HOTEL HR PEOPLE ANALYTICS

Analizamos tres hoteles desde una perspectiva de People Analytics.

Con el siguiente proyecto, intentaré emular un análisis de recursos humanos con el fin de continuar practicando mis habilidades analíticas. Además, con este proyecto, quiero fortalecer mis conocimientos en ***MySql y SQL***. Usaré ***Python*** y ***Power BI*** para el análisis de datos.

Comenzaré haciendo un poco de ingeniería de datos para diseñar las bases de datos con la que voy a trabajar. Usaré ***Figma*** para diseñar la base de datos, las tablas que la van a componer y las relaciones entre ellas. Luego, completaré la base datos aleatorios, y más tarde comenzaré el análisis.

1. Importando las librerías

In [1]:
```
# Libraries to manipulate the data
import pandas as pd
import numpy as np
from datetime import datetime, timedelta
import random
import string
from app_pass import dbpass

# Library to deploy charts with the data
import seaborn as sns
import matplotlib.pyplot as plt

# Statmodels for predictions
import statsmodels.api as sm
import statsmodels.formula.api as smf

# Connect to our MySQL database
import mysql.connector
from sqlalchemy import create_engine


# This is to ignore warnings.
import warnings
warnings.filterwarnings('ignore')
```

2. Trabajando con nuestras tablas

Antes de trabajar con Python, ya antes había creado las tablas con las que iba a trabajar. Algunos datos no son muy fáciles de crear aleatoriamente, primero las realicé con Excel y luego las descargué en un archivo *.xlsx* para trabajar con ellos.

El archivo con el que vamos a trabajar se llama ***hotel_hranalytics.xlsx***, vayamos a trabajar con él.

2.1 Tabla 'Employees'

In [2]:
```
# Cargamos nuestra tabla de empleados
df_rawemp = pd.read_excel('../hotel_hranalytics/hotel_hranalytics.xlsx', sheet_n

df_rawemp.head()
```

Out[2]:

	emp_id	Name	Surname	Birthday	Age	Gender	on_license	hotel_id
0	3272	James	Smith	1957-08-09	67	M	0	FUESSP
1	3074	John	Johnson	1981-11-19	42	M	0	FUESSP
2	6627	Robert	Williams	1983-10-15	41	M	0	FUESSP
3	420	Michael	Brown	1976-04-05	48	M	0	FUESSP
4	4856	William	Jones	1968-11-20	55	M	0	FUESSP

2.2 Tabla 'Hotels'

In [3]:
```
df_rawht = pd.read_excel('../hotel_hranalytics/hotel_hranalytics.xlsx', sheet_na

df_rawht.head()
```

Out[3]:

	hotel_id	Name	Location	Opening	Stars	Budget
0	FUESSP	Sandy Shores Park	28 03 18.9N-14 19 21.4W	2001-03-05	4	350000000
1	TFNOBH	Ocean Breeze Haven	28 05 56.5N-16 44 54.6W	1998-10-05	5	550000000
2	ACECWR	Coral Wave Resort	28 51 25.9N-13 47 48.7 W	2000-05-05	5	480000000

2.3 Tabla 'Hotels_Composition'

In [4]:
```
df_rawhtcomp = pd.read_excel('../hotel_hranalytics/hotel_hranalytics.xlsx', shee

df_rawhtcomp.head()
```

Out[4]:

	hc_id	Department	Active_employees	Emp_with_license	Total_employe
0	REFUESSP	Reception_Reservations	11	1	
1	FLFUESSP	Floors_Laundry	35	3	
2	KIFUESSP	Kitchen	35	3	
3	BAFUESSP	Bar_Restaurant	31	7	
4	ANFUESSP	Animation	11	1	

2.4 Tabla 'Employees_Wages'

In [5]:
```
df_rawempwages = pd.read_excel('../hotel_hranalytics/hotel_hranalytics.xlsx', sh

df_rawempwages.head()
```

Out[5]:

	emp_wag_id	Price_$_Hour	Hours_worked	Work_overtime	Ovh$_75%	Gross_pay
0	3272REFUESSP	14	129	4	10.50	1848.00
1	3074REFUESSP	14	143	3	10.50	2033.50
2	6627REFUESSP	18	135	4	13.50	2484.00
3	420REFUESSP	19	121	11	14.25	2455.75
4	4856REFUESSP	14	132	7	10.50	1921.50

2.5 Tabla 'Workforce_Composition'

In [6]:
```
df_rawworkforce = pd.read_excel('../hotel_hranalytics/hotel_hranalytics.xlsx', s

df_rawworkforce.head()
```

Out[6]:

	wkc_id	Department	Position	years_at_position	Entry_date	year
0	3272FUESSP	Reception_Reservations	Staff	1	2023-09-26	
1	3074FUESSP	Reception_Reservations	Staff	1	2023-04-29	
2	6627FUESSP	Reception_Reservations	3rd_Command	4	2014-01-17	
3	420FUESSP	Reception_Reservations	3rd_Command	3	2012-10-25	
4	4856FUESSP	Reception_Reservations	Staff	1	2023-06-18	

2.6 Testeando los 'dtypes'

Decidí combinar todas las tablas en una sola con el fin de poder visualizar todos los tipos de datos que tiene cada columna, así evitar utilizar el comando dtypes para cada tabla en forma individual. Este es el único propósito de esta tabla combinada, y no se utilizará en análisis posteriores.

In [7]:
```
# Combinando todas las tablas
df_combined = pd.concat([df_rawemp, df_rawht, df_rawhtcomp, df_rawworkforce, df_

df_combined.dtypes
```

Out[7]:
```
emp_id                      object
Name                        object
Surname                     object
Birthday            datetime64[ns]
Age                        float64
Gender                      object
on_license                 float64
hotel_id                    object
Location                    object
Opening                     object
Stars                      float64
Budget                     float64
hc_id                       object
Department                  object
Active_employees           float64
Emp_with_license           float64
Total_employees            float64
wkc_id                      object
Position                    object
years_at_position          float64
Entry_date          datetime64[ns]
years_working              float64
Staff                      float64
emp_wag_id                  object
Price_$_Hour               float64
Hours_worked               float64
Work_overtime              float64
Ovh$_75%                   float64
Gross_pay                  float64
Deductions_3%              float64
Total_Payment              float64
Payment_date        datetime64[ns]
dtype: object
```

Luego de verificar los tipos de datos, se pudo comprobar que algunas columnas necesitaban cambiar su tipo de dato. El siguiente paso, enfatizará en corregir esos tipos de datos incorrectos, de acuerdo con los que establecimos en nuestro boceto de **Figma**.

2.7 Arreglando las 'tipos de datos' de las columnas

In [8]:
```
# Tabla Employees
df_rawemp[['Age', 'on_license']].apply(pd.to_numeric)
df_rawemp[['hotel_id']].astype('str')

# Tabla Hotels
df_rawht['Stars'].apply(pd.to_numeric)
df_rawht[['hotel_id']].astype('str')
df_rawht.rename(columns={'Stars': 'Stars_type'}, inplace=True)

# Tabla Hotel Composition
df_rawhtcomp[['Active_employees', 'Emp_with_license', 'Total_employees']].apply(
df_rawhtcomp[['hc_id', 'hotel_id']].astype('str')

# Tabla Workforce Composition
df_rawworkforce[['years_at_position', 'years_working', 'Staff']].apply(pd.to_num
df_rawworkforce[['wkc_id', 'emp_id', 'hotel_id', 'hc_id']].astype('str')

# Tabla Employees Wages
df_rawempwages[['emp_wag_id', 'emp_id', 'hotel_id',     'hc_id']].astype('str')
```

```
df_rawempwages[['Price_$_Hour']].apply(pd.to_numeric)
df_rawempwages.rename(columns={'Ovh$_75%': 'Ovh$_75', 'Deductions_3%': 'Deductio
```

Fué necesario corregir algunos nombres de algunas columnas para que fueran similares a los creados en la base de datos en *MySQL*.

Realizamos una última verificación de nuestros datos con el fin de solucionar aquellos problemas menores que así lo requieran.

In [9]:
```
df_rawemp.dtypes
```

Out[9]:
```
emp_id                object
Name                  object
Surname               object
Birthday      datetime64[ns]
Age                    int64
Gender                object
on_license             int64
hotel_id              object
dtype: object
```

In [10]:
```
df_rawhtcomp.dtypes
```

Out[10]:
```
hc_id               object
Department          object
Active_employees     int64
Emp_with_license     int64
Total_employees      int64
hotel_id            object
dtype: object
```

In [11]:
```
df_rawworkforce = df_rawworkforce.rename(columns={'Position': 'Positions'})
df_rawworkforce.dtypes
```

Out[11]:
```
wkc_id                      object
Department                  object
Positions                   object
years_at_position            int64
Entry_date          datetime64[ns]
years_working                int64
Staff                        int64
emp_id                       int64
hotel_id                    object
hc_id                       object
dtype: object
```

In [12]:
```
df_rawempwages.dtypes
```

Out[12]:
```
emp_wag_id              object
Price_$_Hour             int64
Hours_worked             int64
Work_overtime            int64
Ovh$_75                float64
Gross_pay              float64
Deductions_3           float64
Total_Payment          float64
emp_id                   int64
hotel_id                object
hc_id                   object
Payment_date     datetime64[ns]
dtype: object
```

In [13]:
```
# Verificando si tenemos valores nulos
missing_values = df_rawemp.isnull().sum()
print('Los valores que faltan son: ', missing_values)
```

```
Los valores que faltan son:  emp_id        0
Name          0
Surname       0
Birthday      0
Age           0
Gender        0
on_license    0
hotel_id      0
dtype: int64
```

3.1 Dejemos que las *visualizaciones* nos muestren lo que los datos nos quieren contar.

Comencemos con algo simple. ¿Cómo están distribuidos nuestros empleados por género?

Para responder a la pregunta, vamos a utilizar la tabla *Employees*.

