A Comparative Approach to Addressing Business Use Cases:

Simple Pivoting Techniques with Excel, Power BI, Python, and SQL

We can use various tools to address business use cases, depending on proficiency, available tools, and efficiency in achieving results. While business cases are often complex, we'll simplify the process with a straightforward use case to quickly build understanding and dive into the concepts. This study highlights the benefits of pivoting techniques by comparing solutions in Excel, Power BI, Python, and SQL.

Dan Vukota

linkedin.com/in/danvukota/

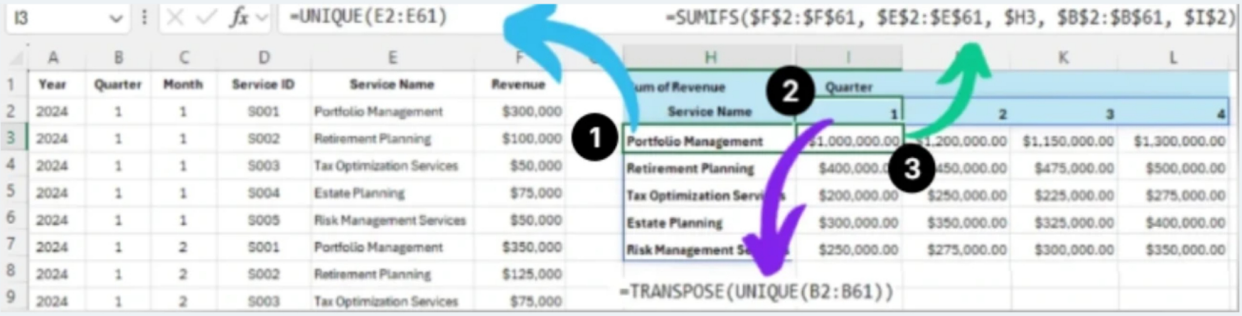
Now, let's say we want to explore and derive insights from a dataset. We can achieve this by analyzing the data from different dimensions (descriptive variables) through aggregation, sorting, slicing, and filtering. This process is often referred to as pivoting, which essentially involves restructuring data to enable better summarization and analysis. Pivoting can include rotating rows into columns (wide format) or columns into rows (long format). For this use case, we'll focus on wide-format pivoting techniques, which are particularly effective for reporting for a small set of dimensions and also are more intuitive to read.

Let's start by examining a subset of data from a table compiled by a financial services firm, detailing monthly revenue for 2024 categorized by financial service type.

Year	Quarter	Month	Service ID	Service Name	Revenue
2024	1	1	S001	Portfolio Management	$300,000
2024	1	1	S002	Retirement Planning	$100,000
2024	1	1	S003	Tax Optimization Services	$50,000
2024	1	1	S004	Estate Planning	$75,000
2024	1	1	S005	Risk Management Services	$50,000
2024	1	2	S001	Portfolio Management	$350,000
2024	1	2	S002	Retirement Planning	$125,000
2024	1	2	S003	Tax Optimization Services	$75,000
2024	1	2	S004	Estate Planning	$100,000
2024	1	2	S005	Risk Management Services	$75,000
2024	1	3	S001	Portfolio Management	$350,000
2024	1	3	S002	Retirement Planning	$175,000
2024	1	3	S003	Tax Optimization Services	$75,000
2024	1	3	S004	Estate Planning	$125,000
2024	1	3	S005	Risk Management Services	$125,000
2024	2	4	S001	Portfolio Management	$325,000
2024	2	4	S002	Retirement Planning	$125,000
⋮			⋮		⋮
2024	4	12	S002	Retirement Planning	$200,000
2024	4	12	S003	Tax Optimization Services	$100,000
2024	4	12	S004	Estate Planning	$150,000
2024	4	12	S005	Risk Management Services	$150,000

To evaluate financial performance in 2024, we can start by analyzing total revenue by service on a quarterly basis.

