Install & import the dependencies

In [17]:
```
import sys
!{sys.executable} -m pip install fpdf2 -q
!{sys.executable} -m pip install pandas -q
!{sys.executable} -m pip install plotly-express -q
!{sys.executable} -m pip install kaleido -q
```

In [18]:
```
from datetime import date
from pathlib import Path
import sqlite3
import pandas as pd
import plotly.express as px
from fpdf import FPDF # pip install fpdf2
```

Define paths and chart style

In [19]:
```
# Define the plotly template. Some other examples:
# "plotly", "ggplot2", "seaborn", "simple_white", "plotly_dark", "plotly_white",
plotly_template = "presentation"
```

In [44]:
```
database_path = Path("C:\\Users\\ceitf\\Desktop\\Sales\\sales.db")


# Modify the current_dir variable to use the database_path directly
current_dir = database_path.parent

output_dir = current_dir / "output"

# Create the output directory and its parent directory if they do not exist
output_dir.mkdir(parents=True, exist_ok=True)
```

Total sales by Month

In [45]:
```
# Create a connection to the database
conn = sqlite3.connect(database_path)
```

In [47]:
```
# Execute the query and load results into a Pandas DataFrame
query = '''
SELECT sale_date, SUM(total_price) as total_sales
FROM sales
GROUP BY sale_date
ORDER BY sale_date ASC
'''
df = pd.read_sql_query(query, conn)
```

In [48]:
```
#print the Dataframe
print(df)
```

```
      sale_date  total_sales
0    2022-01-01          680
1    2022-01-02         1595
2    2022-01-03          865
3    2022-01-04         1570
4    2022-01-05           50
..          ...          ...
342  2022-12-27          200
343  2022-12-28          175
344  2022-12-29         1045
345  2022-12-30          655
346  2022-12-31          730

[347 rows x 2 columns]
```

In [49]:
```
#check the data types
df.info()
```

```
<class 'pandas.core.frame.DataFrame'>
RangeIndex: 347 entries, 0 to 346
Data columns (total 2 columns):
 #   Column       Non-Null Count  Dtype 
---  ------       --------------  ----- 
 0   sale_date    347 non-null    object
 1   total_sales  347 non-null    int64 
dtypes: int64(1), object(1)
memory usage: 5.5+ KB
```

In [50]:
```
#convert sale_date to datetime
df['sale_date'] = pd.to_datetime(df['sale_date'])
df.info()
```

```
<class 'pandas.core.frame.DataFrame'>
RangeIndex: 347 entries, 0 to 346
Data columns (total 2 columns):
 #   Column       Non-Null Count  Dtype         
---  ------       --------------  -----         
 0   sale_date    347 non-null    datetime64[ns]
 1   total_sales  347 non-null    int64         
dtypes: datetime64[ns](1), int64(1)
memory usage: 5.5 KB
```

In [51]:
```
# set the sale_date column as the index
df = df.set_index('sale_date')
df.head(3)
```

Out[51]:

sale_date	total_sales
2022-01-01	680
2022-01-02	1595
2022-01-03	865

In [52]:
```
#Resample the date to a monthly frequency and compute the sum
df_monthly = df.resample('M').sum()
df_monthly
```

Out[52]:

sale_date	total_sales
2022-01-31	24040
2022-02-28	21470
2022-03-31	20250
2022-04-30	26340
2022-05-31	21205
2022-06-30	22275
2022-07-31	29050
2022-08-31	25000
2022-09-30	23310
2022-10-31	27875
2022-11-30	19870
2022-12-31	24260