In [14]:
```
# Trabajamos con la columna de Género, pero antes vamos a crear una variable con
emp_length = len(df_rawemp)
print('El total de registros que tenemos en la tabla de Empleados es: ', emp_len
```

```
El total de registros que tenemos en la tabla de Empleados es:  505
```

In [15]:
```
# Contamos los valores de la columna Gender
emp_gender = df_rawemp['Gender'].value_counts()
print(emp_gender)
```

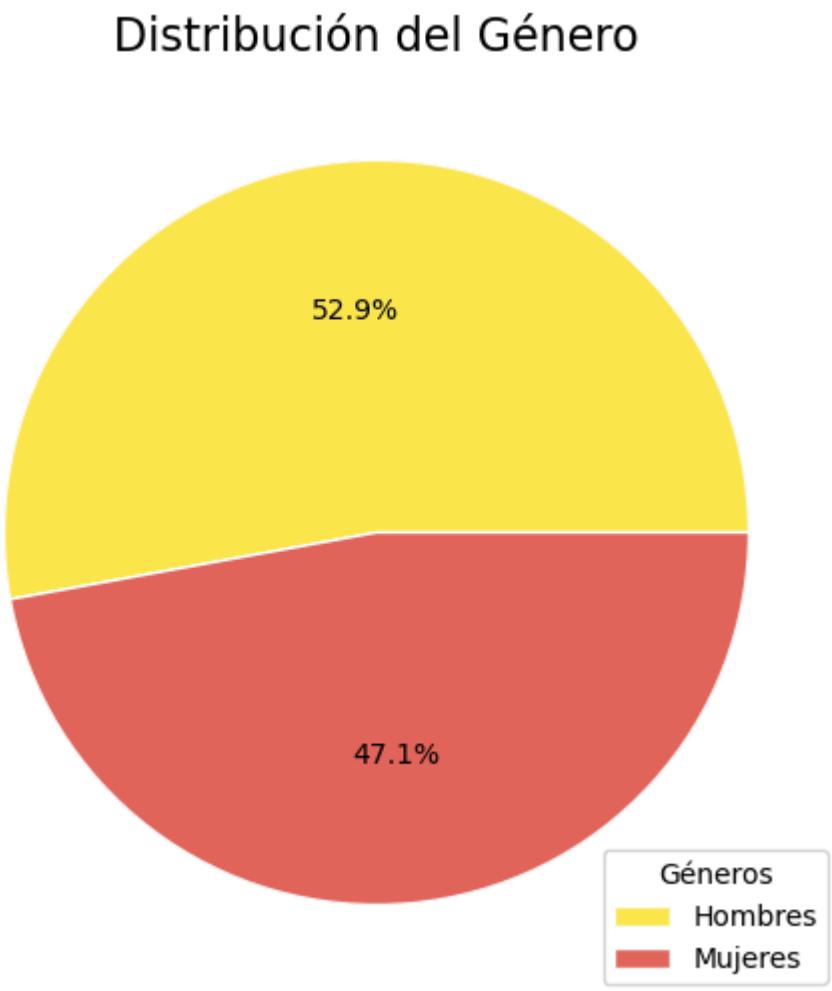
```
Gender
M    267
F    238
Name: count, dtype: int64
```

3.1.a Es hora de visualizar los datos de género

Para visualizar la distribución del género, vamos a utilizar un **gráfico de torta**.

In [44]:
```
# Preparando los datos para el gráfico
colors = plt.get_cmap('Blues')(np.linspace(0.2, 0.7, len(emp_gender)))
labels = 'Hombres', 'Mujeres'
```

```
# Creando el gráfico de Torta
fig, ax = plt.subplots(figsize=(6, 8))
ax.pie(emp_gender, colors=['#FDE74C', '#E3655B'], autopct='%1.1f%%', center=(4,
ax.legend(labels, loc='lower right', title='Géneros')
ax.set_title('Distribución del Género', fontsize=16)

plt.show()
```



Los datos nos muestran la distribución por género en nuestros tres hoteles. Los empleados están repartidos de la siguiente forma, tenemos un total de ***238 mujeres***, lo que representa un **47,1%** de la plantilla total. Y en cuanto a los ***hombres***, tenemos ***267 empleados***, representa un **52,9%** de la plantilla total.

3.1.b Analicemos la distribución por género por Departamentos y Hoteles

Para calcular la distribución por género separa en los distintos departamentos, será necesario combinar las tablas de "*Employees*" y "*Workforce Composition*". Utilizaré la columna '*emp_id*' para combinar ambas tablas.

In [17]:
```
# Preparando los datos para el gráfico
df_rawworkforce['emp_id'] = df_rawworkforce['emp_id'].astype('str')
emp_gender_by_dep = pd.merge(df_rawemp, df_rawworkforce, on='emp_id', how='inner
emp_gender_by_dep.head()
```

Out[17]:

	emp_id	Name	Surname	Birthday	Age	Gender	on_license	hotel_id_x	wkc_
0	3272	James	Smith	1957-08-09	67	M	0	FUESSP	3272FUES
1	3074	John	Johnson	1981-11-19	42	M	0	FUESSP	3074FUES
2	6627	Robert	Williams	1983-10-15	41	M	0	FUESSP	6627FUES
3	420	Michael	Brown	1976-04-05	48	M	0	FUESSP	420FUES
4	4856	William	Jones	1968-11-20	55	M	0	FUESSP	4856FUES

In [18]:
```
# Eliminamos las columnas que no vamos a utilizar
emp_gender_by_dep.drop(columns=['Name', 'Surname', 'Birthday', 'wkc_id', 'years_
emp_gender_by_dep.head()
```

Out[18]:

	emp_id	Age	Gender	on_license	hotel_id_x	Department	Positions
0	3272	67	M	0	FUESSP	Reception_Reservations	Staff
1	3074	42	M	0	FUESSP	Reception_Reservations	Staff
2	6627	41	M	0	FUESSP	Reception_Reservations	3rd_Command
3	420	48	M	0	FUESSP	Reception_Reservations	3rd_Command
4	4856	55	M	0	FUESSP	Reception_Reservations	Staff

In [19]:
```
# Revisamos si tenemos valores nulos
gender_by_dep_missing_values = emp_gender_by_dep.isnull().sum()
print('Los valores que faltan son: ', gender_by_dep_missing_values)
```

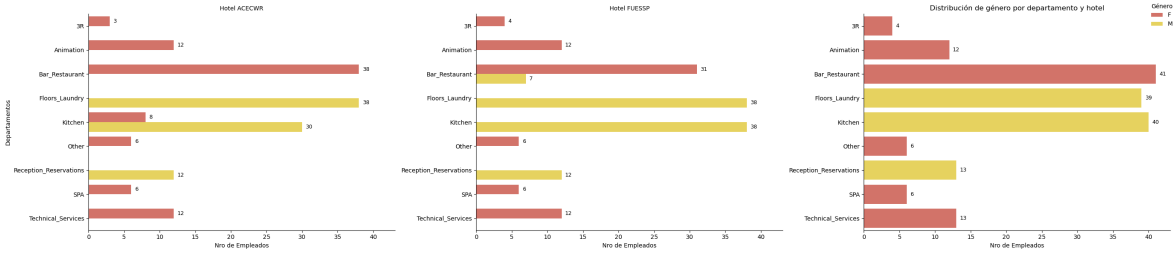
```
Los valores que faltan son:  emp_id        0
Age           0
Gender        0
on_license    0
hotel_id_x    0
Department    0
Positions     0
dtype: int64
```

Es hora de visualizar nuestros datos de distribución de género por *Hotel* y *Departamentos*

In [20]:
```
# Preparando los datos para el gráfico
gender_dist = emp_gender_by_dep.groupby(['hotel_id_x', 'Department', 'Gender']).
gender_dist.rename(columns={0: 'Count'}, inplace=True)
print(gender_dist)

# Convertimos el tipo de dato a número de la columna Count
gender_dist['Count'] = gender_dist['Count'].astype('int64')
gender_dist['Count'].dtypes

# Usamos Seaborn para crear un gráfico de barras con FacetGrid
```

```
grid = sns.FacetGrid(
    gender_dist,
    col='hotel_id_x',
    height=6,
    aspect=1.5,
    sharey=False
)

# Dibujamos el gráfico de barras
grid.map_dataframe(
    sns.barplot,
    y='Department',
    x='Count',
    hue='Gender',
    palette=['#E3655B','#FDE74C']
)

# Añadimos el conteo a cada barra
for ax in grid.axes.flat:
    for container in ax.containers:
        for bar in container:
            bar_value = bar.get_width()
            if bar_value > 0:
                ax.text(
                    bar_value + 0.5,
                    bar.get_y() + bar.get_height() / 2,
                    f"{int(bar_value)}",
                    ha='left',
                    va='center',
                    fontsize=9,
                    color='black'
                )

# Titulos y leyendas
grid.add_legend(title='Género')
grid.legend.set_loc('upper right')
grid.set_titles('Hotel {col_name}')
grid.set_axis_labels('Nro de Empleados', 'Departamentos')
plt.tight_layout()
plt.title('Distribución de género por departamento y hotel')

plt.show()
```

	hotel_id_x	Department	Gender	Count
0	ACECWR	3R	F	3
1	ACECWR	Animation	F	12
2	ACECWR	Bar_Restaurant	F	38
3	ACECWR	Floors_Laundry	M	38
4	ACECWR	Kitchen	F	8
5	ACECWR	Kitchen	M	30
6	ACECWR	Other	F	6
7	ACECWR	Reception_Reservations	M	12
8	ACECWR	SPA	F	6
9	ACECWR	Technical_Services	F	12
10	FUESSP	3R	F	4
11	FUESSP	Animation	F	12
12	FUESSP	Bar_Restaurant	F	31
13	FUESSP	Bar_Restaurant	M	7
14	FUESSP	Floors_Laundry	M	38
15	FUESSP	Kitchen	M	38
16	FUESSP	Other	F	6
17	FUESSP	Reception_Reservations	M	12
18	FUESSP	SPA	F	6
19	FUESSP	Technical_Services	F	12
20	TFNOBH	3R	F	4
21	TFNOBH	Animation	F	12
22	TFNOBH	Bar_Restaurant	F	41
23	TFNOBH	Floors_Laundry	M	39
24	TFNOBH	Kitchen	M	40
25	TFNOBH	Other	F	6
26	TFNOBH	Reception_Reservations	M	13
27	TFNOBH	SPA	F	6
28	TFNOBH	Technical_Services	F	13

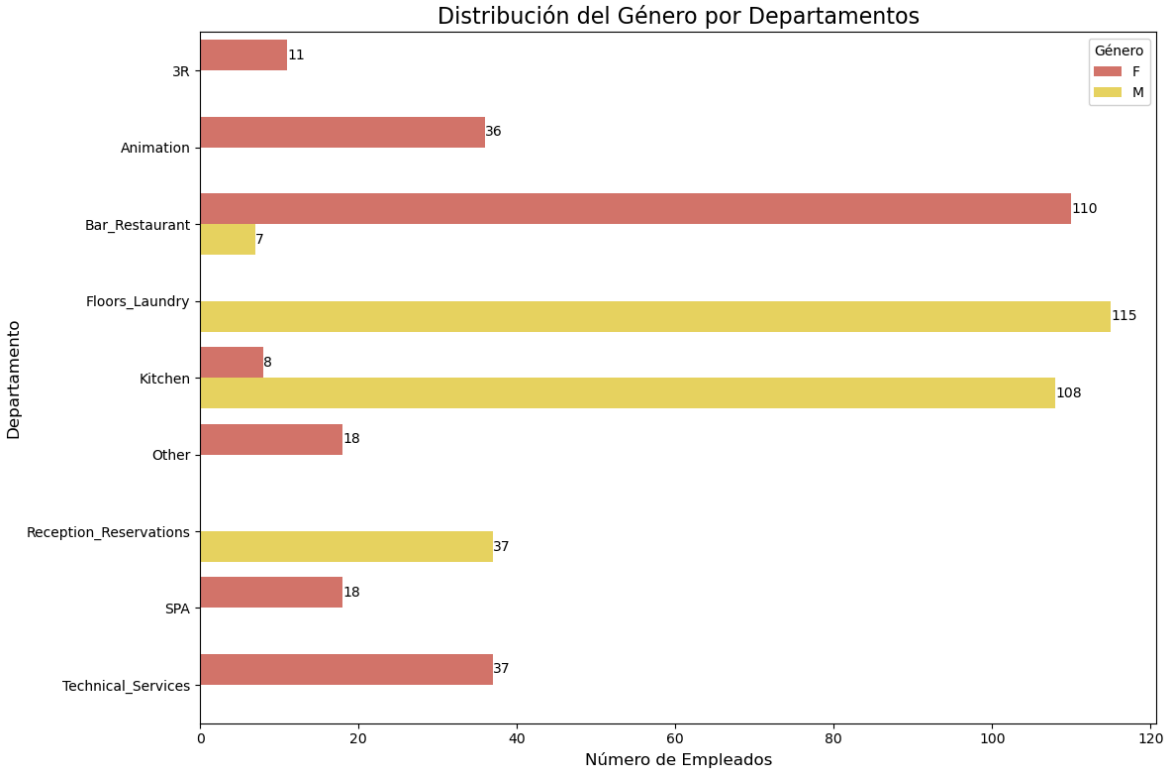


Distribución del Género por *Departamentos*

In [21]:
```
# Preparamos los datos
gender_by_dep = emp_gender_by_dep.groupby(['Department', 'Gender']).size().reset
print(gender_by_dep)

# Creamos el gráfico de barras
plt.figure(figsize=(12, 8))
ax = sns.barplot(
    data=gender_by_dep,
    x='Count',
    y='Department',
    hue='Gender',
    palette=['#E3655B','#FDE74C'],
    ci=None
)

# Añadimos el conteo a cada barra
for container in ax.containers:
    ax.bar_label(container, fmt='%d', label_type='edge', fontsize=10, color='bla
```

```
# Dibujamos el gráfico
plt.title('Distribución del Género por Departamentos', fontsize=16)
plt.xlabel('Número de Empleados', fontsize=12)
plt.ylabel('Departamento', fontsize=12)
plt.legend(title='Género')
plt.tight_layout()


plt.show()
```

