HOTEL HR PEOPLE ANALYTICS

Hotel Analysis from a People Analytics perspective.

This project will try to emulate HR analytics to continue practicing my analytical skills. Also, with this project, I want to strengthen my knowledge in ***MySql and SQL***. I will use ***Python*** and ***Power BI*** for the data analysis.

I will start with doing a little of data engineering to design the databases I am going to work with. I will use ***Figma*** to draft the databases and the relationships between each other. Then I will fill the databases with the data, and later I will start the data analysis.

1. Import libraries

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. Working with our databases

Previously, I created the databases I was going to work with. Because part of the data was difficult to create randomly, I downloaded the databases into an *.xlsx* file and worked with it.

I called the file ***hotel_hranalytics.xlsx***, so now it's time to start working with it.

2.1 Employees Table

In [2]:
```
# Let's load our databases
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 Hotels Table

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 Hotels Composition Table

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 Employees Wages Table

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 Workforce Composition Table

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 dtypes Testing

I decided to combine all the tables into one big data frame to visualize all the data types each column has instead of using the dytpes command for testing each table. This is the only purpose of the table, and it won't be used in further analyses.

In [ ]:
```
# If I needed let's put all together
df_combined = pd.concat([df_rawemp, df_rawht, df_rawhtcomp, df_rawworkforce, df_

df_combined.dtypes
```

Out[ ]:
```
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
```

After the dtypes testing, we can visualize that some columns needed to change their data type. The next step will be emphasized to correct those data types, in accordance with the ones that were established in our Figma sketch.

2.7 Fixing columns dtype

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

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

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

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

# Employees Wages Table
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
```

It was necessary to correct a few column names in order to be similar to the ones created in the *MySQL database*.

Let's proceed to check if all the data is correct and fix all minor errors that are required.

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]:
```
#First we let's check for missing values
missing_values = df_rawemp.isnull().sum()
print('Number of missing values: ', missing_values)
```

```
Number of missing values:  emp_id        0
Name          0
Surname       0
Birthday      0
Age           0
Gender        0
on_license    0
hotel_id      0
dtype: int64
```

3.1 Let *visualizations* show what the data has to told us.

Let's begin with something simple. How are our employees distributed by gender?

To answer this question we will use the *Employees* table.

In [14]:
```
# Let's work with the gender column but firs let's create a variable with the le
emp_length = len(df_rawemp)
print('The total of registries we have at the Employees table is: ', emp_length)
```

```
The total of registries we have at the Employees table is:  505
```

In [15]:
```
# Counting the values by the Gender column
emp_gender = df_rawemp['Gender'].value_counts()
print(emp_gender)
```

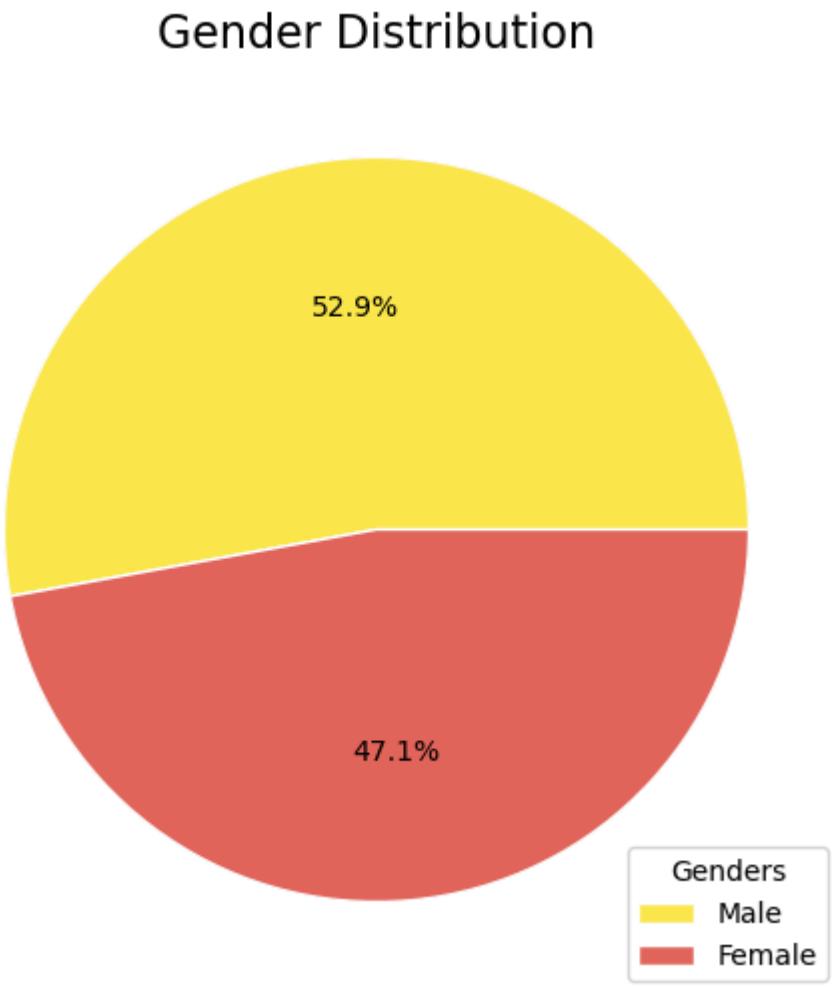
```
Gender
M    267
F    238
Name: count, dtype: int64
```

3.1.a Time to visualize the Gender data

To visualize the gender distribution I will use a **pie chart.**

In [16]:
```
# Preparing the data
colors = plt.get_cmap('Blues')(np.linspace(0.2, 0.7, len(emp_gender)))
labels = 'Male', 'Female'

# Creating the PIE CHART
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='Genders')
ax.set_title('Gender Distribution', fontsize=16)

plt.show()
```



The data shows us the distribution by gender in our three hotels. The employees are distributed, and we have a total of ***238 female employees***, representing **47,1%** of the total workforce. And for the ***males***, we have ***267 employees***, **52,9%** of the total workforce.

3.1.b Let's analyze the gender distribution by Departments and Hotels

In order to calculate the gender distribution according to the different departments, it will be necessary to combine the tables of "*Employees*" and "*Workforce Composition*". I will use the column '*emp_id*' to combine both tables.

In [17]:
```
# Preparing the data
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 [ ]:
```
# Let's drop the columns we are not going to use
emp_gender_by_dep.drop(columns=['Name', 'Surname', 'Birthday', 'wkc_id', 'years_
emp_gender_by_dep.head()
```

Out[ ]:

	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]:
```
# Checking for missing values
gender_by_dep_missing_values = emp_gender_by_dep.isnull().sum()
print('The missing values are: ', gender_by_dep_missing_values)
```

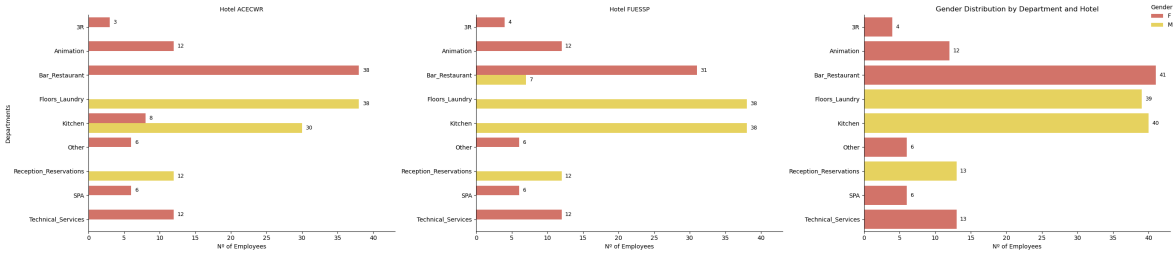
```
The missing values are:  emp_id        0
Age           0
Gender        0
on_license    0
hotel_id_x    0
Department    0
Positions     0
dtype: int64
```

It's time to visualize our data, gender distribution by *Hotel* and *Departments*

In [20]:
```
# Preparing the data
gender_dist = emp_gender_by_dep.groupby(['hotel_id_x', 'Department', 'Gender']).
gender_dist.rename(columns={0: 'Count'}, inplace=True)
print(gender_dist)

