Utiliser PyCharm pour lire les données d'une base de données MySQL dans des pandas

Par Jodie Burchell - Delphine Massenhove (traducteur)

Date de publication : 27 mars 2023

Pour réagir au contenu de cet article, un espace de dialogue vous est proposé sur le forum.
Commentez

Introduction....3
I - Visualiser le contenu de la base de données....3
II - Lire les données avec MySQL Connector....7
III - Lire les données avec SQLAlchemy....9
IV - Options avancées pour travailler avec les bases de données....10
V - Remerciements Developpez.com....14

Copyright ® 2023 Jodie Burchell. Aucune reproduction, même partielle, ne peut être faite de ce site et de l'ensemble de son contenu : textes, documents, images, etc. sans l'autorisation expresse de l'auteur. Sinon vous encourez selon la loi jusqu'à trois ans de prison et jusqu'à 300 000 € de dommages et intérêts.

Introduction

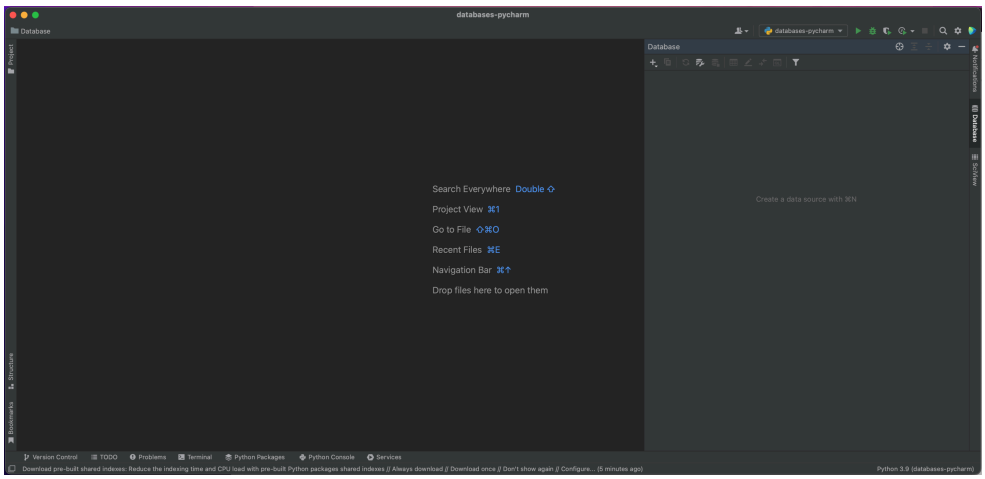
Passer de la lecture d'un fichier CSV stocké localement dans des pandas à la connexion à une base de données pour l'interroger peut s'avérer complexe. Dans cet article, nous expliquons comment lire des données stockées dans une base de données MySQL dans des pandas et présentons plusieurs fonctionnalités de PyCharm qui facilitent la réalisation de ce type de tâches.

I - Visualiser le contenu de la base de données

Dans ce tutoriel, nous allons lire des données concernant les retards et les annulations de vols de compagnies aériennes à partir d'une base de données MySQL dans un DataFrame pandas. Ces données sont une version de l'ensemble de données « **Airline Delays from 2003-2016** » de **Priank Ravichandar** sous licence **CC0 1.0**.

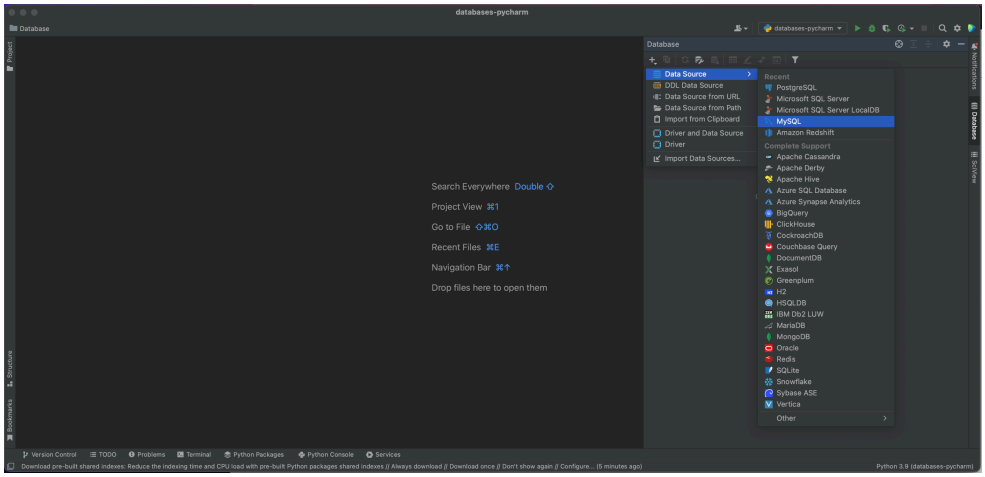
Lorsqu'on travaille avec des bases de données, l'un des points les plus frustrants est de ne pas avoir de vue d'ensemble des données disponibles, les tables étant stockées sur un serveur distant. La première fonctionnalité de PyCharm que nous allons utiliser est la fenêtre d'outils *Database*, qui permet de se connecter à une base de données et de réaliser une introspection complète avant d'effectuer toute requête.

Pour se connecter à notre base de données MySQL, il faut tout d'abord cliquer sur la fenêtre d'outils *Database* située à droite dans PyCharm.

