470 non-null   object
 16  JobSatisfaction           1470 non-null   int64 
 17  MaritalStatus             1470 non-null   object
 18  MonthlyIncome             1470 non-null   int64 
 19  MonthlyRate               1470 non-null   int64 
 20  NumCompaniesWorked        1470 non-null   int64 
 21  Over18                    1470 non-null   object
 22  OverTime                  1470 non-null   object
 23  PercentSalaryHike         1470 non-null   int64 
 24  PerformanceRating         1470 non-null   int64 
 25  RelationshipSatisfaction  1470 non-null   int64 
 26  StandardHours             1470 non-null   int64 
 27  StockOptionLevel          1470 non-null   int64 
 28  TotalWorkingYears         1470 non-null   int64 
 29  TrainingTimesLastYear     1470 non-null   int64 
 30  WorkLifeBalance           1470 non-null   int64 
 31  YearsAtCompany            1470 non-null   int64 
 32  YearsInCurrentRole        1470 non-null   int64 
 33  YearsSinceLastPromotion   1470 non-null   int64 
 34  YearsWithCurrManager      1470 non-null   int64 
dtypes: int64(26), object(9)
memory usage: 402.1+ KB
```

3 Limpiando los Datos

Con el fin de hacer la limpieza de datos, eliminaré las columnas que no usaré para el análisis." Sin embargo, antes de hacerlo, crearé una copia de la base de datos "df_rawdata", lo que me permitirá recuperar la Base de datos original si es necesario, en caso de que se haya eliminado un dato por error.

3.1 Backup de la Base de Datos

In [ ]:
```
# The copy will be called df_padb from dataframe peopleanalyticsdatabase
df_padb = df_rawdata.copy()
df_padb.head()
```

Out[ ]:

	Age	Attrition	BusinessTravel	DailyRate	Department	DistanceFromHome	Education	Educa
0	41	Yes	Travel_Rarely	1102	Sales	1	2	Life
1	49	No	Travel_Frequently	279	Research & Development	8	1	Life
2	37	Yes	Travel_Rarely	1373	Research & Development	2	2	
3	33	No	Travel_Frequently	1392	Research & Development	3	4	Life
4	27	No	Travel_Rarely	591	Research & Development	2	1	

5 rows × 35 columns

In [ ]:
```
# Verificando por valores faltantes
missing_values = df_padb.isnull().sum()
print('Number of missing values: ', missing_values)
```

```
Number of missing values:  Age                         0
Attrition                   0
BusinessTravel              0
DailyRate                   0
Department                  0
DistanceFromHome            0
Education                   0
EducationField              0
EmployeeCount               0
EmployeeNumber              0
EnvironmentSatisfaction     0
Gender                      0
HourlyRate                  0
JobInvolvement              0
JobLevel                    0
JobRole                     0
JobSatisfaction             0
MaritalStatus               0
MonthlyIncome               0
MonthlyRate                 0
NumCompaniesWorked          0
Over18                      0
OverTime                    0
PercentSalaryHike           0
PerformanceRating           0
RelationshipSatisfaction    0
StandardHours               0
StockOptionLevel            0
TotalWorkingYears           0
TrainingTimesLastYear       0
WorkLifeBalance             0
YearsAtCompany              0
YearsInCurrentRole          0
YearsSinceLastPromotion     0
YearsWithCurrManager        0
dtype: int64
```

In [ ]:
```
# Eliminando las columnas que no se utilizarán en el análisis
df_padb.drop(['BusinessTravel', 'DailyRate', 'EmployeeNumber', 'MaritalStatus', 'Nu
df_padb.columns
```

Out[ ]:
```
Index(['Age', 'Attrition', 'Department', 'DistanceFromHome', 'Education',
       'EducationField', 'EnvironmentSatisfaction', 'Gender', 'HourlyRate',
       'JobInvolvement', 'JobLevel', 'JobRole', 'JobSatisfaction',
       'MonthlyIncome', 'MonthlyRate', 'OverTime', 'TotalWorkingYears',
       'TrainingTimesLastYear', 'WorkLifeBalance', 'YearsAtCompany',
       'YearsInCurrentRole', 'YearsSinceLastPromotion'],
      dtype='object')
```

3.2 Creando una columna para los ID

Para los ID voy a utilizar un código de dos letras de los valores de la columna 'Department' y cuatro números random creando una combinación, la cual terminará por establecer un ID único para cada empleado.

In [ ]:
```
# Contamos cuántos departamentos posee nuestra base de datos
departments = df_padb['Department'].value_counts()
print(departments)
```

```
Research & Development    961
Sales                     446
Human Resources            63
Name: Department, dtype: int64
```

Para generar el código de dos letras con los valores de la columna '*Department*', voy a crear una función. Luego voy un array de números enteros de 4 cifras. Para poder desplegar los valores dentro de la columna '*ID*' voy a fusionar ambos valores, el código de 2 letras y los 4 números, y los guardaré dentro de una variable.

In [ ]:
```
# Función para crear los códigos de la columna departamentos
def departments_code(departments):
    if 'Research & Development' in departments:
        return 'RD'
    elif 'Sales' in departments:
        return 'SL'
    else:
        return 'HR'

# Creamos los números random
random_number = np.random.randint(1000, 9999, size=len(df_padb))

# Unimos los valores
new_data = df_padb['Department'].apply(departments_code) + pd.Series(random_number)

# Creamos una nueva columna con el nombre 'ID' y le añadimos los valores anteriores
df_padb.insert(0, 'ID', new_data, True)

df_padb.head()
```

Out[ ]:

	ID	Age	Attrition	Department	DistanceFromHome	Education	EducationField	Environm
0	SL9988	41	Yes	Sales	1	2	Life Sciences	
1	RD8984	49	No	Research & Development	8	1	Life Sciences	
2	RD7769	37	Yes	Research & Development	2	2	Other	
3	RD5670	33	No	Research & Development	3	4	Life Sciences	
4	RD6839	27	No	Research & Development	2	1	Medical	

5 rows × 23 columns

In [ ]:
```
df_padb['ID'].dtype
```

Out[ ]:
```
dtype('O')
```

3.3 Reemplazamos los valores de las columnas *'Education', 'EnvironmentSatisfaction', 'JobInvolvement', 'JobSatisfaction', 'PerformanceRating', 'RelationshipSatisfaction', and 'WorkLifeBalance'*

Para mejorar el entendimiento de los datos y facilitar el trabajo con ellos, voy a modificar los valores de dichas columnas. Para poder trabajar con ellos, primero modificaré los valores del tipo entero al tipo string y luego modificaré los valores por los de su referencia.

In [ ]:
```
# Columna 'Education'
education_ref = {
    1: 'Below College',
    2: 'College',
    3: 'Bachelor',
    4: 'Master',
    5: 'Doctor'
}

df_padb['Education'] = df_padb['Education'].map(education_ref)

# Columna 'EnvironmentSatisfaction'
environment_satisfaction_ref = {
    1: 'Low',
    2: 'Medium',
    3: 'High',
    4: 'Very High'
}

df_padb['EnvironmentSatisfaction'] = df_padb['EnvironmentSatisfaction'].map(enviror

# Columna 'JobInvolvement'
job_involvement_ref = {
    1: 'Low',
    2: 'Medium',
    3: 'High',
    4: 'Very High'
}

df_padb['JobInvolvement'] = df_padb['JobInvolvement'].map(job_involvement_ref)
```

```
# Columna 'JobSatisfaction'
job_satisfaction_ref = {
    1: 'Low',
    2: 'Medium',
    3: 'High',
    4: 'Very High'
}

df_padb['JobSatisfaction'] = df_padb['JobSatisfaction'].map(job_satisfaction_ref)

# Columna 'PerformanceRating'
performance_rating_ref = {
    1: 'Low',
    2: 'Good',
    3: 'Excellent',
    4: 'Outstanding'
}

# Columna 'WorkLifeBalance'
wlb_ref = {
    1: 'Bad',
    2: 'Good',
    3: 'Better',
    4: 'Best'
}

df_padb['WorkLifeBalance'] = df_padb['WorkLifeBalance'].map(wlb_ref)

df_padb.head()
```

Out[ ]:

	ID	Age	Attrition	Department	DistanceFromHome	Education	EducationField	Environm
0	SL9988	41	Yes	Sales	1	College	Life Sciences	
1	RD8984	49	No	Research & Development	8	Below College	Life Sciences	
2	RD7769	37	Yes	Research & Development	2	College	Other	
3	RD5670	33	No	Research & Development	3	Master	Life Sciences	
4	RD6839	27	No	Research & Development	2	Below College	Medical	

5 rows × 23 columns

3.4 Exportamos nuestros datos a una hoja de Excel

Ahora que nuestros datos se encuentran listos para su análisis, podemos exportar nuestra base de datos una hoja de Excel. También vamos a crear un archivo .csv.

In [ ]:
```
# Exportando los datos a una hoja de cálculos Excel
df_padb.to_excel('imb_analytics_2021.xlsx', sheet_name='hr_analytics_2021', index=F

# Exportando los datos a un archivo .csv
df_padb.to_csv('ibm_hranalytics_2021.csv', index=False)
```

4 Trabajamos con la ISO 30414-2018

Ya con nuestros datos limpios, es momento de utilizar la ISO y comenzar a analizar los datos con el fin de obtener información de relevancia.

4.1 DIVERSIDAD

El siguiente análisis lo establecemos con el fin de conocer la composición y el nivel de diversificación de la empresa. Para ello voy a utilizar la información que nos provee la ISO.

4.1.a Creando una columna con el rango de edad

Antes de comenzar a trabajar con nuestros datos, voy a crear una columna para establecer rangos de edad, con el fin de facilitar el entendimiento de nuestros datos. Así podremos conocer las diferentes generaciones que componen la empresa.

In [ ]:
```
# Función para crear los rangos de edad
def age_range(age):
    if age >= 18 and age <= 27:
        return '18 to 27'
    elif age >= 28 and age <= 37:
        return '28 to 37'
    elif age >= 38 and age <= 47:
        return '38 to 47'
    elif age >= 48 and age <= 57:
        return '48 to 57'
    else:
        return 'more than 58'

# Creamos un array con los datos nuevos
new_age_data = df_padb['Age'].apply(lambda x: pd.Series(age_range(x)))
# print(new_age_data)

# Creamos e insertamos una nueva columna con los datos nuevos
df_padb.insert(loc=df_padb.columns.get_loc('Age')+1, column='AgeRange', value=new_a
df_padb.head()
```

Out[ ]:

	ID	Age	AgeRange	Attrition	Department	DistanceFromHome	Education	EducationField
0	SL9988	41	38 to 47	Yes	Sales	1	College	Life Sciences
1	RD8984	49	48 to 57	No	Research & Development	8	Below College	Life Sciences
2	RD7769	37	28 to 37	Yes	Research & Development	2	College	Othe
3	RD5670	33	28 to 37	No	Research & Development	3	Master	Life Sciences
4	RD6839	27	18 to 27	No	Research & Development	2	Below College	Medica

5 rows × 24 columns

4.1.b Analizamos los datos y creamos los gráficos

Con la columna de los rangos de edad en su lugar, revisamos los datos con el fin de identificar los datos que vamos a necesitar para poder crear los gráficos que nos ilustren el

impacto de la diversidad en la organización. Vamos a conocer los siguientes valores: *study field, study level, gender, and age range*. Esto nos permitirá conocer la diversidad en estudios y género de los trabajadores de la empresa.

Tener un conocimiento de la diversidad de nuestra empresa nos permitirá generar poíticas para establecer parámetros de diversidad que nos ayuden en futuras contrataciones, con el fin de crear un ambiente más diverso.

Voy a utilizar la librería matplotlib para crear los gráficos.

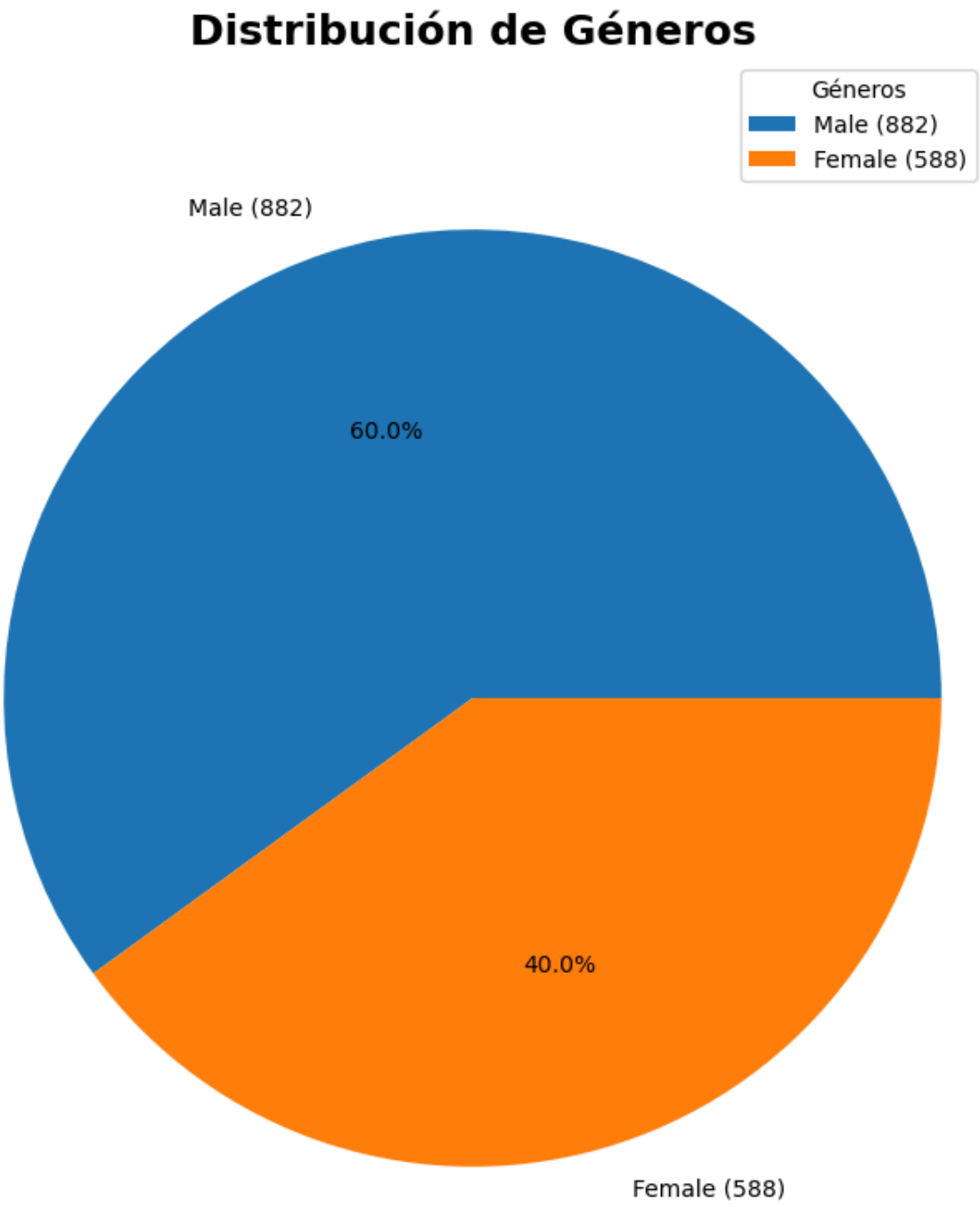
In [ ]:
```
import matplotlib.pyplot as plt
```

4.1.b.1 Género

Examinemos la distribución de género, utilizando los valores de la *columna* '*Gender*'. La distribución de género nos permite conocer cómo se distribuyen los distintos géneros en nuestra empresa. Esto es de gran utilidad para futuras contrataciones.

In [ ]:
```
# Contamos los distintos valores de género y los guardamos en una variable
gender_counts = df_padb['Gender'].value_counts()
print(gender_counts)

# Porcentage y Total de nuestros valores para poder utilizarlos en los gráficos
total_count = gender_counts.sum()
gender_percentage = (gender_counts / total_count) * 100

# labels = gender_counts.index
# Mediante una función creamos las etiquetas de nuestro gráfico
labels = [f'{gender} ({count})' for gender, count in zip(gender_counts.index, gende

# Tamaño del gráfico
fig, ax = plt.subplots(figsize=(8, 10))

# Generamos el gráfico
plt.pie(gender_counts, labels=labels, autopct='%1.1f%%')
plt.legend(title='Géneros')
ax.set_title('Distribución de Géneros', fontsize=18, fontweight='bold')
plt.axis('equal')
plt.show()
```

```
Male      882
Female    588
Name: Gender, dtype: int64
```



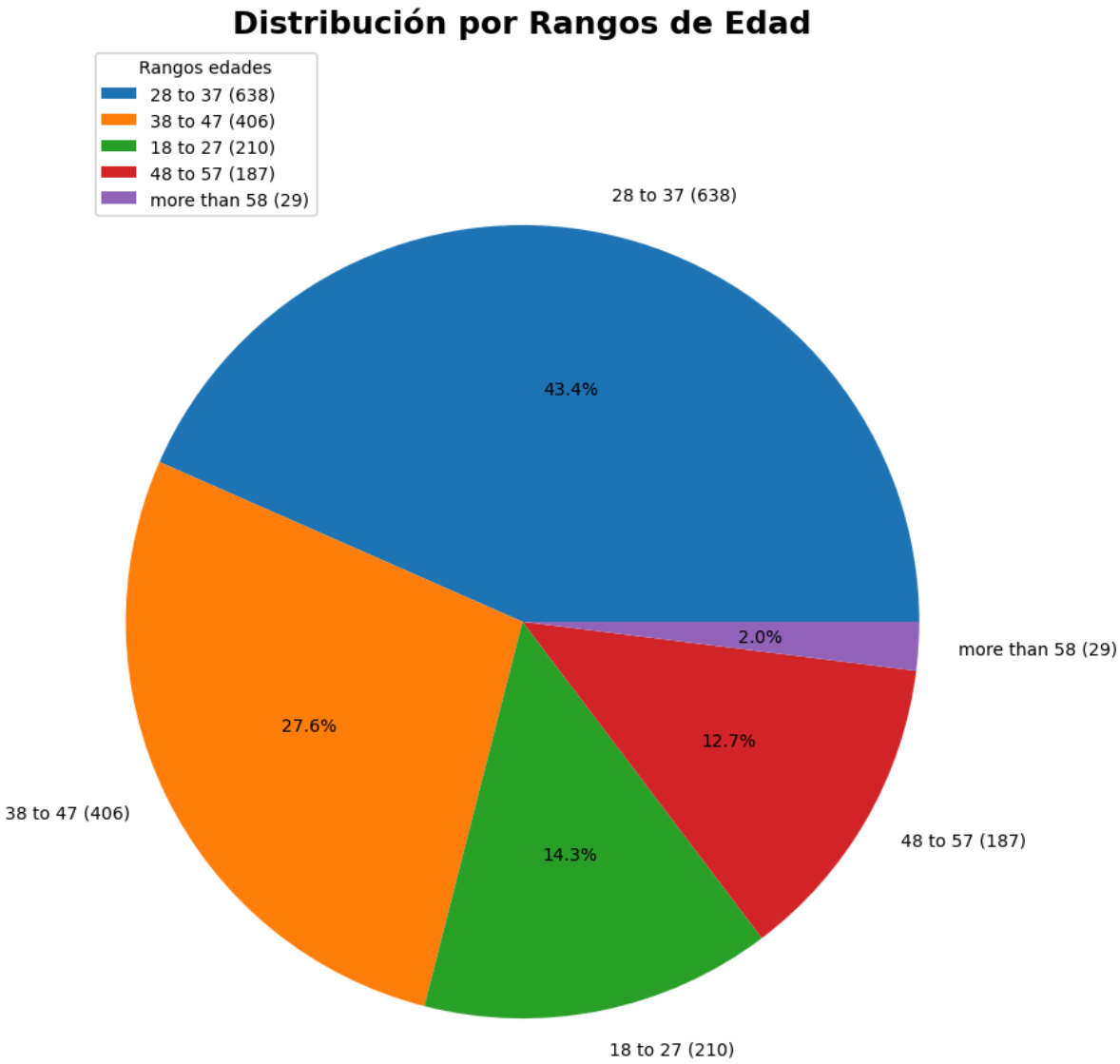
4.1.b.2 Distribución de edades

Continuando con nuestro análisis, ahora vamos a examinar cómo está compuesta por edades nuestra empresa. Para ello, vamos a utilizar los valores de la *columna 'AgeRange'*. Podremos saber la composición de nuestra empresa de acuerdo con los rangos de edades. Este análisis es de utilidad para saber si tenemos varios empleados próximos a retiro.

In [ ]:
```
age_range_count = df_padb['AgeRange'].value_counts()
print(age_range_count)

total_age_count = age_range_count.sum()
age_percentage = (age_range_count / total_age_count) * 100

labels = [f'{age} ({count})' for age, count in zip(age_range_count.index, age_range
```