# Converting the Count column to numeric
gender_dist['Count'] = gender_dist['Count'].astype('int64')

gender_dist['Count'].dtypes

# Using Seaborn to create a barplot with FacetGrid
```

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

# Drawing the plot
grid.map_dataframe(
    sns.barplot,
    y='Department',
    x='Count',
    hue='Gender',
    palette=['#E3655B','#FDE74C']
)

# Adding the counts to each bar
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'
                )

# Legend and title
grid.add_legend(title='Gender')
grid.legend.set_loc('upper right')
grid.set_titles('Hotel {col_name}')
grid.set_axis_labels('Nº of Employees', 'Departments')
plt.tight_layout()
plt.title('Gender Distribution by Department and 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

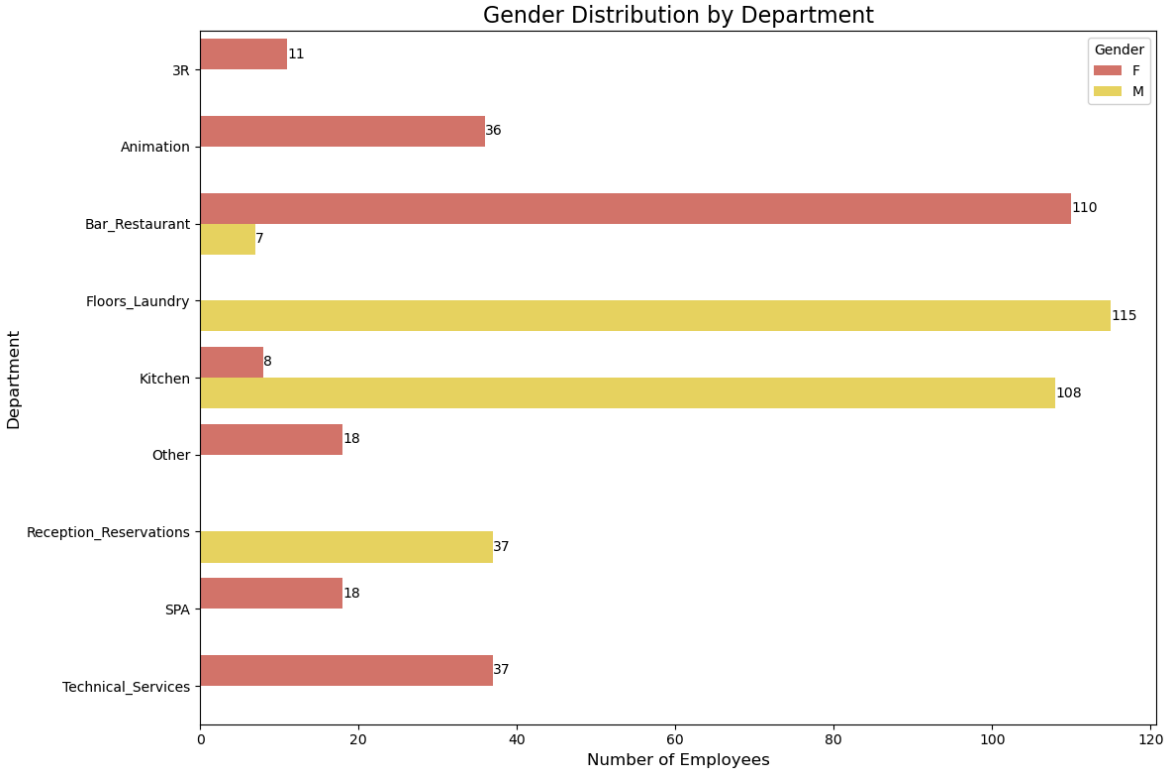


Gender distribution by *Department*

In [21]:
```
# Preparing the data
gender_by_dep = emp_gender_by_dep.groupby(['Department', 'Gender']).size().reset
print(gender_by_dep)

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

# Adding the counts to each bar
for container in ax.containers:
    ax.bar_label(container, fmt='%d', label_type='edge', fontsize=10, color='bla
```

```
# Displaying the plot
plt.title('Gender Distribution by Department', fontsize=16)
plt.xlabel('Number of Employees', fontsize=12)
plt.ylabel('Department', fontsize=12)
plt.legend(title='Gender')
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



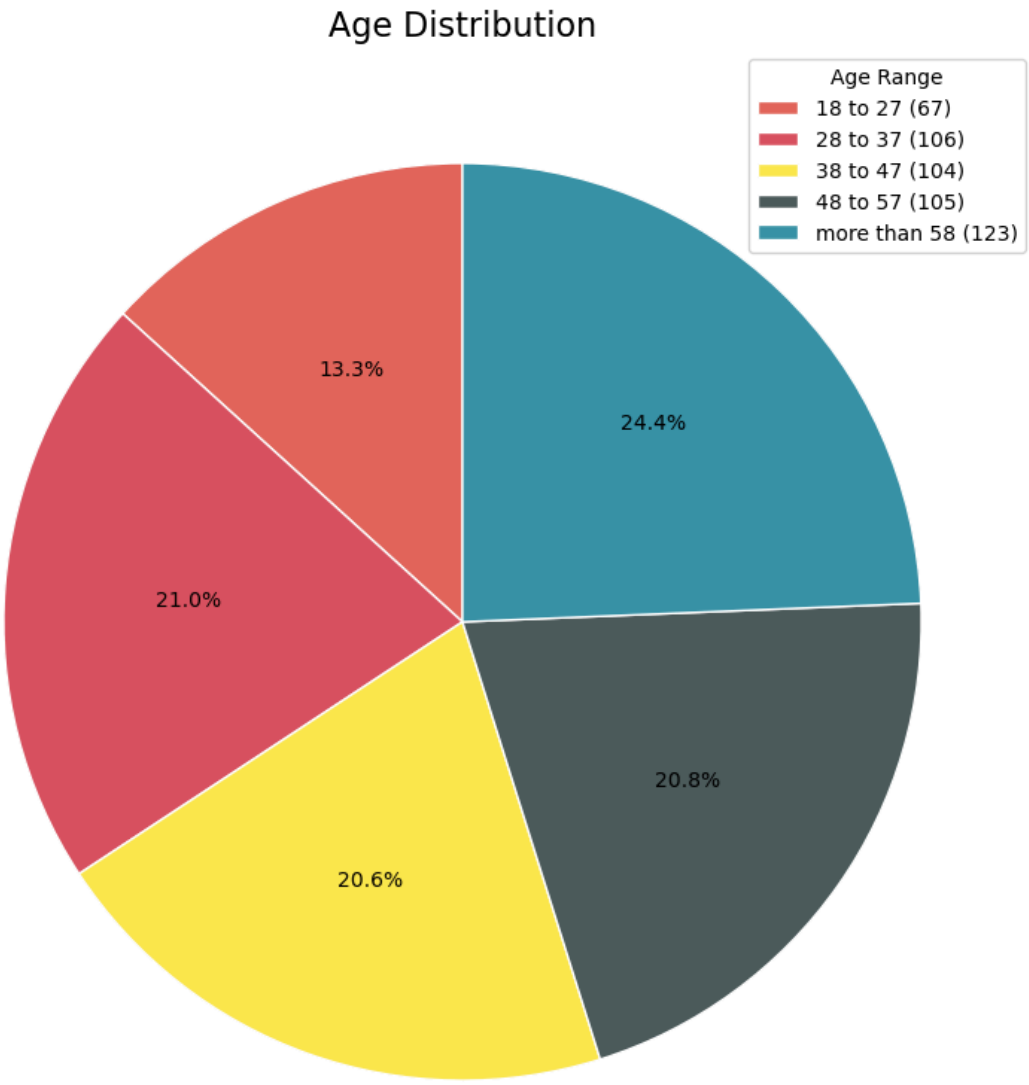
3.2 Let analyze how our employees are distributed by *Age*

