### Write the SQL query to find the employee names who have same salary as another employee in the same department.

### ( [**https://www.youtube.com/watch?v=sf4gxdRxLbI&t=12s**](https://www.youtube.com/watch?v=sf4gxdRxLbI&t=12s) )

|  |  |  |  |
| --- | --- | --- | --- |
| **ID** | **NAME** | **DEPT** | **SALARY** |
| 1 | Akash | Sales | 100 |
| 2 | John | Sales | 110 |
| 3 | Rohit | Sales | 100 |
| 4 | Tom | IT | 200 |
| 5 | Subham | IT | 205 |
| 6 | Vabna | IT | 200 |
| 7 | Prativa | Marketing | 150 |
| 8 | Rahul | Marketing | 155 |
| 9 | yash | Marketing | 160 |

**Ans:**

Approach-1

select NAME,DEPT,SALARY from (

select \*,count(\*) over(partition by dept,salary order by salary) as grp\_cnt from emp\_info

) A

where grp\_cnt >=2;

approach-2

select a.\* from emp\_info a inner join emp\_info b on a.salary=b.salary and a.dept=b.dept and a.id<>b.id;

### 2.

### You have a Sales table with the following columns: sale\_id, employee\_id, sale\_amount, sale\_date.

### Write a query that:

### Shows each employee’s total sales in each month.

### Displays the percentage of total sales that each employee contributed in that month, using SUM() OVER (PARTITION BY sale\_date).

|  |  |  |  |
| --- | --- | --- | --- |
| **EMPLOYEE\_ID** | **SALE\_MONTH** | **TOTAL\_SALES\_PER\_EMPLOYEE** | **SALES\_PERCENTAGE\_OF\_TOTAL** |
| 101 | 30-01-2025 | 5000 | 25 |
| 102 | 30-01-2025 | 10000 | 50 |
| 103 | 30-01-2025 | 5000 | 25 |
| 101 | 28-02-2025 | 4000 | 40 |
| 102 | 28-02-2025 | 6000 | 60 |

**Ans:**

SElect

\*,sum(total\_sales\_per\_employee) over(partition by employee\_id,sale\_month) AS TOT\_SALES from sales order by employee\_id, sale\_month;

### 3.Write the SQL query to the manager who manage more than 4 employees. (Ref: [**https://www.youtube.com/watch?v=w7cCpbLYwGM**](https://www.youtube.com/watch?v=w7cCpbLYwGM))

|  |  |  |  |
| --- | --- | --- | --- |
| **ID** | **NAME** | **DEPARTMENT** | **MANAGERID** |
| 1 | John | HR |  |
| 2 | Bob | HR | 1 |
| 3 | Olivia | HR | 1 |
| 4 | Emma | Finance |  |
| 5 | Sophia | HR | 1 |
| 6 | Mason | Finance | 4 |
| 7 | Ethan | HR | 1 |
| 8 | Ava | HR | 1 |
| 9 | Lucas | HR | 1 |
| 10 | Isabella | Finance | 4 |
| 11 | Harper | Finance | 4 |
| 12 | Hemla | HR | 3 |
| 13 | Aisha | HR | 2 |
| 14 | Himani | HR | 2 |
| 15 | Lily | HR | 2 |

**Ans :**

**Step-1:**

**Do inner join the table and get the corresponding manager id.**

**select a.\*,b.name as mgr\_NM, b.ID as MGR\_ID from employee a inner join employee b on a.managerid=b.id;**

**step-2:**

**Calculate the count of the MGR\_ID based on partition by MGR\_ID.**

**SELECT \* FROM (**

**select \*,count(MGR\_ID) OVER(PARTITION BY MGR\_ID order by MGR\_ID) CNT\_SUB\_ORD from (**

**select a.\*,b.name as mgr\_NM, b.ID as MGR\_ID from employee a inner join employee b on a.managerid=b.id) A**

**)B where CNT\_SUB\_ORD>=4;**

### 4. Write a SQL query to find numbers that appear at least 3 times consequently.

Source :

|  |  |
| --- | --- |
| **ID** | **NUM** |
| 1 | 1 |
| 2 | 1 |
| 3 | 2 |
| 4 | 2 |
| 5 | 2 |
| 6 | 1 |
| 7 | 3 |
| 8 | 4 |
| 9 | 4 |

**Target :**

|  |  |
| --- | --- |
| **ID** | **NUM** |
| 3 | 2 |
| 4 | 2 |
| 5 | 2 |

### 5)

### Write a query to find out if the previous year sales of the product is increased or not.

|  |  |  |
| --- | --- | --- |
| PRODUCT\_NAME | YEAROFSALES | TOTALSALES |
| Nokia | 31-12-2020 | 11 |
| Iphone | 31-12-2020 | 22 |
| Samsung | 31-12-2020 | 33 |
| Oneplue | 31-12-2020 | 44 |
| Nokia | 31-12-2021 | 8 |
| Iphone | 31-12-2021 | 33 |
| Samsung | 31-12-2021 | 44 |
| Oneplue | 31-12-2021 | 55 |

