Travaux pratiques : Vérification de la qualité des données pour un entrepôt de données

Temps estimé nécessaire : 30 minutes

Objectif du laboratoire :

L'objectif principal de ce laboratoire est d'instruire les participants sur le processus de réalisation de contrôles approfondis de la qualité des données dans un environnement d'entreposage de données. Il se concentre sur l'utilisation d'un framework basé sur Python dans une base de données PostgreSQL pour valider l'intégrité des données. Les principaux domaines d'intérêt comprennent l'identification des valeurs nulles, des doublons et des entrées non valides, ainsi que la vérification des plages de données. Le laboratoire vise à doter les apprenants des compétences nécessaires pour mettre en place et utiliser un cadre de test pour la validation des données, garantissant ainsi l'exactitude et la cohérence des données.

Avantages de l'apprentissage en laboratoire :

Participer à ce laboratoire présente plusieurs avantages, notamment en améliorant ses capacités en matière de gestion des données et d'assurance qualité. Les apprenants acquerront une expérience pratique de la mise en œuvre de contrôles automatisés de la qualité des données, une compétence essentielle pour maintenir la fiabilité des données dans les applications du monde réel. Cette compétence est particulièrement bénéfique pour les professionnels travaillant avec de grands ensembles de données, car elle garantit l'intégrité des données utilisées pour l'analyse et la prise de décision. De plus, la compréhension de ces concepts est essentielle pour quiconque aspire à se spécialiser dans la science des données, l'administration de bases de données ou tout autre domaine qui repose fortement sur des données précises et fiables.

Objectifs

Dans ce laboratoire, vous allez :

- Vérifier les valeurs nulles
- Vérifier les valeurs en double
- Vérifier le minimum et le maximum
- Vérifier les valeurs non valides
- Générer un rapport sur la qualité des données

À propos de Skills Network Cloud IDE

L'IDE Cloud de Skills Network (basé sur Theia et Docker) fournit un environnement pour les travaux pratiques liés aux cours et aux projets. Theia est un IDE open source (environnement de développement intégré), qui peut être exécuté sur un ordinateur de bureau ou sur le cloud. Pour réaliser ce laboratoire, nous utiliserons l'IDE Cloud basé sur Theia exécuté dans un conteneur Docker.

Avis important concernant cet environnement de laboratoire

Veuillez noter que les sessions de cet environnement de laboratoire ne sont pas permanentes. Un nouvel environnement est créé pour vous à chaque fois que vous vous connectez à ce laboratoire. Toutes les données que vous avez pu enregistrer lors d'une session précédente seront perdues. Pour éviter de perdre vos données, prévoyez de terminer ces laboratoires en une seule session.

Exercice 1 – Préparer l'environnement

Dans cet exercice, vous préparerez l'environnement afin que nous puissions effectuer les contrôles de qualité des données.

Étape 1 : Démarrez le serveur postgresql.

1. Cliquez sur **la boîte à outils du Réseau de compétences** .

2. Sélectionnez la base de données **PostGres** .

3. Cliquez sur le bouton **Démarrer** .

Étape 2 : Téléchargez le script de configuration de la zone de préparation.

Ouvrez un nouveau terminal en cliquant sur la barre de menu et en sélectionnant **Terminal** -> **Nouveau terminal** , comme indiqué dans l'image ci-dessous.

Cela ouvrira un nouveau terminal en bas de l'écran.

Exécutez la commande ci-dessous pour télécharger le script de configuration de la zone de préparation.

```
wget https://cf-courses-data.s3.us.cloud-object-storage.appdomain.cloud/8ZUkKar_boDbhNgMiwGAWg/setupstagingarea.sh
```

Étape 3 : Exportez votre mot de passe postgres.

Exécutez la commande ci-dessous dans le terminal en remplaçant <votre mot de passe postgres> par votre mot de passe postgres qui se trouve sous l'onglet Informations de connexion.

export PGPASSWORD=<votre mot de passe postgres>

Étape 4 : exécutez le script d'installation.

Exécutez la commande ci-dessous pour exécuter le script de configuration de la zone de préparation.

```
bash setupstagingarea.sh
```

Lorsque vous voyez un message, `Successfully setup the staging area`vous êtes prêt à effectuer des contrôles de qualité des données.

Exercice 2 – Préparation du cadre de test

Vous pouvez effectuer la plupart des contrôles de qualité des données en exécutant manuellement des requêtes SQL sur l'entrepôt de données.

Il est judicieux d'automatiser ces contrôles à l'aide de programmes ou d'outils personnalisés. L'automatisation vous aide à effectuer facilement

- créer de nouveaux tests,
- exécuter des tests,
- et planifier des tests.

Nous utiliserons un framework basé sur Python pour exécuter les tests de qualité des données.

Étape 1 : Téléchargez le framework.

