Bases de Données Relationnelles
TP3
MAM4 - SI3

La base de données du TP3 est composée des tables décrites en Annexe

1 Recursivité

Cette section n'utilise que la table employees.

1. Ecrire une requête qui affiche tous les couples (employee,manager) où manager est l'identifiant d'un chef direct ou indirect de l'employé d'identifiant employee.

employee	*manager*
101	100
102	100
103	102
103	100
...	.....
121	100
122	100
206	100
206	101
206	205

(74 rows)

```
WITH RECURSIVE hierarchie(soldat, general) AS (
    SELECT employees.employee_id, employees.manager_id
    FROM employees
    WHERE manager_id IS NOT NULL
    UNION
    SELECT employees.employee_id, hierarchie.general
    FROM employees JOIN hierarchie
        ON (employees.manager_id= hierarchie.soldat) )
SELECT soldat AS employee, general AS manager
FROM hierarchie
ORDER BY soldat;
```

---

2. Ecrire une requête qui affiche tous les couples (employee, manager) où manager est le nom d'un chef direct ou indirect de l'employé de nom employee.

employee	*manager*
Kochhar	King
De Haan	King
Hunold	De Haan
Hunold	King
...	.....
Gietz	King
Gietz	Kochhar
Gietz	Higgins

(74 rows)

```
WITH RECURSIVE hierarchie(soldat, general) AS (
    SELECT employees.employee_id, employees.manager_id
    FROM employees WHERE manager_id IS NOT NULL
    UNION
    SELECT employees.employee_id, hierarchie.general
    FROM employees JOIN hierarchie
        ON (employees.manager_id= hierarchie.soldat) )
SELECT S.last_name AS employee, G.last_name AS manager
FROM hierarchie
    JOIN employees S ON (S.employee_id = hierarchie.soldat)
    JOIN employees G ON (G.employee_id = hierarchie.general)
ORDER BY soldat;
```

---

3. Écrire une requête qui affiche tous les triplets (employee, manager, distance) où manager est le nom d'un chef direct ou indirect de l'employé de nom employee et distance est la distance hiérarchique entre les deux (1 pour un chef direct, 2 pour le chef direct du chef direct, etc.).

employee	*manager*	*distance*
Kochhar	King	1
De Haan	King	1
Hunold	De Haan	1
....	.....	.....
Gietz	Higgins	1
Gietz	Kochhar	2
Gietz	King	3

(74 rows)

```
WITH RECURSIVE hierarchie(soldat, general, distance) AS (
    SELECT employees.employee_id, employees.manager_id, 1
    FROM employees where manager_id IS NOT NULL
    UNION
    SELECT employees.employee_id, hierarchie.general, hierarchie.distance+1
    FROM employees JOIN hierarchie
        ON (employees.manager_id=hierarchie.soldat))
SELECT S.last_name AS employee, G.last_name AS manager, distance
FROM hierarchie
    JOIN employees S ON (S.employee_id=hierarchie.soldat)
    JOIN employees G ON (G.employee_id=hierarchie.general)
ORDER BY soldat, distance;
```

---

4. Écrire une requête qui affiche pour tous les employés leur identifiant, leur nom, leur distance au PDG, le nom de tous leurs chefs directs ou indirects, les identifiants de tous leurs chefs directs ou indirects.

   Utilisez la fonction STRING_AGG (consultez la doc) qui agrège en les concaténant des chaînes de caractères, et comme des entiers ne sont pas des chaînes de caractères vous aurez aussi besoin de caster les entiers en chaînes de caractères: id::varchar transforme l'entier id en chaîne de caractères contenant l'écriture de l'entier en base dix.