```
fig, ax = plt.subplots(figsize=(9, 12))


plt.pie(age_range_count, labels=labels, autopct='%1.1f%%')
plt.legend(title='Rangos edades', loc= 'upper left')
ax.set_title('Distribución por Rangos de Edad', fontsize=18, fontweight='bold')
plt.axis('equal')
plt.show()
```

```
28 to 37        638
38 to 47        406
18 to 27        210
48 to 57        187
more than 58     29
Name: AgeRange, dtype: int64
```



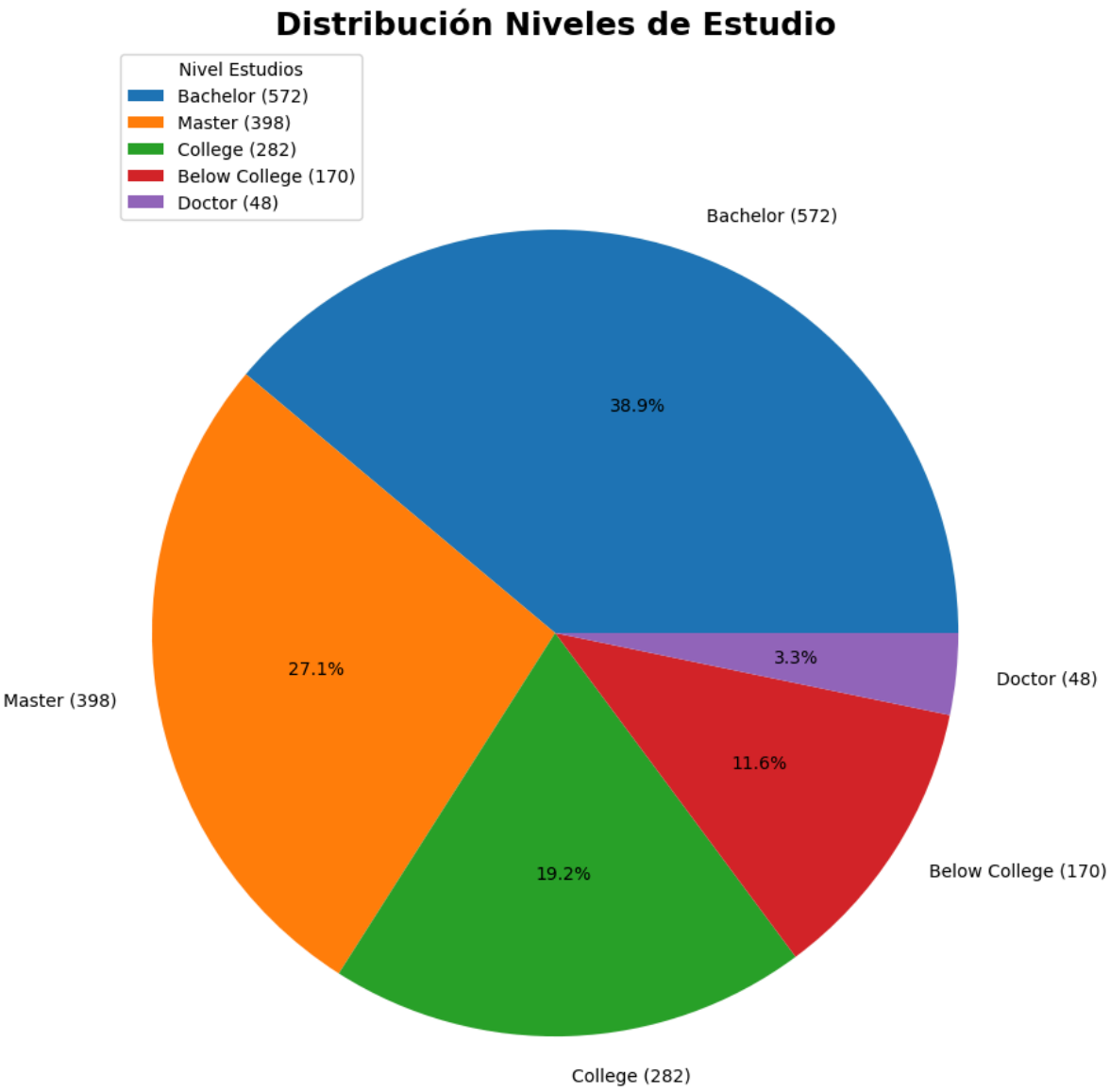
4.1.b.3 Otras distribuciones

¿Qué nivel educativo tienen nuestros empleados? El objetivo de entender la distribución a nivel educativo es brindar asistencia educativa y mejorar el rendimiento académico de nuestros empleados. Para cumplir con esta tarea, vamos a utilizar las columnas "*Education*" y "*EducationField*".

4.1.b.3.a Distribución en Educación

In [ ]:
```
education_count = df_padb['Education'].value_counts()
print(education_count)

total_education_count = education_count.sum()
education_percentage = (education_count / total_education_count) * 100

labels = [f'{education} ({count})' for education, count in zip(education_count.inde

fig, ax = plt.subplots(figsize=(9, 12))


plt.pie(education_count, labels=labels, autopct='%1.1f%%')
plt.legend(title='Nivel Estudios', loc= 'upper left')
ax.set_title('Distribución Niveles de Estudio', fontsize=18, fontweight='bold')
plt.axis('equal')
plt.show()
```

```
Bachelor         572
Master           398
College          282
Below College    170
Doctor            48
Name: Education, dtype: int64
```

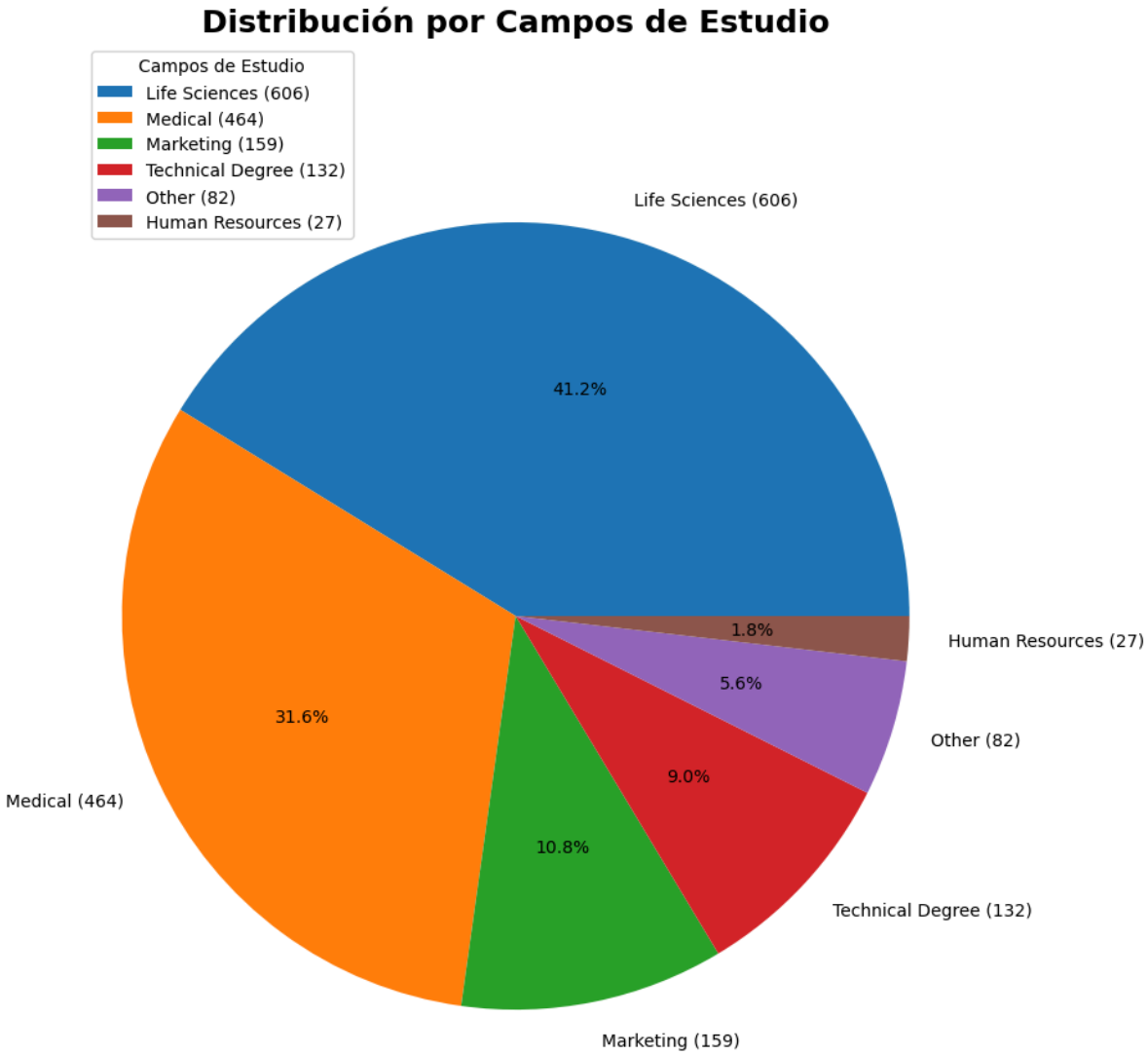


4.1.b.3.b Distribucion por campos de estudio

El objetivo del análisis posterior es determinar cómo se distribuyen las personas según los distintos títulos educativos que poseen.

In [ ]:
```
educationfield_count = df_padb['EducationField'].value_counts()
print(educationfield_count)

total_educationfield_count = educationfield_count.sum()
educationfield_percentage = (educationfield_count / total_educationfield_count) * 1

labels = [f'{education} ({count})' for education, count in zip(educationfield_count

fig, ax = plt.subplots(figsize=(9, 12))


plt.pie(educationfield_count, labels=labels, autopct='%1.1f%%')
plt.legend(title='Campos de Estudio', loc= 'upper left')
ax.set_title('Distribución por Campos de Estudio', fontsize=18, fontweight='bold')
plt.axis('equal')
plt.show()
```

```
Life Sciences       606
Medical             464
Marketing           159
Technical Degree    132
Other                82
Human Resources      27
Name: EducationField, dtype: int64
```



4.1.c Apéndice

Agrego un par de cálculos en el siguiente apéndice para usarlos en futuros análisis. Recordemos que nuestro objetivo es *comparar y comprender cómo evoluciona la diversidad a lo largo del tiempo*.

In [ ]:
```
# Calculamos la edad promedio de los empleados
age_info = df_padb['Age'].describe()
print(age_info)

age_sum = df_padb['Age'].mean().round(0)
print("\nLa edad promedio es:")
print(age_sum)
```

```
count    1470.000000
mean       36.923810
std         9.135373
min        18.000000
25%        30.000000
50%        36.000000
75%        43.000000
max        60.000000
Name: Age, dtype: float64

La edad promedio es:
37.0
```

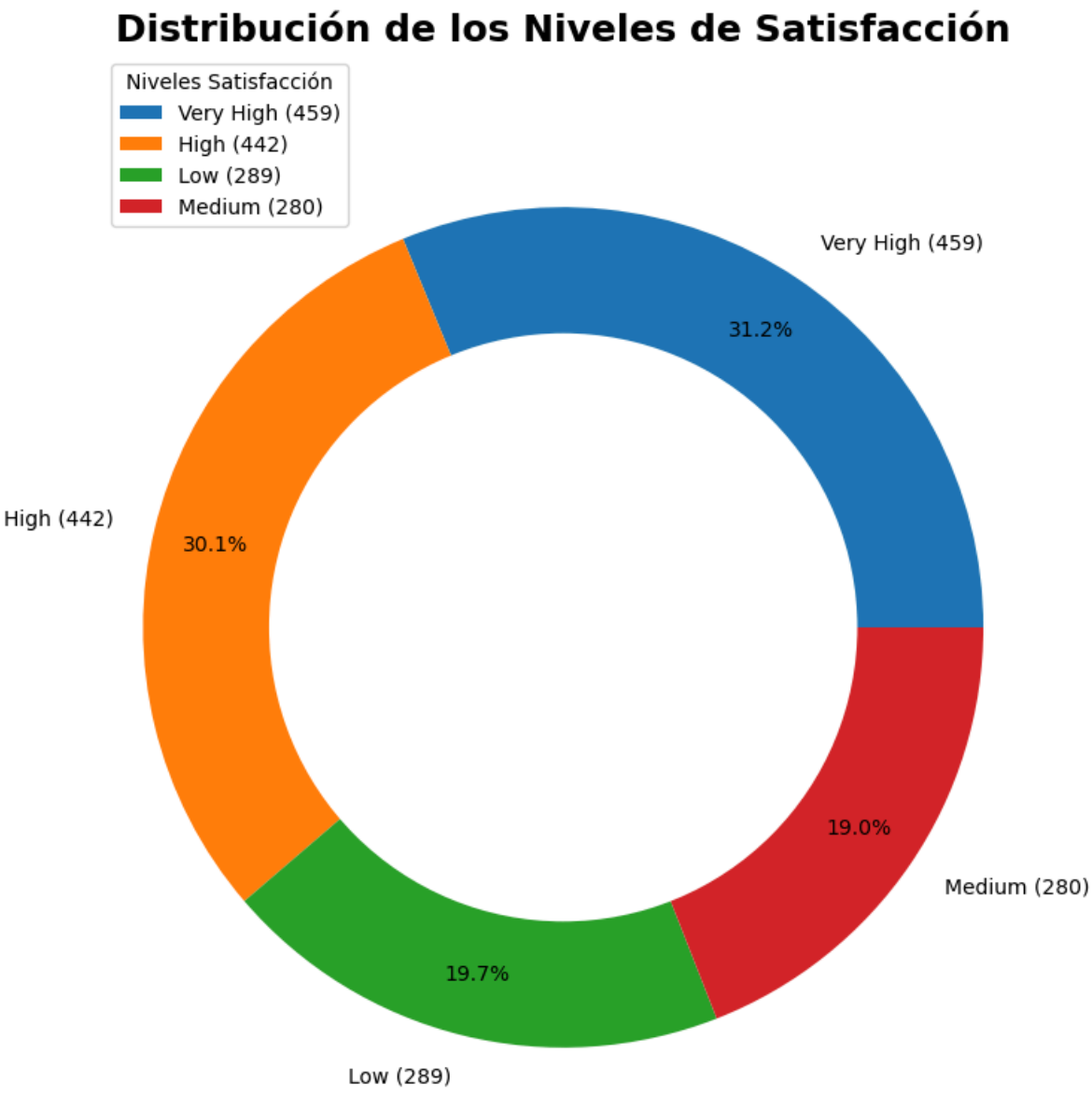
4.2 Satisfacción Laboral

Examinemos los niveles de satisfacción de los empleados en los diferentes roles de la empresa. ¿Cuál rol tiene el mayor o el menor grado de satisfacción? Con esta métrica, podemos identificar aquellos roles donde *satisfacción es **baja***, descubrir lo que está causando los niveles bajos, y diseñar o crear estrategias para mejorarlo.

Voy a empezar contando la distribución de las categorías de satisfacción.

In [ ]:
```
# Contando los valores de las difrentes categorias de satisfacción
job_satisfaction_count = df_padb['JobSatisfaction'].value_counts()
print(job_satisfaction_count)

# Sumatoria total
job_satisfaction_total = df_padb['JobSatisfaction'].sum()

# Creando las etiquetas para el gráfico
labels = [f'{jobsatisfaction} ({count})' for jobsatisfaction, count in zip(job_sati

# Creando el gráfico
fig, ax = plt.subplots(figsize=(8, 10))

plt.pie(job_satisfaction_count, labels=labels, autopct='%1.1f%%', pctdistance=0.85)
plt.legend(title='Niveles Satisfacción', loc= 'upper left')
ax.set_title('Distribución de los Niveles de Satisfacción', fontsize=18, fontweight
plt.axis('equal')
# Dibujamos un círculo
centre_circle = plt.Circle((0, 0), 0.70, fc='white')
fig = plt.gcf()

