# Section 3 : Ecriture des requêtes basiques pour interroger une table

**Scénario** :

Vous venez d’être embauché en tant que Data Analyst par un grand Groupe de concessionnaires de véhicules. Votre Manager, qui est aussi le responsable du département *Analytics* de l’entreprise *First Auto*, est tellement heureux de vous voir qu’il vous a assigné vos premières tâches dès votre premier jour. En gros, il veut que vous fassiez une analyse descriptive des données disponibles afin d’en tirer des informations utiles pour l’entreprise. Sur ce coup, il vous demande de la créativité dans votre analyse mais il veut particulièrement une analyse de haut niveau sur les ventes annuelles chez les concessionnaires à travers le pays.

Vous n'avez pas accès à une base de données tout simplement parce-que

l’entreprise n’en dispose pas à l’heure actuelle. Il existe des fichiers de données (aux formats CSV pour la majorité) éparpillés à travers différents départements de l’entreprise. Le Manager vous a envoyé deux fichiers CSV

(***concessionnaires.csv*** et ***produits.csv***) pour que vous commencez votre analyse.

Etant donné qu’il s’agit de votre première mission dans l’entreprise qui fait office de test pour votre période d’essai, vous devez vous surpasser pour fournir un travail impeccable et des analyses qui aideront les parties prenantes à

prendre de meilleures décisions.

Vos collègues du service *Analytics* et de tous les autres départements de l’entreprise vous considèrent déjà comme le Super Héros venu pour les délivrer de la navigation à vue dans laquelle ils étaient plongés. Ils croient fermement que votre arrivée dans l’entreprise les orientera vers une approche ***Data-Driven*** pour de meilleurs résultats et une performance optimale. Ceci vous met davantage la pression mais heureusement vous n’êtes pas du genre à vous dégonfler :)

C’est donc avec joie que vous enfilez votre costume de SuperMan / SuperWoman pour affronter la quantité de données disponibles et en extraire

toutes les informations utiles à la décision.



**N.B** : ***Une entreprise Data Driven est une entreprise « pilotée par les données ». Autrement dit, il s’agit d’une entreprise qui s’appuie sur l’analyse des données à sa disposition pour prendre des décisions et orienter son évolution.***

C’EST PARTI !

* **Création d’une base de données qui va rassembler toutes les données que vous utiliserez au cours de cette mission**.

Ayant déjà à votre disposition deux fichiers CSV et sachant que vous aurez certainement d’autres fichiers à votre disposition, vous décidez de créer une base de données centrale qui rassemblera toutes ces données. Cela vous sera utile non seulement pour votre mission actuelle mais aussi pour les autres missions à venir et cela peut être le point de départ de la construction d’une

Architecture Data performante pour l’entreprise.

**Exercice 1** : Créez une base de données nommée ***firstauto\_db*** dans laquelle vous devez créer deux tables nommées ***concessionnaires*** et ***produits***

respectivement à partir des fichiers ***concessionnaires.csv*** et ***products.csv***.

* **Ecriture de requêtes avec les mots clés tels que SELECT, WHERE, ORDER BY, LIMIT, etc**.

La syntaxe d’une requête typique SQL est :

*SELECT [COLUMNS LIST]*

*FROM [TABLE NAME]*

*WHERE [CONDITION]*

*ORDER BY [COLUMN NAME] [ASC|DESC]*

**SELECT** est le mot clé principal du langage SQL. Il permet de récupérer les données. Par exemple, la requête *SELECT \* FROM produits* permet de

récupérer toutes les colonnes de la table *produits*.

Au lieu de récupérer toutes les colonnes (grâce au symbole \*), vous pouvez spécifier les noms des colonnes que vous voulez récupérer.

**Exemples** :

Récupérer les colonnes *product\_id*, *model* et *product\_type* de la table *produits*. A l’aide de l’outil de visualisation de Databricks, construire un diagramme à barre montrant les différents types de produits.

Construire un diagramme circulaire (*Pie Chart*) montrant le pourcentage de chaque type de produits.

On peut utiliser SELECT avec le mot clé **AS** (alias) pour renommer des

colonnes.