I will create an age range with the purpose of facilitate the analysis

In [22]:
```
# Creating a function to distribute our employees by age range
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'

# Applying the function to the Age column
emp_age_range = df_rawemp['Age'].apply(lambda x: pd.Series(age_range(x)))

# Creating a new column with the Age Range
df_rawemp['Age_range'] = emp_age_range

# Let's check the distribution of the age range
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

# Creating the Pie Chart
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='Age Range')

ax.set_title('Age Distribution', fontsize=16)
```

Out[22]: Text(0.5, 1.0, 'Age Distribution')



After separating the employees of the three hotels within the age ranges, the following results were obtained:

Age Range

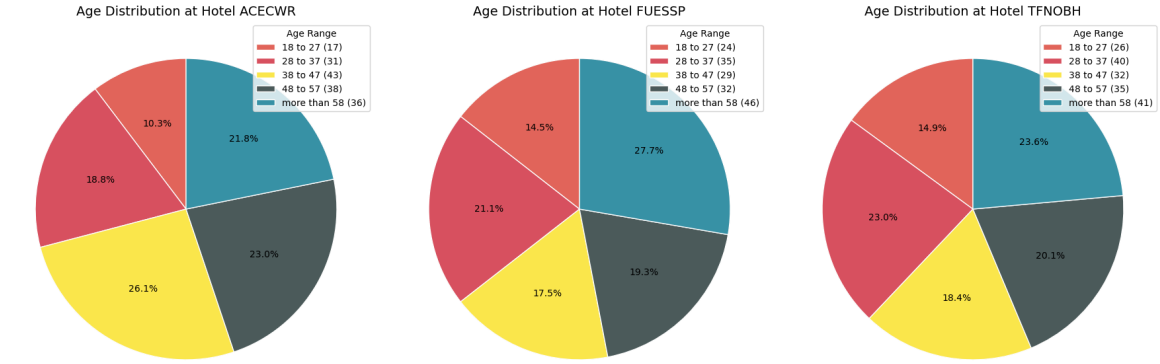
- 18 to 27 a total of **67** employees that represent a ***13,3%***
- 28 to 37 a total of **106** employees that represent a ***21%***
- 38 to 47 a total of **104** employees that represent a ***20,6%***
- 48 to 57 a total of **105** employees that represent a ***20,8%***
- More than 58 a total of **123** employees that represent a ***24,4%***

We can see that the majority of our staff members are *above* 28 after looking at the distribution of personnel by age range. 13.3% of our personnel is between the ages of 18 and 27, which indicates that *we need to start updating our workforce*. It is advised that future hiring staff concentrate on this age group.

Let's determine which hotels should begin hiring more younger employees.

In [23]:
```
# Age distribution by hotels
agerange_by_hotel = df_rawemp.groupby(['hotel_id', 'Age_range']).size().unstack(
print(agerange_by_hotel)

# Creating the Pie Charts
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'Age Distribution at Hotel {hotel}', fontsize=14)
    ax.legend(labelsbyhotel, loc='upper right', title='Age Range')

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
```



Following an analysis of the age ranges of the three hotels, we have determined that **Sandy Shores Park** has the most elderly staff, with 46 employees aged over 58. Additionally, this hotel has been in business for a shorter period of time thanthe other two hotels. While the staff at **Coral Wave Resort** is fine, it is advised to keep a watch on their group from *38 to 47*, because this is the middle of the age ranges, if they are careless they will end up having the most unbalanced workforce. The hotel with a balanced workforce is **Ocean Breeze Haven**.

Let's figure out the *average age* we have for the hotels. The *average working years* the personnel have. And last, how many employees *on license* does each hotel have?

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]:
```
# With the purpose to view the hotels names
hotel_names = df_rawht.set_index('hotel_id')['Name']

# Let's check the average age of the employees by hotel
avg_age_byhotel = df_rawemp.groupby('hotel_id')['Age'].mean().round()

# Let's check the average working years of the employees by hotel
avg_working_years = df_rawworkforce.groupby('hotel_id')['years_working'].mean().

# To print the names of the hotels in the results
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

# To eliminate the name of the column hotel_id
avg_age_byhotel.index.name = None
avg_working_years.index.name = None

print('The average age of the employees is: ','\n', avg_age_byhotel, '\n\n',
    'The average working years of the employees is: ','\n', avg_working_years, '
)
```

```
The average age of the employees is:
 ACECWR (Coral Wave Resort)     45.0
FUESSP (Sandy Shores Park)     45.0
TFNOBH (Ocean Breeze Haven)    44.0
Name: Age, dtype: float64

 The average working years of the employees is:
 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 Analyzing how many of our employees are on license

Now is time to visualize the employees that are on license. Let's figure out If we have a significant number of employees on license.

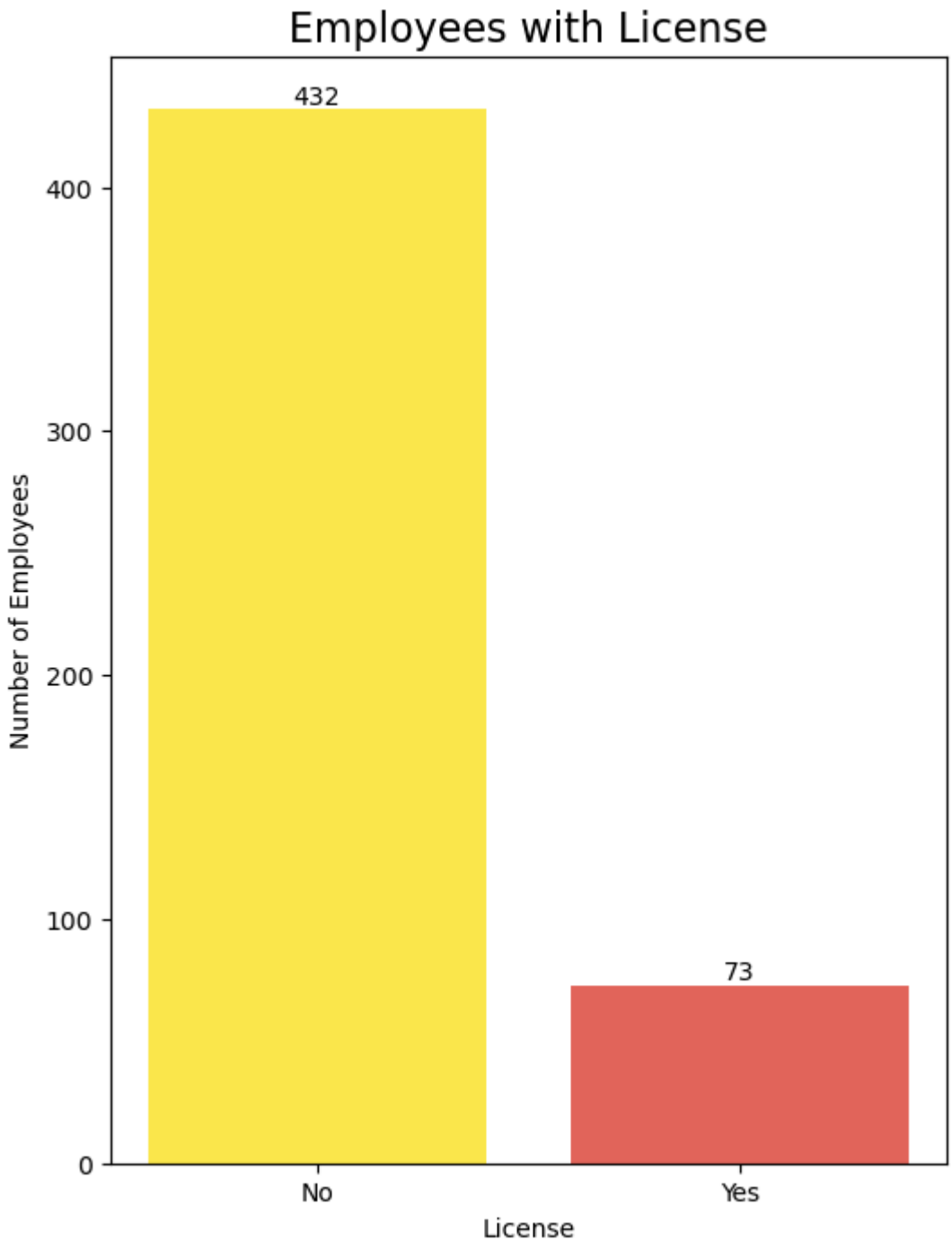
In [26]:
```
# Let's check the number of employees with license
emp_on_license = df_rawemp[(df_rawemp['on_license'] == True)].count()
emp_on_license_byhotel = df_rawemp.groupby('hotel_id')['on_license'].sum()
per_emp_on_license = (emp_on_license['on_license'] / emp_length) * 100