Employé	*Nom*	*niveau*	*Noms des chefs*	*Id des chefs*
105	Austin	3	Hunold - De Haan - King	103|102|100
110	Chen	3	Greenberg - Kochhar - King	108|101|100
104	Ernst	3	Hunold - De Haan - King	103|102|100
109	Faviet	3	Greenberg - Kochhar - King	108|101|100
206	Gietz	3	Higgins - Kochhar - King	205|101|100
107	Lorentz	3	Hunold - De Haan - King	103|102|100
106	Pataballa	3	Hunold - De Haan - King	103|102|100
113	Popp	3	Greenberg - Kochhar - King	108|101|100
111	Sciarra	3	Greenberg - Kochhar - King	108|101|100
112	Urman	3	Greenberg - Kochhar - King	108|101|100
204	Baer	2	Kochhar - King	101|100
116	Baida	2	Raphaely - King	114|100
192	Bell	2	Vollman - King	123|100
119	Colmenares	2	Raphaely - King	114|100
193	Everett	2	Vollman - King	123|100
202	Fay	2	Hartstein - King	201|100
108	Greenberg	2	Kochhar - King	101|100
205	Higgins	2	Kochhar - King	101|100
118	Himuro	2	Raphaely - King	114|100
103	Hunold	2	De Haan - King	102|100
115	Khoo	2	Raphaely - King	114|100
203	Mavris	2	Kochhar - King	101|100
126	Mikkilineni	2	Weiss - King	120|100
117	Tobias	2	Raphaely - King	114|100
200	Whalen	2	Kochhar - King	101|100
102	De Haan	1	King	100
121	Fripp	1	King	100
178	Grant	1	King	100
201	Hartstein	1	King	100
179	Johnson	1	King	100
122	Kaufling	1	King	100
101	Kochhar	1	King	100
177	Livingston	1	King	100
146	Partners	1	King	100
114	Raphaely	1	King	100
145	Russell	1	King	100
176	Taylor	1	King	100
123	Vollman	1	King	100
120	Weiss	1	King	100

(39 rows)

```
WITH RECURSIVE hierarchie(soldat,general, niveau) AS (
    SELECT employees.employee_id, employees.manager_id, 1
    FROM employees WHERE manager_id IS NOT NULL
    UNION
    SELECT employees.employee_id, hierarchie.general, niveau+1
    FROM employees JOIN hierarchie
        ON(employees.manager_id= hierarchie.soldat) )
SELECT e2.employee_id as "Employ ", e2.last_name AS "Nom",
    COUNT(general) AS niveau,
    STRING_AGG(e1.last_name, '␣-␣' ORDER BY niveau) AS "Noms␣des␣chefs",
    STRING_AGG(general::varchar, '|' ORDER BY niveau) AS "Id␣des␣chefs"
FROM hierarchie
    JOIN employees e1 ON (general=e1.employee_id)
    JOIN employees E2 ON (soldat=e2.employee_id)
GROUP BY e2.employee_id
ORDER BY niveau DESC, e2.last_name
```

---

2 Statistiques

Interrogez la base de données pour répondre aux questions suivantes:

1. Écrire une requête qui produit le tableau ci-dessous, c'est à dire pour chaque région, chaque pays et chaque ville le nombre de départements implantés dans la ville et le nombre total de tous les employés des départements concernés.

region_name	*country_name*	*city*	*Nombre de départements*	*Effectif total*
Americas	Canada	Toronto	1	2
Americas	United States of America	Seattle	5	18
Americas	United States of America	South San Francisco	1	7
Americas	United States of America	Southlake	1	5
Europe	Germany	Munich	1	1
Europe	United Kingdom	London	1	1
Europe	United Kingdom	Oxford	1	6

(7 rows)

```
SELECT region_name, country_name, locations.city,
    COUNT(DISTINCT department_id) AS "Nombre de d partements",
    COUNT(DISTINCT employee_id) AS "Effectif total"
FROM employees
    JOIN departments USING(department_id)
    JOIN locations USING(location_id)
    JOIN countries USING(country_id)
    JOIN regions USING(region_id)
GROUP BY region_name, country_name, locations.city
ORDER BY region_name, country_name, locations.city;
```

---

2. Modifiez votre requête pour obtenir le tableau suivant:

region_name	*country_name*	*city*	*Nombre de départements*	*Effectif total*
Americas	Canada	Toronto	1	2
Americas	Canada	null	1	2
Americas	United States of America	Seattle	5	18
Americas	United States of America	South San Francisco	1	7
Americas	United States of America	Southlake	1	5
Americas	United States of America	null	7	30
Americas	null	null	8	32
Europe	Germany	Munich	1	1
Europe	Germany	null	1	1
Europe	United Kingdom	London	1	1
Europe	United Kingdom	Oxford	1	6
Europe	United Kingdom	null	2	7
Europe	null	null	3	8
			11	40

