ANÁLISIS DE RESERVAS DE HOTEL

Analisis de reservas de hoteles.

Los datos son de la web de Kaggle:

Usuario: *MOJTABA*

Título: *Hotel Booking - Hotel booking demand datasets(Data in Brief:2019)*

Enlace: https://www.kaggle.com/datasets/mojtaba142/hotel-booking

Para este proyecto analizaré reservas de hoteles para seguir practicando mis habilidades analíticas utilizando mis conocimientos de ***Python and Power BI***. Siempre que sea posible, se aplicaré técnicas de *Machine Learning*.

Trabajaré con el siguiente archivo:

- ***hotel_booking_mojtaba.csv***

1. Importando las librerías

In [ ]:
```
# Librerías para manipular los datos
import pandas as pd
import numpy as np

# Librerias para crear los gráficos
import seaborn as sns
import matplotlib.pyplot as plt

# Modelos de predicciones
import statsmodels.api as sm
import statsmodels.formula.api as smf

# Para ignorar alertas
import warnings
warnings.filterwarnings('ignore')
```

2. Importanto los datos

Importemos el archivo ***'hotel_booking_mojtaba.csv'*** para iniciar el análisis. Es un archivo .csv. Primero, limpiaré y prepararé los datos para el análisis. Luego vamos a trabajar con ellos para ayudar a los interesados a tomar mejores decisiones basadas en datos.

In [ ]:
```
df_rawdata = pd.read_csv('../hotel_bookings/csv_files/hotel_booking_mojtaba.csv')
df_rawdata.head(5)
```

Out[ ]:

	hotel	is_canceled	lead_time	arrival_date_year	arrival_date_month	arrival_date_week_number
0	Resort Hotel	0	342	2015	July	27
1	Resort Hotel	0	737	2015	July	27
2	Resort Hotel	0	7	2015	July	27
3	Resort Hotel	0	13	2015	July	27
4	Resort Hotel	0	14	2015	July	27

5 rows × 36 columns

2.1 Estructura de los datos

Ahora es el momento de ver cómo están compuestos los datos, comprobar si faltan valores y seleccionar aquellos datos con los que vamos a trabajar. Los nuestra base de datos se encuentra en la variable 'df_rawdata'. Una vez que los datos estén listos para el análisis, cambiaré el nombre de nuestra base. El motivo es tener un backup en caso de que se eliminen datos por error.

In [ ]:
```
# Verificamos las columnas y tipos de datos que poseen
df_rawdata.info()
```

```
<class 'pandas.core.frame.DataFrame'>
RangeIndex: 119390 entries, 0 to 119389
Data columns (total 36 columns):
 #   Column                          Non-Null Count   Dtype  
---  ------                          --------------   -----  
 0   hotel                           119390 non-null  object 
 1   is_canceled                     119390 non-null  int64  
 2   lead_time                       119390 non-null  int64  
 3   arrival_date_year               119390 non-null  int64  
 4   arrival_date_month              119390 non-null  object 
 5   arrival_date_week_number        119390 non-null  int64  
 6   arrival_date_day_of_month       119390 non-null  int64  
 7   stays_in_weekend_nights         119390 non-null  int64  
 8   stays_in_week_nights            119390 non-null  int64  
 9   adults                          119390 non-null  int64  
 10  children                        119386 non-null  float64
 11  babies                          119390 non-null  int64  
 12  meal                            119390 non-null  object 
 13  country                         118902 non-null  object 
 14  market_segment                  119390 non-null  object 
 15  distribution_channel            119390 non-null  object 
 16  is_repeated_guest               119390 non-null  int64  
 17  previous_cancellations          119390 non-null  int64  
 18  previous_bookings_not_canceled  119390 non-null  int64  
 19  reserved_room_type              119390 non-null  object 
 20  assigned_room_type              119390 non-null  object 
 21  booking_changes                 119390 non-null  int64  
 22  deposit_type                    119390 non-null  object 
 23  agent                           103050 non-null  float64
 24  company                         6797 non-null    float64
 25  days_in_waiting_list            119390 non-null  int64  
 26  customer_type                   119390 non-null  object 
 27  adr                             119390 non-null  float64
 28  required_car_parking_spaces     119390 non-null  int64  
 29  total_of_special_requests       119390 non-null  int64  
 30  reservation_status              119390 non-null  object 
 31  reservation_status_date         119390 non-null  object 
 32  name                            119390 non-null  object 
 33  email                           119390 non-null  object 
 34  phone-number                    119390 non-null  object 
 35  credit_card                     119390 non-null  object 
dtypes: float64(4), int64(16), object(16)
memory usage: 32.8+ MB
```

3 Limpieza de los datos

Eliminaré aquellas columnas ya que no son necesarias para el análisis, entre ellas, las columnas que almacenan los datos de los clientes.

Modificaré el nombre de la base de datos por el de: '*df_hb*'.

In [ ]:
```
# Realizando una copia de seguridad de nuestros datos
df_hb = df_rawdata.copy()
df_hb.columns
```

Out[ ]:
```
Index(['hotel', 'is_canceled', 'lead_time', 'arrival_date_year',
       'arrival_date_month', 'arrival_date_week_number',
       'arrival_date_day_of_month', 'stays_in_weekend_nights',
       'stays_in_week_nights', 'adults', 'children', 'babies', 'meal',
       'country', 'market_segment', 'distribution_channel',
       'is_repeated_guest', 'previous_cancellations',
       'previous_bookings_not_canceled', 'reserved_room_type',
       'assigned_room_type', 'booking_changes', 'deposit_type', 'agent',
       'company', 'days_in_waiting_list', 'customer_type', 'adr',
       'required_car_parking_spaces', 'total_of_special_requests',
       'reservation_status', 'reservation_status_date', 'name', 'email',
       'phone-number', 'credit_card'],
      dtype='object')
```

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

```
Number of missing values:  hotel                           0
is_canceled                     0
lead_time                       0
arrival_date_year               0
arrival_date_month              0
arrival_date_week_number        0
arrival_date_day_of_month       0
stays_in_weekend_nights         0
stays_in_week_nights            0
adults                          0
children                        4
babies                          0
meal                            0
country                       488
market_segment                  0
distribution_channel            0
is_repeated_guest               0
previous_cancellations          0
previous_bookings_not_canceled  0
reserved_room_type              0
assigned_room_type              0
booking_changes                 0
deposit_type                    0
agent                       16340
company                    112593
days_in_waiting_list            0
customer_type                   0
adr                             0
required_car_parking_spaces     0
total_of_special_requests       0
reservation_status              0
reservation_status_date         0
name                            0
email                           0
phone-number                    0
credit_card                     0
dtype: int64
```

3.1 Observaciones

Descubrí que las columnas *'country', 'agent', and 'company'* son las que tienen la mayor cantindad de datos faltantes. Recomiendo para un futuro completar en lo mayor posible los datos de la columna *'agent'* con el fin de conocer los agentes que proporcionan la mayor cantidad de clientes. Esto permitirá crear ofertas para cada uno de ellos.

Como poseo interés en trabajar con la columna *'country'*, voy a reemplazar los valores faltantes por el valor con código **'OTR'**. El propósito es saber los países de los que provienen nuestros clientes.

Eliminaré los cuatro valores faltantes de la columna *'children'*; eliminarlos no afecta a nuestro análisis.

In [ ]:
```
# Reemplazando os valores faltantes en la columna 'country'
df_hb['country'].fillna('OTR', inplace=True)

# Eliminando las columnas que NO se utilizarán en nuestro análisis
df_hb.drop(['previous_cancellations', 'previous_bookings_not_canceled', 'days_in_wa

# Verificando que poseemos las columnas necesarias para nuestro análisis
df_hb.columns
```

Out[ ]:
```
Index(['hotel', 'is_canceled', 'lead_time', 'arrival_date_year',
       'arrival_date_month', 'arrival_date_week_number',
       'arrival_date_day_of_month', 'stays_in_weekend_nights',
       'stays_in_week_nights', 'adults', 'children', 'babies', 'meal',
       'country', 'market_segment', 'distribution_channel',
       'is_repeated_guest', 'reserved_room_type', 'assigned_room_type',
       'booking_changes', 'deposit_type', 'customer_type', 'adr',
       'required_car_parking_spaces', 'total_of_special_requests',
       'reservation_status', 'reservation_status_date'],
      dtype='object')
```

In [ ]:
```
# Removiendo los valores de la columna 'children'
df_hb.dropna(subset=['children'], inplace=True)

# Comprobando que no tenemos más valores faltantes
missing_values2 = df_hb.isnull().sum()
print('Number of missing values: ', missing_values2)
```

```
Number of missing values:  hotel                           0
is_canceled                     0
lead_time                       0
arrival_date_year               0
arrival_date_month              0
arrival_date_week_number        0
arrival_date_day_of_month       0
stays_in_weekend_nights         0
stays_in_week_nights            0
adults                          0
children                        0
babies                          0
meal                            0
country                         0
market_segment                  0
distribution_channel            0
is_repeated_guest               0
reserved_room_type              0
assigned_room_type              0
booking_changes                 0
deposit_type                    0
customer_type                   0
adr                             0
required_car_parking_spaces     0
total_of_special_requests       0
reservation_status              0
reservation_status_date         0
dtype: int64
```

In [ ]:
```
# Vereficando la cantidad de filas que posee nuestra base
df_hb_lens = len(df_hb)
print('El número de filas de nuestra base de datos es: ', df_hb_lens)
```

```
El número de filas de nuestra base de datos es:  119386
```

4 Trabajando con los datos

Con los datos limpios y listos para el análisis, es hora de generar la información que ayudará a nuestros **stakeholders** a tomar *mejores decisiones basadas en datos.*

4.1 Filtrado de datos

Separaré los datos en dos categorías: *City Hotel* and a *Resort Hotel*. Primero los analizaré por separado, y luego, procederé a analizarlos juntos para entender las temporadas.

Es momento de crear dos bases de datos.

In [ ]:
```
# CITY HOTEL
df_hb_CH = df_hb.groupby(by=['hotel']).get_group('City Hotel')

df_hb_CH.head(5)
```

Out[ ]:

	hotel	is_canceled	lead_time	arrival_date_year	arrival_date_month	arrival_date_week_numb
40060	City Hotel	0	6	2015	July	
40061	City Hotel	1	88	2015	July	
40062	City Hotel	1	65	2015	July	
40063	City Hotel	1	92	2015	July	
40064	City Hotel	1	100	2015	July	

5 rows × 27 columns

In [ ]:
```
# RESORT HOTEL
df_hb_RH = df_hb.groupby(by=['hotel']).get_group('Resort Hotel')

df_hb_RH.head(5)
```

Out[ ]:

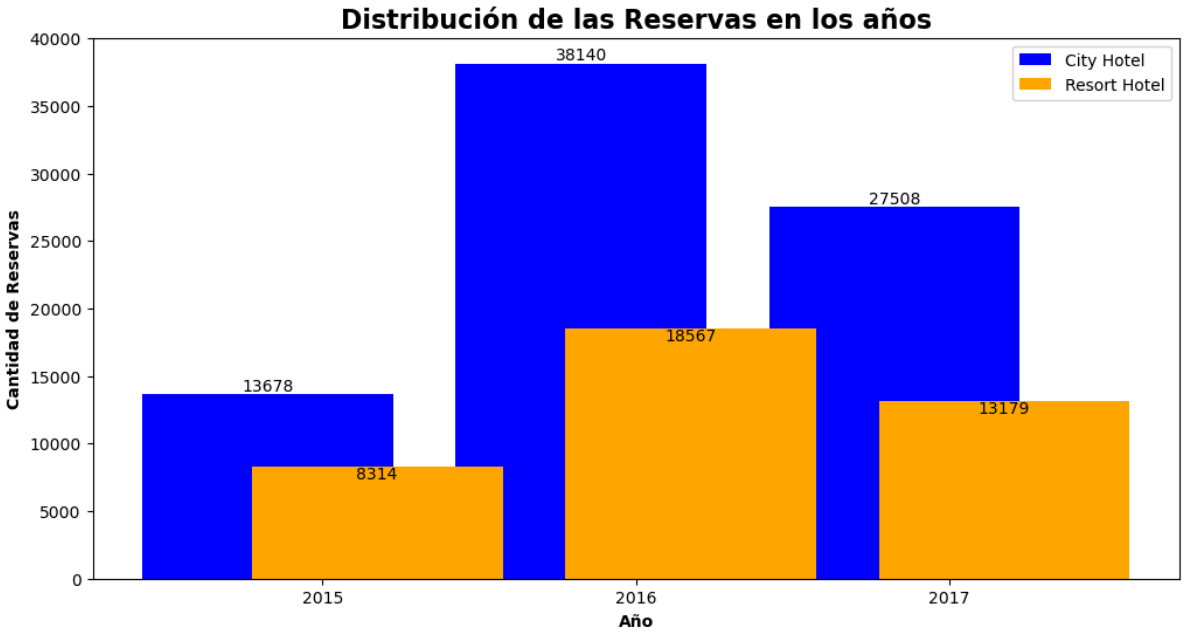
	hotel	is_canceled	lead_time	arrival_date_year	arrival_date_month	arrival_date_week_number
0	Resort Hotel	0	342	2015	July	27
1	Resort Hotel	0	737	2015	July	27
2	Resort Hotel	0	7	2015	July	27
3	Resort Hotel	0	13	2015	July	27
4	Resort Hotel	0	14	2015	July	27

5 rows × 27 columns

In [ ]:

```
# Analizemos qué hotel tuvo mayor actividad en el año
bookings_activity = df_hb.groupby(['arrival_date_year','hotel']).size().unstack()
print(bookings_activity)

# Variable TOTAL y PORCENTAJE para utilizar con los gráficos
total_bk_count = bookings_activity.sum()
canceled_percentage = (bookings_activity / total_bk_count) * 100

# Extraemos los valores para crear un gráfico de barras
years = bookings_activity.index
ch_count = bookings_activity['City Hotel']
rh_count = bookings_activity['Resort Hotel']

# Tamaño del gráfico
fig, ax = plt.subplots(figsize=(12, 6))

# Posicionamiento de las barras
r1 = range(len(years))
r2 = [x + 0.35 for x in r1]

# Barras
bar1 = ax.bar(r1, ch_count, color='blue', label='City Hotel')
bar2 = ax.bar(r2, rh_count, color='orange', label='Resort Hotel')

# Leyenda
ax.set_xlabel('Año', fontweight='bold')
ax.set_ylabel('Cantidad de Reservas', fontweight='bold')
ax.set_title('Distribución de las Reservas en los años', fontsize=16, fontweight='b
ax.set_xticks([r + 0.35/2 for r in range(len(years))])
ax.set_xticklabels(years)
ax.legend()

# Añadimos los valores del conteo a las barras
for bar in bar1:
    height = bar.get_height()
    ax.text(bar.get_x() + bar.get_width()/2.0, height, f'{int(height)}', ha='center

for bar in bar2:
    height = bar.get_height()
    ax.text(bar.get_x() + bar.get_width()/2.0, height, f'{int(height)}', ha='center
plt.show()
```

```
hotel              City Hotel  Resort Hotel
arrival_date_year
2015                    13678          8314
2016                    38140         18567
2017                    27508         13179
```



4.2 Análisis City Hotel

Modificaré los valores de la columna *'is_canceled'*. Voy a reemplazar los valores de 0 y 1 por los siguientes 0 = cancelado y 1 = no cancelado. Como el motivo es practicar el modificar datos, decidí realizar este proceso por separado para cada uno de los hoteles. Pero, este paso se puede realizar previo a separar los datos en dos bases.

