01.all.5.projects.rev.02

September 19, 2024

[1]:
```python
import pandas as pd
from sqlalchemy import create_engine, text

def execute_query(conn, query):
    """Executes a SQL query using the provided connection and returns the␣
 ↪results as a list of tuples."""
    result = conn.execute(text(query))  # Execute the query directly

    # Get column names
    columns = result.keys()  # Get column names from the result object
    print("Columns returned by the query:", columns)

    # Fetch all rows
    rows = result.fetchall()

    # Print the shape of the results for debugging
    print("Shape of results:", len(rows), "rows,", len(rows[0]) if rows else 0,␣
 ↪"columns")

    # Print the first few rows of the results
    print("First few rows of results:", rows[:5])  # Adjust the number of rows␣
 ↪as needed

    return rows  # Return the fetched rows

# Database connection details
server = '10.10.11.241'
database = 'omar.rme1'
user = 'omar'
password = 'omar123'

# Create SQLAlchemy engine
connection_string = f'mssql+pyodbc://{user}:{password}@{server}/{database}?
 ↪driver=SQL+Server'
engine = create_engine(connection_string)

try:
    # Check if the connection is successful
    with engine.connect() as conn:  # Use a context manager to handle the␣
 ↪connection
        print("Connected to SQL Server successfully!")

except Exception as e:
    print("Error connecting to SQL Server:", e)
```

```
Connected to SQL Server successfully!
```

[2]:
```
import pandas as pd

# SQL query to sum amounts for each project
query = """
SELECT project_no, SUM(amount) AS TotalAmount
FROM [omar.rme1].[dbo].[cost_dist]
WHERE project_no IN ('144', '173', '172', '184', '198')
GROUP BY project_no;
"""

try:
    # Execute the query using SQLAlchemy's execute method
    with engine.connect() as conn:
        result = conn.execute(text(query))  # Use text() to wrap the query

        # Fetch all rows and column names
        rows = result.fetchall()
        columns = result.keys()

        # Create a DataFrame from the results
        df = pd.DataFrame(rows, columns=columns)

    # Display the DataFrame
    print(df.to_string(index=False))  # Display without the index for cleaner␣
 ↪output

except Exception as e:
    print("Error executing query:", e)
```

```
project_no  TotalAmount
       144 8.569741e+08
       172 5.966704e+08
       173 4.277230e+07
       184 3.008438e+08
       198 1.223076e+08
```

[3]:
```
import plotly.express as px
import plotly.io as pio
```

```python
import sqlalchemy

# 1. SQL query to fetch project_no and project_name
query_names = """
SELECT DISTINCT project_no, project_name
FROM [omar.rme1].[dbo].[cost_dist]
WHERE project_no IN ('144', '173', '172', '184', '198');
"""

# 2. Execute the query using pd.read_sql and the connection string
df_names = pd.read_sql(query_names, connection_string)

# 3. Merge DataFrames using 'project_no' as the key
df_merged = pd.merge(df, df_names, on='project_no', how='left')

# 4. Fill in missing values with 0
df_merged.fillna(0, inplace=True)

# 5. Combine project_no and project_name for labels
df_merged['Project'] = df_merged['project_no'] + ' - ' +␣
 ↪df_merged['project_name']

# 6. Calculate total cost
total_cost = df_merged['TotalAmount'].sum()

# 7. Create bar chart with adjusted width, project numbers in labels, and␣
 ↪annotations
fig = px.bar(
    df_merged,
    x='Project',
    y='TotalAmount',
    title=f'Total Amount by Project (Total: {int(total_cost):,})',  # Format␣
 ↪total_cost with commas, no decimals
    labels={'project_name': 'Project', 'TotalAmount': 'Total Amount'}
)

# Add annotations (text labels) to each bar with comma formatting and no␣
 ↪decimals
for i, row in df_merged.iterrows():
    fig.add_annotation(
        x=row['Project'],
        y=row['TotalAmount'],
        text=f"{int(row['TotalAmount']):,}",  # Convert to integer (remove␣
 ↪decimals) and format with commas
        showarrow=False,
        yshift=10,
    )
```

```
# Customize layout
fig.update_layout(width=800)

# 8. Display chart
fig.write_html('total_amount_by_project_bar_chart.html')
fig.show()
```

