01.all.5.projects.rev.03

September 23, 2024

[15]:

```
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 856974103.92
       172 596670367.19
       173  42772299.35
       184 300843751.02
       198 122307643.32
```

[3]:

```
import plotly.express as px
import plotly.io as pio
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', 274181577.1100003), ('144',
None, 272591965.47000027), ('144', 'Miscellaneous supplier', 54742415.94000004),
('144', '                    -                    ',
42419560.059999995), ('144', '             ', 23168751.6)]
```

[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', 164534537.6899997), ('144', 'Hirings daily wages',
155780705.25000003), ('144', 'Site Staff', 65975102.07), ('144', 'Hirings
Payrool', 55954280.09999998)]
```

[6]:

```
# Seach for Steel Rft        in "comment" column



import plotly.express as px

# 1. SQL query to get total steel reinforcement cost for each project
query_steel_cost = """
SELECT project_no, SUM(amount) AS SteelReinforcementCost
FROM [omar.rme1].[dbo].[cost_dist]
WHERE project_no IN ('144', '173', '172', '184', '198')
AND comment LIKE N'%      %'
GROUP BY project_no;
"""

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

# 3. Create DataFrame for steel reinforcement costs
df_steel_cost = pd.DataFrame(results_steel_cost, columns=['project_no',␣
 ↪'SteelReinforcementCost'])

# 4. Merge with project names DataFrame
df_steel_merged = pd.merge(df_steel_cost, df_names, on='project_no', how='left')

# 5. Fill in missing values with 0 (in case some projects have no steel␣
 ↪reinforcement)
```

```
df_steel_merged.fillna(0, inplace=True)

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

# 7. Calculate total steel reinforcement cost
total_steel_cost = df_steel_merged['SteelReinforcementCost'].sum()

# 8. Create bar chart
fig = px.bar(
    df_steel_merged,
    x='Project',
    y='SteelReinforcementCost',
    title=f'Total Steel Reinforcement Cost by Project (Total:␣
 ↪{int(total_steel_cost):,})',
    labels={'project_name': 'Project', 'SteelReinforcementCost': 'Steel␣
 ↪Reinforcement Cost'}
)

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

# Customize layout
fig.update_layout(width=800)

# 9. Display or save the chart
fig.show()
fig.write_html('steel_reinforcement_cost_by_project.html')
```

```
Columns returned by the query: RMKeyView(['project_no',
'SteelReinforcementCost'])
Shape of results: 2 rows, 2 columns
First few rows of results: [('144', 8561778.26), ('172', 10539564.78)]
```

[7]:

```
# Seach for Steel Rft        in "line_desc" column


import plotly.express as px
```

```python
# 1. SQL query to get total steel reinforcement cost for each project (from line_desc)
query_steel_cost_line_desc = """
SELECT project_no, SUM(amount) AS SteelReinforcementCost
FROM [omar.rme1].[dbo].[cost_dist]
WHERE project_no IN ('144', '173', '172', '184', '198')
AND line_desc LIKE N'%      %'
GROUP BY project_no;
"""

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

# 3. Create DataFrame for steel reinforcement costs (from line_desc)
df_steel_cost_line_desc = pd.DataFrame(results_steel_cost_line_desc, columns=['project_no', 'SteelReinforcementCost'])

# 4. Merge with project names DataFrame
df_steel_merged_line_desc = pd.merge(df_steel_cost_line_desc, df_names, on='project_no', how='left')

# 5. Fill in missing values with 0 (in case some projects have no steel reinforcement)
df_steel_merged_line_desc.fillna(0, inplace=True)

# 6. Combine project_no and project_name for labels
df_steel_merged_line_desc['Project'] = df_steel_merged_line_desc['project_no'] + ' - ' + df_steel_merged_line_desc['project_name']

# 7. Calculate total steel reinforcement cost (from line_desc)
total_steel_cost_line_desc = df_steel_merged_line_desc['SteelReinforcementCost'].sum()

# 8. Create bar chart
fig = px.bar(
    df_steel_merged_line_desc,
    x='Project',
    y='SteelReinforcementCost',
    title=f'Total Steel Reinforcement Cost by Project (from line_desc) (Total: {int(total_steel_cost_line_desc):,})',
    labels={'project_name': 'Project', 'SteelReinforcementCost': 'Steel Reinforcement Cost'}
)
```

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

# Customize layout
fig.update_layout(width=800)

# 9. Display or save the chart
fig.show()
fig.write_html('steel_reinforcement_cost_by_project_line_desc.html')
```

```
Columns returned by the query: RMKeyView(['project_no',
'SteelReinforcementCost'])
Shape of results: 2 rows, 2 columns
First few rows of results: [('144', 8561778.26), ('172', 10539564.78)]
```

