1. Visualización de datos

En esta sesión abordaremos técnicas de visualización de datos usando Matplotlib y Seaborn. Exploraremos varias visualizaciones y las aplicaremos en un estudio de caso para tomar decisiones basadas en datos.

Objetivos

Los objetivos de esta sesión son:

- Aprender a utilizar técnicas avanzadas de visualización de datos con Matplotlib y Seaborn
- Aplicarlas en un estudio de caso para la toma de decisiones basada en indicadores clave de desempeño (KPIs)

2. Importación de Librerías Necesarias

In [1]:
```
import pandas as pd
# Importamos matplotlib.pyplot para crear gráficos
import matplotlib.pyplot as plt
# Importamos seaborn para crear gráficos avanzados y estilizados
import seaborn as sns

# Configuramos el estilo de los gráficos para que tengan una apariencia limpia
sns.set(style="whitegrid")
# Configuramos el tamaño predeterminado de los gráficos
plt.rcParams['figure.figsize'] = (12, 6)
```

3. Carga y Preparación de Datos

In [2]:
```
data = sns.load_dataset('tips')

# Mostrando las primeras filas del dataset
data.head()
```

Out[2]:

	total_bill	tip	sex	smoker	day	time	size
0	16.99	1.01	Female	No	Sun	Dinner	2
1	10.34	1.66	Male	No	Sun	Dinner	3
2	21.01	3.50	Male	No	Sun	Dinner	3
3	23.68	3.31	Male	No	Sun	Dinner	2
4	24.59	3.61	Female	No	Sun	Dinner	4

4. Visualizaciones Básicas con Matplotlib

4.1. Gráfico de Barras

¿Por qué elegir un gráfico de barras?

- El gráfico de barras es ideal para comparar valores medios o totales entre diferentes categorías.
- Es fácil de interpretar y útil para mostrar las diferencias en una sola variable categórica.

Pros:

- Claridad: Fácil de leer y comparar categorías.
- Versatilidad: Puede utilizarse para una amplia gama de datos categóricos.
- Interpretación directa: Muestra diferencias claras entre categorías.

Contras:

- No muestra tendencias: No es adecuado para visualizar cómo cambian los valores a lo largo del tiempo.
- Puede volverse desordenado si hay demasiadas categorías.

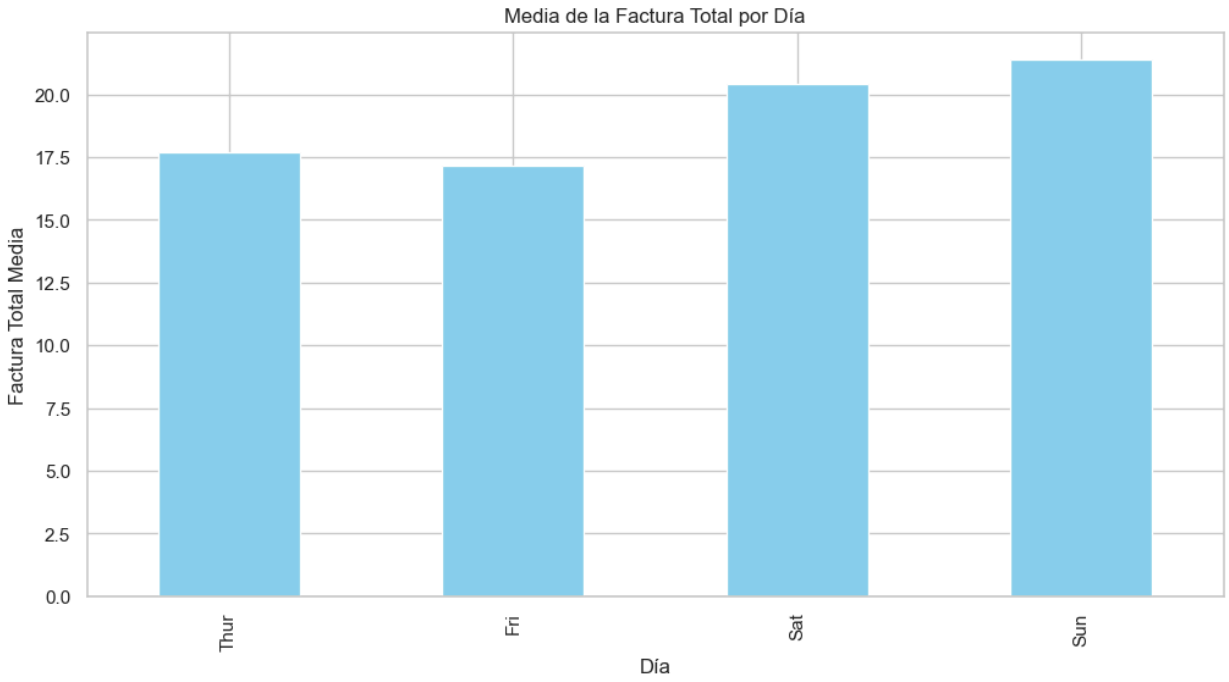
In [3]:
```
#Agrupamos los datos por día de la semana y calculamos la media de la factura total po
data.groupby('day')['total_bill'].mean().plot(kind='bar', color='skyblue')

# Añadimos un título al gráfico
plt.title('Media de la Factura Total por Día')

# Etiquetamos el eje X
plt.xlabel('Día')

# Etiquetamos el eje Y
plt.ylabel('Factura Total Media')

# Mostramos el gráfico
plt.show()
```



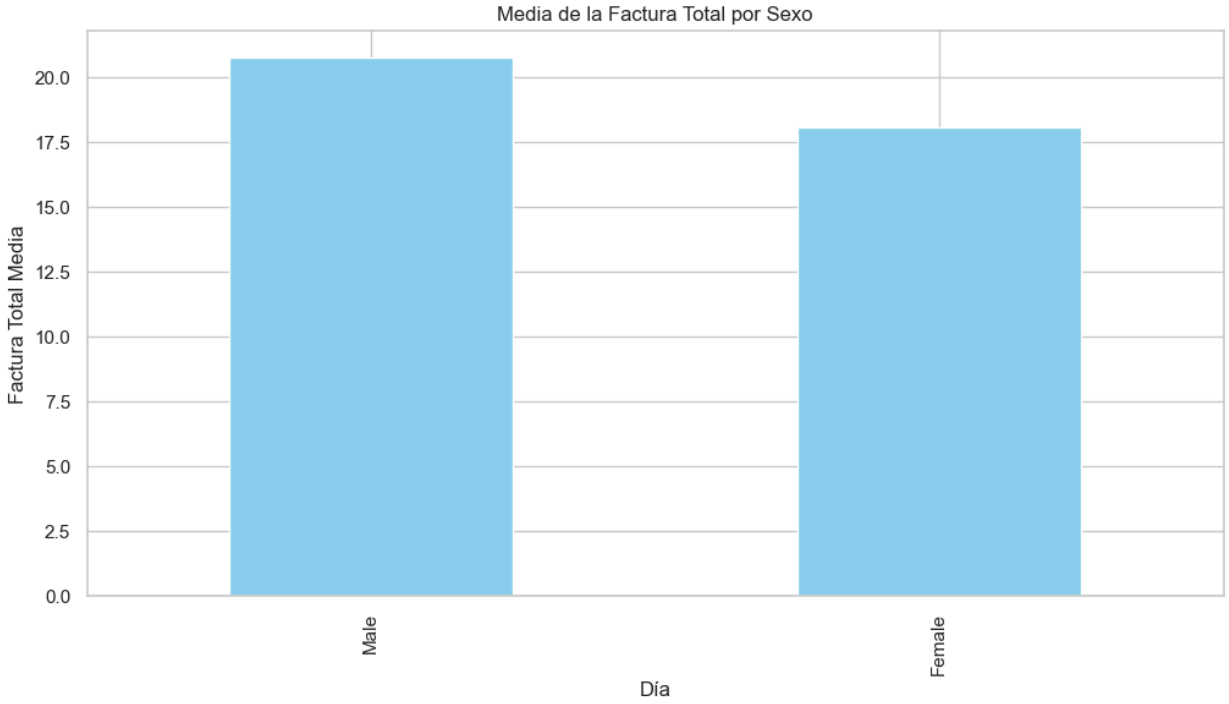
In [4]:
```
#Agrupamos los datos por día de la semana y calculamos la media de la factura total po
data.groupby('sex')['total_bill'].mean().plot(kind='bar', color='skyblue')

# Añadimos un título al gráfico
plt.title('Media de la Factura Total por Sexo')

# Etiquetamos el eje X
plt.xlabel('Día')

# Etiquetamos el eje Y
plt.ylabel('Factura Total Media')

# Mostramos el gráfico
plt.show()
```



4.2. Gráfico de Líneas

¿Por qué elegir un gráfico de líneas?

