|  |  |
| --- | --- |
| SELECT **\*** FROM DELIVERERS; | (\*) Permet la sélection de tout la table |
| SELECT **DISTINCT** CompanyID FROM COMPANYDEL; | (DISTINCT) **retourner uniquement des lignes uniques** en supprimant les doublons dans les résultats d'une requête. |
| SELECT nom, DELIVERERID FROM DELIVERERS WHERE nom **LIKE 'B%'**; | (LIKE) permet de **chercher des valeurs dans une colonne qui correspondent à un modèle précis** (pattern), souvent pour des recherches partielles dans du texte.  **LIKE 'A%'** : Trouve les valeurs qui commencent par "A".  **LIKE '%z'** : Trouve les valeurs qui se terminent par "z".  **LIKE '%car%'** : Trouve les valeurs contenant "car".  **LIKE '\_b%'** : Trouve les valeurs où le 2ᵉ caractère est "b". |
| SELECT nom, sexe, DELIVERERID  FROM DELIVERERS  WHERE numero\_tele\_bureau <> '8467' or numero\_tele\_bureau **IS NULL**; | (IS [NOT] NULL) utilisé pour vérifier si une valeur dans une colonne est **nulle (.resp non nulle)** , c'est-à-dire qu'elle ne contient aucune donnée (.resp contient des données) |
| SELECT nom, ville FROM DELIVERERS WHERE DELIVERERID **IN** (  SELECT DELIVERERID  FROM PENALTIES  **GROUP BY** DELIVERERID  **HAVING** **COUNT(DELIVERERID)** >= 1);  //////////////////////////////////////////////////  SELECT DELIVERERID, **COUNT(DELIVERERID) AS compte**  FROM PERNALTIES  GROUP BY DELIVERERID ; | (IN) permet de tester l’appartenance à un ensemble de valeurs. L’ensemble peut etre spécifié explicitement **Colonne IN(v1, v2, …)** comme il peut être le résultat d’une autre requête imbriquée -> c’est notre cas  (GROUP BY colonne) en SQL est utilisé pour **regrouper les lignes d'une table** en fonction des valeurs d'une ou plusieurs colonnes, souvent pour appliquer des **fonctions d'agrégation** (comme COUNT, SUM, AVG, etc.) sur chaque groupe.  (HAVING condition) permet d’exprimer des conditions sur le groupe de ligne généré.  COUNT(colonne) compte seulement les lignes où la colonne spécifiée n’est pas nulle.  //////////////////////////////////////////////////  (COUNT(colonne) AS Alias) utilisé pour compter le nombre de valeurs non nulles dans une colonne spécifique et lui donner un nom (alias) pour simplifier la lecture du résultat.  //////////////////////////////////////////////////  COUNT(\*) compte toutes les lignes, même celles avec des valeurs nulles |
| SELECT nom, prenom  FROM DELIVERERS  WHERE DELIVERERID IN (  SELECT DELIVERERID  FROM PENALITES  WHERE **TO\_DATE(date\_amende, 'DD/MM/YYYY') > TO\_DATE('31/12/2017', 'DD/MM/YYYY')**  AND DELIVERERID IN (  SELECT DELIVERERID FROM COMPANYDEL  )  )  **ORDER BY nom**;  //////////////////////////////////////////////////  SELECT **nom**, age, ville  FROM clients  **ORDER BY 1**; | (TO\_DATE('chaîne\_de\_caractères', 'format') est une fonction utilisée pour **convertir une chaîne de caractères en une valeur de type** DATE. Elle permet de spécifier le format de la date pour s'assurer que la chaîne est interprétée correctement.  (ORDER BY colonne1 [ASC|DESC], colonne2 [ASC|DESC], ...;) utilisé pour **trier les résultats** d'une requête par une ou plusieurs colonnes. Il permet d'organiser les données par ordre croissant (ASC) ou décroissant (DESC).  //////////////////////////////////////////////////  ORDER BY 1 : Trie les résultats en fonction de la **première colonne** de la liste des colonnes sélectionnées dans la requête.  ORDER BY 2 : Trie les résultats en fonction de la **deuxième colonne**, et ainsi de suite. |
| SELECT DELIVERERID, COMPANYID  FROM COMPANYDEL  WHERE NUMCOLLECTIONS >= 2  AND NUMDELIVERIES = 1  AND DELIVERERID IN (  SELECT DELIVERERID  FROM DELIVERERS  WHERE ville='Stratford'  ); |  |
| SELECT DELIVERERID  FROM DELIVERERS  WHERE annee\_de\_naissance > 1982  AND DELIVERERID IN (  SELECT DELIVERERID  FROM COMPANYDEL  WHERE NUMDELIVERIES >= 1  AND DELIVERERID IN (  SELECT DELIVERERID  FROM COMPANIES  WHERE MANDAT = 'First'  )  )  ORDER BY DELIVERERID DESC; |  |
| SELECT nom  FROM DELIVERERS  WHERE (ville='Inglewood' OR ville='Stratford')  AND DELIVERERID IN (  SELECT DELIVERERID  FROM COMPANYDEL  GROUP BY DELIVERERID  HAVING **COUNT(DISTINCT COMPANYID)** > 1  ); | (COUNT(DISTINCT COMPANYID)) utilisé pour garantir que seules les entreprises uniques sont comptées (utile si des doublons existent dans la table « COMPANYDEL ») |
| SELECT DELIVERERID, **SUM(montant) AS TOTAL**  FROM PENALITES  WHERE DELIVERERID IN (  SELECT DELIVERERID  FROM DELIVERERS  WHERE ville='Inglewood'  )  GROUP BY DELIVERERID  HAVING COUNT(DELIVERERID) >= 2; | (SUM(montant) AS TOTAL) calcule la somme totale des **montant**  Les fonction d’agrégation nécessite un groupe c’est pour sa on a ajouté GROUP BY DELIVERERID à la fin |
| SELECT nom, MIN(montant) AS Plus\_Petit\_Montant  FROM DELIVERERS, PENALITES  WHERE DELIVERERS.DELIVERERID = PENALITES.DELIVERERID  AND DELIVERERS.DELIVERERID IN (  SELECT DELIVERERID  FROM PENALITES  GROUP BY DELIVERERID  HAVING COUNT(DELIVERERID) BETWEEN 2 AND 4  )  GROUP BY nom; | Les deux tables sont combinées de manière implicite via une **jointure cartésienne**.  La condition de jointure est définie dans la clause WHERE |
| SELECT DELIVERERID, SUM(NUMDELIVERIES) AS TOTAL\_LIVRAISONS, SUM(NUMCOLLECTIONS) AS TOTAL\_COLLECTES  FROM COMPANYDEL  WHERE DELIVERERID IN (  SELECT DELIVERERID  FROM DELIVERERS  WHERE ville <> 'Stratford'  AND nom LIKE 'B%'  )  GROUP BY DELIVERERID; |  |