[8]:
```
import pandas as pd

# 1. SQL query to fetch all columns for the specified projects
query_export = """
SELECT *
FROM [omar.rme1].[dbo].[cost_dist]
WHERE project_no IN ('144', '173', '172', '184', '198');
"""

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

# 3. Create a DataFrame for all the data
df_export = pd.DataFrame(results_export)

# 4. Get the column names from the query output (if needed)
# If you don't know the column names beforehand, uncomment the following line
# columns = result.keys()
# df_export.columns = columns

# 5. Iterate through each project and export its data to a separate Excel file
for project_no in df['project_no'].unique():
    # Filter data for the current project
    project_data = df_export[df_export['project_no'] == project_no]

    # Export to Excel
    project_data.to_excel(f'project_{project_no}_data.xlsx', index=False)
    print(f"Exported data for project {project_no} to project_{project_no}_data.
↪xlsx")
```

```
Columns returned by the query: RMKeyView(['trs_id', 'transaction_source',
'project_no', 'project_name', 'project_zone', 'task_no', 'task_name',
'top_task_no', 'top_task_name', 'po_no', 'gl_date', 'expenditure_type',
'project_location', 'project_floor', 'project_area', 'expenditure_category',
'expend_org', 'amount', 'line_no', 'line_desc', 'inv_no', 'unit', 'qty',
'ipc_no', 'supplier_no', 'supplier_name', 'supplier_site', 'comment',
'inventory_item', 'owner', 'distributions_status', 'distributions_date',
'distributions_details'])
Shape of results: 49319 rows, 33 columns
First few rows of results: [(2378785.0, 'Inventory Misc', '144', 'EGAT
Pelletizing Plant', None, '301', 'Formwork', 3.0, 'CONCRETE WORK', None,
datetime.datetime(2021, 10, 24, 0, 0), 'Materials', None, None, None,
'Materials', 'EGAT Pelletizing Plant-0144', 217.89, 1.0, 'X-Roc Strong Repair',
None, 'Kilogram', 75.0, 44556654.0, None, None, None, 'X-Roc Strong Repair',
'360010000161\xa0 ', 'Purchasing\xa0 ', 'Received', datetime.datetime(2021, 10,
25, 0, 0), 'Accounted transaction received '), (2619047.0, 'Inventory Misc',
'144', 'EGAT Pelletizing Plant', None, '301', 'Formwork', 3.0, 'CONCRETE WORK',
None, datetime.datetime(2021, 12, 18, 0, 0), 'Materials', None, None, None,
'Materials', 'EGAT Pelletizing Plant-0144', 2158.8, 1.0, '          42',
None, 'Ton', 2.0, 46450272.0, None, None, None, '          42',
'140010000015\xa0 ', 'Purchasing\xa0 ', 'Received', datetime.datetime(2021, 12,
19, 0, 0), 'Accounted transaction received '), (2618962.0, 'Inventory Misc',
'144', 'EGAT Pelletizing Plant', None, '301', 'Formwork', 3.0, 'CONCRETE WORK',
None, datetime.datetime(2021, 12, 15, 0, 0), 'Materials', None, None, None,
'Materials', 'EGAT Pelletizing Plant-0144', 230.38, 1.0, '        ', None,
'Each', 15.0, 46343006.0, None, None, None, '        ', '200010000625\xa0
', 'Purchasing\xa0 ', 'Received', datetime.datetime(2021, 12, 19, 0, 0),
'Accounted transaction received '), (2618976.0, 'Inventory Misc', '144', 'EGAT
Pelletizing Plant', None, '301', 'Formwork', 3.0, 'CONCRETE WORK', None,
datetime.datetime(2021, 12, 17, 0, 0), 'Tools', None, None, None, 'Materials',
'EGAT Pelletizing Plant-0144', 102.74, 1.0, '        5', None, 'Bag', 1.0,
46414140.0, None, None, None, '        5', '140010000009\xa0 ',
'Purchasing\xa0 ', 'Received', datetime.datetime(2021, 12, 19, 0, 0), 'Accounted
transaction received '), (2374100.0, 'Inventory Misc', '144', 'EGAT Pelletizing
Plant', None, '301', 'Formwork', 3.0, 'CONCRETE WORK', None,
datetime.datetime(2021, 10, 5, 0, 0), 'Materials', None, None, None,
'Materials', 'EGAT Pelletizing Plant-0144', 944.18, 1.0, 'X-Roc Strong Repair',
None, 'Kilogram', 325.0, 44008715.0, None, None, None, 'X-Roc Strong Repair',
'360010000161\xa0 ', 'Purchasing\xa0 ', 'Received', datetime.datetime(2021, 10,
24, 0, 0), 'Accounted transaction received ')]
Exported data for project 144 to project_144_data.xlsx
Exported data for project 172 to project_172_data.xlsx
Exported data for project 173 to project_173_data.xlsx
Exported data for project 184 to project_184_data.xlsx
Exported data for project 198 to project_198_data.xlsx
```

[9]:

```
import plotly.express as px

# 1. SQL query to get total concrete cost for each project (from line_desc)
query_concrete_cost = """
SELECT project_no, SUM(amount) AS ConcreteCost
FROM [omar.rme1].[dbo].[cost_dist]
WHERE project_no IN ('144', '173', '172', '184', '198')
AND line_desc LIKE N'%   %'
GROUP BY project_no;
"""

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

# 3. Create DataFrame for concrete costs
df_concrete_cost = pd.DataFrame(results_concrete_cost, columns=['project_no',␣
 ↪'ConcreteCost'])

# 4. Merge with project names DataFrame
df_concrete_merged = pd.merge(df_concrete_cost, df_names, on='project_no',␣
 ↪how='left')

# 5. Fill in missing values with 0 (in case some projects have no concrete cost)
df_concrete_merged.fillna(0, inplace=True)

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

# 7. Calculate total concrete cost
total_concrete_cost = df_concrete_merged['ConcreteCost'].sum()

# 8. Create bar chart
fig = px.bar(
    df_concrete_merged,
    x='Project',
    y='ConcreteCost',
    title=f'Total Concrete Cost by Project (Total: {int(total_concrete_cost):
 ↪,})',
    labels={'project_name': 'Project', 'ConcreteCost': 'Concrete Cost'}
)
```