- El gráfico de líneas es ideal para mostrar cómo un conjunto de datos cambia con el tiempo.
- Es útil para identificar tendencias y patrones en series temporales.

Pros:

- Claridad en la tendencia: Muestra claramente cómo una variable cambia con el tiempo.
- Conexión de datos: Las líneas conectan puntos, haciendo evidente la evolución de los datos.
- Eficiencia en series temporales: Es excelente para series temporales y cambios continuos.

Contras:

- No es adecuado para datos categóricos no secuenciales: Si las categorías no están en un orden lógico, un gráfico de líneas puede ser engañoso.
- Puede ser confuso con muchas líneas: Si se grafican demasiadas series en el mismo gráfico, puede volverse difícil de interpretar.

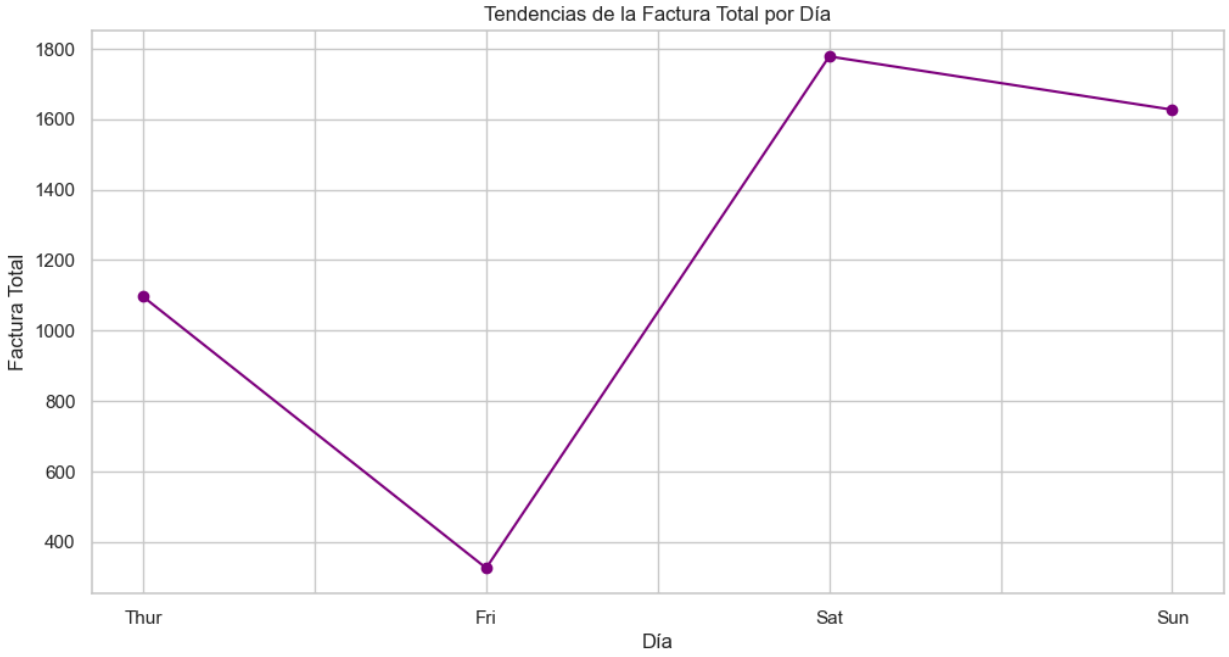
In [5]:
```
# Agrupamos los datos por día de la semana y sumamos las facturas totales por día
# Creamos un gráfico de líneas para visualizar la tendencia de la factura total a lo l
data.groupby('day')['total_bill'].sum().plot(kind='line', marker='o', color='purple')

# Añadimos un título al gráfico
plt.title('Tendencias de la Factura Total por Día')

# Etiquetamos el eje X
plt.xlabel('Día')

# Etiquetamos el eje Y
plt.ylabel('Factura Total')

# Mostramos el gráfico
plt.show()
```



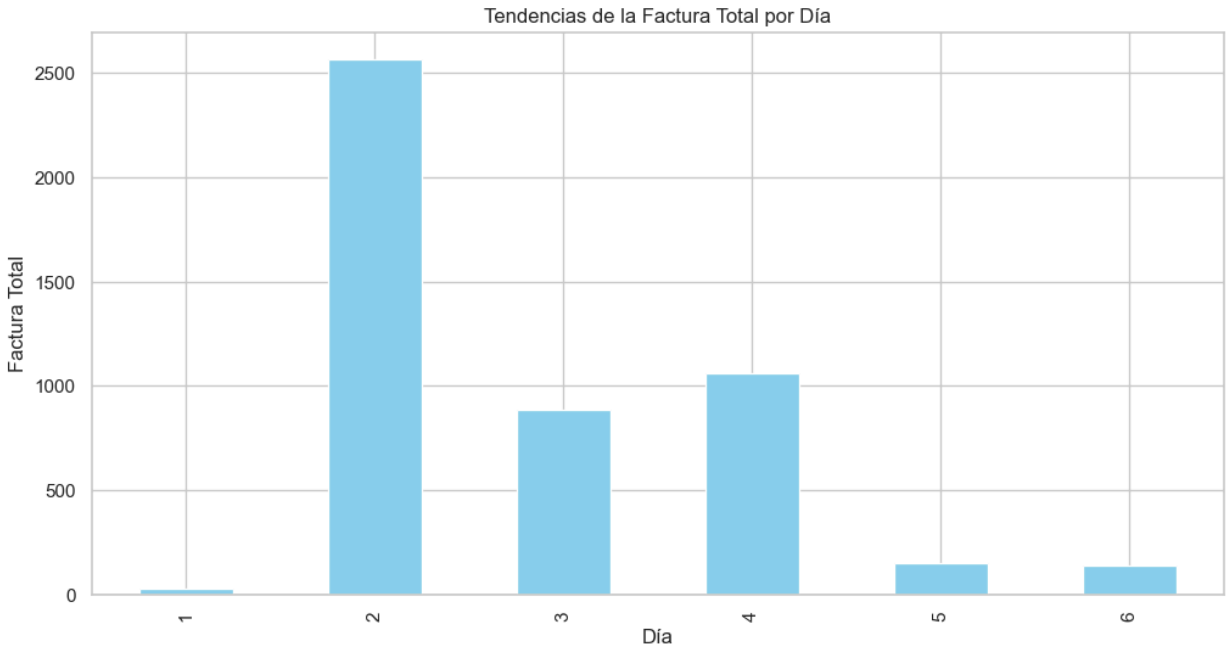
In [6]:
```
# Agrupamos los datos por día de la semana y sumamos las facturas totales por día
# Creamos un gráfico de líneas para visualizar la tendencia de la factura total a lo l
data.groupby('size')['total_bill'].sum().plot(kind='bar', color='skyblue')

# Añadimos un título al gráfico
plt.title('Tendencias de la Factura Total por Día')

# Etiquetamos el eje X
plt.xlabel('Día')

# Etiquetamos el eje Y
plt.ylabel('Factura Total')

# Mostramos el gráfico
plt.show()
```



5. Visualizaciones Avanzadas con Seaborn

5.1. Gráfico de Cajas (Box Plot)

¿Por qué elegir un gráfico de cajas?