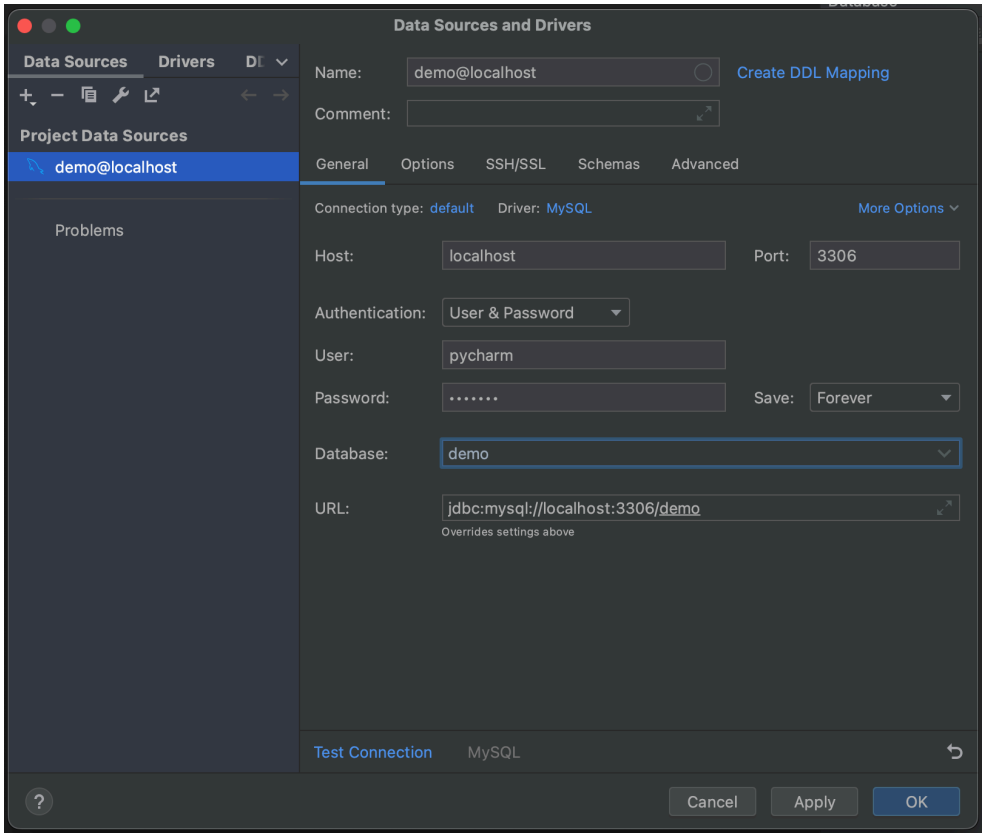


En haut à gauche de cette fenêtre, vous verrez un bouton plus. Cliquez sur ce bouton pour afficher la boite de dialogue avec menu déroulant et sélectionnez *Data Source* | *MySQL*.

Copyright ® 2023 Jodie Burchell. Aucune reproduction, même partielle, ne peut être faite de ce site et de l'ensemble de son contenu : textes, documents, images, etc. sans l'autorisation expresse de l'auteur. Sinon vous encourez selon la loi jusqu'à trois ans de prison et jusqu'à 300 000 € de dommages et intérêts.

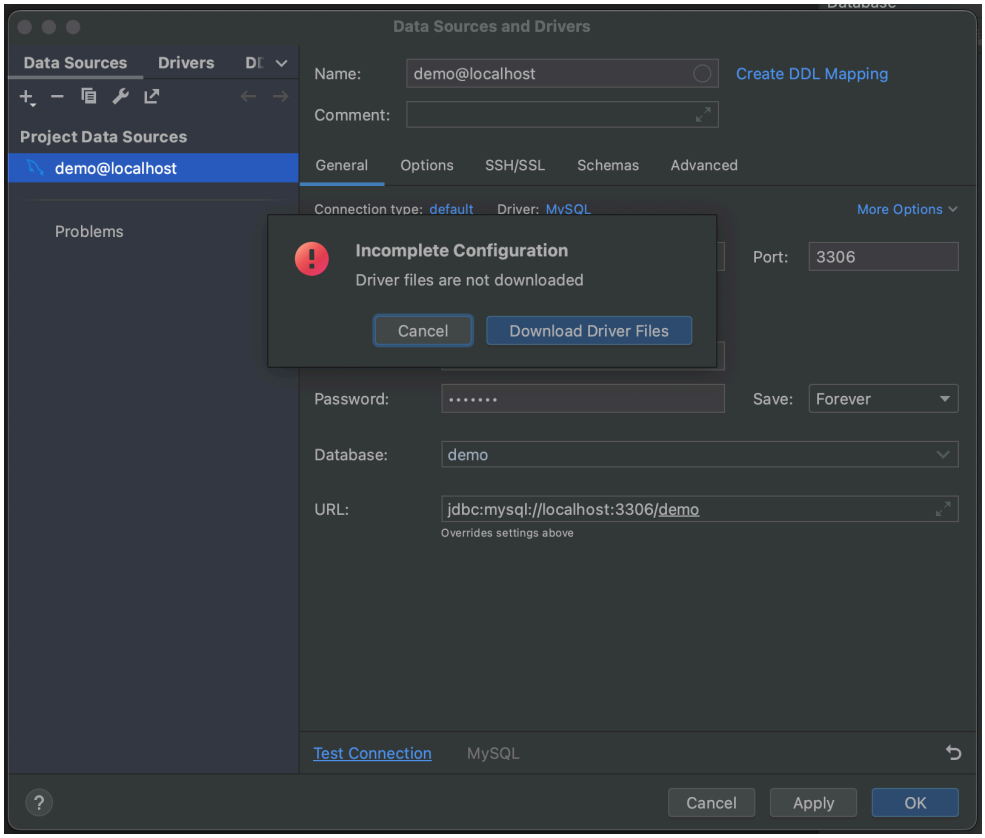


Cette fenêtre contextuelle nous permet de nous connecter à notre base de données MySQL. Dans le cas présent, nous utilisons une base de données hébergée localement et conservons donc l'hôte « localhost » et le port par défaut « 3306 » comme paramètres de connexion. Nous utilisons l'option d'authentification « User & Password » et saisissons « pycharm » dans le champ User comme dans le champ Password. Puis, nous attribuons le nom « demo » à notre Database. Bien entendu, pour vous connecter à votre propre base de données MySQL, vous aurez besoin de l'hôte spécifique, du nom de la base de données, de votre nom d'utilisateur et de votre mot de passe. Consultez **la documentation** pour voir l'ensemble des options de connexion.