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

# Customize layout
fig.update_layout(width=800)

# 9. Display or save the chart
fig.show()
fig.write_html('concrete_cost_by_project.html')
```

```
Columns returned by the query: RMKeyView(['project_no', 'ConcreteCost'])
Shape of results: 4 rows, 2 columns
First few rows of results: [('144', 70354628.11000001), ('172', 112126924.74),
('173', 313974.79000000004), ('184', 13846523.059999999)]
```

[13]:

```
import plotly.express as px
import plotly.subplots as sp

# 1. Columns to analyze
columns_to_analyze = ['transaction_source', 'task_name', 'top_task_name',␣
 ↪'expenditure_category', 'expend_org', 'line_desc', 'owner']

# 2. Create subplots (7 rows for columns, 5 columns for projects)
fig = sp.make_subplots(rows=7, cols=5, subplot_titles=[f"{proj} -␣
 ↪{project_names_dict[proj]}" for proj in df['project_no'].unique()] *␣
 ↪len(columns_to_analyze))

# 3. Iterate through columns and projects to create bar charts
for row_idx, column_name in enumerate(columns_to_analyze):
    for col_idx, project_no in enumerate(df['project_no'].unique()):
        # SQL query to get top 10 amounts for the current column and project
        query_top_amounts = f"""
        SELECT {column_name}, SUM(amount) AS TotalAmount
        FROM [omar.rme1].[dbo].[cost_dist]
        WHERE project_no = '{project_no}'
        GROUP BY {column_name}
        ORDER BY TotalAmount DESC
        """
        # Execute the query
        with engine.connect() as conn:
            results_top_amounts = execute_query(conn, query_top_amounts)

        # Create DataFrame for top amounts
        df_top_amounts = pd.DataFrame(results_top_amounts,␣
↪columns=[column_name, 'TotalAmount'])

        # Filter to top 10
        df_top_amounts = df_top_amounts.nlargest(10, 'TotalAmount')

        # Create the bar chart and add it to the subplot
        fig.add_trace(
            px.bar(
                df_top_amounts,
                x=column_name,
                y='TotalAmount',
                labels={column_name: column_name.replace('_', ' ').title(),␣
↪'TotalAmount': 'Total Amount'}
            ).data[0],
            row=row_idx + 1, col=col_idx + 1
        )

        # Update layout for the subplot
        fig.update_xaxes(title_text=column_name.replace('_', ' ').title(),␣
↪row=row_idx + 1, col=col_idx + 1)
        fig.update_yaxes(title_text='Total Amount', row=row_idx + 1,␣
↪col=col_idx + 1)

# 4. Adjust the overall layout
fig.update_layout(
    height=3500,  # Adjust height as needed to accommodate all subplots
    width=1500,   # Adjust width as needed
    showlegend=False,
    title_text="Top 10 Amounts by Column and Project"
)

# 5. Display chart
fig.show()
fig.write_html('data_analysis_for_each.html')
```

```
Columns returned by the query: RMKeyView(['transaction_source', 'TotalAmount'])
Shape of results: 5 rows, 2 columns
First few rows of results: [('Oracle Payables Supplier Invoices',
450398793.94000006), ('Inventory Misc', 272591965.47), ('WORK_CONFRIMATION',
125372258.15), ('Oracle Purchasing Receipt Accruals', 7563875.08), ('Non-
Recoverable Tax from Purchasing Receipts', 1047211.2800000001)]
Columns returned by the query: RMKeyView(['transaction_source', 'TotalAmount'])
Shape of results: 5 rows, 2 columns
First few rows of results: [('Inventory Misc', 366241470.0300001), ('Oracle
Payables Supplier Invoices', 154747599.75999996), ('WORK_CONFRIMATION',
65141732.62000001), ('Oracle Purchasing Receipt Accruals', 9245232.29), ('Non-
Recoverable Tax from Purchasing Receipts', 1294332.4900000002)]
Columns returned by the query: RMKeyView(['transaction_source', 'TotalAmount'])
Shape of results: 3 rows, 2 columns
First few rows of results: [('Inventory Misc', 19096025.169999998), ('Oracle
Payables Supplier Invoices', 13119068.539999997), ('WORK_CONFRIMATION',
10557205.639999999)]
Columns returned by the query: RMKeyView(['transaction_source', 'TotalAmount'])
Shape of results: 3 rows, 2 columns
First few rows of results: [('Inventory Misc', 226576665.81999978),
('WORK_CONFRIMATION', 46073348.15000001), ('Oracle Payables Supplier Invoices',
28193737.04999999)]
Columns returned by the query: RMKeyView(['transaction_source', 'TotalAmount'])
Shape of results: 3 rows, 2 columns
First few rows of results: [('WORK_CONFRIMATION', 113258938.41000001), ('Oracle
Payables Supplier Invoices', 5024336.680000002), ('Inventory Misc', 4024368.23)]
Columns returned by the query: RMKeyView(['task_name', 'TotalAmount'])
Shape of results: 45 rows, 2 columns
First few rows of results: [('Mobilization', 318660895.18), ('Formwork',
198072878.73999998), ('Private Cars', 65487786.56), ('Cast Place Concrete',
53453125.530000016), ('Safety tools', 43382002.95999999)]
Columns returned by the query: RMKeyView(['task_name', 'TotalAmount'])
Shape of results: 55 rows, 2 columns
First few rows of results: [('Mat', 358789109.9599999), ('Hirings payrool',
63833646.42), ('Mobilization', 37416917.61), ('Steel Reinforcement',
25919802.41), ('Excavation', 16369636.39)]
Columns returned by the query: RMKeyView(['task_name', 'TotalAmount'])
Shape of results: 26 rows, 2 columns
First few rows of results: [('Other Equipment', 19251317.34999999),
('Mobilization', 8822791.420000002), ('Backfilling', 7049029.13), ('Excavation',
2529079.3100000005), ('Rented Cars', 1638413.0399999998)]
Columns returned by the query: RMKeyView(['task_name', 'TotalAmount'])
Shape of results: 42 rows, 2 columns
First few rows of results: [('Mat', 210375739.57999998), ('Electrical Works
(G)', 21302660.199999996), ('Safety tools', 15766525.64), ('Plumbing Works(G)',
13541344.06), ('Hirings payrool', 11024149.19)]
Columns returned by the query: RMKeyView(['task_name', 'TotalAmount'])
Shape of results: 11 rows, 2 columns
First few rows of results: [('Plumbing Works(G)', 116485018.40999995), ('Mat',
2916273.82), ('Mobilization', 1672696.98), ('HVAC AIR DUCT', 674763.28),
('Safety tools', 433331.13)]
Columns returned by the query: RMKeyView(['top_task_name', 'TotalAmount'])
Shape of results: 9 rows, 2 columns
```