If we were to use **Excel**, we can do the calculation in many ways. One way would be to use **UNIQUE**, **TRANSPOSE**, and **SUMIFS** formulas. Assuming we have this table in cells **A1** to **F61**, we can create a summary table in columns **H** to **L**. Let's break this down into these steps:



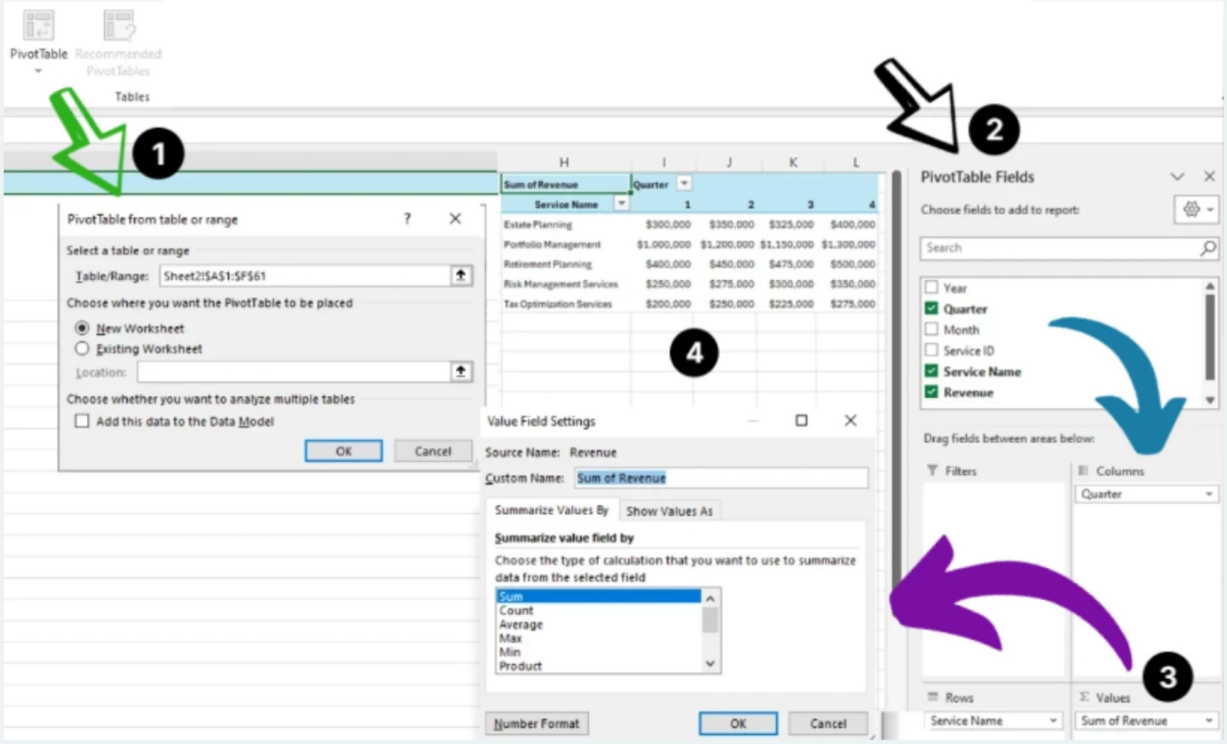
1. In cell **H3**, we can use the formula **=UNIQUE(E2:E61)** to extract and span the unique service names from column **E**.

2. In cell **I2**, we can use the formula **=TRANSPOSE(UNIQUE(B2:B61))** to extract and span the unique service names from column **B** and rotate the data row-wise to cells **I2** to **L2**.

3. In cell **I3**, we can apply the formula **=SUMIFS($F$2:$F$61, $E$2:$E$61, $H3, $B$2:$B$61, $I$2)** to sum the Revenue in column **F** where the **Service Name** in **H3** matches those in column **E** and the **Quarter** in **I2** matches those in column **B**. We then repeat this type of formula in the rest of the cells by changing the match criteria for column **H** cell reference and/or cell references of columns **I** to **L** for Quarter.

A more effective method that is a feature of Excel is to use a **Pivot Table**. For larger datasets, Pivot Tables can be more efficient for summarization, as formulas may slow down performance with extensive calculations. They offer a more visual and interactive way to analyze data, making it easier to transform rows into columns and aggregate information across multiple dimensions. In contrast, using formulas for such tasks can be more time-consuming and complex.

