Projet pratique : Introduction à l'entreposage de données

Durée estimée nécessaire : 90 minutes

Ce laboratoire pratique complet est conçu pour fournir une expérience pratique de la conception et de la mise en œuvre d'un entrepôt de données. Vous avez été embauché en tant qu'ingénieur de données pour une entreprise de vente au détail d'électronique grand public et l'entreprise souhaite que vous créiez et mettiez en œuvre un entrepôt de données pour analyser ses performances de vente et sa gestion des stocks.

Ce que vous apprendrez :

Le laboratoire offre divers avantages, notamment pour ceux qui cherchent à améliorer leurs compétences en ingénierie des données et en intelligence d'affaires.

Expérience pratique dans la conception d'entrepôts de données : le laboratoire offre une expérience pratique dans la conception et la mise en œuvre d'un schéma en étoile, ce qui est crucial pour tout projet d'entreposage de données.

Compétences en rédaction de requêtes SQL : il améliore votre capacité à écrire des requêtes SQL complexes, y compris des ensembles de regroupement, des cumuls et des cubes, essentiels pour l'analyse et la création de rapports de données.

Chargement et transformation des données : Le laboratoire propose une pratique du chargement et de la transformation des données, une compétence essentielle pour la gestion des entrepôts de données.

Applications de scénarios du monde réel : L’approche basée sur des scénarios du laboratoire garantit que les compétences acquises sont pertinentes et applicables aux projets réels d’entreposage de données et de veille économique.

Évolution de carrière : Ces compétences sont très demandées dans les domaines de l’ingénierie des données, de la veille stratégique et de l’analyse, contribuant de manière significative à la croissance et aux opportunités professionnelles.

En un mot, ce laboratoire sert de guide complet pour quiconque souhaite renforcer son expertise en matière d'entreposage de données et de veille économique, en fournissant des compétences pratiques directement applicables dans les environnements professionnels.

À propos de SN Labs Cloud IDE

L'IDE Cloud de Skills Network Labs fournit un environnement pratique pour les laboratoires et utilise Theia, une plate-forme d'environnement de développement intégré (IDE) open source qui peut s'exécuter sur un ordinateur de bureau ou sur le cloud. Pour réaliser ce laboratoire, vous utiliserez l'IDE Cloud basé sur Theia et PostgreSQL.

Exercice en séance unique

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 enregistrées lors d'une session précédente peuvent être perdues. Pour éviter de perdre vos données, prévoyez de terminer ces laboratoires en une seule session.

Logiciel utilisé en laboratoire

Dans ce laboratoire, vous utiliserez la base de données PostgreSQL. PostgreSQL est un système de gestion de base de données relationnelle (SGBDR) conçu pour stocker, manipuler et récupérer des données de manière efficace.

Scénario

Vous êtes un ingénieur de données embauché par une entreprise de vente au détail d'électronique grand public. L'entreprise vend divers produits électroniques via ses canaux en ligne et hors ligne dans les principales villes des États-Unis. Elle exploite plusieurs magasins et entrepôts pour gérer son inventaire et ses opérations de vente. L'entreprise souhaite créer un entrepôt de données pour analyser ses performances de vente et sa gestion des stocks et vise à générer des rapports, tels que :

- Chiffre d'affaires total par an et par ville
- Chiffre d'affaires total par mois et par ville
- Chiffre d'affaires total par trimestre et par ville
- Chiffre d'affaires total par an par catégorie de produits
- Chiffre d'affaires total par catégorie de produits et par ville
- Chiffre d'affaires total par catégorie de produits et par magasin

Objectifs d'apprentissage

Après avoir terminé ce laboratoire, vous serez capable de :

- Développer des tables de dimensions et de faits pour organiser et structurer efficacement les données à des fins d'analyse
- Utiliser des requêtes SQL pour créer et charger des données dans des tables de dimensions et de faits
- Créez des vues matérialisées pour optimiser les performances des requêtes

Remarque : captures d'écran

Tout au long de ce laboratoire, vous serez invité à prendre des captures d'écran et à les enregistrer sur votre propre appareil. Vous pouvez utiliser divers outils de capture d'écran gratuits ou les touches de raccourci de votre système d'exploitation (Alt + Impr écran sous Windows, Commande + Maj + 5 sur Mac, Maj + Ctrl + Afficher les fenêtres sur Chromebook) pour capturer les captures d'écran requises. Les captures d'écran peuvent être au format jpg ou png.