```
First few rows of results: [('INDIRECT COST', 530604170.45000005), ('CONCRETE
WORK', 262974690.8600001), ('SITE WORK', 28576392.31), ('THERMAL & MOISTURE',
11684887.56), ('DOORS & WINDOWS WORK', 9256835.33)]
Columns returned by the query: RMKeyView(['top_task_name', 'TotalAmount'])
Shape of results: 11 rows, 2 columns
First few rows of results: [('SPECIALITIES', 367012430.2700003), ('Indirect
Cost', 150107197.61999997), ('CONCRETE WORK', 43060015.510000005), ('SITE WORK',
19512853.580000002), ('METAL WORK', 5290741.4399999995)]
Columns returned by the query: RMKeyView(['top_task_name', 'TotalAmount'])
Shape of results: 5 rows, 2 columns
First few rows of results: [('Indirect Cost', 32348323.2), ('SITE WORK',
10450411.32), ('EQUIPMENT', 6207.3), ('MECHANICAL WORKS', -4964.7),
('SPECIALITIES', -27677.76999999999)]
Columns returned by the query: RMKeyView(['top_task_name', 'TotalAmount'])
Shape of results: 11 rows, 2 columns
First few rows of results: [('SPECIALITIES', 214116737.22999987), ('Indirect
Cost', 40013467.69000001), ('ELECTRICAL WORKS', 21302660.2), ('MECHANICAL
WORKS', 13537688.040000001), ('SITE WORK', 4023920.3)]
Columns returned by the query: RMKeyView(['top_task_name', 'TotalAmount'])
Shape of results: 5 rows, 2 columns
First few rows of results: [('MECHANICAL WORKS', 116485018.41000001),
('SPECIALITIES', 2916273.8200000003), ('Indirect Cost', 2231587.81), ('HVAC',
674763.28), ('CONCRETE WORK', 0.0)]
Columns returned by the query: RMKeyView(['expenditure_category',
'TotalAmount'])
Shape of results: 15 rows, 2 columns
First few rows of results: [('Materials', 296449118.41000015), ('Indirect',
200494698.39000005), ('Subcontracts', 165409544.68), ('Direct Manpower',
160174791.18), ('Depreciation', 18393895.879999995)]
Columns returned by the query: RMKeyView(['expenditure_category',
'TotalAmount'])
Shape of results: 15 rows, 2 columns
First few rows of results: [('Materials', 414909854.1499999), ('Subcontracts',
65430784.38), ('Indirect', 53893516.21000001), ('Direct Manpower',
35355458.68000001), ('Contingencies', 10394322.049999999)]
Columns returned by the query: RMKeyView(['expenditure_category',
'TotalAmount'])
Shape of results: 11 rows, 2 columns
First few rows of results: [('Materials', 26324632.450000003), ('Subcontracts',
11778237.280000003), ('Indirect', 2601493.77), ('Direct Manpower', 1671417.26),
('Depreciation', 327222.19999999995)]
Columns returned by the query: RMKeyView(['expenditure_category',
'TotalAmount'])
Shape of results: 12 rows, 2 columns
First few rows of results: [('Materials', 224896635.71999994), ('Subcontracts',
50220153.2), ('Indirect', 19247570.43), ('Depreciation', 2122804.27), ('Finance
cost', 2041566.62)]
Columns returned by the query: RMKeyView(['expenditure_category',
'TotalAmount'])
Shape of results: 7 rows, 2 columns
First few rows of results: [('Subcontracts', 120008396.41000001), ('Materials',
4024292.49), ('Indirect', 675288.0), ('Depreciation', 673755.65), ('Finance
cost', 440088.76999999996)]
Columns returned by the query: RMKeyView(['expend_org', 'TotalAmount'])
Shape of results: 2 rows, 2 columns
First few rows of results: [('Alrowad Construction_OU', 575257182.44), ('EGAT
Pelletizing Plant-0144', 281716921.4800001)]
Columns returned by the query: RMKeyView(['expend_org', 'TotalAmount'])
Shape of results: 2 rows, 2 columns
First few rows of results: [('Rolling Mill#4-0172', 379133629.5400001),
('Alrowad Construction_OU', 217536737.65000004)]
Columns returned by the query: RMKeyView(['expend_org', 'TotalAmount'])
Shape of results: 1 rows, 2 columns
First few rows of results: [('Alrowad Construction_OU', 42772299.35)]
Columns returned by the query: RMKeyView(['expend_org', 'TotalAmount'])
Shape of results: 2 rows, 2 columns
First few rows of results: [('Suez Steel Intake\xa0 & Pump Stations-0184',
226576665.82000005), ('Alrowad Construction_OU', 74267085.19999999)]
Columns returned by the query: RMKeyView(['expend_org', 'TotalAmount'])
Shape of results: 2 rows, 2 columns
First few rows of results: [('Alrowad Construction_OU', 118283275.09000006),
('EGAT Mechanical Installations', 4024368.230000001)]
Columns returned by the query: RMKeyView(['line_desc', 'TotalAmount'])
Shape of results: 5792 rows, 2 columns
First few rows of results: [('Concrete 300 kg/cm2 SRC 350 Kg/m3', 66489169.71),
('Ready Mix Concrete With Strength 470 kg/cm2 & Cement Content 500 kg/m3 (OPC)
-Slag-MF', 43661060.07), ('   ', 19697433.700000037), (None,
15919147.800000036), ('     Concrete 250 kg/cm2 OPC 325 Kg/m3',
15836382.180000002)]
Columns returned by the query: RMKeyView(['line_desc', 'TotalAmount'])
Shape of results: 3937 rows, 2 columns
First few rows of results: [('          400   / 2
410    /3 ', 35768655.97000001), ('Ready Mix Concrete With Strength 470 kg/cm2
& Cement Content 500 kg/m3 (OPC) -Slag-MF', 29024753.169999998), ('Master Flow
950', 27531000.000000004), ('Ready Mix Concrete 400 kg/cm2 SRC 425 kg/m3',
27018005.7), ('          400   / 2             410    /3 ',
20990798.92)]
Columns returned by the query: RMKeyView(['line_desc', 'TotalAmount'])
Shape of results: 495 rows, 2 columns
First few rows of results: [('Geogrid (Stratagrid - SGU 120) (1.9m * 100m)',
6945597.970000001), ('                                    ', 3874405.2500000005), ('Straragrid - SGU
80,(1.9M*100M ROLL SIZE)', 2555743.209999999), ('

