1. Given a table orders with columns order\_id (sequential integers), order\_date, write a query to identify the missing order IDs.

(<https://www.youtube.com/watch?v=UCywRDY4F68>)

Source:

|  |  |
| --- | --- |
| **ORDER\_ID** | **ORDER\_DATE** |
| 1 | 01-07-2024 |
| 2 | 02-07-2024 |
| 3 | 03-07-2024 |
| 5 | 05-07-2024 |
| 6 | 06-07-2024 |
| 7 | 07-07-2024 |
| 9 | 09-07-2024 |
| 10 | 10-07-2024 |
| 11 | 11-07-2024 |
| 13 | 13-07-2024 |
| 14 | 14-07-2024 |
| 15 | 15-07-2024 |
| 16 | 16-07-2024 |
| 18 | 18-07-2024 |
| 19 | 19-07-2024 |
| 20 | 20-07-2024 |

**Expected Target:**

|  |
| --- |
| **ORDER\_ID** |
| 4 |
| 8 |
| 12 |
| 17 |

Ans:

with cre\_seq\_no as (

select min(order\_id) as sno from orders\_1

union all

select sno+1 as sno from cre\_seq\_no where sno < (select max(order\_id) from orders\_1)

)

select SNO as ORDER\_ID from cre\_seq\_no a left join orders\_1 b on a.sno=b.order\_id where order\_id IS NULL

1. Write an SQL query using a recursive CTE to retrieve the hierarchical structure of employees.

( https://www.youtube.com/watch?v=\_7uouRjwzn8 )

Source:

|  |  |  |
| --- | --- | --- |
| EMPLOYEEID | MANAGERID | EMPLOYEENAME |
| 1 |  | Alice |
| 2 | 1 | Bob |
| 3 | 1 | Charlie |
| 4 | 2 | David |
| 5 | 2 | Eve |
| 6 | 3 | Frank |
| 7 | 4 | Grace |

Expected Output:

|  |  |  |  |
| --- | --- | --- | --- |
| EMPLOYEEID | MANAGERID | EMPLOYEENAME | LEVEL |
| 1 |  | Alice | 1 |
| 2 | 1 | Bob | 2 |
| 3 | 1 | Charlie | 2 |
| 4 | 2 | David | 3 |
| 5 | 2 | Eve | 3 |
| 6 | 3 | Frank | 3 |
| 7 | 4 | Grace | 4 |

Ans:

with cte as (

select \*,1 AS LEVEL from employees\_1 where managerid is null

union all

select e.employeeid,e.managerid,e.employeename,e1.level+1 as level from employees\_1 e

inner join cte e1 on e.managerid=e1.employeeid where e.managerid is not null

)

select \* from cte

1. Transform Rows Into Columns:

( https://www.youtube.com/watch?v=Ebm29FSB7Z0 )

Source:

|  |  |  |
| --- | --- | --- |
| MONTH | CATEGORY | AMOUNT |
| January | Electronics | 1500 |
| January | Clothing | 1200 |
| February | Electronics | 1800 |
| February | Clothing | 1300 |
| March | Electronics | 1600 |
| March | Clothing | 1100 |
| April | Electronics | 1700 |
| April | Clothing | 1400 |

Target:

|  |  |  |
| --- | --- | --- |
| Month | Clothing | Electronics |
| January | 1200 | 1500 |
| Febraury | 1300 | 1800 |
| March | 1100 | 1600 |
| April | 1400 | 1700 |

Ans:

select month,

sum(case when category='Electronics' then AMOUNT END) As ELECTRONICS,

sum(case when category='Clothing' then AMOUNT END) AS CLOTHING

from sales\_data group by month

1. Retrieve the total revenue generated from each product.

2. Find products that have not been ordered at all.

( <https://www.youtube.com/watch?v=PQogKI6_RQU> )

|  |  |  |
| --- | --- | --- |
| PRODUCTID | PRODUCTNAME | PRODUCTPRICE |
| 1 | Product A | 10 |
| 2 | Product B | 15 |
| 3 | Product C | 20 |
| 4 | Product D | 25 |
| 5 | Product E | 30 |
| 6 | Product F | 35 |
| 7 | Product G | 40 |
| 8 | Product H | 45 |
| 9 | Product I | 50 |
| 10 | Product J | 55 |

|  |  |  |  |
| --- | --- | --- | --- |
| ORDERID | PRODUCTID | QUANTITY | ORDERDATE |
| 1 | 1 | 2 | 01-07-2023 |
| 2 | 2 | 1 | 02-07-2023 |
| 3 | 3 | 5 | 03-07-2023 |
| 4 | 1 | 3 | 04-07-2023 |
| 5 | 5 | 2 | 05-07-2023 |
| 6 | 6 | 1 | 06-07-2023 |
| 7 | 2 | 4 | 07-07-2023 |
| 8 | 8 | 3 | 08-07-2023 |
| 9 | 9 | 1 | 09-07-2023 |
| 10 | 10 | 2 | 10-07-2023 |

Ans:

a)

select o.productid,

sum(quantity\*productprice) as revenue from orders\_2 o join products p on o.productid=p.productid

group by o.productid

b)

select p.productid,p.productname from products p left join orders\_2 o on o.productid=p.productid

where o.orderid IS NULL

### Find the Average Temperature for Each Day?

( <https://www.youtube.com/watch?v=v7eeooMgvBU>)