- El gráfico de cajas es ideal para mostrar la distribución de un conjunto de datos a través de sus cuartiles, resaltando la mediana, los valores atípicos y la variabilidad.

Pros:

- Resumen estadístico claro: Proporciona un resumen visual del rango, mediana y valores atípicos de los datos.
- Comparación fácil: Es excelente para comparar la distribución entre diferentes categorías.
- Detección de valores atípicos: Destaca claramente los valores atípicos en los datos.

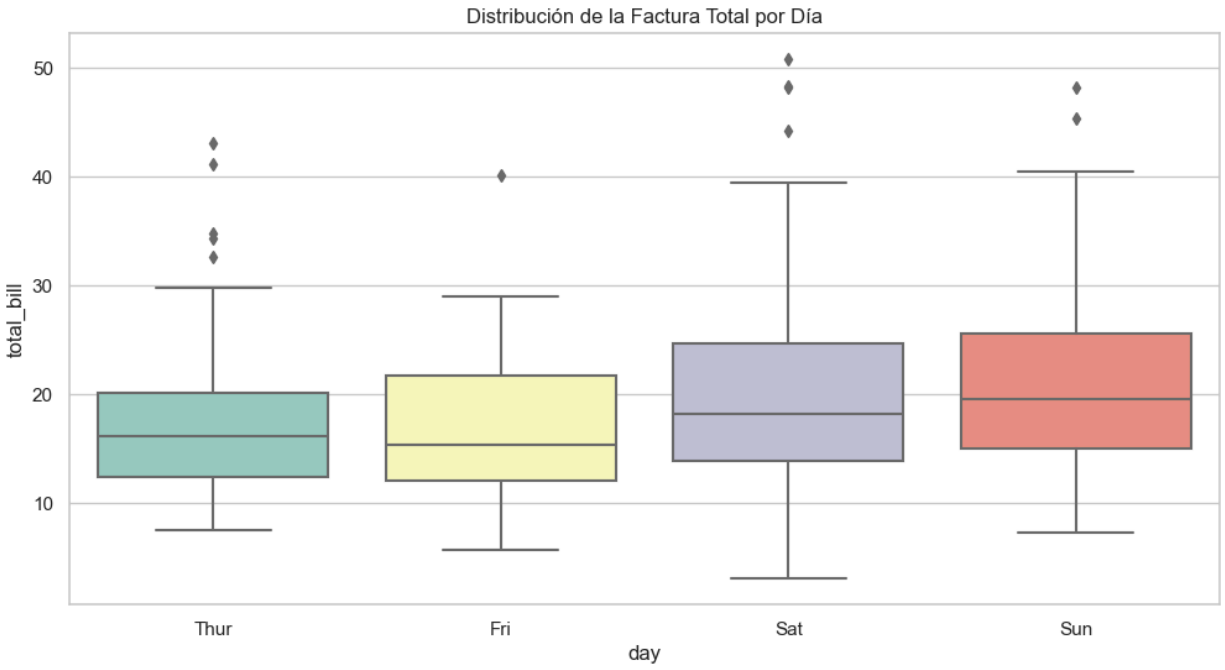
Contras:

- No muestra la forma de la distribución: No proporciona información detallada sobre la forma exacta de la distribución, como un histograma.
- Interpretación puede ser compleja: Para aquellos no familiarizados con el concepto de cuartiles, puede ser un poco complicado interpretar.

In [7]:
```
# Creamos un gráfico de cajas para visualizar la distribución de la factura total por
sns.boxplot(x='day', y='total_bill', data=data, palette='Set3')

# Añadimos un título al gráfico
plt.title('Distribución de la Factura Total por Día')

# Mostramos el gráfico
plt.show()
```



5.2. Mapa de Calor (Heatmap)

¿Por qué elegir un mapa de calor?

- El mapa de calor es excelente para visualizar la matriz de correlación entre diferentes variables, mostrando qué variables están altamente correlacionadas entre sí.

Pros:

- Visualización de relaciones: Permite observar visualmente las relaciones entre múltiples variables.
- Compacto: Puede representar una gran cantidad de datos en un espacio reducido.
- Fácil identificación de patrones: Es fácil de identificar patrones y relaciones a simple vista.

Contras:

- No muestra la naturaleza de la relación: Mientras que muestra la fuerza de la correlación, no indica si la relación es positiva o negativa.
- Puede volverse confuso con demasiadas variables: Si hay demasiadas variables, el mapa de calor puede volverse difícil de interpretar.

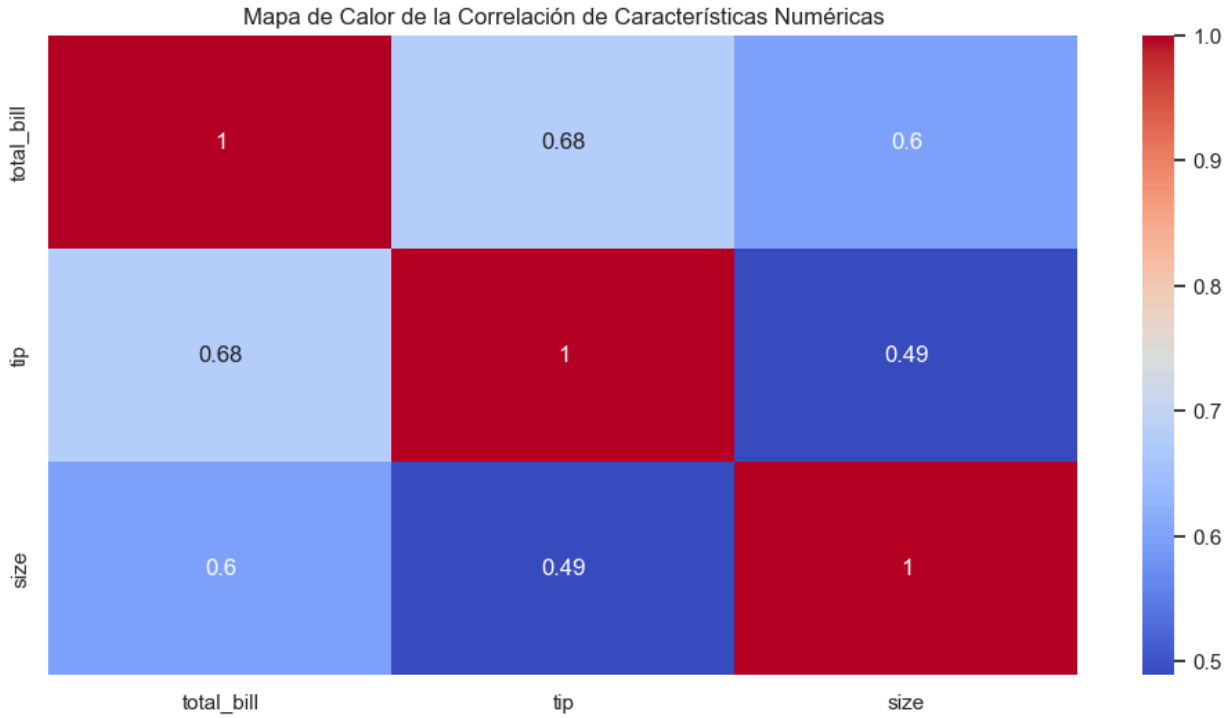
In [8]:
```
# Calculamos la matriz de correlación entre las variables numéricas del dataset
corr = data.corr()

# Creamos un mapa de calor para visualizar las correlaciones entre las diferentes vari
sns.heatmap(corr, annot=True, cmap='coolwarm')

# Añadimos un título al gráfico
plt.title('Mapa de Calor de la Correlación de Características Numéricas')
```

```
# Mostramos el gráfico
plt.show()
```

```
C:\Users\luism\AppData\Local\Temp\ipykernel_19252\3620108139.py:2: FutureWarning: The default value of numeric_only in DataFrame.corr is deprecated. In a future version, it will default to False. Select only valid columns or specify the value of numeric_only to silence this warning.
  corr = data.corr()
```



5.3. Diagrama Sankey (Sankey Diagram)

¿Por qué elegir un diagrama Sankey?