Indice : utilisez GROUPING SETS

```
SELECT region_name, country_name, locations.city,
    COUNT(DISTINCT department_id) AS "Nombre de d partements",
    COUNT(DISTINCT employee_id) AS "Effectif total"
FROM employees
    JOIN departments USING(department_id)
    JOIN locations USING(location_id)
    JOIN countries USING(country_id)
    JOIN regions USING(region_id)
GROUP BY
 GROUPING SETS(
        (region_name, country_name, locations.city),
        (region_name, country_name),
        (region_name),
```

```
        ()
    )
ORDER BY region_name, country_name, locations.city;
```

---

3. Modifiez à nouveau votre requête pour remplacer les "null" par des informations plus explicites et obtenir le tableau suivant:

Region	*Pays*	*Villes*	*Nombre de départements*	*Effectif total*
Americas	Canada	Toronto	1	2
Americas	Canada	Toutes les villes	1	2
Americas	United States of America	Seattle	5	18
Americas	United States of America	South San Francisco	1	7
Americas	United States of America	Southlake	1	5
Americas	United States of America	Toutes les villes	7	30
Americas	Tous les pays	Toutes les villes	8	32
Europe	Germany	Munich	1	1
Europe	Germany	Toutes les villes	1	1
Europe	United Kingdom	London	1	1
Europe	United Kingdom	Oxford	1	6
Europe	United Kingdom	Toutes les villes	2	7
Europe	Tous les pays	Toutes les villes	3	8
Toutes les regions	Tous les pays	Toutes les villes	11	40

(14 rows)

Indice : Utilisez COALESCE (rtfm!)

```
SELECT
    COALESCE(region_name, 'Toutes les regions') AS "Region",
    COALESCE(country_name, 'Tous les pays') AS "Pays",
    COALESCE(locations.city, 'Toutes les villes') AS "Villes",
    COUNT(DISTINCT department_id) AS "Nombre de d partements",
    COUNT(DISTINCT employee_id) AS "Effectif total"
FROM employees
    JOIN departments USING(department_id)
    JOIN locations  USING(location_id)
    JOIN countries USING(country_id)
    JOIN regions USING(region_id)
GROUP BY
 GROUPING SETS(
        (region_name, country_name, locations.city),
        (region_name, country_name),
        (region_name),
        ()
    )
ORDER BY region_name, country_name, locations.city;
```

---

4. En utilisant STRING_AGG modifiez à nouveau votre requête pour faire apparaître les noms des départements.

Region	*Pays*	*Villes*	*nbdept*	*Nom des départements*	*Effectif total*
Americas	Canada	Toronto	1	Marketing	2
Americas	Canada	Toutes les villes	1	Marketing	2
Americas	United States of America	Seattle	5	Accounting - Administration - Executive - Finance - Purchasing	18
Americas	United States of America	South San Francisco	1	Shipping	7
Americas	United States of America	Southlake	1	IT	5
Americas	United States of America	Toutes les villes	7	Accounting - Administration - Executive - Finance - IT - Purchasing - Shipping	30
Americas	Tous les pays	Toutes les villes	8	Accounting - Administration - Executive - Finance - IT - Marketing - Purchasing - Shipping	32
Europe	Germany	Munich	1	Public Relations	1
Europe	Germany	Toutes les villes	1	Public Relations	1
Europe	United Kingdom	London	1	Human Resources	1
Europe	United Kingdom	Oxford	1	Sales	6
Europe	United Kingdom	Toutes les villes	2	Human Resources - Sales	7
Europe	Tous les pays	Toutes les villes	3	Human Resources - Public Relations - Sales	8
Toutes les regions	Tous les pays	Toutes les villes	11	Accounting - Administration - Executive - Finance - Human Resources - IT - Marketing - Public Relations - Purchasing - Sales - Shipping	40

(14 rows)

```
SELECT
    COALESCE(region_name, 'Toutes les regions') AS "Region",
        COALESCE(country_name, 'Tous les pays') AS "Pays",
        COALESCE(locations.city, 'Toutes les villes') AS "Villes",
        COUNT(DISTINCT department_id) NbDept,
```

```
        STRING_AGG(DISTINCT department_name, '␣-␣' ORDER BY department_name)
            AS "Nom␣des␣d partements",
        COUNT(DISTINCT employee_id) AS "Effectif␣total"
FROM employees
    JOIN departments USING(department_id)
    JOIN locations USING(location_id)
    JOIN countries USING(country_id)
    JOIN regions USING(region_id)
GROUP BY
 GROUPING SETS(
        (region_name,country_name, locations.city),
        (region_name,country_name),
        (region_name),
        ()
    )
ORDER BY region_name,country_name, locations.city;
```