# Añadimos el círculo al gráfico
fig.gca().add_artist(centre_circle)
plt.show()
```

```
Very High    459
High         442
Low          289
Medium       280
Name: JobSatisfaction, dtype: int64
```



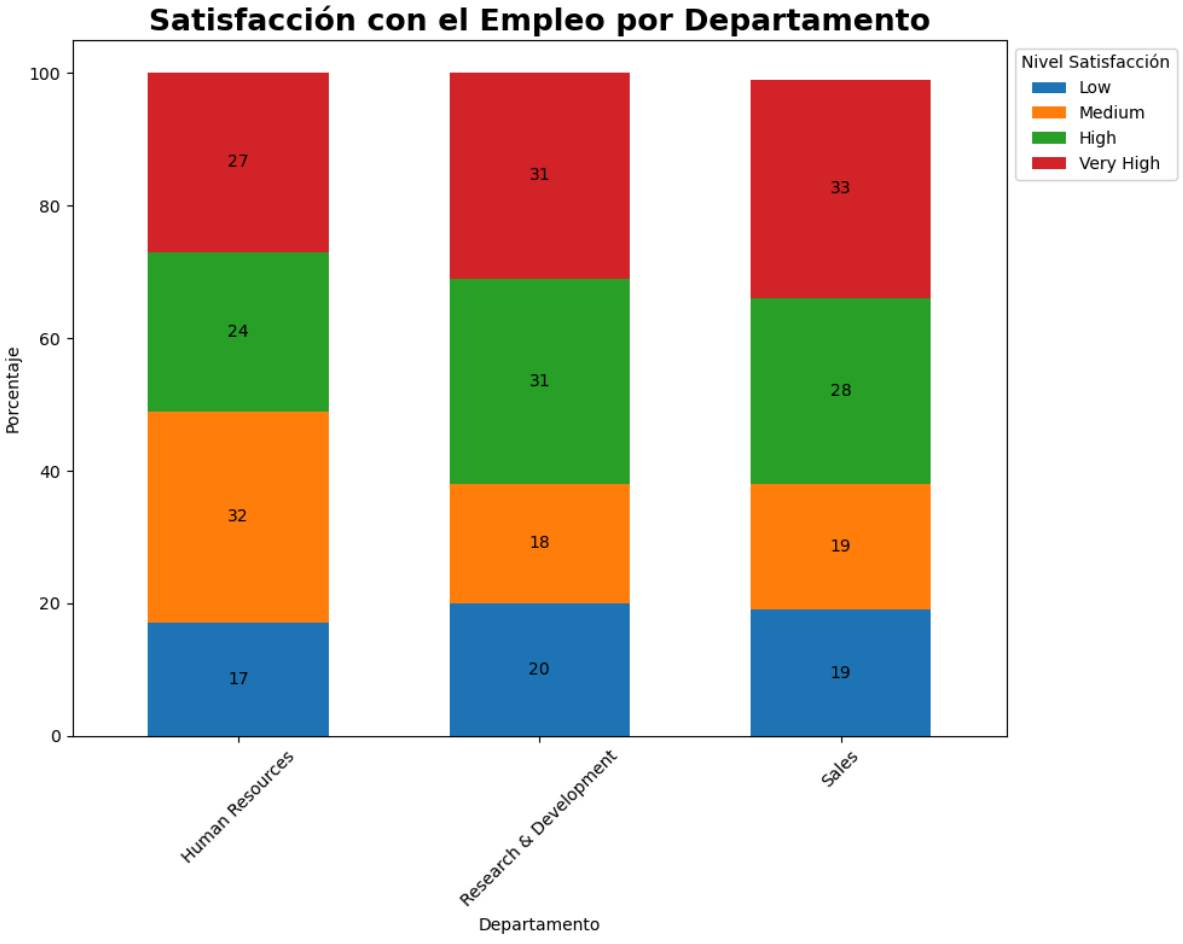
Los datos muestran que ***El 19,7%*** de los empleados tienen Baja satisfacción con su trabajo. Identificaré cuáles de los roles no están tan satisfechos con su trabajo

In [ ]:
```
# Utilizando una tabla dinámica para identificar las respuestas de cada departament
pivot = pd.pivot_table(df_padb[['Department', 'JobSatisfaction']], index='Departmer
print("Conteo Tabla Dinámica:")
print(pivot)

# Calculamos el porcentaje de satisfacción para cada departamento
pivot_percentage = pivot.div(pivot.sum(axis=1), axis=0).round(2) * 100

# Ordenames el orden de los valores
desired_order = ['Low', 'Medium', 'High', 'Very High']

# Aplicamos el nuevo orden
pivot_percentage_ordered = pivot_percentage.reindex(desired_order, axis=1)

print("\nPorcentajes Tabla Dinámica:")
print(pivot_percentage)
```

```
Conteo Tabla Dinámica:
JobSatisfaction         High  Low  Medium  Very High
Department
Human Resources           15   11      20         17
Research & Development   300  192     174        295
Sales                    127   86      86        147

Porcentajes Tabla Dinámica:
JobSatisfaction         High   Low  Medium  Very High
Department
Human Resources         24.0  17.0    32.0       27.0
Research & Development  31.0  20.0    18.0       31.0
Sales                   28.0  19.0    19.0       33.0
```

Creemos un gráfico para visualizar nuestros datos.

In [ ]:
```
# Creamos el gráfico para visualizar los datos
ax = pivot_percentage_ordered.plot(kind='bar', stacked=True, figsize=(10, 8), width

# Añadimos los valores a las barras
for container in ax.containers:
    ax.bar_label(container, label_type='center', fontsize=10)

plt.title('Satisfacción con el Empleo por Departamento', fontsize=18, fontweight='b
plt.xlabel('Departamento')
plt.ylabel('Porcentaje')
plt.xticks(rotation=45)
plt.legend(title='Nivel Satisfacción', bbox_to_anchor=(1, 1), loc='upper left')
plt.tight_layout()
plt.show()
```



Anteriormente, mirando el porcentaje total entre los departamentos, encontramos que el 19,7% de todos los empleados tienen una baja motivación con su trabajo.

Para tener un informe más detallado, se procedió a desglosar por departamentos con el fin de conocer aún más la insatisfacción con el trabajo. Tenemos que tener en cuenta que de los *1470 empleados, 63 pertenecen al departamento de RRHH, 961 al Departamento de Investigación y Desarrollo y 446 a las Ventas*. Los porcentajes de insatisfacción entre los tres departamentos eran muy similares. *RRHH* presentó un **17%** de alrededor de 11 empleados, *R&D* un total de **20%** de cerca de 192 empleados y *Ventas* total de **19%** aproximadamente 86 empleados.

Podemos indicar que en el *departamento I+D* tenemos el mayor número de empleados que están insatisfechos o poco motivados con su trabajo. Se recomienda investigar qué es lo que está causando esto y encontrar algunas soluciones.

Quisiera hacer una evaluación: *en RRHH hay 17% de los empleados con baja motivación y 32% con motivación media*. Se recomienda seguir adelante para ver si estos índices sufren aumentos, si así sucediera, tendríamos la mayoría de la mitad del departamento con una baja motivación con sus tareas. Mientras que los otros dos departamentos mostraron altas tasas de media y alta satisfacción con su trabajo.

4.2.a Apéndice

A fin de continuar mi investigación, vamos a analizar cuántos empleados con valoración **"Muy Alto"** de satisfacción con su trabajo y qué proporción de ellos tienen motivación **"Baja"**. Identificaré a los empleados que muestren motivación **"Baja"** utilizando la columna *"JobInvolvement"*. Quienes tengan participación **"Alta"** y satisfacción **"Baja"** también serán examinados.

In [ ]:
```
# Creamos una mini base con las columnas a trabajar
job_db = df_padb[['JobInvolvement', 'JobSatisfaction']].copy()

# Filtrando los datos
jobFilteredData_VHL = job_db[(job_db['JobInvolvement'] == 'Very High') & (job_db['J
jobFilteredData_HL = job_db[(job_db['JobInvolvement'] == 'High') & (job_db['JobSati

# Conteo de los datos filtrados.
countVH_Low = jobFilteredData_VHL.shape[0]
countH_Low = jobFilteredData_HL.shape[0]
print("El número total de empleados con Very High Job Involvement y Low Job Satisfa
print("El número total de empleados con High Job Involvement y Low Job Satisfactior

# Porcentaje que representan los datos
percentageVH_Low = countVH_Low / len(df_padb) * 100
percentageH_Low = countH_Low / len(df_padb) * 100
print('El porcentaje de empleados con Very High Job Involvement y Low Job Satisfact
print('El porcentaje de empleados con High Job Involvement y Low Job Satisfaction e
```

```
El número total de empleados con Very High Job Involvement y Low Job Satisfaction
es:  34
El número total de empleados con High Job Involvement y Low Job Satisfaction es:
166
El porcentaje de empleados con Very High Job Involvement y Low Job Satisfaction e
s:  2.31
El porcentaje de empleados con High Job Involvement y Low Job Satisfaction es:  1
1.29
```

4.3 Desarrollo de Carrera

¿Hay un crecimiento de carrera decente en la empresa? A través del análisis de datos, mi objetivo es determinar si la organización proporciona a sus empleados una oportunidad decente para el avance profesional.

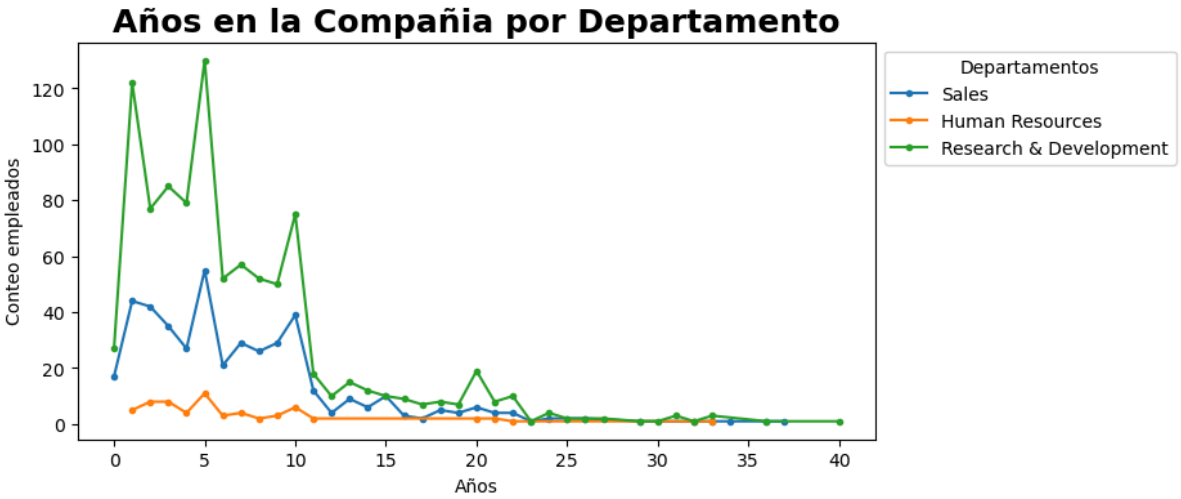
¿Puede el desarrollo de carrera estar relacionado con la baja satisfacción laboral?. Utilizaré para el análisis los datos de la columna 'YearsAtCompany'.

In [ ]:
```
df_total_records = len(df_padb)
print(df_total_records)
```

```
1470
```

In [ ]:
```
departments_count = df_padb['Department'].value_counts()
print('El número de empleados por departamento es: ', "\n", departments_count)
# Preparing the data
sales_count = df_padb[df_padb['Department'] == 'Sales'].groupby('YearsAtCompany').s
rrhh_count = df_padb[df_padb['Department'] == 'Human Resources'].groupby('YearsAtCo
rd_count = df_padb[df_padb['Department'] == 'Research & Development'].groupby('Year

# Creando un gráfico de línea
plt.figure(figsize=(8, 4))

# Graficando las líneas
plt.plot(sales_count.index, sales_count.values, label='Sales', marker='o', ms = 3)
plt.plot(rrhh_count.index, rrhh_count.values, label='Human Resources', marker='o',
plt.plot(rd_count.index, rd_count.values, label='Research & Development', marker='o

# Añadiendo etiquetas al gráfico
plt.xlabel('Años')
plt.ylabel('Conteo empleados')
plt.title('Años en la Compañia por Departamento', fontsize=18, fontweight='bold')
plt.legend(title='Departamentos', bbox_to_anchor=(1, 1), loc='upper left')
plt.show()
```

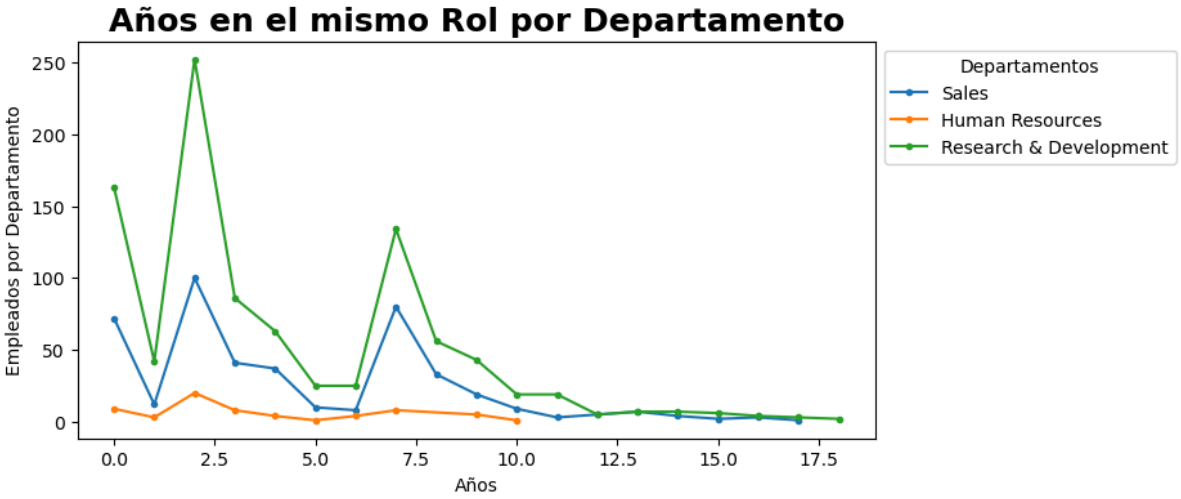
```
El número de empleados por departamento es:
 Research & Development    961
Sales                      446
Human Resources             63
Name: Department, dtype: int64
```



Examinemos cuántos años el empleado permanece en el mismo rol para los distintos departamentos.

In [ ]:
```
# Preparando los datos
sales_count_cr = df_padb[df_padb['Department'] == 'Sales'].groupby('YearsInCurrentR
rrhh_count_cr = df_padb[df_padb['Department'] == 'Human Resources'].groupby('YearsI
```

```
rd_count_cr = df_padb[df_padb['Department'] == 'Research & Development'].groupby('Y

# Creando el gráfico de líneas
plt.figure(figsize=(8, 4))

# Graficando las líneas
plt.plot(sales_count_cr.index, sales_count_cr.values, label='Sales', marker='o', ms
plt.plot(rrhh_count_cr.index, rrhh_count_cr.values, label='Human Resources', marker
plt.plot(rd_count_cr.index, rd_count_cr.values, label='Research & Development', mar

# Añadiendo etiquetas
plt.xlabel('Años')
plt.ylabel("Empleados por Departamento")
plt.title('Años en el mismo Rol por Departamento', fontsize=18, fontweight='bold')
plt.legend(title='Departamentos', bbox_to_anchor=(1, 1), loc='upper left')
plt.show()
```



In [ ]:
```
average_years = df_padb['YearsAtCompany'].mean().round()
average_years_at_role = df_padb['YearsInCurrentRole'].mean().round()
average_years_promotion = df_padb['YearsSinceLastPromotion'].mean().round()
max_years_in_role = df_padb['YearsInCurrentRole'].max()

print(
    "Promedio de Años en la Compañia: ", average_years, "\n",
    "Promedio de Años en un mismo Puesto: ", average_years_at_role, "\n",
    "Promedio de Años desde la última promoción: ", average_years_promotion, "\n",
    "Máximo de Años en un mismo Puesto: ", max_years_in_role)
# print(max_years_in_role)
```

```
Promedio de Años en la Compañia:  7.0
 Promedio de Años en un mismo Puesto:  4.0
 Promedio de Años desde la última promoción:  2.0
 Máximo de Años en un mismo Puesto:  18
```

La mayoría de los empleados pasan *entre 0 y 10 años trabajando para la empresa*. Veremos que la permanencia en la empresa decrece significativamente después de los diez años. Así que traté de averiguar *cuánto tiempo en promedio permanecen los empleados en la empresa* y descubrí que alrededor de unos 7 años, con *4 años en promedio trabajando en la misma posición*. Entonces decidí investigar el sistema de promoción de la empresa y descubrí que nuestra empresa tiene un promedio de dos años para conceder promociones.

Una primera promoción puede verse *alrededor de 2 años* después de unirse a la compañía, y luego una segunda promoción ocurre *5 años más tarde*. Entonces puede que un empleado cambie de empresa, o la persona continúe trabajando en la posición hasta su jubilación.

Con el fin de obtener más información, se examinó el número máximo de años en un solo rol. Se encontró que la duración *más larga en una posición es* ***18 años***.

4.4 ANALISIS del SUELDO

Es hora de analizar *si existe una disparidad significativa entre los* ***salarios de los empleados*** *y los* ***logros educativos***.

Se utilizarán los valores de la columna '*MonthlyIncome*', y los valores se distribuirán de acuerdo con los valores de la columna '*Education*'.

In [ ]:
```
# Para facilitar el trabajo creamos una tabla con las columnas 'MonthlyIncome' y 'E
df_monthedu = df_padb[['MonthlyIncome', 'Education']].copy()
df_monthedu.head()
```

Out[ ]:

	MonthlyIncome	Education
0	5993	College
1	5130	Below College
2	2090	College
3	2909	Master
4	3468	Below College

Es hora de trabajar con nuestro nueva tabla de datos. Calculemos la mediana para cada categoría de educación.

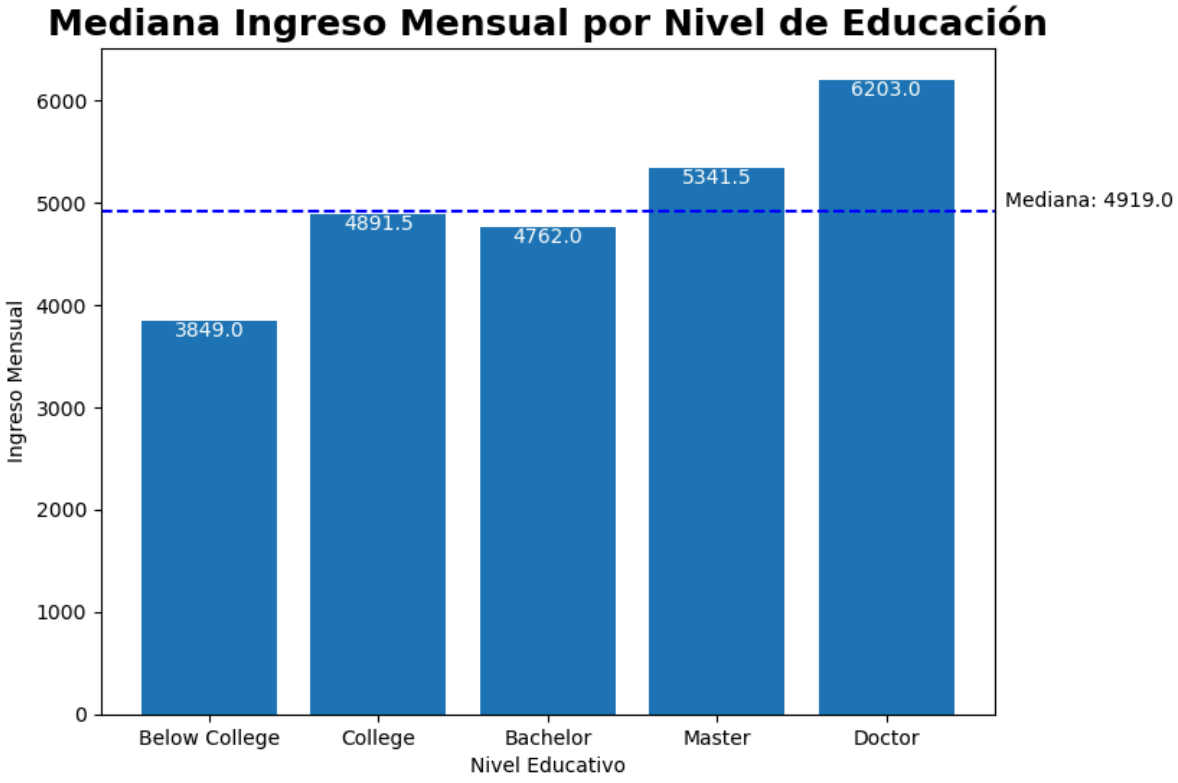
In [ ]:
```
# Calculamos la mediana para la columna 'MonthlyIncome'.
monthlyIncome_median = df_monthedu['MonthlyIncome'].median()