**Exemples** :

Sélectionnez les colonnes *product\_id* et *model* tout en renommant la colonne *product\_id* en *id*.

Au cours de la formation, nous verrons certainement d’autres possibilités d’utilisation du mot clé AS.

**WHERE** est le mot clé utilisé pour filtrer les observations. Si vous voulez récupérer des données précises et pertinentes pour analyser un phénomène particulier, WHERE doit être forcément intégrée dans votre requête. Il permet de facto de limiter le nombre d’observations retournées par SELECT.

**Exemple** : Quels sont les noms des modèles des véhicules produits en 2014 ?

WHERE s’utilise aussi avec les mots clés **AND**, **OR**, **IN**, **NOT IN**, **IS NULL**,

**IS NOT NULL**, etc.

**Exemples** :

Quels sont les modèles qui ont été construits en 2014 et qui ont également un prix de détail suggéré par le fabricant (attribut *base\_msrp*) de moins de 1 000 $ ?

Quels sont les modèles de véhicules construits en 2016 ou en 2019 ?

Écrivez une requête pour obtenir tous les produits avec des modèles entre 2014 et 2016 inclus, ainsi que tous les produits qui sont des scooters.

Quels sont les modèles des véhicules construits en 2014, 2016 ou 2019 ?

Écrivez une requête pour récupérer les noms des modèles des véhicules qui n’ont pas été construits en 2014, 2016 ou 2019.

Quels sont les produits qui sont encore en cours de production, c’est-à-dire ceux pour lesquels la date de fin de production n’est pas encore connue ?

Quels sont les produits qui ne sont plus en production, c’est-à-dire ceux pour lesquels la date de fin de production est connue ?

**ORDER BY**

Pour mieux comprendre vos données, vous aurez souvent besoin de voir les

observations (c’est-à-dire les lignes) dans un ordre spécifique. C’est le mot clé ORDER BY qui permet d’ordonner les lignes retournées par une requête.

Notez que par défaut ORDER BY classe les données dans un ordre croissant. Si la colonne par laquelle vous ordonnez les observations est une colonne de type STRING (chaînes de caractères) alors dans ce cas l’ordre ascendant veut dire ordre alphabétique. S’il s’agit d’une colonne numérique, l’ordre ascendant veut dire du plus petit au plus grand. S’il s’agit d’une colonne de type TIMESTAMP (date et temps) alors l’ordre ascendant veut dire de la date la plus ancienne à la date la plus récente.

**Exemple** :

Écrivez une requête afin de voir tous les produits répertoriés par date de

fabrication, du plus ancien au plus récent.

Il est possible d’ordonner les observations en fonction de plusieurs colonnes.

**Exemple** :

Écrivez une requête pour trier toutes les observations d'abord par année du modèle, de la plus récente à la plus ancienne, puis par le prix (attribut ***base\_msrp***) du plus petit au plus grand.

**LIMIT**

Si vous avez une table avec des milliers voir des millions de lignes, il n’est pas forcément nécessaire de toutes les afficher. Dans ce cas, vous pouvez limiter le nombre de lignes retournées par la requête.

**Exemple** : Affichez les 5 premières observations de la table nommée

*concessionnaires*.

Nous venons d’apprendre l’utilisation de certains mots clés rencontrés dans les requêtes SQL. Vous êtes maintenant capables d’interroger une table de données afin de récupérer des données pour répondre à des questions précises. Dans la suite de la formation, vous en apprendrez davantage sur la création de requêtes SQL ainsi que son utilisation pour la préparation, la transformation et l’analyse des données.

Avant de passer à la section 4, vous devez réaliser les deux exercices ci-dessous afin de pratiquer tout ce que vous avez appris.

**Exercice 2** :Interrogation de la table des commerciaux de l’entreprise

Le responsable du département Marketing et Ventes vient de rentrer des congés et il est très heureux de votre arrivée dans l’entreprise. Selon lui, vous serez d’une grande aide pour son département qui est l’un des plus importants de la

société.

Il vous a fourni un nouveau fichier nommé ***commerciaux.csv*** qui contient les informations sur les commerciaux hommes et femmes de son département. Dans l’immédiat, il veut la liste des 10 premières vendeuses embauchées classées de la première à la dernière embauchée avec leurs noms, prénoms et

noms d’utilisateurs.