---

5. Modifiez à nouveau votre requête pour obtenir le résultat suivant:

region	*pays*	*ville*	*nbdept*	*Nom des départements*	*Effectif total*
Americas	Canada	Toronto	1	Marketing	2
Americas	Canada	Toutes les villes: Toronto	1	Marketing	2
Americas	United States of America	Seattle	5	Accounting - Administration - Executive - Finance - Purchasing	18
Americas	United States of America	South San Francisco	1	Shipping	7
Americas	United States of America	Southlake	1	IT	5
Americas	United States of America	Toutes les villes: Seattle, South San Francisco, Southlake	7	Accounting - Administration - Executive - Finance - IT - Purchasing - Shipping	30
Americas	Tous les pays: Canada, United States of America	Toutes les villes: Seattle, South San Francisco, Southlake, Toronto	8	Accounting - Administration - Executive - Finance - IT - Marketing - Purchasing - Shipping	32
Europe	Germany	Munich	1	Public Relations	1
Europe	Germany	Toutes les villes: Munich	1	Public Relations	1
Europe	United Kingdom	London	1	Human Resources	1
Europe	United Kingdom	Oxford	1	Sales	6
Europe	United Kingdom	Toutes les villes: London, Oxford	2	Human Resources - Sales	7
Europe	Tous les pays: Germany, United Kingdom	Toutes les villes: London, Munich, Oxford	3	Human Resources - Public Relations - Sales	8
Toutes les regions: Americas, Europe	Tous les pays: Canada, Germany, United Kingdom, United States of America	Toutes les villes: London, Munich, Oxford, Seattle, South San Francisco, Southlake, Toronto	11	Accounting - Administration - Executive - Finance - Human Resources - IT - Marketing - Public Relations - Purchasing - Sales - Shipping	40

(14 rows)

```
SELECT
    COALESCE(region_name,
            'Toutes les regions: ' ||
            STRING_AGG(DISTINCT region_name, ', ' ORDER BY region_name))
        AS region,
    COALESCE(country_name,
```

```
            'Tous les pays: ' ||
            STRING_AGG(DISTINCT country_name, ', ' ORDER BY country_name))
        AS pays,
    COALESCE(locations.city ,
            'Toutes les  villes: ' ||
            STRING_AGG(DISTINCT locations.city, ', ' ORDER BY locations.city))
        AS ville,
    COUNT(DISTINCT department_name) NbDept,
        STRING_AGG(DISTINCT department_name, ' - ' ORDER BY department_name)
            AS "Nom des d partements",
        COUNT(DISTINCT employee_id) AS "Effectif total"
FROM employees
    JOIN departments USING(department_id)
    JOIN locations USING(location_id)
    JOIN countries USING(country_id)
    JOIN regions USING(region_id)
GROUP BY
 GROUPING SETS(
        (region_name,country_name, locations.city),
        (region_name,country_name),
        (region_name),
        ()
    )
ORDER BY region_name,country_name, locations.city;
```

---

6. Dernière modification, pour obtenir le tableau suivant:

   Indice : Utilisez CASE

region	*pays*	*ville*	*Nom des départements*
Americas	Canada	Toronto	Un département: Marketing
Americas	Canada	La ville: Toronto	Un département: Marketing
Americas	United States of America	Seattle	5 départements: Accounting - Administration - Executive - Finance - Purchasing
Americas	United States of America	South San Francisco	Un département: Shipping
Americas	United States of America	Southlake	Un département: IT
Americas	United States of America	Les 3 villes: Seattle, South San Francisco, Southlake	7 départements: Accounting - Administration - Executive - Finance - IT - Purchasing - Shipping
Americas	Les 2 pays: Canada, United States of America	Les 4 villes: Seattle, South San Francisco, Southlake, Toronto	8 départements: Accounting - Administration - Executive - Finance - IT - Marketing - Purchasing - Shipping
Europe	Germany	Munich	Un département: Public Relations
Europe	Germany	La ville: Munich	Un département: Public Relations
Europe	United Kingdom	London	Un département: Human Resources
Europe	United Kingdom	Oxford	Un département: Sales
Europe	United Kingdom	Les 2 villes: London, Oxford	2 départements: Human Resources - Sales
Europe	Les 2 pays: Germany, United Kingdom	Les 3 villes: London, Munich, Oxford	3 départements: Human Resources - Public Relations - Sales
Les 2 regions: Americas, Europe	Les 4 pays: Canada, Germany, United Kingdom, United States of America	Les 7 villes: London, Munich, Oxford, Seattle, South San Francisco, Southlake, Toronto	11 départements: Accounting - Administration - Executive - Finance - Human Resources - IT - Marketing - Public Relations - Purchasing - Sales - Shipping