**Ans:**

with sales\_1 as (

select PRODUCT\_NAME,YEAROFSALES,TOTALSALES,lag(totalsales) over(partition by product\_name order by yearofsales) PREV\_YR\_SALES from sales order by product\_name,yearofsales)

select product\_name,

yearofsales,

TOTALSALES,

prev\_yr\_sales,

case

when TOTALSALES > PREV\_yr\_sales then 'incresed'

WHEN TOTALSALES < PREV\_YR\_SALES THEN 'DECRESED'

ELSE 'nO cHANGE'

END AS SALES\_STATUS

from sales\_1

WHERE PREV\_YR\_SALES is noT NULL;

### 6. Write a SQL query to find consecutive 3 rows with the vistited people more than 500. (https://www.youtube.com/watch?v=khQ3QgCPvpw)

|  |  |  |
| --- | --- | --- |
| ID | DATE\_ID | VISITED\_PEOPLE |
| 1 | 01-01-2024 | 700 |
| 2 | 01-02-2024 | 460 |
| 3 | 01-03-2024 | 550 |
| 4 | 01-04-2024 | 510 |
| 5 | 01-05-2024 | 550 |
| 6 | 01-06-2024 | 540 |
| 7 | 01-07-2024 | 90 |
| 8 | 01-08-2024 | 650 |
| 9 | 01-09-2024 | 580 |
| 10 | 01-10-2024 | 590 |

ANS:

WITH visited\_t as (

SELECT DATE\_ID,VISITED\_PEOPLE,

LAG(VISITED\_PEOPLE,1) OVER(ORDER BY DATE\_ID) PREV\_VISITED\_1,

LAG(VISITED\_PEOPLE,2) OVER(ORDER BY DATE\_ID) PREV\_VISITED\_2 FROM tourist ORDER BY DATE\_ID

)

select \* from visited\_t WHERE

VISITED\_PEOPLE > 500 AND PREV\_VISITED\_1 > 500 AND PREV\_VISITED\_2 > 500

order by DATE\_ID

### 7.

### https://www.youtube.com/watch?v=P6kNMyqKD0A

Source:

|  |  |  |  |  |
| --- | --- | --- | --- | --- |
| NAME | ADDRESS | EMAIL | FLOOR | RESOURCES |
| A | Bangalore | A@gmail.com | 1 | CPU |
| A | Bangalore | A1@gmail.com | 1 | CPU |
| A | Bangalore | A2@gmail.com | 2 | DESKTOP |
| B | Bangalore | B@gmail.com | 2 | DESKTOP |
| B | Bangalore | B1@gmail.com | 2 | DESKTOP |
| B | Bangalore | B2@gmail.com | 1 | MONITOR |

Target:

|  |  |  |  |
| --- | --- | --- | --- |
| NAME | TOTAL\_VISITS | Most\_Visited\_floor | resources\_used |
| A | 3 | 1 | CPU,DESKTOP |
| B | 3 | 2 | DESKTOP,MONITOR |

WITH VisitCount AS (

SELECT

NAME,

COUNT(\*) AS TOTAL\_VISITS

FROM entries

GROUP BY NAME

),

FloorRanking AS (

SELECT

NAME,

FLOOR,

COUNT(\*) AS FloorVisitCount,

RANK() OVER (PARTITION BY NAME ORDER BY COUNT(\*) DESC) AS rnk

FROM entries

GROUP BY NAME, FLOOR

),

MostVisitedFloor AS (

SELECT NAME, FLOOR AS Most\_Visited\_floor

FROM FloorRanking

WHERE rnk = 1

),

ResourceAggregation AS (

SELECT

NAME,

LISTAGG(DISTINCT RESOURCES, ', ') AS resources\_used

FROM entries

GROUP BY NAME

)

SELECT

vc.NAME,

vc.TOTAL\_VISITS,

mvf.Most\_Visited\_floor,

ra.resources\_used

FROM VisitCount vc

JOIN MostVisitedFloor mvf ON vc.NAME = mvf.NAME

JOIN ResourceAggregation ra ON vc.NAME = ra.NAME;

### 8. Write a query to find the repeat customer and existing customer.

### ( <https://www.youtube.com/watch?v=MpAMjtvarrc> )

**Source:**

|  |  |  |  |
| --- | --- | --- | --- |
| **ORDER\_ID** | **CUSTOMER\_ID** | **ORDER\_DATE** | **ORDER\_AMOUNT** |
| 1 | 100 | 01-01-2022 | 2000 |
| 2 | 200 | 01-01-2022 | 2500 |
| 3 | 300 | 01-01-2022 | 2100 |
| 4 | 100 | 02-01-2022 | 2000 |
| 5 | 400 | 02-01-2022 | 2200 |
| 6 | 500 | 02-01-2022 | 2700 |
| 7 | 100 | 03-01-2022 | 3000 |
| 8 | 400 | 03-01-2022 | 1000 |
| 9 | 600 | 03-01-2022 | 3000 |