- El diagrama Sankey es excelente para visualizar flujos y proporciones entre diferentes elementos en un proceso o sistema.
- Es especialmente útil para mostrar cómo se distribuyen o transfieren los recursos o cantidades entre diferentes estados.

Pros:

- Visualización de flujos: Muestra claramente cómo los valores se distribuyen o transfieren entre diferentes categorías.
- Proporciones evidentes: Las proporciones entre los diferentes flujos son fácilmente visibles.
- Ideal para procesos complejos: Excelente para entender procesos con múltiples etapas o categorías.

Contras:

- Puede ser complicado de leer: En procesos muy complejos con muchas categorías, puede volverse confuso.

- Difícil de crear manualmente: Requiere herramientas o bibliotecas específicas para ser creado adecuadamente.

In [9]:
```python
# Para crear un diagrama Sankey en Python, utilizaremos la biblioteca plotly
import plotly.graph_objects as go

# Definimos las etiquetas de las categorías
labels = ['Clientes', 'Almuerzo', 'Cena', 'Total Factura Baja', 'Total Factura Alta']

# Definimos las fuentes y destinos de los flujos
source = [0, 0, 1, 1, 2, 2]  # De dónde vienen los flujos
target = [1, 2, 3, 4, 3, 4]  # A dónde van los flujos

# Definimos el valor de cada flujo
value = [8, 4, 2, 6, 3, 1]

# Creamos el diagrama Sankey
fig = go.Figure(data=[go.Sankey(
    node=dict(
        pad=15,
        thickness=20,
        line=dict(color="black", width=0.5),
        label=labels
    ),
    link=dict(
        source=source,
        target=target,
        value=value
    )
)])

# Añadimos un título al gráfico
fig.update_layout(title_text="Diagrama Sankey de Facturación por Tipo de Cliente", for

# Mostramos el gráfico
fig.show()
```

6. Estudio de Caso: Visualización de Indicadores para la Toma de Decisiones

6.1. Contexto y Planteamiento del Problema

En este estudio de caso, asumimos el rol de un analista de datos en un restaurante que busca optimizar su operación diaria. El objetivo es ayudar a la dirección a tomar decisiones informadas sobre la asignación de personal y la gestión del menú, basándonos en los patrones de gasto de los clientes durante el día y la noche, y cómo estos patrones afectan las ventas y las propinas.

Objetivo del negocio: Optimizar la asignación de personal y ajustar los precios del menú para maximizar la rentabilidad durante las horas pico.

Problema específico:

- ¿Cuándo es el momento más rentable del día y qué estrategia de personal y precios debería implementar el restaurante para maximizar las ganancias?

6.2. Proceso de Análisis para la Toma de Decisiones

- Recopilación de datos: Utilizaremos un conjunto de datos que contiene información sobre las facturas de los clientes, el momento del día (almuerzo o cena), y las propinas dejadas.
- Análisis preliminar: Realizaremos visualizaciones para identificar patrones de gasto durante las diferentes horas del día.
- Generación de insights: Extraeremos insights de las visualizaciones para ayudar a la dirección a tomar decisiones sobre la asignación de personal y ajustes en los precios del menú.

In [10]:
```
df = sns.load_dataset('tips')
data.info()
```

```
<class 'pandas.core.frame.DataFrame'>
RangeIndex: 244 entries, 0 to 243
Data columns (total 7 columns):
 #   Column      Non-Null Count  Dtype   
---  ------      --------------  -----   
 0   total_bill  244 non-null    float64 
 1   tip         244 non-null    float64 
 2   sex         244 non-null    category
 3   smoker      244 non-null    category
 4   day         244 non-null    category
 5   time        244 non-null    category
 6   size        244 non-null    int64   
dtypes: category(4), float64(2), int64(1)
memory usage: 7.4 KB
```

In [11]:
```
df.describe()
```

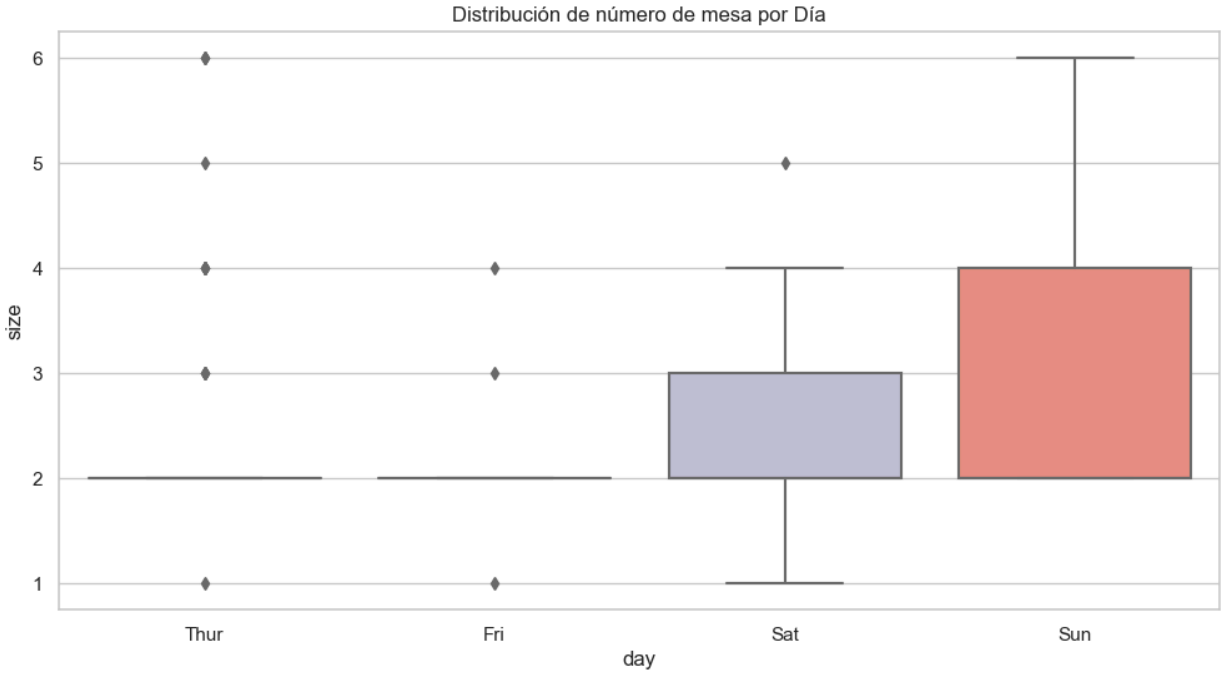
Out[11]:

	total_bill	tip	size
count	244.000000	244.000000	244.000000
mean	19.785943	2.998279	2.569672
std	8.902412	1.383638	0.951100
min	3.070000	1.000000	1.000000
25%	13.347500	2.000000	2.000000
50%	17.795000	2.900000	2.000000
75%	24.127500	3.562500	3.000000
max	50.810000	10.000000	6.000000

