SQL

Programmation avancée

Guillaume Raschia — Nantes Université

Dernière mise-à-jour : 27 novembre 2023

Plan de la session

Les super-pouvoirs de SQL

Amuse-gueules

Tables temporaires

Fenêtrage

Cas d’usage pour l’analyse

Les super-pouvoirs de SQL

SQL en (très) bref

langage de domaine : manipulations élémentaires de données

langage déclaratif : fondé sur la logique du premier ordre

langage standardisé : indépendant des systèmes et des applications

La plus célèbre des phrases

$$\texttt{SELECT } A \texttt{ FROM } R \texttt{ WHERE } c$$

Éléments du langage SQL

La base : désigner la source (FROM) filtrer (WHERE) et collecter (SELECT)

- lier : JOIN
- réaliser des opérations ensemblistes : UNION, INTERSECT, EXCEPT
- trier : ORDER BY
- agréger : COUNT, SUM, MIN, MAX, AVG
- grouper : GROUP BY - HAVING
- enrichir le filtre : IN, EXISTS, ANY, ALL, avec des requêtes enchâssées

Préséance des clauses SQL

1. FROM, JOIN
2. WHERE
3. GROUP BY
4. fonctions d'agrégation
5. HAVING
6. fonctions fenêtrées
7. SELECT
8. DISTINCT
9. UNION/INTERSECT/EXCEPT
10. ORDER BY
11. OFFSET
12. LIMIT/FETCH/TOP

Détermine un ordre d'évaluation au sein d'une requête

SQL, la préhistoire

Un récit proposé par D. Poulain

- 1968 : **David L. Childs** (Univ. Michigan). *Feasability of a Set-Theoretic Data Structure*. Toute question peut être réduite à 3 fonctions : « sélection », « relation », « regroupement ».
- 1970 : **Edgar F. Codd**, IBM (PhD Univ. Michigan). *A Relational Model of Data for Large Shared Data Banks*. Naissance du modèle relationel et esquisse d'un langage universel « Universal Data Sublanguage », celui vraisemblablement rêvé par Childs.

SQL, la génèse

- 1974 : **Donald D. Chamberlin** et **Raymond F. Boyce** (IBM). *Sequel : a Structured English QUEry Language* pour System/R de IBM. Première apparition de SELECT/WHERE, FROM et GROUP BY.
- 1975 : **D. Chamberlin**, **Jim N. Gray** et **Irving L. Traiger** (IBM). *Views, Authorization, and Locking in a Relational Data Base System*. Complément INSERT, UPDATE, DELETE, GRANT, REVOKE, VIEW.
- 1976 : **D. Chamberlin** et al. (IBM) réalisent la synthèse : *SEQUEL-2 : A Unified Approach to Data Definition, Manipulation and Control*.

À cette époque IBM mise sur les gros ordinateurs (mainframe). System/R et Sequel (et Codd et Chamberlin) ne sont pas des projets prioritaires...

SQL, un ami quinquagénère

- 1979 : **Larry Ellison** (Relational Software Inc.) commercialise la première version de SQL[1] (**Structured Query Language**) dans un tout nouveau système baptisé Oracle V2 (la V1 est une pré-version de 1978).

Tous les acteurs du marché (IBM System R puis DB2, Teradata, Informix, Sybase, Ingres, Postgres) émergent dans les années 80 et adoptent SQL comme langage universel.

1. SEQUEL est devenu SQL pour des questions de marque déposée.

Les évolutions de SQL

1986	ISO/CEI 9075:1986	SQL-86 ou SQL-87	Édité par l'ANSI puis adopté par l'ISO en 1987.
1989	ISO/CEI 9075:1989	SQL-89 ou SQL-1	Révision mineure.
1992	ISO/CEI 9075:1992	SQL-92 alias SQL2	Révision majeure.
1999	ISO/CEI 9075:1999	SQL-99 alias SQL3	Expressions rationnelles, requêtes récursives, déclencheurs, types non-scalaires et quelques fonctions orientées objet.