**Target:**

|  |  |  |
| --- | --- | --- |
| **order\_date** | **new\_customer\_count** | **repeat\_customer\_count** |
| 01-01-2022 | 3 | 0 |
| 02-01-2022 | 2 | 1 |
| 03-01-2022 | 1 | 2 |

**ANS:**

**Step1: Extract the first time orders based on the customer\_id.**

**Step2: Compare all the orders with first time order date**

WITH FirstOrders AS (

SELECT CUSTOMER\_ID, MIN(ORDER\_DATE) AS FIRST\_ORDER\_DATE

FROM Orders

GROUP BY CUSTOMER\_ID

),

CustomerType AS (

SELECT

o.ORDER\_DATE,

o.CUSTOMER\_ID,

CASE

WHEN f.FIRST\_ORDER\_DATE = o.ORDER\_DATE THEN 'New'

ELSE 'Repeat'

END AS CUSTOMER\_TYPE

FROM Orders o

JOIN FirstOrders f ON o.CUSTOMER\_ID = f.CUSTOMER\_ID

)

SELECT

ORDER\_DATE,

SUM(CASE WHEN CUSTOMER\_TYPE = 'New' THEN 1 ELSE 0 END) AS new\_customer\_count,

SUM(CASE WHEN CUSTOMER\_TYPE = 'Repeat' THEN 1 ELSE 0 END) AS repeat\_customer\_count

FROM CustomerType

GROUP BY ORDER\_DATE

**ORDER BY ORDER\_DATE;**

### **9)** Write query by using below two tables

### **a) Departments**

### **b) Employees**

EMPLOYEES data:

|  |  |  |  |  |  |
| --- | --- | --- | --- | --- | --- |
| EMPLOYEEID | FIRSTNAME | LASTNAME | DEPARTMENTID | SALARY | DATEHIRED |
| 1 | Alice | Smith | 1 | 50000 | 15-01-2020 |
| 2 | Bob | Johnson | 1 | 60000 | 22-03-2018 |
| 3 | Charlie | Williams | 2 | 70000 | 30-07-2019 |
| 4 | David | Brown | 2 | 80000 | 11-11-2017 |
| 5 | Eve | Davis | 3 | 90000 | 25-02-2021 |
| 6 | Frank | Miller | 3 | 55000 | 10-09-2020 |
| 7 | Grace | Wilson | 2 | 75000 | 05-04-2016 |
| 8 | Henry | Moore | 1 | 65000 | 17-06-2022 |

Departments data:

|  |  |
| --- | --- |
| DEPARTMENTID | DEPARTMENTNAME |
| 1 | HR |
| 2 | Engineering |
| 3 | Sales |

### Part 1: Find the top 3 highest-paid employees in each department.

with cte1 as (

select e.\*, rank() over(partition by departmentid order by salary desc) rnk from employees e

)

select \* from cte1 where rnk<=2

### Part 2: Find the average salary of employees hired in the last 5 years.

SELECT AVG(SALARY) AS avg\_salary

FROM employees

WHERE DATEHIRED >= DATEADD(YEAR, -5, CURRENT\_DATE);

### Part 3: Find the employees whose salary is less than the average salary of employees hired in the last 5 years.

with e\_avg\_sal as

(

select cast(avg(salary) AS DECIMAL(8,2)) as AVG\_SAL from employees where datehired>=DATEADD(YEAR,-5,CURRENT\_DATE)

)

select \* from employees where salary < (select avg\_sal from e\_avg\_sal)

### 10.

Write query to convert below source to target.

|  |  |  |
| --- | --- | --- |
| **Source:** |  |  |
| **Team\_1** | **Team\_2** | **Winner** |
| India | SL | India |
| SL | AUS | AUS |
| SA | ENG | ENG |
| ENG | NZ | NZ |
| AUS | India | India |

|  |  |  |  |
| --- | --- | --- | --- |
| **Target:** |  |  |  |
| **Team\_Name** | **Matches\_Played** | **no\_of\_wins** | **no\_of\_losses** |
| India | 2 | 2 | 0 |
| SL | 2 | 0 | 2 |
| SA | 1 | 0 | 1 |
| ENG | 2 | 1 | 1 |
| AUS | 2 | 1 | 1 |
| NZ | 1 | 1 | 0 |

Ans:

WITH Matches AS (

SELECT Team, COUNT(\*) AS Matches\_Played

FROM (

SELECT Team\_1 AS Team FROM ICC\_WORLD\_CUP

UNION ALL

SELECT Team\_2 AS Team FROM ICC\_WORLD\_CUP

) AS AllTeams

GROUP BY Team

),

Wins AS (

SELECT Winner, COUNT(\*) AS no\_of\_wins

FROM ICC\_WORLD\_CUP

GROUP BY Winner

)

SELECT

M.Team AS Team\_Name,

M.Matches\_Played,

COALESCE(W.no\_of\_wins, 0) AS no\_of\_wins,

M.Matches\_Played - COALESCE(W.no\_of\_wins, 0) AS no\_of\_losses

FROM Matches M

LEFT JOIN Wins W ON M.Team = W.Winner

ORDER BY Team\_Name;