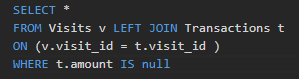
For checking null use ‘is null’:  


**Cartesian Product can also be used rather than join sometimes**

DATEDIFF function to check differences in dates:  
A black background with white text

Description automatically generated

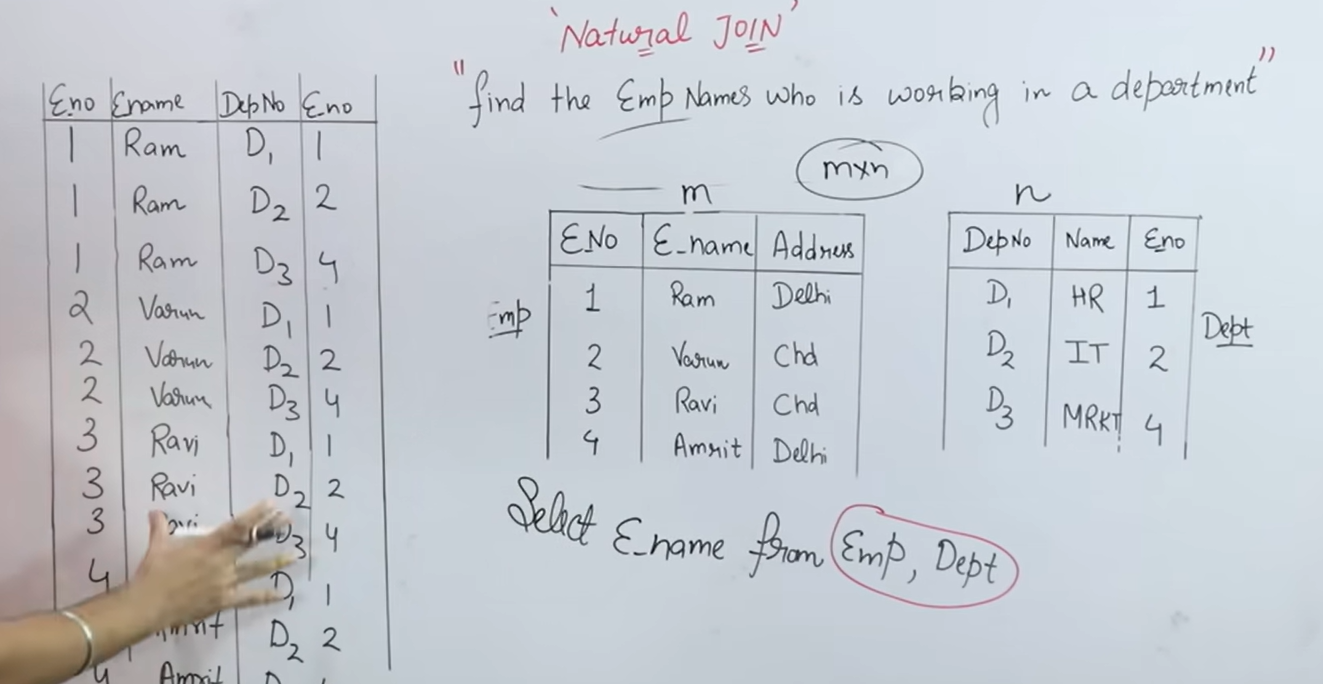
You can also apply conditions on joins and DATEADD function:

A screenshot of a computer program

Description automatically generated

Cross Product:

Multiply each row of one table from each row of other table



A close up of a white board

Description automatically generated

In Equi Join, we can compare any fields with each other.

In Natural Join, you apply join between the equal values.

A white board with black writing

Description automatically generated

A hand writing on a white board

Description automatically generated

This Query is wrong:  
A computer screen with colorful text

Description automatically generated

Table:  
A screenshot of a computer program

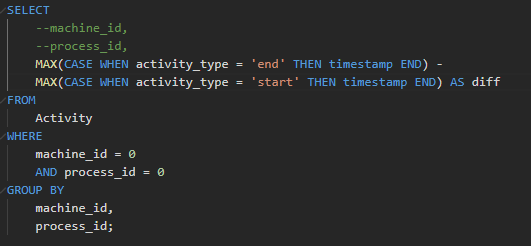
Description automatically generated

Because Select takes one row at a time, so activity\_time will be either start or close at a time, that’s why it won’t return any value.

Group by:

When we use group by, Select will pick bunch of values with it.

In SQL, when using GROUP BY, all columns in the SELECT statement that are not part of an aggregate function

The below Query will work because we are grouping them and then among the group we are choosing the max value of end and start:  


If we don’t apply MAX then SQL don’t know which activity\_type to pick out of all groups.