To create a Pivot Table for our use case, let's break it down into the following steps:



1. Start by clicking on a cell, we'll select **H1**. Then select **PivotTable** in the Insert menu at the top and enter the data range.

2. For the **PivotTable Fields**, we designate the **Service Name** column for **Rows**, **Quarter** column as **Columns**, and the **Revenue** column as **Values**.

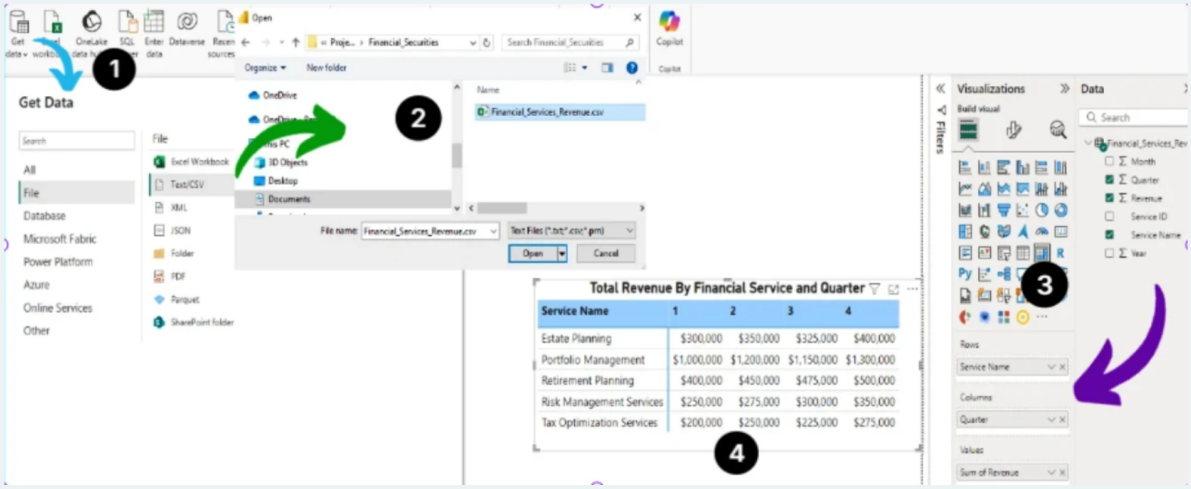
3. In the **Value Field Settings** for **Revenue**, make sure to summarize the values using the **Sum** function.

4. The results are displayed in **H1** and can be further formatted and filtered if desired.

Now let's look at how we can do this in **Power BI**.

Power BI allows us to do this easily and quickly. A **Matrix** visual, has the same type of functionality and interactivity as a Pivot Table, and is excellent for transforming rows into columns and aggregating information across multiple dimensions.

For this use case, we can perform the following steps:



1. Click on **Get Data** in the top menu.

2. Import the excel data as a csv file.

3. Choose the **Matrix** visual from the Visualizations menu on the right. Select **Service Name** column values as **Rows**, **Quarter** column values as **Columns**, and **Revenue** column values as **Values** aggregated as a sum.

