Exploratory Data Analysis (EDA) and data visualization

Exploratory Data Analysis (EDA) and data visualization are crucial steps in understanding and gaining insights from your data. Databricks provides a powerful platform for performing EDA and visualizing data using its integrated tools and libraries such as Apache Spark, Matplotlib, Seaborn, and Plotly. Here's a basic guide on how to perform EDA and visualize data in Databricks:

1.Importing Data:

Start by importing your dataset into Databricks. You can do this by uploading your data file directly to Databricks or by connecting Databricks to your data source (like Azure Blob Storage, AWS S3, or any other data lake).

2.Data Exploration:

Once your data is imported, begin exploring it to understand its structure, contents, and quality. You can use various Spark DataFrame operations and functions to perform basic exploration tasks such as checking the schema, summarizing statistics, and sampling the data.

python

```
# Read data into a Spark DataFrame
df = spark.read.csv("path_to_your_data.csv", header=True, inferSchema=True)

# Display the schema
df.printSchema()

# Summary statistics
df.describe().show()
```

 # Sample data

df.sample(False, 0.1).show()

1.Data Visualization:

After getting a basic understanding of the data, visualize it to gain deeper insights. Databricks supports multiple visualization libraries like Matplotlib, Seaborn, and Plotly. You can create various types of plots such as histograms, scatter plots, box plots, etc., to visualize the distribution, relationships, and trends in your data.

python

 # Import required libraries

import matplotlib.pyplot as plt

import seaborn as sns

 # Example: Histogram

sns.histplot(df.select("column_name").toPandas(), bins=20)

plt.title("Histogram of Column")

plt.xlabel("Values")

plt.ylabel("Frequency")

plt.show()

 # Example: Scatter Plot

sns.scatterplot(x="column1", y="column2", data=df.toPandas())

plt.title("Scatter Plot")

plt.xlabel("Column 1")

plt.ylabel("Column 2")

plt.show()

- **Interactive Visualization**:

For interactive and more sophisticated visualizations, you can use libraries like Plotly. Plotly allows you to create interactive plots that can be customized and explored interactively within Databricks notebooks.

python

 # Import Plotly

import plotly.express as px

 # Example: Interactive Scatter Plot

fig = px.scatter(df.toPandas(), x="column1", y="column2", color="category_column", title="Interactive Scatter Plot")

fig.show()

- **Dashboard Creation (Optional):**

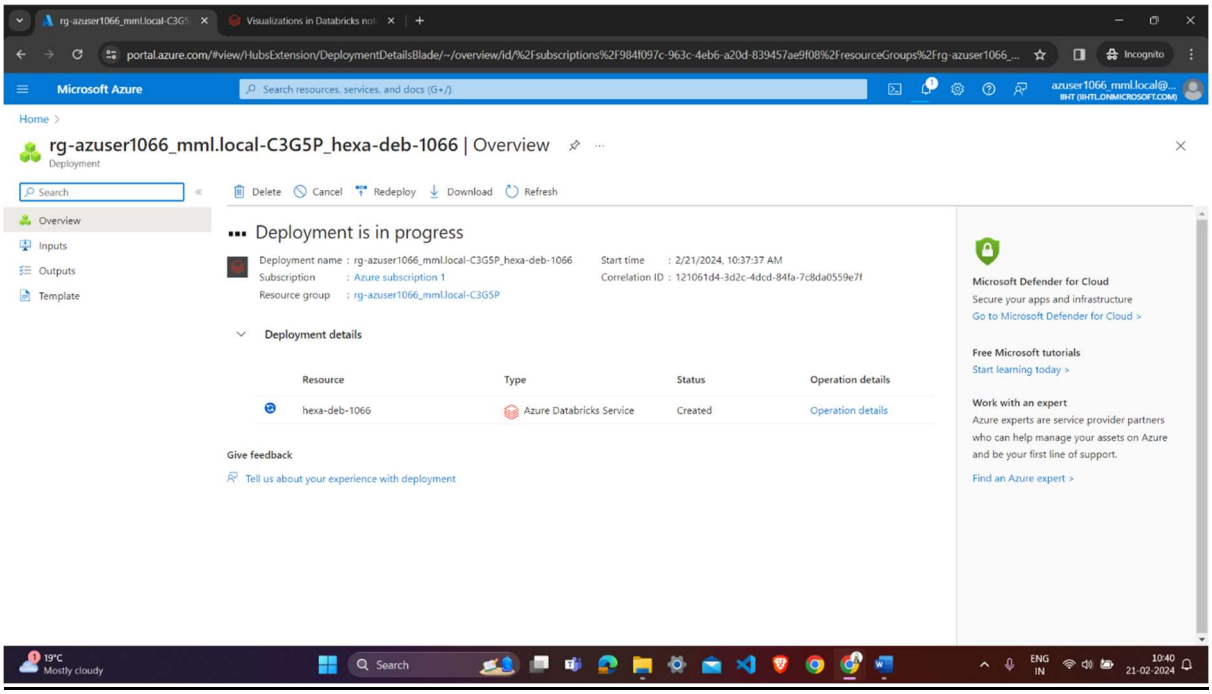
If you want to create interactive dashboards with multiple visualizations, you can use Databricks' built-in dashboarding capabilities or integrate with external dashboarding tools like Tableau or Power BI.