(14 rows)

```
SELECT
    COALESCE(region_name, 'Les␣' || COUNT(DISTINCT region_name) || '␣regions:␣'
            || STRING_AGG(DISTINCT region_name, ',␣' ORDER BY region_name))
        AS region,
    COALESCE(country_name, 'Les␣' || COUNT(DISTINCT country_name) || '␣pays:␣'
            || STRING_AGG(DISTINCT country_name, ',␣' ORDER BY country_name))
        AS pays,
    COALESCE(locations.city,
        CASE COUNT(DISTINCT locations.city)
                WHEN 1 THEN 'La␣ville:␣'
                ELSE 'Les␣'|| COUNT(DISTINCT locations.city) || '␣villes:␣'
                END
                || STRING_AGG(DISTINCT locations.city,',␣' ORDER BY locations.ci
        AS ville,
    CASE COUNT(DISTINCT department_name)
```

```
        WHEN 1 THEN 'Un d partement: '
        ELSE COUNT(DISTINCT department_name) || ' d partements: '
        END
        || STRING_AGG(DISTINCT department_name,' - ' ORDER BY department_name)
        AS "Nom des d partements"
FROM departments
    JOIN locations USING(location_id)
    JOIN countries USING(country_id)
    JOIN regions USING(region_id)
GROUP BY
 GROUPING SETS(
        (region_name,country_name, locations.city),
        (region_name,country_name),
        (region_name),
        ()
    )
ORDER BY region_name,country_name, locations.city;
```

---

7. Dans votre requête remplacez les GROUPING SETS par CUBE(region_name,country_name, locations.city).

   Quelle(s) différence(s)? Expliquez.

   Est ce une bonne idée ici?

   On obtient des lignes supplémentaires car c'est équivalent à mettre dans les GROUPING SETS les 8 sous-ensembles de (region_name,country_name, locations.city).

   D'où 39 lignes dont beaucoup d'inutiles ici, par exemple le GROUP BY locations.city, region_name n'aurait un effet différent du GROUP BY locations.city que s'il y avait deux villes homonymes dans deux régions différentes. Et même dans ce cas pourquoi voudrait-on confondre les deux villes?

---

8. En utilisant MAX() OVER(), affichez pour chaque employé

   - prénom
   - nom
   - nom de son département
   - salaire
   - salaire le plus élevé dans son département.

first_name	*last_name*	*department_name*	*salary*	*SalaireMaximum*
Jennifer	Whalen	Administration	4400.00	4400.00
Pat	Fay	Marketing	6000.00	13000.00
Michael	Hartstein	Marketing	13000.00	13000.00
Karen	Colmenares	Purchasing	2500.00	11000.00
Guy	Himuro	Purchasing	2600.00	11000.00
Sigal	Tobias	Purchasing	2800.00	11000.00
Shelli	Baida	Purchasing	2900.00	11000.00
Alexander	Khoo	Purchasing	3100.00	11000.00
Den	Raphaely	Purchasing	11000.00	11000.00
Susan	Mavris	Human Resources	6500.00	6500.00
Britney	Everett	Shipping	3900.00	8200.00
Sarah	Bell	Shipping	4000.00	8200.00
Irene	Mikkilineni	Shipping	2700.00	8200.00
Shanta	Vollman	Shipping	6500.00	8200.00
Payam	Kaufling	Shipping	7900.00	8200.00
Adam	Fripp	Shipping	8200.00	8200.00
Matthew	Weiss	Shipping	8000.00	8200.00
Diana	Lorentz	IT	4200.00	9000.00
Valli	Pataballa	IT	4800.00	9000.00
David	Austin	IT	4800.00	9000.00
Bruce	Ernst	IT	6000.00	9000.00
Alexander	Hunold	IT	9000.00	9000.00
Hermann	Baer	Public Relations	10000.00	10000.00
Charles	Johnson	Sales	6200.00	14000.00
Kimberely	Grant	Sales	7000.00	14000.00
Jack	Livingston	Sales	8400.00	14000.00
Jonathon	Taylor	Sales	8600.00	14000.00
Karen	Partners	Sales	13500.00	14000.00
John	Russell	Sales	14000.00	14000.00
Steven	King	Executive	30100.00	30100.00
Lex	De Haan	Executive	17000.00	30100.00
Neena	Kochhar	Executive	17000.00	30100.00
Luis	Popp	Finance	6900.00	12000.00
Jose Manuel	Urman	Finance	7800.00	12000.00
Ismael	Sciarra	Finance	7700.00	12000.00
John	Chen	Finance	8200.00	12000.00
Daniel	Faviet	Finance	9000.00	12000.00
Nancy	Greenberg	Finance	12000.00	12000.00
William	Gietz	Accounting	8300.00	12000.00
Shelley	Higgins	Accounting	12000.00	12000.00