Exécutez les commandes ci-dessous pour télécharger le framework

```
wget https://cf-courses-data.s3.us.cloud-object-storage.appdomain.cloud/IBM-DB0260EN-SkillsNetwork/labs/Verifying%20Data%20Quality%2
wget https://cf-courses-data.s3.us.cloud-object-storage.appdomain.cloud/HB0XK4MDrGwigMmVPmPoeQ/dbconnect.py
wget https://cf-courses-data.s3.us.cloud-object-storage.appdomain.cloud/IBM-DB0260EN-SkillsNetwork/labs/Verifying%20Data%20Quality%2
wget https://cf-courses-data.s3.us.cloud-object-storage.appdomain.cloud/saTxV8y9Kt-e8Zxe29M0TA/generate-data-quality-report.py
ls
```

Étape 2 : installez le pilote Python pour Postgresql.

Exécutez la commande ci-dessous pour installer le pilote Python pour la base de données Postgresql

```
python3 -m pip install psycopg2
```

Étape 3 : mettre à jour le mot de passe dans dbconnect.py

Allez dans Fichier > Ouvrir. Une fenêtre apparaîtra ; double-cliquez sur dbconnect.py pour l'ouvrir.

Une fois le fichier ouvert, à la ligne 3, remplacez **<remplacez ceci par votre mot de passe postgres>** par votre **POSTGRES_PASSWORD** qui se trouve sous l' onglet **Informations de connexion** lorsque vous faites défiler vers le bas.

Une fois terminé, enregistrez le fichier en sélectionnant **Fichier>Enregistrer** .

Répétez l'étape 3 ci-dessus pour le fichier generate-data-quality-report.py. Mettez à jour votre mot de passe à la ligne 15 du script et enregistrez-le.

Étape 4 : tester la connectivité de la base de données.

Maintenant, nous devons vérifier

- si le pilote Python Postgresql est correctement installé.
- si le serveur Postgresql est opérationnel.
- si notre micro framework peut se connecter à la base de données.

La commande ci-dessous permet de vérifier tous les cas ci-dessus.

```
python3 dbconnect.py
```

Si tout se passe bien, vous devriez recevoir un message `Successfully connected to database`.

La commande se déconnecte également du serveur avec un message `Connection closed`.

Exercice 3 – Créer un exemple de rapport sur la qualité des données

Exécutez la commande ci-dessous pour installer pandas.

```
python3 -m pip install pandas tabulate
```

Exécutez la commande ci-dessous pour générer un exemple de rapport de qualité des données.

```
python3 generate-data-quality-report.py
```

Vous devriez voir une liste des tests qui ont été exécutés et leur statut.

Exercice 4 – Explorer les tests de qualité des données

Ouvrez le fichier `mytests.py`dans l'éditeur en suivant les étapes ci-dessous.

Vous devriez maintenant voir le fichier ouvert dans l'éditeur.

Le fichier `mytests.py`contient tous les tests de qualité des données.

Il fournit un moyen rapide et simple de créer et d'exécuter de nouveaux tests de qualité des données.

Le cadre de test fournit les tests suivants :

- check_for_nulls - ce test recherchera les valeurs nulles dans une colonne
- check_for_min_max - ce test vérifiera si les valeurs d'une colonne correspondent à une plage de valeurs min et max
- check_for_valid_values - ce test recherchera toutes les valeurs non valides dans une colonne
- check_for_duplicates - ce test recherchera les doublons dans une colonne

Chaque test peut être rédigé en mentionnant un minimum de 4 paramètres.

- testname - Le nom lisible par l'homme du test à des fins de création de rapports
- test - Le nom de test réel fourni par le micro-cadre de test
- table - Le nom de la table sur laquelle le test doit être effectué
- colonne - Le nom de la table sur laquelle le test doit être effectué

Exercice 5 – Vérification des valeurs nulles

Voyons maintenant à quoi `check_for_nulls`ressemble un test.

Voici un exemple `check_for_nulls`de test :

```
test1={
    "testname":"Check for nulls",
    "test":check_for_nulls,
    "column": "monthid",
    "table": "DimMonth"
}
```

Tous les tests doivent être nommés comme `test`suit par un numéro unique pour identifier le test.

- Donnez une description facile à comprendre pour`testname`
- mentionner `check_for_nulls`pour`test`
- mentionnez le nom de la colonne sur laquelle vous souhaitez vérifier les valeurs nulles
- mentionner le nom de la table où cette colonne existe

Créons maintenant un nouveau `check_for_nulls`test et exécutons-le.

Le test ci-dessous vérifie s'il y a des valeurs nulles dans la colonne `year`du tableau `DimMonth`.

Le test échoue si des valeurs nulles existent.

Copiez et collez le code ci-dessous à la fin du fichier mytests.py.

```
test5={
    "testname":"Check for nulls",
    "test":check_for_nulls,
    "column": "year",
    "table": "DimMonth"
}
```

Enregistrez le fichier en utilisant Menu -> Fichier -> Enregistrer

Exécutez la commande ci-dessous pour générer le nouveau rapport de qualité des données.

```
python3 generate-data-quality-report.py
```

Exercice 6 – Vérification de la plage min max

Voyons maintenant à quoi `check_for_min_max`ressemble un test.