Les évolutions de SQL (suite)

2003	ISO/CEI 9075:2003	SQL:2003	Introduction de fonctions pour la manipulation XML, « fonctions fenêtrées », ordres standardisés et colonnes avec valeurs auto-produites.
2008	ISO/CEI 9075:2008	SQL:2008	Ajout de quelques fonctions fenêtrées (ntile, lead, lag, first value, last value, nth value), limitation du nombre de lignes (OFFSET / FETCH), amélioration mineure sur les types distincts, curseurs et mécanismes d'auto-incrémentation.
2011	ISO/CEI 9075:2011	SQL:2011	Ajout du support des tables temporelles (historisation automatique).

Longévité et plébiscite

Classement des technologies populaires, StackOverflow 2023 Developer Survey

...et qu'on adore détester.

Incontournable pour un ingénieur/analyste des données.

Du langage de domaine au langage générique

Conçu pour manipuler des données, SQL est devenu Turing-complet par accident

comme HTML5/CSS3, les templates C++, Excel, Minecraft ou…les cartes Magic !

Les ingrédients clés :

- « fonctions fenêtrées » et
- CTE récursives

YT : Créer des fractales avec SQL, Michael Malis

Tobias Burghardt, Denis Hirn, Torsten Grust. *Functional Programming on Top of SQL Engines*, Proc. of PADL'22, pp 59–78 (2022).

Ensemble de Mandelbrot

Exemple de requête récursive, tirée de la documentation officielle de SQLite

```
WITH RECURSIVE
  xaxis(x) AS (VALUES(-2.0) UNION ALL SELECT x+0.05 FROM xaxis WHERE x<1.2),
  yaxis(y) AS (VALUES(-1.0) UNION ALL SELECT y+0.1 FROM yaxis WHERE y<1.0),
  m(iter, cx, cy, x, y) AS (
    SELECT 0, x, y, 0.0, 0.0 FROM xaxis, yaxis
    UNION ALL
    SELECT iter+1, cx, cy, x*x-y*y + cx, 2.0*x*y + cy FROM m
     WHERE (x*x + y*y) < 4.0 AND iter<28
  ),
  m2(iter, cx, cy) AS (
    SELECT max(iter), cx, cy FROM m GROUP BY cx, cy
  ),
  a(t) AS (
    SELECT group_concat( substr(' .+*#', 1+min(iter/7,4), 1), '')
    FROM m2 GROUP BY cy
  )
SELECT group_concat(rtrim(t),x'0a') FROM a;
```

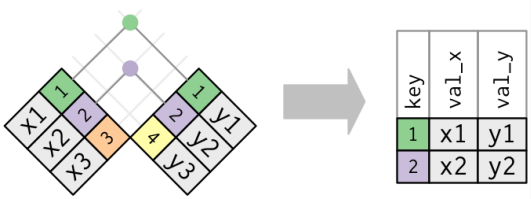
à tester de toute urgence !

Amuse-gueules

Rappel : la jointure $\bowtie$

Opération binaire qui associe les nuplets d'une table R avec ceux d'une table S, sur un critère de comparaison θ entre une colonne A de R et une colonne B de S.

$$R \underset{R.A\ \theta\ S.B}{\bowtie} S$$



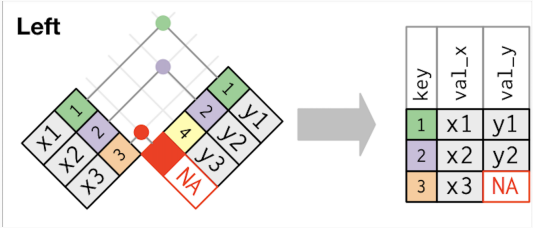

```
SELECT <OUTPUT SCHEMA>
FROM TABLE_R R
JOIN TABLE_S S ON R.A = S.B
```

INNER JOIN ou simplement JOIN en SQL

Diagramme « de correspondance », R for Data Science (alt. diagramme de Venn)

La semi-jointure est obtenue par projection dans la clause SELECT

La jointure externe en SQL



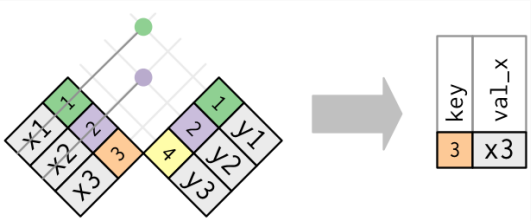

```
SELECT <OUTPUT SCHEMA>
FROM TABLE_R R
LEFT JOIN TABLE_S S ON R.A = S.B
```

Alternatives : FULL (OUTER) ~~et RIGHT~~

Formes complémentaires : CROSS JOIN ($\times$) et NATURAL JOIN

L’anti-jointure

Les valeurs d’une table qui ne sont pas associées à celles d’une autre table.




```
SELECT <OUTPUT SCHEMA>
FROM TABLE_R R
LEFT JOIN TABLE_S S ON R.A = S.B
WHERE S.B IS NULL;
```