**Exercice 3** : Préparation des campagnes Marketing

Depuis que le Responsable Ventes et Marketing est revenu des congés, il ne vous lâche pas d’une semelle surtout qu’il découvre votre expertise en SQL. Son département est en pleine préparation de campagnes Marketing pour la promotion des véhicules produits par les concessionnaires du Groupe et vous leur serez d’une grande utilité dans cette préparation. Il vous a remis un fichier

CSV nommé ***clients.csv*** et il veut les informations suivantes :

* liste des e-mails de tous les clients de l'État de Floride par ordre

alphabétique ;

* liste des clients à New York dans l'État de New York avec leurs prénoms, noms de famille et coordonnées e-mail. Ils doivent être classés par ordre

alphabétique du nom de famille suivi du prénom.

* Liste de tous les clients avec leurs numéros de téléphones triée par la date

à laquelle le client a été ajouté dans le fichier de données.

Félicitations à vous :). Vous avez appris énormément de choses dans cette section 3. Vous êtes maintenant capables d’écrire des requêtes basiques pour récupérer des données pertinentes afin de répondre à des questions précises.

Votre Manager ainsi que tous vos autres collègues sont de plus en plus fiers de vous et vous pouvez l’être aussi :). Ne vous arrêtez pas en si bon chemin. Rejoignez-moi dans la prochaine section où vous apprendrez à joindre des

tables et à utiliser des sous-requêtes.

# SECTION 4 : Jointures de tables et écriture de sous-requêtes

Dans la section précédente nous avons appris comment écrire des requêtes basiques sur des tables prises individuellement. La plupart du temps, les

données qui nous intéressent réellement sont réparties à travers plusieurs tables.

Nous devons donc écrire des requêtes pour pouvoir joindre ces tables et en

extraire les données que nous recherchons.

Dans cette section, vous apprendrez différents types de jointures.

Mais avant d’apprendre à effectuer des jointures de tables, laissez-moi vous présenter le concept de Vues Temporaires (*Temporary Views* en Anglais).

**Une vue temporaire** (ou tout simplement vue) est comme une table. Mais contrairement à une table classique, la vue ne se poursuit pas lorsque vous redémarrez le cluster ou basculez vers un nouveau notebook. De plus, les vues temporaires n'apparaîtront pas dans l'onglet Data de l’interface de Databricks. C’est pourquoi on les qualifie de temporaires.

**Quelle est l’utilité des vues ?** De manière pratique, une vue vous permet de donner un nom au résultat de votre requête SQL afin de pouvoir l’utiliser ultérieurement. Elles sont très utiles pour simplifier l’écriture de requêtes complexes y compris les jointures.

**Exemples** :

Créez une vue temporaire pour stocker la table contenant les données sur les concessionnaires de l'État de Californie. Nommez la vue par

***concessionnaires\_californie***.

Pour comprendre les différents types de jointure dans SQL, je vous conseille de lire cet excellent article de Wikipédia qui explique non seulement la théorie mais aussi donne des exemples pratiques avec codes et résultats : <https://en.wikipedia.org/wiki/Join_(SQL)>

Dans cette section, vous apprendrez les types de jointures très souvent

rencontrés en entreprises à savoir : vous apprendrez six (06) différents types de jointure : ***inner join***, ***left join***, ***right join*** et ***full join***.

**INNER JOIN**

La jointure interne de deux tables A et B permet de créer une nouvelle table contenant des observations qui sont à la fois présentes dans A et dans B. Il s’agit de l’intersection de A et de B.