                      ', 2137428.4999999995), ('ADJ LOCK&LOAD to
Projects May23', 1881638.34)]
Columns returned by the query: RMKeyView(['line_desc', 'TotalAmount'])
```

```
Shape of results: 2065 rows, 2 columns
First few rows of results: [('    HDPE       710      10   ',
65059183.76), ('    HDPE       710      16   ', 32830204.5), ('
HDPE       560      10   ', 13657698.87), ('Standart Vertical / End
Suction/ Single Stage pump model: PCV-M 300-400AB-3K, 350 l/s @ 37 m head
complete with Electric Motor (Turkey/ IE3) 250 Kw, 1500 rpm, IP 56, Insulation
Class F, Temperature Rise Class B, Bearing Life 100.000 hrs', 7997531.92),
('2.2.002 73088/4', 7923373.92)]
Columns returned by the query: RMKeyView(['line_desc', 'TotalAmount'])
Shape of results: 443 rows, 2 columns
First few rows of results: [('1.1 80735/6', 16767756.0), ('1.1 80735/5',
15777570.0), ('2.1 87029/7', 13000004.0), ('2.1 80735/7', 12625272.0), ('2.1
87029/9', 8000000.18)]
Columns returned by the query: RMKeyView(['owner', 'TotalAmount'])
Shape of results: 17 rows, 2 columns
First few rows of results: [('Finacial\xa0 ', 552040508.8599999),
('Procurement\xa0 ', 222140595.16000003), ('Purchasing\xa0 ',
63212201.389999986), ('MEP\xa0 ', 14580522.48), ('Automation\xa0 ',
2254988.1700000004)]
Columns returned by the query: RMKeyView(['owner', 'TotalAmount'])
Shape of results: 14 rows, 2 columns
First few rows of results: [('Procurement\xa0 ', 241384135.73), ('Finacial\xa0
', 194378496.13), ('Purchasing\xa0 ', 129655366.3), ('MEP\xa0 ', 30133527.47),
('Safety\xa0 ', 3321832.2799999993)]
Columns returned by the query: RMKeyView(['owner', 'TotalAmount'])
Shape of results: 10 rows, 2 columns
First few rows of results: [('Finacial\xa0 ', 23750540.000000004),
('Purchasing\xa0 ', 18092770.31), ('Procurement\xa0 ', 468467.64), ('MEP\xa0 ',
425466.67000000004), ('Safety\xa0 ', 22779.47)]
Columns returned by the query: RMKeyView(['owner', 'TotalAmount'])
Shape of results: 11 rows, 2 columns
First few rows of results: [('MEP\xa0 ', 188975046.99000004), ('Finacial\xa0 ',
72698933.64), ('Procurement\xa0 ', 20959951.089999996), ('Purchasing\xa0 ',
15988453.590000002), ('Mechanical\xa0 ', 1573151.56)]
Columns returned by the query: RMKeyView(['owner', 'TotalAmount'])
Shape of results: 10 rows, 2 columns
First few rows of results: [('Finacial\xa0 ', 115057223.22999999),
('Mechanical\xa0 ', 3226080.0), ('Scaffolding\xa0 ', 2100881.8),
('Purchasing\xa0 ', 1628975.95), ('MEP\xa0 ', 120398.51)]
```

[17]:

