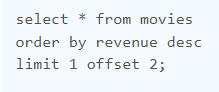
**ALL DAYS COMPILATION**

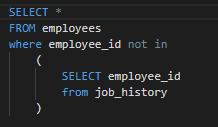
1. Write a query to find **all** the details of the **movie** that has the **third-highest** revenue.

* Use offset when needed…

 **OR** 

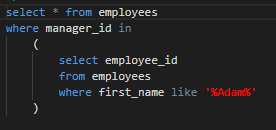
1. Display all the details of the employees who did **not work** at any job in the**past**.

* Referring the same table to establishing a condition…



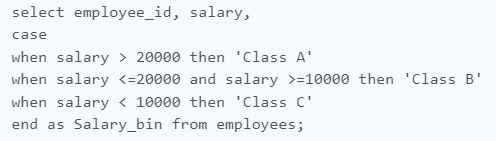
1. Write a query to display the employee details who report to **Adam**

* **Just like above…**



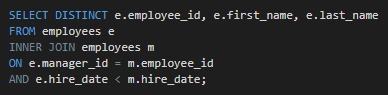
1. Based on the employee's salary, divide the employees into three different classes.

* Classic use of CASE…as you can see no commas or brackets just WHEN-THEN…END



1. Display the details of the employees who joined the company before their managers joined the company.

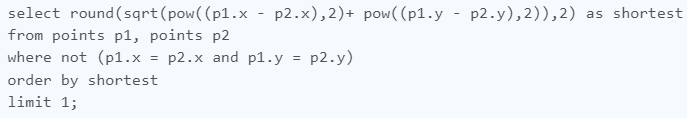
* Self-join to establish a condition



1. Write a query to find the shortest distance between any two points from the **points** table. Round the distance to two decimal points

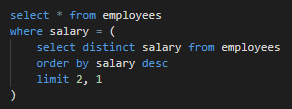
* Cross join so that every row is compared with every other & sort the distant using the classic distance formula - √ ((x2 - x1)2 + (y2 - y1)2)

|  |  |
| --- | --- |
| X | Y |
| 1 | 1 |
| 2 | 3 |
| 0 | 1 |



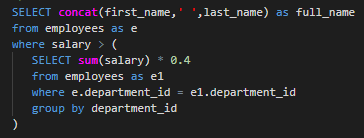
1. Write a query to find all the details of those employees who earn the **third-highest** salary

* Use Offset with sub-query…



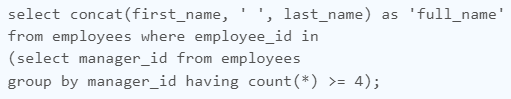
1. Display the employee's full name (first name and last name separated by space) as 'full name' of all those employees whose salary is greater than 40% of their department’s total salary.

* Just like above…



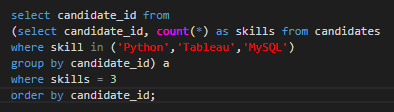
1. Display the 'full name' (first and last name separated by space) of a manager who manages 4 or more employees.

* Get the count by correct aggregation…

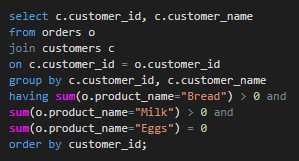


1. Given a table of candidates and their skills, you're asked to find the candidates best suited for an open Data Science job. We want to find candidates who are proficient in '**Python**', '**Tableau**', and '**MySQL**'.

Write a query to list the candidates who possess all three required skills for the job. Sort the output by **candidate\_id** in ascending order.

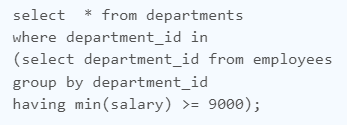


1. Write a query to find the **customer\_id** and **customer\_name** of customers who bought products "**Bread**", and "**Milk**" but did not buy the product "**Eggs**" since we want to recommend them to purchase this product.

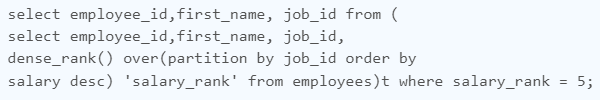


1. Display all the details of those departments where the salary of any employee in that department is **at least 9000**.