	Department	Gender	Count
0	3R	F	11
1	Animation	F	36
2	Bar_Restaurant	F	110
3	Bar_Restaurant	M	7
4	Floors_Laundry	M	115
5	Kitchen	F	8
6	Kitchen	M	108
7	Other	F	18
8	Reception_Reservations	M	37
9	SPA	F	18
10	Technical_Services	F	37



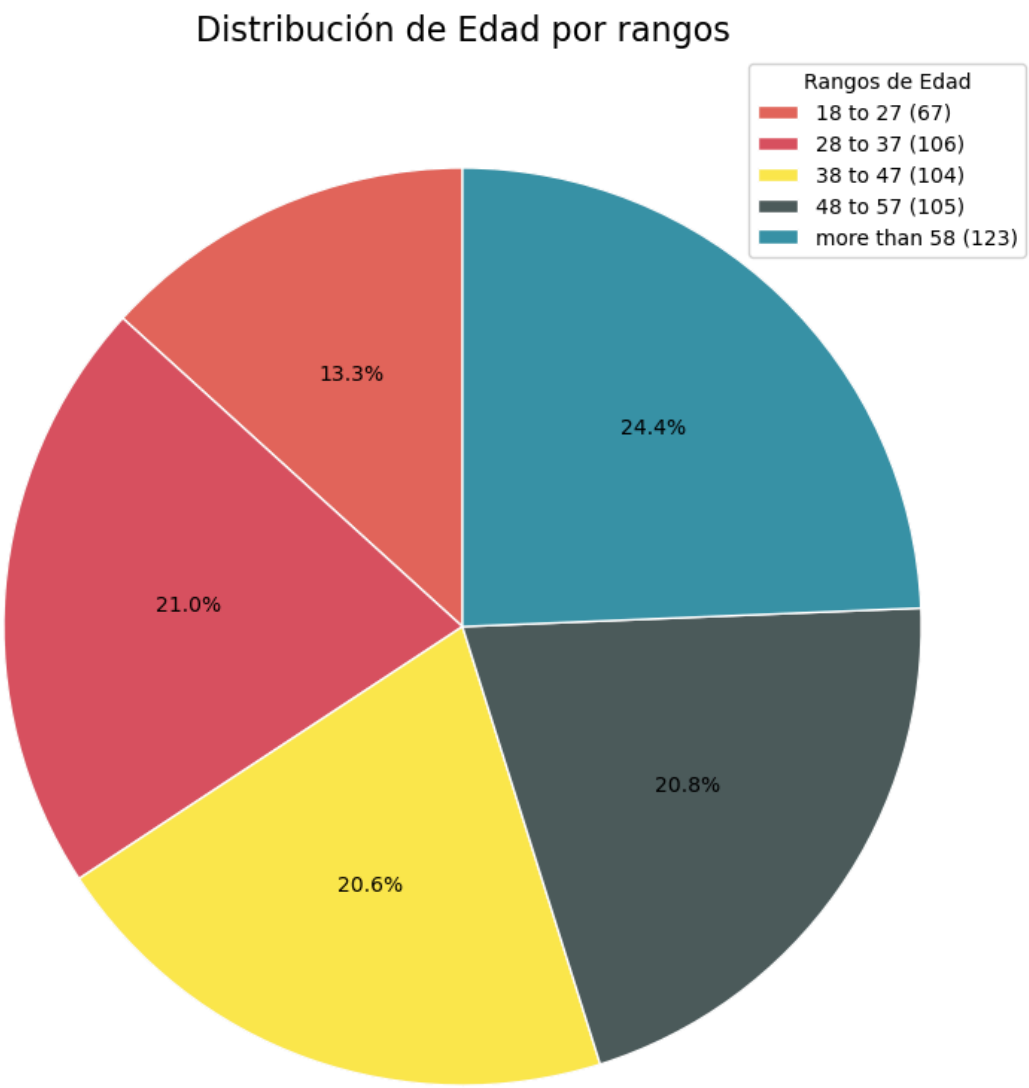
3.2 Analicemos cómo se distribuyen nuestros empleados por *edad*

Crearé un rango de edad con el fin de facilitar el análisis

In [22]:
```
# Creamos una función para clasificar la edad en rangos
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'

# Aplicamos la función a la columna Age
emp_age_range = df_rawemp['Age'].apply(lambda x: pd.Series(age_range(x)))

# Creamos una nueva columna con los rangos de edad
df_rawemp['Age_range'] = emp_age_range

# Chequeamos los valores del rango de edad
age_range_count = df_rawemp['Age_range'].value_counts().sort_index()
total_agerange_count = age_range_count.sum()
percentage = (age_range_count / total_agerange_count) * 100

# Creamos un gráfico de torta
fig, ax = plt.subplots(figsize=(12, 10))
colors = ['#E3655B', '#DB5461', '#FDE74C', '#4C5B5C', '#3891A6']
labels = [f'{age} ({count})' for age, count in zip(age_range_count.index, age_ra
labels_sort = df_rawemp['Age_range'].value_counts().sort_index()
graph_labels = '18 to 27', '28 to 37', '38 to 47', '48 to 57', 'more than 58'

ax.pie(age_range_count, autopct='%1.1f%%', center=(4, 4), wedgeprops={"linewidth

plt.legend(labels, loc='upper right', title='Rangos de Edad')

ax.set_title('Distribución de Edad por rangos', fontsize=16)
```

Out[22]: Text(0.5, 1.0, 'Distribución de Edad por rangos')



Tras separar a los empleados de los tres hoteles por rangos de edad, se obtuvieron los siguientes resultados:

Rango de edad

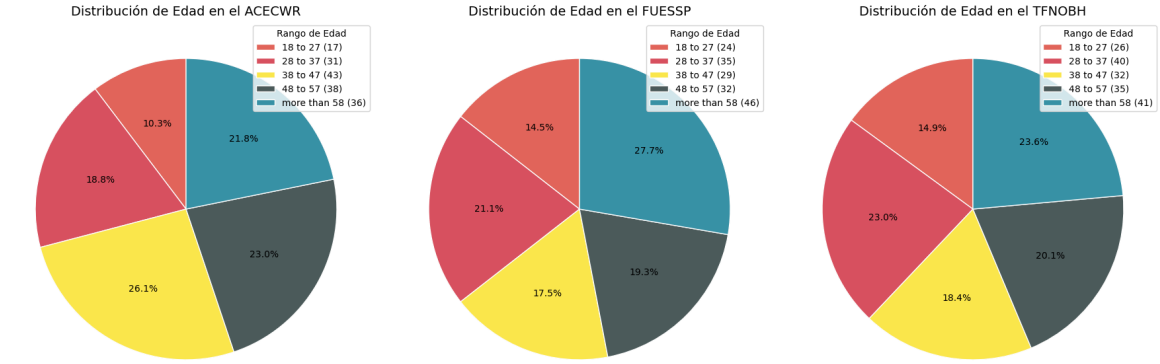
- De 18 a 27 años un total de **67** empleados que representan un ***13,3%***
- De 28 a 37 años un total de **106** empleados que representan un ***21%***
- De 38 a 47 años un total de **104** empleados que representan un ***20,6%***
- De 48 a 57 años un total de **105** empleados que representan un ***20,8%***
- Más de 58 años un total de **123** empleados que representan un ***24,4%***

Al observar la distribución del personal por rango de edad, podemos observar que la mayoría de nuestro personal tiene más de 28 años. El 13,3% de nuestro personal tiene entre 18 y 27 años, lo que indica que *necesitamos comenzar a actualizar nuestra plantilla*. Se recomienda que las futuras contrataciones de personal se concentren en este grupo de edad.

Determinemos qué hoteles deberían empezar a contratar más empleados jóvenes.

In [23]:
```
# Distribución de Edad por Hotel
agerange_by_hotel = df_rawemp.groupby(['hotel_id', 'Age_range']).size().unstack(
print(agerange_by_hotel)

# Creando el gráfico de torta
fig, axs = plt.subplots(1, 3, figsize=(18, 6))

colors = ['#E3655B', '#DB5461', '#FDE74C', '#4C5B5C', '#3891A6']

for i, hotel in enumerate(agerange_by_hotel.index):
    ax = axs[i]
    labelsbyhotel = [f'{age} ({count})' for age, count in zip(age_range_count.in
    ax.pie(agerange_by_hotel.loc[hotel], autopct='%1.1f%%', center=(4, 4), wedge
    ax.set_title(f'Distribución de Edad en el {hotel}', fontsize=14)
    ax.legend(labelsbyhotel, loc='upper right', title='Rango de Edad')

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

```
Age_range  18 to 27  28 to 37  38 to 47  48 to 57  more than 58
hotel_id
ACECWR           17        31        43        38            36
FUESSP           24        35        29        32            46
TFNOBH           26        40        32        35            41
```



Tras analizar los rangos de edad de los tres hoteles, hemos determinado que el **Sandy Shores Park** es el que tiene la plantilla más adulta, con 46 empleados mayores de 58 años. Además, este hotel lleva menos tiempo en funcionamiento que los otros dos hoteles. Aunque el personal del **Coral Wave Resort** está bien, se recomienda vigilar a su grupo de *38 a 47*, es el rango que encuentra en la mitad de los rangos de edad y, si se descuidan, podrían terminar con una plantilla de mayor edad en años próximos. El hotel con una plantilla equilibrada es el **Ocean Breeze Haven**.

Continuamos nuestro análisis conociendo la *edad media* que tenemos en los hoteles. La *antigüedad media* del personal. Y por último, ¿cuántos empleados *con licencia* tiene cada hotel?

In [24]:
```
df_rawemp['Age'].describe().round()
```

Out[24]:
```
count    505.0
mean      45.0
std       14.0
min       21.0
25%       33.0
50%       45.0
75%       57.0
max       69.0
Name: Age, dtype: float64
```

In [25]:
```
# Para ver los nombres de los hoteles
hotel_names = df_rawht.set_index('hotel_id')['Name']

# Edad promedio de los empleados por hotel
avg_age_byhotel = df_rawemp.groupby('hotel_id')['Age'].mean().round()

# Años promedio de trabajo de los empleados por hotel
avg_working_years = df_rawworkforce.groupby('hotel_id')['years_working'].mean().