# Second, let's calculate the median for each category of education
below_college_median = df_monthedu[df_monthedu['Education'] == 'Below College'].gro
college_median = df_monthedu[df_monthedu['Education'] == 'College'].groupby('Educat
bachelor_median= df_monthedu[df_monthedu['Education'] == 'Bachelor'].groupby('Educa
master_median = df_monthedu[df_monthedu['Education'] == 'Master'].groupby('Educatio
doctor_median = df_monthedu[df_monthedu['Education'] == 'Doctor'].groupby('Educatio
print(
    "La mediana para Below College Education es: ", below_college_median.values, "\
    "La mediana para College Education es: ", college_median.values, "\n",
    "La mediana para Bachelor Education es: ", bachelor_median.values, "\n",
    "La mediana para Master Education es: ", master_median.values, "\n",
    "La mediana para Doctor Education es: ", doctor_median.values, "\n",
    'La mediana para la columna "MonthlyIncome" es: ', monthlyIncome_median )
```

```
La mediana para Below College Education es:  [[3849.]]
 La mediana para College Education es:  [[4891.5]]
 La mediana para Bachelor Education es:  [[4762.]]
 La mediana para Master Education es:  [[5341.5]]
 La mediana para Doctor Education es:  [[6203.]]
 La mediana para la columna "MonthlyIncome" es:  4919.0
```

Para visualizar nuestros datos, vamos a crear algunos gráficos. Utilizaré un gráfico ***Barras*** para comparar el ingreso mensual mediano para cada nivel de educación.

In [ ]:
```
# Creamos un gráfico de barras
plt.figure(figsize = (8, 6))
```

```
# Añadiendo las variables
bar_data = [below_college_median['MonthlyIncome'].values[0],
            college_median['MonthlyIncome'].values[0],
            bachelor_median['MonthlyIncome'].values[0],
            master_median['MonthlyIncome'].values[0],
            doctor_median['MonthlyIncome'].values[0]]
education_cat = ['Below College', 'College', 'Bachelor', 'Master', 'Doctor']

# Añadiendo etiquetas
for i, value in enumerate(bar_data):
    plt.text(i, value, str(value), ha='center', va='top', color='white')

# Agregamos la línea de promedio
plt.axhline(monthlyIncome_median, color='blue', linestyle='--', label='Media')

# Agregamos el valor de la línea promedio
plt.text(len(education_cat) + 0.7, monthlyIncome_median, f'Mediana: {monthlyIncome_

# Unificamos los datos y creamos el gráfico
plt.bar(education_cat, bar_data)
plt.xlabel('Nivel Educativo')
plt.ylabel('Ingreso Mensual')
plt.title('Mediana Ingreso Mensual por Nivel de Educación', fontsize=18, fontweight
plt.show()
```



Los empleados con '*Bachelor degree*' tienen menos ingresos que los que tienen un '*College degree*', según los hallazgos cuando analizamos mediante la mediana. Pero tenemos que tener en cuenta hay más empleados con títulos de '*Bachelor*' que '*College*', por lo tanto, investigaré si con el promedio, se presenta la misma anormalidad.

In [ ]:
```
# Calculamos el promedio para la columna 'MonthlyIncome'.
monthlyIncome_mean = df_monthedu['MonthlyIncome'].mean().round(2)

# Calculamos el promedio para cada categoría de Nivel Educativo
below_college_mean = df_monthedu[df_monthedu['Education'] == 'Below College'].group
college_mean = df_monthedu[df_monthedu['Education'] == 'College'].groupby('Educatic
```

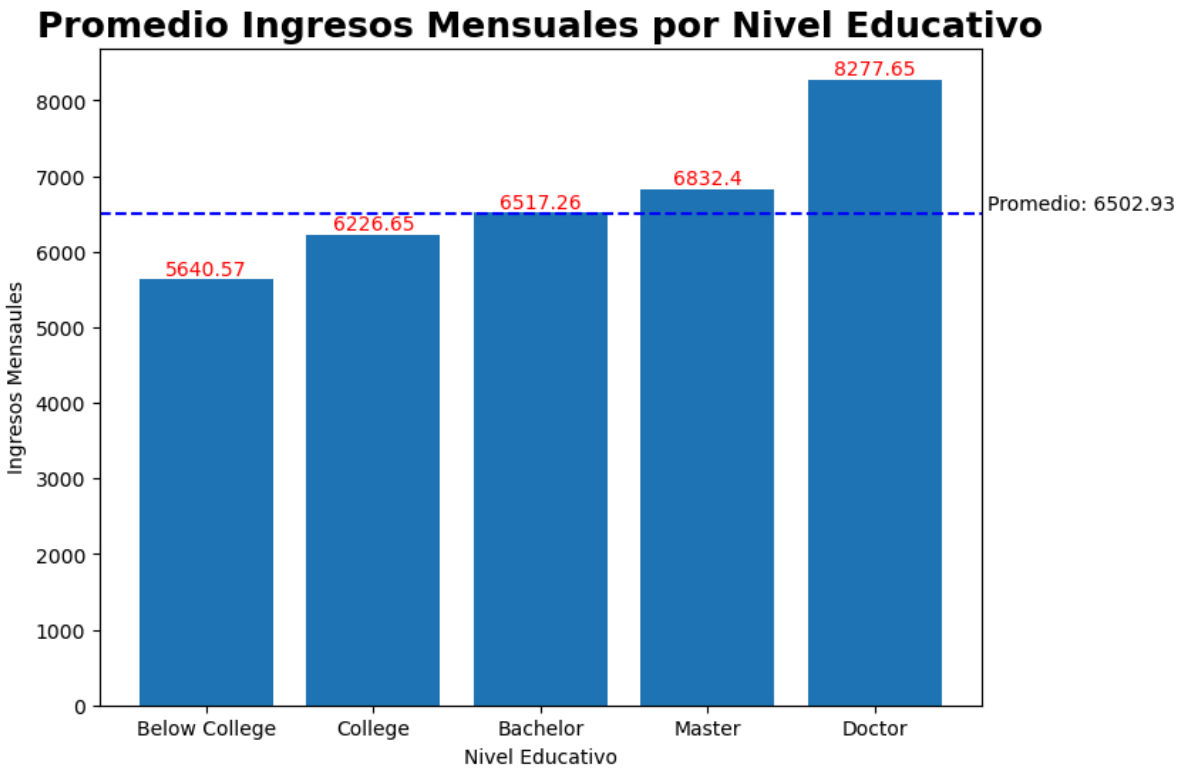
```
bachelor_mean= df_monthedu[df_monthedu['Education'] == 'Bachelor'].groupby('Educati
master_mean = df_monthedu[df_monthedu['Education'] == 'Master'].groupby('Education'
doctor_mean = df_monthedu[df_monthedu['Education'] == 'Doctor'].groupby('Education'
print(
    "El promedio para Below College Education es: ", below_college_mean.values, "\r
    "El promedio para College Education es: ", college_mean.values, "\n",
    "El promedio para Bachelor Education es: ", bachelor_mean.values, "\n",
    "El promedio para Master Education es: ", master_mean.values, "\n",
    "El promedio para Doctor Education es: ", doctor_mean.values, "\n",
    'El promedio para la columna "MonthlyIncome" es: ', monthlyIncome_mean )
```

```
El promedio para Below College Education es:  [[5640.57]]
 El promedio para College Education es:  [[6226.65]]
 El promedio para Bachelor Education es:  [[6517.26]]
 El promedio para Master Education es:  [[6832.4]]
 El promedio para Doctor Education es:  [[8277.65]]
 El promedio para la columna "MonthlyIncome" es:  6502.93
```

Se pudo determinar que los empleados con un título de *Bachelor* tienen más ingresos que los que tienen un título '*College*' al analizar la media. No es algo negativo para la empresa, pero es crucial notar que algunos empleados de con titulos *Bacherlor* no están recibiendo lo suficiente. La media para la titulación *Bachelor's degree* es de **6517,26** frente a los **6226,65** de la titulacion *College*.

In [ ]:

```
# Creamos un gráfico de barras
plt.figure(figsize = (8, 6))

# Añadiendo los valores
bar_mean_data = [below_college_mean['MonthlyIncome'].values[0],
            college_mean['MonthlyIncome'].values[0],
            bachelor_mean['MonthlyIncome'].values[0],
            master_mean['MonthlyIncome'].values[0],
            doctor_mean['MonthlyIncome'].values[0]]
education_cat = ['Below College', 'College', 'Bachelor', 'Master', 'Doctor']

# Creando las etiquetas
for i, value in enumerate(bar_mean_data):
    plt.text(i, value, str(value), ha='center', va='bottom', color='red')

# Linea promedio
plt.axhline(monthlyIncome_mean, color='blue', linestyle='--', label='Media')

# Agregando el valor linea promedio
plt.text(len(education_cat) + 0.8, monthlyIncome_mean, f'Promedio: {monthlyIncome_m

# Unificando los datos en un gráfico
plt.bar(education_cat, bar_mean_data)
plt.xlabel('Nivel Educativo')
plt.ylabel('Ingresos Mensaules')
plt.title('Promedio Ingresos Mensuales por Nivel Educativo', fontsize=18, fontweigh
plt.show()
```



4.4 FORMACIÓN Y DESARROLLO

Continuando con nuestro análisis descriptivo, exploraré los años que la empresa se ha comprometido a brindar capacitación a sus empleados en cada departamento. Se intentará identificar áreas potenciales para mejorar la capacitación y cuáles departamentos podrían beneficiarse con ello.

In [ ]:
```
# Para facilitar el trabajo crearé una tabla con las columnas 'MonthlyIncome' y 'Ed
df_timesTraining = df_padb[['Department', 'TrainingTimesLastYear']].copy()
df_timesTraining.head()
```

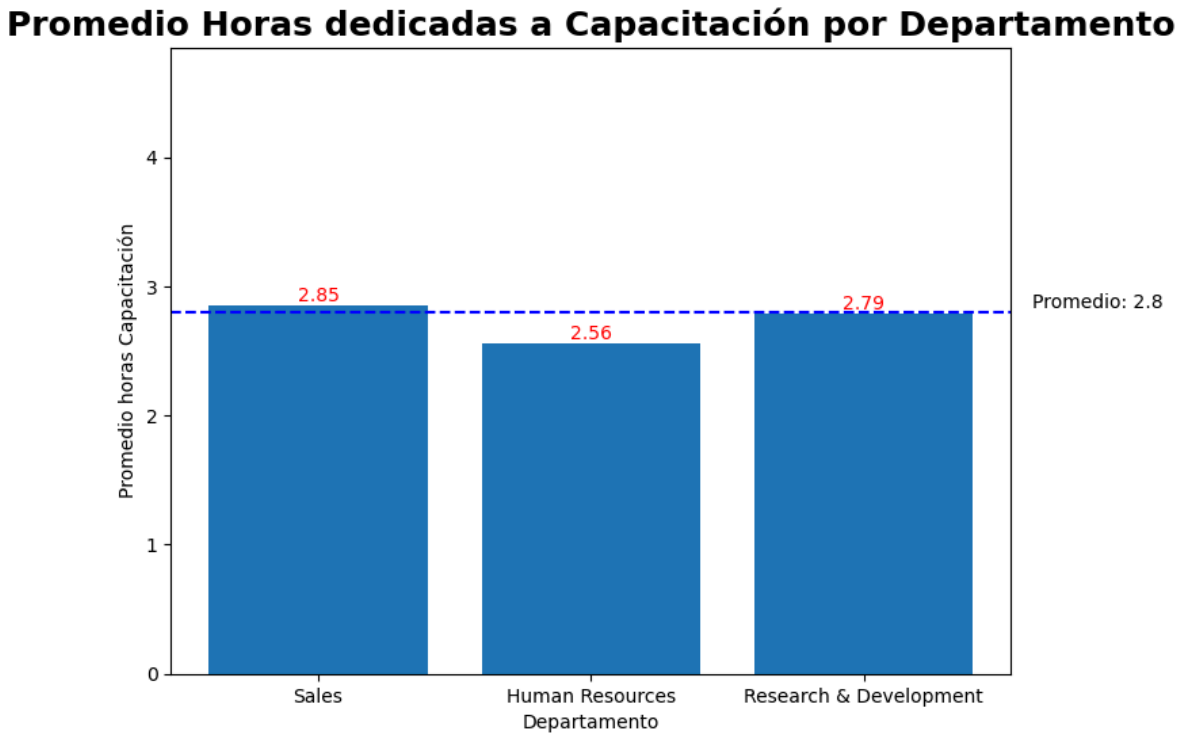
Out[ ]:

	Department	TrainingTimesLastYear
0	Sales	0
1	Research & Development	3
2	Research & Development	3
3	Research & Development	3
4	Research & Development	3

In [ ]:
```
timesTraining_average = df_timesTraining['TrainingTimesLastYear'].mean().round(2)

# Calculamos el promedio de horas dedicadas a Capacitación
sales_training_average = df_timesTraining[df_timesTraining['Department'] == 'Sales'
rrhh_training_average = df_timesTraining[df_timesTraining['Department'] == 'Human R
rd_training_average= df_timesTraining[df_timesTraining['Department'] == 'Research &

print(
    "El promedio de horas dedicadas a capacitación en Sales es: ", sales_training_a
    "El promedio de horas dedicadas a capacitación en Human Resources es: ", rrhh_t
    "El promedio de horas dedicadas a capacitación en Research & Development es: ",
    'TEl promedio de horas para la columna "TrainingTimeLastYear" es: ', timesTrain
```

```
El promedio de horas dedicadas a capacitación en Sales es:  [[2.85]]
 El promedio de horas dedicadas a capacitación en Human Resources es:  [[2.56]]
 El promedio de horas dedicadas a capacitación en Research & Development es:  [[2.
79]]
 TEl promedio de horas para la columna "TrainingTimeLastYear" es:  2.8
```

In [ ]:
```
# Creamos un gráfico de barras
plt.figure(figsize = (8, 6))

# Añadimos los variables
bar_training_data = [sales_training_average['TrainingTimesLastYear'].values[0],
            rrhh_training_average['TrainingTimesLastYear'].values[0],
            rd_training_average['TrainingTimesLastYear'].values[0]
        ]
department_cat = ['Sales', 'Human Resources', 'Research & Development']

# Sumamos las etiquetas
for i, value in enumerate(bar_training_data):
    plt.text(i, value, str(value), ha='center', va='bottom', color='red')

# Creamos la línea promedio
plt.axhline(timesTraining_average, color='blue', linestyle='--', label='Average Tra

# Añadimos el valor promedio a la línea
plt.text(len(department_cat) + 0.1, timesTraining_average, f'Promedio: {timesTraini

# Adjust y-axis limits to ensure the line is visible
plt.ylim(0, max(bar_training_data) + 2)

# Unificamos los datos y creamos el gráfico
plt.bar(department_cat, bar_training_data)
plt.xlabel('Departamento')
plt.ylabel('Promedio horas Capacitación')
plt.title('Promedio Horas dedicadas a Capacitación por Departamento', fontsize=18,
plt.show()
```

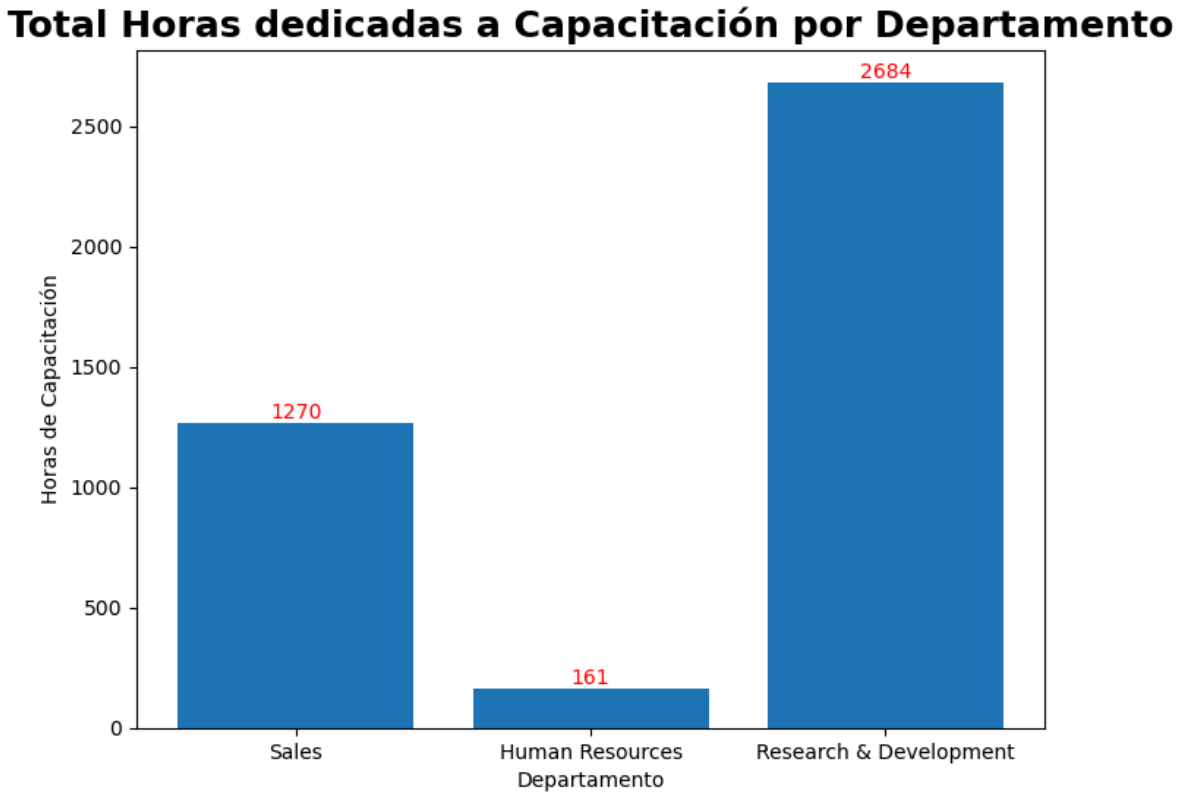


In [ ]:
```
# Calculamos el promedio de Años dedicados a Cacitación por Departamento
sales_training_sum = df_timesTraining[df_timesTraining['Department'] == 'Sales'].gr
rrhh_training_sum = df_timesTraining[df_timesTraining['Department'] == 'Human Resou
rd_training_sum= df_timesTraining[df_timesTraining['Department'] == 'Research & Dev
```

```
print(
    "El total de horas dedicadas a Capacitación en Sales es: ", sales_training_sum.
    "El total de horas dedicadas a Capacitación en Human Resources es: ", rrhh_trai
    "El total de horas dedicadas a Capacitación en Research & Development es: ", rd
    'El total de años dedicados a capacitación "TrainingTimeLastYear" es: ', timesl
```

```
El total de horas dedicadas a Capacitación en Sales es:  [[1270]]
 El total de horas dedicadas a Capacitación en Human Resources es:  [[161]]
 El total de horas dedicadas a Capacitación en Research & Development es:  [[268
4]]
 El total de años dedicados a capacitación "TrainingTimeLastYear" es:  2.8
```

In [ ]:
```
# Creamos un gráfico de barras
plt.figure(figsize = (8, 6))