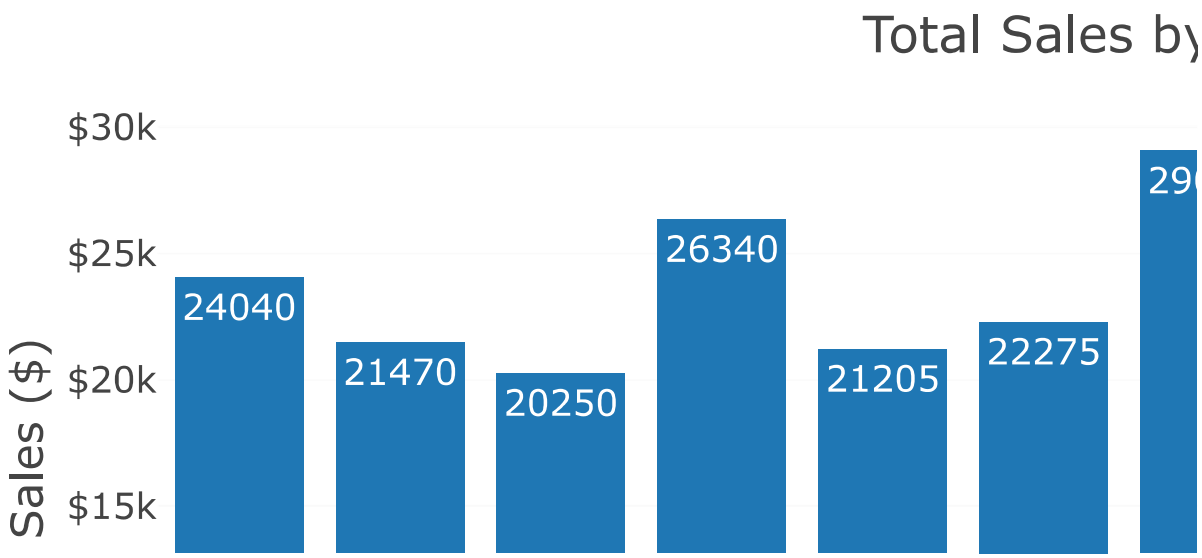
In [53]:
```
#MAP the month number to short  month name
df_monthly['month_name']=df_monthly.index.strftime('%b')
df_monthly
```

Out[53]:

sale_date	total_sales	month_name
2022-01-31	24040	Jan
2022-02-28	21470	Feb
2022-03-31	20250	Mar
2022-04-30	26340	Apr
2022-05-31	21205	May
2022-06-30	22275	Jun
2022-07-31	29050	Jul
2022-08-31	25000	Aug
2022-09-30	23310	Sep
2022-10-31	27875	Oct
2022-11-30	19870	Nov
2022-12-31	24260	Dec

In [65]:
```
# Create the Plotly figure with text
fig = px.bar(df_monthly,
             x='month_name',
             y='total_sales',
```

```
            template=plotly_template,
            text='total_sales')
#Set the Layout
fig.update_layout (
    title = 'Total Sales by Month',
    xaxis_title = 'Month',
    yaxis_title =  'Total Sales ($)',
    yaxis_tickprefix = '$',
)
#Show the plot
fig.show ()
# Save the chart as a PNG image
fig.write_image(output_dir / 'monthly_sales.png',
                width=1200,
                height=400,
                scale=4)
```



Total Sales by Product

In [71]:
```
# Execute the query and Load results into a pandas Dataframe
query = '''
SELECT p.product_name, SUM(s.total_price) as total_sales
FROM sales s
JOIN products p ON s.product_id = p.product_id
```

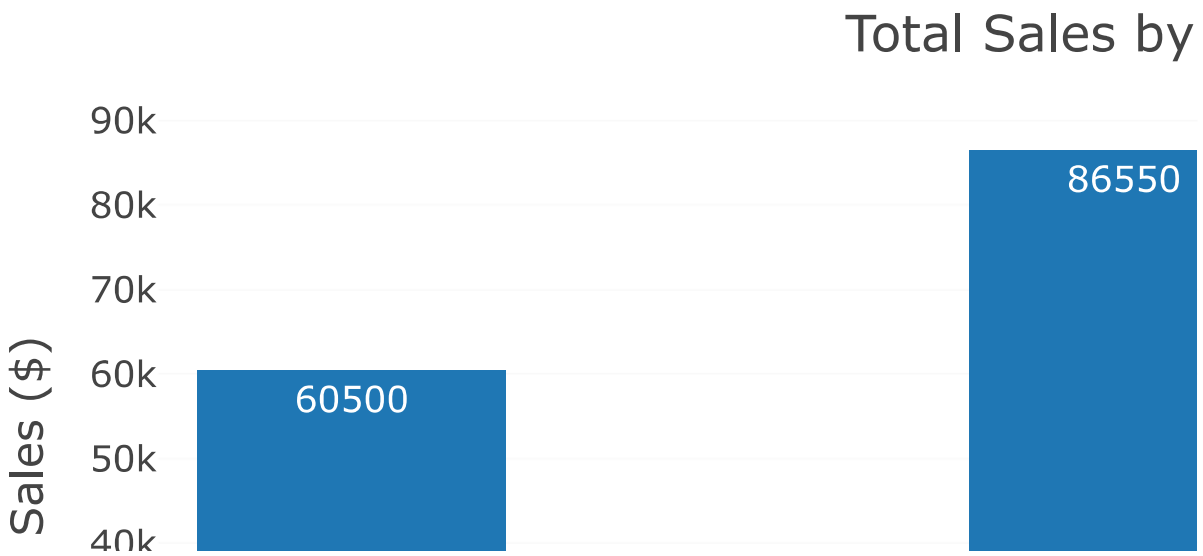
```
GROUP BY p.product_name
'''
df = pd.read_sql_query(query, conn)
```