À propos de l'ensemble de données

L'ensemble de données que vous utiliserez dans ce projet n'est pas un ensemble de données réel. Il a été créé par programmation pour les besoins de ce projet.

Prérequis

Vous devez utiliser la base de données PostgreSQL pour terminer ce laboratoire.

Vous démarrerez votre projet en concevant un entrepôt de schéma en étoile en identifiant les colonnes des différentes dimensions et tables de faits du schéma.

Exercice 1 : Concevoir un entrepôt de données

ID de vente	Type de produit	Prix unitaire	Quantité vendue	Ville	Date
001	Électronique	299,99 $	30	New York	01/04/2024
002	Vêtements	49,99 $	50	Los Angeles	01/04/2024
003	Meubles	399,99 $	10	Chicago	02/04/2024
004	Électronique	199,99 $	20	Houston	02/04/2024
005	Épiceries	2,99 $	100	Miami	03/04/2024

Tâche 1 : Concevoir la table de dimension MyDimDateTâche 1 - Concevoir la table de dimension MyDimDate

Notez les champs de la table MyDimDate dans n'importe quel éditeur de texte, un champ par ligne.
L'entreprise recherche une granularité de jour, ce qui signifie qu'elle aimerait avoir la possibilité de générer le rapport sur une base annuelle, mensuelle, quotidienne et hebdomadaire.
Voici une liste partielle de champs à titre d'exemple :

dateid
mois
nom du mois
…

Solution

Champs dans la table MyDimDate :

- identifiant de la date
- année
- mois
- nom du mois
- jour
- jour de la semaine
- nom du jour de la semaine

La table aura un identifiant unique (dateid) pour chaque entrée de date. D'autres champs tels que l'année, le mois, le nom du mois, le jour, le jour de la semaine et le nom du jour de la semaine fourniront des informations détaillées sur chaque date, permettant des options de rapport flexibles basées sur différents intervalles de temps.

Tâche 2 : Concevoir la table de dimensions MyDimProduct

Notez les champs de la table MyDimProduct dans n'importe quel éditeur de texte, un champ par ligne.

Solution:

Dans cette tâche, vous allez concevoir la table de dimensions MyDimProduct pour stocker des informations relatives aux produits.

Champs dans la table MyDimProduct :

- ID produit
- nom du produit

Le tableau contiendra un identifiant unique (productid) pour chaque entrée de produit. Le champ productname stockera le nom ou la description du produit. Ce tableau facilitera l'analyse et la création de rapports en fonction des différents produits vendus.

Tâche 3 : Concevoir la table de dimension MyDimCustomerSegment

Notez les champs de la table MyDimCustomerSegment dans n'importe quel éditeur de texte, un champ par ligne.

Solution:

Dans cette tâche, vous allez concevoir la table de dimension MyDimCustomerSegment pour stocker les informations liées au segment de clientèle.

Champs dans la table DimCustomerSegment :

- segmentide
- nom de segment

Tâche 4 : Concevoir la table de faits MyFactSales

Notez les champs de la table MyFactSales dans n'importe quel éditeur de texte, un champ par ligne.

Solution:

Dans cette tâche, vous allez concevoir la table de faits MyFactSales pour stocker les informations liées aux ventes.

Champs de la table FactSales :

- ID commercial
- ID produit
- quantités vendues
- prixpar unité
- segmentide
- identifiant de la date

La table de faits FactSales stockera des informations sur chaque transaction de vente, notamment l'identifiant unique (salesid), l'identifiant du produit (productid), la quantité vendue (quantitysold), le prix unitaire (priceperunit), l'identifiant du segment de clientèle (segmentid) et l'identifiant de la date (dateid). Cette table servira de référentiel central pour les données de vente et permettra l'analyse et la création de rapports sur diverses dimensions.

Exercice 2 : Créer un schéma pour un entrepôt de données sur PostgreSQL

Ouvrez pgAdmin et créez une base de données nommée **Practice** , puis créez les tables suivantes.

Tâche 5 : Créer la table de dimension MyDimDate

Créez la table MyDimDate.