# Adding the variables
bar_training_data = [sales_training_sum['TrainingTimesLastYear'].values[0],
            rrhh_training_sum['TrainingTimesLastYear'].values[0],
            rd_training_sum['TrainingTimesLastYear'].values[0]
        ]
department_cat = ['Sales', 'Human Resources', 'Research & Development']

# Añadimos las etiquetas
for i, value in enumerate(bar_training_data):
    plt.text(i, value, str(value), ha='center', va='bottom', color='red')

# Unificamos los valores para crear el gráfico
plt.bar(department_cat, bar_training_data)
plt.xlabel('Departamento')
plt.ylabel('Horas de Capacitación')
plt.title('Total Horas dedicadas a Capacitación por Departamento', fontsize=18, for
plt.show()
```



4.5 BALANCE ENTRE TRABAJO Y FAMILIA - ANÁLISIS DE HORAS EXTRAS

Actualmente, la evaluación del ***equilibrio entre trabajo y familia*** de los trabajadores es una de las evaluaciones más cruciales. Investiguemos si están trabajando demasiado y si son

capaces de compensar sus horas de trabajo con su tiempo libre. Recuerde que la mayoría de los *trabajadores valoran un equilibrio saludable entre trabajo y familia*, especialmente ante la creciente popularidad del empleo remoto</u>.

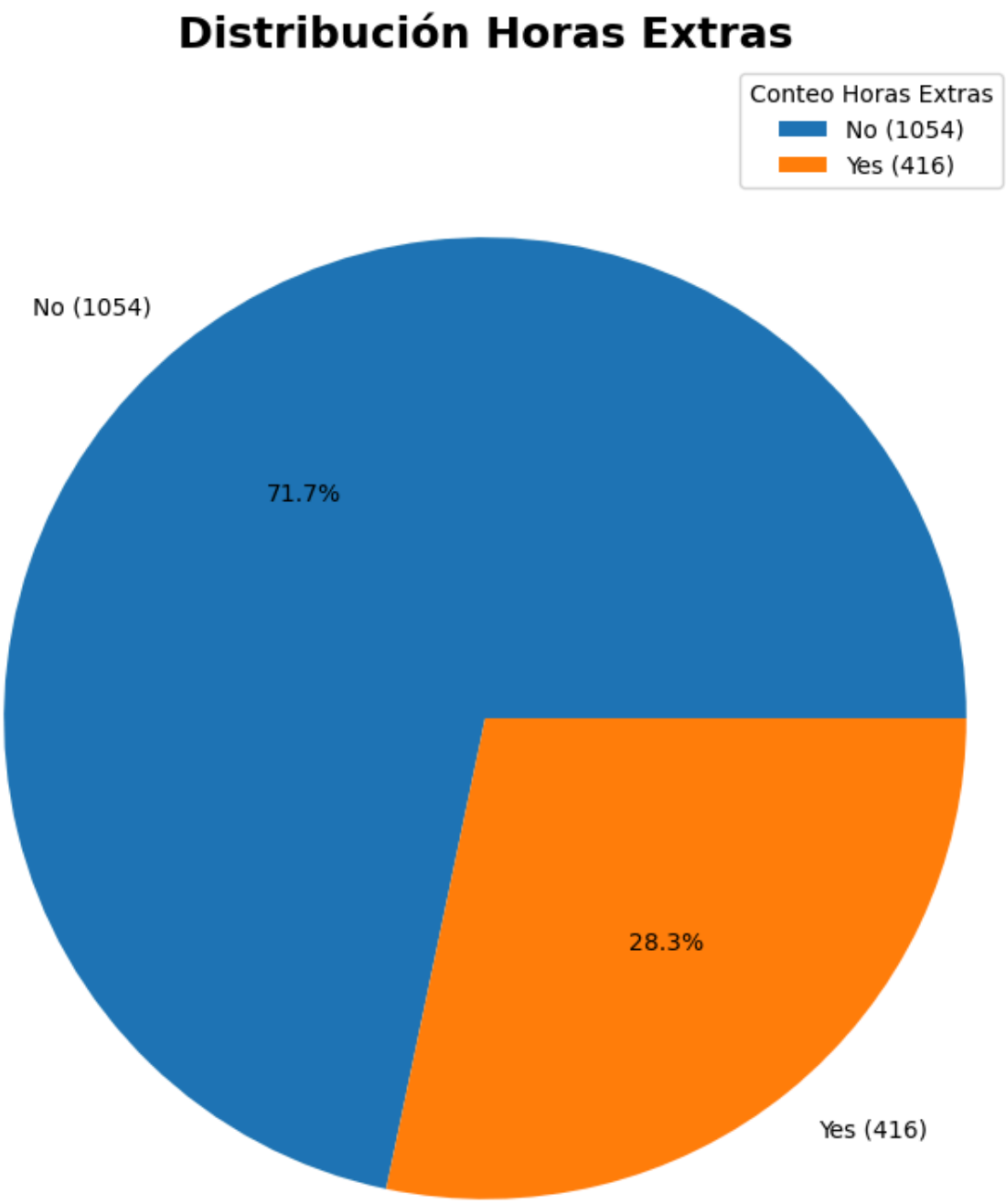
In [ ]:
```
df_wlb = df_padb[['OverTime', 'JobSatisfaction', 'WorkLifeBalance', 'DistanceFromHo
df_wlb.head()
```

Out[ ]:

	OverTime	JobSatisfaction	WorkLifeBalance	DistanceFromHome
0	Yes	Very High	Bad	1
1	No	Medium	Better	8
2	Yes	High	Better	2
3	Yes	High	Better	3
4	No	Medium	Better	2

In [ ]:
```
# Analizamos los datos de la columna OverTime
overtime_count = df_wlb['OverTime'].value_counts()
print(overtime_count)

# Las varables total y porcentaje para usar en los gráficos
total_overtime_count = overtime_count.sum()
overtime_percentage = (overtime_count / total_count) * 100

# Mediante una función creamos las etiquetas
labels = [f'{overtime} ({count})' for overtime, count in zip(overtime_count.index,

# Tamaño del gráfico
fig, ax = plt.subplots(figsize=(8, 10))

# Creamos el gráfico
plt.pie(overtime_count, labels=labels, autopct='%1.1f%%')
plt.legend(title='Conteo Horas Extras')
ax.set_title('Distribución Horas Extras', fontsize=18, fontweight='bold')
plt.axis('equal')
plt.show()
```

```
No     1054
Yes     416
Name: OverTime, dtype: int64
```

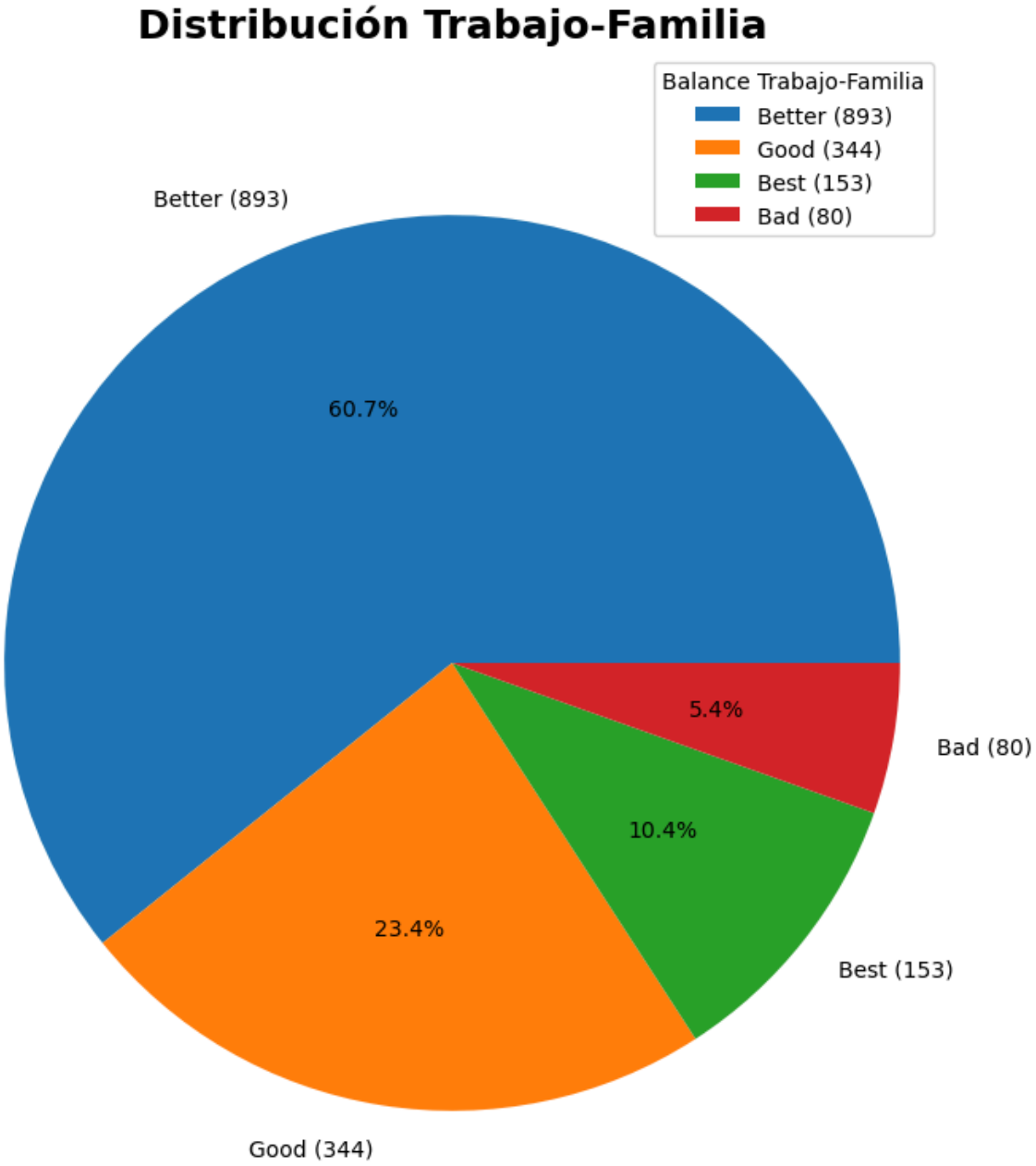


Luego de examinar de los datos de la columna '*OverTime*' tenemos que sólo el **28,3%** de nuestros empleados trabajan horas extras. no es posible determinar cómo el equilibrio entre trabajo y familia de un empleado se vea afectado por las horas extras</u>. Para obtener un resultado más concluyente, vamos a examinar los valores de la columna "*WorkLifeBalance*".

In [ ]:
```
# Analizamos los datos de la columna WorkLifeBalance
wlbalance_count = df_wlb['WorkLifeBalance'].value_counts()
print(wlbalance_count)

# Las varables total y porcentaje para usar en los gráficos
total_wlbalance_count = wlbalance_count.sum()
wlbalance_percentage = (wlbalance_count / total_count) * 100
```

```
# Mediante una función creamos las etiquetas
labels = [f'{wlbalance} ({count})' for wlbalance, count in zip(wlbalance_count.inde

# Tamaño del gráfico
fig, ax = plt.subplots(figsize=(8, 10))

# Creamos el gráfico
plt.pie(wlbalance_count, labels=labels, autopct='%1.1f%%')
plt.legend(title='Balance Trabajo-Familia')
ax.set_title('Distribución Trabajo-Familia', fontsize=18, fontweight='bold')
plt.axis('equal')
plt.show()
```

```
Better    893
Good      344
Best      153
Bad        80
Name: WorkLifeBalance, dtype: int64
```



En la columna 'WorkLifeBalance' se pudo encontrar información más pertinente sobre la apreciación que tienen los empleados por el equilibrio entre trabajo y familia. Según mi investigación, **5,4%** de los trabajadores piensan que no existe un buen equilibrio entre vida laboral y la familia. Se intentará determinar si existe otro factor que afecte su decisión. Vamos a restringir los datos para ver sólo aquellos cuyas "*JobSatisfaction*" y "*WorkLifeBalance*" son ambos **"Bajo"**. También exploraré si aplica para las '*OverTime*'.

In [ ]:
```
# Filtramos los datos
wlb_JsWlb_low = df_wlb[(df_wlb['WorkLifeBalance'] == 'Bad') & (df_wlb['JobSatisfact
wlb_JsWlb_good = df_wlb[(df_wlb['WorkLifeBalance'] == 'Good') & (df_wlb['JobSatisfa

# Conteo de los datos filtrados
countWlbJs_low = wlb_JsWlb_low.shape[0]
countWlbJs_good = wlb_JsWlb_good.shape[0]
print("El número total de empleados con Low Job Satisfaction, Bad Work-Life Balance
print("El número total de empleados con Good Job Satisfaction, Bad Work-Life Balanc
```

```
El número total de empleados con Low Job Satisfaction, Bad Work-Life Balance y Ove
r time Yes, es:  2
El número total de empleados con Good Job Satisfaction, Bad Work-Life Balance y Ov
er time Yes es:  17
```

In [ ]:
```
df_wlb['DistanceFromHome'].describe()
```

Out[ ]:
```
count    1470.000000
mean        9.192517
std         8.106864
min         1.000000
25%         2.000000
50%         7.000000
75%        14.000000
max        29.000000
Name: DistanceFromHome, dtype: float64
```

Después de analizar los datos, se descubrió que los empleados están satisfechos con el equilibrio trabajo-familia que proporciona la organización. *Es crucial enfatizar e intentar aplicar una política para reducir el **28,3%** de los trabajadores que realizan horas extras. Además, descubrí que muy pocos trabajadores realmente tienen problemas para encontrar un equilibrio entre trabajo y familia, por lo que se aconseja asistir a los empleados con mayores dificultades.*

4.6 ANÁLISIS DE RENUNCIAS

*Es hora de trabajar con una de las métricas más importantes para nuestra empresa: **la tasa de renuncias de la empresa durante 2021**. Conocer la tasa de renuncias en nuestra organización nos ayudará a saber si los empleados que abandonan la empresa son aquellos que rinden bien o aquellos que rinden poco laboralmente. Tal vez una alta tasa de renuncias signifique que estamos perdiendo empleados de bajos resultados; esta métrica nos ayudará a mantener buenos talentos, además de conocer un poco las causas por las cuales se producen las renuncias en nuestra empresa.*

In [ ]:
```
# Creamos una tabla con las columnas a utilizar en el análisis
df_attrition = df_padb[['Attrition', 'EnvironmentSatisfaction', 'JobInvolvement',
df_attrition.head()
```

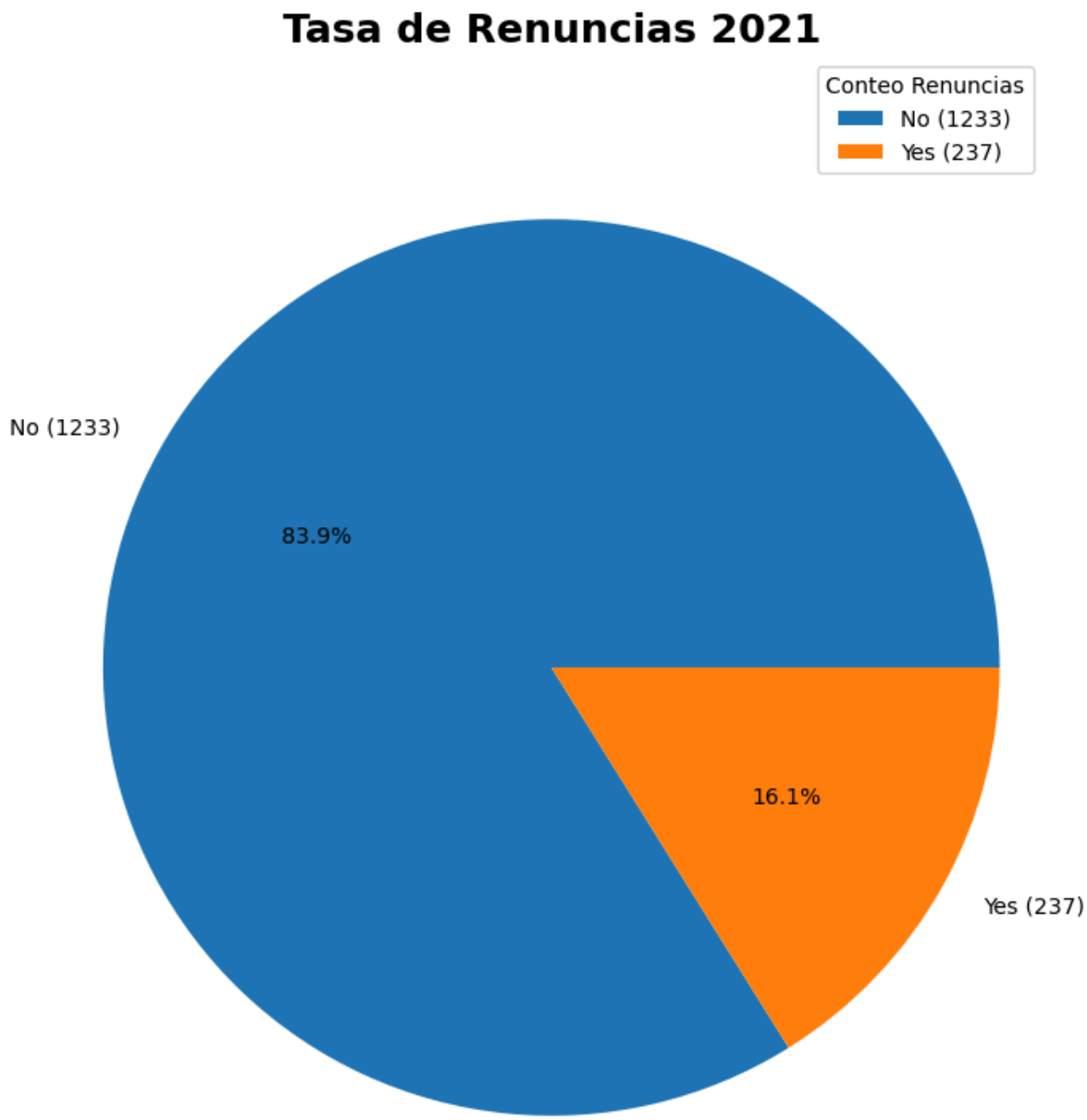
Out[ ]:

	Attrition	EnvironmentSatisfaction	JobInvolvement	JobSatisfaction	MonthlyIncome	OverTime
0	Yes	Medium	High	Very High	5993	Yes
1	No	High	Medium	Medium	5130	No
2	Yes	Very High	Medium	High	2090	Yes
3	No	Very High	High	High	2909	Yes
4	No	Low	High	Medium	3468	No

Con la nueva tabla, ahora podemos calcular la tasa de renuncias de nuestra empresa.

In [ ]:
```
# Analizamos los datos de la columna Attrition
attrition_count = df_attrition['Attrition'].value_counts()
print('El conteo de renuncias es: ', '\n', attrition_count)

# Las varables total y porcentaje para usar en los gráficos
total_attrition_count = attrition_count.sum()
attrition_percentage = (attrition_count / total_count) * 100
print('El porcentaje de renuncias es: ', '\n', round(attrition_percentage, 1))

