CASE 2
READ
ANALYSIS

Microsoft®
SQL Server®

Integration

With

AND

jupyter

The Microsoft SQL Server Integration Project (Read Analysis), or CASE 2, aims to demonstrate the powerful synergy between and Microsoft SQL Server for the purposes of Exploratory Data Analysis (EDA) and visualization of business insights

1. Context and Overview

The Microsoft SQL Server Integration Project (CASE 2) uses (specifically pandas and matplotlib) to perform Exploratory Data Analysis (EDA) and generate visualizations from the ContosoRetailDW data warehouse. This project demonstrates the seamless synergy between Microsoft SQL Server data and data science tools

2. Preparing the Database for Analysis

We will focus on the existing DimProduct table

```
SELECT ColorName, UnitPrice FROM DimProduct
```

```
SELECT
        ColorName,
        UnitPrice
FROM DimProduct;
```

View Code

This query will be used in Python to:

- ▲ Analyze the total number of products per color.
- ■ Explore product price distribution.
- ● Create visual summaries using graphs

3. Initial Configurations in jupyter

Step 1 – Install required libraries:

View Code

```
!pip install pyodbc pandas matplotlib
```

```
In [1]: pip install pyodbc
```

Step 2 – Import libraries and set up connection:

View Code

```
[2]: import pyodbcimport pyodbc
     import pandas as pd
     import matplotlib.pyplot as plt

     connection_data = (
         "Driver={SQL Server};"
         "Server=LAPTOP-SRP0M4NC;"
         "Database=ContosoRetailDW;"
     )

     connection = pyodbc.connect(connection_data)
     print("Connection successful!")
```

3. Initial Configurations in jupyter

Step 2 – Import libraries and set up connection:

View Code

```
import pyodbc
import pandas as pd
import matplotlib.pyplot as plt

connection_data = (
    "Driver={SQL Server};"
    "Server=SEU_HOSTNAME;"
    "Database=ContosoRetailDW;"
)

connection = pyodbc.connect(connection_data)
print("Connection successful!")
```

- **pyodbc**: connects to Microsoft SQL Server
- **pandas**: data analysis and manipulation
- **matplotlib**: plotting graphs and visualizations

This setup enables direct integration and interaction with SQL Server data

3. Initial Configurations in jupyter

Step 3 – Define SQL Query and Extract Data:

View Code

```
sql_command = "SELECT ColorName, UnitPrice FROM DimProduct"

data_df = pd.read_sql(sql_command, connection)
```

```
[3]: sql_command = "SELECT ColorName, UnitPrice FROM DimProduct"

     data_df = pd.read_sql(sql_command, connection)
```

Defined SQL Query: Created a request to select product color and price data

Executed Query: Ran the SQL request against the database connection.

Loaded Data: Stored the results as a Pandas DataFrame (data_df) in

3. Initial Configurations in jupyter

Data Display

View Code

display(data_df)

```
[4]: display(data_df)
```

	ColorName	UnitPrice
0	Silver	12.99
1	Blue	12.99
2	White	14.52
3	Silver	21.57
4	Red	21.57
...	...	...
2512	Red	129.99
2513	White	129.99
2514	White	3.35
2515	Black	3.35
2516	Silver	3.35

2517 rows × 2 columns

It displayed the table of product data (ColorName and UnitPrice) that you just extracted from the database

4. Reading Data from Microsoft SQL Server

View Code

```
total_products_by_color = data_df.groupby('ColorName').count()
```

```
[5]: total_products_by_color = data_df.groupby('ColorName').count()
```

ColorName	UnitPrice
Azure	14
Black	602
Blue	197
Brown	77
Gold	50
Green	74
Grey	283
Orange	55
Pink	84
Purple	6
Red	99
Silver	417
Silver Grey	14
Transparent	1
White	505
Yellow	38
blue	3

```
display(total_products_by_color)
```

```
[6]: display(total_products_by_color)
```

- **This shows the total number of products for each color**
- **groupby organizes the data for aggregation**

4. Reading Data from Microsoft SQL Server

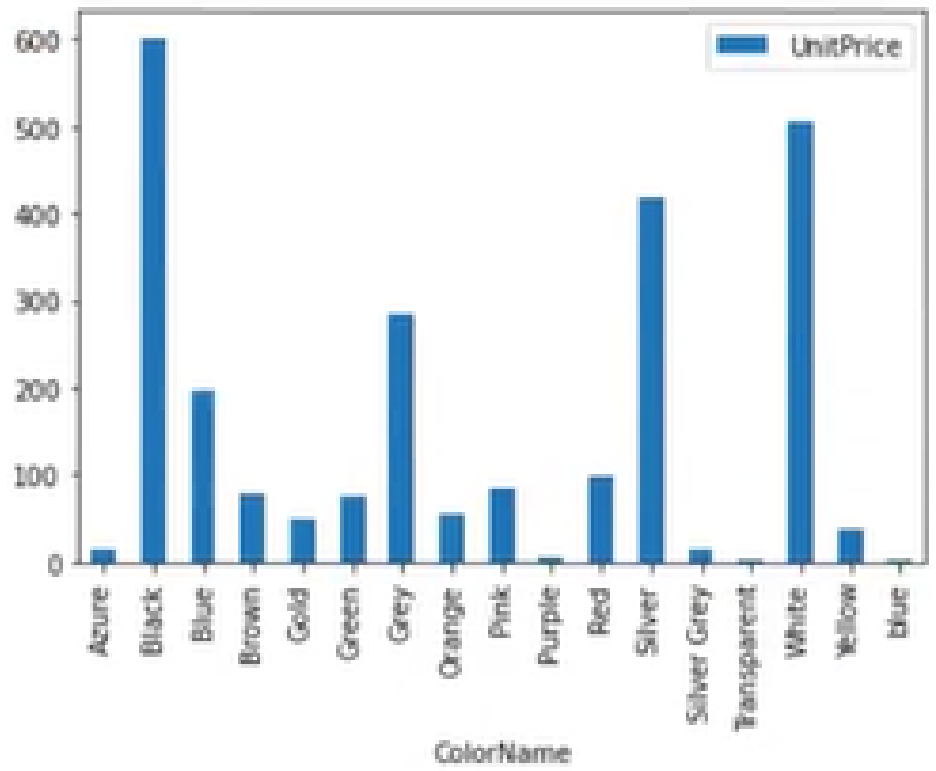
View Code

```
total_products_by_color.plot(kind='bar')
```

```
[7]: total_products_by_color.plot(kind='bar')
```

- Simple bar chart to visualize the distribution of products by color
- Makes it easier to interpret the dataset and identify trends

```
<AxesSubplot:xlabel='ColorName'>
```



Can also aply :

total_products_by_color.plot(kind='bar', ylabel='Quantity')

5. Conclusion

The Python and Microsoft SQL Server Integration Project (Read Analysis) demonstrates in practice how to:

- Connect Python directly to SQL Server using pyodbc.
- Execute SQL queries inside Jupyter Notebook.
- Perform basic grouping and aggregation with pandas.
- Create visual summaries with matplotlib.

This project showcases the potential of Python and SQL Server integration for data exploration and KPI analysis. It's a practical approach to understanding database content quickly and generating insights without advanced complexity.