Principe : jointure externe à gauche, avec un filtre des valeurs à NULL

Exemple

Bar (nom, adresse) Biere (nom, brasserie) Carte (bar, biere, prix)

Requête : *les bières proposées dans aucun bar.*

```
SELECT b.* FROM Biere b
LEFT JOIN Carte c ON b.nom = c.biere
WHERE c.biere IS NULL;
```

Anti-jointure vs. EXCEPT

Les bières proposées dans aucun bar.

```
SELECT * FROM Biere b
EXCEPT
SELECT b.* FROM Carte c JOIN Biere b ON c.biere = b.nom;
```

EXCEPT rend nécessaire une jointure pour collecter les attributs manquants.

Groupement et agrégation

GROUP BY... HAVING COUNT, MIN, MAX, SUM, AVG

```
count(*)  -- compte toutes les valeurs, dont les nulls
count(id) -- compte toutes les valeurs de la colonne id, sauf les nulls
count(1)  -- identique à count(*)
```

Les bars dont le prix moyen est supérieur à 3€.

```
SELECT bar, AVG(prix) as prix_moy FROM Carte
GROUP BY bar
HAVING AVG(prix) > 3.0
```

Mécanismes incontournables pour le « profilage » des données

Une tranche du résultat

LIMIT et OFFSET

Les 5 bars dont le prix moyen est immédiatement supérieur à 3€, à l'exclusion du moins cher.

```sql
SELECT bar, AVG(prix) as prix_moy FROM Carte
GROUP BY bar
HAVING AVG(prix) > 3.0
ORDER BY AVG(prix) ASC
LIMIT 5
OFFSET 1;
```

✎ Qu'en est-il des ex aequo ? (5 et 6e par exemple)

Fonctions SQL natives

Maths, sur les nombres

$+ - * /$ ABS COS EXP LN MOD RAND ROUND SQRT…

Texte, sur les chaînes de caractères

LENGTH LOWER LPAD CONCAT REGEXP REPLACE SUBSTRING…

Temps, sur les dates et les heures

NOW DAY MONTH YEAR DATEDIFF TO_DATE…

Des variations importantes d'un système à l'autre $\Rightarrow$ consulter la documentation !

Valeurs à NULL et valeurs par défaut

Test à NULL pour une valeur : A IS NULL

Test à NULL pour une expression :

```
-- /!\ SQL non conforme: il manque la clause FROM
SELECT ISNULL(1,0);        -- retourne 1
SELECT ISNULL(NULL,10);    -- retourne 10
SELECT ISNULL(1/0,10);     -- retourne 10
SELECT ISNULL(1/0,'oui'); -- retourne 'oui'
SELECT ISNULL(1/0);        -- retourne true
```

Rappel : SQL admet une logique tri-valuée : 1 (true), 0 (false) et 0,5 (unknown)

- (NULL = 1) $\Leftrightarrow$ (NULL > 1) $\Leftrightarrow$ (NULL <> 1) $\Leftrightarrow$ (NULL = NULL) $\Leftrightarrow$ unknown !

ISNULL s'appelle COALESCE dans certains systèmes

Expressions conditionnelles

CASE...WHEN

```
SELECT
    bar,
    AVG(prix) AS prix_moy,
    CASE
        WHEN AVG(prix) < 2.5 THEN 'yeah!'
        WHEN AVG(prix) < 5.0 THEN 'hum'
        ELSE 'wtf?!'
    END AS frequentable
FROM Carte
GROUP BY bar;
```

Variante du CASE... WHEN

```
SELECT *,
    CASE nom
        WHEN 'Bud' THEN '*'
        WHEN 'Titan' THEN '***'
        ELSE '**'
    END AS j_aime
FROM Biere;
```

CASE...WHEN s'emploie partout :

```
SELECT ... ORDER BY (CASE WHEN A IS NULL THEN B ELSE A END);
SELECT ... WHERE CASE WHEN x <> 0 THEN y/x > 1.5 ELSE false END;
```