Ensuite, cliquez sur *Test Connection*. PyCharm nous indique que les fichiers du pilote ne sont pas installés. Continuez et cliquez sur *Download Driver Files*. La fenêtre d'outils *Database* fournit une fonctionnalité qui trouve et installe automatiquement le pilote requis à notre place.

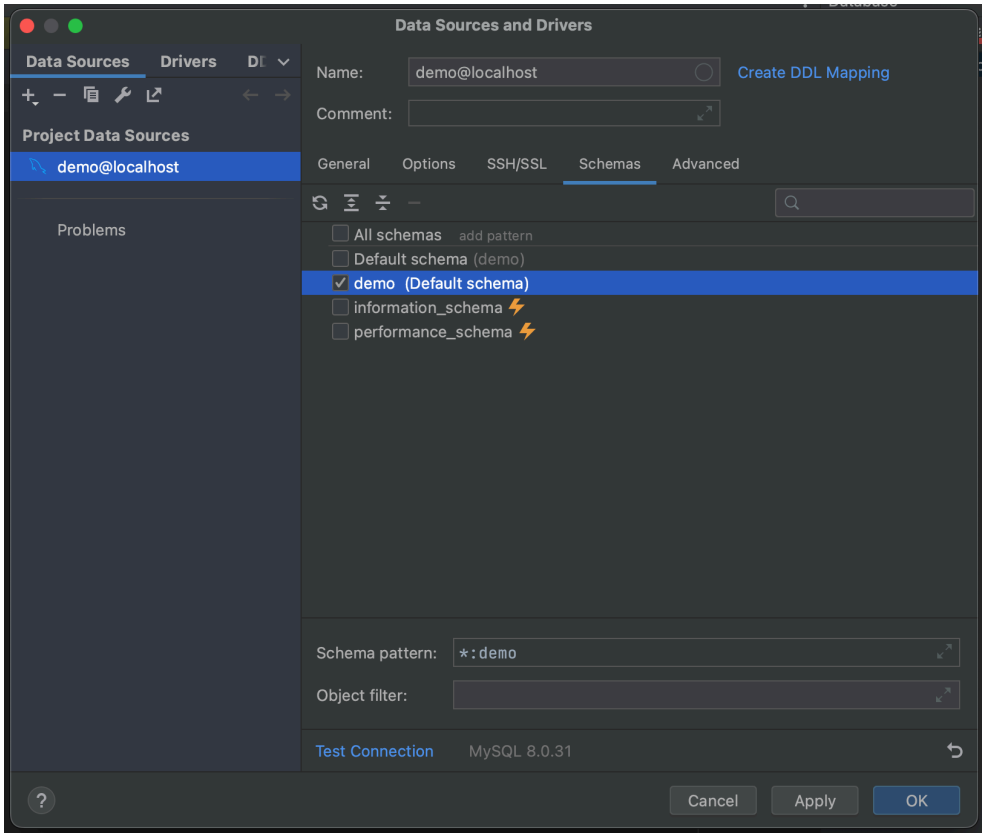
Copyright ® 2023 Jodie Burchell. Aucune reproduction, même partielle, ne peut être faite de ce site et de l'ensemble de son contenu : textes, documents, images, etc. sans l'autorisation expresse de l'auteur. Sinon vous encourez selon la loi jusqu'à trois ans de prison et jusqu'à 300 000 € de dommages et intérêts.



Et voilà ! Nous sommes maintenant connectés à notre base de données. Nous pouvons aller dans l'onglet *Schemas* et sélectionner les schémas que nous voulons introspecter. Dans notre exemple de base de données, il n'y en a qu'un (« demo »), mais si vous travaillez avec de très grandes bases de données, vous pouvez gagner du temps en limitant l'introspection aux schémas pertinents.

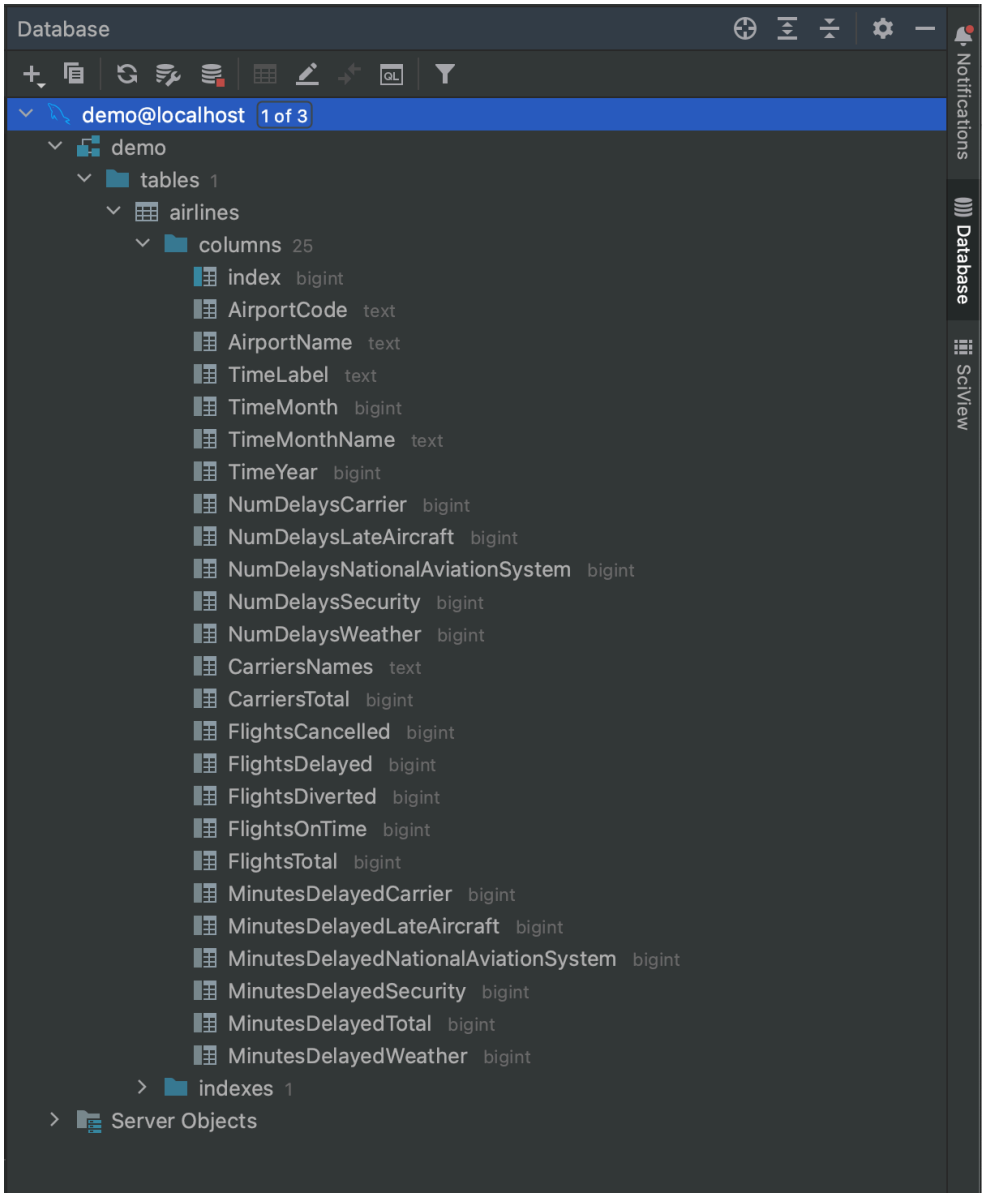
Remarque DVP : dans MySQL (contrairement aux autres SGBD) , le schéma et la database sont une seule et même notion.