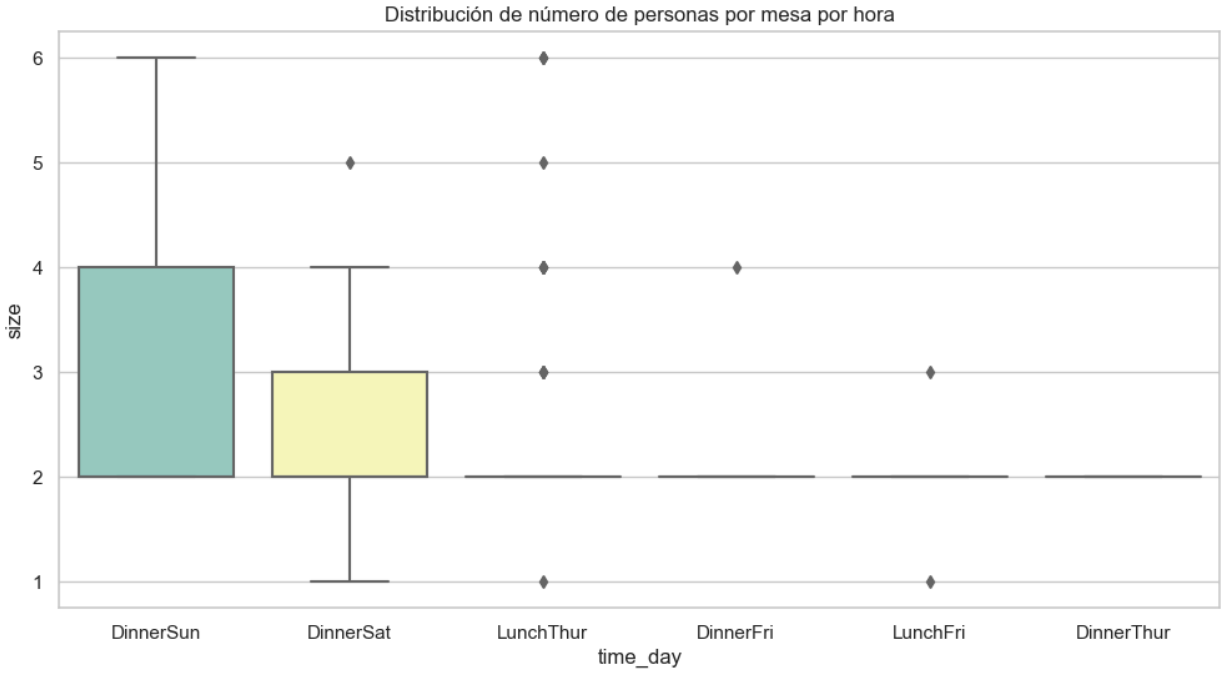
In [12]:
```
# ver número de personas = size
# ver ticket promedio por time y día
# segmento gasta más
# cuando van más los clientes que gastan más
# relación de propina / ticket por día y hora
# Comparar gráficas
```

Número de personas por día y hora

In [13]:
```
sns.boxplot(x = 'day', y = 'size', data = df, palette = 'Set3')
plt.title('Distribución de número de mesa por Día')
plt.show()
```



In [14]:
```
df['time_day'] = df['time'].astype(str) + df['day'].astype(str)
sns.boxplot(x = 'time_day', y = 'size', data = df, palette = 'Set3')
plt.title('Distribución de número de personas por mesa por hora')
plt.show()
```



In [15]:
```
# Agrupamos los datos por 'time_day' y 'size' y contamos el número de mesas para cada
df.groupby(['time_day', 'size']).size().unstack().plot(kind='bar', cmap = 'Set1')
```

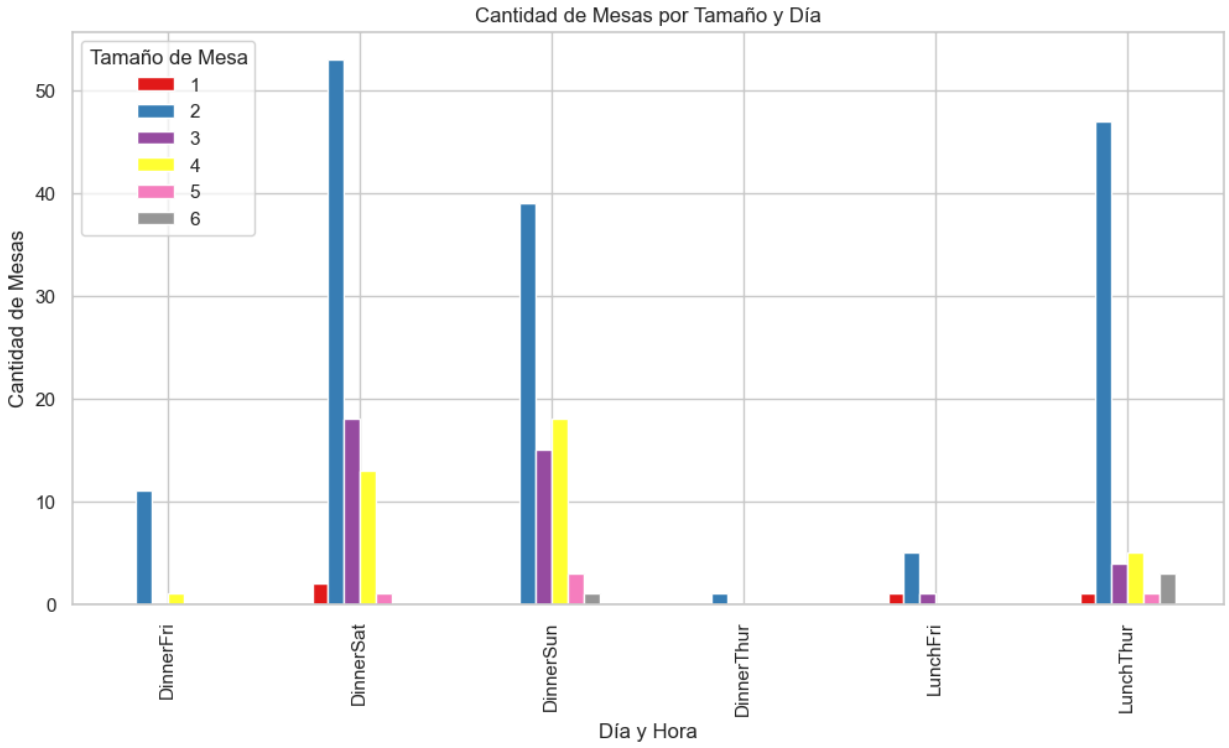
```
# Añadimos un título al gráfico
plt.title('Cantidad de Mesas por Tamaño y Día')

# Etiquetamos el eje X
plt.xlabel('Día y Hora')

# Etiquetamos el eje Y
plt.ylabel('Cantidad de Mesas')

# Añadimos una leyenda
plt.legend(title='Tamaño de Mesa')

# Mostramos el gráfico
plt.show()
```



Ticket promedio por día y hora

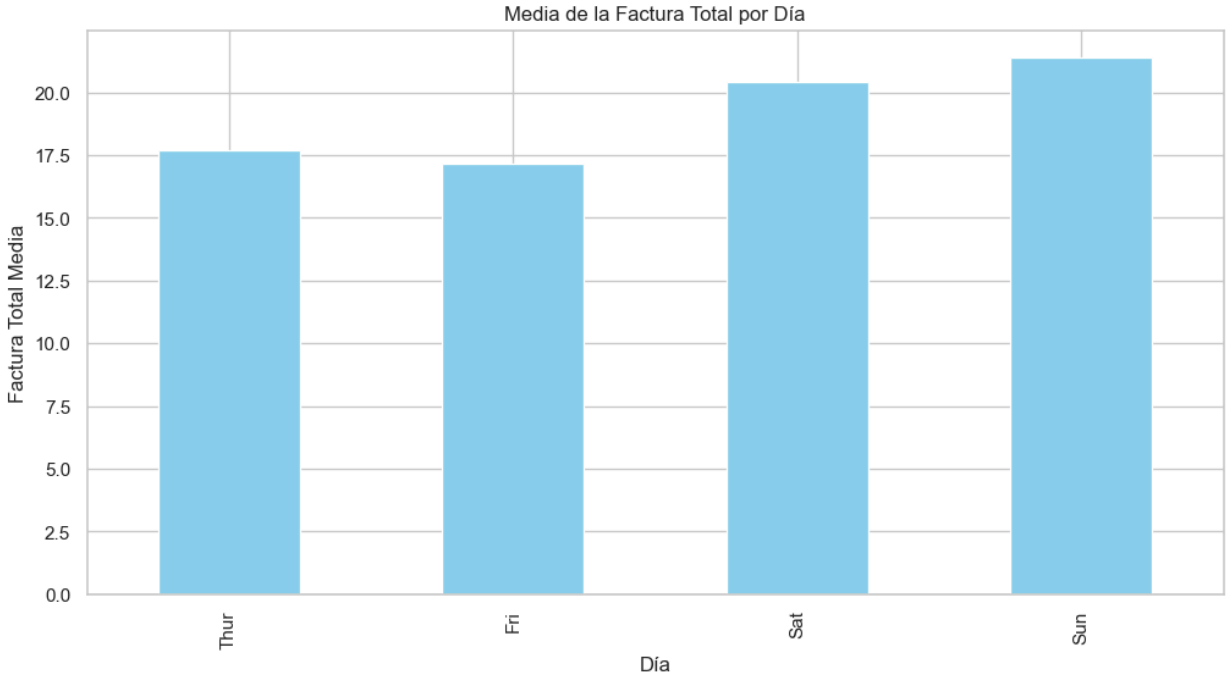
In [16]:
```
#Agrupamos los datos por día de la semana y calculamos la media de la factura total po
df.groupby('day')['total_bill'].mean().plot(kind='bar', color='skyblue')

# Añadimos un título al gráfico
plt.title('Media de la Factura Total por Día')

# Etiquetamos el eje X
plt.xlabel('Día')

# Etiquetamos el eje Y
plt.ylabel('Factura Total Media')

# Mostramos el gráfico
plt.show()
```