Prenez une capture d'écran de l'instruction SQL que vous avez utilisée pour créer la table MyDimDate.

```
CREATE TABLE MyDimDate (
```

```
    dateid INT PRIMARY KEY,
    year INT,
    month INT,
    monthname VARCHAR(20),
    day INT,
    weekday INT,
    weekdayname VARCHAR(20)
);
```

Vous pouvez exécuter cette instruction SQL dans votre outil de gestion de base de données SQL préféré, tel que MySQL Workbench, pgAdmin ou SQL Server Management Studio, pour créer la table DimDate. Une fois créée, vous pouvez vérifier la structure de la table et en prendre une capture d'écran.

Tâche 6 : Créer la table de dimension MyDimProduct

Créez la table MyDimProduct.

```
CREATE TABLE MyDimProduct (
    productid INT PRIMARY KEY,
    productname VARCHAR(255)
);
```

Cette instruction SQL va créer une table nommée DimProduct avec deux colonnes : productid comme clé primaire et productname pour stocker le nom ou la description du produit. Vous pouvez exécuter cette instruction SQL dans votre outil de gestion de base de données SQL préféré pour créer la table DimProduct.

Tâche 7 : Créer la table de dimension MyDimCustomerSegment

```
CREATE TABLE MyDimCustomerSegment (
    segmentid INT PRIMARY KEY,
    segmentname VARCHAR(255)
);
```

Cette instruction SQL crée une table nommée DimCustomerSegment avec deux colonnes : segmentid comme clé primaire et segmentname pour stocker le nom ou la description du segment client. Vous pouvez exécuter cette instruction SQL dans votre outil de gestion de base de données SQL préféré pour créer la table DimCustomerSegment.

Tâche 8 : Créer la table de faits MyFactSales

```
CREATE TABLE MyFactSales (
    salesid INT PRIMARY KEY,
    productid INT,
    quantitysold INT,
    priceperunit DECIMAL (10, 2),
    segmentid INT,
    dateid INT
);
```

Cette instruction SQL créera une table nommée MyFactSales avec les colonnes suivantes :

- salesid : clé primaire permettant d'identifier de manière unique chaque enregistrement de vente.
- productid : Identifiant du produit vendu.
- quantitévendue : Quantité du produit vendue.
- priceperunit : Prix par unité du produit vendu.
- segmentid : Identifiant du segment client.
- dateid : Identifiant de la date de la transaction.

Exercice 3 – Charger des données dans l'entrepôt de données

Dans cet exercice, vous allez charger les données dans les tables.

Après la conception initiale du schéma, vous avez été informé que les données ne pouvaient pas être collectées dans le format initialement prévu en raison de problèmes opérationnels. Cela signifie que les tables précédentes (MyDimDate, MyDimProduct, MyDimCustomerSegment, MyFactSales) de la base de données *Practice* et leurs attributs associés ne sont plus applicables à la conception actuelle. L'entreprise a désormais fourni des données dans des fichiers CSV conformément à la nouvelle conception.

Vous devrez charger les données fournies par l'entreprise au format CSV. Tout d'abord, créez une nouvelle base de données nommée **PracProj** . Ensuite, créez les tables DimDate, DimProduct, DimCustomerSegment et FactSales conformément au nouveau schéma. Suivez les instructions du laboratoire pour exécuter les nouveaux scripts et charger les données des fichiers CSV dans les tables appropriées.

Tâche 9 : Charger des données dans la table de dimension DimDate

Téléchargez les données depuis https://cf-courses-data.s3.us.cloud-object-storage.appdomain.cloud/-omGFpVSWBIZKFSCxUkBwg/DimDate.csv .

Chargez ces données dans la table DimDate.

Prenez une capture d'écran des 5 premières lignes de la table DimDate.

Nommez la capture d'écran `9-DimDate.jpg`. (Les images peuvent être enregistrées avec l'extension .jpg ou .png.)

Solution:

Étape 1 : Ouvrez pgAdmin. Créez ensuite la base de données **PracProj** et connectez-vous à la base de données PracProj.