Vayamos a ello

In [ ]:
```
# Modificando los valores
cancel = {
    0: 'cancelado',
    1:'no cancelado'
}

df_hb_CH['is_canceled'] = df_hb_CH['is_canceled'].map(cancel)

df_hb_CH['is_canceled'].head(5)
```

Out[ ]:
```
40060       cancelado
40061    no cancelado
40062    no cancelado
40063    no cancelado
40064    no cancelado
Name: is_canceled, dtype: object
```

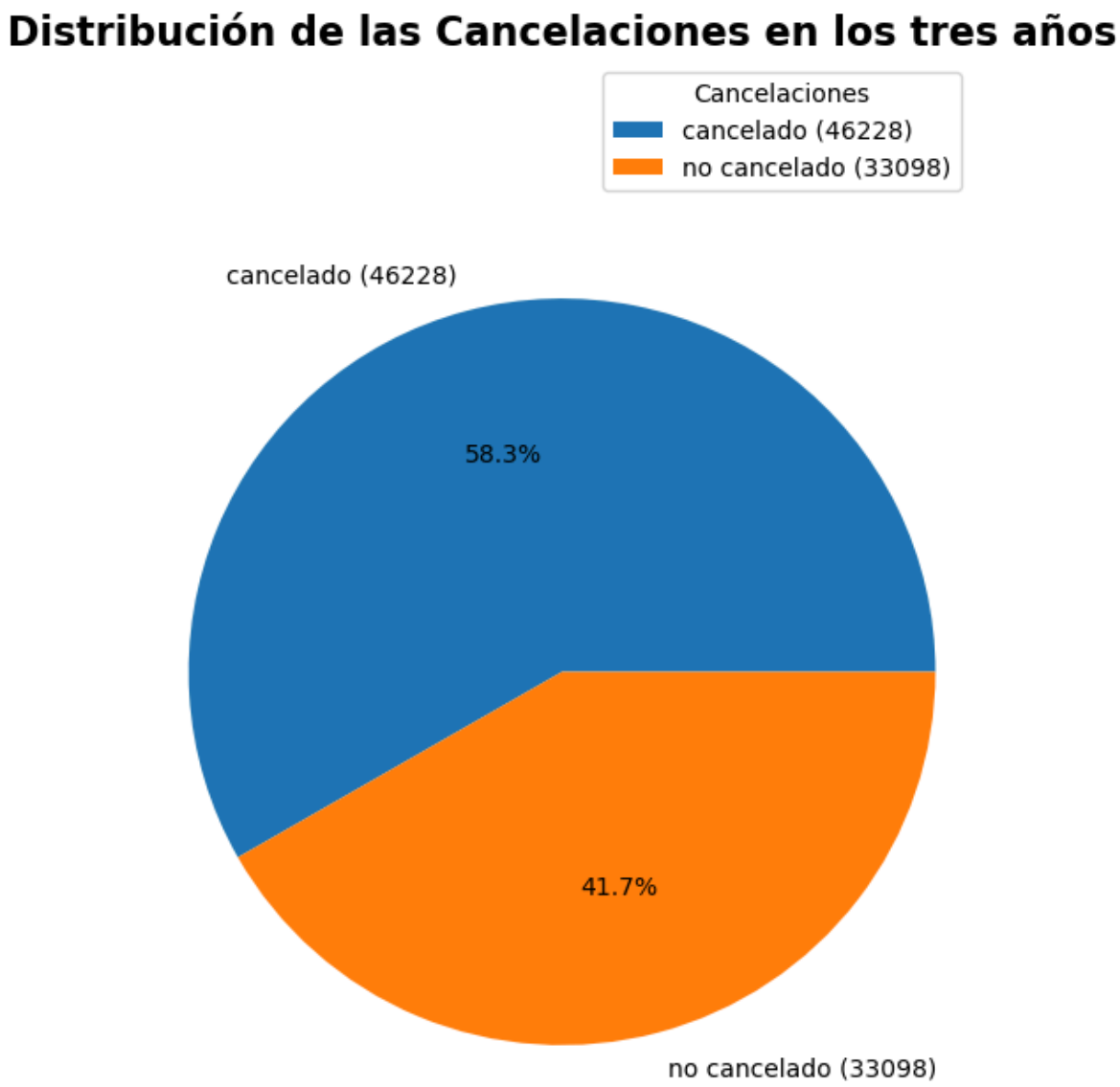
4.2.a Reservaciones canceladas

La siguente base de datos posee los datos de las reservas realizadas en los años *2015, 2016,* y *2017*. Entonces, miraré a las cancelaciones analizando el total de los tres años y luego las **separaré** de *forma individual por año*.

Esto permitirá a los stakeholders saber *el número total de cancelaciones y poder compararlas año con año.*

In [ ]:
```
# Realizando el conteo de los valores cancelado y no cancelado. Los almaceno en una
canceled_counts = df_hb_CH['is_canceled'].value_counts()
print(canceled_counts)

# Variable TOTAL y PORCENTAJE para utilizar con los gráficos
total_count = canceled_counts.sum()
canceled_percentage = (canceled_counts / total_count) * 100

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

# Tamaño del gráfico
fig, ax = plt.subplots(figsize=(6, 9))

# Generando nuestro gráfico
plt.pie(canceled_counts, labels=labels, autopct='%1.1f%%')
plt.legend(title='Cancelaciones')
ax.set_title('Distribución de las Cancelaciones en los tres años', fontsize=16, for
plt.axis('equal')
plt.show()
```

```
cancelado       46228
no cancelado    33098
Name: is_canceled, dtype: int64
```



Los datos muestran que en tres años hubo más reservas canceladas representando **un porcentaje total de 58.3%**.

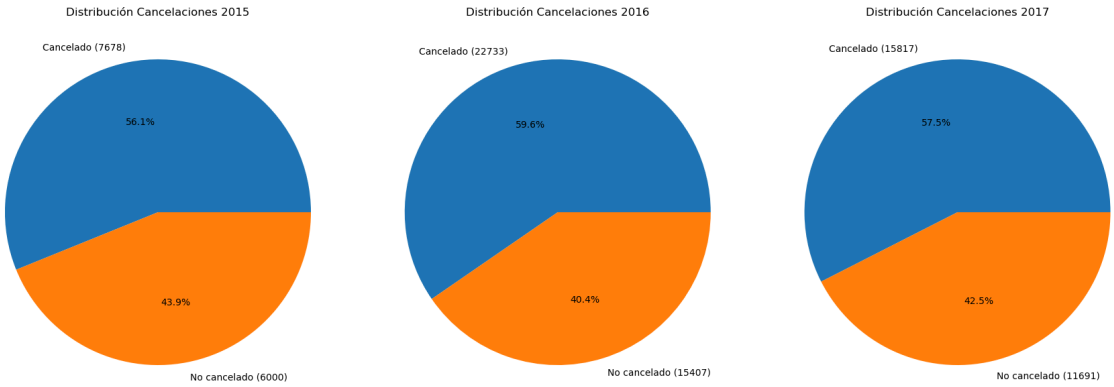
Se recomiendo tratar de identificar el motivo o causas de las cancelaciones en las reservas. Este proceso ayudará a entender mejor a sus clientes y con ello, crear promociones para atraer a los clientes y así, reducir el rango de cancelaciones.

Analizemos cuál de los tres años registra el mayor número de cancelaciones.

In [ ]:
```
# Separamos las cancelaciones por año
canceled_by_years_count = df_hb_CH.groupby('arrival_date_year', group_keys=False)[[
print(canceled_by_years_count)

# Creamos los gráficos
fig, axs = plt.subplots(1, 3, figsize=(18, 6))

# Etiquetas
for i, year in enumerate(canceled_by_years_count.index):
    data = canceled_by_years_count.loc[year]
```

```
    labels = [f"{status.capitalize()} ({count})" for status, count in data.items()]
    axs[i].pie(data, labels=labels, autopct='%1.1f%%')
    axs[i].set_title(f'Distribución Cancelaciones {year}')

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

```
is_canceled         cancelado  no cancelado
arrival_date_year
2015                     7678          6000
2016                    22733         15407
2017                    15817         11691
```



4.3 Análisis Resort Hotel

Como el proyecto es para practicar, voy a realizar el mismo análisis que realicé para el hotel de Ciudad con el Resort Hotel. Voy a modificar los valores de la columna '*is_canceled*', reemplazando los valores 0 y 1, donde 0 = cancelado y 1 = no cancelado

Comencemos a trabajar con nuestros datos.

In [ ]:
```
# Modificando los valores
cancel = {
    0: 'cancelado',
    1:'no cancelado'
}

df_hb_RH['is_canceled'] = df_hb_RH['is_canceled'].map(cancel)

df_hb_RH['is_canceled'].head(5)
```

Out[ ]:
```
0    cancelado
1    cancelado
2    cancelado
3    cancelado
4    cancelado
Name: is_canceled, dtype: object
```

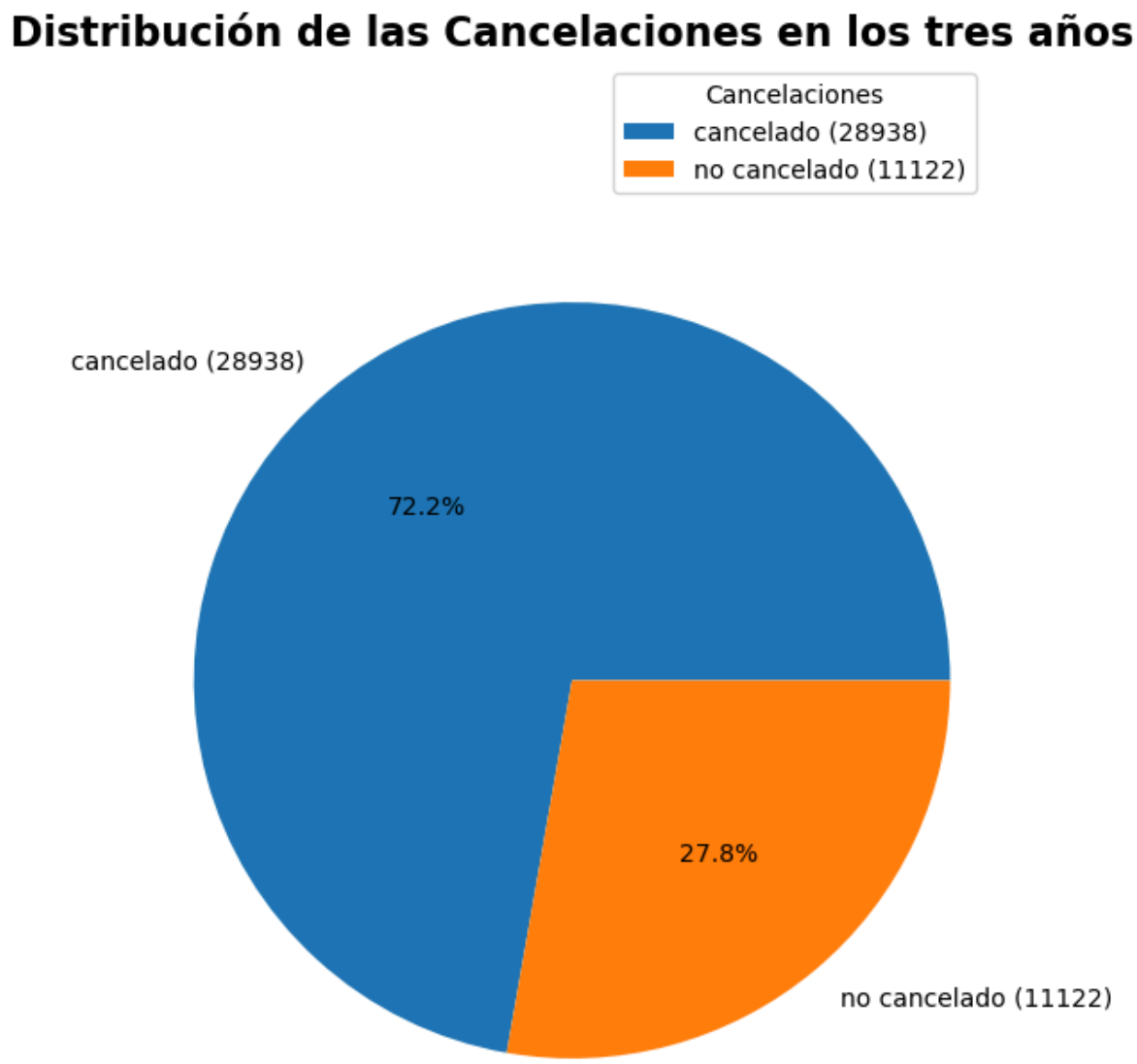
4.3.a Reservaciones Canceladas

La siguente base de datos posee los datos de las reservas realizadas en los años *2015, 2016,* y *2017*. Entonces, miraré a las cancelaciones analizando el total de los tres años y luego las **separaré** de *forma individual por año*.

Esto permitirá a los stakeholders saber *el número total de cancelaciones y poder compararlas año con año*.

In [ ]:
```
# Contando los valores y los almaceno en una variable - RESORT HOTEL
canceled_counts = df_hb_RH['is_canceled'].value_counts()
print(canceled_counts)