# To print the names of the hotels in the results
emp_on_license_byhotel.index = emp_on_license_byhotel.index.map(lambda x: f"{x}

# To eliminate the name of the column hotel_id
```

```
emp_on_license_byhotel.index.name = None

print('The number of employees with license is: ','\n', emp_on_license_byhotel,
    'The total employees with license is: ', emp_on_license['on_license'], '\n\n
    'The percentage of employees with license is: ', per_emp_on_license.round(2)
)
```

```
The number of employees with license is:
 ACECWR (Coral Wave Resort)     28
FUESSP (Sandy Shores Park)     18
TFNOBH (Ocean Breeze Haven)    27
Name: on_license, dtype: int64

 The total employees with license is:  73

 The percentage of employees with license is:  14.46 %
```

In [27]:
```
# Preparing the data to visualize the employees with license
on_license_count = df_rawemp['on_license'].value_counts()
print(on_license_count)

# Creating the barplot
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('Employees with License', fontsize=16)
plt.ylabel('Number of Employees')
plt.xlabel('License')
plt.show()
```

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



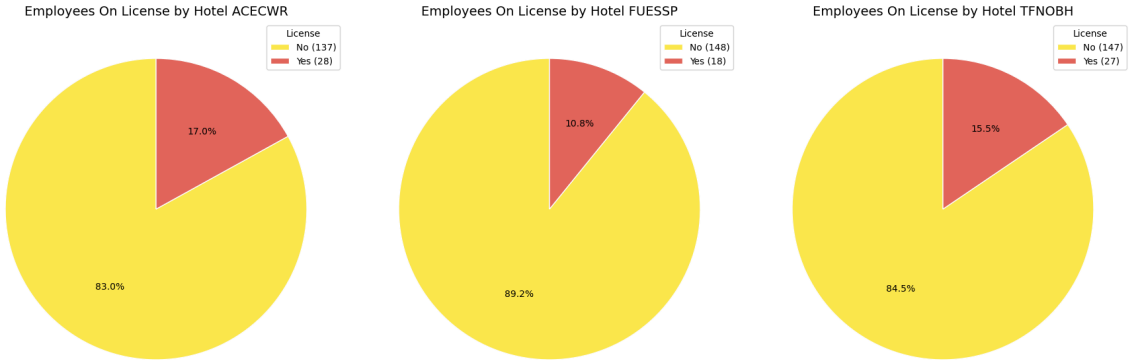
From a total of *505 employees,* only 73 are on leave by license; this represents **14,46%** of the total personnel. The company now will have to determine what percentage will be considered serious.

It's time to break down the licenses into the three hotels to visualize which hotel has the most employees on license.

In [28]:
```
# Preparing the data
onlicense_by_hotel = df_rawemp.groupby(['hotel_id', 'on_license']).size().unstac
print(onlicense_by_hotel)

# Creating the Pie Charts
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'Employees On License by Hotel {hotel}', fontsize=14)
    ax.legend(labels2, loc='upper right', title='License')

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

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



After breaking down the data, it shows that the *Coral Wave Resort*, with **28** employees, and *Ocean Breeze Haven*, with **27** employees, both are the hotels with more staff on license.

4. Analyzing the Employees Wages Table

We can now examine the annual salary payments since we know how our workforce is composed. I will start by merging the *Employees Wages* table with the *Workforce Composition* table in order to better comprehend our data. This will enable us to filter the data by Positions and Departments.

In [29]:
```
# Let's work with the employees wages table
# First let merge the workforce composition table with the employees wages table
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

With the tables combined, let's drop those columns that we will not use during our analysis.

In [30]:
```
# Droping the columns we are not going to use
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]:
```
# Checking for missing values
new_missing_values = emp_wages_wfc.isnull().sum()
print('Number of missing values: ', new_missing_values)
```

```
Number of missing values:  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 For the first analysis, let's discover the **average price** *per hour* the hotels are paying to their employees. Also, let's figure out the **average hours** the employees work. How much did the hotels pay in salaries over the year?

I will add to the analysis the **average over time hours** worked by the employees. The **total overtime hours** worked by the personnel during the entire year and how much the hotels paid for all those hours.

In [ ]:
```
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('The average  price per hour: ','€', avg_hour_price, '\n\n',
    'The average hours working by our employees is: ', avg_hours_worked, '\n\n',
    'The total SUM we paid for normal hours is: ', '€', total_paid_NH, '\n\n',
    'The average Over Time hours worked by our employees is: ', avg_OT_hours_wor
    'The total Over Time hours worked by: ', total_OT_hours, '\n\n'
    'The total SUM paid for OverTime hours: ', '€', total_paid_OT
)
```

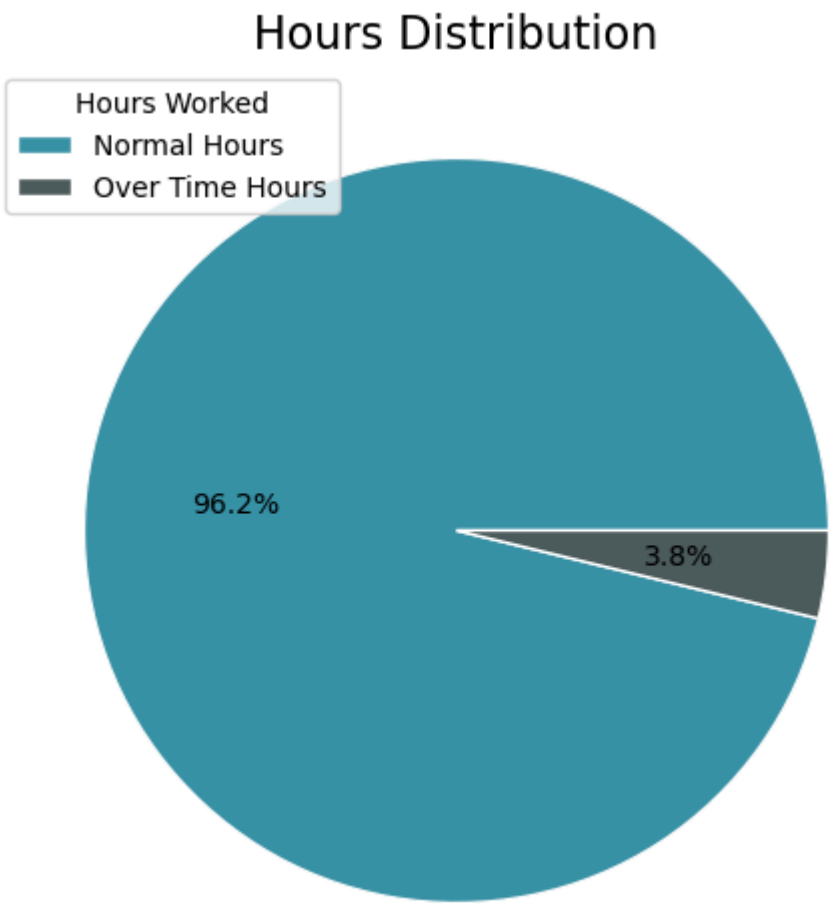
```
The average  price per hour:  € 15.0

 The average hours working by our employees is:  140.0

 The total SUM we paid for normal hours is:  € 12950211

 The average Over Time hours worked by our employees is:  6.0

 The total Over Time hours worked by:  33501

The total SUM paid for OverTime hours:  € 382709.25
```

With a *Pie chart* we will visualize how the working hours are distributed between normal and overtime.

In [33]:
```
# Let's visualize the percentage of the total hours worked by the employees
# First we need to calculate the total hours worked by the employees
```