Étape 2 : Créer la table DimDate

```
CREATE TABLE DimDate (
    Dateid INT PRIMARY KEY,
    date DATE NOT NULL,
    Year INT NOT NULL,
    Quarter INT NOT NULL,
    QuarterName VARCHAR(2) NOT NULL,
    Month INT NOT NULL,
    Monthname VARCHAR(255) NOT NULL,
    Day INT NOT NULL,
    Weekday INT NOT NULL,
    WeekdayName VARCHAR(255) NOT NULL
);
```

Étape 3 : Utilisez l'outil d'importation dans pgAdmin pour charger votre fichier CSV dans la table

- Accédez à la table DimDate dans pgAdmin, faites un clic droit dessus, sélectionnez « Importer/Exporter ».

  Remarque : assurez-vous de télécharger les fichiers vers ce chemin : /var/lib/pgadmin/

- Choisissez « Importer », sélectionnez votre fichier CSV et suivez les instructions pour importer les données.

Étape 4 : Exécutez cette requête pour sélectionner les 5 premières lignes.

```
SELECT * FROM DimDate LIMIT 5;
```

Après avoir exécuté la requête SELECT, les 5 premières lignes de votre table DimDate seront affichées dans le panneau de sortie de la requête.

Étape 5 : Prenez la capture d'écran.

Prenez une capture d'écran de ce panneau avec les résultats. Vous pouvez utiliser l'outil de capture sur Windows, `Shift+Command+4`sur Mac ou l'équivalent sur votre système d'exploitation.

Tâche 10 : Charger des données dans la table de dimension DimProduct

Téléchargez les données depuis https://cf-courses-data.s3.us.cloud-object-storage.appdomain.cloud/Y-76u4An3zb5R6HxxFPabA/DimProduct.csv .

Chargez ces données dans la table DimProduct.
Prenez une capture d'écran des 5 premières lignes de la table DimProduct.
Nommez la capture d'écran `10-DimProduct.jpg`. (Les images peuvent être enregistrées avec l'extension .jpg ou .png.)

Solution:

Étape 1 : Créer la table DimProduct

```
CREATE TABLE DimProduct (
    Productid INT PRIMARY KEY,
    Producttype VARCHAR(255) NOT NULL
);
```

Étape 2 : Utilisez l'outil d'importation dans pgAdmin pour charger votre fichier CSV dans la table

- Accédez à la table DimProduct dans pgAdmin, faites un clic droit dessus, sélectionnez « Importer/Exporter ».
- Choisissez « Importer », sélectionnez votre fichier CSV et suivez les instructions pour importer les données.

Étape 3 : Exécutez cette requête pour sélectionner les 5 premières lignes

```
SELECT * FROM DimProduct LIMIT 5;
```

Après avoir exécuté la requête SELECT, les 5 premières lignes de votre table DimProduct seront affichées dans le panneau de sortie de la requête.

Étape 4 : Prenez la capture d'écran.

Prenez une capture d'écran de ce panneau avec les résultats. Vous pouvez utiliser l'outil de capture sur Windows, `Shift+Command+4`sur Mac ou l'équivalent sur votre système d'exploitation.

Tâche 11 : Charger des données dans la table de dimension DimCustomerSegment

Téléchargez les données depuis https://cf-courses-data.s3.us.cloud-object-storage.appdomain.cloud/h_dnxb8yzQyVjeb8oYnm8A/DimCustomerSegment.csv .

Chargez ces données dans la table DimCustomerSegment.

Prenez une capture d'écran des 5 premières lignes de la table DimCustomerSegment.

Nommez la capture d'écran `11-DimCustomerSegment.jpg`. (Les images peuvent être enregistrées avec l'extension .jpg ou .png.)

Solution:

Étape 1 : Créer la table DimCustomerSegment

```
CREATE TABLE DimCustomerSegment (
    Segmentid INT PRIMARY KEY,
    City VARCHAR(255) NOT NULL
);
```

Étape 2 : Utilisez l'outil d'importation dans pgAdmin pour charger votre fichier CSV dans la table

- Accédez à la table DimCustomerSegment dans pgAdmin, faites un clic droit dessus, sélectionnez « Importer/Exporter ».
- Choisissez « Importer », sélectionnez votre fichier CSV et suivez les instructions pour importer les données.

Étape 3 : Exécutez cette requête pour sélectionner les 5 premières lignes

```
SELECT * FROM DimCustomerSegment LIMIT 5;
```

Après avoir exécuté la requête SELECT, les 5 premières lignes de votre table DimCustomerSegment seront affichées dans le panneau de sortie de la requête.