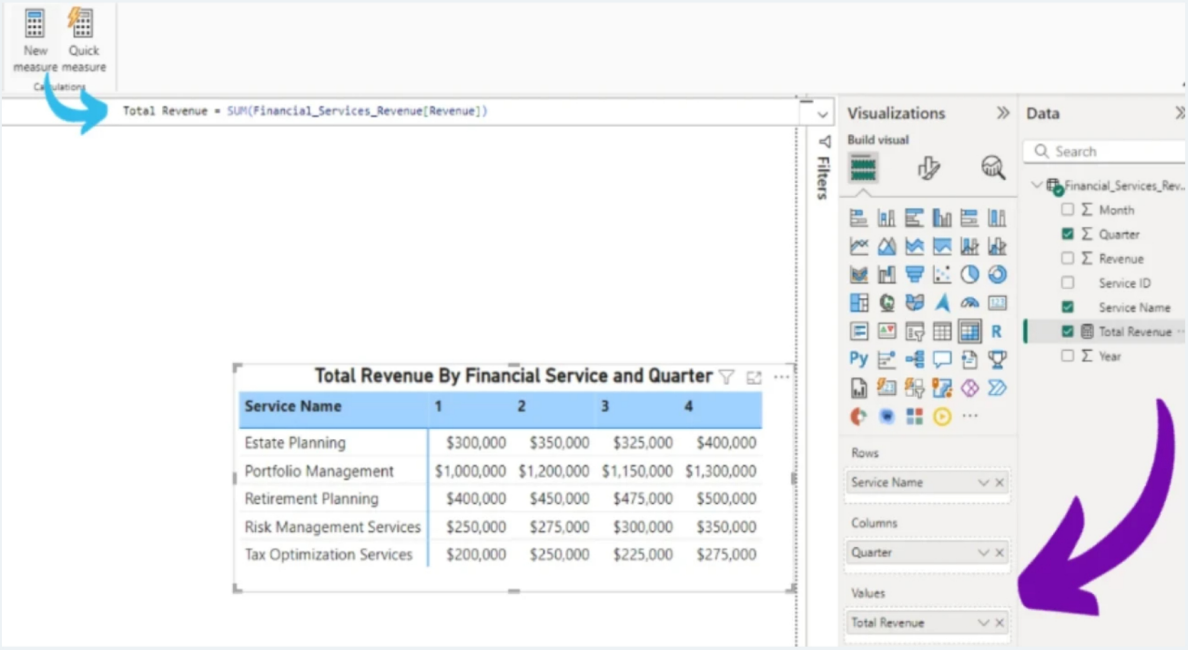
4. The results are displayed in the report view that can be further formatted and filtered if desired.

We can also aggregate the **Revenue** column by creating a simple **measure** using **DAX (Data Analysis Expressions)**. Measures are powerful calculations or aggregations created to analyze data. DAX measures, built using formulas, offer greater flexibility for performing complex calculations while being optimized for performance.

We'll create a simple measure called **Total Revenue** using the following DAX formula:

Total Revenue = SUM(Financial_Services_Revenue[Revenue])

We'll replace the **Values** with **Total Revenue** to achieve the same results:



Business software may not always accommodate every use case, requiring extensions through programming languages and tools. However, in some situations, a few lines of code can add new capabilities to software tools, transform data or offer a quicker, more efficient ad-hoc solution, such as in this case using **Python**.

Using Python in **Jupyter Notebooks**, we can import the csv file into a **pandas Dataframe** called **df_service**, perform a **groupby** aggregation and **sum** the **Revenue** column by **Service Name** and **Quarter**. We'll further reshape the DataFrame by pivoting the Quarter values into columns using the **unstack()** method, and format the total revenue numbers in each cell as currency with no decimal places.

The full code is shown below:

jupyter Total_Revenue Last Checkpoint: 12/01/2025 (autosaved)

Logout

File Edit View Insert Cell Kernel Widgets Help Trusted ib3.9 Kernel

Run Code

Import pandas library and declare the csv file path variable

In [1]:
```
import pandas as pd

inputfile = r'C:\[redacted]Financial_Securities\Financial_Services_Revenue.csv'
```

Load csv file in the *df_service* pandas dataframe

In [2]:
```
df_services = pd.read_csv(inputfile)
```

Explore the 1st 5 rows

In [3]:
```
df_services.head()
```

Out[3]:

	Year	Quarter	Month	Service ID	Service Name	Revenue
0	2024	1	1	S001	Portfolio Management	300000
1	2024	1	1	S002	Retirement Planning	100000
2	2024	1	1	S003	Tax Optimization Services	50000
3	2024	1	1	S004	Estate Planning	75000
4	2024	1	1	S005	Risk Management Services	50000

Create the *df_tot_revenue* summary Dataframe by grouping the *df_services* Dataframe by *'Service Name'* and *'Quarter'* columns, aggregate the *'Revenue'* column using the sum() function for each group, reshape the DataFrame by pivoting the *'Quarter'* level of the index into columns using unstack(), and format the numbers in each cell with two decimal places