# Imprimir los nombres de los hoteles en los resultados
avg_age_byhotel.index = avg_age_byhotel.index.map(lambda x: f"{x} ({hotel_names[
avg_working_years.index = avg_working_years.index.map(lambda x: f"{x} ({hotel_na

# Para eliminar que figure el nombre del índice
avg_age_byhotel.index.name = None
avg_working_years.index.name = None

print('La edad promedio de los empleados es: ','\n', avg_age_byhotel, '\n\n',
    'El promedio de años de trabajo de los empleados es: ','\n', avg_working_yea
)
```

```
La edad promedio de los empleados es:
 ACECWR (Coral Wave Resort)     45.0
FUESSP (Sandy Shores Park)     45.0
TFNOBH (Ocean Breeze Haven)    44.0
Name: Age, dtype: float64

 El promedio de años de trabajo de los empleados es:
 ACECWR (Coral Wave Resort)     12.0
FUESSP (Sandy Shores Park)     12.0
TFNOBH (Ocean Breeze Haven)    14.0
Name: years_working, dtype: float64
```

3.3 Analizamos la cantindad de empleados que están con licencia

Ahora es el momento de visualizar los empleados que están en licencia. Averigüemos si tenemos una cantidad significativa de empleados en licencia.

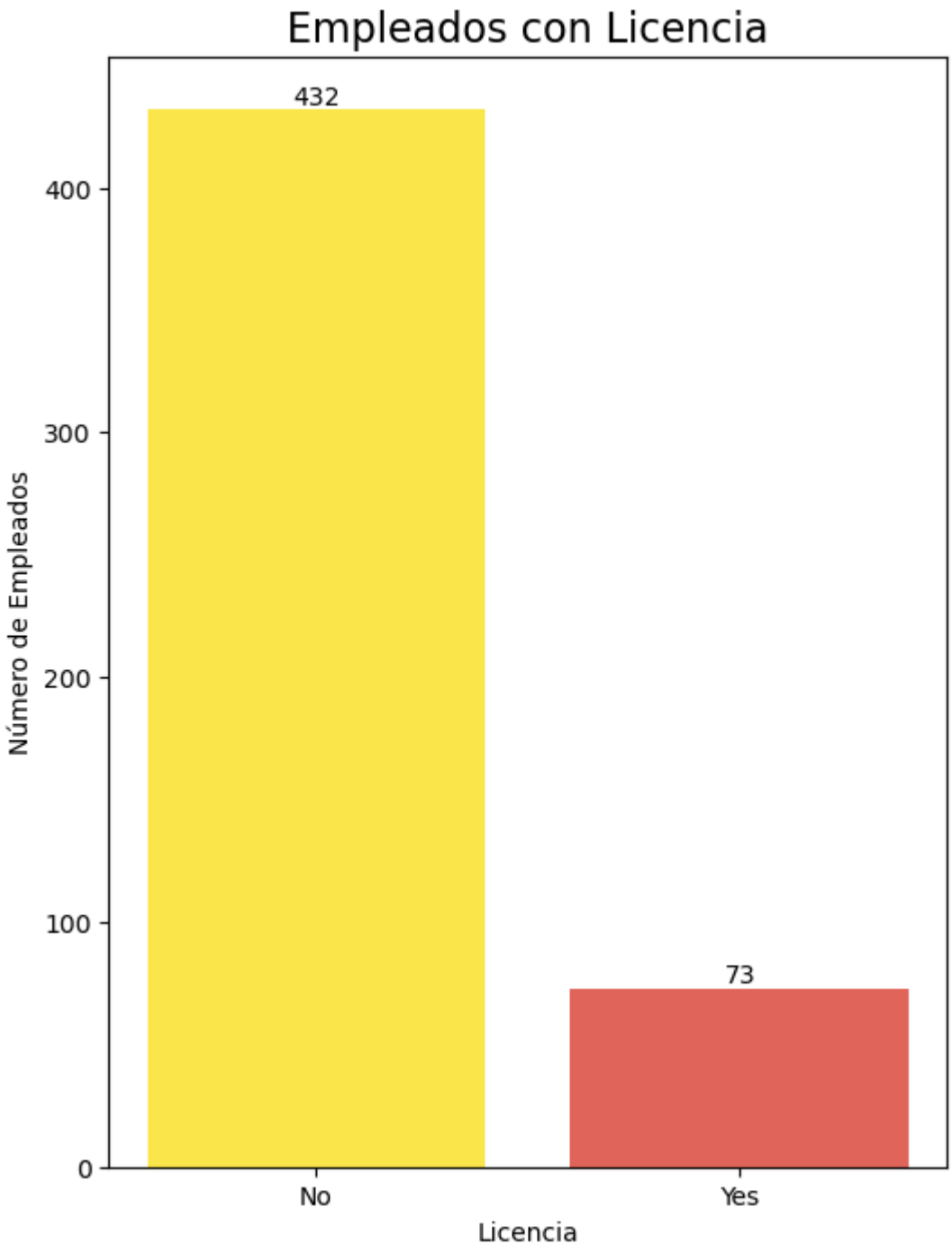
In [26]:
```
# Números de empleados con licencia
emp_on_license = df_rawemp[(df_rawemp['on_license'] == True)].count()
emp_on_license_byhotel = df_rawemp.groupby('hotel_id')['on_license'].sum()
per_emp_on_license = (emp_on_license['on_license'] / emp_length) * 100

# Imprimir los nombres de los hoteles en los resultados
emp_on_license_byhotel.index = emp_on_license_byhotel.index.map(lambda x: f"{x}

# Eliminar el nombre del índice
```

```
emp_on_license_byhotel.index.name = None

print('El número de empleados con licencia es: ','\n', emp_on_license_byhotel, '
    'El total de empleados con licencia es: ', emp_on_license['on_license'], '\n
    'El porcentaje de empleados con licencia es: ', per_emp_on_license.round(2),
)
```

```
El número de empleados con licencia es:
 ACECWR (Coral Wave Resort)     28
FUESSP (Sandy Shores Park)     18
TFNOBH (Ocean Breeze Haven)    27
Name: on_license, dtype: int64

 El total de empleados con licencia es:  73

 El porcentaje de empleados con licencia es:  14.46 %
```

In [27]:
```
# Preparamos los datos para el gráfico
on_license_count = df_rawemp['on_license'].value_counts()
print(on_license_count)

# Creamos un gráfico de barras
fig, ax = plt.subplots(figsize=(6, 8))

labels = 'No', 'Yes'
colors = '#FDE74C', '#E3655B'

ax.bar(labels, on_license_count, color=colors)
ax.bar_label(ax.containers[0], fontsize=10)

plt.title('Empleados con Licencia', fontsize=16)
plt.ylabel('Número de Empleados')
plt.xlabel('Licencia')
plt.show()
```

```
on_license
0    432
1     73
Name: count, dtype: int64
```



De un total de 505 empleados, sólo 73 están de baja con licencia, lo que supone un **14,46%** del total de la plantilla. La empresa tendrá que determinar ahora qué porcentaje se considerará grave.

Es hora de desglosar las licencias de los tres hoteles para visualizar qué hotel tiene más empleados con licencia.

In [28]:
```
# Preparando nuestros datos
onlicense_by_hotel = df_rawemp.groupby(['hotel_id', 'on_license']).size().unstac
print(onlicense_by_hotel)

# Creando los gráficos de torta
fix, axs = plt.subplots(1, 3, figsize=(18, 6))
labels_onlicense = 'No', 'Yes'

for i, hotel in enumerate(onlicense_by_hotel.index):
```

```
    ax = axs[i]
    labels2 = [f'{age} ({count})' for age, count in zip(labels_onlicense, onlice
    ax.pie(onlicense_by_hotel.loc[hotel], autopct='%1.1f%%', center=(4, 4), wedg
    ax.set_title(f'Empleados con licencia por hotel {hotel}', fontsize=14)
    ax.legend(labels2, loc='upper right', title='Licencia')

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

```
on_license    0   1
hotel_id
ACECWR      137  28
FUESSP      148  18
TFNOBH      147  27
```

Al desglosar los datos, se observa que el *Coral Wave Resort*, con **28** empleados, y el *Ocean Breeze Haven*, con **27** empleados, son ambos los hoteles con más personal en licencia.

4. Analizamos la tabla 'Employees Wages'

Ahora procederemos a examinar los pagos salariales anuales. Comenzaré fusionando la tabla de '*Employees_Wages*' con la tabla de '*Workforce_Composition*' para comprender mejor nuestros datos. Esto nos permitirá realizar filtros de datos por Puestos y Departamentos.

In [29]:
```
# Trabajamos con la tabla employees wages
# Para un mejor análisis primero fusionamos las tablas de Employees Wages y Work
df_rawempwages['emp_id'] = df_rawempwages['emp_id'].astype('str')
emp_wages_wfc = pd.merge(df_rawempwages, df_rawworkforce, on='emp_id', how='inne
emp_wages_wfc.head()
```

Out[29]:

	emp_wag_id	Price_$_Hour	Hours_worked	Work_overtime	Ovh$_75	Gross_pay	D
0	3272REFUESSP	14	129	4	10.50	1848.00	
1	3074REFUESSP	14	143	3	10.50	2033.50	
2	6627REFUESSP	18	135	4	13.50	2484.00	
3	420REFUESSP	19	121	11	14.25	2455.75	
4	4856REFUESSP	14	132	7	10.50	1921.50	

5 rows × 21 columns

Con las tablas combinadas, eliminemos aquellas columnas que no utilizaremos durante nuestro análisis.

In [30]:
```
# Eliminamos las columnas que no vamos a utilizar
emp_wages_wfc.drop(columns=['hotel_id_y', 'hc_id_y', 'wkc_id', 'years_at_positio
emp_wages_wfc.head()
```

Out[30]:

	emp_wag_id	Price_$_Hour	Hours_worked	Work_overtime	Ovh$_75	Gross_pay	D
0	3272REFUESSP	14	129	4	10.50	1848.00	
1	3074REFUESSP	14	143	3	10.50	2033.50	
2	6627REFUESSP	18	135	4	13.50	2484.00	
3	420REFUESSP	19	121	11	14.25	2455.75	
4	4856REFUESSP	14	132	7	10.50	1921.50	

In [31]:
```
# Verificamos si tenemos valores nulos
new_missing_values = emp_wages_wfc.isnull().sum()
print('Los valores que faltan son: ', new_missing_values)
```

```
Los valores que faltan son:  emp_wag_id       0
Price_$_Hour     0
Hours_worked     0
Work_overtime    0
Ovh$_75          0
Gross_pay        0
Deductions_3     0
Total_Payment    0
emp_id           0
hotel_id_x       0
hc_id_x          0
Payment_date     0
Department       0
Positions        0
dtype: int64
```

4.1 Para el primer análisis, descubramos el **precio promedio** *por hora* que los hoteles pagan a sus empleados. Además, vamos a calcular las **horas promedio** que trabajan los empleados. ¿Cuánto pagaron los hoteles en salarios durante el año?

Añadiré al análisis el **promedio de horas extras** trabajadas por los empleados, el **total de horas extras** trabajadas por el personal durante todo el año y cuánto pagaron los hoteles por todas esas horas.