CASE... WHEN pour pivoter une table

Usage fréquent en analyse de données

Pivoter les lignes vers des colonnes :

```
SELECT bar
    SUM(CASE WHEN biere = 'Bud' THEN prix END) AS prixBud,
    SUM(CASE WHEN biere = 'Grim' THEN prix END) AS prixGrim,
    SUM(CASE WHEN biere = 'Titan' THEN prix END) AS prixTitan,
    AVG(CASE WHEN biere NOT IN ('Bud','Grim','Titan')
         THEN prix ELSE 0 END) AS prixMoyenAutre,
FROM Carte GROUP BY bar
```

À retenir

1. L'anti-jointure est un pattern alternatif à EXCEPT
2. Le groupement et l'agrégation sont votre seconde nature !
3. Le catalogue de fonctions sur les types natifs est très riche
4. LIMIT et OFFSET simulent des requêtes top-k
   - performances médiocres
   - approximation du résultat (problème des ex aequo)
5. CASE…WHEN introduit les expressions conditionnelles

Tables temporaires

« Common Table Expression » ou CTE

Une CTE est déclarée à l'aide de la clause WITH

Elle permet de construire une table temporaire en préambule d'une requête

```
WITH bo_bar (nom, avg_prix)
AS (
    SELECT bar, AVG(prix) FROM Carte
    GROUP BY bar HAVING AVG(prix < 2.5) )
SELECT adresse FROM bar NATURAL JOIN bo_bar;
```

Décomposition de la requête et réutilisation de résultats intermédiaires

Requête avec plusieurs CTE

```
WITH bo_bar (nom, avg_prix)
AS (
    SELECT bar, AVG(prix) FROM Carte
    GROUP BY bar HAVING AVG(prix < 2.5) ),
     bonne_biere (nom, brasserie)
AS (
    SELECT * FROM Biere
    WHERE brasserie LIKE 'Bouffay' )
SELECT bba.nom FROM bo_bar bba
JOIN Carte c ON bba.nom = c.bar
JOIN bonne_biere bbi ON bbi.nom = c.bonne_biere;
```

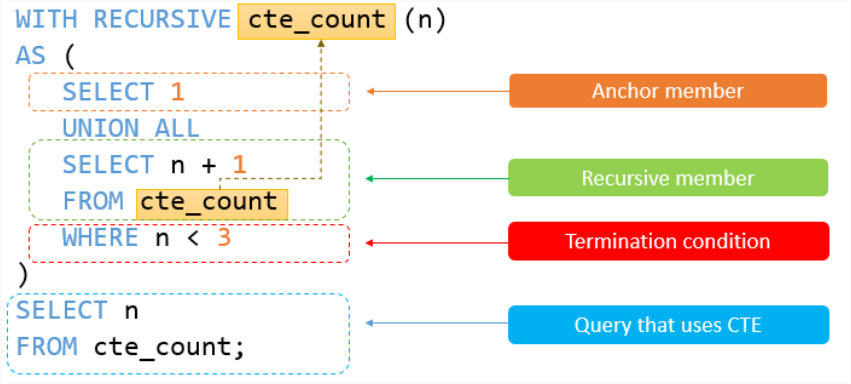
Les CTE peuvent également être corrélées : l'une référence l'autre, dans l'ordre de leur déclaration

CTE pour l'agrégation d'un attribut calculé

On veut savoir *combien de bars proposent le même nombre de bières à la vente.*

```
WITH bar_biche (nom, nbre_bieres)
AS (
    SELECT bar, COUNT(*) FROM Carte
    GROUP BY bar )
SELECT nbre_bieres, count(nom) as nbre_bars
FROM bar_biche
GROUP BY nbre_bieres;
```

Les CTE récursives



Source : A Definitive Guide To MySQL Recursive CTE

Création d'un compteur…éventuellement infini…

Pas très spectaculaire, mais essentiel

Limite de l'algèbre relationnelle

Encodage d'une relation binaire : H (id, par)