**SQL Scripts**

CREATE TABLE DELIVERERS (

DELIVERERID INT NOT NULL PRIMARY KEY,

nom VARCHAR(20) NOT NULL,

prenom VARCHAR(20) NOT NULL,

annee\_de\_naissance INT NOT NULL,

sexe CHAR(1) NOT NULL CHECK (sexe IN ('M','F')),

annee\_embauche INT,

rue VARCHAR(20) NOT NULL,

numero\_maison INT NOT NULL,

ville VARCHAR(20) NOT NULL,

code\_postal VARCHAR(20) NOT NULL,

numero\_tele VARCHAR(15) NOT NULL,

numero\_tele\_bureau VARCHAR(15)

);

CREATE TABLE COMPANIES (

CompanyID INT NOT NULL PRIMARY KEY,

MANDAT VARCHAR(10) NOT NULL,

DELIVERERID INT,

CONSTRAINT fk\_deliverer FOREIGN KEY (DELIVERERID) REFERENCES DELIVERERS (DELIVERERID)

);

CREATE TABLE COMPANYDEL (

DELIVERERID INT NOT NULL,

CompanyID INT NOT NULL,

NUMDELIVERIES INT NOT NULL,

NUMCOLLECTIONS INT NOT NULL,

CONSTRAINT C\_Deliver PRIMARY KEY (DELIVERERID, CompanyID),

CONSTRAINT CD\_Deliver CHECK (NUMDELIVERIES > 0 OR NUMCOLLECTIONS > 0),

CONSTRAINT fk\_deliverer1 FOREIGN KEY (DELIVERERID) REFERENCES DELIVERERS (DELIVERERID),

CONSTRAINT fk\_company FOREIGN KEY (CompanyID) REFERENCES COMPANIES (CompanyID)

);

CREATE TABLE PENALITES (

PENALTYID INT NOT NULL PRIMARY KEY,

date\_amende DATE NOT NULL,

montant INT NOT NULL,

DELIVERERID INT,

CONSTRAINT fk\_deliverer2 FOREIGN KEY (DELIVERERID) REFERENCES DELIVERERS (DELIVERERID)

);