# Mediante una función creamos las etiquetas
labels = [f'{attrition} ({count})' for attrition, count in zip(attrition_count.inde

# Tamaño del gráfico
fig, ax = plt.subplots(figsize=(8, 10))

# Creando el gráfico
plt.pie(attrition_count, labels=labels, autopct='%1.1f%%')
plt.legend(title='Conteo Renuncias')
ax.set_title('Tasa de Renuncias 2021', fontsize=18, fontweight='bold')
plt.axis('equal')
plt.show()
```

```
El conteo de renuncias es:
 No     1233
Yes     237
Name: Attrition, dtype: int64
El porcentaje de renuncias es:
 No     83.9
Yes    16.1
Name: Attrition, dtype: float64
```



*El análisis nos muestra una tasa de renuncias de un **16,1% en 2021**, por lo tanto, 237 empleados en total abandonaron la organización. Intentaré averiguar si fueron impulsados por alguna razón particular o si la renuncia es producto de un cambio que ellos mismos deseaban.*

In [ ]:
```
# Filtrando los datos
attrition_worstvalues = df_attrition[(df_attrition['EnvironmentSatisfaction'] == 'L
attrition_mediumvalues = df_attrition[(df_attrition['EnvironmentSatisfaction'] == '

# Contando los datos filtrados
count_attrition_wv = attrition_worstvalues.shape[0]
count_attrition_mv = attrition_mediumvalues.shape[0]
print("El número de empleados que renunciaron con Job Satisfaction y Environment Sa
print("El número de empleados que renunciaron con Job Satisfaction 'Low' y Environm
```

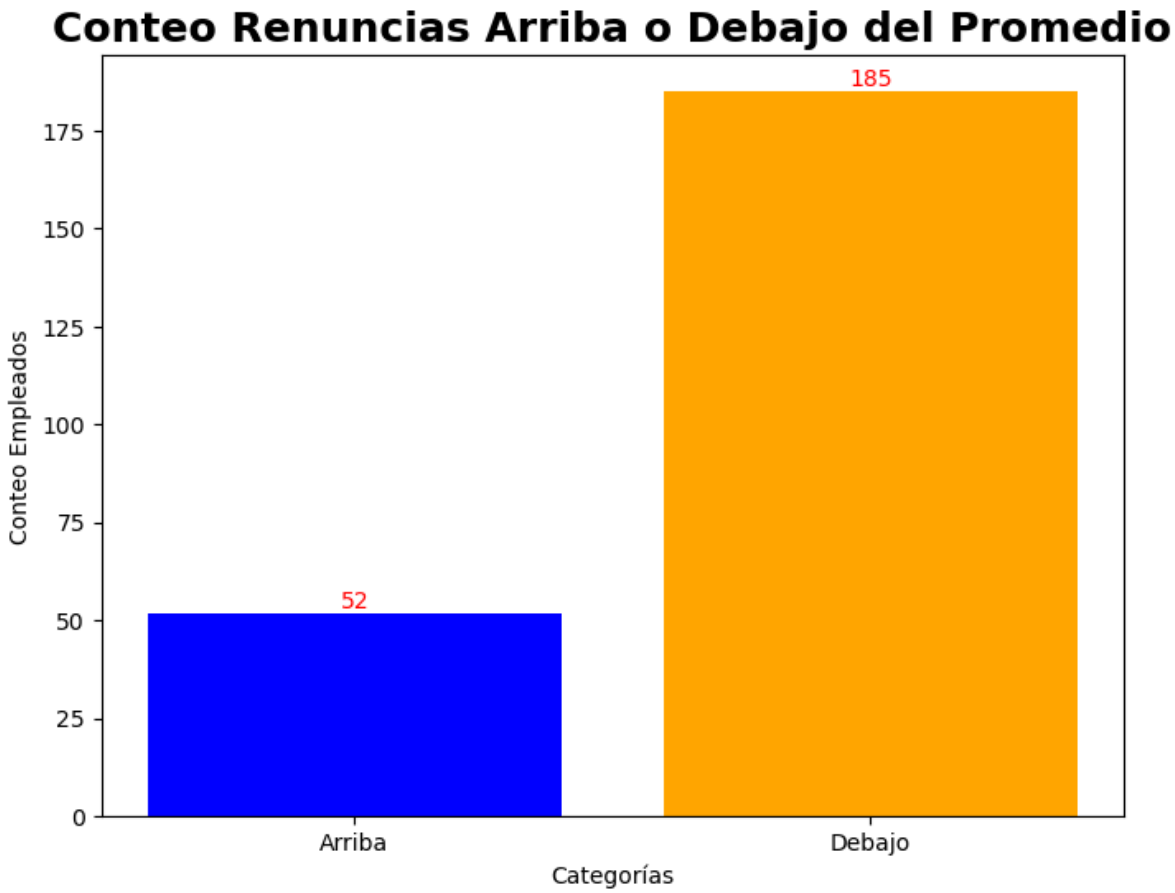
```
El número de empleados que renunciaron con Job Satisfaction y Environment Satisfac
tion 'Low', y working Over time es:  15
El número de empleados que renunciaron con Job Satisfaction 'Low' y Environment Sa
tisfaction 'Medium', y working Over time es:  16
```

Vamos a averiguar si la elección del empleado de dejar la empresa fue influenciada por su ***ingreso mensual***.

In [ ]:
```
# Calculando el promedio de Ingresos mensuales
attrition_mincome_mean = df_attrition['MonthlyIncome'].mean().round(2)
print(attrition_mincome_mean)

# Filtrando los datos entre aquellos que se encuantran por debajo o arriba del prom
attrition_over_mean_mi = df_attrition[(df_attrition['MonthlyIncome'] > attrition_mi
attrition_under_mean_mi = df_attrition[(df_attrition['MonthlyIncome'] < attrition_m

count_attrition_over_mean = attrition_over_mean_mi.shape[0]
count_attrition_under_mean = attrition_under_mean_mi.shape[0]

print('El número de empleados que renunciaron y estaban por arriba del promedio, es
print('El número de empleados que renunciaron y estaban por debajo del promedio, es
```

```
6502.93
El número de empleados que renunciaron y estaban por arriba del promedio, es:  52
El número de empleados que renunciaron y estaban por debajo del promedio, es:  185
```

In [ ]:
```
# Creamos un gráfico de barras
plt.figure(figsize = (8, 6))

# Añadimos las variables
attrition_data = [count_attrition_over_mean, count_attrition_under_mean]
attrition_cat = ['Arriba', 'Debajo']

# Creamos las etiquetas
for i, value in enumerate(attrition_data):
    plt.text(i, value, str(value), ha='center', va='bottom', color='red')

# Unificamos los datos en el gráfico
plt.bar(attrition_cat, attrition_data, color=['blue', 'orange'])
plt.xlabel('Categorías')
plt.ylabel('Conteo Empleados')
plt.title('Conteo Renuncias Arriba o Debajo del Promedio', fontsize=18, fontweight=
plt.show()
```



Luego del análisis se concluye que 52 de los 237 empleados que salieron de la empresa tenían ingresos mensuales que estaban por encima de la media, mientras que 185 de los trabajadores que renunciaron, sus ingresos eran inferiores a la media.

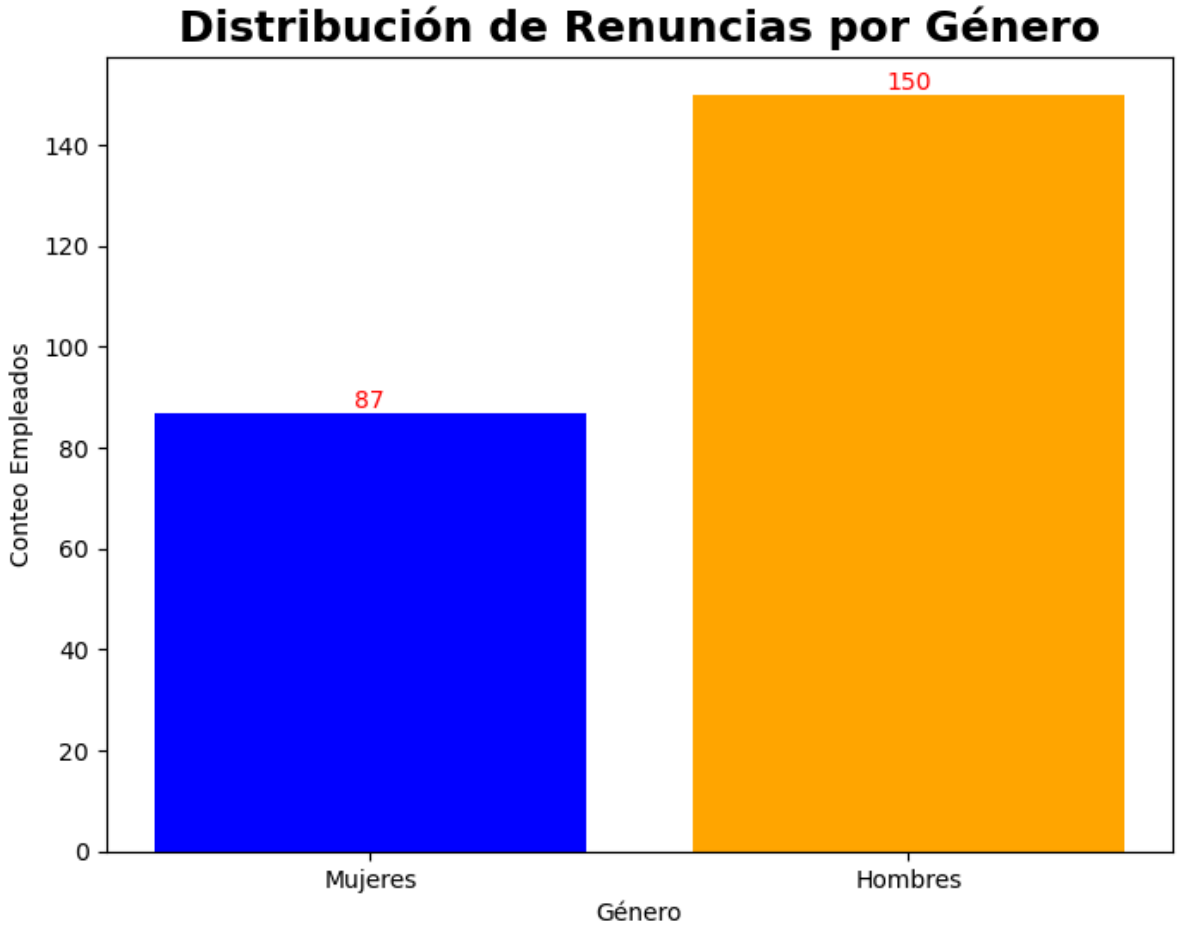
Examinemos la distribución de las renuncias por ***Género*** *y* ***Distribución de la edad****.*

In [ ]:
```
# Analizamos las renuncias de acuerdo al género
attrition_female = df_attrition[(df_attrition['Gender'] == 'Female') & (df_attritic
attrition_male = df_attrition[(df_attrition['Gender'] == 'Male') & (df_attrition['A

# Contamos la distribución de las renuncias por género
count_attrition_female = attrition_female.shape[0]
count_attrition_male = attrition_male.shape[0]

print('Número de mujeres que renunciaron a la empresa: ', count_attrition_female)
print('Número de hombres que renunciaron a la empresa: ', count_attrition_male)
```

```
Número de mujeres que renunciaron a la empresa:  87
Número de hombres que renunciaron a la empresa:  150
 87
Número de hombres que renunciaron a la empresa:  150
```

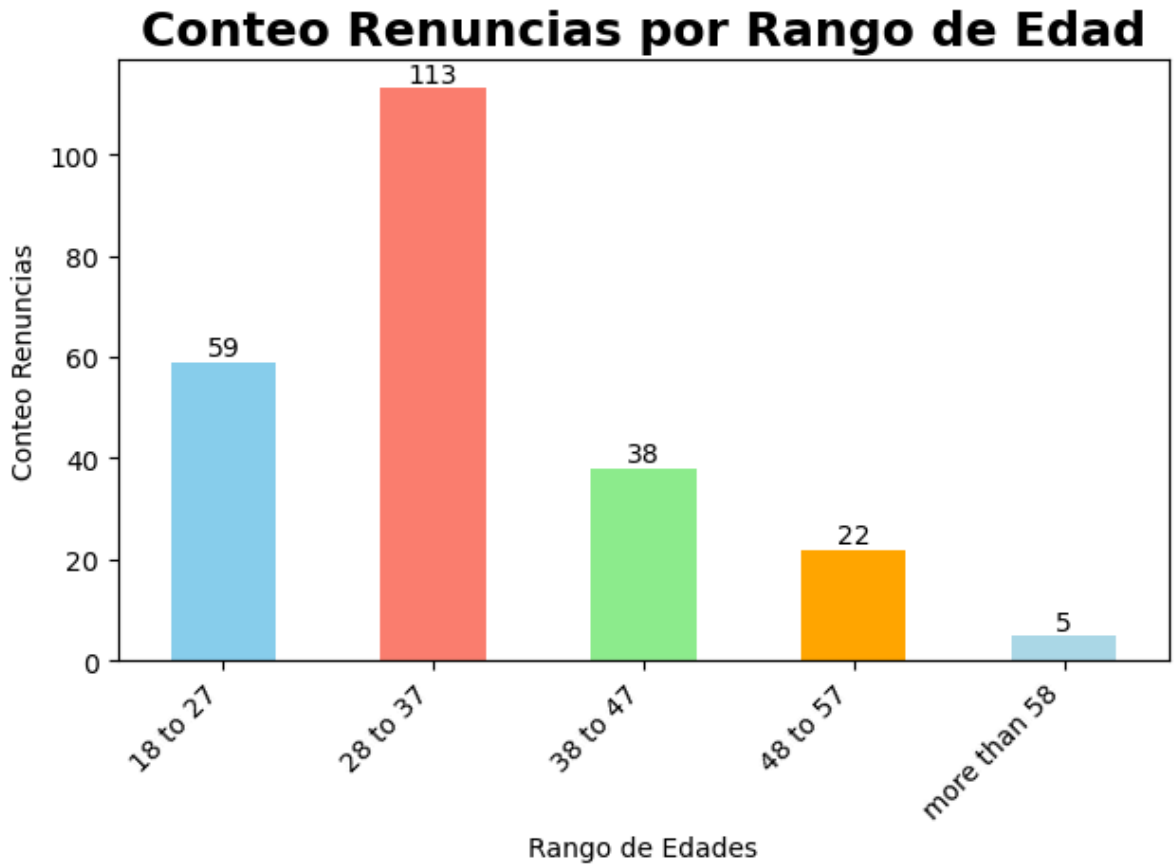
In [ ]:
```
# Creamos un gráfico de barras
plt.figure(figsize = (8, 6))

# Añadimos las variables
gender_attrition_data = [count_attrition_female, count_attrition_male]
attrition_gender_cat = ['Mujeres', 'Hombres']

# Creamos las etiquetas
for i, value in enumerate(gender_attrition_data):
    plt.text(i, value, str(value), ha='center', va='bottom', color='red')
```

```
# Unificamos los datos en el gráfico
plt.bar(attrition_gender_cat, gender_attrition_data, color=['blue', 'orange'])
plt.xlabel('Género')
plt.ylabel('Conteo Empleados')
plt.title('Distribución de Renuncias por Género', fontsize=18, fontweight='bold')
plt.show()
```



In [ ]:

```
# Filtramos los datos por el valor 'Yes' en las renuncias
age_range_attrition_filter = df_attrition[(df_attrition['Attrition'] == 'Yes')]

colors = ['skyblue', 'salmon', 'lightgreen', 'orange', 'lightblue']

# Agrupamos los valores por la columna 'AgeRange'
attrition_by_age = age_range_attrition_filter.groupby('AgeRange').size()

# Creamos un gráfico de barras
ax = attrition_by_age.plot(kind='bar', color=colors)

# Organizamos las etiquetas
ax.set_xticks(range(len(attrition_by_age)))
ax.set_xticklabels(attrition_by_age.index, rotation=45, ha='right')

# Añadimos los valores a las barras
for i, v in enumerate(attrition_by_age):
    ax.text(i, v + 0.1, str(v), ha='center', va='bottom')

# Título y etiquetas
plt.xlabel('Rango de Edades')
plt.ylabel('Conteo Renuncias')
plt.title('Conteo Renuncias por Rango de Edad', fontsize=18, fontweight='bold')

plt.tight_layout()
plt.show()
```

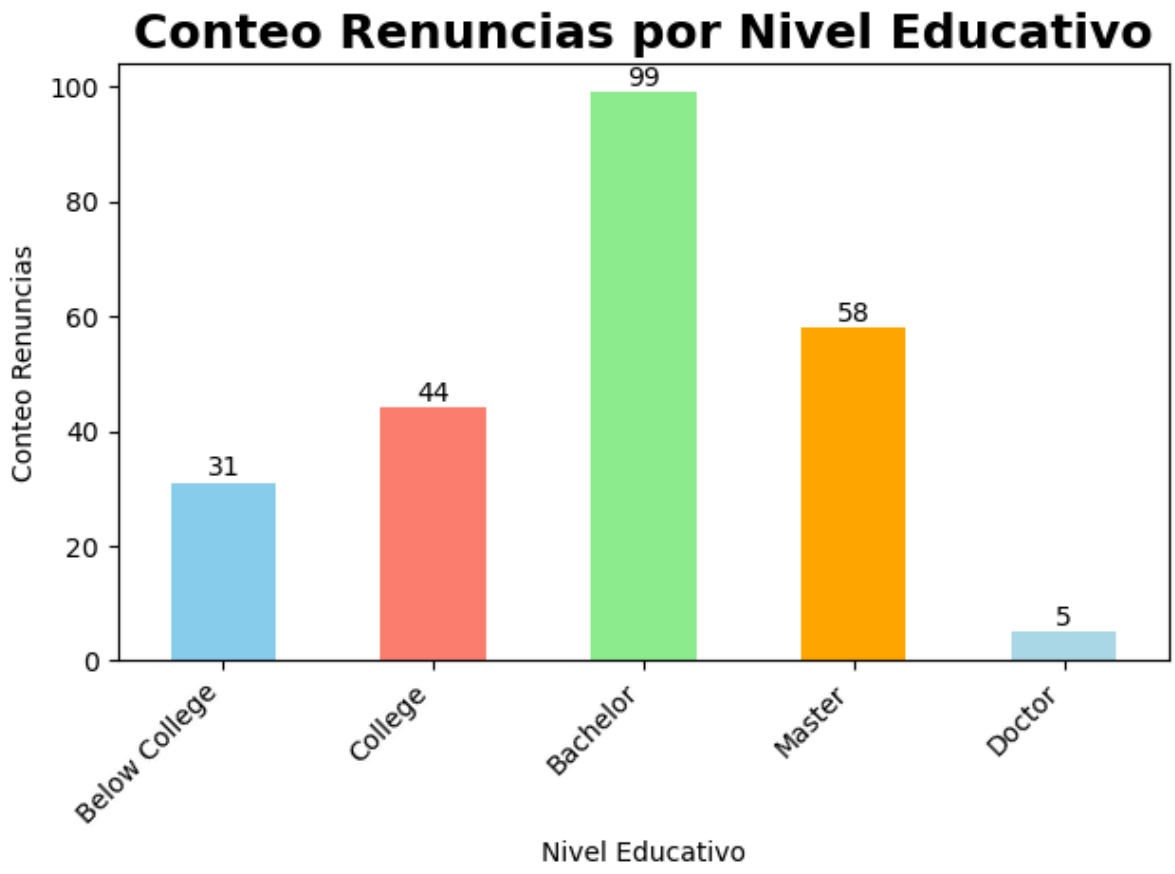


*Cuando desglosamos las renuncias por rango de edad, vemos que en el rango de edades de18 a 47 años es donde la empresa pierde el mayor número de empleados, con un totla de **210 retiros**. El rango de edad de 28 a 37 años es donde la empresa registra el mayor número de renuncias.*

*Por lo analizado con el rango de edad, vamos a investigar para conocer si, a partir de esas dimisiones, la empresa está **perdiendo talento**. Analizaré las renuncias utilizando el nivel de educación del los empleados.*

In [ ]:
```
# Filtramos los datos por el valor 'Yes' en las renuncias
education_attrition_filter = df_attrition[(df_attrition['Attrition'] == 'Yes')]

colors = ['skyblue', 'salmon', 'lightgreen', 'orange', 'lightblue']