* Classic grouping by department…**having** is use to establish condition for aggregate functions

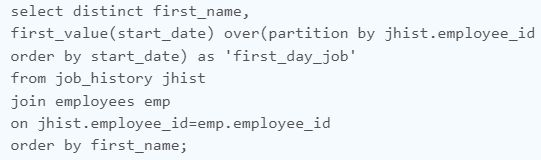


1. Show the details of the employees who have the 5th highest salary in each **job category**.



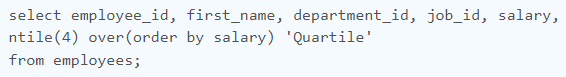
1. Write a Query to find the **first day of the first job** of every employee and return it as 'first\_day\_job'.

* First\_value & last\_value over a partition can help u in these cases…



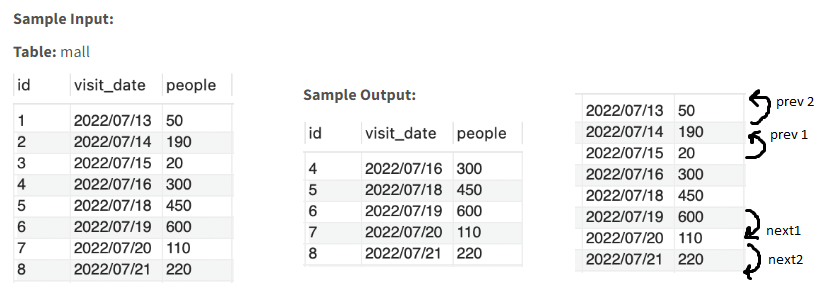
1. Find the **quartile** of each record based on the **salary** of the employee save as 'Quartile'.

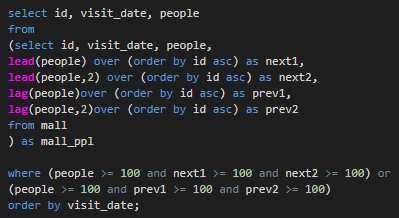
* Quartile is nothing but dividing salary into 4 buckets…NTILE () function is used here



1. Each row in the table contains the visit\_date and visit\_id to the mall with the number of people during the visit. No two rows will have the same visit\_date

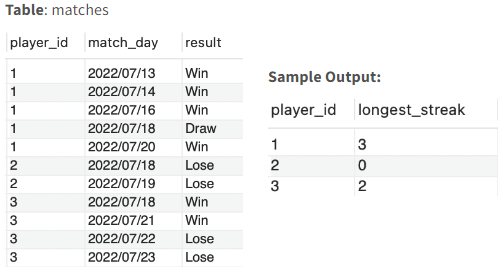
Write a query to display the records with **three or more** rows with consecutive id's, and the number of people is greater than or equal to 100 for each.

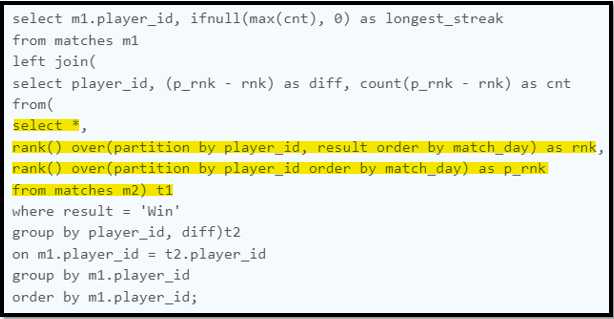




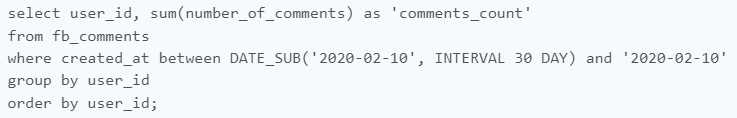
1. The winning streak of a player is calculated as the number of consecutive wins uninterrupted by draws or losses. Write a query to count the **longest winning streak** for **each** player and save the new column as 'longest\_streak'.

