Window functions in SQL

Here's a rephrased version:

SQL window functions allow for computations over a specific set of rows linked to the current row. They differ from standard aggregate functions by providing results for each individual row rather than summarizing entire data groups. These functions enable tasks such as calculating a seven-day rolling sales total for every day or ranking a product's sales performance within its category.

OVER clause: Defines the window

Syntax:

```
SELECT column_name1,
 window_function(cloumn_name2)
 OVER([PARTITION BY column_name1] [ORDER BY column_name3]) AS
new_column
FROM table_name;


window_function= any aggregate or ranking function
column_name1= column to be selected
column_name2= column on which window function is to be applied
column_name3= column on whose basis partition of rows is to be
done
new_column= Name of new column
table_name= Name of table
```

SQL Example:

SELECT Name, Age, Department, Salary,

AVG(Salary) OVER(PARTITION BY Department) AS Avg_Salary

FROM employee

Name	Age	Department	Salary	Avg_Salary
Ramesh	20	Finance	50,000	40,000
Suresh	22	Finance	50,000	40,000
Ram	28	Finance	20,000	40,000
Deep	25	Sales	30,000	25,000
Pradeep	22	Sales	20,000	25,000

The output

Using Pandas:

```
data = {
   'Name': ['Ramesh', 'Suresh', 'Ram', 'Deep', 'Pradeep'],
   'Age': [20, 22, 28, 25, 22],
   'Department': ['Finance', 'Finance', 'Finance', 'Sales', 'Sales'],
   'Salary': [50000, 50000, 20000, 30000, 20000]
}

df = pd.DataFrame(data)

df['Avg_Salary'] = df.groupby('Department')['Salary'].transform('mean')

df['Salary'] = df['Salary'].apply(lambda x: f"{x:,}")
df['Avg_Salary'] = df['Avg_Salary'].apply(lambda x: f"{int(x):,}")
```

Reference: https://www.geeksforgeeks.org/window-functions-in-sql/