**Exemple** : Votre Manager vous demande la liste des commerciaux du Groupe

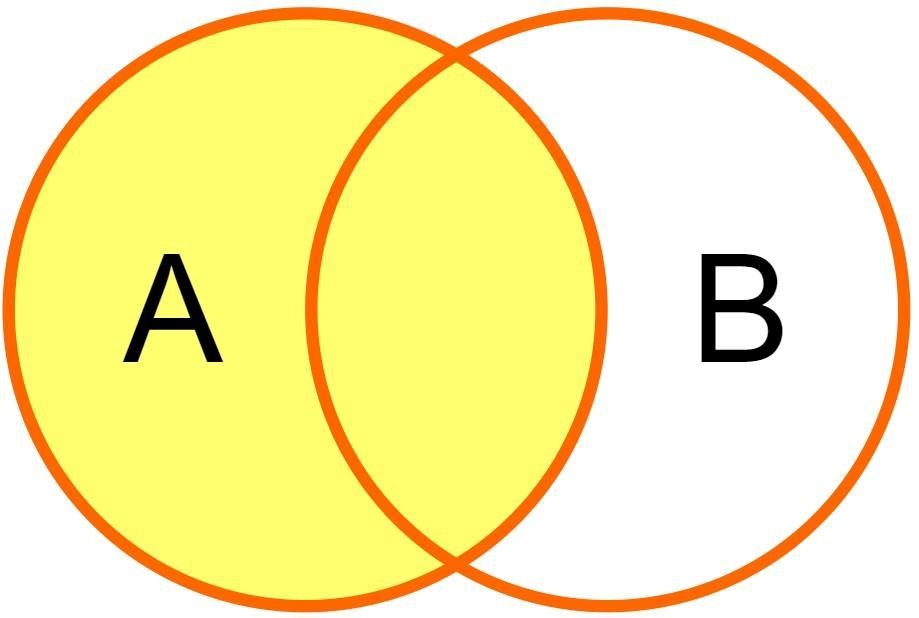
qui travaillent chez les concessionnaires de Californie.

* Nous pouvons identifier ces commerciaux en utilisant la vue

*concessionnaires\_ca* précédemment créée ;

* Nous pouvons aussi identifier ces commerciaux en procédant directement à une jointure interne des tables *concessionnaires* et *commerciaux*.

**LEFT JOIN (LEFT OUTER JOIN)**



La jointure à gauche permet de conserver toutes les observations dans la première table (ou table de gauche) qu'elles apparaissent ou non dans la deuxième table (ou table de droite). Par conséquent, des valeurs manquantes peuvent apparaître au niveau du résultat.

Autrement dit une jointure à gauche de A et de B retourne toutes les

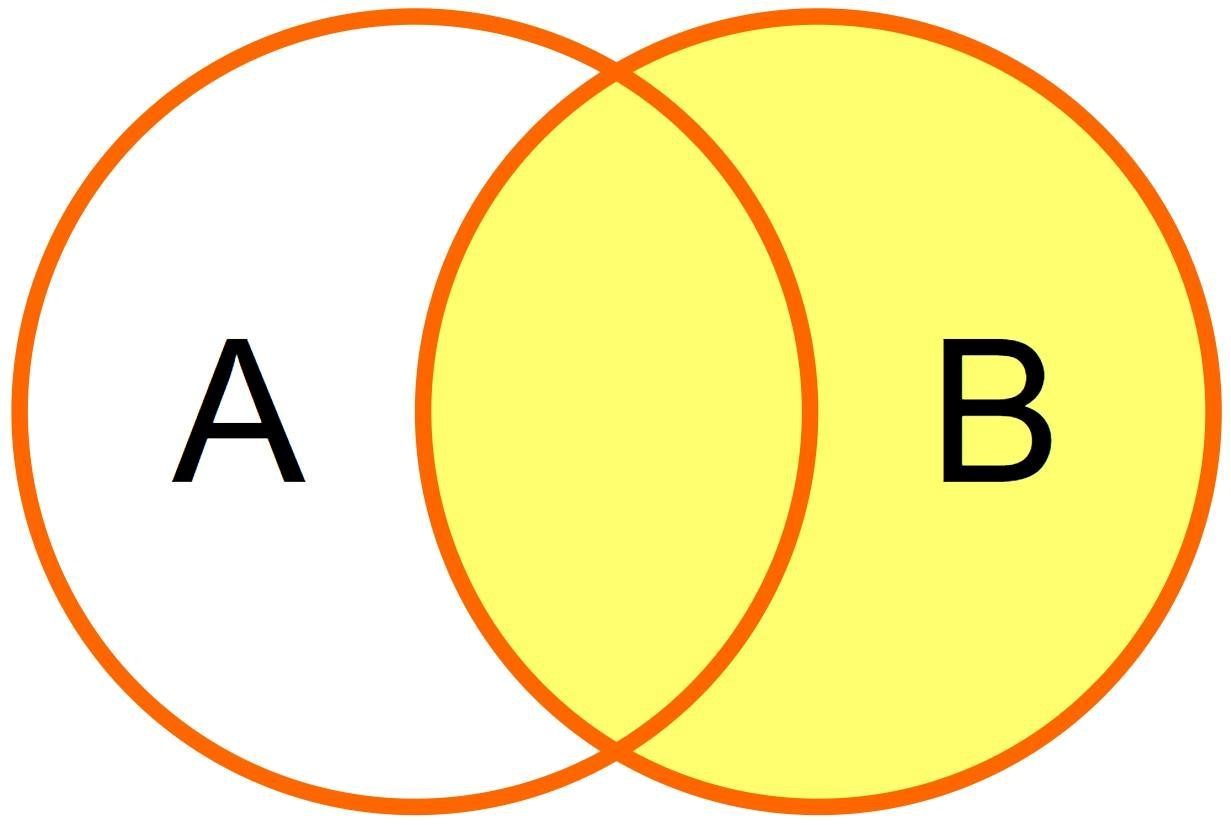
observations de la jointure interne de A et de B plus toutes les observations de la table de gauche (A) qui ne correspondent pas à la table de droite (B) y compris les lignes avec des valeurs NULL (vides) dans la colonne de lien (colonne par laquelle la jointure est effectuée).