* Think harder on this question \*\*\*

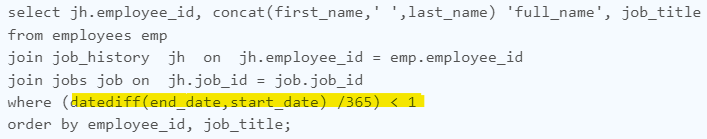




1. Write a query to calculate the total number of comments received for each user in the **30 or less days** before **2020-02-10** and save the column as 'comments\_count'

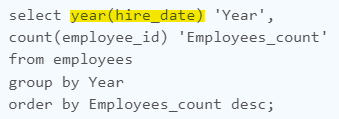


1. Display the details of the employees who had worked **less** **than** **a year**.



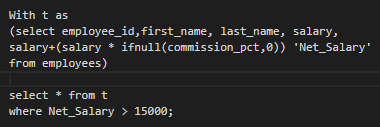
1. Display the **year** from the hire\_date as 'Year' and count the number of employees who joined in that year and save it as 'Employees\_count'

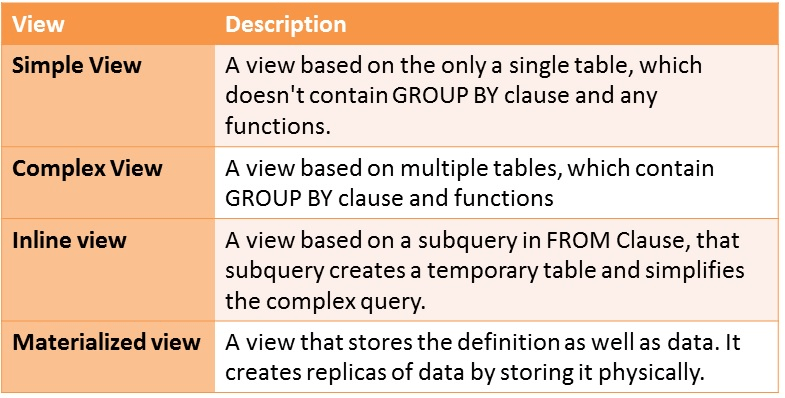
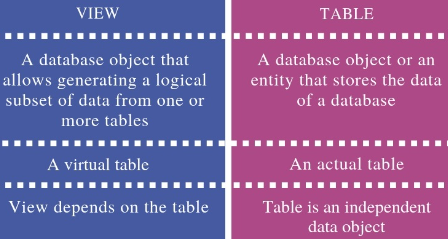
* Extracting **year**



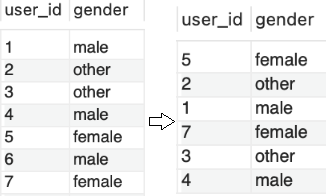
1. Calculate the net salary for the employees and save the column as '**Net salary**' and display the details of those employees whose net salary is greater than **15000**(Use the CTE method)

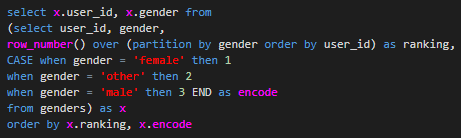
* Use of IFNULL & making CTE (**common table expression -** is a temporary named result set created from a simple SELECT statement that can be used in a subsequent SELECT statement)



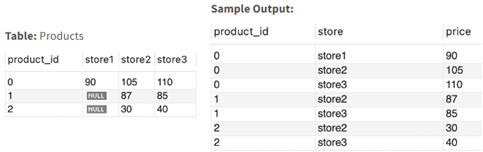


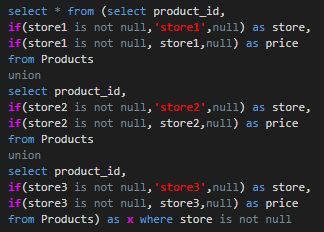
1. Write a query to reorder the entries of the **genders** table so that "**female**," "**other**," and "**male**" appear in that order in alternating rows. The table should be rearranged such that the IDs of each gender are sorted in ascending order.





1. **Transposing** the Table with substituting 0 with **NULL**





1. Write a query to calculate the total sales amount of **each item for each year**, with the corresponding product\_id, product\_name, and report\_year.

* Problem you would encounter would be to **segregate data** for each year e.g. for product 2 period start is 2019 & end is 2022 so data would be for 2019, 2020, 2021, 2022. To get this data say for 2019 we need to select that specific year data and **union** it with other years. To get this data we’ll get **DATEDIFF** between least (2019-12-31, period end) and greatest (2019-01-01, period start)