# Variable TOTAL y PORCENTAJE para utilizar con los gráficos
total_count = canceled_counts.sum()
canceled_percentage = (canceled_counts / total_count) * 100

# Con una función creamos las etiquetas
labels = [f'{canceled} ({count})' for canceled, count in zip(canceled_counts.index,

# Tamaño del gráfico
fig, ax = plt.subplots(figsize=(6, 9))

# Gráfico
plt.pie(canceled_counts, labels=labels, autopct='%1.1f%%')
plt.legend(title='Cancelaciones')
ax.set_title('Distribución de las Cancelaciones en los tres años', fontsize=16, for
plt.axis('equal')
plt.show()
```

```
cancelado       28938
no cancelado    11122
Name: is_canceled, dtype: int64
```



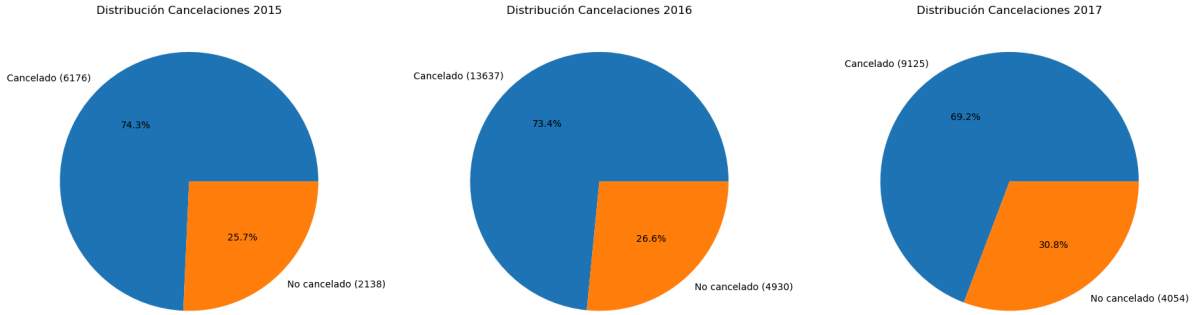
Los datos muestran que en tres años hubo más reservas canceladas representando **un porcentaje total de 72.2%**.

Analizemos cuál de los tres años registra el mayor número de cancelaciones.

In [ ]:
```
# Separamos los datos por año
canceled_by_years_count = df_hb_RH.groupby('arrival_date_year', group_keys=False)[[
print(canceled_by_years_count)

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

# Creamos las etiquetas
for i, year in enumerate(canceled_by_years_count.index):
    data = canceled_by_years_count.loc[year]
    labels = [f"{status.capitalize()} ({count})" for status, count in data.items()]
    axs[i].pie(data, labels=labels, autopct='%1.1f%%')
    axs[i].set_title(f'Distribución Cancelaciones {year}')
```

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

```
is_canceled         cancelado  no cancelado
arrival_date_year
2015                     6176          2138
2016                    13637          4930
2017                     9125          4054
```



Al analizar ambos hoteles, **CITY** y **RESORT**, se puede apreciar que el *RESORT* tiene un mayor porcentaje de cancelaciones, presentendo un total de **72.2%** contra los **58.3%** que presenta el hotel *CITY*. Sin embargo, al analizar las cancelaciones por año, se observa que las cancelaciones en el hotel *RESORT* disminuyen con el paso de los años, en contrario con el hotel *CITY* que presenta un incremento en su último año.

4.4 ¿Qué afecta a las cancelaciones?

Para el siguiente análisis, voy a crear una nueva base de datos utilizando las columnas *hotel, is_canceled, lead_time, market_segment, and customer_type*, con el fin de examinar y poder determinar qué fenómeno puede estar causando las cancelaciones.

Procedamos a crear la base de datos con la que vamos a trabajar.

4.4.a Nueva Base de datos

In [ ]:
```
# Creando la nueva base de datos
df_cancellations = df_hb[['hotel','is_canceled','lead_time','market_segment','custo

df_cancellations.head(5)
```

Out[ ]:

	hotel	is_canceled	lead_time	market_segment	customer_type	arrival_date_year
0	Resort Hotel	0	342	Direct	Transient	2015
1	Resort Hotel	0	737	Direct	Transient	2015
2	Resort Hotel	0	7	Direct	Transient	2015
3	Resort Hotel	0	13	Corporate	Transient	2015
4	Resort Hotel	0	14	Online TA	Transient	2015

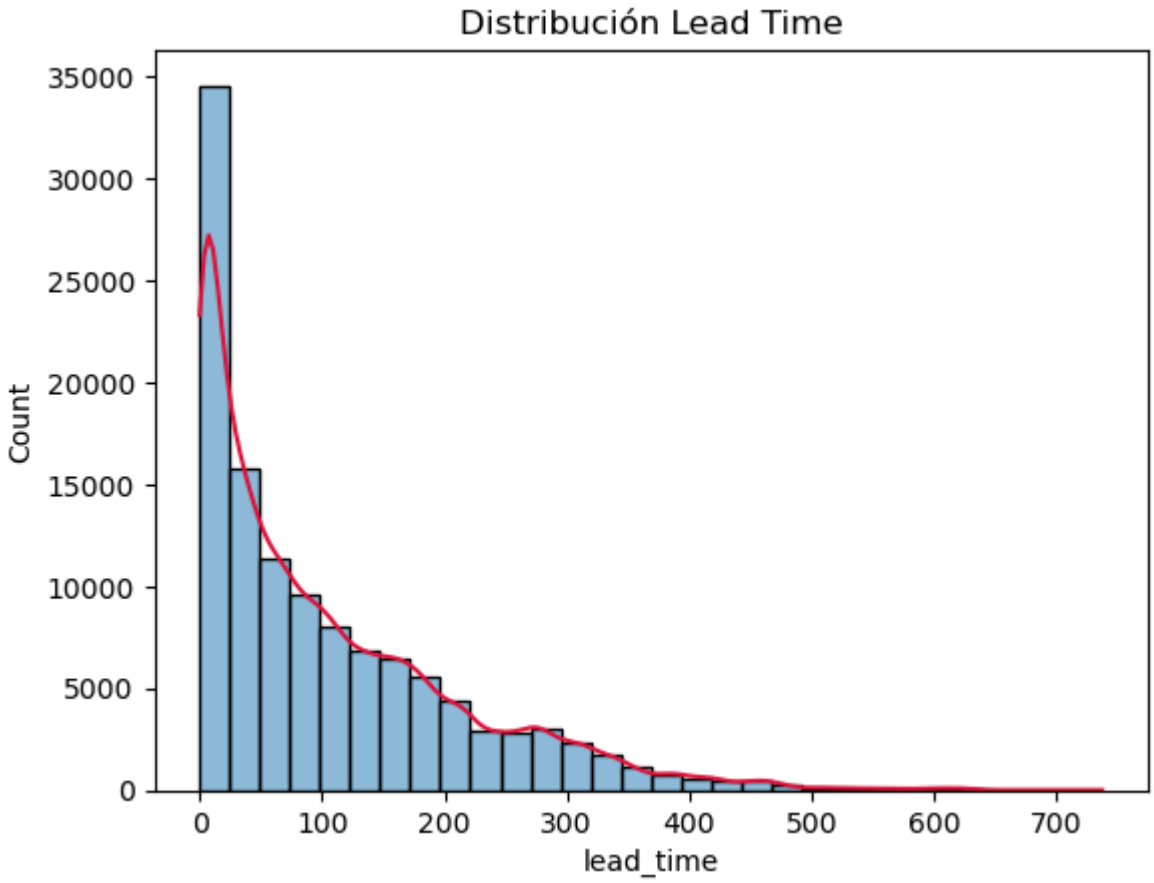
In [ ]:
```
df_cancellations.dtypes
```

Out[ ]:
```
hotel                object
is_canceled           int64
lead_time             int64
market_segment       object
customer_type        object
arrival_date_year     int64
dtype: object
```

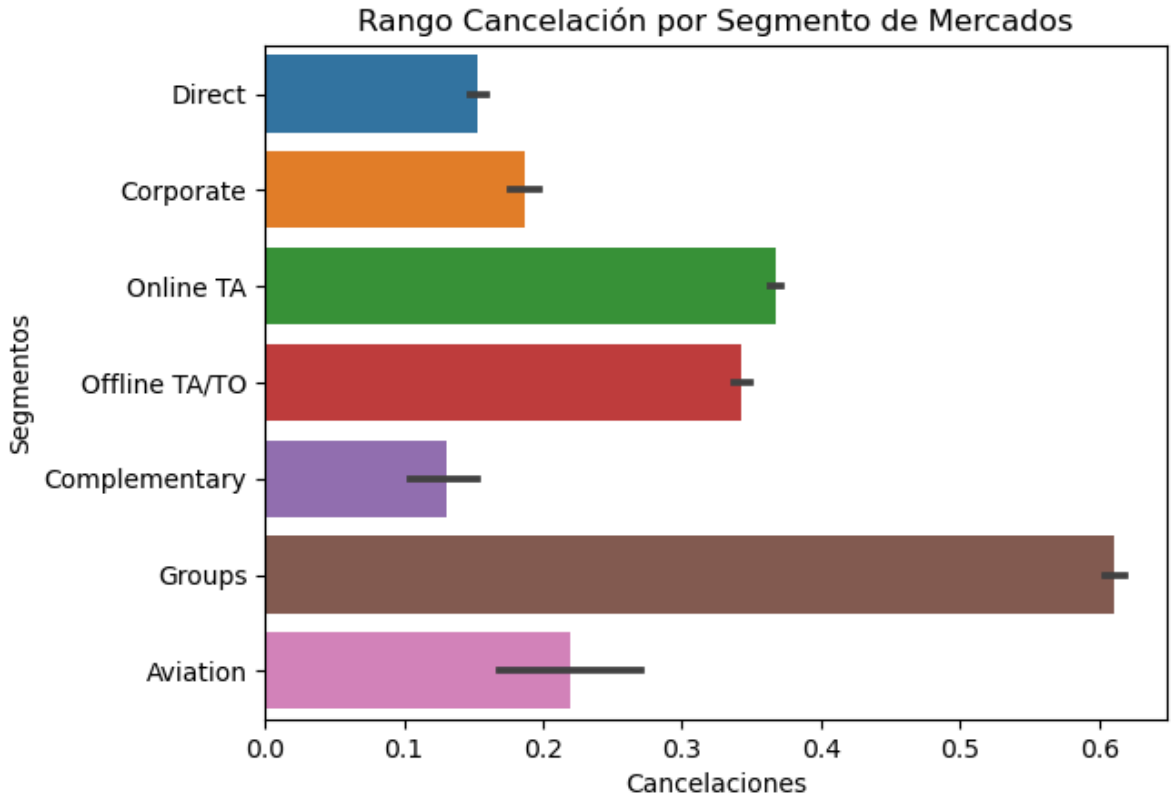
Ahora que tenemos nuestra nueva base de datos, es momento de visualizar las distribuciones y relaciones entre las variables.

Voy a utilizar los valores de la columna'*lead_time*' junto con un gráfico de histograma para visualizar el número de días pasados entre la reserva y el momento del arribo al hotel. El histograma nos mostrará la distribución de una variable contando el número de observaciones de ella.

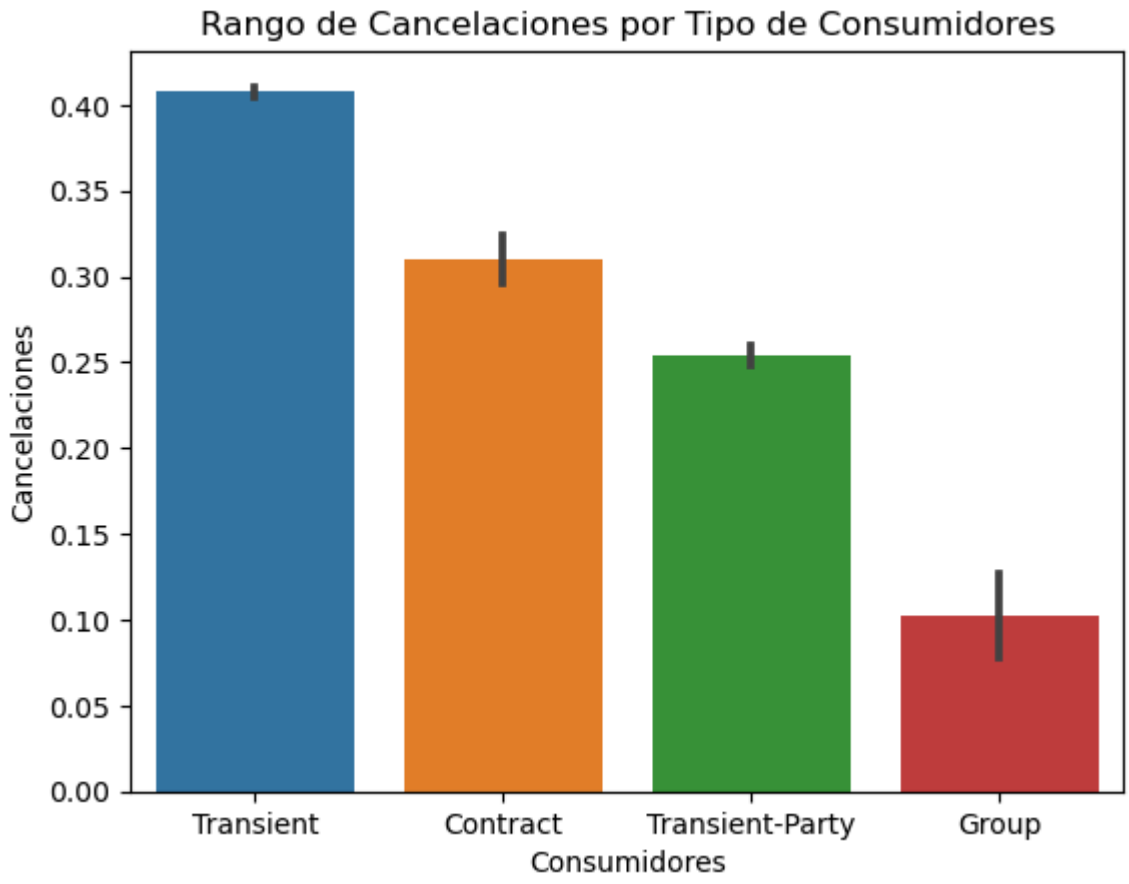
In [ ]:
```
ax = sns.histplot(df_cancellations['lead_time'], bins=30, kde=True)
ax.lines[0].set_color('crimson')
plt.title('Distribución Lead Time')
plt.show()
```



In [ ]:
```
# Cancellation rate by market segment

sns.barplot(x='is_canceled', y='market_segment', data=df_cancellations)
plt.title('Rango Cancelación por Segmento de Mercados')
plt.xlabel('Cancelaciones')
plt.ylabel('Segmentos')
plt.show()
```



In [ ]:
```
# Cancellation rate by customer type
sns.barplot(x='customer_type', y='is_canceled', data=df_cancellations)
plt.title('Rango de Cancelaciones por Tipo de Consumidores')
plt.xlabel('Consumidores')
plt.ylabel('Cancelaciones')
plt.show()
```



4.4.b Analizando el impacto de las variables utilizadas

Ahora llegó el momento de utilizar los modelos de *Machine Learning* con el fin de comprender cuál de nuestras variables es la que afecta a las cancelaciones. Utilizaré el modelo de *regresión logística* para comprender la relación entre los datos de las columnas *'is_canceled'*, *'lead_time'*, *'market_segment'*, y*'customer_type'*.

In [ ]:
```
# Creamos el modelo de Regresión Logística
cancellation_model = smf.logit('is_canceled ~ lead_time + C(market_segment) + C(cus