In [72]:
```
df
```

Out[72]:

	product_name	total_sales
0	Product A	60500
1	Product B	26475
2	Product C	86550
3	Product D	46320
4	Product E	65100

In [73]:
```
#create the plotly figure with text parameter
fig = px.bar(df,
             x='product_name',
             y='total_sales',
             template=plotly_template,
             text='total_sales')
#Set the Layout
fig.update_layout(
    title='Total Sales by Product',
    xaxis_title='product',
    yaxis_title='Total Sales ($)',
)
#show the plot
fig.show()
#Save the chart as a PNG image
fig.write_image(output_dir /'product_sales.png',
                width=1200,
               height=400,
               scale=4)
```



Top Customer by Sales

In [76]:
```
# Execute the query and load results into a Pandas DataFrame
query = '''
SELECT c.first_name || ' ' || c.last_name as customer_name, SUM(S.total_price) as tota
FROM sales
JOIN customers c ON s.customer_id = c.customer_id
GROUP BY customer_name
ORDER BY total_sales DESC
LIMIT 10
'''
df = pd.read_sql_query(query, conn)
```

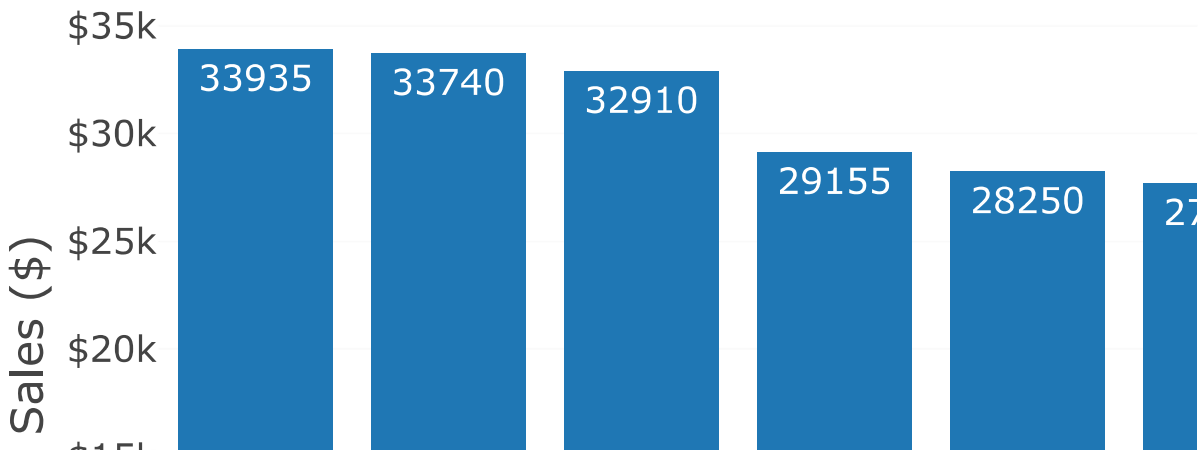
In [77]:
```
df
```

Out[77]:

	customer_name	total_sales
0	Alice Jones	33935
1	Frank Wilson	33740
2	Isabel Garcia	32910
3	Emily Davis	29155
4	Grace Lee	28250
5	John Doe	27700
6	Henry Chen	27350
7	Jane Doe	25235
8	David Brown	24445
9	Bob Smith	22225

In [83]:

```
#create the plotly figure
fig = px.bar(df,
             x='customer_name',
             y='total_sales',
             template=plotly_template,
             text='total_sales')
#set the layout
fig.update_layout(
    title='Top Customer by Sales',
    xaxis_title='Customer',
    yaxis_title='Total Sales ($)',
    yaxis_tickprefix='$',
)
#Show the plot
fig.show()
#save the chart as a PNG image
fig.write_image(output_dir / 'customer_sales.png',
                width=1200,
                height=400,
                scale=4)
```