(40 rows)

```
SELECT
    first_name,
    last_name,
    department_name,
    salary,
    MAX(salary) OVER(PARTITION BY department_id) "SalaireMaximum"
FROM employees JOIN departments USING(department_id);
```

---

9. En utilisant LEAD() OVER() et LAG() OVER(), affichez pour chaque employé

   - prénom
   - nom
   - nom de son département
   - salaire
   - salaire immédiatement inférieur dans son département
   - salaire immédiatement supérieur dans son département

Triez par département, salaire et nom

department_name	*last_name*	*salary*	*next_highest_salary*	*previous_highest_salary*
Accounting	Gietz	8300.00	12000.00	
Accounting	Higgins	12000.00		8300.00
Administration	Whalen	4400.00		
Executive	De Haan	17000.00	30100.00	17000.00
Executive	Kochhar	17000.00	17000.00	
Executive	King	30100.00		17000.00
Finance	Popp	6900.00	7700.00	
Finance	Sciarra	7700.00	7800.00	6900.00
Finance	Urman	7800.00	8200.00	7700.00
Finance	Chen	8200.00	9000.00	7800.00
Finance	Faviet	9000.00	12000.00	8200.00
Finance	Greenberg	12000.00		9000.00
Human Resources	Mavris	6500.00		
IT	Lorentz	4200.00	4800.00	
IT	Austin	4800.00	4800.00	4200.00
IT	Pataballa	4800.00	6000.00	4800.00
IT	Ernst	6000.00	9000.00	4800.00
IT	Hunold	9000.00		6000.00
Marketing	Fay	6000.00	13000.00	
Marketing	Hartstein	13000.00		6000.00
Public Relations	Baer	10000.00		
Purchasing	Colmenares	2500.00	2600.00	
Purchasing	Himuro	2600.00	2800.00	2500.00
Purchasing	Tobias	2800.00	2900.00	2600.00
Purchasing	Baida	2900.00	3100.00	2800.00
Purchasing	Khoo	3100.00	11000.00	2900.00
Purchasing	Raphaely	11000.00		3100.00
Sales	Johnson	6200.00	7000.00	
Sales	Grant	7000.00	8400.00	6200.00
Sales	Livingston	8400.00	8600.00	7000.00
Sales	Taylor	8600.00	13500.00	8400.00
Sales	Partners	13500.00	14000.00	8600.00
Sales	Russell	14000.00		13500.00
Shipping	Mikkilineni	2700.00	3900.00	
Shipping	Everett	3900.00	4000.00	2700.00
Shipping	Bell	4000.00	6500.00	3900.00
Shipping	Vollman	6500.00	7900.00	4000.00
Shipping	Kaufling	7900.00	8000.00	6500.00
Shipping	Weiss	8000.00	8200.00	7900.00
Shipping	Fripp	8200.00		8000.00

(40 rows)

```
SELECT  department_name,
        last_name,
        salary,
        LEAD(salary,1) OVER(PARTITION BY department_name ORDER BY salary)
            AS next_highest_salary,
        LAG(salary,1) OVER(PARTITION BY department_name ORDER BY salary)
            AS previous_highest_salary
           FROM employees JOIN departments USING(department_id)
ORDER BY department_name, salary, last_name;
```

---

10. Pour chaque département, affichez le nom du département, le prénom, le nom et le salaire de l'employé dont le salaire est le troisième par ordre décroissant dans son département.