Source:

|  |  |  |  |
| --- | --- | --- | --- |
| READING\_ID | SENSOR\_ID | READING\_TIME | TEMPERATURE |
| 1 | 1 | 01-07-2024 08:00 | 25 |
| 2 | 1 | 01-07-2024 12:00 | 28 |
| 3 | 1 | 01-07-2024 16:00 | 29 |
| 4 | 2 | 02-07-2024 08:00 | 25 |
| 5 | 2 | 02-07-2024 12:00 | 26 |
| 6 | 2 | 02-07-2024 16:00 | 28 |
| 7 | 3 | 03-07-2024 08:00 | 24 |
| 8 | 3 | 03-07-2024 12:00 | 25 |
| 9 | 3 | 03-07-2024 16:00 | 28 |
| 10 | 1 | 04-07-2024 08:00 | 23 |

Expected Target:

|  |  |
| --- | --- |
| READING\_DATE | AVG\_TEMPERATURE |
| 01-07-2024 | 27.33 |
| 02-07-2024 | 26.33 |
| 03-07-2024 | 25.67 |
| 04-07-2024 | 23 |

Ans:

select date\_trunc(DAY,READING\_TIME::date) as reading\_date, cast(avg(temperature) as decimal(6,2)) as avg\_temperature from temperature\_readings

group by date\_trunc(DAY,READING\_TIME::date)

### Generate a report showing the average salary of employees hired each year since 2015.

( <https://www.youtube.com/watch?v=OzTYE8jur4k>)

Source:

|  |  |  |  |  |  |
| --- | --- | --- | --- | --- | --- |
| **EMPLOYEEID** | **FIRSTNAME** | **LASTNAME** | **DEPARTMENTID** | **SALARY** | **HIREDATE** |
| 1 | John | Doe | 101 | 5000 | 15-01-2022 |
| 2 | Jane | Smith | 102 | 6000 | 10-05-2021 |
| 3 | Michael | Johnson | 101 | 5500 | 01-03-2023 |
| 4 | Emily | Davis | 103 | 4800 | 20-09-2022 |
| 5 | David | Wilson | 101 | 5200 | 05-01-2023 |

Target :

|  |  |
| --- | --- |
| YEAR | AVG\_SAL |
| 01-01-2022 | 4900 |
| 01-01-2021 | 6000 |
| 01-01-2023 | 5350 |

ANS:

SELECT date\_trunc(YEAR,HIREDATE::DATE) as YEAR,AVG(SALARY) as AVG\_SAL FROM Employees\_2 where hiredate >= date\_trunc(YEAR,'2015-01-01'::DATE) group by date\_trunc(YEAR,HIREDATE::DATE)

### Transform the data as- the table contains two category of data called ‘ADULT’ and ‘CHILD’. Each and every adult has to be paired with the child.

### ( <https://www.youtube.com/watch?v=02XLUeIVRSE> )

Source:

|  |  |  |
| --- | --- | --- |
| **PERSON** | **TYPE** | **AGE** |
| A1 | Adult | 54 |
| A2 | Adult | 53 |
| A3 | Adult | 52 |
| A4 | Adult | 58 |
| A5 | Adult | 54 |
| C1 | Child | 20 |
| C2 | Child | 19 |
| C3 | Child | 22 |
| C4 | Child | 15 |

Target:

|  |  |  |
| --- | --- | --- |
|  | **ADULT** | **CHILD** |
| 1 | A4 | C4 |
| 2 | A5 | C2 |
| 3 | A1 | C1 |
| 4 | A2 | C3 |
| 5 | A3 | NULL |

Ans:

with cte\_adult as (

select \*, row\_number() over(order by person) rno from family where TYPE='Adult'

),

cte\_child as (

select \*,row\_number() over(order by person) rno from family where TYPE='Child'

)

select a.person as ADULT,b.person as CHILD from cte\_adult a left join cte\_child b on a.rno=b.rno

### 8) Write the SQL to extract the product where sales are increasing year to year.

### ( <https://www.youtube.com/watch?v=dWHSt0BVlv0> )

Source:

|  |  |  |  |
| --- | --- | --- | --- |
| PROD\_KEY | SALE\_YEAR | BRAND | AMOUNT |
| 1 | 31-12-2018 | apple | 45000 |
| 2 | 31-12-2019 | apple | 35000 |
| 3 | 31-12-2020 | apple | 75000 |
| 4 | 31-12-2018 | samsung | 15000 |
| 5 | 31-12-2019 | samsung | 20000 |
| 6 | 31-12-2020 | samsung | 25000 |
| 7 | 31-12-2018 | nokia | 21000 |
| 8 | 31-12-2019 | nokia | 17000 |
| 9 | 31-12-2020 | nokia | 15000 |

Expected Target:

|  |  |  |  |
| --- | --- | --- | --- |
| PROD\_KEY | SALE\_YEAR | BRAND | AMOUNT |
| 4 | 31-12-2018 | samsung | 15000 |
| 5 | 31-12-2019 | samsung | 20000 |
| 6 | 31-12-2020 | samsung | 25000 |

Query:

with cte as (

select \*,lead(amount,1,amount+1) over(partition by brand order by sale\_year) as LEAD\_AMT,

case when amount< lead(amount,1,amount+1) over(partition by brand order by sale\_year) then 1 else 0 END as FLAG from product\_sales

)

select \* from product\_sales where brand not in (select brand from cte where flag=0)

9) Remove duplicate

select sale\_year,brand,amount,row\_number() over(partition by sale\_year,brand,amount order by prod\_key) as rno from product\_sales

10)

(https://www.youtube.com/watch?v=UrIrBraLvZU)