



- L3 MIASHS/Ingémath
- Université Paris Cité
- Année 2023-2024
- Course Homepage
- Moodle

Créer les fonctions SQL et vues correspondant aux questions suivantes.

Les questions portent sur le schéma `nycflights13` issu de <https://github.com/tidyverse/nycflights13>.

Voir [Documentation package R nycflights13](#)

Exercice

- Nombre vols retardés d'un délai minimal à l'arrivée à un aéroport donné
- Donnés sur schéma `nycflights13`

Écrire une fonction SQL nommée `cc_fonc_11` qui prend en argument un code d'aéroport `p_faa` de type `text`, une année `p_year` (`int4`), un mois `p_month` (`int4`), un délai plancher `p_delay` (`int4`) et renvoie le nombre de vols retardés de strictement plus de `p_delay` minutes à l'arrivée à l'aéroport `faa`, pendant le mois `month` de l'année `year`.

La fonction retourne un entier (`bigint`)

💡 Réponse attendue pour

```
SELECT *
FROM cc_fonc_11('LAX', 2013, 3, 40);
```

73

💡 solution

```
CREATE OR REPLACE FUNCTION correction_cc3.cc_fonc_11(p_faa TEXT,
                                                     p_year int4,
                                                     p_month int4,
                                                     p_delay int4)
RETURNS bigint
LANGUAGE sql
AS $function$
SELECT COUNT(*) as n
FROM nycflights13.flights f
WHERE f.arr_delay > p_delay AND
      f.dest = p_faa AND
      f."year" = p_year AND
      f."month" = p_month;
$function$
```

Exercice

Donnés sur schéma `nycflights`.

Écrire une fonction SQL nommée `cc_fonc_12` qui prend en argument une année `p_year`, un mois `p_month`, une compagnie aérienne `p_carrier` (`text`) et renvoie le nombre de couples aéroport d'origine `origin`/aéroports de destination `dest` distincts desservis par cette compagnie `p_carrier` pendant le mois `p_month` de l'année `p_year`.

La fonction retourne un entier (`bigint`)

Réponse attendue pour `SELECT * FROM cc_fonc_12('UA', 2013, 2);`

39

 solution

```
CREATE OR REPLACE FUNCTION correction_cc3.cc_fonc_12(p_carrier TEXT, p_year int4, p_month int4
    RETURNS bigint
    LANGUAGE sql
AS $function$
SELECT COUNT(DISTINCT (origin, dest)) AS n
FROM nycflights13.flights f
WHERE f.carrier =p_carrier AND
      f.year=p_year AND
      f.month=p_month ;
$function$
```

Exercice

Données dans le schéma nycflights13

Écrire une vue nommée cc_vue_13 de schema (origin, dest, year, month, carrier, model, maxspeed) qui indique pour chaque couple (origin, dest), pour chaque couple (year,month), la vitesse maximale maxspeed (numeric) d'un vol reliant origin à dest pendant le mois month de l'année year, le modèle de l'avion qui a réalisé le vol le plus rapide, et la compagnie aérienne qui a assuré ce vol.

maxspeed sera exprimée en km/h. distance est exprimée en milles nautiques (1 mille = 1.852 km).

Écrire une vue nommée cc_vue_13 de schéma :

(year INT4, month INT4, origin TEXT, dest TEXT, name TEXT, model TEXT, maxspeed numeric)

Réponse attendue pour

```
SELECT *
FROM cc_vue_13
WHERE dest='LAX' AND month=4 ;
```

year	month	origin	dest	name	model	maxspeed
2013	4	EWR	LAX	United Air Lines Inc.	A320-232	921.613333333335
2013	4	JFK	LAX	United Air Lines Inc.	757-222	951.9362068965518

Précision pour maxspeed : (1 km/h)

 solution

```
CREATE MATERIALIZED VIEW correction_cc3.cc_vue_13 AS (
WITH r AS (
    SELECT f.year, f.month, f.origin, f.dest, max(1.852 * 60*f.distance/f.air_time) as maxspeed
    FROM nycflights13.flights f
    WHERE f.distance > 0 AND f.air_time > 0
    GROUP BY f.year, f.month, f.origin, f.dest
), s AS (
    SELECT r.year, r.month, r.origin, r.dest, r.maxspeed, ff.tailnum, ff.carrier
    FROM nycflights13.flights ff JOIN r ON (r.month=ff.month AND r.year=ff.year AND r.origin=ff.origin AND r.dest=ff.dest)
    WHERE ff.distance > 0 AND ff.air_time > 0 AND (1.852 * 60* ff.distance/ff.air_time) >= r.maxspeed
)
SELECT s.year, s.month, s.origin, s.dest, a.name, p.model ,s.maxspeed
FROM s
JOIN nycflights13.panes p ON (s.tailnum=p.tailnum)
JOIN nycflights13.airlines a ON (s.carrier=a.carrier))
WITH DATA ;
```

Exercice

Donnés sur schéma nycflights13.