**Exemple** :

Le responsable du département Marketing et Ventes souhaite connaître les emails des clients qui n’ont jamais reçu un email de la part de l’entreprise. Il vous a remis un fichier CSV nommé ***emails.csv***.

Afin de répondre à cette demande, vous pouvez effectuer une jointure à gauche de la table *clients* à la table *emails*.

**RIGHT JOIN (RIGHT OUTER JOIN)**



La jointure à droite permet de conserver toutes les observations dans la deuxième table (ou table de droite) qu'elles apparaissent ou non dans la première table (ou table de gauche). Par conséquent, des valeurs manquantes peuvent apparaître au niveau du résultat.

Autrement dit une jointure à droite de A et de B retourne toutes les observations de la jointure interne de A et de B plus toutes les observations de la table de droite (B) qui ne correspondent pas à la table de gauche (A) y compris les lignes avec des valeurs NULL (vides) dans la colonne de lien (colonne par laquelle la

jointure est effectuée).

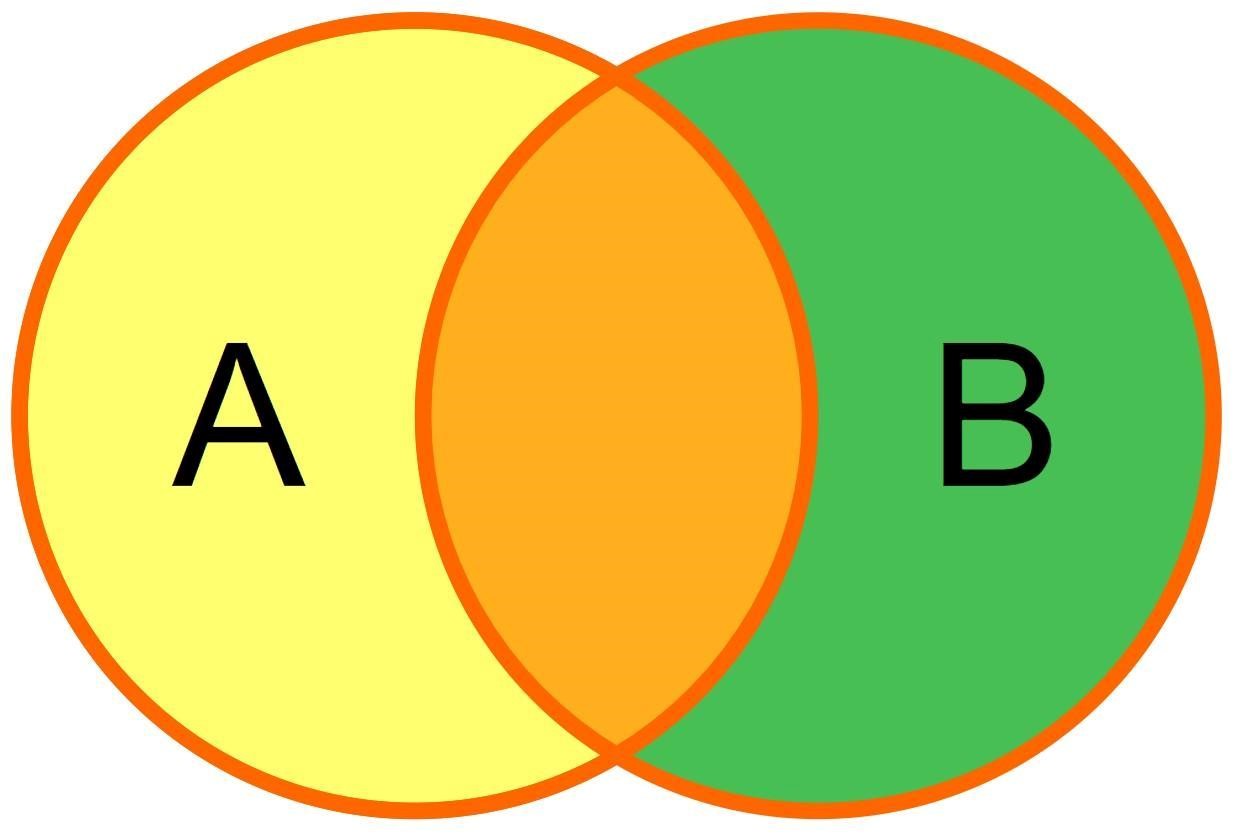
Les jointures gauche et droite sont en effet des images l'une de l'autre.

**Exemple** :

Répondez à la demande précédente du responsable marketing en effectuant une

jointure à droite des tables *emails* et *clients*.

**FULL JOIN (FULL OUTER JOIN)**



Conceptuellement, une jointure externe complète combine l'effet de

l'application de jointures externes gauche et droite.

Si vous voulez conserver toutes les observations présentes au niveau des deux tables, qu'elles correspondent ou non, alors vous devez réaliser une jointure complète.

