Introduction

Les bases de données modernes supportent souvent le type `JSON`, permettant de stocker et manipuler des documents JSON directement dans les colonnes. SQL propose des fonctions dédiées pour extraire, manipuler et agréger les données JSON. Ce document présente les fonctions JSON principales utilisées dans SQL (notamment PostgreSQL, MySQL et SQL Server).

Note : La disponibilité et la syntaxe précise peuvent varier selon le SGBD.

Fonctions pour Extraire des Données JSON

1. `JSON_VALUE()`

Extrait une valeur scalaire (chaîne, nombre) d'un document JSON via un chemin JSONPath.

```
SELECT JSON_VALUE(data, '$.nom') AS nom
FROM clients;
```

Ici, `data` est une colonne JSON, on extrait la valeur associée à la clé `"nom"`.

2. `JSON_QUERY()`

Extrait un objet JSON ou un tableau JSON. Utile pour récupérer des structures plus complexes.

```
SELECT JSON_QUERY(data, '$.adresses') AS adresses
FROM clients;
```

Fonctions de Construction JSON

1. `JSON_OBJECT()`

Construit un objet JSON à partir de paires clé-valeur.

```
SELECT JSON_OBJECT('nom', nom, 'age', age) AS client_json
FROM clients;
```

Produit par exemple : `{"nom": "Dupont", "age": 30}`

2. `JSON_ARRAY()`

Crée un tableau JSON à partir de plusieurs valeurs.

```
SELECT JSON_ARRAY('Paris', 'Lyon', 'Marseille') AS villes;
```


Fonctions d'Agrégation JSON

1. JSON_AGG()

Agrège plusieurs lignes en un tableau JSON.

```
SELECT JSON_AGG(nom) AS noms_clients
FROM clients;
```

Crée un tableau JSON avec tous les noms des clients.

Fonctions spécifiques PostgreSQL

1. -> et ->> opérateurs

Extraient respectivement un objet JSON ou une valeur texte.

```
SELECT data->'nom' AS nom_json,
       data->>'nom' AS nom_texte
FROM clients;
```

2. jsonb_set()

Met à jour un élément dans un document JSONB.

```
UPDATE clients
SET data = jsonb_set(data, '{age}', '31')
WHERE id = 1;
```

Autres Fonctions Utiles

- `ISJSON()` — teste si une chaîne est un JSON valide (ex : SQL Server)
- `JSON_TYPE()` — retourne le type de valeur JSON (objet, tableau, etc.)
- `JSON_LENGTH()` — retourne la taille d'un tableau JSON

Exercice

Vous disposez d'une table `clients` contenant une colonne `data` de type JSON (ou JSONB) stockant des informations détaillées sur chaque client. Voici un exemple d'une valeur possible dans la colonne `data` :


```
{
  "nom": "Durand",
  "prenom": "Alice",
  "contacts": {
    "email": "alice.durand@example.com",
    "telephones": ["0123456789", "0987654321"]
  },
  "commandes": [
    {"id": 101, "date": "2024-01-10", "montant": 150.5},
    {"id": 102, "date": "2024-02-15", "montant": 89.9}
  ],
  "preferences": {
    "newsletter": true,
    "langue": "fr"
  }
}
```

Questions

1. Écrire une requête SQL pour extraire le prénom du client à partir de la colonne `data`.

   Indice : utilisez la fonction adaptée pour extraire une valeur scalaire d'un champ JSON.

2. Écrire une requête qui retourne tous les emails des clients.

   Indice : le champ email est dans un objet imbriqué `contacts`.

3. Écrire une requête qui liste toutes les commandes (id et montant) pour un client donné, par exemple où `nom = 'Durand'`.

   Indice : vous devez extraire un tableau JSON et décomposer ses éléments (penser aux fonctions JSON pour tableau ou jointure lateral si disponible).

4. Calculer le montant total des commandes pour chaque client.

   Indice : décomposer le tableau de commandes et agréger les montants.

5. Modifier le document JSON d'un client pour désactiver la newsletter (`"newsletter": false`).

   Indice : utilisez une fonction d'update JSON qui modifie un champ précis dans l'objet JSON.

6. Ajouter un nouveau téléphone `"0112233445"` dans le tableau `telephones` pour un client donné.

Indice : il faudra extraire le tableau, y ajouter l'élément, puis mettre à jour la colonne JSON.

Bonus

- Écrire une requête qui liste les clients ayant passé une commande d'un montant supérieur à 100.
- Extraire la langue préférée des clients et afficher `français` si c'est `"fr"` ou `anglais` si `"en"`.