Étape 4 : Prenez la capture d'écran.

Prenez une capture d'écran de ce panneau avec les résultats. Vous pouvez utiliser l'outil de capture sur Windows, `Shift+Command+4`sur Mac ou l'équivalent sur votre système d'exploitation.

Tâche 12 : Charger des données dans la table de faits FactSales

Téléchargez les données depuis https://cf-courses-data.s3.us.cloud-object-storage.appdomain.cloud/a8kTjzvpdqzOp46ODatyAA/FactSales.csv

Chargez ces données dans la table FactSales.

Prenez une capture d'écran des 5 premières lignes du tableau FactSales.

Nommez la capture d'écran `12-FactSales.jpg`. (Les images peuvent être enregistrées avec l'extension .jpg ou .png.)

Solution:

Étape 1 : Créer la table FactSales

```
CREATE TABLE FactSales (
    Salesid VARCHAR(255) PRIMARY KEY,
    Dateid INT NOT NULL,
    Productid INT NOT NULL,
    Segmentid INT NOT NULL,
    Price_PerUnit DECIMAL(10, 2) NOT NULL,
    QuantitySold INT NOT NULL,
    FOREIGN KEY (Dateid) REFERENCES DimDate(Dateid),
    FOREIGN KEY (Productid) REFERENCES DimProduct(Productid),
    FOREIGN KEY (Segmentid) REFERENCES DimCustomerSegment(Segmentid)
);
```

Étape 2 : Utilisez l'outil d'importation dans pgAdmin pour charger votre fichier CSV dans la table

- Accédez à la table FactSales dans pgAdmin, faites un clic droit dessus, sélectionnez « Importer/Exporter ».
- Choisissez « Importer », sélectionnez votre fichier CSV et suivez les instructions pour importer les données.

Étape 3 : Exécutez cette requête pour sélectionner les 5 premières lignes

```
SELECT * FROM FactSales LIMIT 5;
```

Après avoir exécuté la requête SELECT, les 5 premières lignes de votre table FactSales seront affichées dans le panneau de sortie de la requête.

Étape 4 : Prenez la capture d'écran.

Prenez une capture d'écran de ce panneau avec les résultats. Vous pouvez utiliser l'outil de capture sur Windows, `Shift+Command+4`sur Mac ou l'équivalent sur votre système d'exploitation.

Exercice 4 – Écrire des requêtes d'agrégation et créer une vue matérialisée

Dans cet exercice, vous allez interroger les données que vous avez chargées dans l'exercice précédent.

Tâche 13 : Créer une requête d'ensembles de regroupement

Créez une requête de regroupement d'ensembles à l'aide des colonnes productid, producttype, total sales. Prenez une capture d'écran du SQL et des lignes de sortie.

Nommez la capture d'écran `13-groupingsets.jpg`. (Les images peuvent être enregistrées avec l'extension .jpg ou .png.)

Solution:

```
SELECT
    p.Productid,
    p.Producttype,
    SUM(f.Price_PerUnit * f.QuantitySold) AS TotalSales
FROM
    FactSales f
INNER JOIN
    DimProduct p ON f.Productid = p.Productid
GROUP BY GROUPING SETS (
    (p.Productid, p.Producttype),
    p.Productid,
    p.Producttype,
    ()
)
ORDER BY
    p.Productid,
    p.Producttype;
```

Dans cette requête :

- Vous joignez FactSales 'f' avec DimProduct 'p' sur leur productid pour corréler chaque vente avec son type de produit.
- Vous utilisez GROUPING SETS pour spécifier les différents niveaux d'agrégation :

– Par Productid et Producttype
– Par Productid seul

– Par Producttype seul
– Et un total général avec (), qui ne regroupe par aucune colonne et renvoie donc la somme de toutes les ventes.

- Vous calculez le TotalSales en multipliant le Price_PerUnit par QuantitySold pour chaque vente.

La clause ORDER BY garantit que les résultats sont classés par identifiant de produit, puis par type de produit.

Tâche 14 : Créer une requête cumulative

Créez une requête cumulative à l'aide des colonnes année, ville, ID produit et ventes totales. Prenez une capture d'écran du SQL et des lignes de sortie.