Modélise une structure d'arbre

- Les enfants de 42 ? ..... $\pi_{\texttt{id}}(\sigma_{\texttt{par}=42}(H))$
- Les petits-enfants de 42 ? ..... $\pi_{H_1.\texttt{id}}(\sigma_{H_2.\texttt{par}=42}(H_1 \underset{H_1.\texttt{par}=H_2.\texttt{id}}{\bowtie} H_2))$
- Les arrière-petits-enfants de 42 ? ..... $H_1 \bowtie H_2 \bowtie H_3$
- La $n$ième génération après 42 ? ..... $H_1 \bowtie H_2 \bowtie \ldots \bowtie H_{n-1}$

Comment atteindre « la dernière génération de 42 » ?! (sans connaître n a priori)

Au-delà de l'algèbre relationnelle

La récursivité au secours des requêtes de connectivité et de fermeture transitive

```
WITH fermeture_transitive (id, ancetre)
AS (
    SELECT * FROM H
    UNION
    SELECT ft.id, H.par FROM fermeture_transitive ft
    JOIN H ON ft.ancetre = H.id )
SELECT id from fermeture_transitive
WHERE ancetre = 42;
```

Retourne tout le sous-arbre de 42.

Les champs d'application

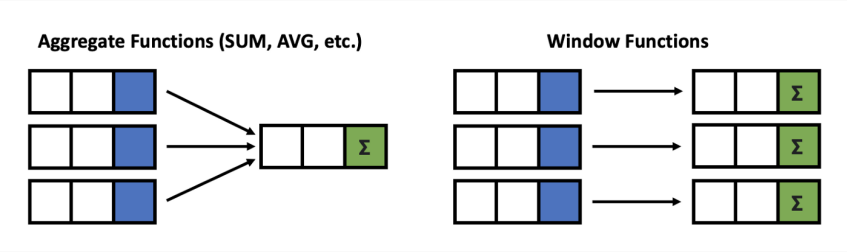
Les graphes (réseaux) et les arbres (hiérarchies)

- réseaux sociaux
- réseaux de transport : routier, ferrovière, aérien, etc.
- réseaux naturels : hydrauliques, biologiques, etc.
- réseaux d'infrastructures : distribution d'eau, d'électricité, internet, etc.
- organigrames d'entreprise
- systèmes de fichiers
- généalogie, phylogénétique
- etc.

Et partout où l'on a besoin d'une boucle

Fenêtrage

Agrégation simple vs. fenêtrage



Autrice : Julia Kho, Medium

Exemple

Agrégation

bière	prix_moyen
Bud	2,5
Grim	3,2
Titan	3,7

Fenêtrage

bière	prix	prix_moyen
Bud	2,2	2,5
Bud	2,7	2,5
Grim	3,0	3,2
Grim	3,4	3,2
Grim	3,2	3,2
Titan	4,2	3,7
Titan	3,2	3,7

OVER... (PARTITON BY...) : ma première requête à fenêtres

```
SELECT  biere,
        prix,
        AVG(prix) OVER (PARTITION BY biere) AS prix_moyen
FROM Carte
```

- OVER marque la déclaration d'un fenêtrage
- AVG joue le rôle de la fonction fenêtrée (« window function »)
- PARTITION BY définit le critère de groupement

Le fenêtrage est autorisé dans les clauses SELECT et ORDER BY.

OVER... (ORDER BY...) : une seule fenêtre triée

```
SELECT  biere,
        prix,
        SUM(prix) OVER (ORDER BY prix DESC) AS somme_cumulee,
        ROW_NUMBER() OVER (ORDER BY prix DESC) AS num_ligne,
        RANK() OVER (ORDER BY prix DESC) AS rang,
        DENSE_RANK() OVER (ORDER BY prix DESC) AS rang_dense
FROM Carte ORDER BY prix DESC, biere ASC;
```

biere	prix	somme_cumulée	num_ligne	rang	rang_dense
Titan	4,2	4,2	1	1	1
Grim	3,4	7,6	2	2	2
Grim	3,2	14,0	4	3	3
Titan	3,2	10,8	3	3	3
Grim	3,0	17,0	5	5	4
...	...	...	...	...	...

OVER... (PARTITION BY... ORDER BY...)

le combo – quasi – ultime !

```
SELECT  biere,
        prix,
        prix - LAG(prix, 1, prix)
            OVER (PARTITION BY biere ORDER BY prix ASC)
            AS variation
FROM Carte ORDER BY biere ASC;
```

biere	prix	variation
Bud	2,2	0,0
Bud	2,7	0,5
Grim	3,0	0,0
Grim	3,2	0,2
Grim	3,4	0,2
...	...	...