```
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)
      )

# Now that we have the percentage we can visualize them
hours = [per_NH, per_OTh]
colors = ['#3891A6', '#4C5B5C']
labels = ['Normal Hours', 'Over Time Hours']
# 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('Hours Distribution', fontsize=16)

plt.show()
```

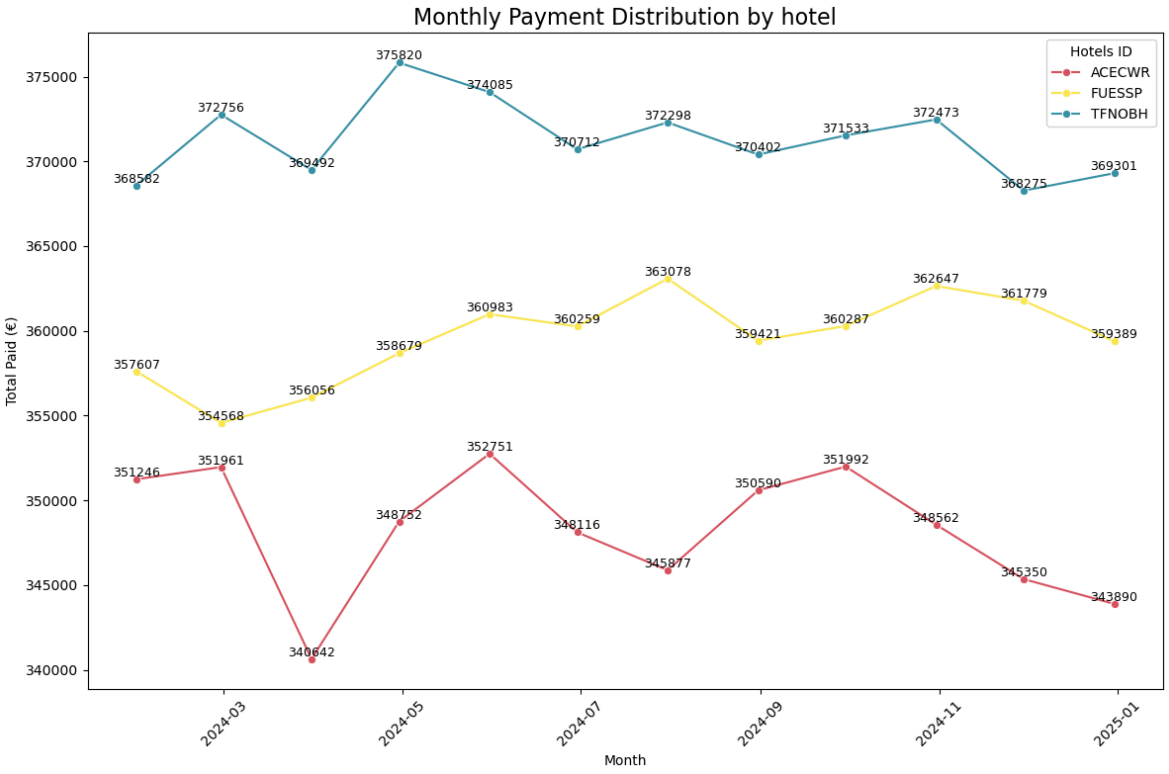
```
The total hours worked by our employees is:  882410

 Percentage Normal Hours:  96.2

 Percentage Over Time Hours:  3.8
```



4.2 How much did each hotel pay in salaries over the year?

In [34]:
```
# Collecting the data to visualize the total paid by the hotels
monthly_payment_by_hotel = emp_wages_wfc.groupby([pd.Grouper(key='Payment_date',
```

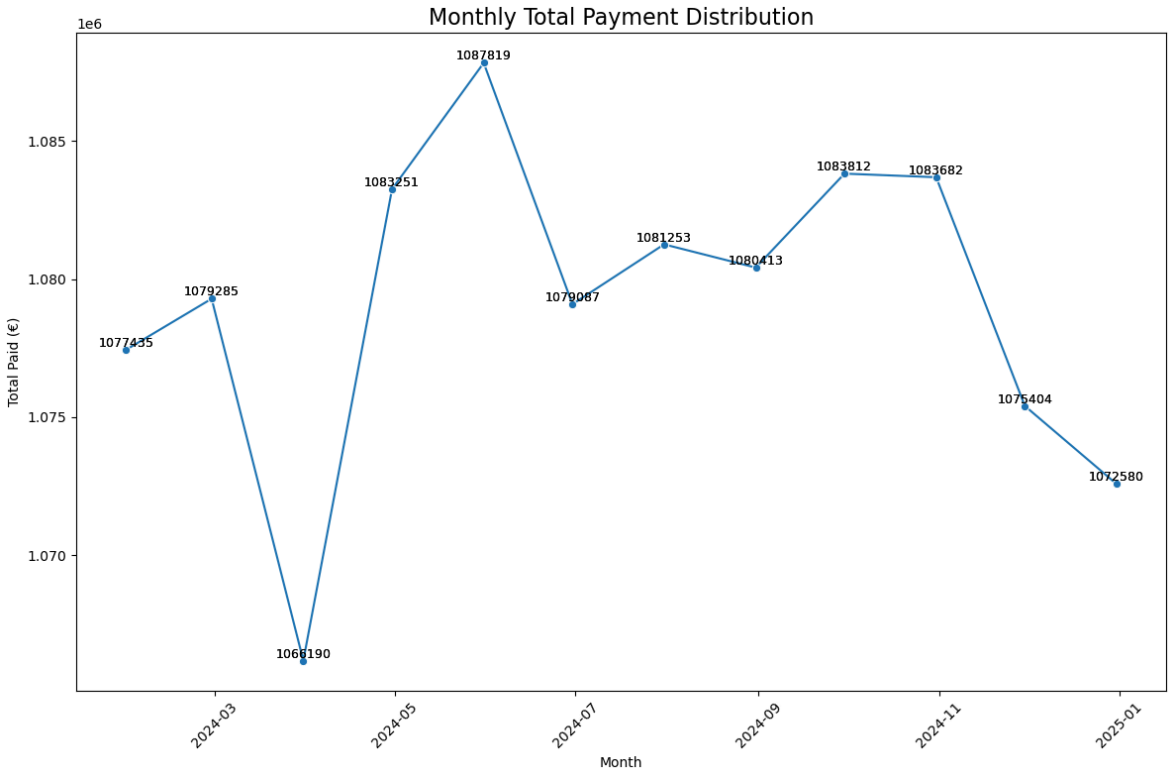
```
    'total_paid_NH': 'sum',
}).reset_index()

# Creating the lineplot
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('Monthly Payment Distribution by hotel', fontsize=16)
plt.xlabel('Month')
plt.ylabel('Total Paid (€)')
plt.xticks(rotation=45)
plt.legend(title='Hotels ID')
plt.tight_layout()

plt.show()
```



According to our data, **Ocean Breeze Haven** is the hotel that pays more in salaries. **Sandy Shores Park** comes next, and this hotel likewise pays more consistently. Last but not least is **Coral Wave Resort**, which has the lowest payments and has seen a decline since the year started.

To see the total amount of wages paid by the three hotels, let's aggregate all the data. First, the total paid for *nomal hours* and later, the total paid for *overtime hours*.

In [35]:
```
# Collecting the data to visualize the total paid by the hotels in normal hours
monthly_payment = emp_wages_wfc.groupby(pd.Grouper(key='Payment_date', freq='M')