# Sipnosis del modelo
print(cancellation_model.summary())
```

```
Optimization terminated successfully.
         Current function value: 0.563430
         Iterations 6
                           Logit Regression Results
==============================================================================
Dep. Variable:            is_canceled   No. Observations:               119386
Model:                          Logit   Df Residuals:                   119375
Method:                           MLE   Df Model:                           10
Date:                Fri, 05 Jul 2024   Pseudo R-squ.:                 0.1452
Time:                        16:03:57   Log-Likelihood:                -67266.
converged:                       True   LL-Null:                       -78695.
Covariance Type:            nonrobust   LLR p-value:                     0.000
==============================================================================
                                                 coef    std err          z      P>|z|
[0.025      0.975]
------------------------------------------------------------------------------
Intercept                                    -2.3478      0.163    -14.375      0.000
-2.668      -2.028
C(market_segment)[T.Complementary]           -0.7135      0.192     -3.711      0.000
-1.090      -0.337
C(market_segment)[T.Corporate]                0.0173      0.162      0.106      0.915
-0.301       0.335
C(market_segment)[T.Direct]                  -0.7147      0.160     -4.462      0.000
-1.029      -0.401
C(market_segment)[T.Groups]                   1.8303      0.160     11.443      0.000
1.517       2.144
C(market_segment)[T.Offline TA/TO]            0.3801      0.159      2.388      0.017
0.068       0.692
C(market_segment)[T.Online TA]                0.2978      0.159      1.879      0.060
-0.013       0.608
C(customer_type)[T.Group]                    -0.4647      0.149     -3.126      0.002
-0.756      -0.173
C(customer_type)[T.Transient]                 1.1394      0.041     27.881      0.000
1.059       1.220
C(customer_type)[T.Transient-Party]          -0.6750      0.043    -15.556      0.000
-0.760      -0.590
lead_time                                     0.0055   7.26e-05     76.162      0.000
0.005       0.006
==============================================================================
```

Una vez desarrollado y presentado el modelo, es necesario comprender la importancia de **cada variable**.

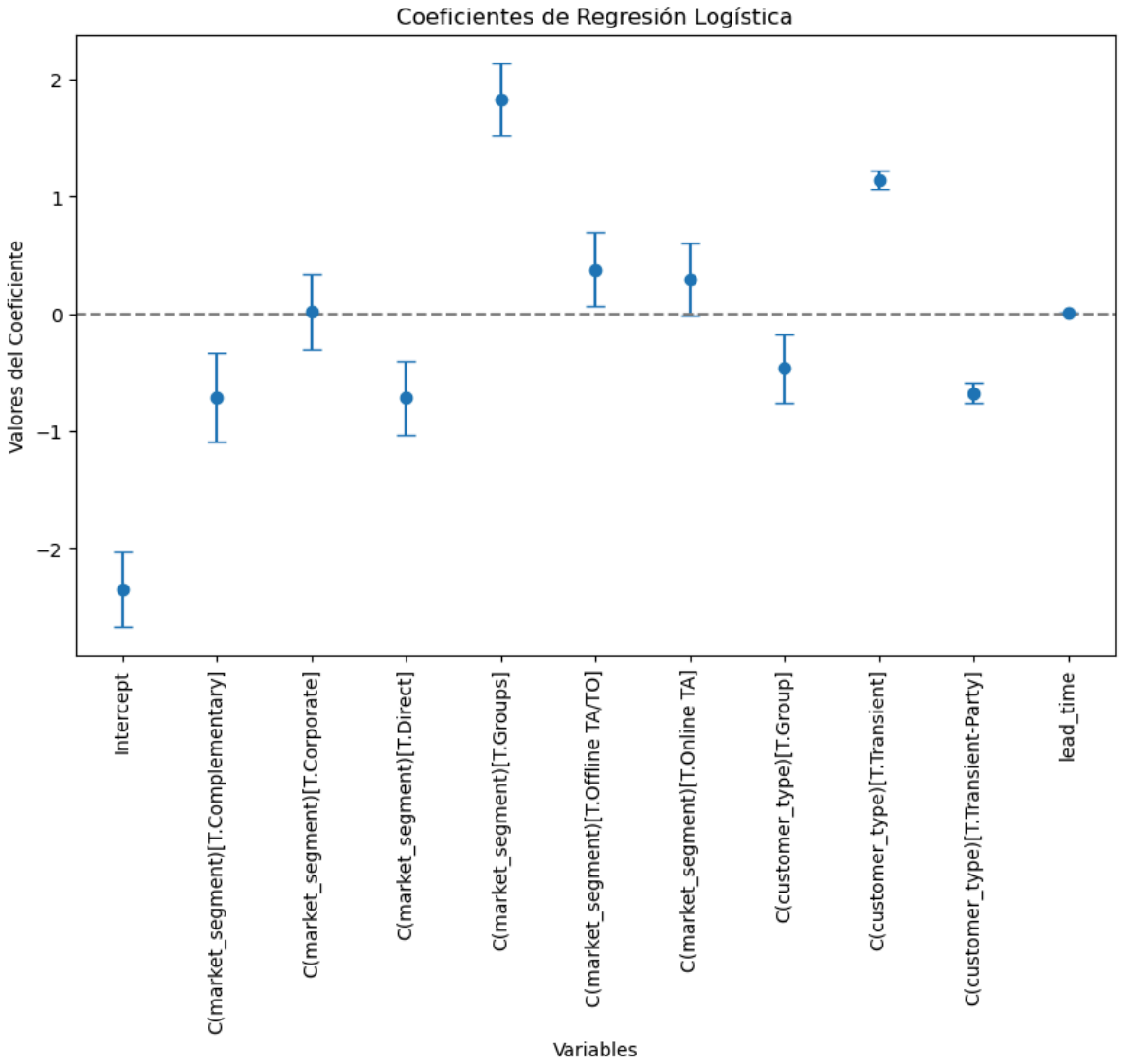
In [ ]:
```
# Extraemos los valores y coeficientes
cancellation_coef = cancellation_model.params
cancellation_values = cancellation_model.pvalues
```

```
# Desplegamos los coeficientes y los valores p
print(f"Coeficientes:\n{cancellation_coef}\n")
print(f"Valores-P:\n{cancellation_values}\n")
```

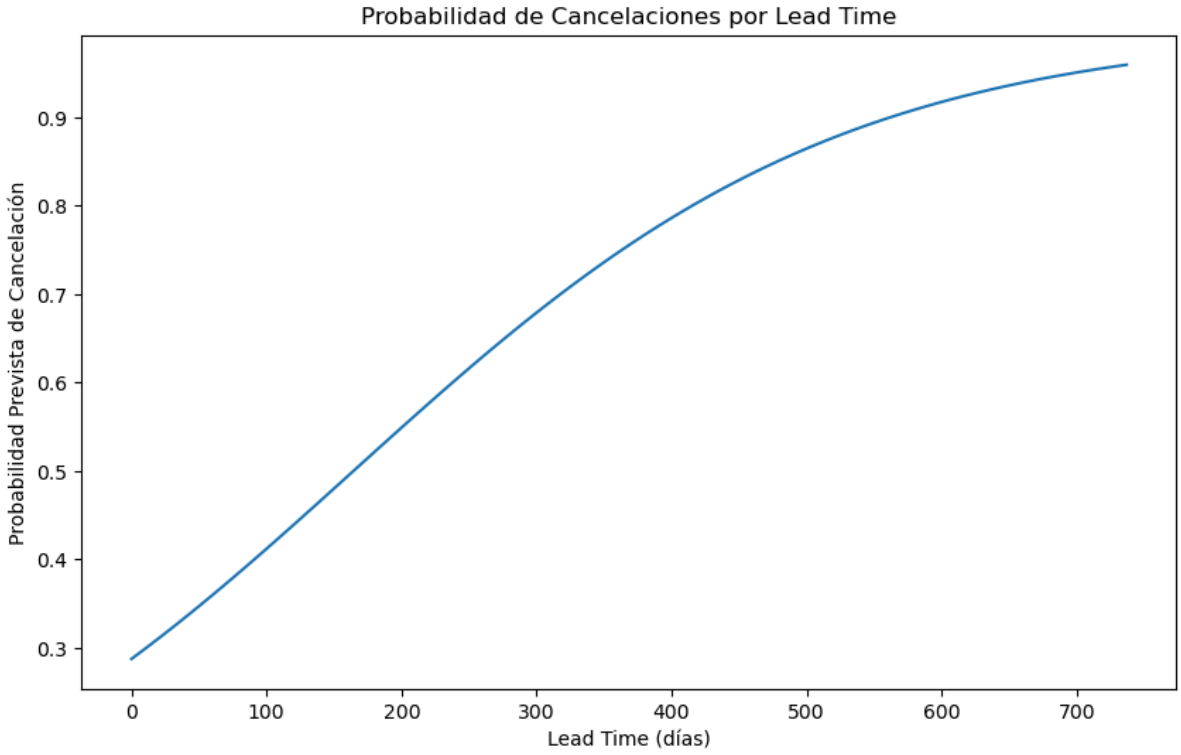
```
Coeficientes:
Intercept                               -2.347786
C(market_segment)[T.Complementary]      -0.713530
C(market_segment)[T.Corporate]           0.017277
C(market_segment)[T.Direct]             -0.714733
C(market_segment)[T.Groups]              1.830286
C(market_segment)[T.Offline TA/TO]       0.380110
C(market_segment)[T.Online TA]           0.297807
C(customer_type)[T.Group]               -0.464664
C(customer_type)[T.Transient]            1.139408
C(customer_type)[T.Transient-Party]     -0.675040
lead_time                                0.005531
dtype: float64

Valores-P:
Intercept                                7.434710e-47
C(market_segment)[T.Complementary]       2.063348e-04
C(market_segment)[T.Corporate]           9.152015e-01
C(market_segment)[T.Direct]              8.137130e-06
C(market_segment)[T.Groups]              2.546703e-30
C(market_segment)[T.Offline TA/TO]       1.694970e-02
C(market_segment)[T.Online TA]           6.027475e-02
C(customer_type)[T.Group]                1.769141e-03
C(customer_type)[T.Transient]           4.542420e-171
C(customer_type)[T.Transient-Party]      1.446957e-54
lead_time                                0.000000e+00
dtype: float64
```

Creamos un gráfico para visualizar los datos de nuestro modelo

In [ ]:
```
# Necesitamos los intervalos de confidencia
cancell_conf = cancellation_model.conf_int()

# Añadimos los coeficientes a los intervalos
cancell_conf['coeff'] = cancellation_coef

# Renombramos a las columnas para entender los datos
cancell_conf.columns = ['2.5%','97.5%', 'coeff']

# Gráfico
plt.figure(figsize=(10, 6))
plt.errorbar(cancell_conf.index, cancell_conf['coeff'], yerr=[cancell_conf['coeff']
plt.axhline(0, color='gray', linestyle='--')
plt.xticks(rotation=90)
plt.title('Coeficientes de Regresión Logística')
plt.xlabel('Variables')
plt.ylabel('Valores del Coeficiente')
plt.show()
```



In [ ]:
```
# Generamos rangos para los valores de Lead Time
lead_time_range = np.linspace(df_cancellations['lead_time'].min(), df_cancellations
predict_data = pd.DataFrame({
    'lead_time': lead_time_range,
    'market_segment': 'Online TA',
    'customer_type': 'Transient'
})