La liste des « fonctions fenêtrées »

Fonctions d'agrégation

Catalogue usuel : AVG(.) MIN(.) MAX(.) SUM(.) COUNT(.)

Fonctions d'ordre

Couplées à ORDER BY. Elles s'emploient sans argument

ROW_NUMBER() RANK() DENSE_RANK() PERCENT_RANK() NTILE(n)

Fonctions de sélection

Permettent d'associer une valeur spécifique à la valeur courante

LAG(.) LEAD(.) FIRST_VALUE(.) LAST_VALUE(.) NTH_VALUE(.,n)

La portée des fenêtres

Par défaut

Du début de la fenêtre au nuplet courant

```
ROWS BETWEEN UNBOUNDED PRECEDING AND CURRENT ROW
```

Existe aussi

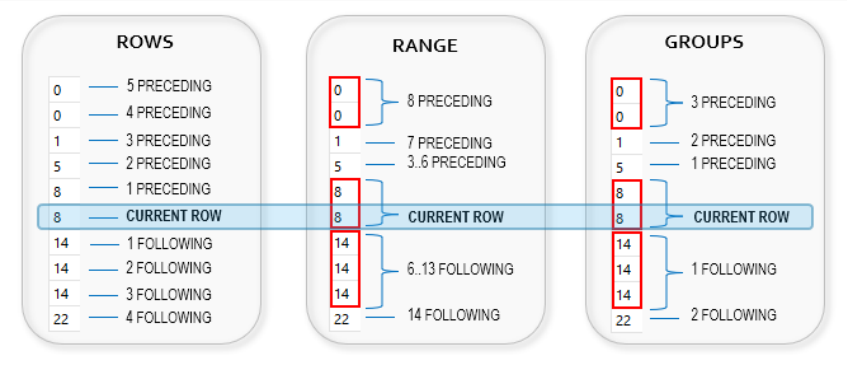
```
-- nuplet courant jusqu'à la fin de la fenêtre :
ROWS BETWEEN CURRENT ROW AND UNBOUNDED FOLLOWING
-- du début à la fin de la fenêtre :
RANGE BETWEEN UNBOUNDED PRECEDING AND UNBOUNDED FOLLOWING
```

La portée des fenêtres : variantes

ROWS vs. RANGE vs. GROUPS

```
SELECT bière,
   SUM(prix) OVER (ORDER BY prix ROWS 3 PRECEDING) AS "ROWS",
   SUM(prix) OVER (ORDER BY prix RANGE 3 PRECEDING) AS "RANGE",
   SUM(prix) OVER (ORDER BY prix GROUPS 3 PRECEDING) AS "GROUPS"
FROM Carte
ORDER BY 1 -- mode ordinal: équivalent à ORDER BY bière
```

La portée des fenêtres : récapitulatif



Source : Christian Scutaru, Data Xtractor

À retenir

1. OVER signale la construction d'un fenêtrage [2]
   - majoritairement employé dans une clause SELECT
   - mais également dans un ORDER BY
2. Plusieurs fenêtrages peuvent coexister dans la même requête
3. PARTITION BY détermine le groupement en fenêtres
4. ORDER BY spécifie l'ordre dans lequel les nuplets sont traités au sein de chaque fenêtre

2. ✎ Essayer OVER().

À retenir, aussi

5. Chaque fenêtre triée par ORDER BY dispose d'une portée :
   - du début de la fenêtre au nuplet courant, par défaut
   - spécifiée précisément à l'aide de ROWS, RANGE ou GROUPS, combinés avec PRECEDING et FOLLOWING
6. Sans la clause ORDER BY, l'intégralité des nuplets d'une fenêtre est prise en compte dans le calcul
7. Les fonctions fenêtrées agrègent, classent ou sélectionnent les nuplets

Cas d’usage pour l’analyse

Quelques cas d'usage

Programmation SQL pour les ingénieur·e·s data

publiés par Nate Rosidi, StrataScratch, le 05 avril 2022 sur KDnuggets

Cas numéro 1

Pour un job chez Meta

Medium | Meta/Facebook | Interview Questions | ID 10295 | Interview Question Date: Nov 2020