# Creating the lineplot
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('Monthly Total Payment Distribution', fontsize=16)
plt.xlabel('Month')
plt.ylabel('Total Paid (€)')
plt.xticks(rotation=45)
plt.tight_layout()

plt.show()
```

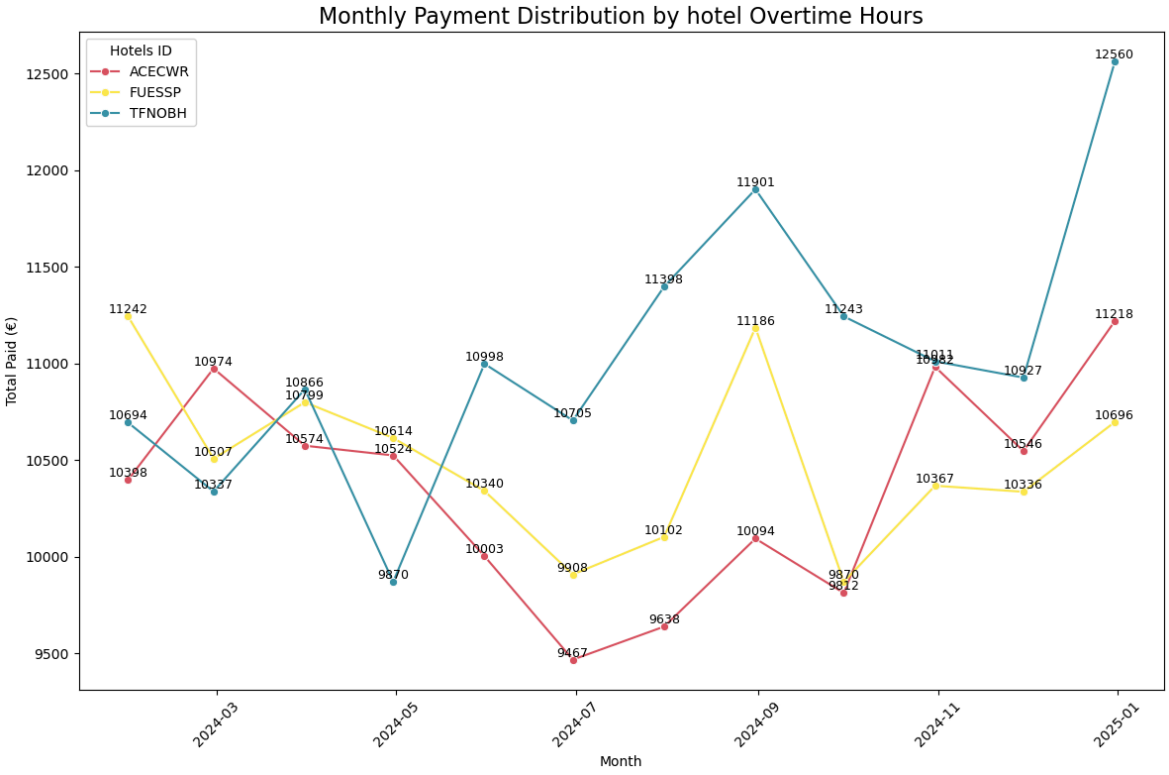


The data shows that March registered the lowest payment in salaries with a total of *€ 1.066.190* and the highest paid in salaries is May with a total *€ 1.087.819*

Let's see what the overtime hours have to show us.

In [36]:
```
# Collecting the data to visualize the total paid by the hotels in Over Time hou
monthly_payment_by_hotel_ot = emp_wages_wfc.groupby([pd.Grouper(key='Payment_dat
    'total_paid_OT': 'sum',
```

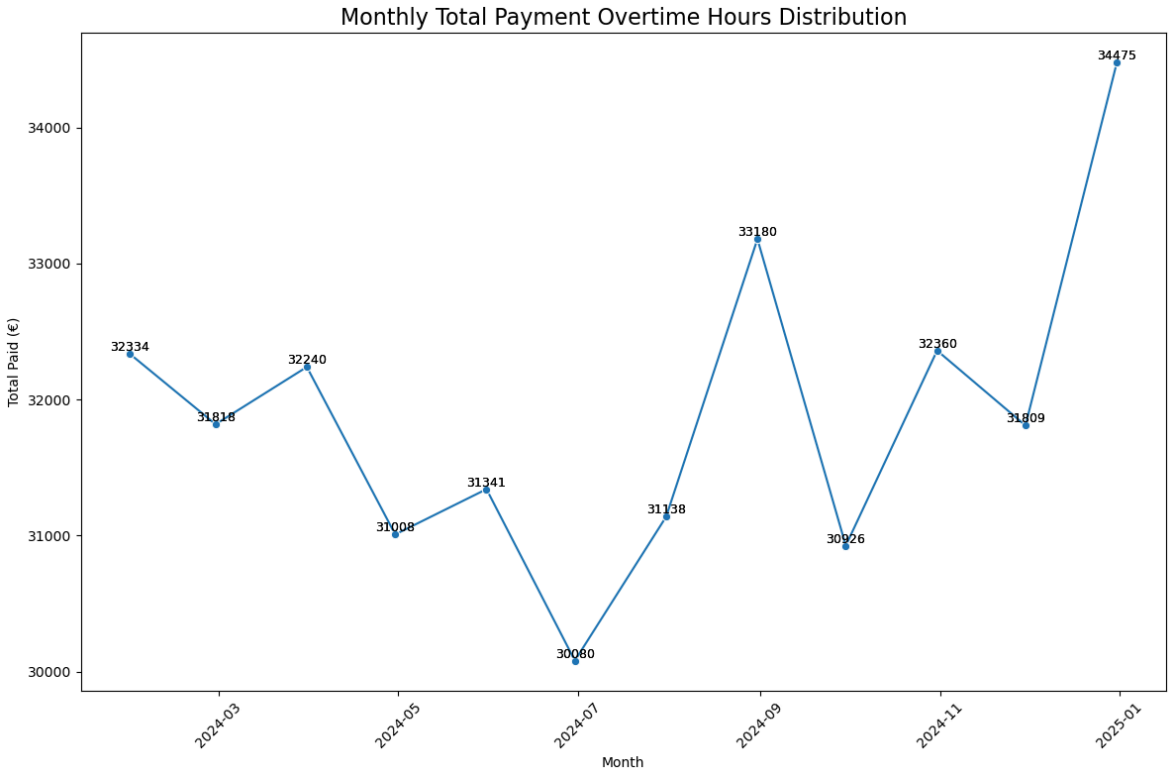
```
}).reset_index()

# Creating the lineplot
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('Monthly Payment Distribution by hotel Overtime Hours', fontsize=16)
plt.xlabel('Month')
plt.ylabel('Total Paid (€)')
plt.xticks(rotation=45)
plt.legend(title='Hotels ID')
plt.tight_layout()

plt.show()
```



The following conclusion may be drawn from the line plot: **Ocean Breeze Haven** is the hotel that paid more for overtime hours than the other two. In January, *Sandy Shores Park* recorded their highest payment, totaling **€11,242**. Meanwhile, with a total of **€12,218** in December, *Coral Wave Resort* recorded their biggest payment. Last but not least, *Ocean Breeze Haven* began to boost overtime compensation in May and reached a peak in December, totaling **€12,560.0**.

Now is time to visualize the sum of the total paid on overtime hours for the three hotels.

In [37]:
```
# Collecting the data to visualize the total paid between the hotels in Over Tim
monthly_payment_ot = emp_wages_wfc.groupby(pd.Grouper(key='Payment_date', freq='