In [4]:
```
df_tot_revenue = df_services.groupby(['Service Name', 'Quarter'])['Revenue'].sum().unstack('Quarter').style.format('${:,.0f}')
```

The new Dataframe **df_tot_revenue** encompasses the aggregation, pivoting and formatting in one line of code:

```
df_tot_revenue = df_services.groupby(['Service Name', 'Quarter'])['Revenue'].sum().unstack('Quarter').style.format('${:,.0f}')
```

We can also use a Pivot Table with a pandas DataFrame, which is specifically designed for this purpose and provides a more intuitive way to aggregate data and achieve the same result. The **pivot_table()** function is similar to how the Excel Pivot Tables and the Matrix visuals in Power BI function. We'll create a DataFrame called **df_tot_revenue_pivot** and as parameters, we can use the **df_services** DataFrame we already defined. Other parameters used are **index** as **Service Name** to define the **Rows**, the **Quarter** column to define the **Columns**, the values in the **Revenue** column to define the **Values**, the **sum()** aggregate function to summarize the data, and we format the numbers as currency with no decimal places.

Here is the code to define the Pivot Table:

```
df_tot_revenue_pivot = pd.pivot_table(
   df_services,
   values='Revenue',
   index='Service Name',
   columns='Quarter',
   aggfunc='sum'
).style.format('${:,.0f}')
```

The code shown below incorporates this Pivot Table and displays the desired results:

Define the df_tot_revenue_pivot DataFrame as a pivot table derived from the df_services DataFrame

In [5]:
```
df_tot_revenue_pivot = pd.pivot_table(
    df_services,
    values='Revenue',
    index='Service Name',
    columns='Quarter',
    aggfunc='sum'
).style.format('${:,.0f}')
```

Display the Total Revenue

In [6]:
```
df_tot_revenue_pivot
```

Out[6]:

Quarter	1	2	3	4
Service Name				
Estate Planning	$300,000	$350,000	$325,000	$400,000
Portfolio Management	$1,000,000	$1,200,000	$1,150,000	$1,300,000
Retirement Planning	$400,000	$450,000	$475,000	$500,000
Risk Management Services	$250,000	$275,000	$300,000	$350,000
Tax Optimization Services	$200,000	$250,000	$225,000	$275,000

Sometimes, when dealing with large or complex datasets, aggregation can be more efficient using **SQL**.

Like programming languages, SQL databases can also enhance business software capabilities through seamless data integration and data transformation, while also offering efficient ad-hoc solutions with just a few lines of SQL code. When you need to transform rows into columns and aggregate data, the PIVOT function is available in SQL Server and Oracle. In other databases, you can achieve similar results by using a combination of CASE statements and aggregation functions.

To demonstrate the method of creating a Pivot Table for our use case that can be used in just about any database, we'll use **SQL Server**.

We'll start by creating a table called **Service_Revenue** using the DDL statement **CREATE TABLE** in a **Schema** called **Test**. Then we perform a **BULK INSERT** to import the csv file into the table to store the revenue data.

```sql
-- Create the Financial Services Revenue table for the 6 columns found in the csv file
CREATE TABLE Test.Service_Revenue (
    Year INT,
    Quarter INT,
    Month INT,
    Service_ID VARCHAR(4),
    Service_Name VARCHAR(100),
    Revenue DECIMAL(15, 2)
);

/* Insert the csv data into the Financial Services Revenue table, starting from the 2nd row,
using a comma as the delimeter and a newline as the row terminator */
BULK INSERT Test.Service_Revenue
FROM 'C:\[redacted]Financial_Securities\Financial_Services_Revenue.csv'
WITH (
    FIELDTERMINATOR = ',',
    ROWTERMINATOR = '\n',
    FIRSTROW = 2
);
```