Copyright ® 2023 Jodie Burchell. Aucune reproduction, même partielle, ne peut être faite de ce site et de l'ensemble de son contenu : textes, documents, images, etc. sans l'autorisation expresse de l'auteur. Sinon vous encourez selon la loi jusqu'à trois ans de prison et jusqu'à 300 000 € de dommages et intérêts.



Ceci fait, nous sommes prêts à nous connecter à notre base de données. Cliquez sur *OK* et attendez quelques secondes. Vous verrez alors que l'introspection a été effectuée pour l'intégralité de notre base de données, jusqu'au niveau des colonnes des tables et de leur type. Nous disposons ainsi d'une vue d'ensemble de ce qui se trouve dans la base de données avant d'exécuter une requête.

Copyright ® 2023 Jodie Burchell. Aucune reproduction, même partielle, ne peut être faite de ce site et de l'ensemble de son contenu : textes, documents, images, etc. sans l'autorisation expresse de l'auteur. Sinon vous encourez selon la loi jusqu'à trois ans de prison et jusqu'à 300 000 € de dommages et intérêts.



II - Lire les données avec MySQL Connector

Maintenant que nous savons ce que contient notre base de données, nous sommes prêts à effectuer une requête. Disons que nous voulons voir quels aéroports ont eu au moins 500 vols retardés en 2016. En examinant les colonnes de la table *airlines* introspectée, nous voyons que ces données peuvent être obtenues avec la requête suivante :

```
1. SELECT AirportCode,
2.        SUM(FlightsDelayed) AS TotalDelayed
3.    FROM airlines
4.    WHERE TimeYear = 2016
5.    GROUP BY AirportCode
6.    HAVING SUM(FlightsDelayed) > 500;
```

La première façon dont nous pouvons exécuter cette requête avec Python consiste à utiliser un paquet appelé MySQL Connector, qui peut être installé depuis **PyPI** ou **Anaconda**. Consultez la documentation correspondante si vous avez besoin de conseils pour configurer les environnements **pip** ou **conda** ou pour **installer des dépendances**. Une fois l'installation terminée, nous ouvrons un **nouveau notebook Jupyter** et importons le connecteur MySQL et les bibliothèques pandas.

Copyright ® 2023 Jodie Burchell. Aucune reproduction, même partielle, ne peut être faite de ce site et de l'ensemble de son contenu : textes, documents, images, etc. sans l'autorisation expresse de l'auteur. Sinon vous encourez selon la loi jusqu'à trois ans de prison et jusqu'à 300 000 € de dommages et intérêts.

```
import mysql.connector
import pandas as pd
```

Pour lire les données de notre base, nous devons créer un **connector**. Pour ce faire, nous utilisons la méthode connect, à laquelle nous transmettons les informations d'identification nécessaires pour accéder à la base de données : *host*, *database*, *user* et *password*. Ce sont les mêmes informations d'identification que celles que nous avons utilisées précédemment pour accéder à la base de données avec la fenêtre d'outils *Database*.

```
1. mysql_db_connector = mysql.connector.connect(
2.     host="localhost",
3.     database="demo",
4.     user="pycharm",
5.     password="pycharm"
6. )
```

Nous devons maintenant créer un **curseur**. Il servira pour l'exécution de nos requêtes SQL dans la base de données et utilisera les informations d'identification spécifiées dans notre connecteur pour obtenir l'accès.

```
mysql_db_cursor = mysql_db_connector.cursor()
```