# Creating the lineplot
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('Monthly Total Payment Overtime Hours Distribution', fontsize=16)
plt.xlabel('Month')
plt.ylabel('Total Paid (€)')
plt.xticks(rotation=45)
plt.tight_layout()

plt.show()
```



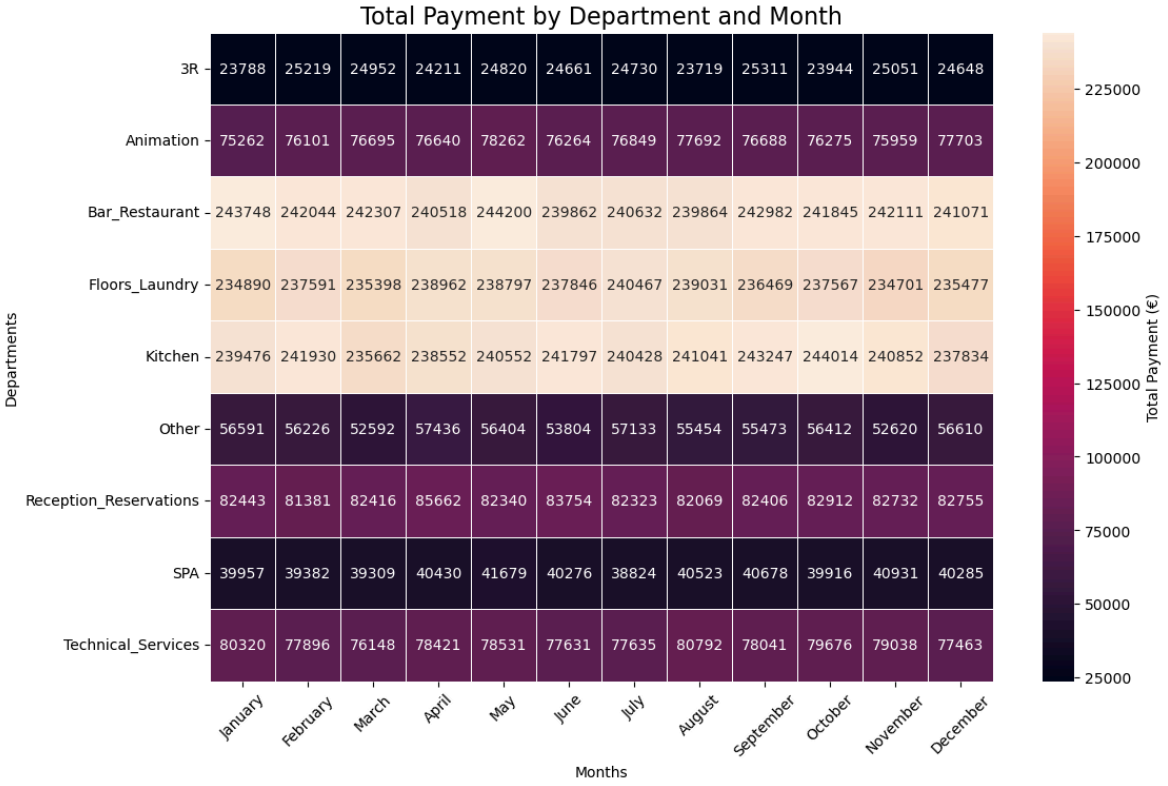
The data indicates that the total amount paid for extra hours decreased between January and June, reaching the minimum payout of **€30,080**. From then on, the total amount of overtime payments began to rise, peaking at **€34,475** in December.

It is important to investigate whether the growth is seasonal or occurred due to a lack of personnel for the operation.

Let's continue with our analysis. Now is time to filter our data by *Departments*.

In [38]:
```
# Let's check the total payment by department and month
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)

# Creating the heatmap
plt.figure(figsize=(12, 8))
sns.color_palette("mako", as_cmap=True)
sns.heatmap(heatmap_pivot, annot=True, fmt=".0f", cbar_kws={'label': 'Total Paym

plt.title('Total Payment by Department and Month', fontsize=16)
plt.ylabel('Departments')
plt.xlabel('Months')
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
```



When we analyze the data filtered by *Departments* we can visualize that **"Bar & Restaurant", "Floors & Laundry"**, and **"Kitchen"**, are the three departments that demand the largest part of the budget. Then **"Technical Services", "Reception & Reservations"**, and **"Animation"** are the following departments with a middle-low demand of the budget. All the rest share a small part of the budget.

For the last, let's figure out the average salary that the hotels are paying in general and filtered by departments

In [39]:
```
# Average Salary
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('The Total Average Salary is: €', avg_salary, '\n\n',
      'The Average Salary by Hotel is: ', '\n',avg_salary_by_hotel, '\n\n',
      'The Average Salary by Department is: ', '\n', avg_salary_by_department)
```

```
The Total Average Salary is: € 2134.15

 The Average Salary by Hotel is:
 ACECWR (Coral Wave Resort)     2108.51
FUESSP (Sandy Shores Park)     2162.40
TFNOBH (Ocean Breeze Haven)    2131.51
Name: Total_Payment, dtype: float64

 The Average Salary by Department is:
 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. Loading the data to MySQL

The data type is now corrected and transformed. It's time to upload the data into ***"hrhotelpa"*** that is what our database is called in MySQL. I will use *Python* to upload all the data to their corresponding table. Remember we have created the tables using *PopSQL*.

Let's start working with **MySQL**. We are required to connect to MySQL and later create a cursor to work with the queries.

In [40]:
```python
# Let's create the connection to MySQL
try:

    db = mysql.connector.connect(
        host = "localhost",
        user = "root",
        password = dbpass
    )
    print("Connection established")
    # Creating the cursor to execute queries
    cursor = db.cursor()

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

```
Connection established
```

In [41]:
```python
# Creating a enginge connection
hostname = "localhost"
database = "hrhotelpa"
username = "root"
password = dbpass

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

In [42]:
```python
# Add the databases to MySQL
# 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
```

Now... that all the data have been uploaded to our MySQL database, it's time to close our connection.