Now we'll run a simple pivot table type query on the data. We create a **CTE (Common Table Expression)** called **Quarter_Revenue** using **WITH** to calculate the total revenue by **Service_Name** using **GROUP BY**, the **SUM** function to aggregate revenue for each service and quarter, and pivot the separate **Quarter** column values into columns using the **CASE** statement. And we **SELECT** from **Quarter_Revenue** and format the headers and Quarter column revenue values using the **FORMAT** function with US currency format. After executing the query, we are shown the desired results.

```sql
/* Run a query to create a pivot table that aggregates revenue by quarter for each service.
Use GROUP BY to aggregate the data, SUM to calculate the total revenue for each service and quarter,
pivot the separate Quarter values into columns using the CASE statement,
and format the headers and Quarter column values using the FORMAT function with US currency format */
WITH Quarter_Revenue AS (
    SELECT
        Service_Name,
        SUM(CASE WHEN Quarter = 1 THEN Revenue ELSE 0 END) AS Quarter1,
        SUM(CASE WHEN Quarter = 2 THEN Revenue ELSE 0 END) AS Quarter2,
        SUM(CASE WHEN Quarter = 3 THEN Revenue ELSE 0 END) AS Quarter3,
        SUM(CASE WHEN Quarter = 4 THEN Revenue ELSE 0 END) AS Quarter4
    FROM Test.Service_Revenue
    GROUP BY Service_Name
)
SELECT
    Service_Name AS "Service Name",
    FORMAT(Quarter1, 'C', 'en-US') AS "Quarter 1",
    FORMAT(Quarter2, 'C', 'en-US') AS "Quarter 2",
    FORMAT(Quarter3, 'C', 'en-US') AS "Quarter 3",
    FORMAT(Quarter4, 'C', 'en-US') AS "Quarter 4"
FROM Quarter_Revenue;
```

Results | Messages

Service Name	Quarter 1	Quarter 2	Quarter 3	Quarter 4
Estate Planning	$300,000.00	$350,000.00	$325,000.00	$400,000.00
Portfolio Management	$1,000,000.00	$1,200,000.00	$1,150,000.00	$1,300,000.00
Retirement Planning	$400,000.00	$450,000.00	$475,000.00	$500,000.00
Risk Management Services	$250,000.00	$275,000.00	$300,000.00	$350,000.00
Tax Optimization Services	$200,000.00	$250,000.00	$225,000.00	$275,000.00

This query uses the **PIVOT** function where the inner query, named as **Service_Revenue**, selects the only columns used in the aggregation, and the PIVOT function pivots the **Quarter** column values as **Columns** and uses the **SUM** function to calculate the total revenue for each service by Quarter. The outer query formats the headers and Quarter column revenue values using the **FORMAT** function with US currency format. After executing the query, we are shown the desired results which are the same as the prior method results.

```sql
/* Run a query to create a pivot table using the PIVOT function.
The inner query named as "Service_Revenue" selects the only columns used in the aggregation.
Then the PIVOT function will pivot the Quarter columns values as the Columns and use SUM function
to calculate the total revenue for each service by Quarter.
The outer query will format the headers and Quarter column values using the FORMAT function
with US currency format */
SELECT
    Service_Name AS "Service Name",
    FORMAT([1], 'C', 'en-US') AS "Quarter 1",
    FORMAT([2], 'C', 'en-US') AS "Quarter 2",
    FORMAT([3], 'C', 'en-US') AS "Quarter 3",
    FORMAT([4], 'C', 'en-US') AS "Quarter 4"
FROM
    (SELECT
        Service_Name,
        Quarter,
        Revenue
     FROM Test.Service_Revenue) AS Services_Revenue
PIVOT
    (SUM(Revenue) FOR Quarter IN ([1], [2], [3], [4])) AS PivotTable;
```

Results | Messages

Service Name	Quarter 1	Quarter 2	Quarter 3	Quarter 4
Estate Planning	$300,000.00	$350,000.00	$325,000.00	$400,000.00
Portfolio Management	$1,000,000.00	$1,200,000.00	$1,150,000.00	$1,300,000.00
Retirement Planning	$400,000.00	$450,000.00	$475,000.00	$500,000.00
Risk Management Services	$250,000.00	$275,000.00	$300,000.00	$350,000.00
Tax Optimization Services	$200,000.00	$250,000.00	$225,000.00	$275,000.00

Thank You

Comment

Repost

Save

linkedin.com/in/danvukota/

Dan Vukota