Voici un exemple `check_for_min_max`de test

```
test2={
    "testname":"Check for min and max",
    "test":check_for_min_max,
    "column": "monthid",
    "table": "DimMonth",
    "minimum":1,
    "maximum":12
}
```

En plus des champs habituels, vous disposez ici de deux champs supplémentaires.

- minimum est la valeur valide la plus basse pour cette colonne. (Exemple 1 dans le cas du numéro de mois)
- maximum est la valeur valide la plus élevée pour cette colonne. (Exemple 12 dans le cas du numéro de mois)

Créons maintenant un nouveau `check_for_min_max`test et exécutons-le.

Le test ci-dessous vérifie le minimum de 1 et le maximum de 4 dans la colonne `quarter`du tableau `DimMonth`.

Le test échoue s'il y a des valeurs inférieures au minimum ou supérieures au maximum.

Copiez et collez le code ci-dessous à la fin du fichier mytests.py.

```
test6={
    "testname":"Check for min and max",
    "test":check_for_min_max,
    "column": "quarter",
    "table": "DimMonth",
    "minimum":1,
    "maximum":4
}
```

Enregistrez le fichier en utilisant Menu -> Fichier -> Enregistrer

Exécutez la commande ci-dessous pour générer le nouveau rapport de qualité des données.

```
python3 generate-data-quality-report.py
```

Exercice 7 – Vérifiez les entrées non valides

Voyons maintenant à quoi `check_for_valid_values`ressemble un test.

Voici un exemple `check_for_valid_values`de test :

```
test3={
    "testname":"Check for valid values",
    "test":check_for_valid_values,
    "column": "category",
    "table": "DimCustomer",
    "valid_values":{'Individual','Company'}
}
```

En plus des champs habituels, vous disposez ici d'un champ supplémentaire.

- utilisez le champ `valid_values`pour mentionner quelles sont les valeurs valides pour cette colonne.

Créons maintenant un nouveau `check_for_valid_values`test et exécutons-le.

Le test ci-dessous vérifie les valeurs valides dans la colonne `quartername`du tableau `DimMonth`.

Les valeurs valides sont Q1, Q2, Q3, Q4

Le test échoue s'il y a des valeurs inférieures au minimum ou supérieures au maximum.

Copiez et collez le code ci-dessous à la fin du fichier mytests.py.

```
test7={
    "testname":"Check for valid values",
    "test":check_for_valid_values,
    "column": "quartername",
    "table": "DimMonth",
    "valid_values":{'Q1','Q2','Q3','Q4'}
}
```

Enregistrez le fichier en utilisant Menu -> Fichier -> Enregistrer

Exécutez la commande ci-dessous pour générer le nouveau rapport de qualité des données.

```
python3 generate-data-quality-report.py
```

Exercice 8 – Rechercher les doublons

Voyons maintenant à quoi `check_for_duplicates`ressemble un test.

Voici un exemple `check_for_duplicates`de test

```
test4={
    "testname":"Check for duplicates",
    "test":check_for_duplicates,
    "column": "monthid",
    "table": "DimMonth"
}
```

Créons maintenant un nouveau `check_for_duplicates`test et exécutons-le.

Le test ci-dessous vérifie les valeurs en double dans la colonne `customerid`du tableau `DimCustomer`.

Le test échoue s'il existe des doublons.

Copiez et collez le code ci-dessous à la fin du fichier mytests.py.

```
test8={
    "testname":"Check for duplicates",
    "test":check_for_duplicates,
    "column": "customerid",
    "table": "DimCustomer"
}
```

Enregistrez le fichier en utilisant Menu -> Fichier -> Enregistrer

Exécutez la commande ci-dessous pour générer le nouveau rapport de qualité des données.

```
python3 generate-data-quality-report.py
```

Exercices pratiques

1. Problème:

   Créer un `check_for_nulls`test sur une colonne `billedamount`du tableau`FactBilling`

▶ Cliquez ici pour un indice
▼ Cliquez ici pour la solution

Copiez et collez le code ci-dessous à la fin du fichier mytests.py.

```
test9={
    "testname":"Check for nulls",
    "test":check_for_nulls,
    "column": "billedamount",
    "table": "FactBilling"
}
```

2. Problème:

   Créer un `check_for_duplicates`test sur une colonne `billid`du tableau`FactBilling`

▶ Cliquez ici pour un indice
▼ Cliquez ici pour la solution

Copiez et collez le code ci-dessous à la fin du fichier mytests.py.

```
test10={
    "testname":"Check for duplicates",
    "test":check_for_duplicates,
    "column": "billid",
    "table": "FactBilling"
}
```

3. Problème:

   Créer un `check_for_valid_values`test sur une colonne `quarter`du tableau `DimMonth`. Les valeurs valides sont 1, 2, 3, 4

▶ Cliquez ici pour un indice
▶ Cliquez ici pour la solution

Félicitations !! Vous avez terminé avec succès ce laboratoire.

Auteurs

Ramesh Sannareddy

Autres contributeurs

Rav Ahuja