In [32]:
```
avg_hour_price = emp_wages_wfc['Price_$_Hour'].mean().round()
avg_hours_worked = emp_wages_wfc['Hours_worked'].mean().round()
emp_wages_wfc['total_paid_NH'] = emp_wages_wfc['Hours_worked'] * emp_wages_wfc['
total_paid_NH = emp_wages_wfc['total_paid_NH'].sum()
avg_OT_hours_worked = emp_wages_wfc['Work_overtime'].mean().round()
total_OT_hours = emp_wages_wfc['Work_overtime'].sum()
emp_wages_wfc['total_paid_OT'] = emp_wages_wfc['Work_overtime'] * emp_wages_wfc[
total_paid_OT = emp_wages_wfc['total_paid_OT'].sum()

print('El precio promedio por hora: ','€', avg_hour_price, '\n\n',
    'El promedio de horas trabajadas por nuestros empleados es: ', avg_hours_wor
    'La SUMA total que pagamos por horas normales es: ', '€', total_paid_NH, '\n
    'El promedio de horas extras trabajadas por nuestros empleados es: ', avg_OT
    'El total de horas extra trabajadas es: ', total_OT_hours, '\n\n'
    'La SUMA total pagada por horas extras: ', '€', total_paid_OT
)
```

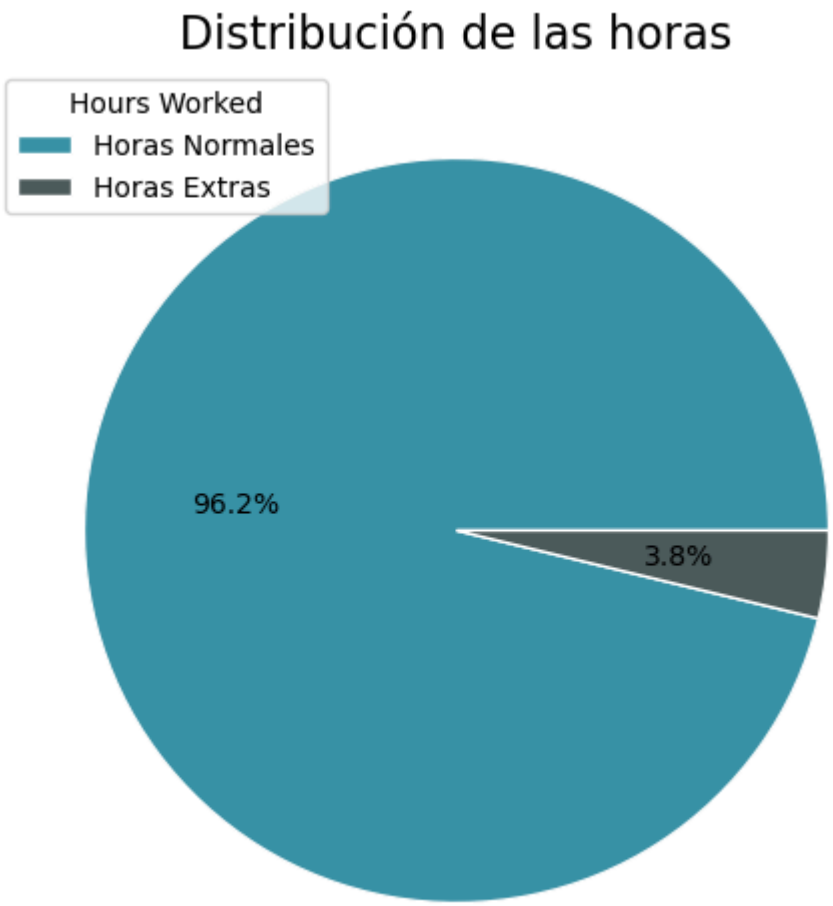
```
El precio promedio por hora:  € 15.0

 El promedio de horas trabajadas por nuestros empleados es:  140.0

 La SUMA total que pagamos por horas normales es:  € 12950211

 El promedio de horas extras trabajadas por nuestros empleados es:  6.0

 El total de horas extra trabajadas es:  33501

La SUMA total pagada por horas extras:  € 382709.25
```

Con un *gráfico circular* visualizaremos cómo se distribuyen las horas de trabajo entre horas normales y horas extras.

In [33]:
```
# Visualizamos el porcentaje de horas trabajadas por nuestros empleados
# Debemos calcular el total de horas trabajadas
```

```
total_hours_worked = emp_wages_wfc['Hours_worked'].sum() + emp_wages_wfc['Work_o
per_NH = (emp_wages_wfc['Hours_worked'].sum() / total_hours_worked) * 100
per_OTh = (emp_wages_wfc['Work_overtime'].sum() / total_hours_worked) * 100
print('The total hours worked by our employees is: ', total_hours_worked, '\n\n'
      'Percentage Normal Hours: ', per_NH.round(2), '\n\n',
      'Percentage Over Time Hours: ', per_OTh.round(2)
      )

# Con los datos anteriores, creamos un gráfico de torta
hours = [per_NH, per_OTh]
colors = ['#3891A6', '#4C5B5C']
labels = ['Horas Normales', 'Horas Extras']
# Creating the PIE CHART
fig, ax = plt.subplots(figsize=(6, 8))
ax.pie(hours, colors=colors, autopct='%1.1f%%', center=(4, 4), wedgeprops={"line
ax.legend(labels, loc='upper left', title='Hours Worked')
ax.set_title('Distribución de las horas', fontsize=16)

plt.show()
```

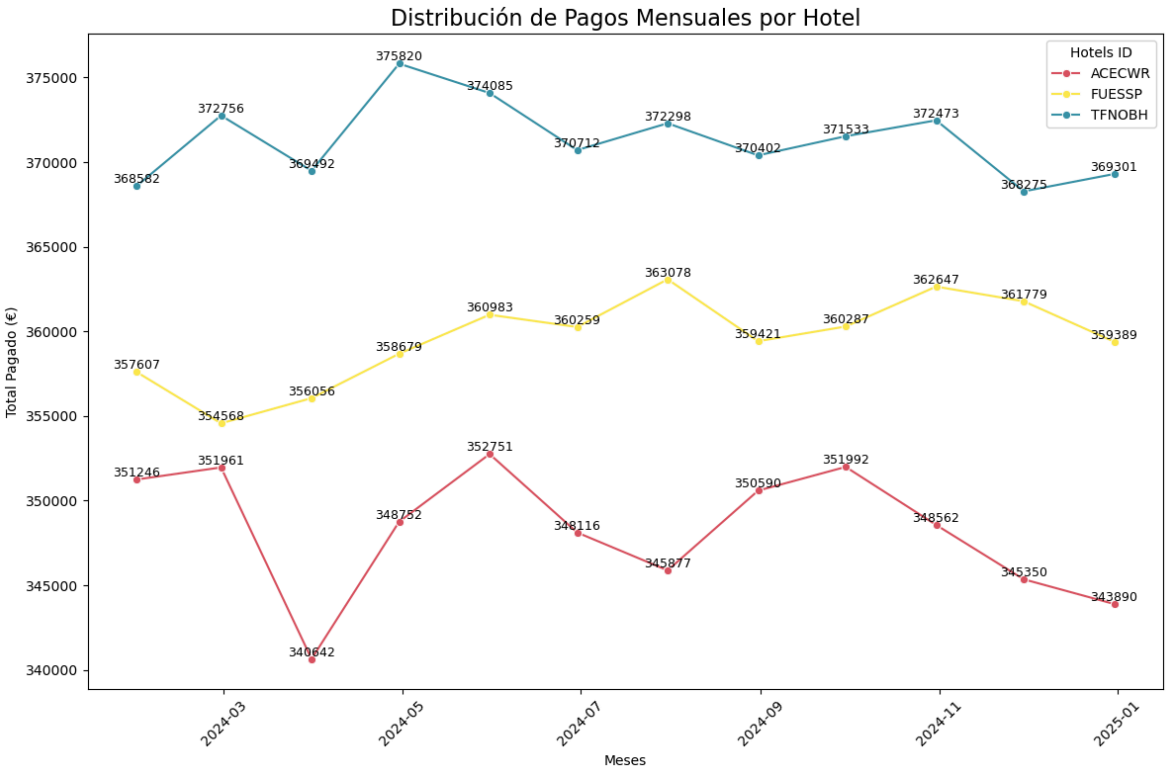
```
The total hours worked by our employees is:  882410

 Percentage Normal Hours:  96.2

 Percentage Over Time Hours:  3.8
```



4.2 ¿Cuánto pagó cada hotel en salarios a lo largo del año?

In [34]:
```
# Preparamos los datos para el gráfico
monthly_payment_by_hotel = emp_wages_wfc.groupby([pd.Grouper(key='Payment_date',
```

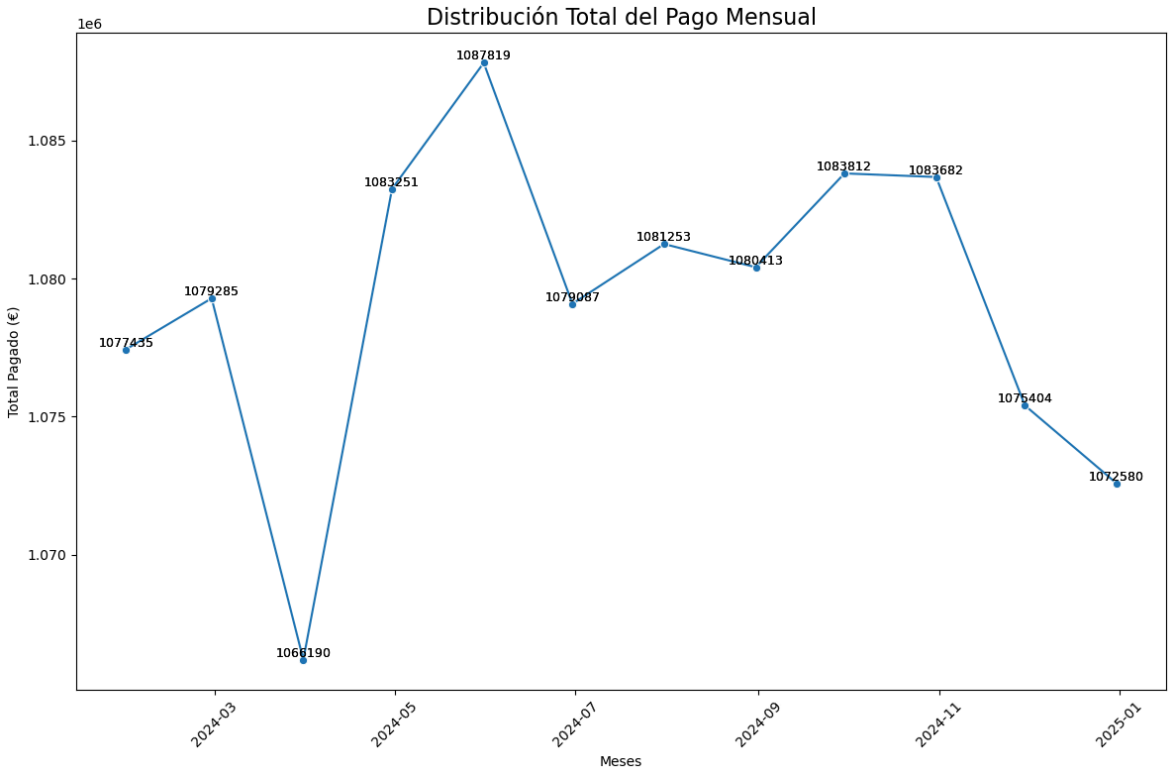
```
    'total_paid_NH': 'sum',
}).reset_index()

# Creamos un gráfico de líneas
plt.figure(figsize=(12, 8))
sns.lineplot(
    data=monthly_payment_by_hotel,
    x='Payment_date',
    y='total_paid_NH',
    hue='hotel_id_x',
    palette=['#DB5461', '#FDE74C', '#3891A6'],
    marker='o',
    )

for hotel in monthly_payment_by_hotel['hotel_id_x'].unique():
    hotel_data = monthly_payment_by_hotel[monthly_payment_by_hotel['hotel_id_x']
    for x, y in zip(hotel_data['Payment_date'], hotel_data['total_paid_NH']):
        plt.text(x, y, f'{y:.0f}', fontsize=9, ha='center', va='bottom')

plt.title('Distribución de Pagos Mensuales por Hotel', fontsize=16)
plt.xlabel('Meses')
plt.ylabel('Total Pagado (€)')
plt.xticks(rotation=45)
plt.legend(title='Hotels ID')
plt.tight_layout()

plt.show()
```



Según nuestros datos, **Ocean Breeze Haven** es el hotel que paga más en salarios. Luego se situa el hotel **Sandy Shores Park**, que cual es más consistente en sus pagos mensuales. Por último, pero no por ello menos importante, se encuentra **Coral Wave Resort**, que tiene los salarios más bajos y ha experimentado un descenso desde que comenzó el año.

