Dbtufils.fs.help(‘ls ‘)

1. `dbutils.fs` (Databricks File System)

Path = « dbfs:/mnt/path/…csv»

Le module `dbutils.fs` est utilisé pour interagir avec le système de fichiers DBFS. Voici les principales commandes :

- **Lister les fichiers (`ls`)**

  ```
  python
  Copier le code
  dbutils.fs.ls("/path")
  ```

  Liste les fichiers et répertoires dans un chemin spécifique.

- **Copier des fichiers (`cp`)**

  ```
  python
  Copier le code
  dbutils.fs.cp("/source/path", "/destination/path", recurse=True)
  ```

  Copie un fichier ou un répertoire d'une destination à une autre. Le paramètre `recurse=True` permet de copier récursivement tout le contenu d'un répertoire.

- **Déplacer ou renommer des fichiers (`mv`)**

  ```
  python
  Copier le code
  dbutils.fs.mv("/source/path", "/destination/path")
  ```

  Déplace ou renomme un fichier ou un répertoire.

- **Supprimer des fichiers ou répertoires (`rm`)**

  ```
  python
  Copier le code
  dbutils.fs.rm("/path", recurse=True)
  ```

  Supprime un fichier ou un répertoire (récursivement si nécessaire avec `recurse=True`).

- **Créer un répertoire (`mkdirs`)**

  ```
  python
  Copier le code
  dbutils.fs.mkdirs("/path")
  ```

  Crée un répertoire à l'emplacement spécifié.

- **Lire les premiers octets d'un fichier (`head`)**

  ```
  python
  Copier le code
  dbutils.fs.head("/path/to/file", maxBytes=1024)
  ```

Affiche les premiers octets d'un fichier (par défaut 1024 octets).

- **Afficher les statistiques d'un fichier (`stat`)**

```
python
Copier le code
file_info = dbutils.fs.ls("/path/to/file")[0]
print(file_info.name, file_info.size)
```

Récupère des informations sur un fichier, telles que son nom et sa taille.

2. `dbutils.notebook` (Gestion des notebooks)

`dbutils.notebook` permet d'interagir avec des notebooks Databricks.

- **Exécuter un autre notebook (`run`)**

```
python
Copier le code
dbutils.notebook.run("/path/to/notebook", timeout_seconds=60,
arguments={"key": "value"})
```

Exécute un autre notebook Databricks avec un délai spécifié (en secondes) et des arguments facultatifs.

- **Arrêter un notebook en cours (`exit`)**

```
python
Copier le code
dbutils.notebook.exit("Message or data")
```

Arrête l'exécution du notebook actuel et renvoie un message ou des données.

3. `dbutils.widgets` (Widgets interactifs)

Les widgets permettent d'ajouter des entrées interactives dans les notebooks.

- **Créer un widget de texte (`text`)**

```
python
Copier le code
dbutils.widgets.text("name", "default_value")
```

Crée un champ de saisie de texte avec une valeur par défaut.

- **Créer un widget de sélection (`dropdown`)**

```
python
Copier le code
dbutils.widgets.dropdown("dropdown_name", "default", ["option1",
"option2", "option3"])
```

Crée un menu déroulant avec plusieurs options.

- **Obtenir la valeur d'un widget (`get`)**

```
python
Copier le code
value = dbutils.widgets.get("name")
```

Récupère la valeur actuelle du widget spécifié.

- **Supprimer un widget ou tous les widgets (`remove` ou `removeAll`)**

```
python
Copier le code
dbutils.widgets.remove("name")
dbutils.widgets.removeAll()
```

Supprime un widget spécifique ou tous les widgets.

4. `dbutils.secrets` (Gestion des secrets)

`dbutils.secrets` permet de gérer les secrets en toute sécurité dans Databricks, comme les clés API ou les mots de passe.

- **Obtenir un secret (`get`)**

```
python
Copier le code
dbutils.secrets.get(scope="scope_name", key="secret_key")
```

Récupère un secret depuis un scope spécifique.

- **Lister les scopes (`listScopes`)**

```
python
Copier le code
dbutils.secrets.listScopes()
```

Liste tous les scopes de secrets disponibles.

5. `dbutils.library` (Gestion des bibliothèques)

`dbutils.library` gère les bibliothèques installées sur les clusters Databricks.

- **Installer une bibliothèque (`install`)**

```
python
Copier le code
dbutils.library.install("library_name")
```

Installe une bibliothèque sur le cluster.

- **Désinstaller une bibliothèque (`uninstall`)**

```
python
Copier le code
dbutils.library.uninstall("library_name")
```

Désinstalle une bibliothèque du cluster.

- **Redémarrer le cluster après une modification des bibliothèques (`restartPython`)**

  ```
  python
  Copier le code
  dbutils.library.restartPython()
  ```

  Redémarre l'environnement Python après l'installation ou la désinstallation de bibliothèques.

6. Commandes Spark

Databricks est basé sur Apache Spark, vous utiliserez donc souvent les commandes Spark pour manipuler des données massives.

- **Lire un fichier CSV en tant que DataFrame**

  ```
  python
  Copier le code
  df = spark.read.csv("/path/to/csv", header=True, inferSchema=True)
  ```

- **Afficher les 5 premières lignes d'un DataFrame**

  ```
  python
  Copier le code
  df.show(5)
  ```

- **Enregistrer un DataFrame en tant que fichier Parquet**

  ```
  python
  Copier le code
  df.write.parquet("/path/to/output")
  ```

- **Exécuter une requête SQL sur un DataFrame**

  ```
  python
  Copier le code
  df.createOrReplaceTempView("table_name")
  spark.sql("SELECT * FROM table_name").show()
  ```

7. `dbutils.fs.mounts` (Gestion des montages de systèmes de fichiers)

- **Lister les montages de fichiers (`mounts`)**

  ```
  python
  Copier le code
  dbutils.fs.mounts()
  ```

  Liste tous les systèmes de fichiers montés sur DBFS.

- **Monter un nouveau système de fichiers**

  ```
  python
  Copier le code
  dbutils.fs.mount(
      source = "s3a://bucket_name",
      mount_point = "/mnt/bucket_name",
  ```

```
    extra_configs = {"aws_access_key_id": "YOUR_ACCESS_KEY",
"aws_secret_access_key": "YOUR_SECRET_KEY"}
)
```

Monte un bucket S3 ou un autre système de fichiers sur DBFS.

- **Démonter un système de fichiers**

```
python
Copier le code
dbutils.fs.unmount("/mnt/bucket_name")
```

Démonte un système de fichiers du DBFS.

Liste tous les fichiers.

%fs

LS

%sh

ps

Commandes en shell