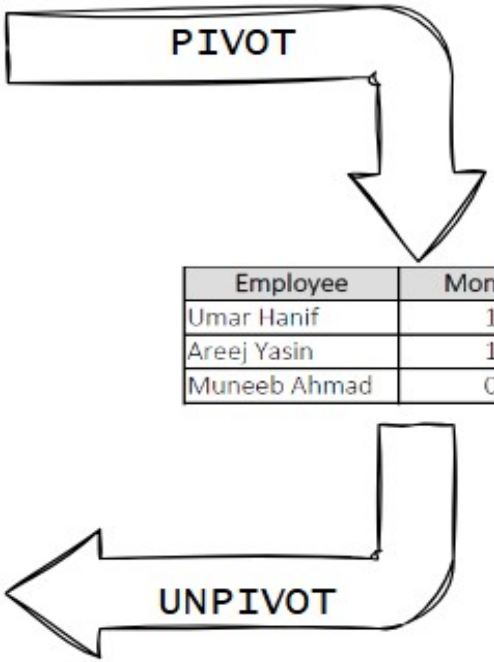
PIVOT vs UNPIVOT

Employee	Day	Present
Umar Hanif	Monday	1
Umar Hanif	Tuesday	1
Umar Hanif	Wednesday	1
Umar Hanif	Thursday	0
Umar Hanif	Friday	1
Areej Yasin	Monday	1
Areej Yasin	Tuesday	1
Areej Yasin	Wednesday	1
Areej Yasin	Thursday	1
Areej Yasin	Friday	1
Muneeb Ahmad	Monday	0
Muneeb Ahmad	Tuesday	1
Muneeb Ahmad	Wednesday	0
Muneeb Ahmad	Thursday	1
Muneeb Ahmad	Friday	1



Employee	Monday	Tuesday	Wednesday	Thursday	Friday
Umar Hanif	1	1	1	0	1
Areej Yasin	1	1	1	1	1
Muneeb Ahmad	0	1	0	1	1

PIVOT:

The PIVOT function in SQL is used to transform data by rotating unique values in one column into multiple columns. This can be particularly helpful for generating reports, summarizing data, or presenting data in a more organized manner. Here are examples with different use cases to demonstrate the flexibility and utility of PIVOT.

Syntax:

```
SELECT <columns>
FROM (
    SELECT <columns>
    FROM <table_name>
) AS SourceTable
PIVOT (
    <aggregate_function>(<column_to_aggregate>)
    FOR <column_with_unique_values> IN (<column_value1>,
<column_value2>, ...)
) AS PivotTable;
```

Summary of When to Use Each Aggregation in PIVOT:

- **SUM()**: Total values across categories (e.g., total sales, total quantity).
- **AVG()**: Average values across categories (e.g., average score, average sale price).
- **MAX()**: Maximum value within a category (e.g., highest temperature, peak sale).
- **COUNT()**: Count occurrences or records in categories (e.g., number of complaints, frequency of attendance).

Example 01: Sales Data by Month (Basic PIVOT Example)

Suppose you have a Sales table that records sales for different products each month:

Product	Month	Sales
Apples	January	150
Apples	February	200
Oranges	January	100
Oranges	February	180

Goal: Pivot the Month column values into separate columns so that each month becomes a column with the corresponding sales data for each product.

```
SELECT Product, [January], [February]
FROM (
    SELECT Product, Month, Sales
    FROM Sales
) AS SourceTable
PIVOT (
    SUM(Sales) FOR Month IN ([January], [February])
) AS PivotTable;
```

Product	January	February
Apples	150	200
Oranges	100	180

This example shows how to pivot data by turning the Month column values into columns and aggregating the Sales values using SUM.

Example 02: Employee Attendance by Day (Aggregation)

Consider a table called `Attendance` that records whether employees attended each day of the week:

EmployeeID	Day	Present
101	Monday	1
101	Tuesday	1
102	Monday	0
102	Tuesday	1

Goal: Transform the Day values into columns to get a weekly attendance summary for each employee.

```
SELECT EmployeeID, [Monday], [Tuesday]
FROM (
    SELECT EmployeeID, Day, Present
    FROM Attendance
) AS SourceTable
PIVOT (
    MAX(Present) FOR Day IN ([Monday], [Tuesday])
) AS PivotTable;
```

EmployeeID	Monday	Tuesday
101	1	1
102	0	1

In this case, the MAX function is used to get the Present status (0 or 1) for each day.

Example 03: Exam Scores by Subject (Pivoting with Multiple Aggregates)

Suppose you have an `ExamScores` table:

Student	Subject	Score
Alice	Math	85
Alice	Science	90
Bob	Math	88
Bob	Science	92

Goal: Show each student's scores for each subject as separate columns.

```
SELECT Student, [Math], [Science]
FROM (
    SELECT Student, Subject, Score
    FROM ExamScores
) AS SourceTable
PIVOT (
    AVG(Score) FOR Subject IN ([Math], [Science])
) AS PivotTable;
```

Student	Math	Science
Alice	85	90
Bob	88	92

In this example, AVG is used in case there were multiple scores per student in a subject.