Para ver el monto total de salarios pagados por los tres hoteles, unifiquemos todos los datos. Sumaremos, el total pagado por *horas normales* el total pagado por *horas extra*. Luego los analizaremos los pagos por horas separados.

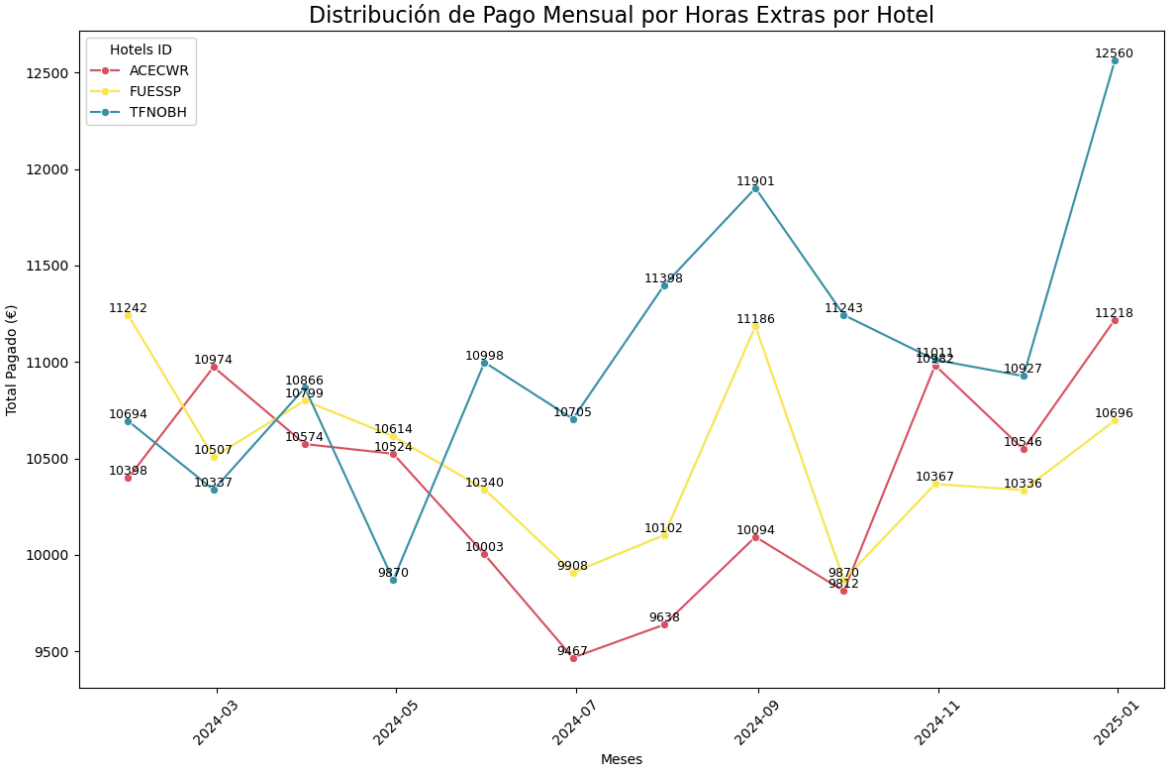
In [35]:
```
# Preparamos los datos para el gráfico
monthly_payment = emp_wages_wfc.groupby(pd.Grouper(key='Payment_date', freq='M')

# Creamos un gráfico de líneas
plt.figure(figsize=(12, 8))
sns.lineplot(
    data=monthly_payment,
    x='Payment_date',
    y='total_paid_NH',
    palette=['#3891A6'],
    marker='o',
    )

for hotel in monthly_payment:
    hotel_data = monthly_payment
    for x, y in zip(hotel_data['Payment_date'], hotel_data['total_paid_NH']):
        plt.text(x, y, f'{y:.0f}', fontsize=9, ha='center', va='bottom')

plt.title('Distribución Total del Pago Mensual', fontsize=16)
plt.xlabel('Meses')
plt.ylabel('Total Pagado (€)')
plt.xticks(rotation=45)
plt.tight_layout()

plt.show()
```



Los datos muestran que el mes de Marzo registra el valor más bajo abonado en salarios con un total de *1.066.190 €*, mientras que Mayo es el mes que registra el pago más alto con un total de *1.087.819 €*.

Veamos qué nos muestran las horas extras.

In [36]:
```
# Preparamos los datos para el gráfico de horas extras
monthly_payment_by_hotel_ot = emp_wages_wfc.groupby([pd.Grouper(key='Payment_dat
    'total_paid_OT': 'sum',
}).reset_index()

# Creamos un gráfico de líneas
plt.figure(figsize=(12, 8))
sns.lineplot(
    data=monthly_payment_by_hotel_ot,
    x='Payment_date',
    y='total_paid_OT',
    hue='hotel_id_x',
    palette=['#DB5461', '#FDE74C', '#3891A6'],
    marker='o',
    )

for hotel in monthly_payment_by_hotel_ot['hotel_id_x'].unique():
    hotel_data = monthly_payment_by_hotel_ot[monthly_payment_by_hotel_ot['hotel_
    for x, y in zip(hotel_data['Payment_date'], hotel_data['total_paid_OT']):
        plt.text(x, y, f'{y:.0f}', fontsize=9, ha='center', va='bottom')

plt.title('Distribución de Pago Mensual por Horas Extras por Hotel', fontsize=16
plt.xlabel('Meses')
plt.ylabel('Total Pagado (€)')
plt.xticks(rotation=45)
plt.legend(title='Hotels ID')
plt.tight_layout()

plt.show()
```

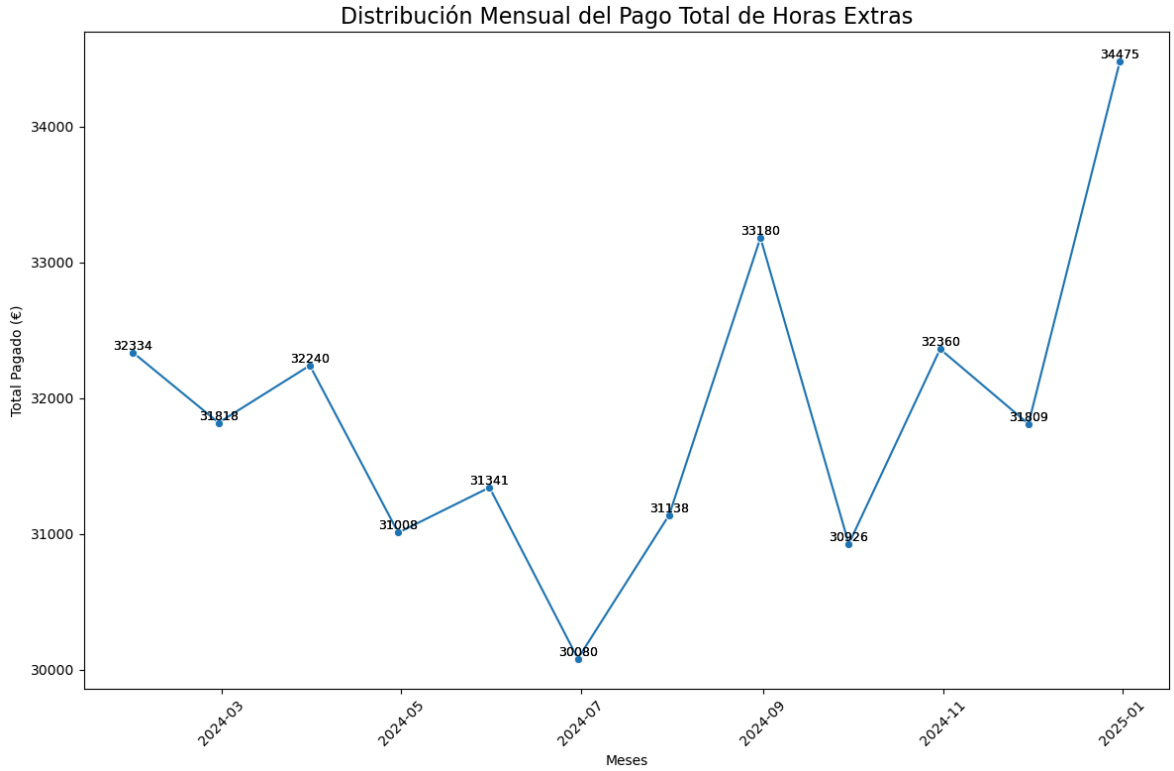


Del gráfico de líneas extraemos la siguiente conclusión: **Ocean Breeze Haven** es el hotel que más paga por horas extras, a diferencia de los otros dos hoteles. En enero, *Sandy Shores Park* registró su pago más alto, con un total de **11.242 €**. Mientras que, *Coral*

Wave Resort registró su pago más alto con un total de **12.218 €** en diciembre. Por último, pero no por ello menos importante, *Ocean Breeze Haven* muestra un incremento en el pago de horas extras en mayo, alcanzando su pico máximo en diciembre, con un total de **12.560,0 €**.

Ahora es el momento de visualizar la suma del total pagado por horas extras para los tres hoteles.

In [37]:
```
# Preparando los datos para el gráfico
monthly_payment_ot = emp_wages_wfc.groupby(pd.Grouper(key='Payment_date', freq='

# Creamos un gráfico de líneas
plt.figure(figsize=(12, 8))
sns.lineplot(
    data=monthly_payment_ot,
    x='Payment_date',
    y='total_paid_OT',
    palette=['#3891A6'],
    marker='o',
    )

for hotel in monthly_payment_ot:
    hotel_data = monthly_payment_ot
    for x, y in zip(hotel_data['Payment_date'], hotel_data['total_paid_OT']):
        plt.text(x, y, f'{y:.0f}', fontsize=9, ha='center', va='bottom')

plt.title('Distribución Mensual del Pago Total de Horas Extras', fontsize=16)
plt.xlabel('Meses')
plt.ylabel('Total Pagado (€)')
plt.xticks(rotation=45)
plt.tight_layout()

plt.show()
```



Los datos indican que el importe total pagado por horas extras disminuyó entre enero y junio, hasta alcanzar el pago mínimo de **30.080 €**. A partir de ese punto, el importe total abonado en horas extras comenzó a incrementarse, alcanzando un máximo de **34.475 €** en diciembre.