**Exemple** : Réalisez une jointure complète des tables *emails* et *clients*.

**Exercice 4** : Le responsable des ventes de l’entreprise *First Auto* souhaite une liste de tous les clients qui ont acheté une voiture. Vous devez créer une requête qui renverra les attributs de la table *clients* tels que les prénoms, noms de famille et numéros de téléphone valides des clients qui ont acheté une voiture.

Pour cette tâche, il vous a remis un nouveau fichier csv nommé ***ventes.csv***.

* Faites une jointure interne de la table *ventes* à la table *clients* et stockez le

résultat dans une vue temporaire nommée *ventes\_inter\_clients*.

* Faites une jointure de la vue *ventes\_inter\_clients* à la table produits et

filtrez pour obtenir uniquement les observations concernant les voitures.

* Essayez d’obtenir le résultat avec une seule requête.

**SOUS-REQUÊTES (SUBQUERIES)**

Une sous-requête est une requête à l’intérieur d’une requête. Elle est mise dans une parenthèse. Les sous-requêtes sont souvent utilisées lorsqu’on veut

effectuer des jointures.

**Exemple** : Utilisez une sous-requête pour récupérer les données des

commerciaux travaillant chez des concessionnaires en Californie.

**UNION**

Avec les jointures, des nouvelles colonnes sont ajoutées de manière horizontale. Pour une raison ou une autre, vous pouvez être intéressés par l'assemblage vertical de plusieurs requêtes ; c'est-à-dire en conservant le même nombre de colonnes mais en ajoutant plusieurs lignes. C’est ce que permet de faire les requêtes avec le mot clé UNION.

**Exemple** : Votre Manager vous demande la liste des clients et des

concessionnaires avec leurs adresses de rue, villes, états et codes postaux car il veut visualiser ces adresses dans Google Maps.