This is how group by on machine\_id internally looks:  
When you apply a `GROUP BY` on the `machine\_id` column in your Activity table, the internal data will be grouped by the `machine\_id`. Here's how your internal data will look like after grouping by `machine\_id`:

For machine\_id = 0:

+------------+------------+---------------+-----------+

| machine\_id | process\_id | activity\_type | timestamp |

+------------+------------+---------------+-----------+

| 0 | 0 | start | 0.712 |

| 0 | 0 | end | 1.520 |

| 0 | 1 | start | 3.140 |

| 0 | 1 | end | 4.120 |

+------------+------------+---------------+-----------+

For machine\_id = 1:

+------------+------------+---------------+-----------+

| machine\_id | process\_id | activity\_type | timestamp |

+------------+------------+---------------+-----------+

| 1 | 0 | start | 0.550 |

| 1 | 0 | end | 1.550 |

| 1 | 1 | start | 0.430 |

| 1 | 1 | end | 1.420 |

+------------+------------+---------------+-----------+

For machine\_id = 2:

+------------+------------+---------------+-----------+

| machine\_id | process\_id | activity\_type | timestamp |

+------------+------------+---------------+-----------+

| 2 | 0 | start | 4.100 |

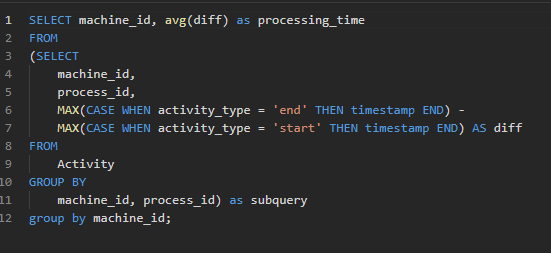
| 2 | 0 | end | 4.512 |

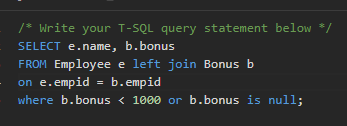
| 2 | 1 | start | 2.500 |

| 2 | 1 | end | 5.000 |

+------------+------------+---------------+-----------+

Each group represents the activities performed on each machine, with the `process\_id`, `activity\_type`, and `timestamp` information specific to that machine.



Applying the alias improves coding speed night and day:  


A screenshot of a graph

Description automatically generated

A screenshot of a computer program

Description automatically generated

A screenshot of a computer

Description automatically generated

Nested queries are slower than joins

**Cross Join:**

When you want to have all the rows from two tables then apply it.

For ex:- If you want to check the each students result for every subject whether if gave it or not then apply cross join between students and subjects and then left join to the marks

Two Solutions for finding an employee who is the manager of 5 people at least, Having use:  
A screenshot of a computer program

Description automatically generated

Commented one doesn’t pass the case when 2 manager IDs with the same names come.

Need to Find confirmed/total entries:  
A screenshot of a computer

Description automatically generated  
A computer screen shot of a black screen

Description automatically generated

Above we used case with if and else and can find total numbers of confirmed

Think about the table you need to find

Check odd even in where:

A screen shot of a computer code

Description automatically generated

A screenshot of a computer

Description automatically generated

A computer screen with white and blue text

Description automatically generated

Some times we just need a single value from the other table, apply nested query in this case:  
A screen shot of a computer

Description automatically generated

Medium Difficulty:  
A screenshot of a computer

Description automatically generated

Solution:  
A computer screen shot of a program

Description automatically generated

Notice group by CONVERT(VARCHAR(7), trans\_date, 120)

Extract year-month from the date [First 7 chars, 120 is yyyy-mm-dd format]

Need to find the orders with same day delivery:  
A screenshot of a computer

Description automatically generated

Solution with nested query and join:  
A computer screen with white text

Description automatically generated

Improved version:  
A computer screen shot of a program

Description automatically generated

Used AVG instead COUNT

Problem with this one is that in internal query I am using group by and due to that I can’t select all the columns and I need to perform a join in outer query to get all the columns

Using row\_num(), partition by, order by:

A screenshot of a computer

Description automatically generated  
A computer screen with white text

Description automatically generated

A screenshot of a computer

Description automatically generated

A computer screen with white text

Description automatically generated

DATEADD can be used for reduction of date: [- days]  
A black background with white text

Description automatically generated

Coded but TLE:  
select

s1.product\_id

, s1.year as first\_year,

s1.quantity,

s1.price

from sales s1

join (select

product\_id, min(year) as start\_year

from sales

group by product\_id) s2

on s1.year = s2.start\_year

and s1.product\_id = s2.product\_id

If using nested queries then it is mandatory to alias it: [alias t]  
A screenshot of a computer program