```python
import plotly.express as px
import plotly.subplots as sp

# 1. Columns to analyze
columns_to_analyze = ['transaction_source', 'task_name', 'top_task_name',␣
 ↪'expenditure_category', 'expend_org', 'line_desc', 'owner']
```

```python
# 2. Iterate through columns to create separate figures
for column_name in columns_to_analyze:
    # Create subplots (1 row, 5 columns for projects)
    fig = sp.make_subplots(rows=1, cols=5, subplot_titles=[f"{proj} - {project_names_dict[proj]}" for proj in df['project_no'].unique()])

    # Iterate through projects to create bar charts
    for col_idx, project_no in enumerate(df['project_no'].unique()):
        # SQL query to get top 10 amounts for the current column and project
        query_top_amounts = f"""
        SELECT {column_name}, SUM(amount) AS TotalAmount
        FROM [omar.rme1].[dbo].[cost_dist]
        WHERE project_no = '{project_no}'
        GROUP BY {column_name}
        ORDER BY TotalAmount DESC
        """

        # Execute the query
        with engine.connect() as conn:
            results_top_amounts = execute_query(conn, query_top_amounts)

        # Create DataFrame for top amounts
        df_top_amounts = pd.DataFrame(results_top_amounts, columns=[column_name, 'TotalAmount'])

        # Filter to top 10
        df_top_amounts = df_top_amounts.nlargest(10, 'TotalAmount')

        # Create the bar chart and add it to the subplot
        fig.add_trace(
            px.bar(
                df_top_amounts,
                x=column_name,
                y='TotalAmount',
                title=f'Project {project_no} - {project_names_dict[project_no]}',  # Include project number and name in the title
                labels={column_name: column_name.replace('_', ' ').title(), 'TotalAmount': 'Total Amount'}
            ).data[0],
            row=1, col=col_idx + 1
        )

        # Update layout for the subplot
        fig.update_xaxes(title_text=column_name.replace('_', ' ').title(), row=1, col=col_idx + 1)
        fig.update_yaxes(title_text='Total Amount', row=1, col=col_idx + 1)
    # Adjust the overall layout
    fig.update_layout(
        height=600,
        width=1500,
        showlegend=False,
        title_text=f"Top 10 {column_name.replace('_', ' ').title()} Amounts by Project"
    )

    # Save the chart as an HTML file
    fig.write_html(f'top_10_{column_name}_by_project.html')

    print(f"Generated chart for {column_name}")
```

```
Columns returned by the query: RMKeyView(['transaction_source', 'TotalAmount'])
Shape of results: 5 rows, 2 columns
First few rows of results: [('Oracle Payables Supplier Invoices',
450398793.9400001), ('Inventory Misc', 272591965.4700001), ('WORK_CONFRIMATION',
125372258.15), ('Oracle Purchasing Receipt Accruals', 7563875.08), ('Non-
Recoverable Tax from Purchasing Receipts', 1047211.2800000001)]
Columns returned by the query: RMKeyView(['transaction_source', 'TotalAmount'])
Shape of results: 5 rows, 2 columns
First few rows of results: [('Inventory Misc', 366241470.03000015), ('Oracle
Payables Supplier Invoices', 154747599.76), ('WORK_CONFRIMATION',
65141732.61999999), ('Oracle Purchasing Receipt Accruals', 9245232.29), ('Non-
Recoverable Tax from Purchasing Receipts', 1294332.4900000002)]
Columns returned by the query: RMKeyView(['transaction_source', 'TotalAmount'])
Shape of results: 3 rows, 2 columns
First few rows of results: [('Inventory Misc', 19096025.17), ('Oracle Payables
Supplier Invoices', 13119068.540000003), ('WORK_CONFRIMATION', 10557205.64)]
Columns returned by the query: RMKeyView(['transaction_source', 'TotalAmount'])
Shape of results: 3 rows, 2 columns
First few rows of results: [('Inventory Misc', 226576665.81999996),
('WORK_CONFRIMATION', 46073348.15), ('Oracle Payables Supplier Invoices',
28193737.050000004)]
Columns returned by the query: RMKeyView(['transaction_source', 'TotalAmount'])
Shape of results: 3 rows, 2 columns
First few rows of results: [('WORK_CONFRIMATION', 113258938.40999997), ('Oracle
Payables Supplier Invoices', 5024336.680000002), ('Inventory Misc',
4024368.2300000004)]
Generated chart for transaction_source
Columns returned by the query: RMKeyView(['task_name', 'TotalAmount'])
Shape of results: 45 rows, 2 columns
First few rows of results: [('Mobilization', 318660895.17999995), ('Formwork',
198072878.74000004), ('Private Cars', 65487786.559999995), ('Cast Place
Concrete', 53453125.53000001), ('Safety tools', 43382002.96)]
Columns returned by the query: RMKeyView(['task_name', 'TotalAmount'])
```