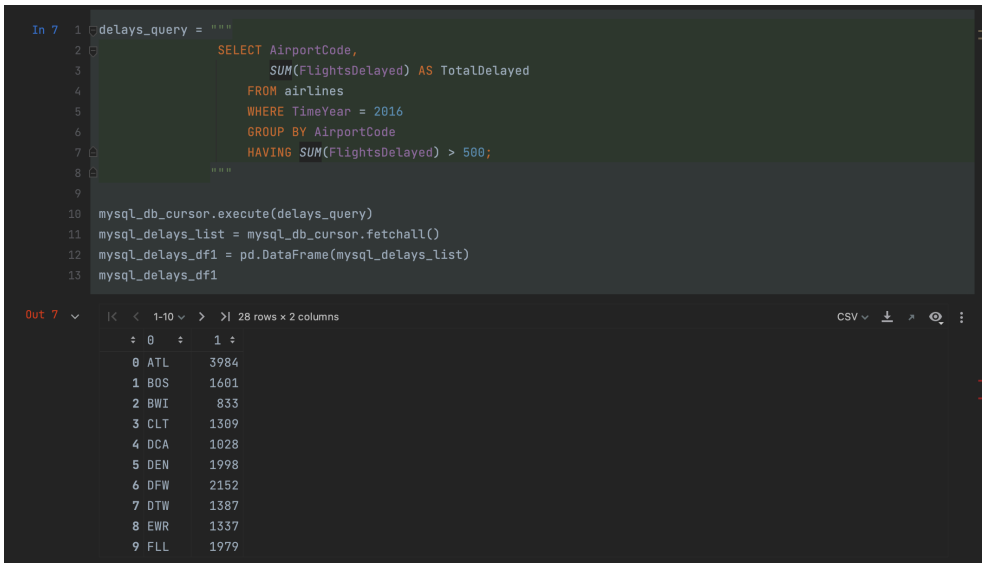
Nous sommes maintenant prêts à exécuter notre requête. Pour ce faire, nous utilisons la **méthode** execute du curseur en transmettant la requête comme argument.

```
1. delays_query = """
2.                SELECT AirportCode,
3.                       SUM(FlightsDelayed) AS TotalDelayed
4.                    FROM airlines
5.                    WHERE TimeYear = 2016
6.                    GROUP BY AirportCode
7.                    HAVING SUM(FlightsDelayed) > 500;
8.                """
9. mysql_db_cursor.execute(delays_query)
```

Nous récupérons ensuite le résultat en utilisant la méthode fetchall du curseur.

```
mysql_delays_list = mysql_db_cursor.fetchall()
```

Cependant, un problème survient à ce stade : fetchall restitue les données sous forme de liste. Pour lire le résultat dans pandas, nous pouvons le passer dans un *DataFrame*, mais les noms de nos colonnes seront perdus et il faudra les spécifier manuellement lors de la création du *DataFrame*.

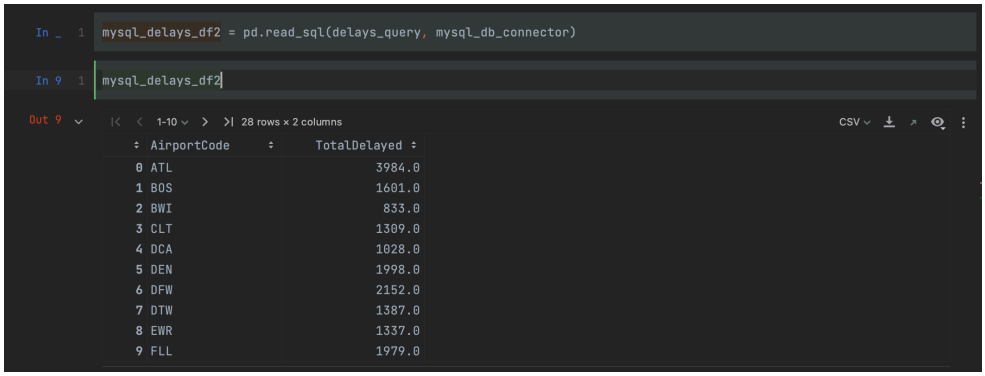


Copyright ® 2023 Jodie Burchell. Aucune reproduction, même partielle, ne peut être faite de ce site et de l'ensemble de son contenu : textes, documents, images, etc. sans l'autorisation expresse de l'auteur. Sinon vous encourez selon la loi jusqu'à trois ans de prison et jusqu'à 300 000 € de dommages et intérêts.

Heureusement, pandas offre une meilleure solution. Plutôt que de créer un curseur, nous pouvons lire notre requête dans un DataFrame en une seule étape, en utilisant **la méthode** read_sql.

```
mysql_delays_df2 = pd.read_sql(delays_query, con=mysql_db_connector)
```

Il nous suffit de passer notre requête et notre connecteur en arguments pour lire les données de la base de données MySQL. En examinant notre *dataframe*, nous pouvons voir que nous avons exactement les mêmes résultats que ci-dessus, mais que les noms de nos colonnes ont été préservés cette fois-ci.



Vous avez peut-être remarqué que PyCharm applique la mise en évidence de la syntaxe à la requête SQL, même lorsqu'elle est contenue dans une chaîne Python. Nous verrons une autre façon de travailler avec SQL avec l'aide de PyCharm plus loin dans cet article.

III - Lire les données avec SQLAlchemy

Il est également possible d'utiliser un paquet appelé **SQLAlchemy** à la place de MySQL Connector. Ce paquet offre une méthode unique pour se connecter à différentes bases de données, parmi lesquelles MySQL. L'un des avantages de SQLAlchemy est que la syntaxe des requêtes reste similaire d'un type de base de données à l'autre, ce qui évite d'avoir à se souvenir de toutes les commandes multiples si vous travaillez avec un grand nombre de bases de données différentes.

Pour commencer, nous devons installer SQLAlchemy depuis **PyPI** ou **Anaconda**. Nous importons ensuite la méthode create_engine, et bien sûr, les pandas.

```
1. import pandas as pd
2. from sqlalchemy import create_engine
```

Nous devons maintenant créer notre **moteur**. Le moteur permet d'indiquer à pandas quel dialecte SQL nous utilisons (dans notre cas, MySQL) et de lui fournir les informations d'identification dont il a besoin pour accéder à notre base de données. Tout ceci est passé en tant que chaîne, sous la forme : *[dialect]://[user]:[password]@[host]/[database]*.

Voyons à quoi cela ressemble avec notre base de données MySQL :

```
mysql_engine = create_engine("mysql+mysqlconnector://pycharm:pycharm@localhost/demo")
```

Une fois le moteur créé, il nous suffit d'utiliser à nouveau read_sql, en transmettant cette fois le moteur à l'argument con :

```
mysql_delays_df3 = pd.read_sql(delays_query, con=mysql_engine)
```

Comme vous pouvez le voir, nous obtenons le même résultat qu'en utilisant read_sql avec MySQL Connector.