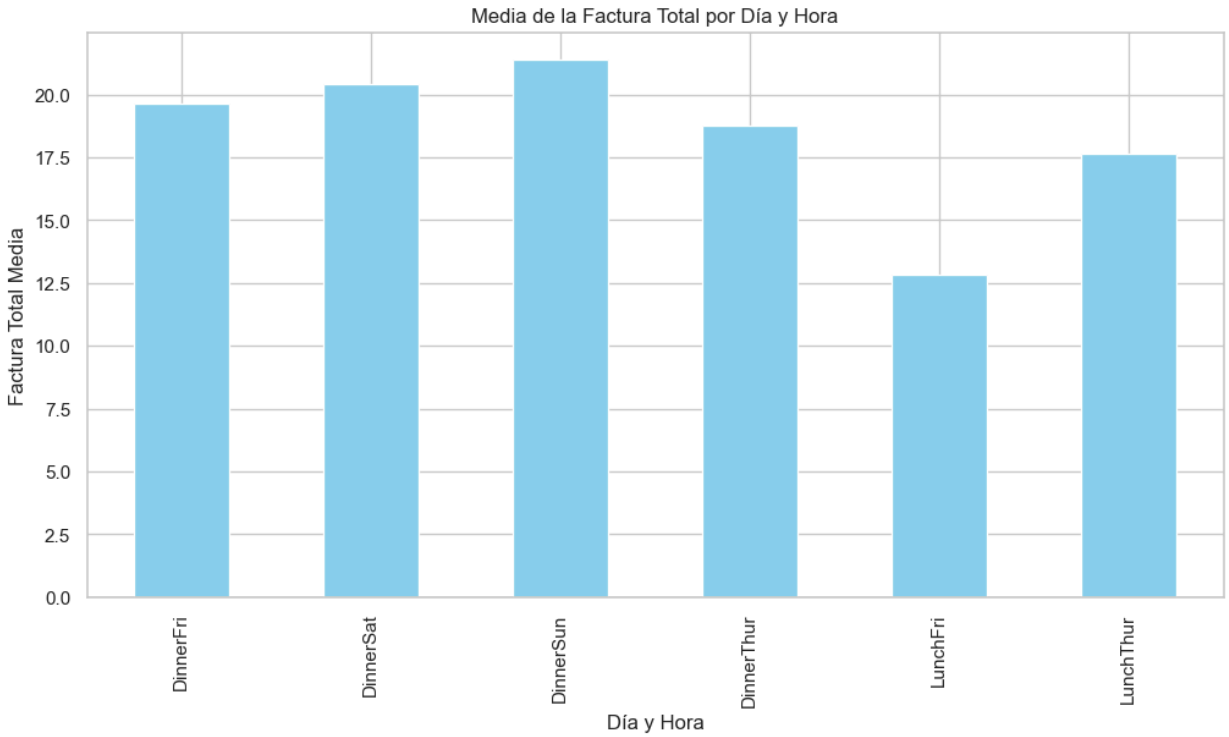
In [17]:
```
#Agrupamos los datos por día de la semana y calculamos la media de la factura total po
df.groupby('time_day')['total_bill'].mean().plot(kind='bar', color='skyblue')

# Añadimos un título al gráfico
plt.title('Media de la Factura Total por Día y Hora')

# Etiquetamos el eje X
plt.xlabel('Día y Hora')

# Etiquetamos el eje Y
plt.ylabel('Factura Total Media')

# Mostramos el gráfico
plt.show()
```



Segmento que más gasta

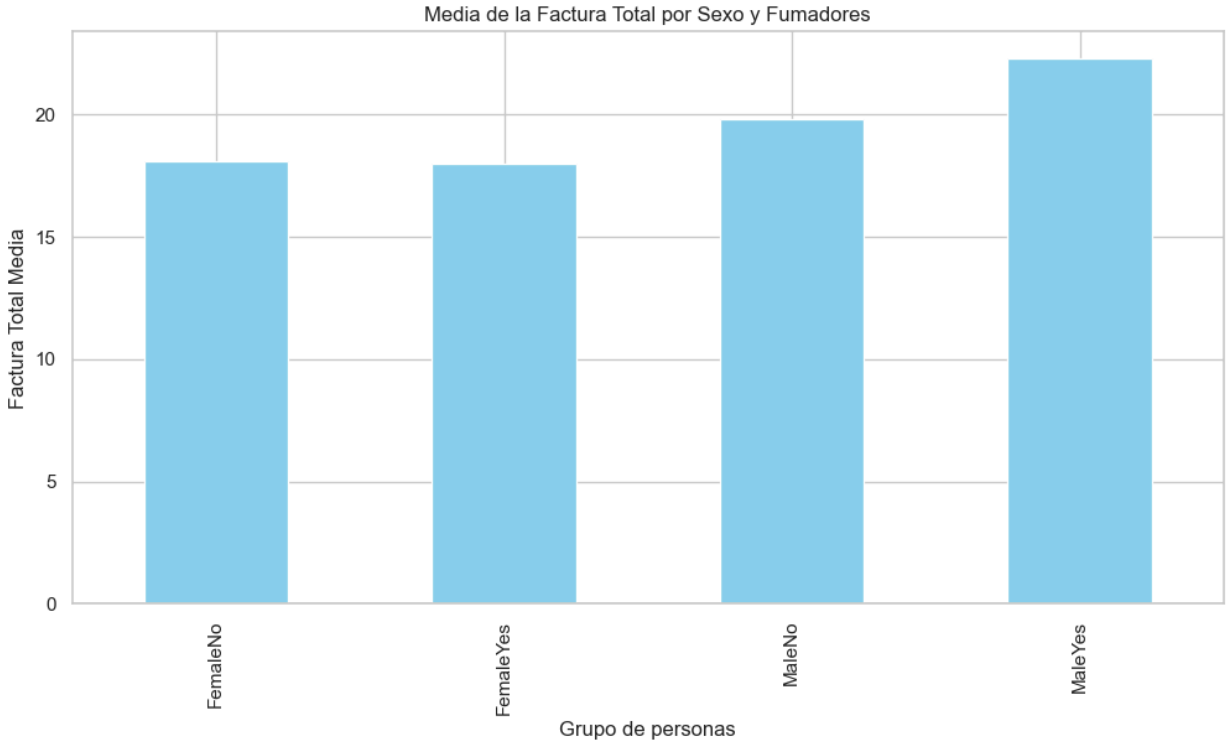
In [18]:
```
df['sex_smoker'] = df['sex'].astype(str) + df['smoker'].astype(str)
#Agrupamos los datos por día de la semana y calculamos la media de la factura total po
df.groupby('sex_smoker')['total_bill'].mean().plot(kind='bar', color='skyblue')

# Añadimos un título al gráfico
plt.title('Media de la Factura Total por Sexo y Fumadores')

# Etiquetamos el eje X
plt.xlabel('Grupo de personas')

# Etiquetamos el eje Y
plt.ylabel('Factura Total Media')

# Mostramos el gráfico
plt.show()
```