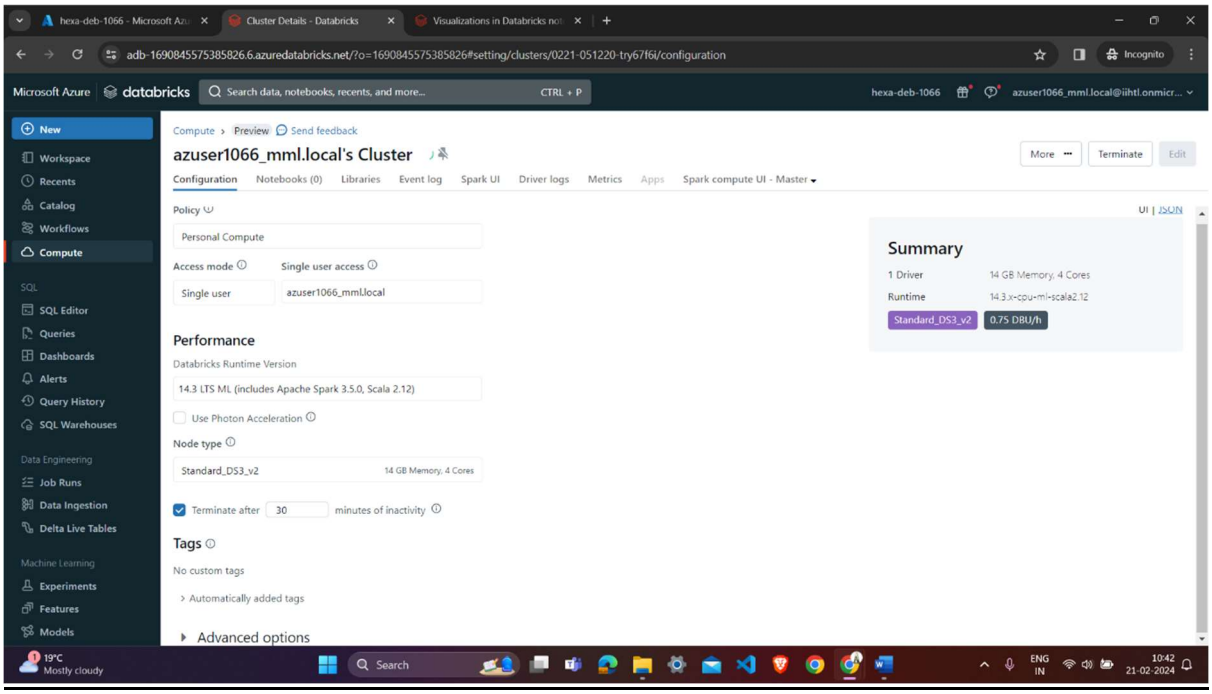
- **Iterate and Refine:**

EDA is an iterative process. Continue exploring and visualizing your data, refine your analysis based on insights gained, and repeat until you have a good understanding of your data.