Copyright ® 2023 Jodie Burchell. Aucune reproduction, même partielle, ne peut être faite de ce site et de l'ensemble de son contenu : textes, documents, images, etc. sans l'autorisation expresse de l'auteur. Sinon vous encourez selon la loi jusqu'à trois ans de prison et jusqu'à 300 000 € de dommages et intérêts.

```
In 12
mysql_delays_df3 = pd.read_sql(delays_query, con=mysql_engine)
mysql_delays_df3
```

Out 12 — 1-10 — 28 rows × 2 columns — CSV

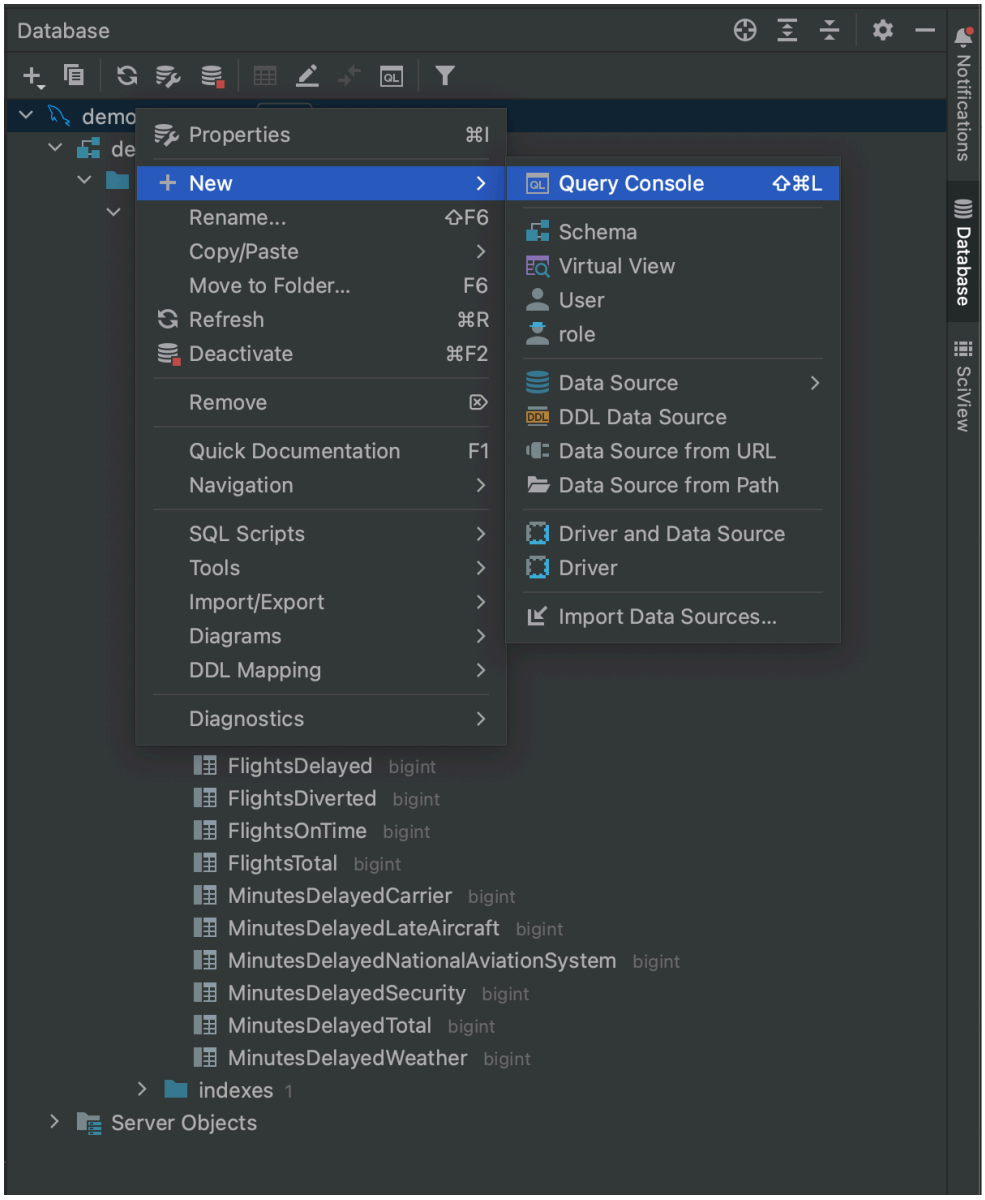
	AirportCode	TotalDelayed
0	ATL	3984.0
1	BOS	1601.0
2	BWI	833.0
3	CLT	1309.0
4	DCA	1028.0
5	DEN	1998.0
6	DFW	2152.0
7	DTW	1387.0
8	EWR	1337.0
9	FLL	1979.0

IV - Options avancées pour travailler avec les bases de données

Ces méthodes de connexion sont très efficaces pour extraire une requête que nous connaissons déjà, mais qu'en est-il si nous voulons avoir un aperçu de ce à quoi ressembleront nos données avant d'exécuter la requête complète ou obtenir une estimation du temps que prendra la requête ? Là encore, PyCharm fournit plusieurs fonctionnalités avancées qui peuvent nous aider.

En revenant à la fenêtre d'outils Database, si l'on fait un clic doit sur la base de données, on peut voir une option située sous *New* qui permet de créer une *Query Console*.