**Exercice 5** : Le Département Marketing veut organiser une soirée festive entre les commerciaux de Los Angeles en Californie et les clients de cette même ville et de ce même état. Pour les préparatifs, vous devez leur fournir la liste de ces commerciaux et de ces clients avec leurs prénoms et noms de famille. La liste devra contenir une troisième colonne indiquant si l’invité est un client ou s’il est

un commercial.

# SECTION 5 : Transformation des données

Très souvent nous voulons que les données soient sous une certaine forme, autre que celle des données brutes, avant de les analyser. C’est là qu'interviennent les techniques de transformation des données. Les tâches de transformation des données concernent entre autres la suppression de valeurs, la substitution de valeurs, la création de nouvelles colonnes, le nettoyage des données, etc.

Dans cette section, vous apprendrez de nouveaux outils qui vous aideront à transformer vos données brutes afin de mieux les comprendre et de les analyser.

**CASE WHEN**

La fonction *CASE WHEN* est souvent utilisée pour créer une nouvelle colonne à partir de colonnes existantes dans une table.

**Exemple** : L’équipe Marketing vous demande de créer une colonne nommée

*types\_clients* afin de catégoriser les clients en trois groupes : ***Clients Élites*** pour

les clients vivant dans le code postal 33111, ***Clients Premium*** pour ceux qui vivent dans le code postal 33124 et ***Clients Standards*** pour tous les autres

clients restants.

**Exercice 6** : Votre Manager vous demande une liste des clients avec deux colonnes : l’identifiant du client et sa région. La nouvelle colonne nommée ***region*** que vous devez créer prendra la valeur ***Nouvelle-Angleterre*** pour les clients vivant dans les états MA, NH, VT, ME, CT, ou RI, ***Sud-Est*** pour les clients vivants dans les états GA, FL, MS, AL, LA, KY, VA, NC, SC, TN, VI,

WV, ou AR et ***Autre*** pour les clients vivant dans les autres états restants.

N’oubliez pas d’ordonner selon l’ordre ascendant des identifiants des clients.

**COALESCE**

Lorsque vous voulez par exemple remplacer toutes les valeurs manquantes d’une colonne par une valeur par défaut, alors vous pouvez utiliser la fonction COALESCE.

**Exemple** : L’équipe Marketing vous demande la liste des clients avec leurs prénoms, noms de familles et coordonnées géographiques (latitude et longitude). Mais pour les clients dont les coordonnées géographiques sont manquantes, ils veulent que vous remplaciez *null* par “No Latitude” et “No Longitude” respectivement pour la latitude et la longitude.

**LEAST et GREATEST**

Chacune de ces fonctions prend un nombre quelconque de valeurs et renvoie respectivement la plus petite (LEAST) ou la plus grande des valeurs (GREATEST).

**Exemples** :

L’équipe Marketing veut les données sur les scooters dont le prix est inférieur ou égal à 600$.

Créez une autre requête pour récupérer les données des scooters dont le prix est supérieur ou égal à 600$.

**Exercice 7** :

L'équipe du service *Analytics* dans lequel vous avez été embauché souhaite créer un nouveau modèle pour aider à prédire quels clients sont les meilleurs prospects pour le remarketing. Un nouveau data scientist a rejoint l’équipe et ne connaît pas assez bien la base de données pour extraire un ensemble de données pour ce nouveau modèle.

C'est à vous qu'incombe la responsabilité d'aider le nouveau data scientist à préparer et à créer un ensemble de données à utiliser pour former un modèle. Écrivez une requête pour assembler un ensemble de données qui effectuera les

opérations suivantes :

* une jointure interne de la table ***clients*** à la table ***ventes*** ;
* une jointure interne de la table ***ventes*** à la table ***produits*** ;
* une jointure à gauche de la table ***produits*** à la table ***concessionnaires.*** Votre requête doit retourner les colonnes *dealership\_id* et *sales\_amount* de la table ***ventes*** ainsi que toutes les colonnes des tables ***clients*** et ***produits.*** Veillez à remplacer toutes les valeurs manquantes de la colonne *dealership\_id* par - 1. Pour finir, votre requête doit créer une nouvelle variable nommée *benefice* qui sera égale à 1 si *sales\_amount* est supérieure ou égale à *base\_msrp*. Dans le cas

contraire, la variable *benefice* prendra la valeur 0.