Es importante investigar si el crecimiento es estacional o se produjo por falta de personal para la operación.

Continuemos con nuestro análisis. Ahora es el momento de filtrar nuestros datos por *Departamentos*.

In [45]:
```
# Verificamos el pago total por departamento
emp_wages_wfc['Payment_month'] = emp_wages_wfc['Payment_date'].dt.strftime('%B')

month_order = ['January', 'February', 'March', 'April', 'May', 'June',
               'July', 'August', 'September', 'October', 'November', 'December']


heatmap_pivot = emp_wages_wfc.pivot_table(index='Department', columns='Payment_m
heatmap_pivot = heatmap_pivot.reindex(columns=month_order)
print(heatmap_pivot)

# Creamos un mapa de calor
plt.figure(figsize=(12, 8))
sns.color_palette("mako", as_cmap=True)
sns.heatmap(heatmap_pivot, annot=True, fmt=".0f", cbar_kws={'label': 'Total Abon

plt.title('Pago Total Mensual por Departamento', fontsize=16)
plt.ylabel('Departamento')
plt.xlabel('Meses')
plt.xticks(rotation=45)
plt.show()
```

Payment_month	January	February	March	April	\
Department					
3R	23788.0375	25218.5450	24951.7950	24210.7150	
Animation	75262.0575	76100.8650	76694.5050	76639.7000	
Bar_Restaurant	243748.3900	242044.1000	242306.9700	240518.0475	
Floors_Laundry	234890.1075	237591.0725	235397.6600	238962.1675	
Kitchen	239475.5400	241929.8825	235661.5000	238551.6150	
Other	56590.5275	56225.5650	52591.7025	57435.8825	
Reception_Reservations	82443.4525	81381.3025	82415.5650	85662.1550	
SPA	39957.4525	39382.4850	39308.5225	40429.6000	
Technical_Services	80320.3650	77896.0925	76148.3950	78421.3475	

Payment_month	May	June	July	August	\
Department					
3R	24819.8750	24660.5525	24729.6650	23719.4100	
Animation	78262.2675	76263.5825	76848.9775	77691.9075	
Bar_Restaurant	244199.9250	239861.8425	240631.7800	239864.0250	
Floors_Laundry	238797.0250	237845.6975	240466.8800	239030.5525	
Kitchen	240551.9975	241796.5075	240427.8375	241041.1200	
Other	56404.0450	53803.7175	57132.7575	55454.1725	
Reception_Reservations	82340.1475	83753.6800	82323.1725	82068.7900	
SPA	41678.9600	40275.8550	38823.7650	40522.9625	
Technical_Services	78530.9575	77630.7975	77634.9200	80792.2700	

Payment_month	September	October	November	December
Department				
3R	25311.1800	23944.4500	25051.4625	24647.7000
Animation	76688.2000	76275.2225	75959.2450	77702.8200
Bar_Restaurant	242982.3325	241844.7650	242111.2725	241070.7050
Floors_Laundry	236468.5400	237566.8225	234701.4425	235476.7150
Kitchen	243247.1425	244014.4125	240852.4550	237833.5725
Other	55472.8450	56411.8050	52620.3175	56609.6850
Reception_Reservations	82405.6225	82912.2050	82731.7850	82755.0650
SPA	40678.4050	39915.7425	40930.6050	40284.5850
Technical_Services	78041.1075	79675.5575	79038.0250	77462.7450

Pago Total Mensual por Departamento

Cuando analizamos los datos filtrados por *Departamentos* podemos visualizar que **"Bar & Restaurante", "Pisos & Lavandería"**, y **"Cocina"**, son los tres departamentos que demandan la mayor parte del presupuesto. Luego le siguen los departamentos de **"Servicios Técnicos", "Recepción & Reservas"**, y **"Animación"** con una demanda media-baja del presupuesto. Todos los demás se reparten una pequeña parte del presupuesto.

Por último, veamos el salario medio que pagan los hoteles en general y filtrado por departamentos

In [39]:
```
# Salario promedio por hotel y departamento
avg_salary = emp_wages_wfc['Total_Payment'].mean().round(2)
avg_salary_by_hotel = emp_wages_wfc.groupby('hotel_id_x')['Total_Payment'].mean(
avg_salary_by_department = emp_wages_wfc.groupby('Department')['Total_Payment'].


avg_salary_by_hotel.index = avg_salary_by_hotel.index.map(lambda x: f"{x} ({hote
avg_salary_by_hotel.index.name = None
avg_salary_by_department.index.name = None

print('El Salario Promedio Total es: €', avg_salary, '\n\n',
      'El Salario Promedio por Hotel es: ', '\n',avg_salary_by_hotel, '\n\n',
      'El Salario Promedio por Departamento es: ', '\n', avg_salary_by_departmen
```

```
El Salario Promedio Total es: € 2134.15

 El Salario Promedio por Hotel es:
 ACECWR (Coral Wave Resort)     2108.51
FUESSP (Sandy Shores Park)     2162.40
TFNOBH (Ocean Breeze Haven)    2131.51
Name: Total_Payment, dtype: float64

 El Salario Promedio por Departamento es:
 3R                          2235.25
Animation                   2130.53
Bar_Restaurant              2066.37
Floors_Laundry              2063.18
Kitchen                     2072.83
Other                       3086.82
Reception_Reservations      2236.92
SPA                         2232.36
Technical_Services          2120.70
Name: Total_Payment, dtype: float64
```

5. Carga de datos a MySQL

El tipo de datos ya está corregido y transformado. Es momento de cargar los datos en ***"hrhotelpa"*** que es como se llama nuestra base de datos en MySQL. Usaré *Python* para cargar todos los datos a su correspondiente tabla Recuerden que hemos creado las tablas usando *PopSQL*.

Para comenzar a trabajar con **MySQL**. Necesitamos crear una conexión a MySQL y luego crear un cursor para trabajar con las consultas.

In [40]:
```
# Creamos una conexion a MySQL
try:
```

```
    db = mysql.connector.connect(
        host = "localhost",
        user = "root",
        password = dbpass
    )
    print("Connection established")
    # Creamos un cursor para ejecutar las consultas
    cursor = db.cursor()

except mysql.connector.Error as err:
    print("An error occurred: ", err)
```

```
Connection established
```

In [41]:
```
# Creamos una conexión a la base de datos
hostname = "localhost"
database = "hrhotelpa"
username = "root"
password = dbpass

engine = create_engine("mysql+pymysql://{user}:{pw}@{host}/{db}".format(host=hos
```

In [42]:
```
# Añadimos los datos a las tablas
# df_rawht.to_sql('Hotels', engine, if_exists='append', index=False)
# df_rawemp.to_sql('Employees', engine, if_exists='append', index=False)
# df_rawhtcomp.to_sql('Hotel_Composition', engine, if_exists='append', index=Fal
# df_rawworkforce.to_sql('Workforce_Composition', engine, if_exists='append', in
# df_rawempwages.to_sql('Employees_Wages', engine, if_exists='append', index=Fal
```

Ahora... todos los datos se han cargado en nuestra base de datos MySQL, es hora de cerrar nuestra conexión.