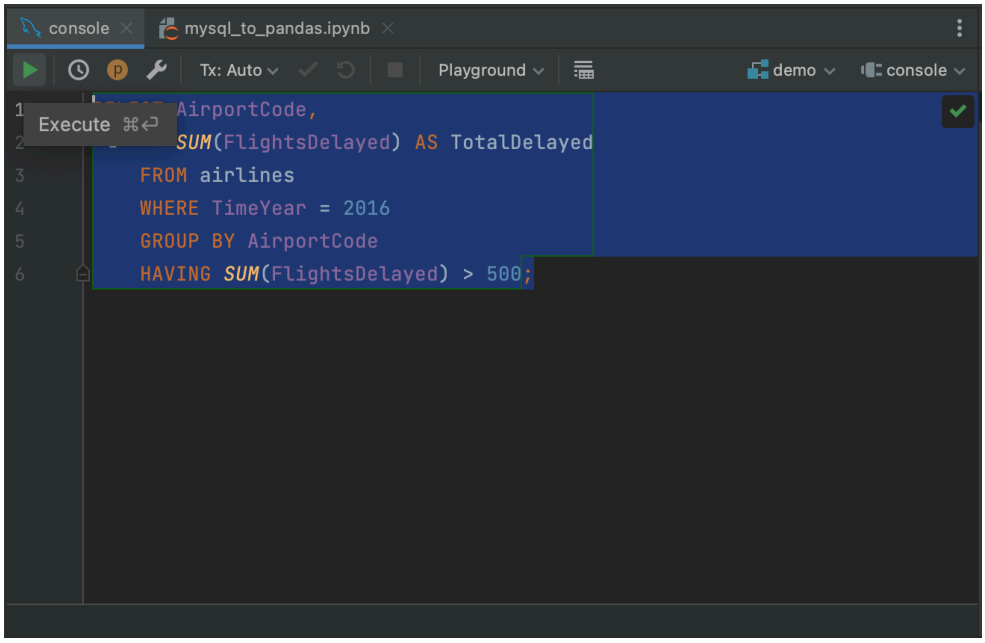
Copyright ® 2023 Jodie Burchell. Aucune reproduction, même partielle, ne peut être faite de ce site et de l'ensemble de son contenu : textes, documents, images, etc. sans l'autorisation expresse de l'auteur. Sinon vous encourez selon la loi jusqu'à trois ans de prison et jusqu'à 300 000 € de dommages et intérêts.



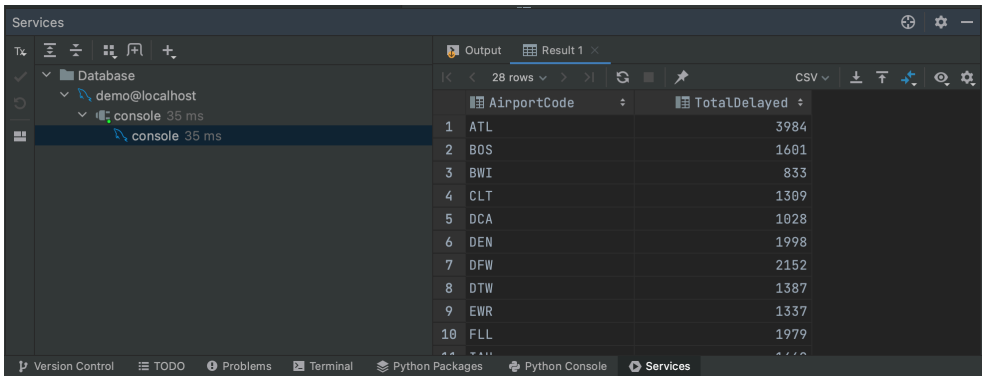
Nous pouvons alors ouvrir une console et l'utiliser pour interroger la base de données en SQL natif. La fenêtre de la console fournit la saisie semi-automatique du code SQL et l'introspection, ce qui facilite la création de vos requêtes avant de les transmettre aux paquets de connecteurs en Python.

Mettez votre requête en évidence et cliquez sur le bouton *Execute* dans le coin supérieur gauche.

Copyright ® 2023 Jodie Burchell. Aucune reproduction, même partielle, ne peut être faite de ce site et de l'ensemble de son contenu : textes, documents, images, etc. sans l'autorisation expresse de l'auteur. Sinon vous encourez selon la loi jusqu'à trois ans de prison et jusqu'à 300 000 € de dommages et intérêts.

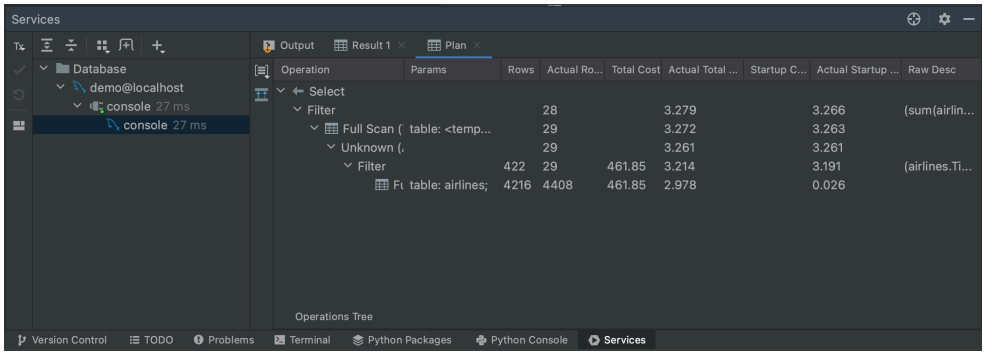


Cela permettra de retrouver les résultats de notre requête dans l'onglet *Services*, où ils pourront être **inspectés ou exportés**. L'un des avantages de l'exécution de requêtes dans la console est que seules les 500 premières lignes sont initialement extraites de la base de données, ce qui permet de se faire une idée des résultats de requêtes plus importantes sans avoir à extraire toutes les données. Vous pouvez ajuster le nombre de lignes récupérées dans *Settings/Preferences* | *Tools* | *Database* | *Data Editor and Viewer*, en modifiant la valeur sous Limit page size to:.