Description automatically generated

Delete duplicate values :  
A screenshot of a computer program

Description automatically generated

Table for this:  
A screenshot of a computer program

Description automatically generated

A screenshot of a computer

Description automatically generated

Need convert every word into camel case  
Substring is used to get the part of string:

substring(<var>, start\_index, <length till which you want the word>)

concat -> to merge the words

Extract month from date:  


A screenshot of a computer

Description automatically generated

Use of having after groupby

A screenshot of a computer

Description automatically generated

See count distinct and Groupconcat if I want all the values in a select all in a single cell

look out put at the end for help

A screenshot of a computer

Description automatically generated

Question:

A screenshot of a computer

Description automatically generated

A screenshot of a computer

Description automatically generated

My Code:  
A computer screen with text

Description automatically generated

Right and Left function to check part of strings, both starts from 1 index and needs the length till which you want to check as second param

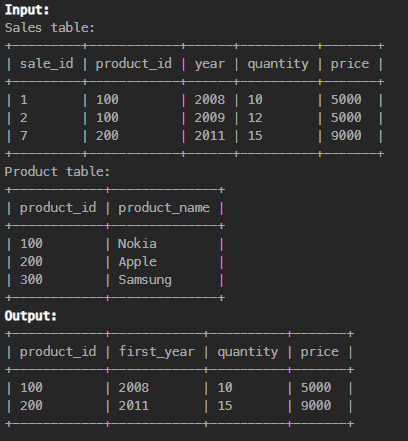
On the third condition check the use of not like with ^ which means if any symbol other than these fail the case

To check in a table where the value of manager has its own row or not:-

A screenshot of a computer

Description automatically generated  
A screenshot of a computer program

Description automatically generated

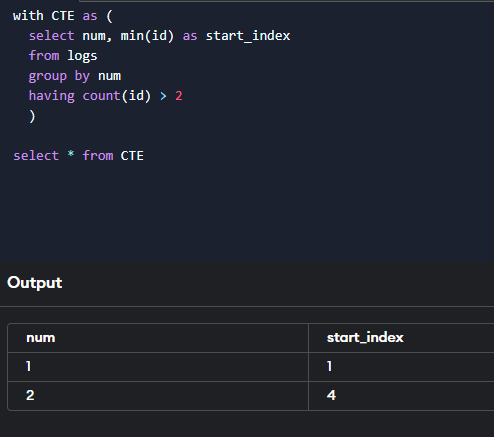
Needed to find the first year sales of each product  


Sometimes you can break the problem into multiple queries and then use join to get the final output:  
A computer screen with white text

Description automatically generated

CTE is a kind of python function:  
A screenshot of a computer program

Description automatically generated

Another example of CTE, Its really useful once you learn it’s just a function:  


Find the top 3 Salaries from each department when we have duplicate salaries:  
A screenshot of a computer

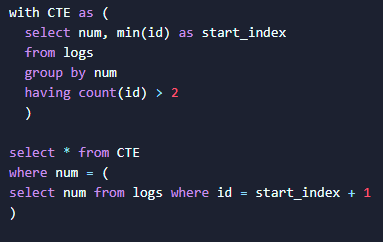
Description automatically generated  
  
I used CTE which is used as a python function and instead of

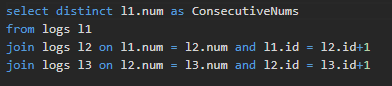
row\_number() -> Returns the rank of each row linearly

I used:  
dense\_rank() -> Return the same rank for duplicate values

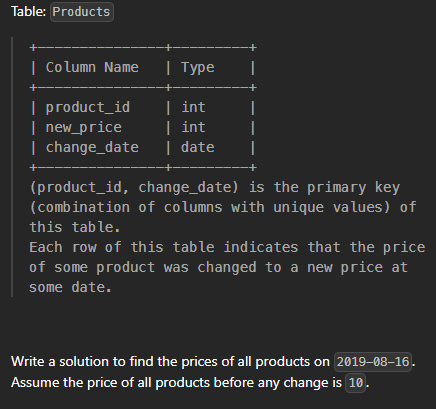
A computer screen shot of a black background

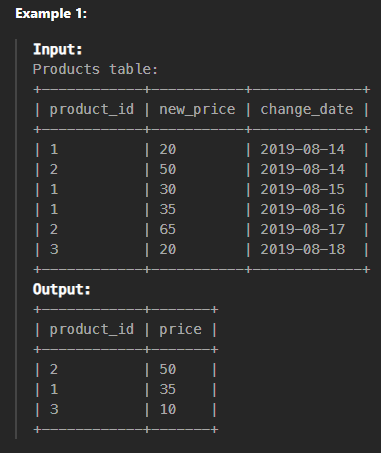
Description automatically generated

For Checking 2 consecutive number by returns only the 1st number with the 3 times occurance:  