Nommez la capture d'écran `14-rollup.jpg`. (Les images peuvent être enregistrées avec l'extension .jpg ou .png.)

Solution:

```
SELECT
    d.Year,
    cs.City,
    p.Productid,
    SUM(f.Price_PerUnit * f.QuantitySold) AS TotalSales
FROM
    FactSales f
JOIN
    DimDate d ON f.Dateid = d.Dateid
JOIN
    DimProduct p ON f.Productid = p.Productid
JOIN
    DimCustomerSegment cs ON f.Segmentid = cs.Segmentid
GROUP BY ROLLUP (d.Year, cs.City, p.Productid)
ORDER BY
    d.Year DESC,
    cs.City,
    p.Productid;
```

Cette requête effectue les opérations suivantes :

- Rejoint FactSales avec DimDate sur Dateid, DimProduct sur Productid et DimCustomerSegment sur Segmentid.
- Sélectionne l'année dans DimDate, la ville dans DimCustomerSegment et l'ID du produit dans DimProduct.
- Calcule les ventes totales en multipliant le prix unitaire par la quantité vendue pour chaque entrée de vente.
- Regroupe les résultats à l'aide de la fonction ROLLUP pour créer un ensemble de regroupement qui inclut toutes les combinaisons d'année, de ville et d'ID de produit, ainsi que leurs sous-totaux respectifs et un total général pour toutes les ventes.

La clause ORDER BY garantit que les résultats sont d'abord classés par année dans l'ordre décroissant, puis par ville et par ID de produit.

Tâche 15 : Créer une requête de cube

Créez une requête de cube en utilisant les colonnes année, ville, ID produit et ventes moyennes.
Prenez une capture d'écran du SQL et des lignes de sortie.
Nommez la capture d'écran `15-cube.jpg`. (Les images peuvent être enregistrées avec l'extension .jpg ou .png.)

Solution:

```
SELECT
    d.Year,
    cs.City,
    p.Productid,
    AVG(f.Price_PerUnit * f.QuantitySold) AS AverageSales
FROM
    FactSales f
INNER JOIN
    DimDate d ON f.Dateid = d.Dateid
INNER JOIN
    DimProduct p ON f.Productid = p.Productid
INNER JOIN
    DimCustomerSegment cs ON f.Segmentid = cs.Segmentid
GROUP BY CUBE (d.Year, cs.City, p.Productid);
```

Dans cette requête :

- La clause CUBE est utilisée dans GROUP BY pour créer des sous-totaux pour toutes les combinaisons d'année, de ville et d'ID de produit en plus du total général sur tous les groupes.
- AVG(f.Price_PerUnit * f.QuantitySold) calcule les ventes moyennes, en prenant en compte à la fois le prix unitaire et la quantité vendue.
- INNER JOIN est utilisé pour joindre la table FactSales avec les tables de dimension DimDate, DimProduct et DimCustomerSegment.

Tâche 16 : Créer une vue matérialisée

Créez une vue matérialisée nommée max_sales en utilisant les colonnes city, productid, producttype et max sales.
Prenez une capture d'écran du SQL.
Nommez le fichier `screenshot 16-mv.jpg`. (Les images peuvent être enregistrées avec l'extension .jpg ou .png.)

Solution:

```
CREATE MATERIALIZED VIEW max_sales AS
SELECT
    cs.City,
    p.Productid,
    p.Producttype,
    MAX(f.Price_PerUnit * f.QuantitySold) AS MaxSales
FROM
    FactSales f
JOIN
    DimProduct p ON f.Productid = p.Productid
JOIN
    DimCustomerSegment cs ON f.Segmentid = cs.Segmentid
GROUP BY
    cs.City,
    p.Productid,
    p.Producttype
WITH DATA;
```

Cette instruction crée la vue matérialisée et la remplit avec les données actuelles des tables jointes. La clause WITH DATA indique à PostgreSQL de remplir immédiatement la vue avec les résultats de la requête. Si vous souhaitez créer la vue sans la remplir de données, vous devez utiliser WITH NO DATA.

Pour mettre à jour la vue matérialisée avec les dernières données, vous devez utiliser la commande REFRESH MATERIALIZED VIEW :

```
REFRESH MATERIALIZED VIEW max_sales;
```

© IBM Corporation 2023. Tous droits réservés.