# Agrupamos los valores por la columna 'AgeRange'
attrition_by_education = education_attrition_filter.groupby('Education').size()

# Definimos el orden de los valores de Educación
desired_order = ['Below College', 'College', 'Bachelor', 'Master', 'Doctor']

# Ordenamos el orden de los valores
attrition_by_education = attrition_by_education.reindex(desired_order)

# Creamos el gráfico
ax = attrition_by_education.plot(kind='bar', color=colors)

# Organizamos las etiquetas
ax.set_xticks(range(len(attrition_by_age)))
ax.set_xticklabels(attrition_by_education.index, rotation=45, ha='right')

# Añadimos los valores a las barras
for i, v in enumerate(attrition_by_education):
```

```
    ax.text(i, v + 0.1, str(v), ha='center', va='bottom')

# Título y etiquetas
plt.xlabel('Nivel Educativo')
plt.ylabel('Conteo Renuncias')
plt.title('Conteo Renuncias por Nivel Educativo', fontsize=18, fontweight='bold')

plt.tight_layout()
plt.show()
```



*Al desglosar las renuncias por nivel de educación, pude encuentrar que los empleados con títulos de "Bachelor" y "Master", seguidos por "College", son los empleados más propensos a renunciar, registrando un total de **201** renuncias. Esta información plantea la posibilidad de que la organización está perdiendo talento como resultado de las dimisiones.*

4.7 MODELOS CON MACHINE LEARNING

Intentaré crear algunos Modelos de Machine Learning usando los datos de la empresa.

*Para este tipo de análisis, necesitamos usar **Numpy**.*

4.7.a Correlación entre Salario y Experiencia - Regresión Lineal

*Usaré los datos de las columnas "MonthlyIncome" y "TotalWorkingYears" para este estudio. El **Modelo de Regresión Lineal** será la herramienta a utilizar. El propósito es saber si la remuneración de los empleados es proporcional a sus años de experiencia.*

Utilizaré dos tipos de cálculos, uno usando Numpy y el otro de StatsModels, que incluyen información como coeficientes, errores estándar, valores t, valores p y R-squared, que son

cruciales para interpretar los resultados del modelo de regresión lineal. Para visualizar los datos, usaré un plano de dispersión, o scatter-plot

In [ ]:
```
# Creamos la base a trabajar
df_salary = df_padb[['MonthlyIncome', 'TotalWorkingYears']].copy()
df_salary.describe()
```

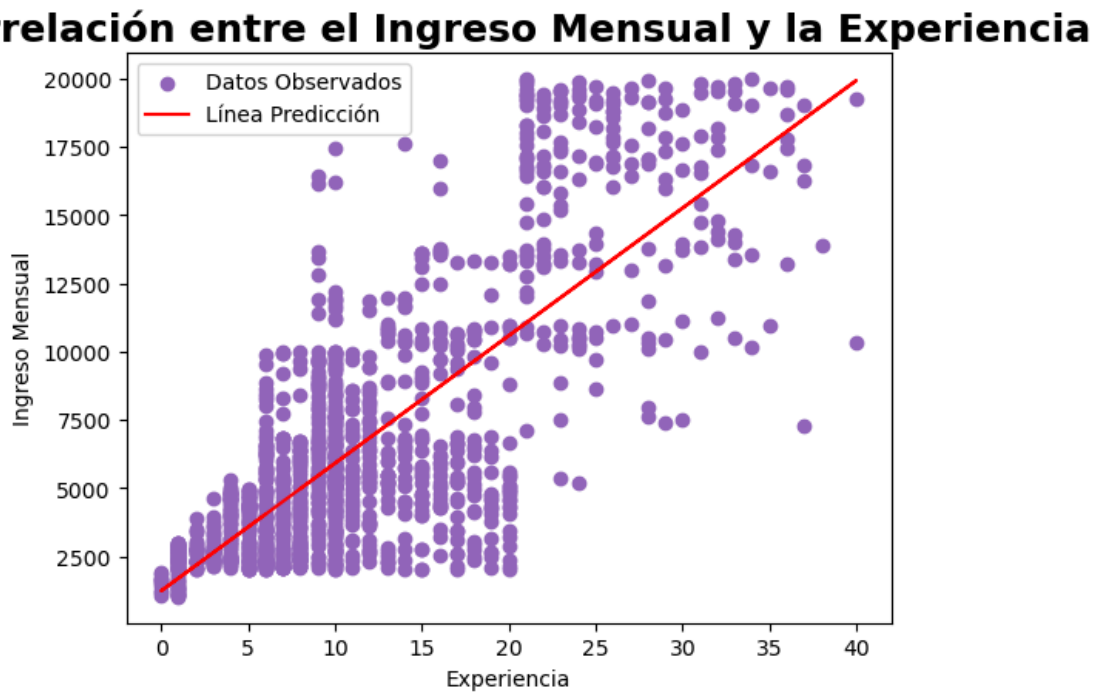
Out[ ]:

	MonthlyIncome	TotalWorkingYears
count	1470.000000	1470.000000
mean	6502.931293	11.279592
std	4707.956783	7.780782
min	1009.000000	0.000000
25%	2911.000000	6.000000
50%	4919.000000	10.000000
75%	8379.000000	15.000000
max	19999.000000	40.000000

4.7.a.1 Regresión Lineal - Numpy

In [ ]:
```
# Definimos la ubicación de los datos en los axis
x = df_salary['TotalWorkingYears'];
y = df_salary['MonthlyIncome'];

# Creamos el gráfico
plt.scatter(x = x, y = y, color='#9467bd')

# Obtenemos el m (slope) y b(intercept) de nuestra línea de regresión
m, b = np.polyfit(x, y, 1)
lreg = np.corrcoef(x, y)
# Graficamos la línea de regresión
plt.plot(x, m*x+b, color='red')

# Adding labels and title
plt.xlabel('Experiencia')
plt.ylabel('Ingreso Mensual')
plt.legend(['Datos Observados', 'Línea Predicción'])
plt.title('Correlación entre el Ingreso Mensual y la Experiencia', fontsize=18, for

# Imprimimos el valor de la línea de regresión
print(lreg)
```

```
[[1.         0.77289325]
 [0.77289325 1.        ]]
```



*Nuestro coeficiente de correlación es aproximadamente **0.77** indica una relación lineal positiva, relativamente fuerte entre los datos de las columnas TotalWorkingYears y MonthlyIncome.*

Algunos salarios deben modificarse para acercarlos de la línea de regresión así la empresa podrá experimentar una mejor correlación entre sueldo y experiencia. De tal manera, la empresa tendrá opciones de sueldo más atractivas.

Es importante indicar que no estamos incorporando a nuestro análisis cualquier otro incentivo financiero que la empresa otorga a sus empleados.

4.7.a.2 Regresión Lineal - Statsmodels

Voy a importar la biblioteca de statmodels para trabajar con la otra fórmula de Regresión Lineal.

In [ ]:
```
import statsmodels.formula.api as smf
model = smf.ols('MonthlyIncome ~ TotalWorkingYears', data = df_salary).fit()
model.summary()
```

Out[ ]:

OLS Regression Results

Dep. Variable:	MonthlyIncome	**R-squared:**	0.597
Model:	OLS	**Adj. R-squared:**	0.597
Method:	Least Squares	**F-statistic:**	2178.
Date:	Sat, 30 Mar 2024	**Prob (F-statistic):**	2.73e-292
Time:	00:16:11	**Log-Likelihood:**	-13848.
No. Observations:	1470	**AIC:**	2.770e+04
Df Residuals:	1468	**BIC:**	2.771e+04
Df Model:	1		
Covariance Type:	nonrobust		

	coef	std err	t	P>|t|	[0.025	0.975]
Intercept	1227.9353	137.299	8.944	0.000	958.612	1497.259
TotalWorkingYears	467.6584	10.021	46.669	0.000	448.002	487.315

Omnibus:	47.473	**Durbin-Watson:**	1.993
Prob(Omnibus):	0.000	**Jarque-Bera (JB):**	79.304
Skew:	0.269	**Prob(JB):**	6.02e-18
Kurtosis:	4.003	**Cond. No.**	24.2

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

Ahora hemos creado el modelo que está realizando un análisis de regresión lineal para explorar la relación entre la variable del predictor TotalWorkingYears y el MonthlyIncome.

Utilizemos el modelo para crear algunas predicciones.

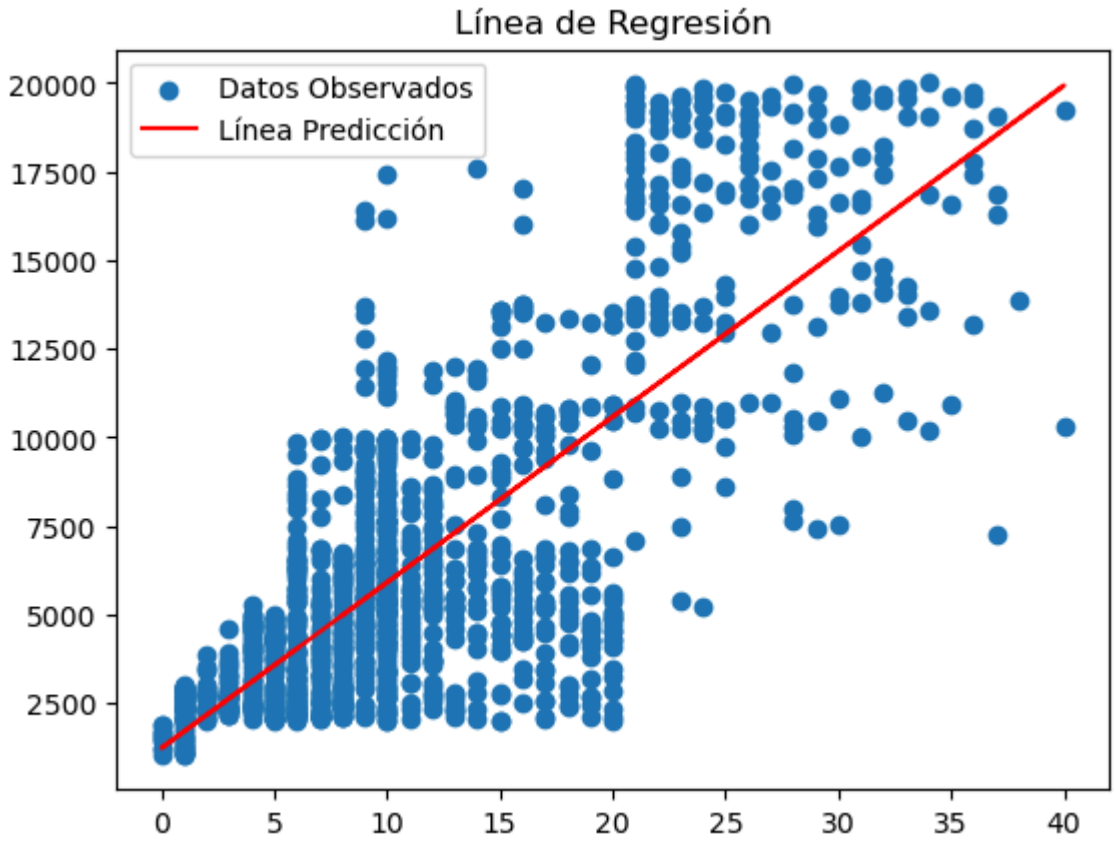
In [ ]:
```
# Utilizamos otro modelo para crear predicciones
# Intentamos predecir los salarios para cada año de experiencia y lo guardamos en u
pred1 = model.predict(pd.DataFrame(df_salary['TotalWorkingYears']))
print(pred1)
```

```
0       4969.202583
1       5904.519406
2       4501.544171
3       4969.202583
4       4033.885759
           ...
1465    9178.128289
1466    5436.860994
1467    4033.885759
1468    9178.128289
1469    4033.885759
Length: 1470, dtype: float64
```

*Con el modelo de regresión lineal (model), predicimos los salarios basados en los años de experiencia. Los valores se almacenan ahora en una variable llamada **pred1**.*

Examinemos la apariencia del modelo en una trama de dispersión (scatter-plot).

In [ ]:
```
# Línea de Regresión
# Utilizamos los valores de X e Y, los visualizamos mediante un scatter-plot
plt.scatter(x, y)
plt.plot(x, pred1, 'r')
# Let's add a legend to our plot
plt.legend(['Datos Observados', 'Línea Predicción'])
plt.title('Línea de Regresión')
plt.show()
```



El resultado del modelo de predicción es idéntico al producido con Numpy.

Examinemos el cálculo de errores.

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

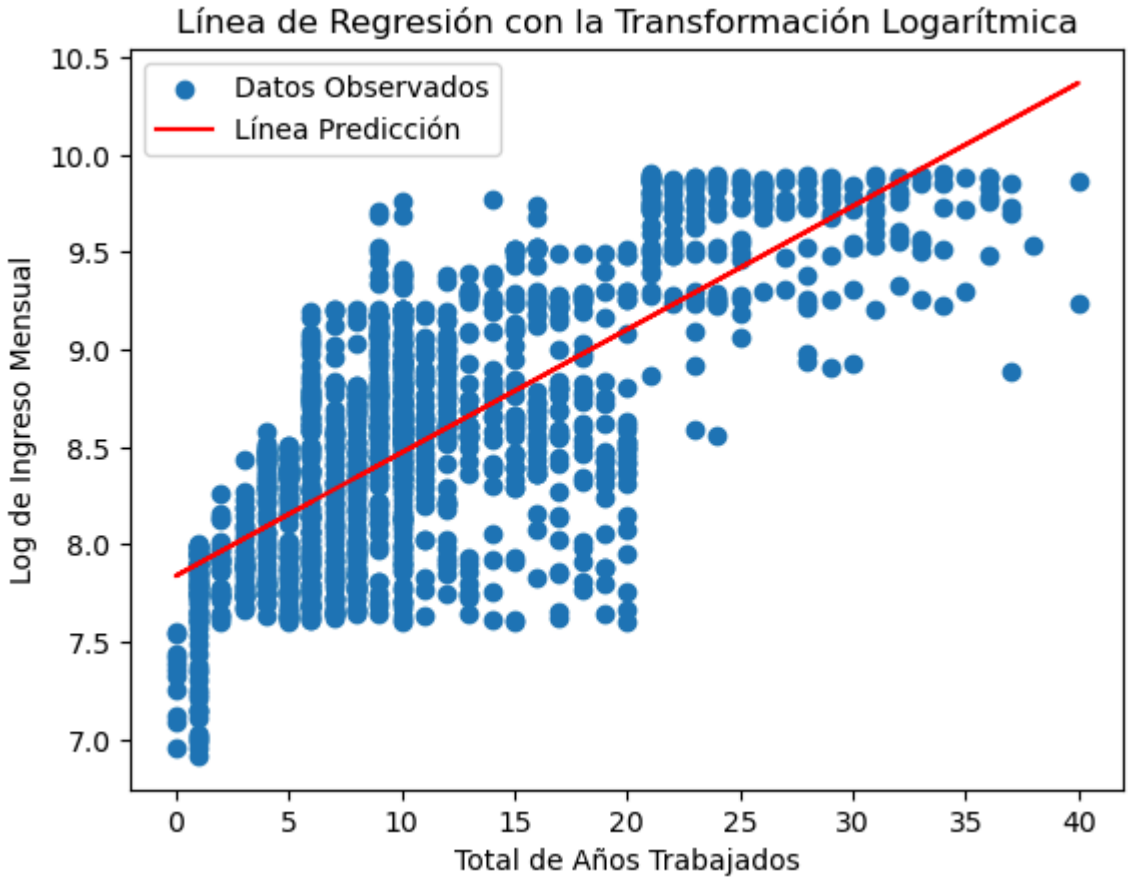
```
2986.3521316844103
```

*Como medida de la capacidad de predicción del modelo de regresión lineal, calculé el error cuadrado medio de raíz (RMSE). El modelo calcula los residuos, los cuadra para obtener el error cuadrado promedio (MSE), y luego toma la raíz cuadrada para obtener la RMSE. Un número RMSE más bajo indica un mayor rendimiento, y da información sobre cuán bien el modelo coincide con los datos observados. Con un valor de **2986.35**, esta es la magnitud media de los errores entre los valores predecibles y los valores reales. El modelo está dando buenos resultados en términos de predicción.*

4.7.a.3 Regresión lineal - Transformación Logarítmica (Log)

Convertimos nuestros datos usando logaritmos.

In [ ]:
```
# Datos Transformados
# Transformación Logarítmica
plt.scatter(x = x, y = np.log(y))
np.corrcoef(x, np.log(y))
model2 = smf.ols('np.log(MonthlyIncome) ~ TotalWorkingYears', data=df_salary).fit()
# Añadimos la línea de regresión
plt.plot(x, model2.predict(df_salary), color='red')