# Predict probabilities
predict_data['predicted_prob'] = cancellation_model.predict(predict_data)

# Plot predicted probabilities
plt.figure(figsize=(10, 6))
plt.plot(predict_data['lead_time'], predict_data['predicted_prob'])
plt.title('Probabilidad de Cancelaciones por Lead Time')
plt.xlabel('Lead Time (días)')
plt.ylabel('Probabilidad Prevista de Cancelación')
plt.show()
```



4.5 Analizando los plazos en las Reservas

Para el siguiente análisis, crearé una nueva base de datos utilizando las columnas *hotel, lead_time, is_canceled, adr, arrival_date_year, arrival_date_month, market_segment, customer_type* de esta forma podremos analizar si el lead time es la variable que está afectando a las cancelaciones en las reservas.

Vamos a tratar de entender los patrones en las reservas para así los hoteles pueden crear estrategias que puedan ofrecer a sus clientes.

4.5.a Análisis de las Reservas, creamos una nueva base de datos

In [ ]:
```
# Creando la base de datos de las reservas
df_book_lt = df_hb[['hotel','lead_time','is_canceled','adr','arrival_date_year','ar

df_book_lt.head(5)
```

Out[ ]:

	hotel	lead_time	is_canceled	adr	arrival_date_year	arrival_date_month	market_segment	cus
0	Resort Hotel	342	0	0.0	2015	July	Direct	
1	Resort Hotel	737	0	0.0	2015	July	Direct	
2	Resort Hotel	7	0	75.0	2015	July	Direct	
3	Resort Hotel	13	0	75.0	2015	July	Corporate	
4	Resort Hotel	14	0	98.0	2015	July	Online TA	

In [ ]:
```
df_book_lt.dtypes
```

Out[ ]:
```
hotel                 object
lead_time              int64
is_canceled            int64
adr                  float64
arrival_date_year      int64
arrival_date_month    object
market_segment        object
customer_type         object
dtype: object
```

Con la nueva base de datos, visualicemos distribuciones y relaciones entre variables.

4.5.a.1 Análisis de las Reservas

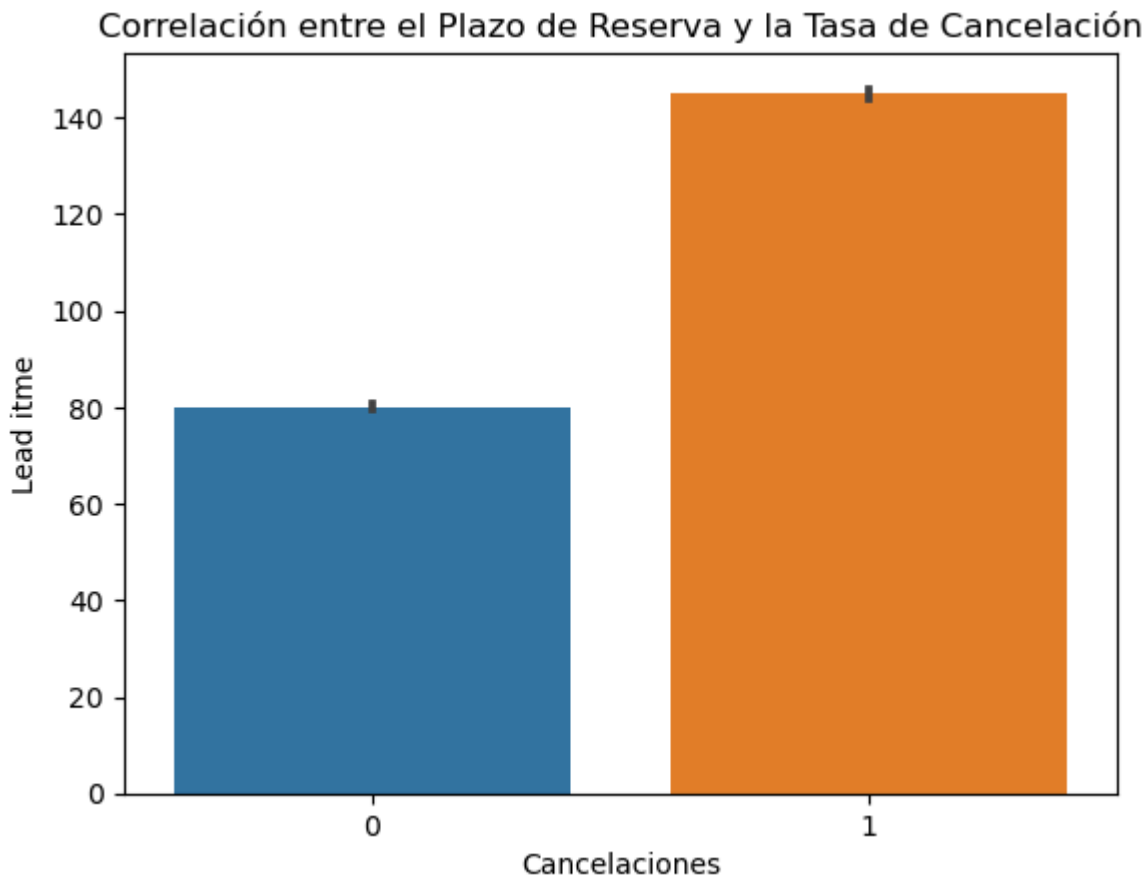
Comenzamos analizando el promedio de 'Lead time' entre el hotel 'City' y 'Resort', con el fin de comprender en cuál hotel se producen las reservas con mayor anticipación.

In [ ]:
```
# Promedio de anticipado de tiempo en las reservas
bavg_lead_time = df_book_lt.groupby('hotel')['lead_time'].mean()
print(bavg_lead_time)
```

```
hotel
City Hotel      109.741106
Resort Hotel     92.675686
Name: lead_time, dtype: float64
```

In [ ]:
```
# Analizamos la correlación entre las cancelaciones y el tiempo anticipado en la re
corr_cancellation = df_book_lt['lead_time'].corr(df_book_lt['is_canceled']).round(2
print(f"Correlación entre el plazo de reserva y la tasa de cancelación es: {corr_ca


# Visualizamos los datos
sns.barplot(x='is_canceled', y='lead_time', data=df_book_lt)
plt.title('Correlación entre el Plazo de Reserva y la Tasa de Cancelación')
plt.xlabel('Cancelaciones')
plt.ylabel('Lead itme')
plt.show()
```

```
Correlación entre el plazo de reserva y la tasa de cancelación es: 0.29
```



Los resultados de nuestro análisis muestran que las variables '*Lead Time*' and '*Cancellations*', poco se relacionan entre ellas, presentando un coeficiente de correlación de **0.29**. Las cancelaciones no se encuentran afectadas por el tiempo anticipado de reserva.

Vamos a introducir una nueva variable a nuestro análisis, ella es, ***ADR (Average Daily rate)***.Trataré de identificar si el ADR está afectando a las cancelaciones en los hoteles.

In [ ]:
```
# Correlación entre Lead Time y ADR
corr_adr = df_book_lt['lead_time'].corr(df_book_lt['adr']).round(2)
print(f"La correlación entre el plazo de reserva y la tasa promedio diaria es: {cor
```

```
La correlación entre el plazo de reserva y la tasa promedio diaria es: -0.06
```

El resultado del análisis muestra un coeficiente de correlación de valor **-0.06**. Por lo que no hay correlación entre las variables, '*Lead Time*' and '*ADR*'. La nueva variable no afecta a las cancelaciones.

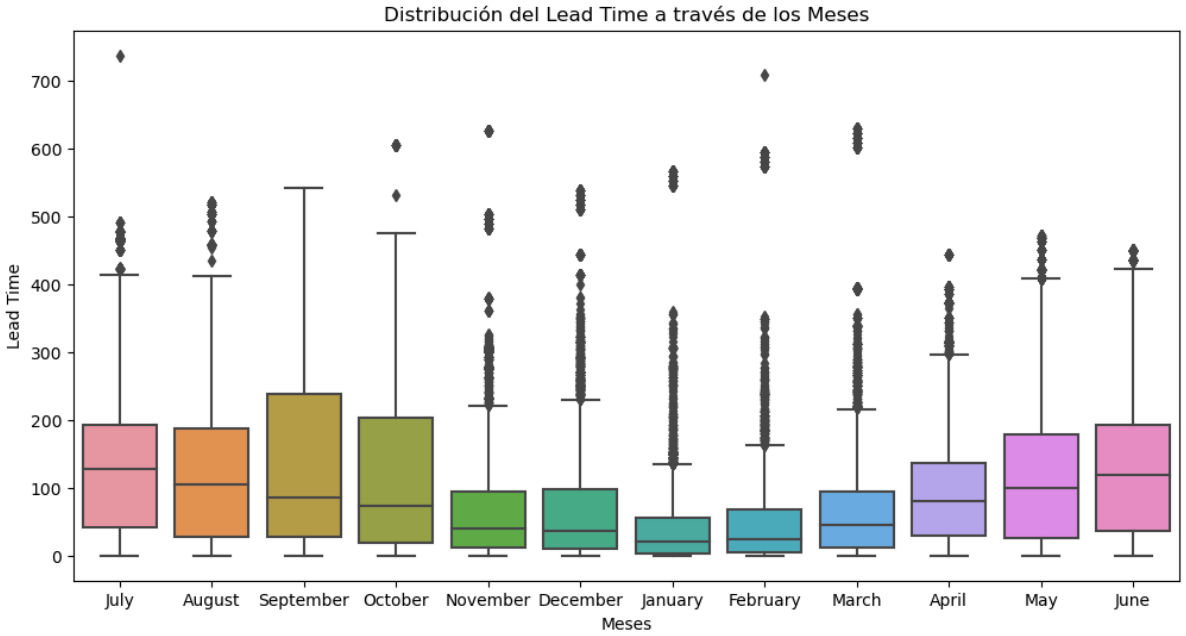
Visualizemos los datos con el fin de encontrar cómo las variables trabajan entre ellas. El resultado de nuestro análisis es encontrar insights que sean significativos.

Visualizando la distribución de Lead Time

Para visualizar nuestros datos, utilizaré los valores de la columna '*arrival_date_month*'. Estos valores junto a un boxplot nos ayudaran an entender mejor el análisis en las correlaciones.

In [ ]:
```
# Creating the visualization with the correlation between 'lead_time' and 'arrival_
# Size of the boxplot
plt.figure(figsize=(12, 6))

# Boxplot for the months
sns.boxplot(x='arrival_date_month', y='lead_time', data=df_book_lt)
```

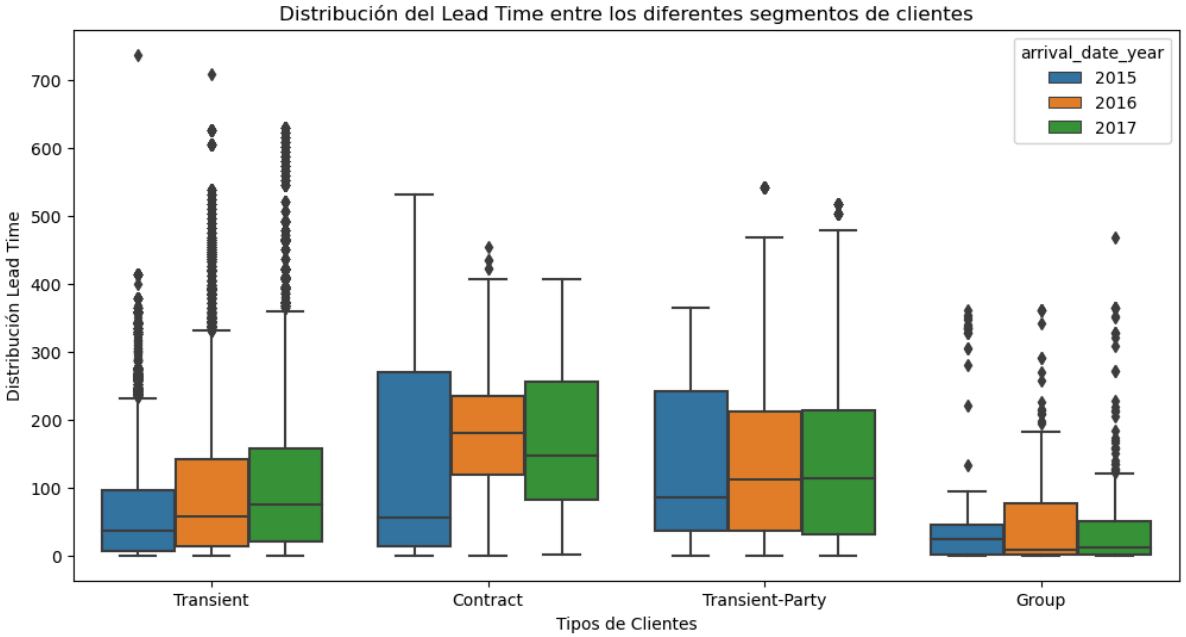
```
plt.title('Distribución del Lead Time a través de los Meses')
plt.xlabel('Meses')
plt.ylabel('Lead Time')
plt.show()
```



Como era de esperarse, la época de mayor ocupación es en la temporada de vacaciones, la cual comienza en mayo y finaliza en octubre. Es importante destacar septiembre como uno de los meses que presenta mayor número de reservas.

Analicemos cómo se distribuye la ocupación para los diferentes segmentos hoteleros

In [ ]:
```
# Utilizaré los valores de la columna customer_type.
# Creamos el gráfico
plt.figure(figsize=(12,6))