**3 Consecutive numbers using join:**  


**Good Question:**

****

****/\* Write your T-SQL query statement below \*/

select distinct p.product\_id, case when date is null then 10 else p.new\_price end as price

from products p left join

(select product\_id, max(change\_date) as date

from Products

where change\_date <= '2019-08-16'

group by product\_id) s

on s.product\_id = p.product\_id

where s.date = p.change\_date or s.date is null

My Solution:  
WITH CTE AS (

    SELECT

        product\_id,

        MAX(change\_date) AS latest\_date

    FROM Products

    WHERE change\_date < '2019-08-17'

    GROUP BY product\_id

)

select

p1.product\_id

, p1.new\_price as price

FROM products p1

JOIN CTE

    ON p1.change\_date = CTE.latest\_date

    AND p1.product\_id = CTE.product\_id

**UNION ALL**

SELECT

    product\_id

    , 10 AS price

FROM Products

GROUP BY product\_id

HAVING MIN(change\_Date) > '2019-08-16'

If you give ans in this format each select becomes a column  
A black screen with white text

Description automatically generated

Output:  
A black background with white text

Description automatically generated

But if you give like this:  
A computer screen shot of text

Description automatically generated

Output:  
A screenshot of a computer

Description automatically generated

Notice the use of **UNION ALL** for converting columns to rows

**Over:**

It is used with window functions to provide a continuous flow of values instead of one single value.

When we use sum (), count (), etc. they return a single value but instead, if you want a cumulative total, for example if you want to calculate a moving average then over clause will be used.

Q. Find out the name of person from which total weight reaches 1000 kg:  
A computer screen shot of a code

Description automatically generated

**Coalesce:**

This function takes a list of values and takes the non-null from the given list.

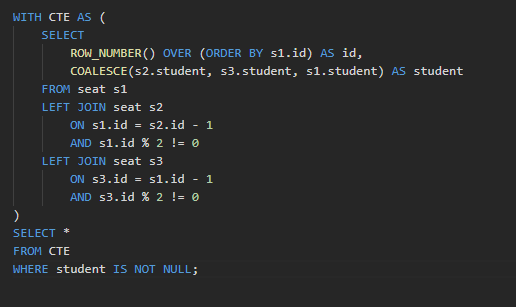
Q. Swap adjacent values and leave the last value as it is if total values are odd

A screenshot of a computer program

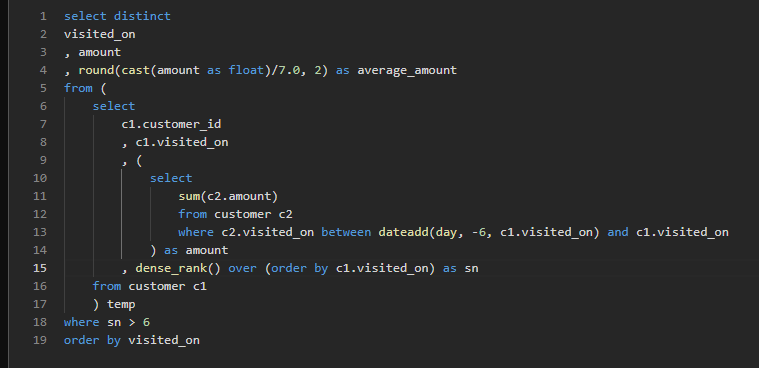
Description automatically generated

This is how the intermediate output was looking:  
A screenshot of a computer

Description automatically generated  
Needed coalesce method to combine the columns



Q. Need to find the moving average of average\_sell for the last 7 days in a hotel, but there can be multiple daily values.

Now I can’t apply over because we can’t apply conditions inside the over clause, so I did it with the subquery:  


**OFFSET Function:**In SQL Server, OFFSET is used to skip a specified number of rows before returning the result set. It is commonly used with the ORDER BY clause, especially in pagination scenarios, where you want to retrieve a subset of rows after skipping a certain number of initial rows.

If you want to skip the first 10 rows and retrieve the next 5 rows, you can do the following:

**A screenshot of a computer

Description automatically generated**

**Q.** Need to find the ID which was appearing most in both the columns

**Note:** I calculatedthesumon both the columns accepter\_id andrequester\_id but for combining the sum I had to use ISNULL(sum(column1), 0) + ISNULL(sum(column2), 0)

And applied **group by on two columns with coalesce**

****

**A screenshot of a computer program

Description automatically generated**