# Título y etiquetas
plt.xlabel('Total de Años Trabajados')
plt.ylabel('Log de Ingreso Mensual')
plt.legend(['Datos Observados', 'Línea Predicción'])
plt.title('Línea de Regresión con la Transformación Logarítmica')

plt.show()
```



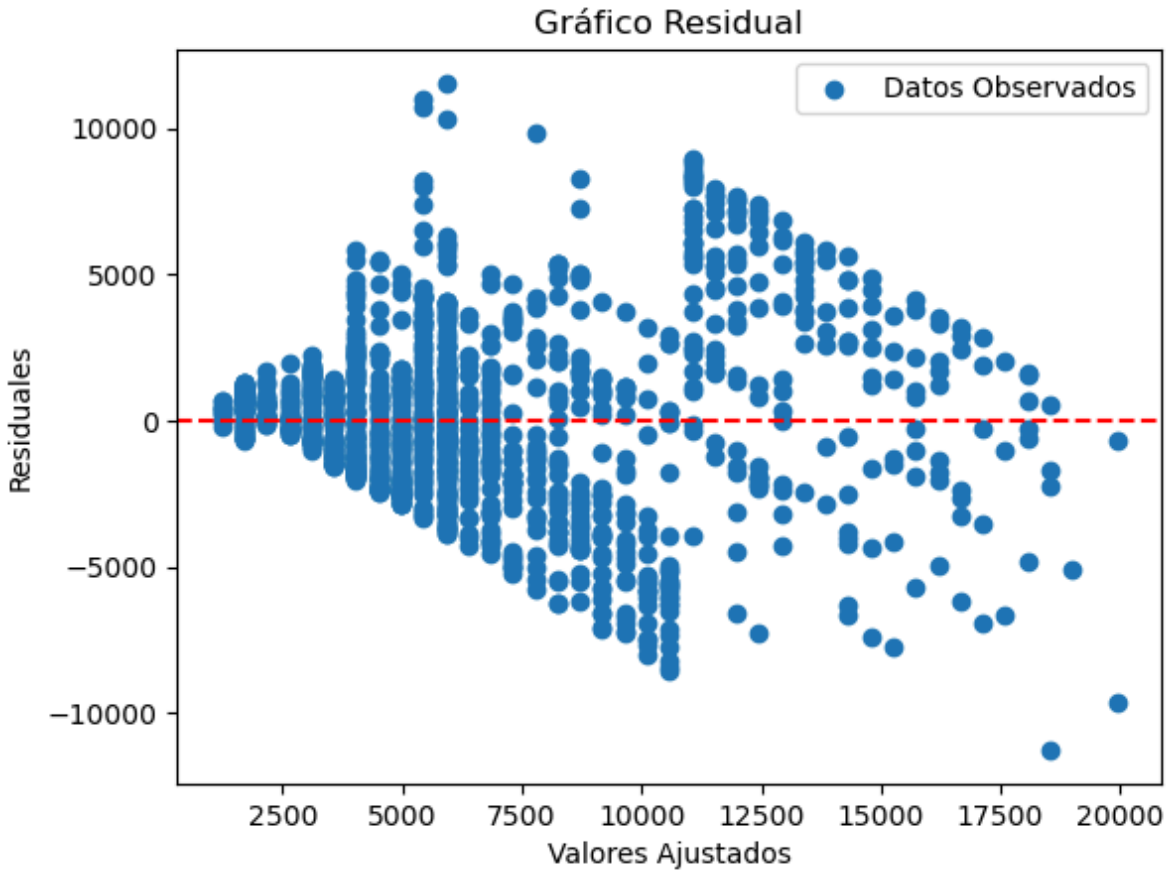
Después de dar a Salario una transformación logarítmica, podemos ver cómo 'Experiencia' y 'Salario' se relacionan entre sí. El coeficiente de correlación entre las dos variables está ahora disponible, y se proporciona un modelo de regresión lineal para examinar aún más la conexión entre Experiencia y el logaritmo salarial.

Se recomienda comparar los datos originales para ver si la relación está mejor representada en una escala logarítmica. Podemos interpretar los coeficientes del modelo de regresión lineal para comprender la relación entre 'TotalWorkingYears' y el valor esperado de 'MonthlyIncome' en la escala logarítmica.

4.7.a.4 Regresión lineal - Comparación

Usando un análisis residual, voy a contrastar los datos transformados por logaritmo con los datos originales.

In [ ]:
```
# Cálculo residual
residuals = model.resid

# Visualizamos el cálculo residual
plt.scatter(model.fittedvalues, residuals)
plt.xlabel('Valores Ajustados')
plt.ylabel('Residuales')
plt.title('Gráfico Residual')
plt.legend(['Datos Observados', 'Línea Predicción'])
# Añadmos la línea horizontal en y=0
plt.axhline(y=0, color='red', linestyle='--')
plt.show()
```



El gráfico de scatter-plot nos muestra los valores residuales contra los valores predecibles.

In [ ]:
```
# Modelo Original
print(model.summary())

# Modelo de Transformación Logarítmica
print(model2.summary())
```

OLS Regression Results

Dep. Variable:	MonthlyIncome	R-squared:	0.597
Model:	OLS	Adj. R-squared:	0.597
Method:	Least Squares	F-statistic:	2178.
Date:	Sat, 30 Mar 2024	Prob (F-statistic):	2.73e-292
Time:	00:16:11	Log-Likelihood:	-13848.
No. Observations:	1470	AIC:	2.770e+04
Df Residuals:	1468	BIC:	2.771e+04
Df Model:	1		
Covariance Type:	nonrobust		

	coef	std err	t	P>|t|	[0.025	0.975]
Intercept	1227.9353	137.299	8.944	0.000	958.612	1497.259
TotalWorkingYears	467.6584	10.021	46.669	0.000	448.002	487.315

Omnibus:	47.473	Durbin-Watson:	1.993
Prob(Omnibus):	0.000	Jarque-Bera (JB):	79.304
Skew:	0.269	Prob(JB):	6.02e-18
Kurtosis:	4.003	Cond. No.	24.2

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

OLS Regression Results

Dep. Variable:	np.log(MonthlyIncome)	R-squared:	0.549
Model:	OLS	Adj. R-squared:	0.548
Method:	Least Squares	F-statistic:	1784.
Date:	Sat, 30 Mar 2024	Prob (F-statistic):	9.04e-256
Time:	00:16:11	Log-Likelihood:	-899.93
No. Observations:	1470	AIC:	1804.
Df Residuals:	1468	BIC:	1814.
Df Model:	1		
Covariance Type:	nonrobust		

	coef	std err	t	P>|t|	[0.025	0.975]
Intercept	7.8391	0.021	382.037	0.000	7.799	7.879
TotalWorkingYears	0.0632	0.001	42.232	0.000	0.060	0.066

Omnibus:	10.743	Durbin-Watson:	1.981
Prob(Omnibus):	0.005	Jarque-Bera (JB):	10.779
Skew:	-0.195	Prob(JB):	0.00456
Kurtosis:	2.845	Cond. No.	24.2

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

*Hay más que estudiar sobre los modelos, en el momento, se encuentra más allá de mi conocimiento, pero **estoy interesado en saber más sobre ellos**.*

4.7.b Predicción de las renuncias - Modelo de supervivencia con el método Kaplan-Meier

Después de averiguar la tasa de renuncias de nuestra empresa, investigaré la posibilidad de desarrollar un modelo de aprendizaje automático que anticipe las dimisiones.

Para este análisis usaré los valores de las columnas 'Attrition', 'YearsAtCompany' y 'AgeRange'.

In [ ]:
```
# Creamos la base de datos para nuestro análisis
df_km_attrition = df_padb[['YearsAtCompany', 'Attrition', 'AgeRange']].copy()
df_km_attrition.head()
```

Out[ ]:

	YearsAtCompany	Attrition	AgeRange
0	6	Yes	38 to 47
1	10	No	48 to 57
2	0	Yes	28 to 37
3	8	No	28 to 37
4	2	No	18 to 27

Para trabajar con nuestro modelo, codificaré los valores de las columnas 'Attrition' y 'AgeRange'.

In [ ]:
```
# Columna Attrition
# Creamos el código para la columna de Attrition
attrition_ref = {
    "Yes": 1,
    "No": 0
}
# Modificamos los valores de la columna Attrition con el código
df_km_attrition['Attrition'] = df_km_attrition['Attrition'].map(attrition_ref)

# Columna AgeRange
# Codificamos los valores de la columna AgeRange
encoded_age_range = pd.get_dummies(df_km_attrition['AgeRange'], prefix='AgeRange')

# Concatenamos los datos y los agregamos a nuestra base
df_km_attencoded = pd.concat([df_km_attrition, encoded_age_range], axis=1)
df_km_attencoded.drop('AgeRange', axis='columns' , inplace=True)
df_km_attencoded.head()
```

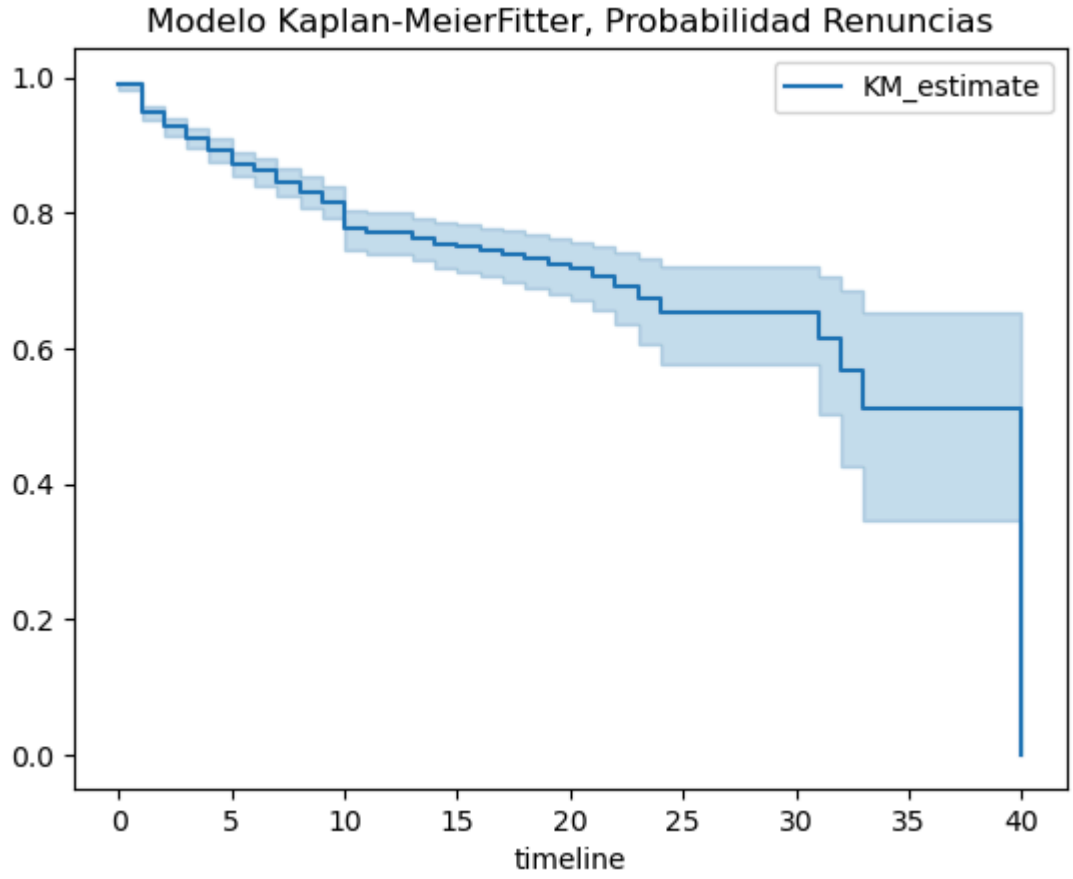
Out[ ]:

	YearsAtCompany	Attrition	AgeRange_18 to 27	AgeRange_28 to 37	AgeRange_38 to 47	AgeRange_48 to 57	AgeRang t
0	6	1	0	0	1	0	
1	10	0	0	0	0	1	
2	0	1	0	1	0	0	
3	8	0	0	1	0	0	
4	2	0	1	0	0	0	

Ahora es el momento de calcular las curvas de supervivencia con el Kaplan-MeierFitter.

In [ ]:
```
# Importamos el paquete Kaplan-MeierFitter para el análisis
from lifelines import KaplanMeierFitter
from matplotlib.pyplot import title

# Inicializamos el modelo Kaplan-MeierFitter y lo almacenamos dentro de una variabl
kmf = KaplanMeierFitter()

# Utilizamos el modelo junto con los datos
kmf.fit(df_km_attencoded['YearsAtCompany'], event_observed = df_km_attencoded['Attr

# Verificamos el estado de la curva
kmf.plot(title="Modelo Kaplan-MeierFitter, Probabilidad Renuncias")
```

Out[ ]:
```
<Axes: title={'center': 'Modelo Kaplan-MeierFitter, Probabilidad Renuncias'}, xlab
el='timeline'>
```



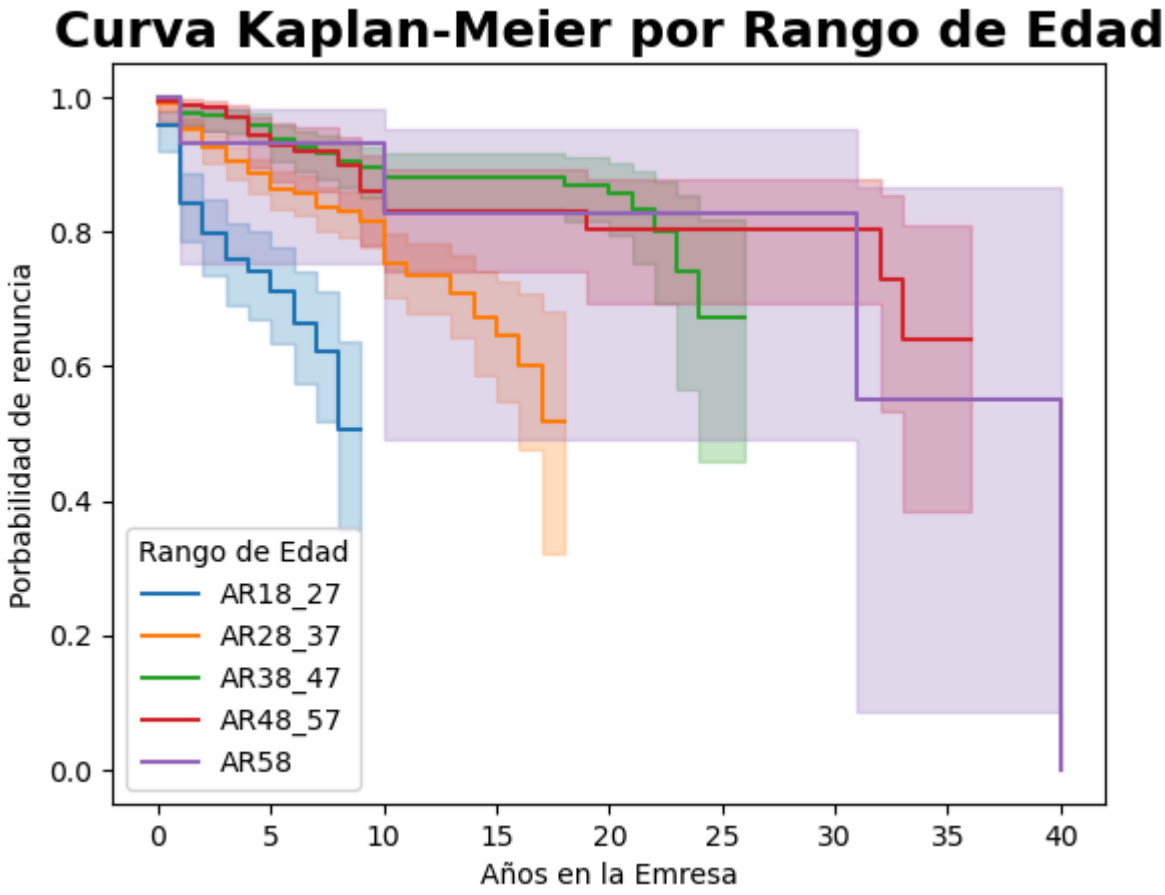
Las primeras impresiones de la investigación nos llevan a creer que el primer período de abandono del trabajo puede ocurrir durante los primeros años de empleo. Necesitamos considerar dónde la trama nos muestra grandes saltos. El segundo período de renuncias es de alrededor de ***10 años*** *en la empresa. Después de eso, se presenta un período de estabilidad, hasta los* ***25 años*** *es cuando comienza el período de jubilación.*

Para tener una idea mejor, vamos a incluir los datos de la columna 'AgeRange'.

In [ ]:
```
# Modificamos el nombre de las columnas del rango de edad
df_km_attencoded.rename(columns={"AgeRange_18 to 27": "AR18_27", "AgeRange_28 to 37
df_km_attencoded.columns
```

Out[ ]:
```
Index(['YearsAtCompany', 'Attrition', 'AR18_27', 'AR28_37', 'AR38_47',
       'AR48_57', 'AR58'],
      dtype='object')
```

In [ ]:
```
# Graficamos las curvas para cada rango de edad
for age_range in ['AR18_27', 'AR28_37', 'AR38_47', 'AR48_57', 'AR58']:
    # Filtramos los datos por rango de edad
    years_at_work = df_km_attencoded.loc[df_km_attencoded[age_range] == 1, 'YearsAt
    attrition = df_km_attencoded.loc[df_km_attencoded[age_range] == 1, 'Attrition']

    # Utilizamos el modelo Kaplan-Meier para cada rango de edad
    kmf.fit(years_at_work, event_observed=attrition, label=age_range)

    # Graficamos los datos
    kmf.plot()

# Añadimos título y etiquetas
plt.xlabel('Años en la Emresa')
plt.ylabel('Porbabilidad de renuncia')
plt.legend(title='Rango de Edad')
plt.title('Curva Kaplan-Meier por Rango de Edad', fontsize=18, fontweight='bold')

plt.show()
```



*Con la adición de los valores 'AgeRange', ahora tenemos 5 líneas. Cada uno de ellos muestra información diferente. Para el rango de **18 a 27**, podemos apreciar que la probabilidad de renunciar la empresa es en el primer año en la compañía, y luego en unos nueve años más tarde. En el rango de **28 a 37**, la probabilidad de renunciar la empresa es también en el primer año, luego diez años más tarde, y el último punto sucede cuando tienen 16 o 17 años trabajando en la empresa. Los rangos **38 a 47** y **48 a 57** muestran más estabilidad en la empresa. Para el último rango, **más de 58**, tenemos tres pausas importantes: el primer año, luego a los diez años, y el último a los 31 años.*

4.8 Conclusión y Observaciones

Después de realizar un análisis exploratorio de la base de datos 'WA_Fn-UseC_-HR-Employee-Attrition', pude determinar que la organización posee un "buen balance" entre los niveles de estudio, el género y la edad de los empleados. A partir de este punto, la organización puede decidir las políticas de diversidad que considere necesarias.

*En cuanto a los niveles de satisfacción, están en **niveles aceptables**, pero se recomienda prestar atención al porcentaje de disconformes, y tomar medidas para reducir el porcentaje y evitar problemas a futuro, especialmente en puestos que están muy involucrados con el trabajo.*

Los planes de carrera pueden necesitar ajustes, particularmente porque, como se puede ver en las renuncias, la empresa está perdiendo empleados en rangos de edad bajos. No se pudo medir es el ingreso de empleados nuevos para tener un mejor panorama de la entrada y salida de los empleados.

Los sueldos muestran estar acordes a los rangos del nivel educativo que poseen los empleados.

Las horas dedicadas a la Capacitación son equitativas para todos los departamentos y están dentro de los rangos normales. Tener presente que los diferentes departamentos tienen diferentes números de empleados y que todos tienen el mismo promedio de horas dedicadas a capacitarse.

La empresa tuvo un buen balance trabajo-familia para sus empleados, y no hubo anomalías significativas en el balance trabajo-familia. Se recomienda seguir con lo que se está haciendo.

*En cuanto a **las renuncias**, se descubrió que la mayoría de ellas podrían estar relacionadas con los salarios por debajo del promedio. Además, las personas de **18 a 47** mostraron la mayor mobilidad. Si al análisis le agregamos el nivel educativo, se puede decir que la empresa perdió talento. Sin embargo no se pudo comprobar si la empresa realizó ingresos para compensarlos.*

Después del análisis descriptivo, el siguiente paso será recrear el tiempo para hacer una comparación de las métricas.