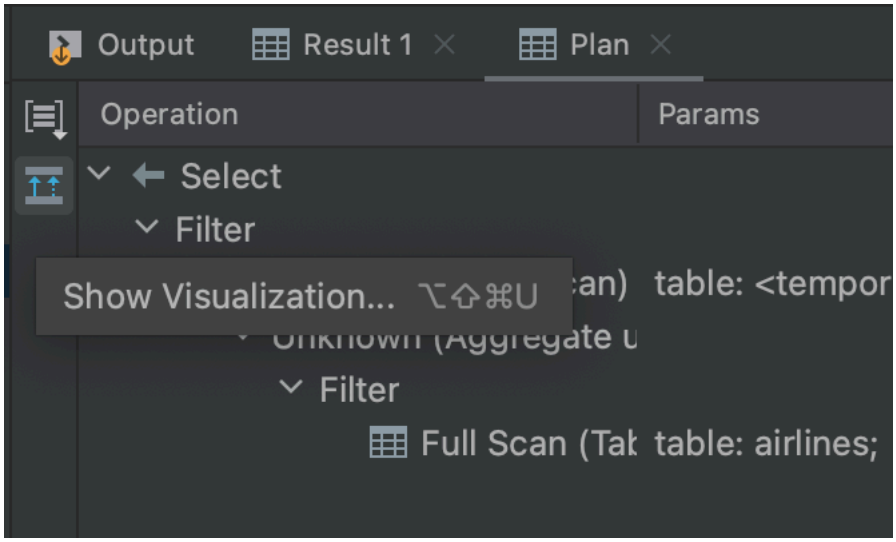


Nous pouvons également obtenir une estimation du temps que prendra notre requête en générant un **plan d'exécution**. En mettant à nouveau notre requête en surbrillance, puis en effectuant un clic droit, nous pouvons sélectionner *Explain Plan* | *Explain Analyse* dans le menu. Cela va générer un plan d'exécution pour notre requête, montrant chaque étape que le planificateur de requêtes effectue pour récupérer nos résultats. Les plans d'exécution constituent un sujet à part entière, mais en l'occurrence, il n'est pas vraiment nécessaire de comprendre tout ce que notre plan indique. Dans notre cas, la colonne la plus pertinente est *Actual Total Time*, car elle nous permet de voir combien de temps il faudra pour renvoyer toutes les lignes à chaque étape. Cela nous donne une bonne estimation de la durée totale de la requête et nous permet de savoir si certaines parties de notre requête sont susceptibles de prendre beaucoup de temps.

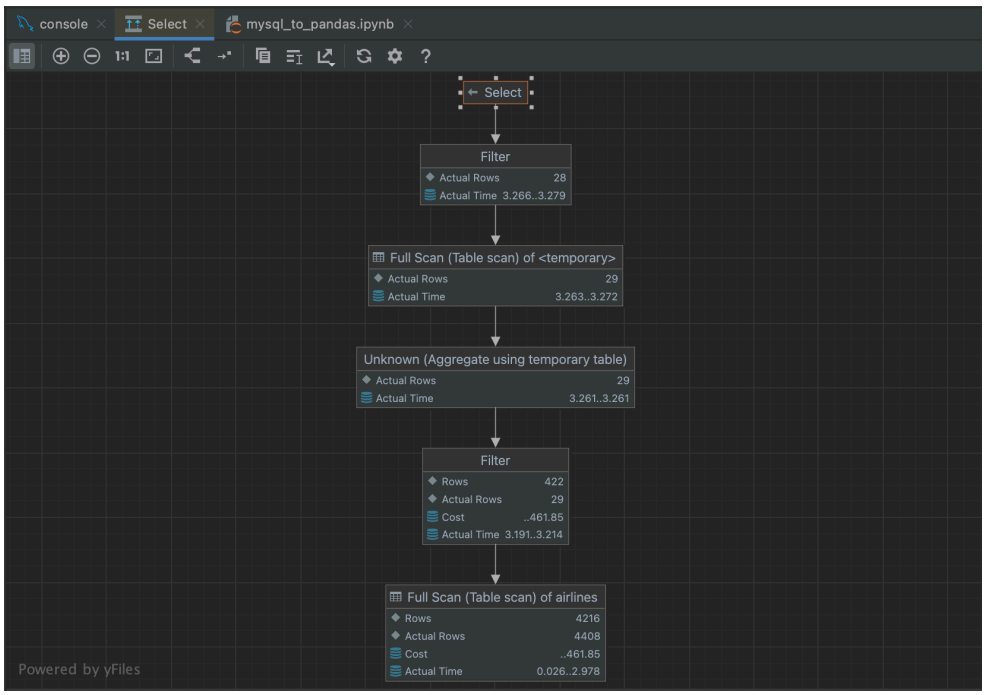
Copyright ® 2023 Jodie Burchell. Aucune reproduction, même partielle, ne peut être faite de ce site et de l'ensemble de son contenu : textes, documents, images, etc. sans l'autorisation expresse de l'auteur. Sinon vous encourez selon la loi jusqu'à trois ans de prison et jusqu'à 300 000 € de dommages et intérêts.



Vous pouvez également visualiser l'exécution en cliquant sur le bouton *Show Visualization* à gauche du panneau *Plan*.



Cela permet d'obtenir un organigramme qui facilite la navigation à travers les étapes du planificateur de requêtes.



Copyright ® 2023 Jodie Burchell. Aucune reproduction, même partielle, ne peut être faite de ce site et de l'ensemble de son contenu : textes, documents, images, etc. sans l'autorisation expresse de l'auteur. Sinon vous encourez selon la loi jusqu'à trois ans de prison et jusqu'à 300 000 € de dommages et intérêts.

Importer des données de MySQL dans des DataFrames pandas est assez simple et PyCharm dispose de nombreux outils qui facilitent le travail avec les bases de données MySQL. Dans le prochain article de cette série, nous verrons comment utiliser PyCharm pour lire des données dans pandas à partir de bases de données PostgreSQL.

V - Remerciements Developpez.com

Nous tenons à remercier **Malick** pour la mise au gabarit et **Escartefigue** pour la relecture orthographique.

Copyright ® 2023 Jodie Burchell. Aucune reproduction, même partielle, ne peut être faite de ce site et de l'ensemble de son contenu : textes, documents, images, etc. sans l'autorisation expresse de l'auteur. Sinon vous encourez selon la loi jusqu'à trois ans de prison et jusqu'à 300 000 € de dommages et intérêts.