# Boxplot con los valores de customer_type
sns.boxplot(x='customer_type', y='lead_time', hue='arrival_date_year', data=df_book
plt.title('Distribución del Lead Time entre los diferentes segmentos de clientes')
plt.xlabel('Tipos de Clientes')
plt.ylabel('Distribución Lead Time')
plt.show()
```



Con la visualización de los datos, vamos con un par de conclusiones desde los diferentes segmentos de clientes:

- Con los clientes *Transient*, obsevamos demasiados valores atípicos. Estos clientes muestran reservas dispares, pero la mayoría de las veces sus reservas son cercanas a la fecha de estadía. Entre los tres años, su tiempo de espera promedio es de aproximadamente 50 días.
- Con los clientes*Contract*, la variabilidad en el tiempo de espera de las reservas en 2015 muestra ser alta, y a partir de 2016 comienza a disminuir. Además, la mediana era de alrededor de 50 días en 2015, pero luego aumentó en 2016 a alrededor de 180 días, y en 2017, disminuyó a aproximadamente 150 días.
- Un análisis detallado de los datos de los clientes *Transient-Party* muestra que el número medio de reservas aumentó significativamente de 2015 a 2017, pasando de 80 a 100 días aproximadamente. No hay mucha variabilidad en el plazo de las reservas. Presentan una reducción en la variabilidad del plazo de reservas en 2017. En 2016, debido a algunos valores atípicos, están empezando a reservar mucho antes de su fecha de llegada.
- Los clientes *Group* muestran un tiempo de espera medio más bajo entre todos los clientes; tienden a reservar cerca de la fecha de estancia. Es importante tener en cuenta que, debido a que hay demasiados valores atípicos, también registran reservas mucho antes de la fecha de la estancia. Son los más constantes con sus reservas a lo largo de los años.

Recomendaciones:

- Crear estrategias de marketing dirigidas a los clientes en función de sus patrones de reserva. Una de ellas puede ser un descuento por reserva con el fin de incentivar a los clientes a reservar con mayor antelación.
- Establecer precios a largo plazo para ayudar a adelantar las reservas con los clientes *Contract*. Este tipo de estrategia ayudará a optimizar los ingresos.
- De ser posible, asignar recursos y personal a segmentos más predecibles debido a su patrón de reserva para darles un enfoque de gestión diferente.

4.6 Análisis de Ingresos

Es hora de analizar el **ADR** (tarifa diaria promedio) de cada hotel. Vamos a analizar cuánto dinero se obtiene de cada uno de los hoteles, *City* y *Resort*. Comenzaré comparando el *ADR* entre ambos. Más tarde, analizaré la variación por segmento y lead time.

Para este nuevo análisis, crearé una nueva base de datos.

In [ ]:
```
# Creando la nueva base de datos
df_revAdr = df_hb[['hotel','adr', 'market_segment', 'customer_type', 'lead_time', '

df_revAdr.head(5)
```

Out[ ]:

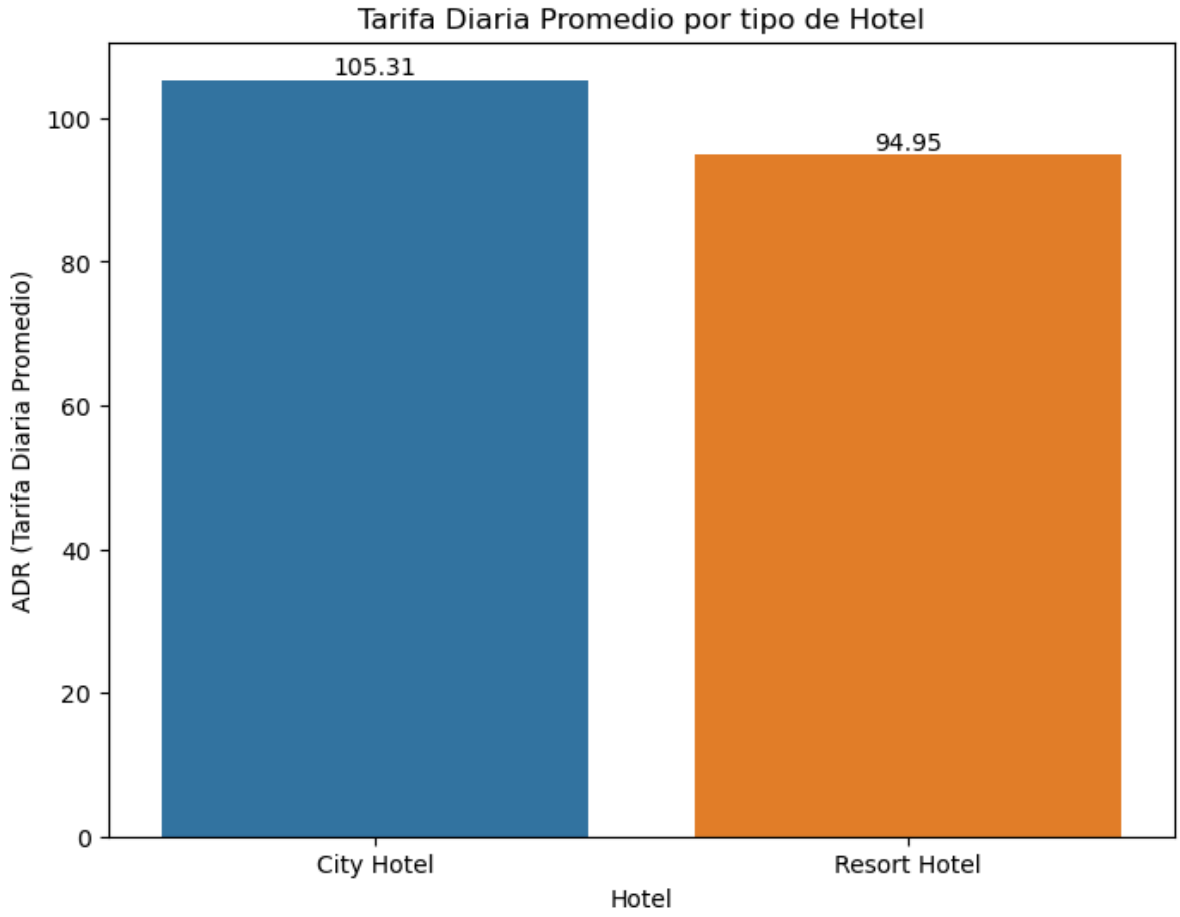
	hotel	adr	market_segment	customer_type	lead_time	arrival_date_year	reserved_room_type
0	Resort Hotel	0.0	Direct	Transient	342	2015	C
1	Resort Hotel	0.0	Direct	Transient	737	2015	C
2	Resort Hotel	75.0	Direct	Transient	7	2015	A
3	Resort Hotel	75.0	Corporate	Transient	13	2015	A
4	Resort Hotel	98.0	Online TA	Transient	14	2015	A

4.6.a Comparando el ADR entre los hoteles City y Resort.

In [ ]:
```
# Agrupando los datos por hotel
mean_Adr = df_revAdr.groupby('hotel')['adr'].mean().round(2).reset_index()

print(mean_Adr)
```

```
          hotel     adr
0    City Hotel  105.31
1  Resort Hotel   94.95
```

In [ ]:
```
# Visualizando los datos con un gráfico bar
plt.figure(figsize=(8, 6))

# Creating the plot
ax = sns.barplot(x='hotel', y='adr', data=mean_Adr)
ax.bar_label(ax.containers[0], fontsize=10)
plt.title('Tarifa Diaria Promedio por tipo de Hotel')
plt.xlabel('Hotel')
plt.ylabel('ADR (Tarifa Diaria Promedio)')
plt.show()
```



Los datos muestran que el ADR (tasa promedio diaria) para el *City Hotel* es **105.31**, y para el *Resort Hotel* es **94.95**.

4.6.b Analizando la variación del ADR entre tipos de habitación, segmento de mercado y plazo de entrega.

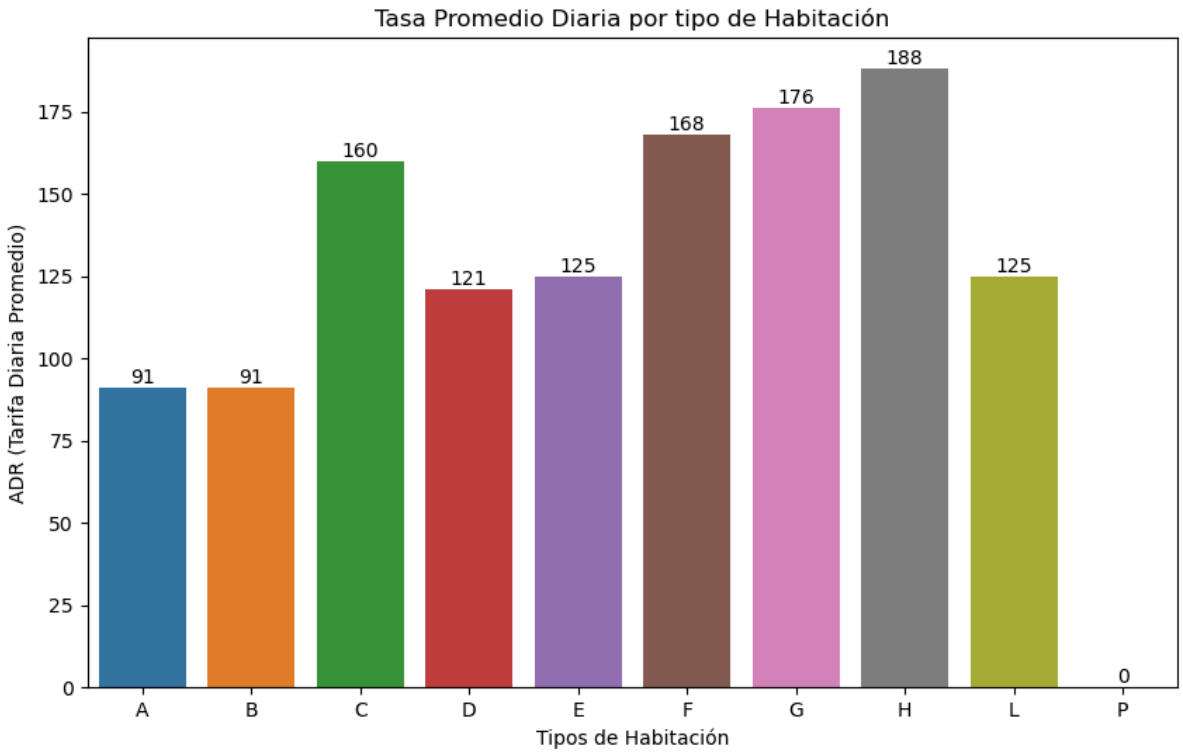
4.6.b.1 ADR variación por los distintos tipos de habitación.

In [ ]:
```
# Variación por tipo de habitación
mean_adr_rt = df_revAdr.groupby('reserved_room_type')['adr'].mean().round().reset_i

print(mean_adr_rt)
```

```
  reserved_room_type    adr
0                  A   91.0
1                  B   91.0
2                  C  160.0
3                  D  121.0
4                  E  125.0
5                  F  168.0
6                  G  176.0
7                  H  188.0
8                  L  125.0
9                  P    0.0
```

In [ ]:
```
# Creando el gráficot
plt.figure(figsize=(10, 6))

ax = sns.barplot(x='reserved_room_type', y='adr',  data=mean_adr_rt)
ax.bar_label(ax.containers[0], fontsize=10)
plt.title('Tasa Promedio Diaria por tipo de Habitación')
plt.xlabel('Tipos de Habitación')
```

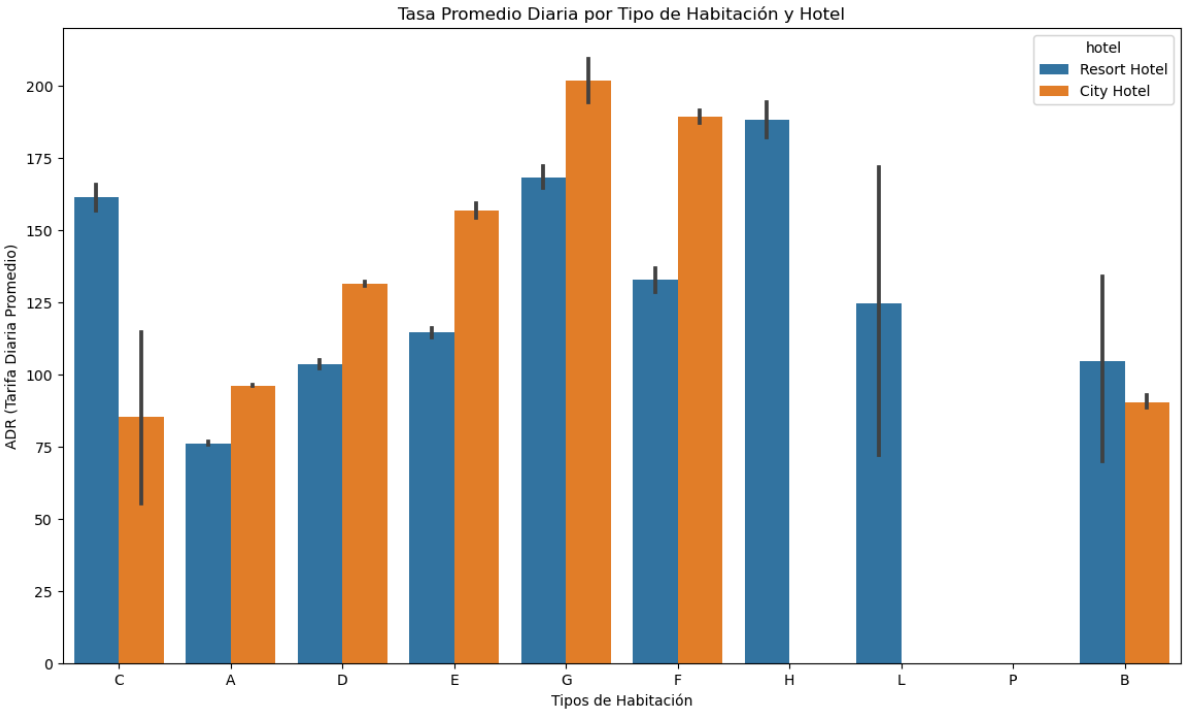
```
plt.ylabel('ADR (Tarifa Diaria Promedio)')
plt.show()
```



Al analizar el ADR por '*Tipo de Habitación*', los datos muestran que la habitación **H** tiene la tarifa diaria promedio más alta con un total de $188.-. Luego, con la tarifa diaria promedio más baja, tenemos las habitaciones **A** y **B**. Para este análisis, no consideré las plazas de estacionamiento.

Variación de los tipos de habitación por Hotel

In [ ]:
```
# Creando el gráficot
plt.figure(figsize=(14, 8))

ax = sns.barplot(x='reserved_room_type', y='adr',  hue='hotel', data=df_revAdr)

plt.title('Tasa Promedio Diaria por Tipo de Habitación y Hotel')
plt.xlabel('Tipos de Habitación')
plt.ylabel('ADR (Tarifa Diaria Promedio)')
plt.show()
```



Al separar los datos entre los dos hoteles, la habitación **H** es exclusiva del Hotel *Resort*. Luego, las habitaciones **C** y **G**, son las que presentan un ADR más alto. Diferente es el caso del Hotel *City*, donde las habitaciones **G** y **F** son las que tienen el ADR más alto. Y las habitaciones **C**, **A** y **B** muestran el ADR más bajo.

Si comparamos ambos análisis, se observan cifras diferentes para el ADR. Es importante tener esta diferencia presente al momento de crear ofertas especiales para cada hotel.

Es momento de analizar si los hoteles asignan las habitaciones a cómo fueron reservadas.

Compararé los valores de la columna *'reserved_room_type'* y *'assigned_room_type'*. Almacenaré el resultado de la comparación en una nueva columna llamada *'room_assigned_correctly'*, con 0 para valores *False* y 1 para valores *True*.

In [ ]:
```
# Comparando los valores de las columnas
df_revAdr['room_assigned_correctly'] = (df_revAdr['reserved_room_type'] == df_revAd

