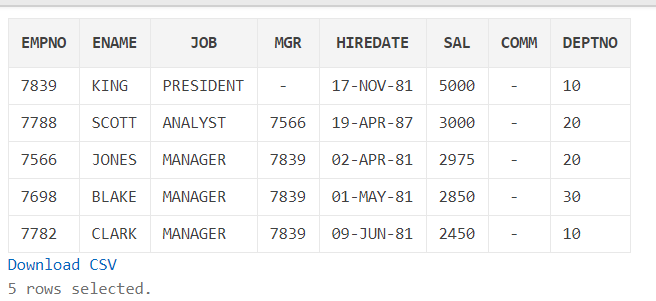
1. Show top 5 highest paid employees from EMP table.

select \* from scott.emp where rownum < 6 order by SAL desc



2. Display highest salary of the employees reporting to the same manager as the employee (Show the MAX Salary from same reporting manager group)

***Smartest Query would be:***

***select \* from scott.emp where SAL in (select Max(SAL) from scott.emp group by MGR) and MGR is not null***

Sol1)

select E.\* from scott.EMP E

inner join

(select max(SAL) S , MGR from scott.EMP group by MGR )Y

on Y.S = E.SAL

and Y.MGR = E.MGR

where E.MGR is not null

order by E.SAL DESC

**OR**

Sol2)

select E.\* from scott.EMP E

left outer join

(select max(SAL) salary, MGR from scott.EMP group by MGR) Y

on Y.salary = E.SAL

and Y.MGR = E.MGR

where (E.MGR is not null and Y.MGR is not null)

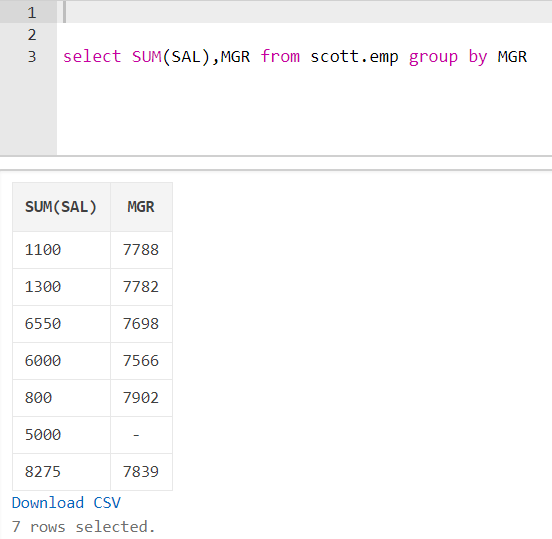
order by E.SAL DESC

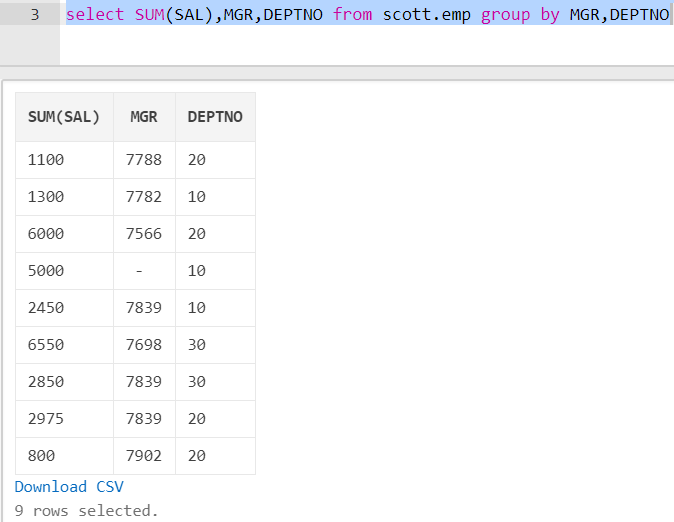
|  |  |  |  |  |  |  |  |
| --- | --- | --- | --- | --- | --- | --- | --- |
| **EMPNO** | **ENAME** | **JOB** | **MGR** | **HIREDATE** | **SAL** | **COMM** | **DEPTNO** |
| 7788 | SCOTT | ANALYST | 7566 | 19-APR-87 | 3000 | - | 20 |
| 7902 | FORD | ANALYST | 7566 | 03-DEC-81 | 3000 | - | 20 |
| 7566 | JONES | MANAGER | 7839 | 02-APR-81 | 2975 | - | 20 |
| 7499 | ALLEN | SALESMAN | 7698 | 20-FEB-81 | 1600 | 300 | 30 |
| 7934 | MILLER | CLERK | 7782 | 23-JAN-82 | 1300 | - | 10 |
| 7876 | ADAMS | CLERK | 7788 | 23-MAY-87 | 1100 | - | 20 |
| 7369 | SMITH | CLERK | 7902 | 17-DEC-80 | 800 | - | 20 |

3. Explain the below SQL clauses with an example (1 example for each is good enough).

1).GROUP BY

* The GROUP BY statement is often used with aggregate functions (COUNT, MAX, MIN, SUM, AVG) to group the result-set by one or more columns.
* If the query contains order by in that case group by should be used before the order by clause in the SQL query
* Group by should contain all the columns mentioned in the select clause except the aggregate functions used on the columns in the select clause
* Group by is generally used along with having
* It groups the data in the results sets into categories and displays and a brief information and its overview
* It is used for computing the group analysis on the data