```
Shape of results: 55 rows, 2 columns
First few rows of results: [('Mat', 358789109.9600001), ('Hirings payrool',
63833646.42), ('Mobilization', 37416917.60999999), ('Steel Reinforcement',
25919802.410000004), ('Excavation', 16369636.39)]
Columns returned by the query: RMKeyView(['task_name', 'TotalAmount'])
Shape of results: 26 rows, 2 columns
First few rows of results: [('Other Equipment', 19251317.350000028),
('Mobilization', 8822791.420000004), ('Backfilling', 7049029.129999999),
('Excavation', 2529079.3099999996), ('Rented Cars', 1638413.0400000003)]
Columns returned by the query: RMKeyView(['task_name', 'TotalAmount'])
Shape of results: 42 rows, 2 columns
First few rows of results: [('Mat', 210375739.57999992), ('Electrical Works
(G)', 21302660.2), ('Safety tools', 15766525.64), ('Plumbing Works(G)',
13541344.06), ('Hirings payrool', 11024149.190000001)]
Columns returned by the query: RMKeyView(['task_name', 'TotalAmount'])
Shape of results: 11 rows, 2 columns
First few rows of results: [('Plumbing Works(G)', 116485018.41000001), ('Mat',
2916273.819999999), ('Mobilization', 1672696.9800000004), ('HVAC AIR DUCT',
674763.2799999999), ('Safety tools', 433331.13)]
Generated chart for task_name
Columns returned by the query: RMKeyView(['top_task_name', 'TotalAmount'])
Shape of results: 9 rows, 2 columns
First few rows of results: [('INDIRECT COST', 530604170.45000005), ('CONCRETE
WORK', 262974690.86000004), ('SITE WORK', 28576392.310000002), ('THERMAL &
MOISTURE', 11684887.559999999), ('DOORS & WINDOWS WORK', 9256835.33)]
Columns returned by the query: RMKeyView(['top_task_name', 'TotalAmount'])
Shape of results: 11 rows, 2 columns
First few rows of results: [('SPECIALITIES', 367012430.2699999), ('Indirect
Cost', 150107197.62), ('CONCRETE WORK', 43060015.51), ('SITE WORK',
19512853.580000002), ('METAL WORK', 5290741.44)]
Columns returned by the query: RMKeyView(['top_task_name', 'TotalAmount'])
Shape of results: 5 rows, 2 columns
First few rows of results: [('Indirect Cost', 32348323.200000003), ('SITE WORK',
10450411.32), ('EQUIPMENT', 6207.3), ('MECHANICAL WORKS', -4964.700000000001),
('SPECIALITIES', -27677.769999999993)]
Columns returned by the query: RMKeyView(['top_task_name', 'TotalAmount'])
Shape of results: 11 rows, 2 columns
First few rows of results: [('SPECIALITIES', 214116737.22999993), ('Indirect
Cost', 40013467.69000001), ('ELECTRICAL WORKS', 21302660.2), ('MECHANICAL
WORKS', 13537688.040000001), ('SITE WORK', 4023920.3)]
Columns returned by the query: RMKeyView(['top_task_name', 'TotalAmount'])
Shape of results: 5 rows, 2 columns
First few rows of results: [('MECHANICAL WORKS', 116485018.41), ('SPECIALITIES',
2916273.8200000008), ('Indirect Cost', 2231587.81), ('HVAC', 674763.2799999999),
('CONCRETE WORK', 0.0)]
Generated chart for top_task_name
Columns returned by the query: RMKeyView(['expenditure_category',
'TotalAmount'])
```

```
Shape of results: 15 rows, 2 columns
First few rows of results: [('Materials', 296449118.4100001), ('Indirect',
200494698.39000005), ('Subcontracts', 165409544.68), ('Direct Manpower',
160174791.17999998), ('Depreciation', 18393895.88)]
Columns returned by the query: RMKeyView(['expenditure_category',
'TotalAmount'])
Shape of results: 15 rows, 2 columns
First few rows of results: [('Materials', 414909854.15), ('Subcontracts',
65430784.38000001), ('Indirect', 53893516.20999999), ('Direct Manpower',
35355458.68), ('Contingencies', 10394322.049999997)]
Columns returned by the query: RMKeyView(['expenditure_category',
'TotalAmount'])
Shape of results: 11 rows, 2 columns
First few rows of results: [('Materials', 26324632.449999996), ('Subcontracts',
11778237.280000001), ('Indirect', 2601493.77), ('Direct Manpower',
1671417.2599999998), ('Depreciation', 327222.1999999999)]
Columns returned by the query: RMKeyView(['expenditure_category',
'TotalAmount'])
Shape of results: 12 rows, 2 columns
First few rows of results: [('Materials', 224896635.71999994), ('Subcontracts',
50220153.2), ('Indirect', 19247570.430000003), ('Depreciation', 2122804.27),
('Finance cost', 2041566.6199999999)]
Columns returned by the query: RMKeyView(['expenditure_category',
'TotalAmount'])
Shape of results: 7 rows, 2 columns
First few rows of results: [('Subcontracts', 120008396.41), ('Materials',
4024292.49), ('Indirect', 675288.0000000001), ('Depreciation',
673755.6499999999), ('Finance cost', 440088.77)]
Generated chart for expenditure_category
Columns returned by the query: RMKeyView(['expend_org', 'TotalAmount'])
Shape of results: 2 rows, 2 columns
First few rows of results: [('Alrowad Construction_OU', 575257182.4399999),
('EGAT Pelletizing Plant-0144', 281716921.48000014)]
Columns returned by the query: RMKeyView(['expend_org', 'TotalAmount'])
Shape of results: 2 rows, 2 columns
First few rows of results: [('Rolling Mill#4-0172', 379133629.5400001),
('Alrowad Construction_OU', 217536737.65)]
Columns returned by the query: RMKeyView(['expend_org', 'TotalAmount'])
Shape of results: 1 rows, 2 columns
First few rows of results: [('Alrowad Construction_OU', 42772299.349999994)]
Columns returned by the query: RMKeyView(['expend_org', 'TotalAmount'])
Shape of results: 2 rows, 2 columns
First few rows of results: [('Suez Steel Intake\xa0 & Pump Stations-0184',
226576665.82000005), ('Alrowad Construction_OU', 74267085.19999999)]
Columns returned by the query: RMKeyView(['expend_org', 'TotalAmount'])
Shape of results: 2 rows, 2 columns
First few rows of results: [('Alrowad Construction_OU', 118283275.09000003),
('EGAT Mechanical Installations', 4024368.2300000004)]
```