[4]:
```
import plotly.express as px
import plotly.subplots as sp

# 1. SQL query to get top 10 suppliers for each project
query_top_suppliers = """
SELECT project_no, supplier_name, SUM(amount) AS TotalAmount
FROM [omar.rme1].[dbo].[cost_dist]
WHERE project_no IN ('144', '173', '172', '184', '198')
GROUP BY project_no, supplier_name
ORDER BY project_no, TotalAmount DESC
"""

# 2. Execute the query
with engine.connect() as conn:
    results_top_suppliers = execute_query(conn, query_top_suppliers)

# 3. Create DataFrame for top suppliers
df_top_suppliers = pd.DataFrame(results_top_suppliers, columns=['project_no',␣
↪'supplier_name', 'TotalAmount'])

# 4. Get the project names from df_merged (assuming it's available from the 3rd␣
↪cell)
project_names_dict = df_merged.set_index('project_no')['project_name'].to_dict()

# 5. Create a subplot for each project, arranged vertically, with project␣
↪number and name in titles
fig = sp.make_subplots(
    rows=5,
    cols=1,
    subplot_titles=[f"{proj} - {project_names_dict[proj]}" for proj in␣
↪df['project_no'].unique()]
)

# 6. Iterate through each project and create a bar chart for its top 10␣
↪suppliers
for i, project_no in enumerate(df['project_no'].unique()):
    # Filter data for the current project and get top 10 suppliers
    project_data = df_top_suppliers[df_top_suppliers['project_no'] ==␣
↪project_no].nlargest(10, 'TotalAmount')
```

```
    # Create the bar chart and add it to the subplot
    fig.add_trace(
        px.bar(
            project_data,
            x='supplier_name',
            y='TotalAmount',
            labels={'supplier_name': 'Supplier', 'TotalAmount': 'Total Amount'}
        ).data[0],
        row=i+1, col=1
    )

    # Update layout for the subplot
    fig.update_xaxes(title_text='Supplier Name', row=i+1, col=1)
    fig.update_yaxes(title_text='Total Amount', row=i+1, col=1)

# 7. Adjust the overall layout
fig.update_layout(
    height=1500,
    width=800,
    showlegend=False,
    title_text="Top 10 Suppliers by Project"
)

# 8. Display chart
fig.write_html('top_10_suppliers_by_project.html')
fig.show()
```

```
Columns returned by the query: RMKeyView(['project_no', 'supplier_name',
'TotalAmount'])
Shape of results: 347 rows, 3 columns
First few rows of results: [('144', 'Staff Loan', 274181577.1100001), ('144',
None, 272591965.4700006), ('144', 'Miscellaneous supplier', 54742415.940000094),
('144', '                        -                     ',
42419560.059999965), ('144', '              ', 23168751.60000001)]
```

[5]:
```
import plotly.express as px
import plotly.subplots as sp

# 1. SQL query to get top 10 expenditure types for each project
query_top_expenditures = """
SELECT project_no, expenditure_type, SUM(amount) AS TotalAmount
FROM [omar.rme1].[dbo].[cost_dist]
WHERE project_no IN ('144', '173', '172', '184', '198')
GROUP BY project_no, expenditure_type
ORDER BY project_no, TotalAmount DESC
"""
```

```
# 2. Execute the query
with engine.connect() as conn:
    results_top_expenditures = execute_query(conn, query_top_expenditures)

# 3. Create DataFrame for top expenditure types
df_top_expenditures = pd.DataFrame(results_top_expenditures,␣
 ↪columns=['project_no', 'expenditure_type', 'TotalAmount'])

# 4. Get the project names from df_merged (assuming it's available from the 3rd␣
 ↪cell)
# (This line remains the same as in the fourth cell)

# 5. Create a subplot for each project, arranged vertically, with project␣
 ↪number and name in titles
# Specify specs to create subplots of type 'domain' for pie charts
fig = sp.make_subplots(
    rows=5,
    cols=1,
    subplot_titles=[f"{proj} - {project_names_dict[proj]}" for proj in␣
 ↪df['project_no'].unique()],
    specs=[[{'type': 'domain'}] for _ in range(5)]  # 5 subplots, each of type␣
 ↪'domain'
)

# 6. Iterate through each project and create a pie chart for its top 10␣
 ↪expenditure types
for i, project_no in enumerate(df['project_no'].unique()):
    # Filter data for the current project and get top 10 expenditure types
    project_data = df_top_expenditures[df_top_expenditures['project_no'] ==␣
 ↪project_no].nlargest(10, 'TotalAmount')

    # Create the pie chart
    pie_chart = px.pie(
        project_data,
        values='TotalAmount',
        names='expenditure_type',
        title=f'Top 10 Expenditure Types for Project {project_no}',
    )

    # Add the trace from the pie chart to the subplot
    fig.add_trace(pie_chart.data[0], row=i+1, col=1)

# 7. Adjust the overall layout
fig.update_layout(
    height=2000,  # Adjust height as needed
    width=800,
    showlegend=True,
    title_text="Top 10 Expenditure Types by Project"
)

# Update layout for traces
fig.update_traces(textposition='inside', textinfo='percent+label')

# 8. Display chart
fig.write_html('top_10_expenditure_types_by_project.html')
fig.show()
```

```
Columns returned by the query: RMKeyView(['project_no', 'expenditure_type',
'TotalAmount'])
Shape of results: 265 rows, 3 columns
First few rows of results: [('144', 'Tools', 240794810.08), ('144',
'Subcontractor', 164534537.69000003), ('144', 'Hirings daily wages',
155780705.2500001), ('144', 'Site Staff', 65975102.07), ('144', 'Hirings
Payrool', 55954280.1)]
```