Most Active Users On Messenger

Meta/Facebook Messenger stores the number of messages between users in a table named 'fb_messages'. In this table 'user1' is the sender, 'user2' is the receiver, and 'msg_count' is the number of messages exchanged between them.
Find the top 10 most active users on Meta/Facebook Messenger by counting their total number of messages sent and received. Your solution should output usernames and the count of the total messages they sent or received

Table: fb_messages

Approach Hints | Expected Output

Cas numéro 2

Pour un job chez Netflix

Hard Netflix Interview Questions ID 10303

Interview Question Date: Nov 2020

Top Percentile Fraud

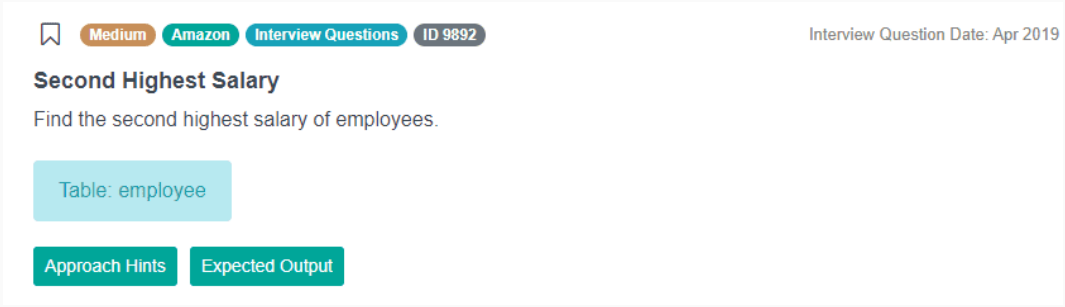
ABC Corp is a mid-sized insurer in the US and in the recent past their fraudulent claims have increased significantly for their personal auto insurance portfolio. They have developed a ML based predictive model to identify propensity of fraudulent claims. Now, they assign highly experienced claim adjusters for top 5 percentile of claims identified by the model.
Your objective is to identify the top 5 percentile of claims from each state. Your output should be policy number, state, claim cost, and fraud score.

Table: fraud_score

Approach Hints | Expected Output

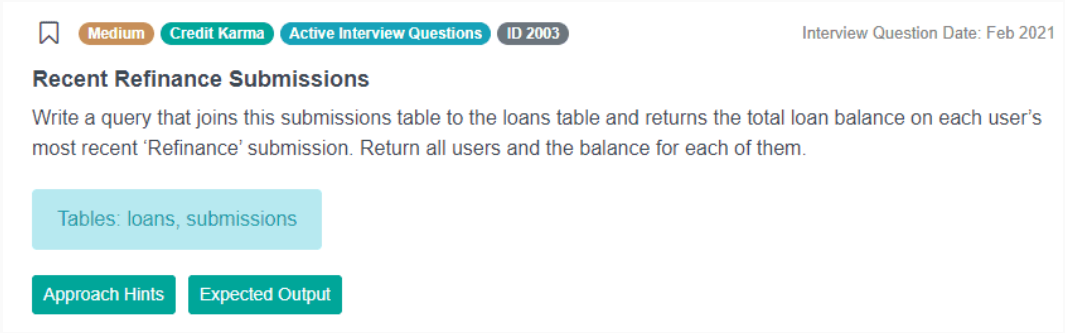
Cas numéro 3

Pour un job chez Amazon



Cas numéro 4

Pour un job chez Credit Karma (random banque/assurance)



Cas numéro 5

Pour un second job chez Amazon

Hard Amazon Interview Questions ID 10319

Interview Question Date: Dec 2020

Monthly Percentage Difference

Given a table of purchases by date, calculate the month-over-month percentage change in revenue. The output should include the year-month date (YYYY-MM) and percentage change, rounded to the 2nd decimal point, and sorted from the beginning of the year to the end of the year.
The percentage change column will be populated from the 2nd month forward and can be calculated as ((this month's revenue - last month's revenue) / last month's revenue)*100.

Table: sf_transactions

Approach Hints Expected Output

En guise de conclusion

SQL est un langage universel et complet

Parfait pour débuter toute analyse

Parfait pour formuler des requêtes décisionnelles complexes

Parfait pour décrocher un job !