In [43]:
```
cursor.close()
db.close()
```

Hotel HR Analysis
Workforce Composition

Filter by Hotels or Departments

- Coral Wave Resort
- Ocean Breeze Haven
- Sandy Shores Park

Employees Average AGE

45

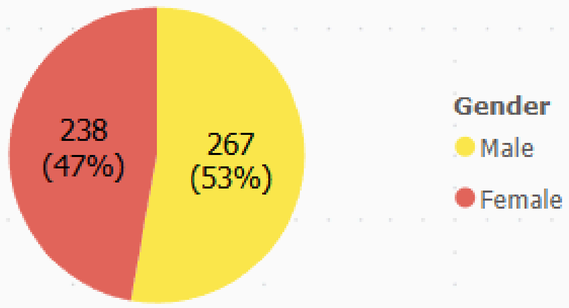
Average Working Years

12

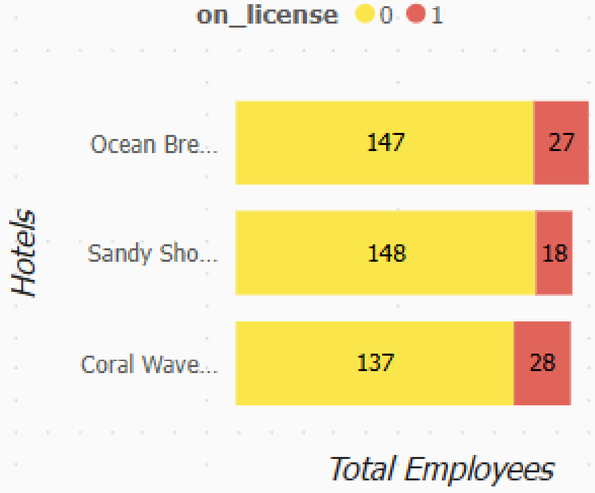
Total employees on license

73

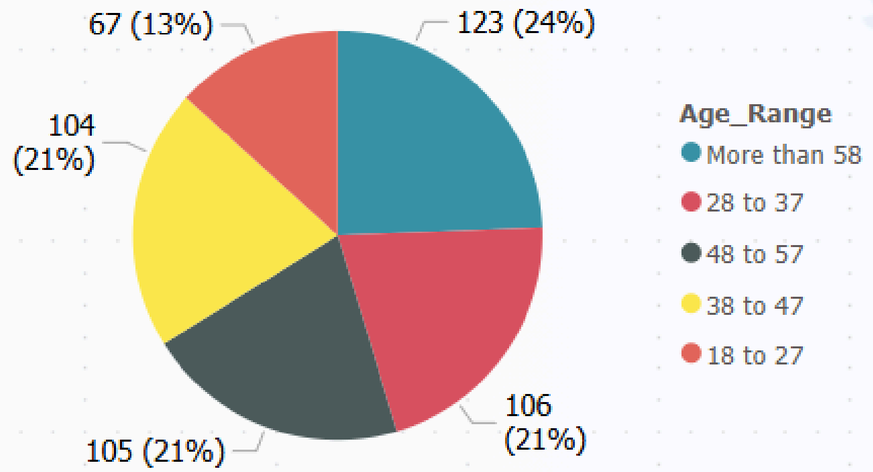
Gender Composition



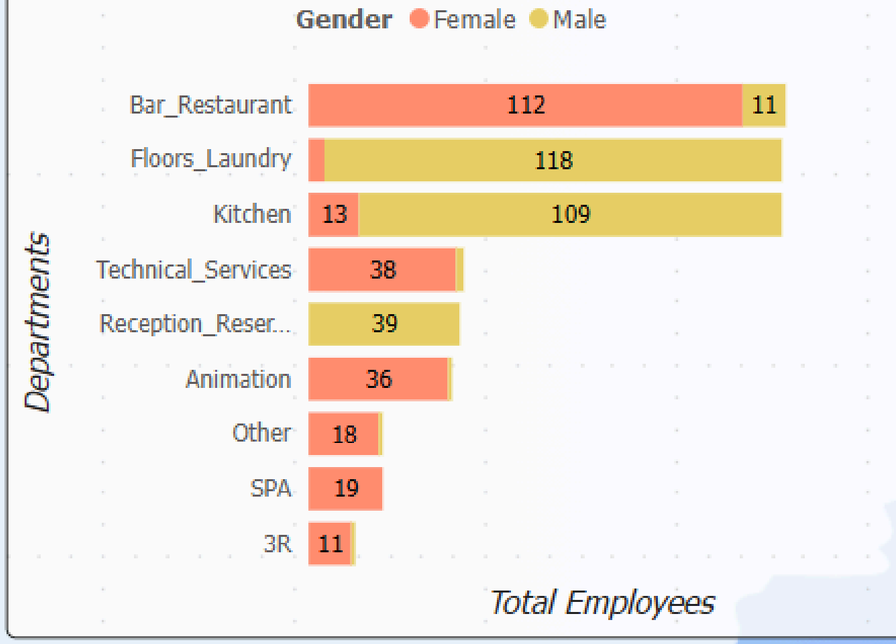
Employees On License



Composition by Age Range



Employees Distribution by Departments

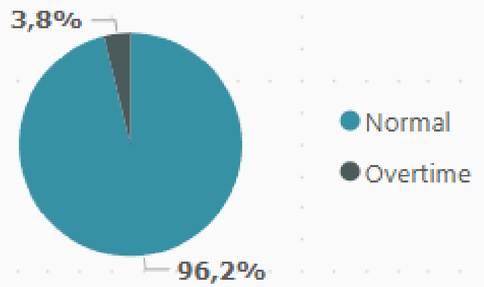


Hotel HR Analysis
Payments by hours

Filter by Hotel

Coral Wave Resort | Ocean Breeze Haven | Sandy Shores Park

Total Hours



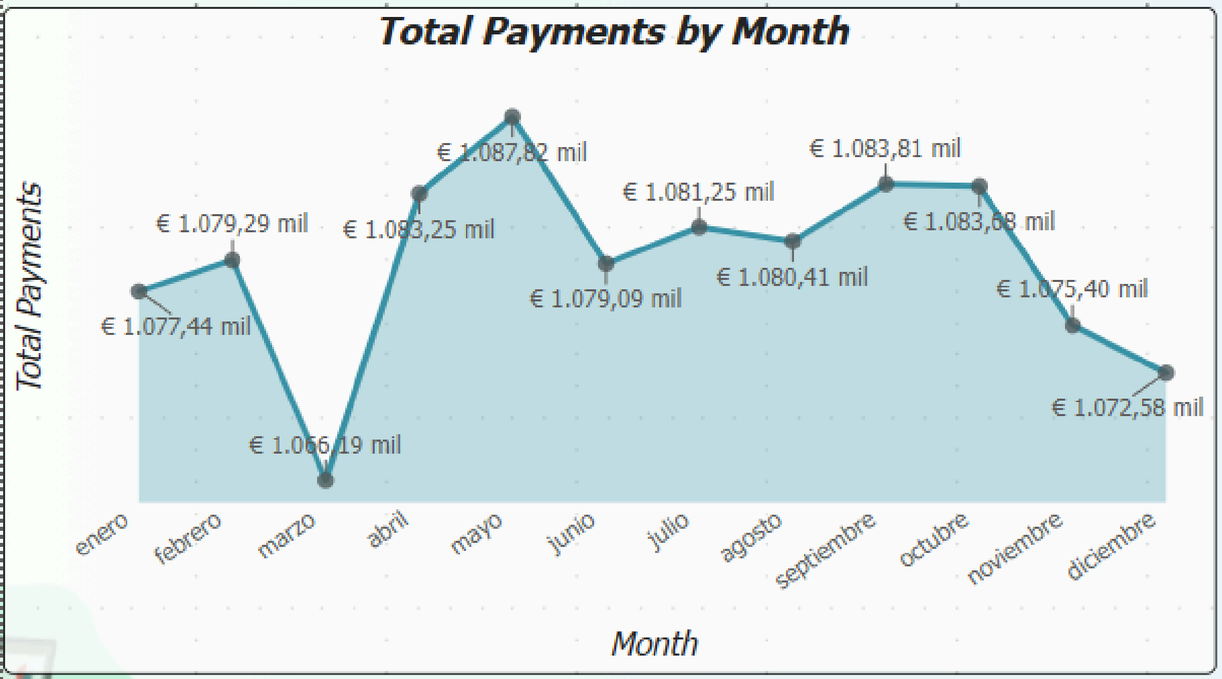
Annual goal of overtime hours

4,28%

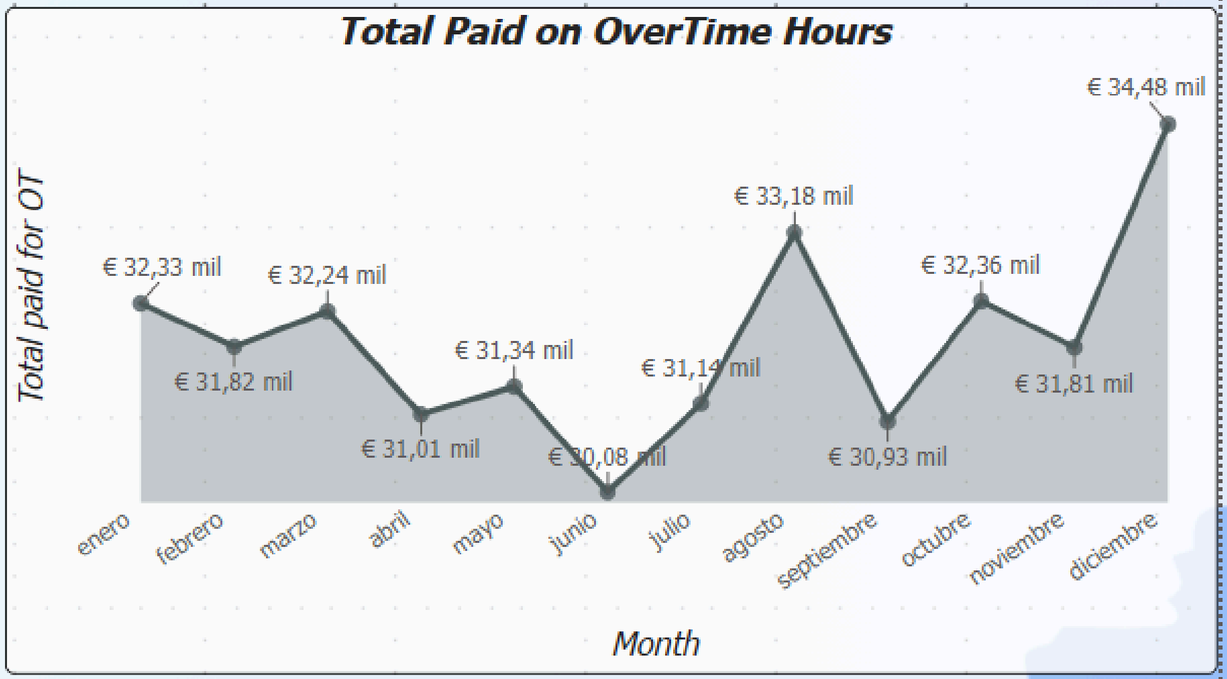
Objetivo: 0,05 (-14.34 %)

Average hours worked	Average hours Price	Total paid on normal hours	Average OT hours worked	Total Overtime hours worked	Total paid on OT hours
140	€ 15	€ 12.950.211,00	6	33501	€ 382.709,25

Total Payments by Month



Total Paid on OverTime Hours



Hotel HR Analysis Payments

Filter by Hotel

Coral Wave Resort | Ocean Breeze Haven | Sandy Shores Park

Filter by Department

Todas

Average Salary

€ 2.134,15

Trimestre	3R	Animation	Bar_Restaurant	Floors_Laundry	Kitchen	Other	Reception_Reservations	SPA	Technical_Services	Total
Qtr 1	**€ 73.958**	**€ 228.057**	**€ 728.099**	**€ 707.879**	**€ 717.067**	**€ 165.408**	**€ 246.240**	**€ 118.648**	**€ 234.365**	**€ 3.219.722**
enero	€ 23.788	€ 75.262	€ 243.748	€ 234.890	€ 239.476	€ 56.591	€ 82.443	€ 39.957	€ 80.320	**€ 1.076.476**
febrero	€ 25.219	€ 76.101	€ 242.044	€ 237.591	€ 241.930	€ 56.226	€ 81.381	€ 39.382	€ 77.896	**€ 1.077.770**
marzo	€ 24.952	€ 76.695	€ 242.307	€ 235.398	€ 235.661	€ 52.592	€ 82.416	€ 39.309	€ 76.148	**€ 1.065.477**
Qtr 2	**€ 73.691**	**€ 231.166**	**€ 724.580**	**€ 715.605**	**€ 720.900**	**€ 167.644**	**€ 251.756**	**€ 122.384**	**€ 234.583**	**€ 3.242.309**
abril	€ 24.211	€ 76.640	€ 240.518	€ 238.962	€ 238.552	€ 57.436	€ 85.662	€ 40.430	€ 78.421	**€ 1.080.831**
mayo	€ 24.820	€ 78.262	€ 244.200	€ 238.797	€ 240.552	€ 56.404	€ 82.340	€ 41.679	€ 78.531	**€ 1.085.585**
junio	€ 24.661	€ 76.264	€ 239.862	€ 237.846	€ 241.797	€ 53.804	€ 83.754	€ 40.276	€ 77.631	**€ 1.075.892**
Qtr 3	**€ 73.760**	**€ 231.229**	**€ 723.478**	**€ 715.966**	**€ 724.716**	**€ 168.060**	**€ 246.798**	**€ 120.025**	**€ 236.468**	**€ 3.240.500**
julio	€ 24.730	€ 76.849	€ 240.632	€ 240.467	€ 240.428	€ 57.133	€ 82.323	€ 38.824	€ 77.635	**€ 1.079.020**
agosto	€ 23.719	€ 77.692	€ 239.864	€ 239.031	€ 241.041	€ 55.454	€ 82.069	€ 40.523	€ 80.792	**€ 1.080.185**
septiembre	€ 25.311	€ 76.688	€ 242.982	€ 236.469	€ 243.247	€ 55.473	€ 82.406	€ 40.678	€ 78.041	**€ 1.081.295**
Qtr 4	**€ 73.644**	**€ 229.937**	**€ 725.027**	**€ 707.745**	**€ 722.700**	**€ 165.642**	**€ 248.399**	**€ 121.131**	**€ 236.176**	**€ 3.230.401**
octubre	€ 23.944	€ 76.275	€ 241.845	€ 237.567	€ 244.014	€ 56.412	€ 82.912	€ 39.916	€ 79.676	**€ 1.082.561**
noviembre	€ 25.051	€ 75.959	€ 242.111	€ 234.701	€ 240.852	€ 52.620	€ 82.732	€ 40.931	€ 79.038	**€ 1.073.997**
diciembre	€ 24.648	€ 77.703	€ 241.071	€ 235.477	€ 237.834	€ 56.610	€ 82.755	€ 40.285	€ 77.463	**€ 1.073.844**
Total	**€ 295.053**	**€ 920.389**	**€ 2.901.184**	**€ 2.847.195**	**€ 2.885.384**	**€ 666.753**	**€ 993.193**	**€ 482.189**	**€ 941.593**	**€ 12.932.933**

Hotel HR Analysis Payments