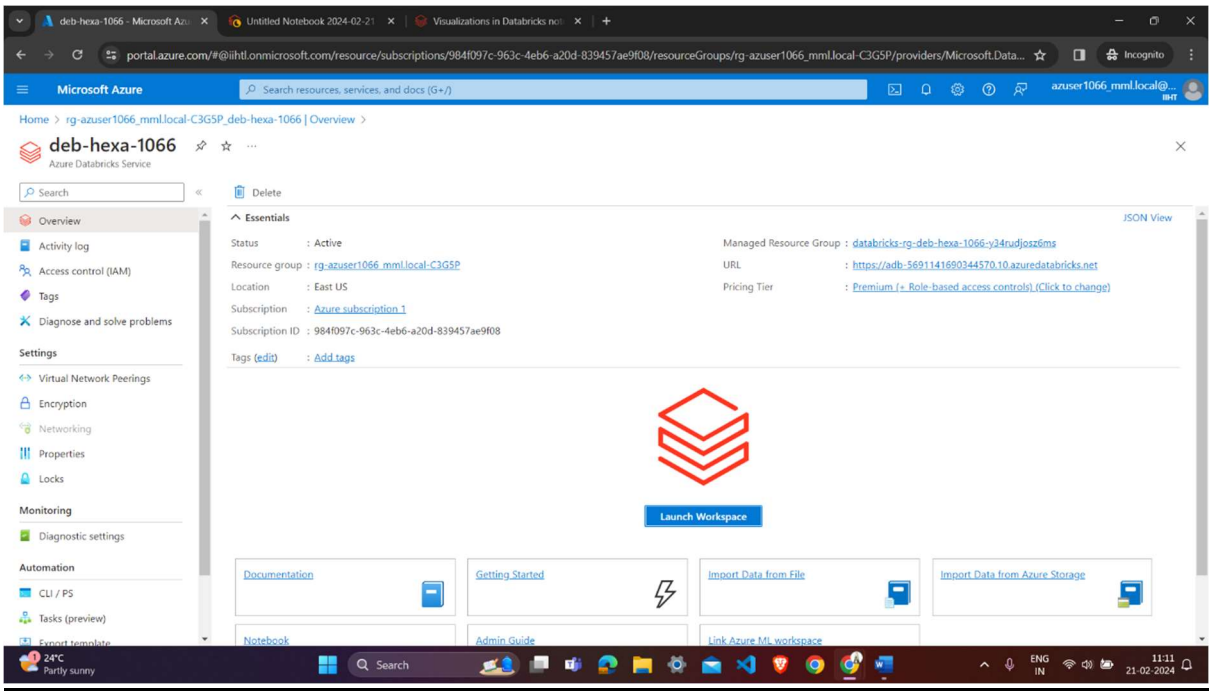
Activity:-

- Create cluster hexa-deb-1066 personalize policy and terminate after 30 min

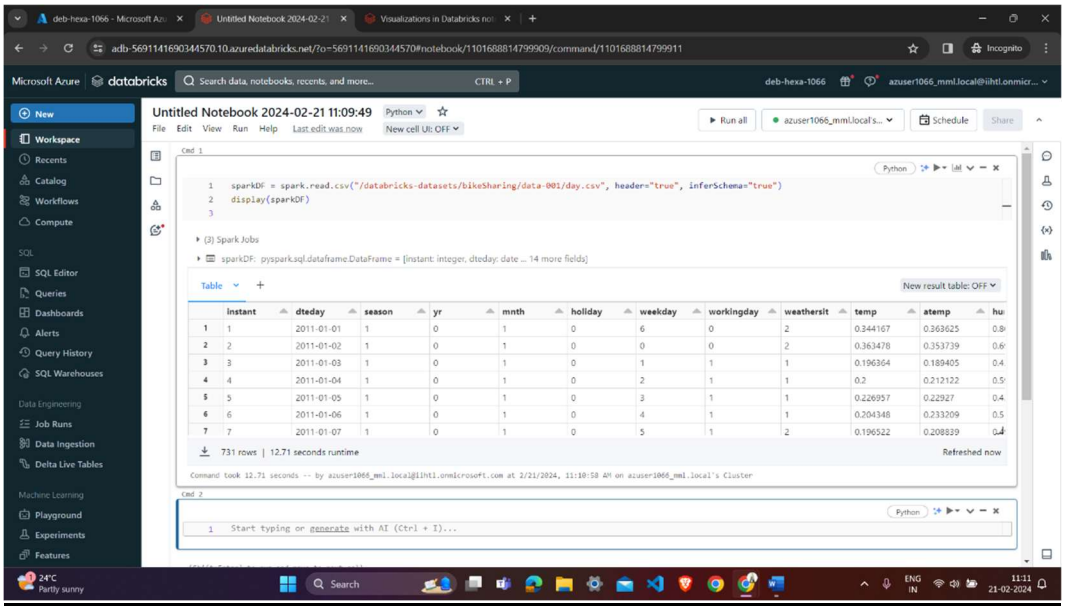




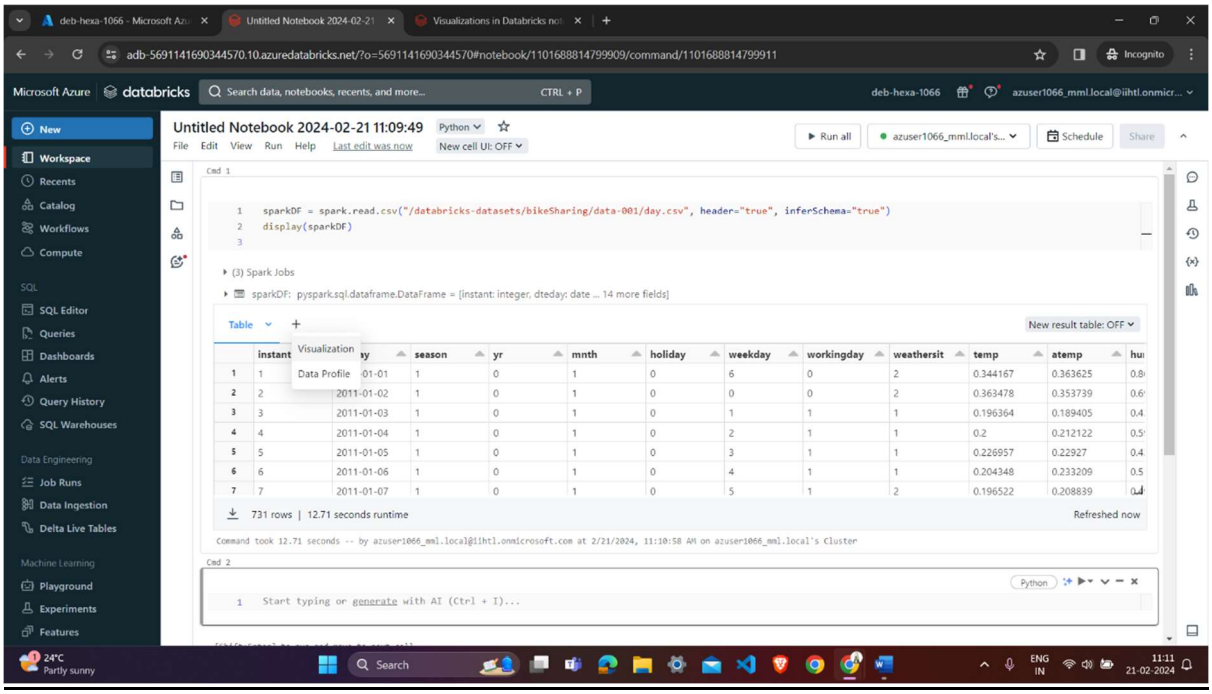
Launch workspace for databricks:-



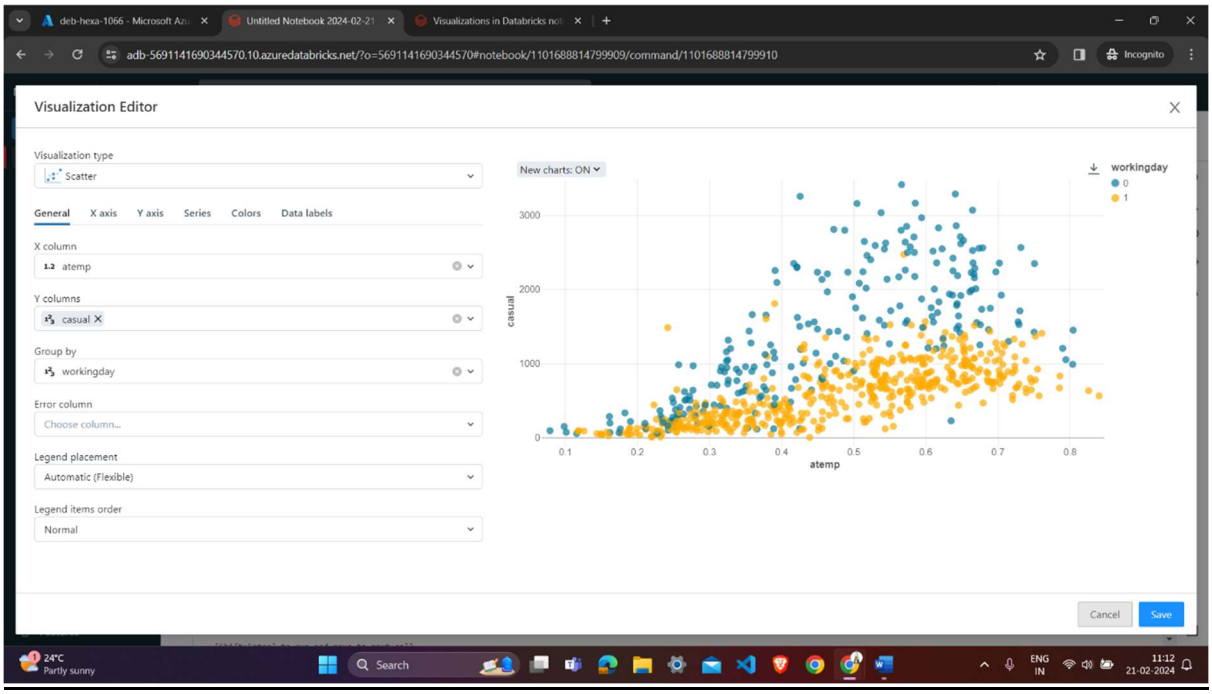
Write code for display data :-



Click on the + icons and get option of visualization:-



Select scatter vizulation type and x column y column group by and placement get a scatterd graph :-



Overview of 3 level namespace and creating Unity Catalog objects:-

The Unity Catalog within Databricks offers a comprehensive solution for managing data and AI assets across Azure Databricks workspaces. It provides centralized access control, auditing, lineage, and data discovery capabilities. Here's an explanation of the key concepts and processes involved in using Unity Catalog, including the three-level namespace and creating Unity Catalog objects:

1.Three-Level Namespace:

Unity Catalog organizes data assets using a three-level namespace structure:

- **Metastore:** This is the top-level container for metadata. It provides a three-level namespace (catalog.schema.table) to organize data.
- **Catalog:** The first layer of the namespace hierarchy, used to categorize data assets.