df_revAdr.head()
```

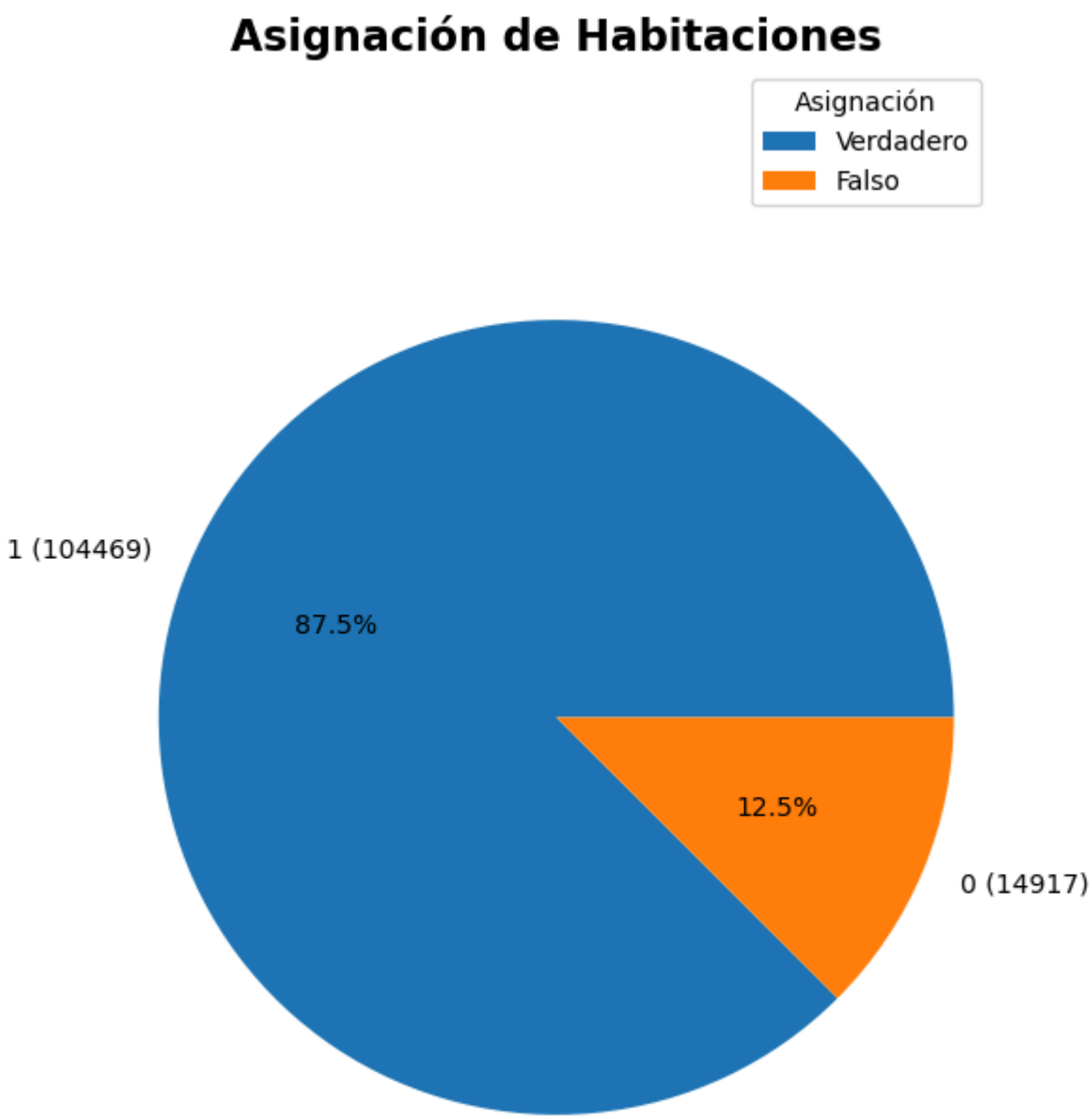
Out[ ]:

	hotel	adr	market_segment	customer_type	lead_time	arrival_date_year	reserved_room_type
0	Resort Hotel	0.0	Direct	Transient	342	2015	C
1	Resort Hotel	0.0	Direct	Transient	737	2015	C
2	Resort Hotel	75.0	Direct	Transient	7	2015	A
3	Resort Hotel	75.0	Corporate	Transient	13	2015	A
4	Resort Hotel	98.0	Online TA	Transient	14	2015	A

Ahora que tenemos la nueva columna, es momento de analizar cómo funciona la asignación de las habitaciones

In [ ]:
```
# Contando los valores
room_corr_assigned = df_revAdr['room_assigned_correctly'].value_counts()

print(room_corr_assigned)

# Variables Total y Porcentaje para nuestros gráficos
total_rca_count = room_corr_assigned.sum()
rca_percentage = (room_corr_assigned / total_rca_count) * 100

# Creamos las etiquetas mediante una función
labels = [f'{rooms} ({count})' for rooms, count in zip(room_corr_assigned.index, rc

# Tamaño del gráfico
fig, ax = plt.subplots(figsize=(6, 9))

# Generando el gráfico
plt.pie(room_corr_assigned, labels=labels, autopct='%1.1f%%')
ax.legend(['Verdadero', 'Falso'], loc='upper right', title='Asignación')
# plt.legend(title='Correctly')
ax.set_title('Asignación de Habitaciones', fontsize=16, fontweight='bold')
plt.axis('equal')
plt.show()
```

```
1    104469
0     14917
Name: room_assigned_correctly, dtype: int64
```



Analizando la asignación de habitaciones, se observa que el **87,5%** de las habitaciones están correctamente asignadas, esto significa que los clientes reciben la habitación que reservaron. Se recomienda a futuro, intentar reducir el **12,5%** de asignación incorrecta de habitaciones. Un porcentaje mayor de asignación incorrecta de habitaciones puede producir malestar en los clientes y en algún momento perderlos.

Será interesante hacer un seguimiento del **12,5%** con el fin de saber si, en ese porcentaje se produjo una mejora de habitación. Una habitación más bonita en el mayor de los casos tendremos un cliente más satisfecho.

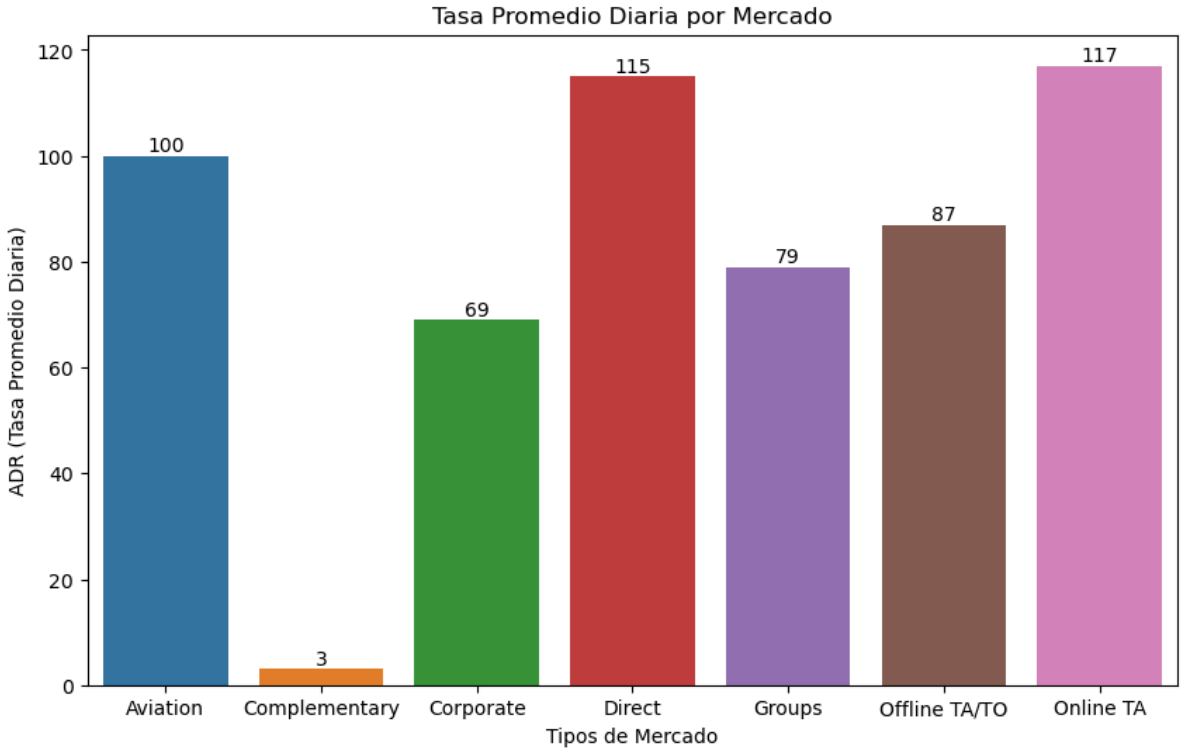
4.6.b.2 Variación ADR por los distintos Segmentos del Mercado.

In [ ]:
```
# Variación por Mercados
mean_adr_mkt = df_revAdr.groupby('market_segment')['adr'].mean().round().reset_inde
```

```
print(mean_adr_mkt)
```

```
  market_segment    adr
0       Aviation  100.0
1  Complementary    3.0
2      Corporate   69.0
3         Direct  115.0
4         Groups   79.0
5  Offline TA/TO   87.0
6      Online TA  117.0
```

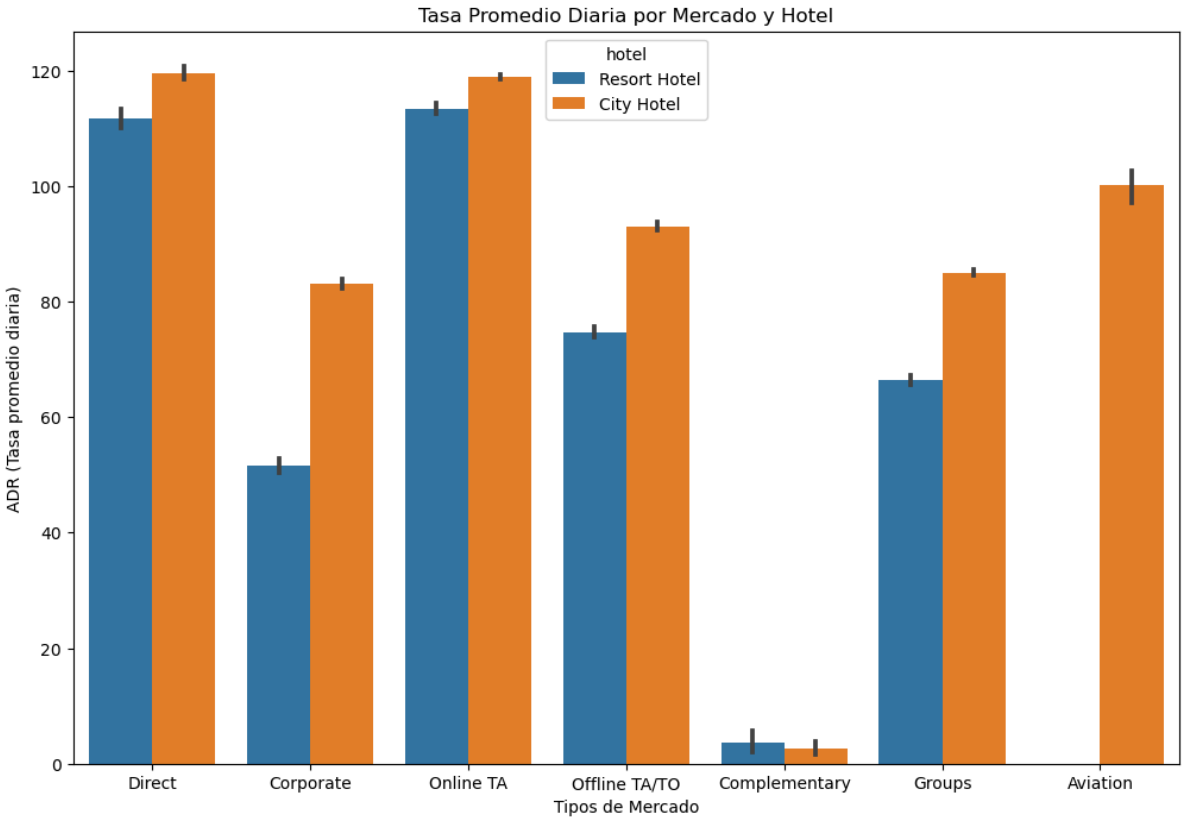
In [ ]:
```
# Creando el gráfico
plt.figure(figsize=(10, 6))

ax = sns.barplot(x='market_segment', y='adr', data=mean_adr_mkt)
ax.bar_label(ax.containers[0], fontsize=10)
plt.title('Tasa Promedio Diaria por Mercado')
plt.xlabel('Tipos de Mercado')
plt.ylabel('ADR (Tasa Promedio Diaria)')
plt.show()
```



Al analizar el ADR por segmento de mercado, se observa que el mercado *Direct*, el *Aviation*, y el *Online* son los mercados en los que el hotel recibe más reservas. En estos mercados es donde el hotel puede ofrecer ventajas especiales o dirigir las ofertas a otros segmentos para ayudarlos a crecer.

Variación de los Mercados por Hotel

In [ ]:
```
# Creando el gráfico
plt.figure(figsize=(12, 8))