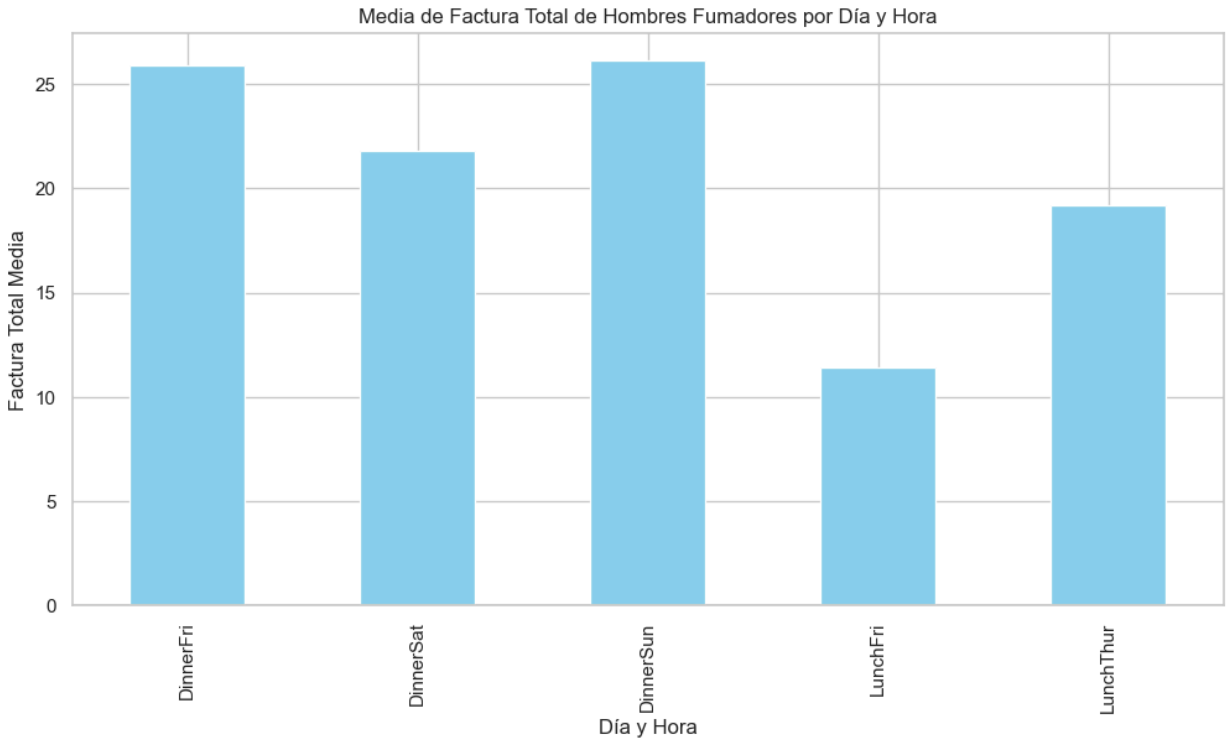
In [19]:
```
df_2 = df.query("sex_smoker == 'MaleYes'")
df_2.groupby('time_day')['total_bill'].mean().plot(kind='bar', color = 'skyblue')

# Añadimos un título al gráfico
plt.title('Media de Factura Total de Hombres Fumadores por Día y Hora')

# Etiquetamos el eje X
plt.xlabel('Día y Hora')

# Etiquetamos el eje Y
plt.ylabel('Factura Total Media')

# Mostramos el gráfico
plt.show()
```



Cuando van más los clientes que más gastan en promedio (Hombres fumadores)

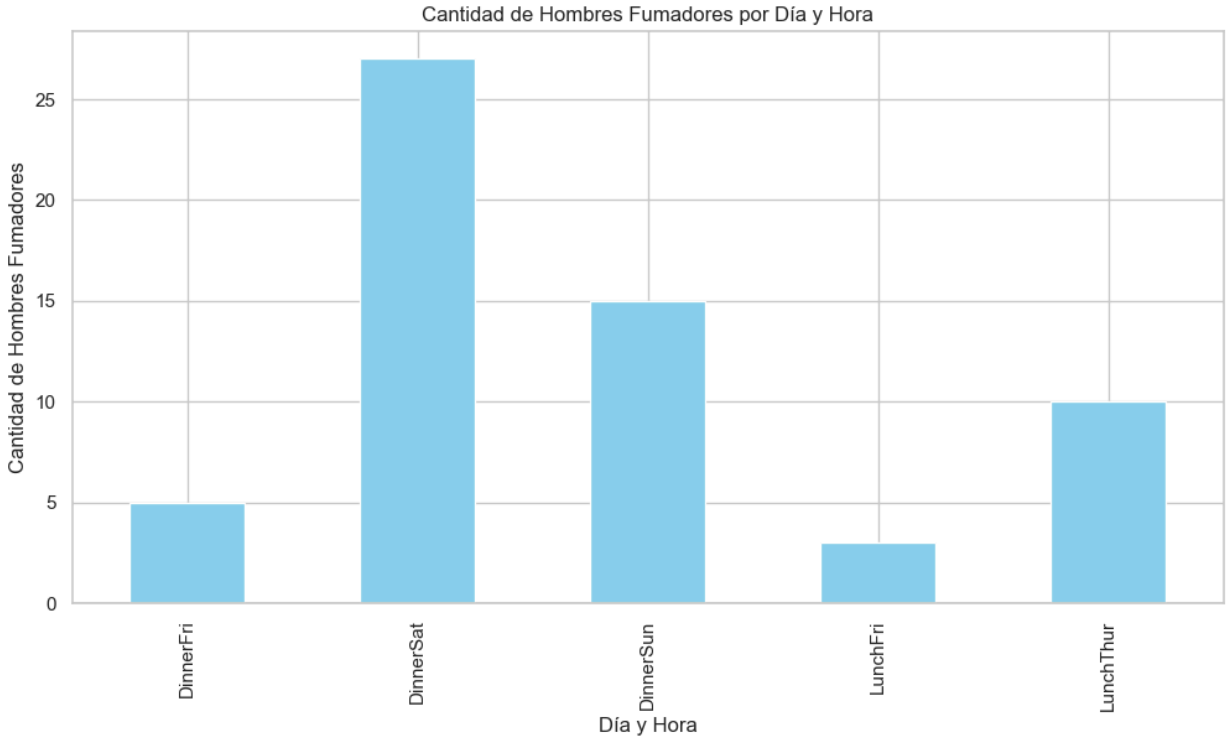
In [20]:
```
df_2.groupby('time_day')['sex_smoker'].count().plot(kind='bar', color = 'skyblue')

# Añadimos un título al gráfico
plt.title('Cantidad de Hombres Fumadores por Día y Hora')

# Etiquetamos el eje X
plt.xlabel('Día y Hora')

# Etiquetamos el eje Y
plt.ylabel('Cantidad de Hombres Fumadores')

# Mostramos el gráfico
plt.show()
```