- **Schema:** Also known as databases, schemas organize tables and views within a catalog.

2.Creating Unity Catalog Objects:

- **Metastore:** Administrators create a metastore for each region, attaching it to Databricks workspaces in the same region. It registers metadata about data and AI assets and manages access permissions.
- **Catalog:** Users can create catalogs to organize data assets. Each catalog contains schemas, and users with appropriate permissions can manage catalogs.

- **Schema:** Schemas, or databases, organize tables and views. They provide logical separation within catalogs. Users can create schemas and manage their contents.

3.Unity Catalog Objects:

- **Tables:** Tables reside within schemas and contain rows of data. They can be managed or external. Managed tables are managed by Unity Catalog, while external tables manage their own lifecycle and file layout.
- **Views**: Views are read-only objects created from tables and views in the metastore. They provide a logical representation of data without storing it physically.
- **Volumes:** Volumes are in Public Preview and provide access to non-tabular data stored in cloud object storage. They contain directories and files and offer governance for non-tabular data.
- **Models:** Although not strictly data assets, registered models can also be managed within Unity Catalog and reside at the lowest level in the object hierarchy.

4.Creating Objects:

- **Managed Storage:** Managed tables and volumes can be stored at the metastore, catalog, or schema levels. Managed storage locations are declared during object creation and determine where data and metadata files are stored.
- **Storage Credentials and External Locations:** To manage access to underlying cloud storage for external tables, external volumes, and managed storage, Unity Catalog uses storage credentials and external locations.

5.Identity Management:

- Unity Catalog uses identities from the Azure Databricks account to resolve users, service principals, and groups for enforcing permissions.
- Users, service principals, and groups must be added to workspaces to access Unity Catalog data. This process is called identity federation, and it enables access to Unity Catalog across workspaces.

In summary, Unity Catalog provides a structured approach to managing data and AI assets within Databricks, offering centralized governance, access control, auditing, and data discovery capabilities. Through its three-level namespace and various object types, users can organize, secure, and manage their data effectively.