Create PDF Report

In [88]:
```
#Define the font color as RGB value(dark gray)
font_color = (64,64,64)
#Find all PNG files in the output folder
chart_filenames = [str(chart_path) for chart_path in output_dir.glob("*.png")]

# Create a PDF document and set the page size
pdf=FPDF()
pdf.add_page()
pdf.set_font('Arial','B' , 24)
#Add the overall page title
title=f"Sales Report as of{date.today().strftime('%m%d%y')}"
pdf.set_text_color(*font_color)
pdf.cell(0,20,title,align='C',ln=1)

#Add each chart to the PDF document
for chart_filename in chart_filenames:
    pdf.ln(10) # Add padding at the top of the next chart
    pdf.image(chart_filename, x=None, y=None, w=pdf.w-20,h=0)
#save the document to a file on disk
pdf.output(output_dir / "sales_report.pdf", "F")
```

```
C:\Users\ceitf\AppData\Local\Temp\ipykernel_8780\454167056.py:13: DeprecationWarning:

The parameter "ln" is deprecated. Instead of ln=1 use new_x=XPos.LMARGIN, new_y=YPos.
NEXT.

C:\Users\ceitf\AppData\Local\Temp\ipykernel_8780\454167056.py:20: DeprecationWarning:

"dest" parameter is deprecated, unused and will soon be removed
```

Customer Segment using SQL

In [91]:
```
#Execute the query and load results into a pandas DataFrame
query = '''
SELECT
  customers.customer_id,
  customers.first_name || ' ' || customers.last_name as customer_name,
  SUM(sales.total_price) as total_sales,
  CASE
    WHEN SUM(sales.total_price) > 30000 THEN 'High Value'
    WHEN SUM(sales.total_price) > 26000 THEN 'Medium Value'
    ELSE 'Low Value'
  END as customer_segment
FROM sales
INNER JOIN customers ON sales.customer_id = customers.customer_id
GROUP BY customers.customer_id
ORDER BY total_sales DESC
'''
df=pd.read_sql_query(query, conn)
df
```

Out[91]:

	customer_id	customer_name	total_sales	customer_segment
0	4	Alice Jones	33935	High Value
1	7	Frank Wilson	33740	High Value
2	10	Isabel Garcia	32910	High Value
3	6	Emily Davis	29155	Medium Value
4	8	Grace Lee	28250	Medium Value
5	1	John Doe	27700	Medium Value
6	9	Henry Chen	27350	Medium Value
7	2	Jane Doe	25235	Low Value
8	5	David Brown	24445	Low Value
9	3	Bob Smith	22225	Low Value

Customer Segment using SQL + Pandas

In [92]:
```
#  Execute the query and Load results into a pandas DataFrame
query ='''
SELECT
```

```
  customers.customer_id,
  customers.first_name|| customers.last_name as customer_name,
  SUM(sales.total_price) as total_sales
FROM sales
INNER JOIN customers ON sales.customer_id = customers.customer_id
GROUP BY customers.customer_id
'''
df=pd.read_sql_query(query, conn)
df
```

Out[92]:

	customer_id	customer_name	total_sales
0	1	JohnDoe	27700
1	2	JaneDoe	25235
2	3	BobSmith	22225
3	4	AliceJones	33935
4	5	DavidBrown	24445
5	6	EmilyDavis	29155
6	7	FrankWilson	33740
7	8	GraceLee	28250
8	9	HenryChen	27350
9	10	IsabelGarcia	32910

In [93]:

```
#Group the data by customer segment
bins=[0,26000,30000,float('inf')]
labels=['Low Value','Medium Value','High Value']
df['customer_segment']=pd.cut(df['total_sales'], bins=bins, labels=labels)

#Order the data by total sales
df=df.sort_values(by='total_sales', ascending= False)
df
```

Out[93]:

	customer_id	customer_name	total_sales	customer_segment
3	4	AliceJones	33935	High Value
6	7	FrankWilson	33740	High Value
9	10	IsabelGarcia	32910	High Value
5	6	EmilyDavis	29155	Medium Value
7	8	GraceLee	28250	Medium Value
0	1	JohnDoe	27700	Medium Value
8	9	HenryChen	27350	Medium Value
1	2	JaneDoe	25235	Low Value
4	5	DavidBrown	24445	Low Value
2	3	BobSmith	22225	Low Value

In [95]:
```
#close the connection
conn.close()
```