Example 04: Product Orders by Quarter(Data Across Periods)

A table Orders contains the quantity of products ordered each quarter:

Product	Quarter	Quantity
Laptop	Q1	100
Laptop	Q2	150
Tablet	Q1	200
Tablet	Q2	50

Goal: Display total quantities ordered for each product by quarter.

```
SELECT Product, [Q1], [Q2]
FROM (
    SELECT Product, Quarter, Quantity
    FROM Orders
) AS SourceTable
PIVOT (
    SUM(Quantity) FOR Quarter IN ([Q1], [Q2])
) AS PivotTable;
```

Product	Q1	Q2
Laptop	100	150
Tablet	200	50

Here, SUM is used to add up the Quantity of each product ordered in each quarter.

Example 05: Counting Records by Category (Pivot for Count)

Suppose you have a table Complaints where you log complaints by type and resolution status:

ComplaintID	Type	Resolved
1	Billing	Yes
2	Service	No
3	Billing	Yes
4	Service	Yes

Goal: Count the number of complaints for each type by resolution status.

```
SELECT Type, [Yes] AS Resolved, [No] AS Unresolved
FROM (
    SELECT Type, Resolved
    FROM Complaints
) AS SourceTable
PIVOT (
    COUNT(ComplaintID) FOR Resolved IN ([Yes], [No])
) AS PivotTable;
```

Type	Resolved	Unresolved
Billing	2	0
Service	1	1

This example shows how PIVOT can be used to count records based on categories.

UNPIVOT:

The UNPIVOT operator in SQL is the opposite of PIVOT—it transforms columns into rows, making it useful for restructuring data that has been pivoted or for simplifying columnar data into a more normalized form.

Syntax:

```
SELECT <columns>
FROM (
    SELECT <columns>
    FROM <table_name>
) AS SourceTable
UNPIVOT (
    <column_with_values> FOR <new_column_name> IN (<columns_to_unpivot>)
) AS UnpivotTable;
```

Here's how the components work:

<**columns**>: Columns you want in the final output.
<**column_with_values**>: The name of the new column where the unpivoted values will be stored.
<**new_column_name**>: The name of the column that will indicate the original column names.
<**columns_to_unpivot**>: The list of columns being transformed into rows.

Example 01: Unpivoting Monthly Sales Data

Suppose you have a table named MonthlySales that stores monthly sales in separate columns:

Product	January	February
Product A	100	150
Product B	200	250

You want to transform the monthly columns into rows, making it easier to analyze sales by month.

```
SELECT Product, Month, Sales
FROM (
    SELECT Product, January, February, March
    FROM MonthlySales
) AS SourceTable
UNPIVOT (
    Sales FOR Month IN (January, February, March)
) AS UnpivotTable;
```

Product	Month	Sales
Product A	January	100
Product A	February	150
Product B	January	200
Product B	February	250

Sales FOR Month: The UNPIVOT operator takes values from January, February, and March columns, putting them under a new column Sales.
Month: This new column (Month) stores the original column names as values (January, February, March).

Example 02: Unpivoting Survey Data for Easier Analysis

Imagine a table SurveyResponses where each row represents a respondent's answer to several questions, with each question as a separate column:

Respondent	Q1	Q2	Q3
1	Yes	No	Yes
2	No	Yes	No

Analyze responses, unpivot data so each question and response are in separate rows.

```
SELECT Respondent, Question, Response
FROM (
    SELECT Respondent, Q1, Q2, Q3
    FROM SurveyResponses
) AS SourceTable
UNPIVOT (
    Response FOR Question IN (Q1, Q2, Q3)
) AS UnpivotTable;
```

Respondent	Question	Response
1	Q1	Yes
1	Q2	No
1	Q3	Yes
2	Q1	No
2	Q2	Yes
2	Q3	No

Response FOR Question: Moves the values of Q1, Q2, and Q3 columns to a single Response column, while the new Question column stores the original column names.

Example 03: Converting Financial Data for Budget Analysis

Suppose you have financial data by category and quarter in a table called BudgetData:

Category	Q1	Q2	Q3	Q4
Marketing	2000	2500	3000	4000
R&D	3000	3500	4000	5000

To analyze budget allocation per quarter more flexibly, unpivot the quarter columns into rows.

```
SELECT Category, Quarter, Amount
FROM (
    SELECT Category, Q1, Q2, Q3, Q4
    FROM BudgetData
) AS SourceTable
UNPIVOT (
    Amount FOR Quarter IN (Q1, Q2, Q3, Q4)
) AS UnpivotTable;
```

Category	Quarter	Amount
Marketing	Q1	2000
Marketing	Q2	2500
Marketing	Q3	3000
Marketing	Q4	4000
R&D	Q1	3000
R&D	Q2	3500
R&D	Q3	4000
R&D	Q4	5000

KEY DIFFERENCE BETWEEN PIVOT AND UNPIVOT:

Purpose:
PIVOT: Used to convert rows into columns. It summarizes data by turning unique row values into new columns, often for better readability or reporting.
UNPIVOT: Used to convert columns into rows, which helps normalize wide tables by flattening data into a more analyzable structure.

Use Cases:
PIVOT: Typically used in reporting and data visualization scenarios where you want to compare values across different categories or time periods side by side.
UNPIVOT: Often used in data preparation when data needs to be converted to a long format, making it easier to aggregate or process in analytics.

Output Structure:
PIVOT: Increases the number of columns and reduces the number of rows. Each unique value in the pivot column becomes a column header.
UNPIVOT: Reduces the number of columns and increases the number of rows, transforming column headers into row values.

Performance Consideration:
PIVOT and UNPIVOT operations may have different performance impacts based on the amount of data and the specific SQL database engine's optimization capabilities.