ax = sns.barplot(x='market_segment', y='adr', hue='hotel', data=df_revAdr)
plt.title('Tasa Promedio Diaria por Mercado y Hotel')
plt.xlabel('Tipos de Mercado')
plt.ylabel('ADR (Tasa promedio diaria)')
plt.show()
```



El análisis de los segmentos de mercado por Hoteles, no muestra demasiada variación. Este análisis nos ayuda a crear ofertas más específicas por hotel y segmento de mercado.

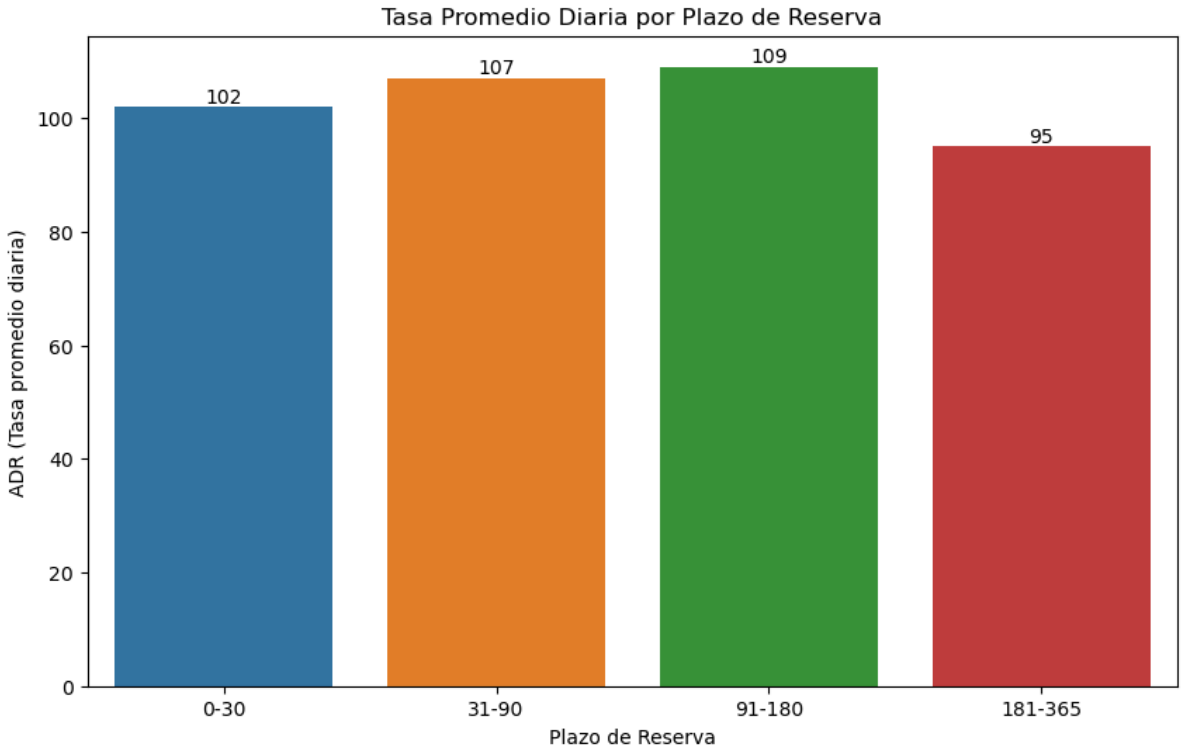
4.6.b.3 Variación ADR por el plazo de Reserva.

In [ ]:
```
# Como para este análisis poseemos demasiados valores, voy a crear conjuntos para a
# Agrupando los valores
df_revAdr['lt_enclosed'] = pd.cut(df_revAdr['lead_time'], bins=[0, 30, 90, 180, 365
mean_adr_ltbucket = df_revAdr.groupby('lt_enclosed')['adr'].mean().round().reset_ir
mean_adr_lt = df_revAdr.groupby('lead_time')['adr'].mean().round().reset_index()

print(mean_adr_lt)
print(mean_adr_ltbucket)
```

```
     lead_time   adr
0            0  83.0
1            1  90.0
2            2  94.0
3            3  93.0
4            4  95.0
..         ...   ...
474        622  62.0
475        626  63.0
476        629  62.0
477        709  68.0
478        737   0.0

[479 rows x 2 columns]
  lt_enclosed    adr
0        0-30  102.0
1       31-90  107.0
2      91-180  109.0
3     181-365   95.0
```

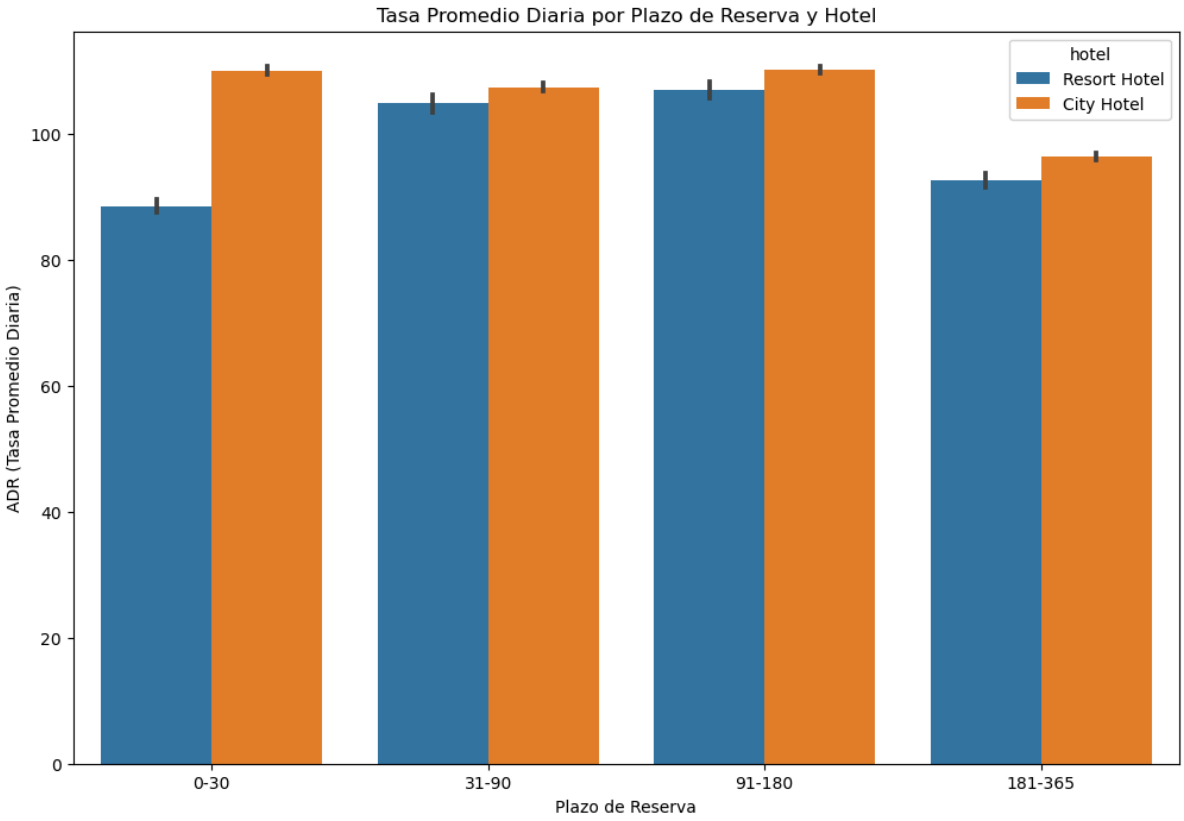
In [ ]:
```
# Creando el gráfico
plt.figure(figsize=(10, 6))
```

```
ax = sns.barplot(x='lt_enclosed', y='adr', data=mean_adr_ltbucket)
ax.bar_label(ax.containers[0], fontsize=10)
plt.title('Tasa Promedio Diaria por Plazo de Reserva')
plt.xlabel('Plazo de Reserva')
plt.ylabel('ADR (Tasa promedio diaria)')
plt.show()
```



Plazo en las Reservas por Hotel

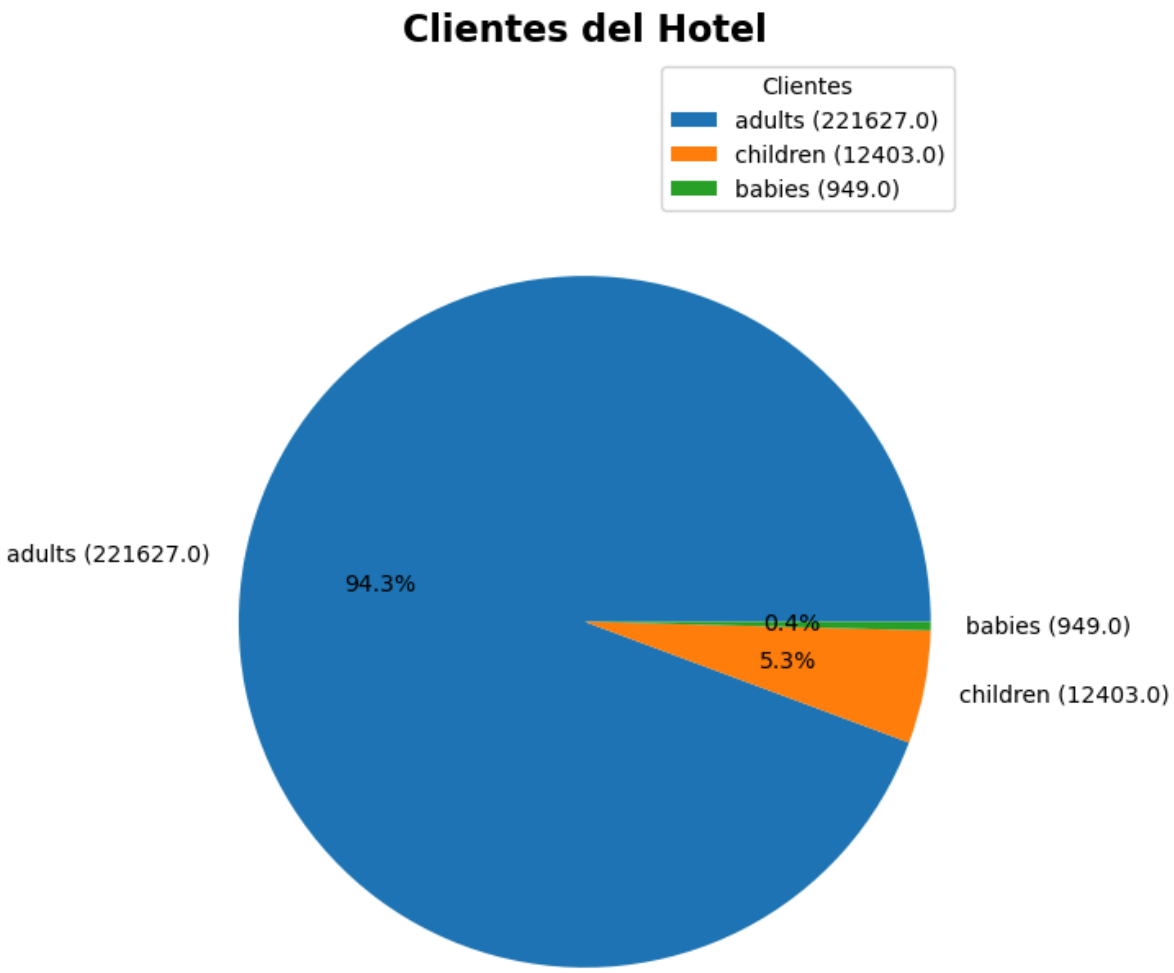
In [ ]:
```
# Creando los gráficos
plt.figure(figsize=(12, 8))

ax = sns.barplot(x='lt_enclosed', y='adr', hue='hotel', data=df_revAdr)
plt.title('Tasa Promedio Diaria por Plazo de Reserva y Hotel')
plt.xlabel('Plazo de Reserva')
plt.ylabel('ADR (Tasa Promedio Diaria)')
plt.show()
```



4.7 Segmentación de los clientes

In [ ]:
```
customer_count = df_hb[['adults', 'children', 'babies']].sum()

print(customer_count)

# Variables Total y Porcentaje para los gráficos
total_ctmr_count = customer_count.sum()
ctmr_percentage = (customer_count / total_ctmr_count) * 100

# Utilizando una función para crear las etiquetas
labels = [f'{customer} ({count})' for customer, count in zip(customer_count.index,

# Tamaño del gráfico
fig, ax = plt.subplots(figsize=(6, 9))

# Gráfico
plt.pie(customer_count, labels=labels, autopct='%1.1f%%')
plt.legend(title='Clientes', loc='upper right')
ax.set_title('Clientes del Hotel', fontsize=16, fontweight='bold')
plt.axis('equal')
plt.show()
```

```
adults      221627.0
children     12403.0
babies         949.0
dtype: float64
```



In [ ]:
```python
# Contando los países de procedencia de los clientes
country_count = df_hb['country'].value_counts()

print(country_count)

# Como se registran muchos países, vamos a recortar los valores, aquellos que posea
less_than_500_customers = 500
new_ctry_count = country_count[country_count < less_than_500_customers].sum()
country_count = country_count[country_count >= less_than_500_customers]
country_count['OTHERS'] = new_ctry_count

# Verificando que la condicion funcionó correctamente
print(country_count)

# Creamos las etiquetas para nuestro gráfico
labels = [f'{country} ({count})' for country, count in country_count.items()]

# Realizamos un gráfico de barras
fig, ax = plt.subplots(figsize=(10, 6))
ax.bar(country_count.index, country_count.values)
ax.set_xticklabels(labels, rotation=90)
ax.set_title('Clientes por País')
ax.set_xlabel('Pais')
ax.set_ylabel('Número de Clientes')
plt.tight_layout()
plt.show()
```

```
PRT    48586
GBR    12129
FRA    10415
ESP     8568
DEU     7287
       ...
DJI        1
BWA        1
HND        1
VGB        1
NAM        1
Name: country, Length: 178, dtype: int64
PRT       48586
GBR       12129
FRA       10415
ESP        8568
DEU        7287
ITA        3766
IRL        3375
BEL        2342
BRA        2224
NLD        2104
USA        2097
CHE        1730
CN         1279
AUT        1263
SWE        1024
CHN         999
POL         919
ISR         669
RUS         632
NOR         607
ROU         500
OTHERS     6871
Name: country, dtype: int64
```