    Pour cela utilisez ROWNUMBER() OVER()

department_name	*first_name*	*last_name*	*salary*
Executive	Neena	Kochhar	17000.00
Sales	Jonathon	Taylor	8600.00
Finance	John	Chen	8200.00
Shipping	Payam	Kaufling	7900.00
IT	Valli	Pataballa	4800.00
Purchasing	Shelli	Baida	2900.00

(6 rows)

```
SELECT
    department_name,
    first_name,
    last_name,
    salary
FROM (
        SELECT
            department_name,
            ROW_NUMBER() OVER(
                PARTITION BY department_name
                ORDER BY salary DESC) row_num,
            first_name,
            last_name,
            salary
        FROM employees e INNER JOIN departments d
            ON d.department_id = e.department_id
    ) t
WHERE row_num = 3
ORDER BY salary DESC;
```

---

11. Affichez :

    - nom de leur département
    - nom de l'employé
    - salaire
    - salaire immediatement inférieur dans le même département
    - différence entre les deux (0 si previous_highest_salary n'est pas défini)

    pour tous les employés pour lequels la différence entre les deux salaires est <300

department_name	*last_name*	*salary*	*previous_highest_salary*	*step*
Accounting	Gietz	8300.00		0
Administration	Whalen	4400.00		0
Executive	De Haan	17000.00	17000.00	0.00
Executive	Kochhar	17000.00		0
Finance	Popp	6900.00		0
Finance	Urman	7800.00	7700.00	100.00
Human Resources	Mavris	6500.00		0
IT	Lorentz	4200.00		0
IT	Pataballa	4800.00	4800.00	0.00
Marketing	Fay	6000.00		0
Public Relations	Baer	10000.00		0
Purchasing	Colmenares	2500.00		0
Purchasing	Himuro	2600.00	2500.00	100.00
Purchasing	Tobias	2800.00	2600.00	200.00
Purchasing	Baida	2900.00	2800.00	100.00
Purchasing	Khoo	3100.00	2900.00	200.00
Sales	Johnson	6200.00		0
Sales	Taylor	8600.00	8400.00	200.00
Shipping	Mikkilineni	2700.00		0
Shipping	Bell	4000.00	3900.00	100.00
Shipping	Weiss	8000.00	7900.00	100.00
Shipping	Fripp	8200.00	8000.00	200.00

(22 rows)

```
WITH ecart AS (
    SELECT
        department_name,
        last_name,
        salary,
        LAG(salary,1) OVER(PARTITION BY department_name ORDER BY salary)
            AS previous_highest_salary,
        CASE    WHEN (  LAG(salary,1)
                        OVER(PARTITION BY department_name ORDER BY salary)
                        IS NULL )
                THEN    0
                ELSE    salary -
                        LAG(salary,1)
                        OVER(PARTITION BY department_name ORDER BY salary)
        END AS step
    FROM employees e JOIN departments d
       ON d.department_id = e.department_id )
SELECT *
FROM ecart
WHERE step<300
ORDER BY department_name, salary, last_name;
```

---

3 Triggers

La base de données contient un trigger updatesalaries qui est déclenché chaque fois que le salaire d'un employé est modifié.

1. (a) Affichez le contenu de la table salary_changes
   (b) Augmentez le salaire de Steven King de la somme que vous voulez
   (c) Affichez le contenu de la table salary_changes
   (d) Augmentez le salaire de Steven King de la somme que vous voulez

(e) Affichez le contenu de la table salary_changes

Sous pgadmin, où est défini le trigger, où se trouve la définition de la fonction ?

```
SELECT * FROM salary_changes;
SELECT * FROM employees
WHERE last_name='King' AND first_name='Steven';
UPDATE employees SET salary='30500'
WHERE last_name='King' AND first_name='Steven';
SELECT * FROM employees
WHERE last_name='King' AND first_name='Steven';
SELECT * FROM salary_changes;
UPDATE employees SET salary='30800'
WHERE last_name='King' AND first_name='Steven';
SELECT * FROM employees
WHERE last_name='King' AND first_name='Steven';
SELECT * FROM salary_changes;
```