```
Generated chart for expend_org
Columns returned by the query: RMKeyView(['line_desc', 'TotalAmount'])
Shape of results: 5792 rows, 2 columns
First few rows of results: [('Concrete 300 kg/cm2 SRC 350 Kg/m3',
66489169.710000016), ('Ready Mix Concrete With Strength 470 kg/cm2 & Cement
Content 500 kg/m3 (OPC) -Slag-MF', 43661060.07), ('   ', 19697433.7), (None,
15919147.800000047), ('    Concrete 250 kg/cm2 OPC 325 Kg/m3',
15836382.179999998)]
Columns returned by the query: RMKeyView(['line_desc', 'TotalAmount'])
Shape of results: 3937 rows, 2 columns
First few rows of results: [('          400   / 2
410    /3 ', 35768655.970000006), ('Ready Mix Concrete With Strength 470 kg/cm2
& Cement Content 500 kg/m3 (OPC) -Slag-MF', 29024753.17), ('Master Flow 950',
27531000.000000004), ('Ready Mix Concrete 400 kg/cm2 SRC 425 kg/m3',
27018005.700000003), ('          400   / 2            410
/3 ', 20990798.91999999)]
Columns returned by the query: RMKeyView(['line_desc', 'TotalAmount'])
Shape of results: 495 rows, 2 columns
First few rows of results: [('Geogrid (Stratagrid - SGU 120) (1.9m * 100m)',
6945597.969999998), ('          ', 3874405.25), ('Straragrid - SGU
80,(1.9M*100M ROLL SIZE)', 2555743.2099999995), ('

                      ', 2137428.5), ('ADJ LOCK&LOAD to Projects
May23', 1881638.34)]
Columns returned by the query: RMKeyView(['line_desc', 'TotalAmount'])
Shape of results: 2065 rows, 2 columns
First few rows of results: [('    HDPE       710      10   ',
65059183.76), ('    HDPE       710      16   ', 32830204.5), ('
HDPE       560      10   ', 13657698.870000001), ('Standart Vertical /
End Suction/ Single Stage pump model: PCV-M 300-400AB-3K, 350 l/s @ 37 m head
complete with Electric Motor (Turkey/ IE3) 250 Kw, 1500 rpm, IP 56, Insulation
Class F, Temperature Rise Class B, Bearing Life 100.000 hrs', 7997531.92),
('2.2.002 73088/4', 7923373.92)]
Columns returned by the query: RMKeyView(['line_desc', 'TotalAmount'])
Shape of results: 443 rows, 2 columns
First few rows of results: [('1.1 80735/6', 16767756.0), ('1.1 80735/5',
15777570.0), ('2.1 87029/7', 13000004.0), ('2.1 80735/7', 12625272.0), ('2.1
87029/9', 8000000.18)]
Generated chart for line_desc
Columns returned by the query: RMKeyView(['owner', 'TotalAmount'])
Shape of results: 17 rows, 2 columns
First few rows of results: [('Finacial\xa0 ', 552040508.86), ('Procurement\xa0
', 222140595.15999997), ('Purchasing\xa0 ', 63212201.39), ('MEP\xa0 ',
14580522.48), ('Automation\xa0 ', 2254988.1700000004)]
Columns returned by the query: RMKeyView(['owner', 'TotalAmount'])
Shape of results: 14 rows, 2 columns
First few rows of results: [('Procurement\xa0 ', 241384135.72999996),
('Finacial\xa0 ', 194378496.12999997), ('Purchasing\xa0 ', 129655366.29999994),
('MEP\xa0 ', 30133527.470000006), ('Safety\xa0 ', 3321832.28)]
Columns returned by the query: RMKeyView(['owner', 'TotalAmount'])
Shape of results: 10 rows, 2 columns
First few rows of results: [('Finacial\xa0 ', 23750540.000000007),
('Purchasing\xa0 ', 18092770.31), ('Procurement\xa0 ', 468467.64), ('MEP\xa0 ',
425466.67000000004), ('Safety\xa0 ', 22779.469999999998)]
Columns returned by the query: RMKeyView(['owner', 'TotalAmount'])
Shape of results: 11 rows, 2 columns
First few rows of results: [('MEP\xa0 ', 188975046.98999995), ('Finacial\xa0 ',
72698933.64), ('Procurement\xa0 ', 20959951.09), ('Purchasing\xa0 ',
15988453.590000002), ('Mechanical\xa0 ', 1573151.560000001)]
Columns returned by the query: RMKeyView(['owner', 'TotalAmount'])
Shape of results: 10 rows, 2 columns
First few rows of results: [('Finacial\xa0 ', 115057223.22999999),
('Mechanical\xa0 ', 3226080.0), ('Scaffolding\xa0 ', 2100881.8000000003),
('Purchasing\xa0 ', 1628975.9499999997), ('MEP\xa0 ', 120398.51000000001)]
Generated chart for owner
```