Créer dans votre schéma, une vue nommée cc_vue_14 de schéma :

```
(origin , year int4, month int4, day int4, hour int4, avg_depdelay bigint, n_cancelled bigint, n_scheduled bigint)
```

qui donne pour chaque aéroport d'origine `origin`, chaque heure `yyyy:mm:dd hh:00:00`, le retard moyen au départ `avg_depdelay` des vols qui ont (effectivement) décollé de `origin` pendant l'heure qui a précédé `yyyy:mm:dd hh:00:00`, et `n_cancelled` le nombre de vols annulés sur cet aéroport pendant cette heure, et enfin `n_scheduled_flights` le nombre de vols prévus pendant cette heure.

Réponse attendue pour :

```
SELECT date_time,
       round(avg_delay::numeric, 1) as avg_delay,
       n_cancelled,
       n_scheduled_flights
FROM cc_vue_14
WHERE origin='JFK' AND
      n_cancelled > .25 * n_scheduled_flights
ORDER BY date_time
LIMIT 10 ;
```

date_time	avg_delay	n_cancelled	n_scheduled_flights
2013-01-30 20:00:00.000	30.7	7	23
2013-02-08 12:00:00.000	5.8	5	11
2013-02-08 15:00:00.000	25.9	7	17
2013-02-08 16:00:00.000	10.6	17	24
2013-02-08 17:00:00.000	6.7	19	25
2013-02-08 18:00:00.000		24	24
2013-02-08 19:00:00.000		24	24
2013-02-08 20:00:00.000		23	23
2013-02-08 21:00:00.000		17	17
2013-02-08 22:00:00.000		6	6

solution

```
CREATE MATERIALIZED VIEW correction_cc3.cc_vue_14 AS (
  SELECT f.origin,
    f."year",
    f."month",
    f."day",
    f."hour"+ 1 AS "hour",
    AVG(dep_delay) AS avg_delay,
    SUM(CASE WHEN f.arr_time IS NULL THEN 1 ELSE 0 END) AS n_cancelled,
    COUNT(*) as n_scheduled_flights
  FROM nycflights13.flights f
  GROUP BY f.origin, f.year, f.month, f."day", f."hour")
  WITH DATA ;
```

Exercice

Donnés sur schéma nycflights13.

Créer une vue cc_vue_15 de schéma :

```
(tailnum text, year int4, woy int4, cumdist numeric, model text)
```

qui recense pour chaque semaine (commençant le dimanche) les avions (identifiés par tailnum) qui ont parcouru la plus grande distance durant cette semaine, la distance parcourue pendant la semaine (en **milles nautiques**), on indiquera aussi le modèle (model) de l'avion

Réponse attendue pour :

```
SELECT *
FROM cc_vue_15
WHERE woy>=26 AND woy <= 30
```

```
+-----+-----+-----+-----+
|tailnum|year|woy|cumdist|model   |
+-----+-----+-----+-----+
|N320AA |2013| 30|22497.0|767-223|
|N324AA |2013| 28|22497.0|767-223|
|N327AA |2013| 26|23697.0|767-223|
|N327AA |2013| 27|22386.0|767-223|
|N332AA |2013| 29|23475.0|767-223|
```

Indications

Manipulation du temps

- [Doc PostgreSQL : types](#)
- [Doc PostgreSQL : fonctions](#)

```
SELECT extract(week from now()), now(), now() + '1 week'::interval ;
```

```
+-----+-----+-----+
|date_part|now           |?column?          |
+-----+-----+-----+
|      46.0|2022-11-20 11:46:59.322 +0100|2022-11-27 11:46:59.322 +0100|
```

Voir aussi `datetime` pour Python ou `lubridate` pour R.

💡 solution

```
CREATE MATERIALIZED VIEW correction_cc3.cc_vue_15 AS (
    WITH r AS (
        SELECT f.tailnum,
            f.year,
            extract(week from f.time_hour)::int4 as woy,
            sum(distance) as cumdist
        FROM nycflights13.flights f
        WHERE tailnum is not null
        GROUP BY f.year, extract(week from f.time_hour), f.tailnum
    ),
    s AS (
        SELECT DISTINCT r.tailnum,
            r.year,
            r.woy,
            r.cumdist,
            rank() OVER v AS rnk
        FROM r
        WINDOW v AS (PARTITION BY r.year, r.woy)
                    ORDER BY cumdist DESC NULLS LAST),
    t AS (
        SELECT s.tailnum, s.year, s.woy, s.cumdist
        FROM s
        WHERE rnk = 1)

        SELECT t.*, p.model
        FROM t JOIN nycflights13.planes p ON (t.tailnum=p.tailnum)
    )
    WITH DATA ;
```