---

2. Vérifiez que les informations dans jobs sur les salaires max et min sont cohérentes avec les informations de la table employees. Au besoin faites les corrections.

```
SELECT jobs.*, employee_id, salary
FROM jobs JOIN employees USING(job_id)
WHERE salary<jobs.min_salary OR salary>jobs.max_salary
```

---

3. Écrire un trigger qui va garantir cette cohérence, c'est à dire n'accepter de faire l'update d'un salaire que si le nouveau salaire est bien dans l'intervalle prévu.

```
CREATE FUNCTION public.checksalary()
    RETURNS trigger
    LANGUAGE 'plpgsql'
    COST 100
    VOLATILE NOT LEAKPROOF
    AS $BODY$
      DECLARE
        salairemin numeric(8,2);
        salairemax numeric(8,2);
      BEGIN
        salairemax=(SELECT max_salary FROM jobs WHERE jobs.job_id=OLD.job_id);
        salairemin=(SELECT min_salary FROM jobs WHERE jobs.job_id=OLD.job_id);
        IF NEW.salary<salairemax AND NEW.salary>salairemin THEN RETURN NEW;
        ELSE RETURN OLD;
        END IF;
      END; $BODY$;

CREATE TRIGGER before_update_salary
    BEFORE UPDATE
    ON public.employees
    FOR EACH ROW
    EXECUTE PROCEDURE public.checksalary();

UPDATE employees SET salary='60000'
WHERE last_name='King' AND first_name='Steven';
SELECT * FROM employees
WHERE last_name='King' AND first_name='Steven';
SELECT * FROM salary_changes;
```

---

4 Annexe

```
CREATE TABLE regions (
    region_id SERIAL PRIMARY KEY,
    region_name character varying(25)
);

CREATE TABLE countries (
    country_id character(2) PRIMARY KEY,
    country_name character varying(40),
    region_id integer NOT NULL REFERENCES regions
);

CREATE TABLE locations (
    location_id SERIAL PRIMARY KEY,
    street_address character varying(40),
    postal_code character varying(12),
    city character varying(30) NOT NULL,
    state_province character varying(25),
    country_id character(2) NOT NULL REFERENCES countries
);

CREATE TABLE departments (
    department_id SERIAL PRIMARY KEY,
    department_name character varying(30) NOT NULL,
    location_id integer REFERENCES locations
);

CREATE TABLE jobs (
    job_id SERIAL PRIMARY KEY,
    job_title character varying(35) NOT NULL,
    min_salary numeric(8,2),
    max_salary numeric(8,2)
);

CREATE TABLE employees (
    employee_id SERIAL PRIMARY KEY,
    first_name character varying(20),
    last_name character varying(25) NOT NULL,
    email character varying(100) NOT NULL,
    phone_number character varying(20),
    hire_date date NOT NULL,
    job_id integer NOT NULL REFERENCES jobs,
    salary numeric(8,2) NOT NULL,
    manager_id integer REFERENCES employees,
    department_id integer REFERENCES departments
);
-- Inutile pour le TP3

CREATE TABLE dependents (
    dependent_id SERIAL PRIMARY KEY,
    first_name character varying(50) NOT NULL,
    last_name character varying(50) NOT NULL,
    relationship character varying(25) NOT NULL,
    employee_id integer NOT NULL REFERENCES employees
);

CREATE TABLE salary_changes (
    employee_id integer NOT NULL,
    changed_at timestamp(4) with time zone DEFAULT CURRENT_TIMESTAMP NOT NULL,
    old_salary numeric(8,2),
    new_salary numeric(8,2),
    modifiant text
);
```

```
CREATE FUNCTION updatesalaries() RETURNS trigger AS $$
    BEGIN
        IF NEW.salary <> OLD.salary THEN
        INSERT INTO salary_changes(employee_id,old_salary,new_salary,modifiant)
        VALUES(NEW.employee_id,OLD.salary,NEW.salary,current_user);
        END IF;
        RETURN NEW;
    END; $$
    LANGUAGE plpgsql;

CREATE TRIGGER after_update_salary
    AFTER UPDATE
    ON public.employees
    FOR EACH ROW
    EXECUTE PROCEDURE public.updatesalaries();
```