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

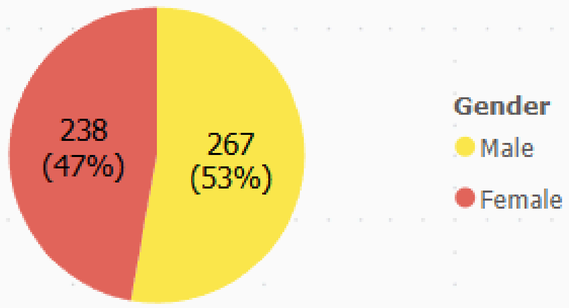
Average Working Years

12

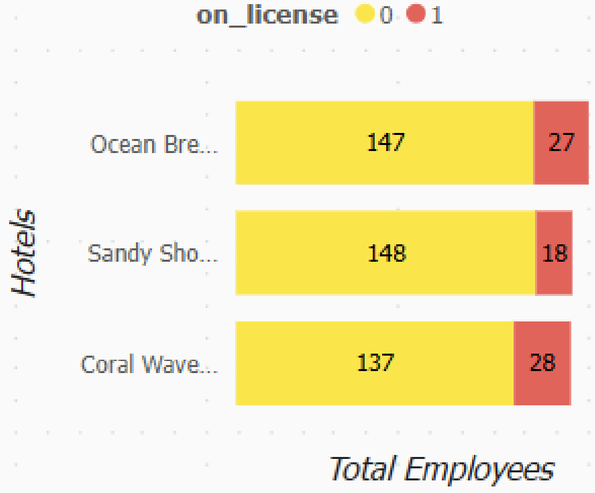
Total employees on license

73

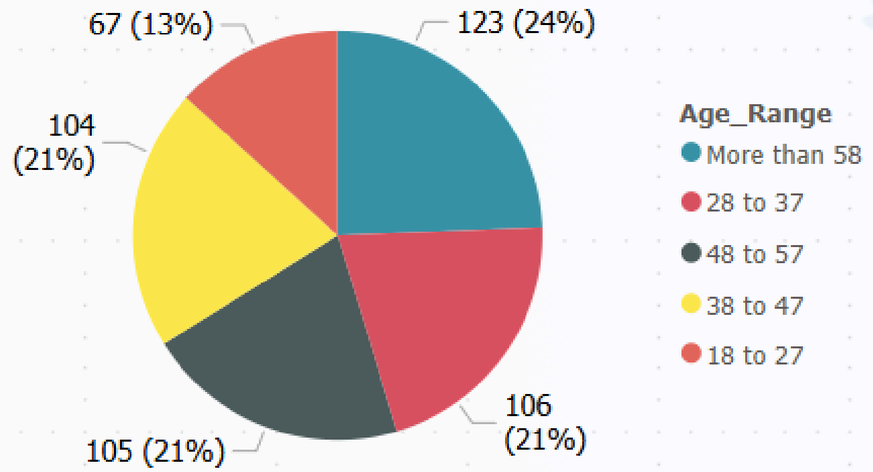
Gender Composition



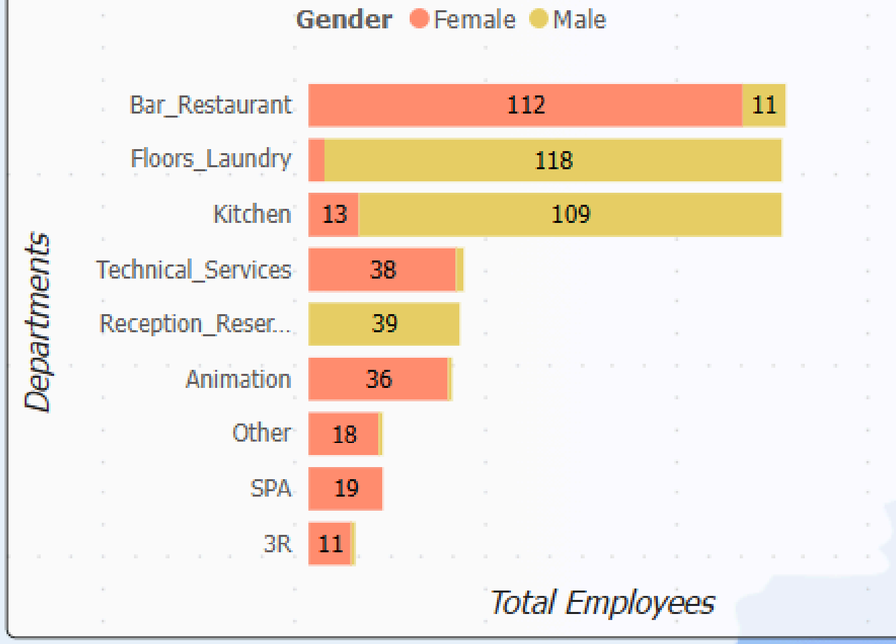
Employees On License



Composition by Age Range



Employees Distribution by Departments

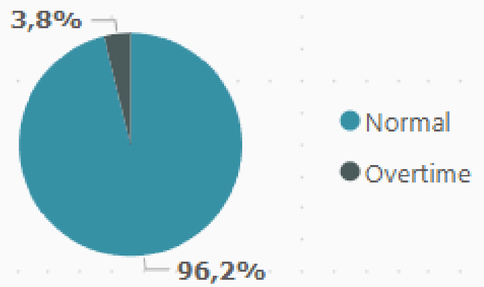


Hotel HR Analysis
Payments by hours

Filter by Hotel

Coral Wave Resort | Ocean Breeze Haven | Sandy Shores Park

Total Hours



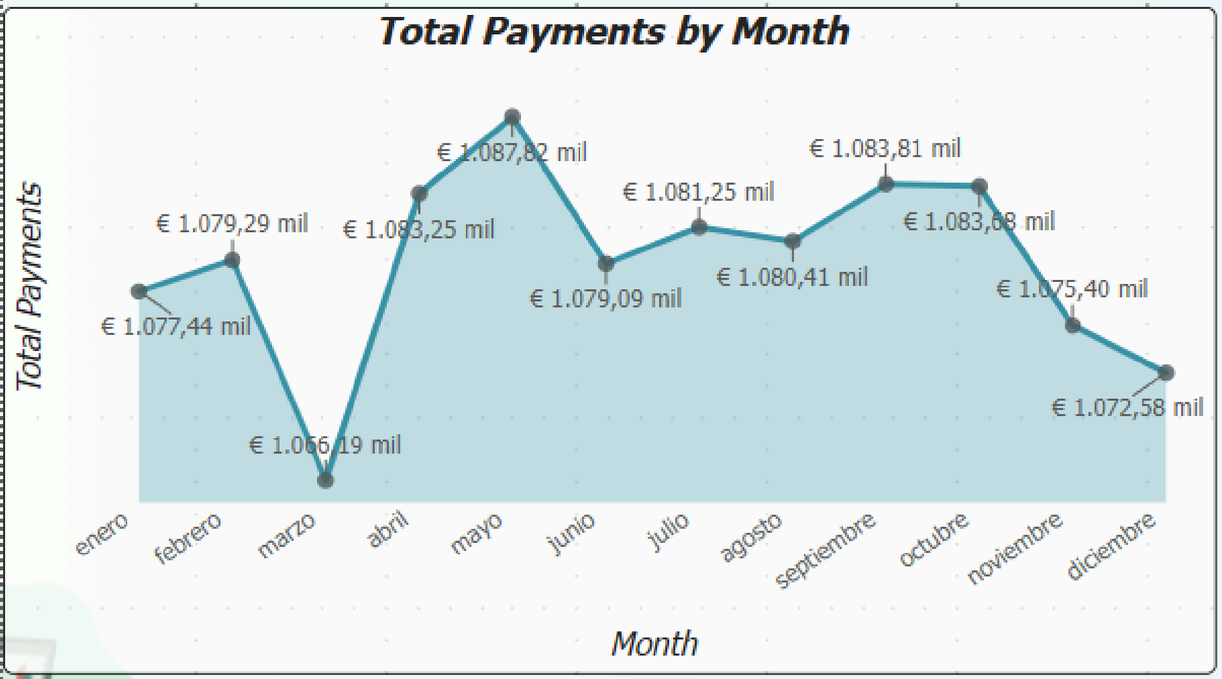
Annual goal of overtime hours

4,28%

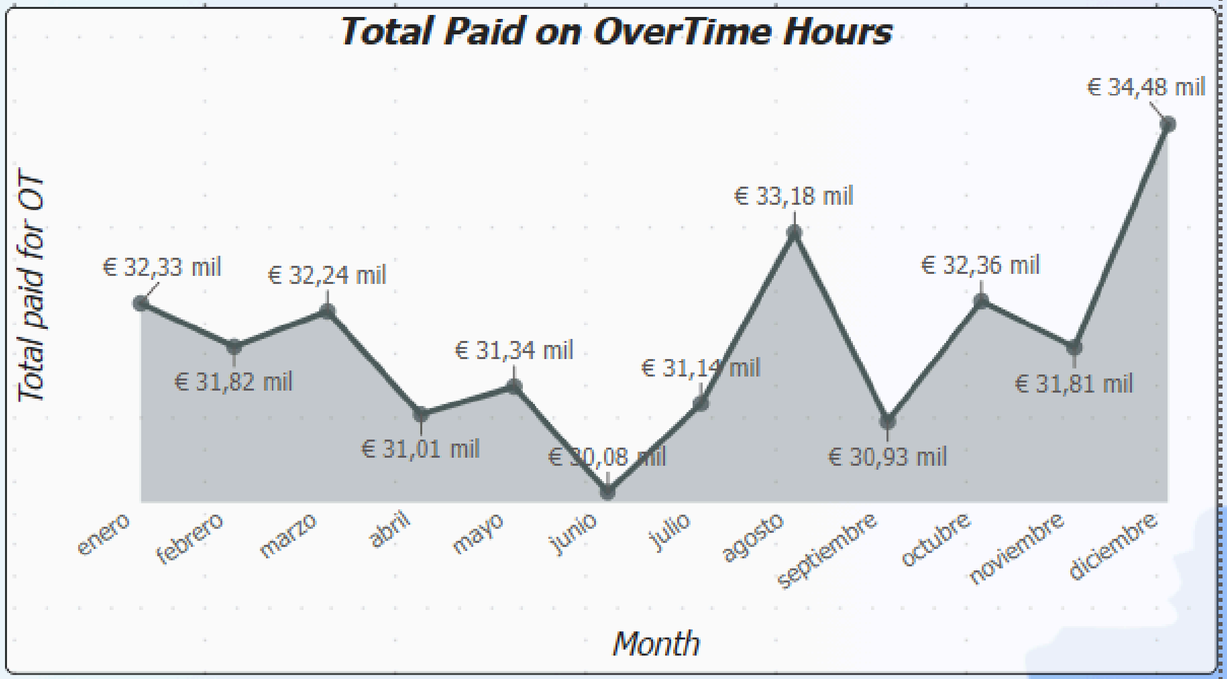
Objetivo: 0,05 (-14.34 %)

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