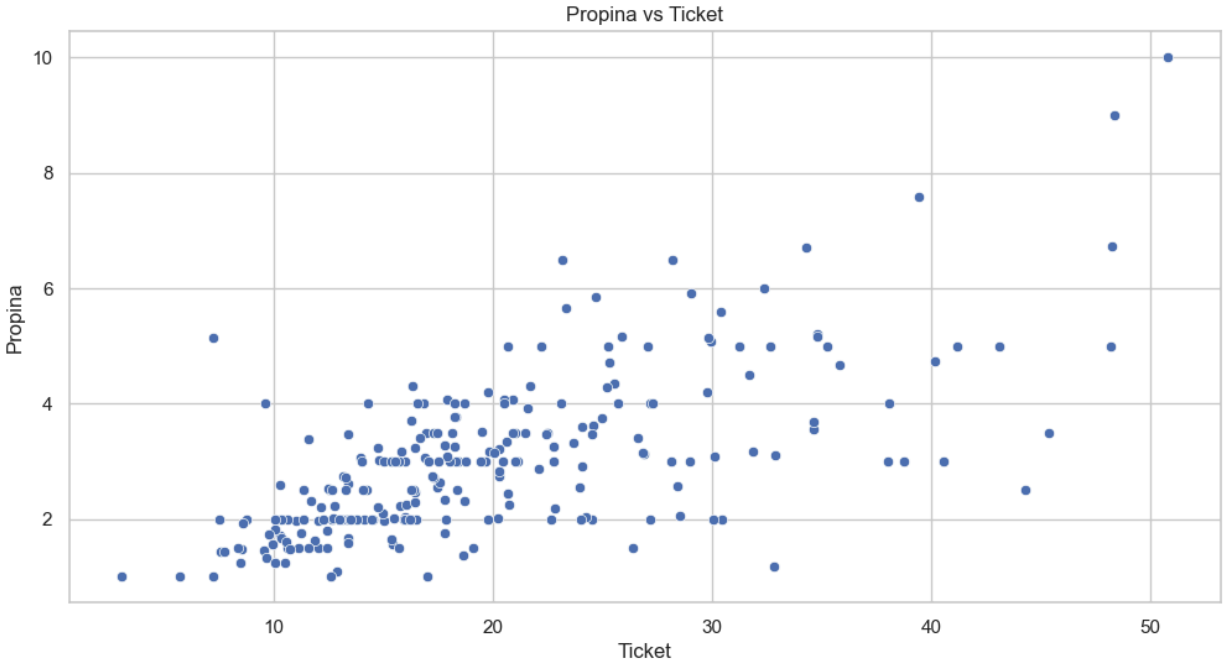
Relación propina / ticket por día y hora

In [21]:
```
sns.scatterplot(x = df.total_bill, y = df.tip)
# Añadimos un título al gráfico
plt.title('Propina vs Ticket')

# Etiquetamos el eje X
plt.xlabel('Ticket')

# Etiquetamos el eje Y
plt.ylabel('Propina')

# Mostramos el gráfico
plt.show()
```



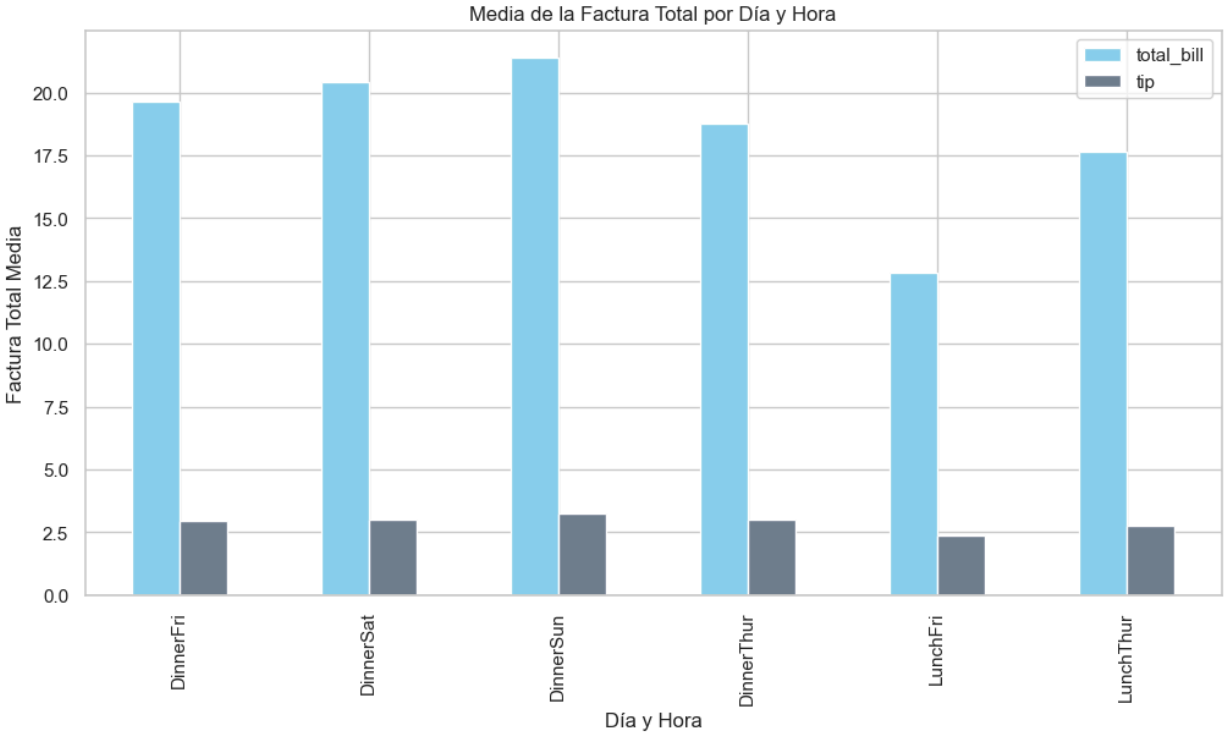
In [22]:
```
#Agrupamos los datos por día de la semana y calculamos la media de la factura total po
df.groupby('time_day')[['total_bill', 'tip']].mean().plot(kind='bar', color = ['skyblu

# Añadimos un título al gráfico
plt.title('Media de la Factura Total por Día y Hora')

# Etiquetamos el eje X
plt.xlabel('Día y Hora')

# Etiquetamos el eje Y
plt.ylabel('Factura Total Media')

# Mostramos el gráfico
plt.show()
```



7. Reporte de Laboratorio 3: Resumen

Dado el proceso que hicimos espero que suban un el reporte de laboratorio con los pasos que desarrollamos y una conclusión sobre los resultados y en que caso usar ciertas visualizaciones.

9. Quiz siguiente clase

Prepárense para el quiz:

1. Media, Varianza
2. Correlación
3. Visualizaciones

Conclusiones

Después de haber analizado las diferentes gráficas realizadas se puede llegar a varias conclusiones que son importantes para el restaurante. Lo primero es identificar el momento más rentable del día, que fue las cenas en domingo ya que cuenta con el ticket promedio más alto, seguido de las cenas en sábado. Esto es útil ya que nos permite saber que días esperamos tener los tickets más altos en la semana. Además analizando la distribución de personas por día y hora se puede observar que lo más común son las mesas de dos personas y que en sábado y domingo hay más variedad en el tamaño de mesas y más gente visita el restaurante.

Viendo esta distribución de personas vemos que el viernes en almuerzo y en la cena es cuando menos personas van al restaurante y que viernes en el almuerzo tiene el ticket promedio más bajo por lo que el viernes es uno de los días menos rentables y más tranquilos para el restaurante por lo que los viernes el restaurante puede asignar menos personal para atender a los clientes.

Cuando analizamos los segmentos de clientes vemos que los hombres que fuman es nuestro segemento que más gasta en promedio y que cuando más gastan en promedio es los domingos, por lo qu ees importante prestar atención a este segmento. También podemos ver que este grupo cuando más va es en sábado, sin embargo, los domingod están dispuestos a gastar más. Nuestros tickets promedio más altos son en sábado y domingo que es cuando más gente va al restaurante, así como, ser cuando más hombres que fuman atienden que es el cliente que más gasta por lo que sería una buena estrategia colocar más pesonal esos dos días.

Los jueves en el almuerzo sorpresivamente va una buena cantidad de personas por lo que sería bueno mantener un buen número de personal a esa hora y reducirlo un poco en la cena que van pocas personas,además de no tener un ticket promedio alto.

Por último si analizamos la gráfica de puntos vemos que hay cierta relación entre a mayor ticket la propina suele ser mayor, si bien no es una ndencia muy fuerte si existe una relación. Además si vemos ticket promedio por día y hora contra la propina vemos que en promedio la propina es mayor si es mayor el ticket promedio. Además el segemnto que más gasta tiende a estar dispuesto a gastar más los domingos y que en general tiene un ticket promedio más alto este día por lo que subir los precios los domingos es una buena estrategia, ya que las personas tienden de forma natural a estar dispuestas sa gastar más por lo uqe subir precios aumentaría ganancias y el tamaño de cuentas por lo que las propinas tenderían a ser mayores ese día.