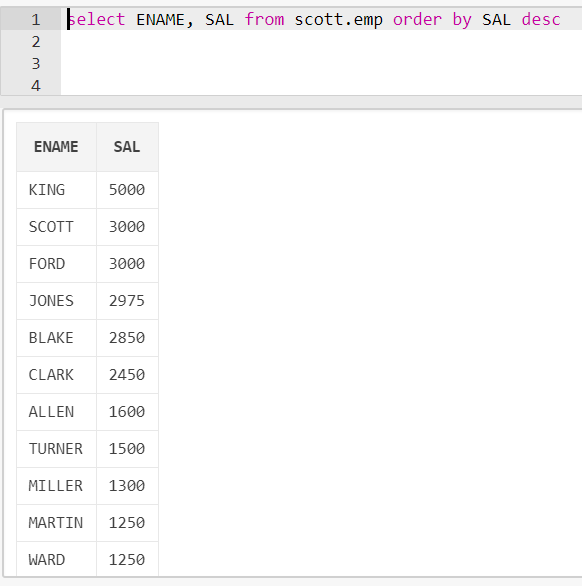




* SO when the group by columns increase the data result set rows increase and it gets grouped using 2 categories in this example.

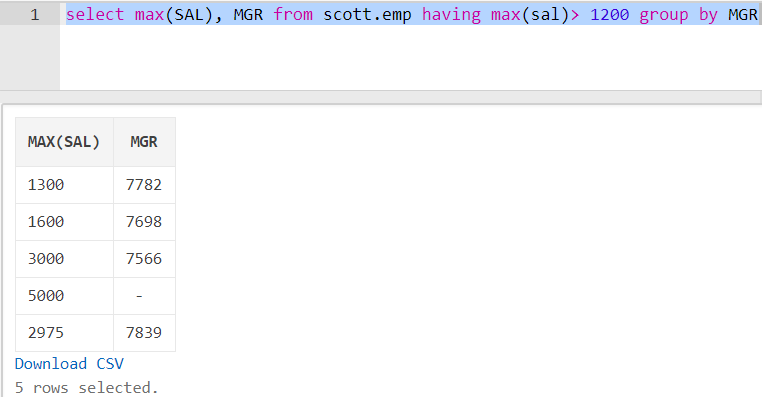
2). ORDER BY

* The ORDER BY keyword is used to sort the result-set in ascending or descending order.
* The ORDER BY keyword sorts the records in ascending order by default. To sort the records in descending order, use the DESC keyword.



3). HAVING BY :

* It is similar to Where Clause but having is specially used for computing the aggregate functions on the columns or attributes
* It is used in conjunction with the Group by clause
* HAVING filters records that work on summarized GROUP BY results.
* HAVING applies to summarized group records, whereas WHERE applies to individual records.
* Only the groups that meet the HAVING criteria will be returned.
* HAVING requires that a GROUP BY clause is present.
* WHERE and HAVING can be in the same query.



4. Display who is making highest commission from EMP table?

select \* from scott.emp where COMM = (select max(COMM) from scott.EMP)



5. search for employees with the pattern A\_B in their name (underscore is part of a name we need to find the employees which contain “\_” character in their names) . Example data is as below -

EMPNO ENAME

---------- ----------

7369 SMITH-A\_B

7499 A\_BLLEN

7521 WA\_BRD

7566 JAZBONES

7654 MARTIN

7698 BLAKE

7782 CLARK

7788 SCOTTAAB

7839 AAABBKING

7844 TURNER

7876 ADAMS

7900 JAMES

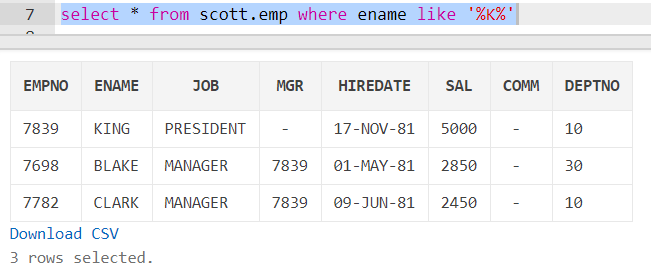
7902 FORD

7934 MILLER

* select \* from scott.emp where ename like '%A\_B%'

No data was found in the livesql

But just to provide an example



6. Display employee ID, Hire date, number of months employed from EMP table.

select EMPNO, HIREDATE, (ROUND((SYSDATE-HIREDATE)/365 ,0)\*12)MONTHS\_HIRED from scott.emp

|  |  |  |
| --- | --- | --- |
| **EMPNO** | **HIREDATE** | **MONTHS\_HIRED** |
| 7839 | 17-NOV-81 | 444 |
| 7698 | 01-MAY-81 | 456 |
| 7782 | 09-JUN-81 | 456 |
| 7566 | 02-APR-81 | 456 |
| 7788 | 19-APR-87 | 384 |
| 7902 | 03-DEC-81 | 444 |
| 7369 | 17-DEC-80 | 456 |
| 7499 | 20-FEB-81 | 456 |
| 7521 | 22-FEB-81 | 456 |
| 7654 | 28-SEP-81 | 444 |
| 7844 | 08-SEP-81 | 456 |
| 7876 | 23-MAY-87 | 384 |
| 7900 | 03-DEC-81 | 444 |
| 7934 | 23-JAN-82 | 444 |

7. Explain LEAD and LAG function with an example.

* Lag function helps to access values from previous records, and Lead function helps to access values from next records in the data set.
* The Structure of working with these functions is similar to each other. you would require to define the source column, offset, and the default value.
* Lead([source column],[offset],[default value])
* this means that Lead(OrderDate,1,null) will return the value in OrderDate column from the very next record’s in the recordset, and if that record doesn’t exists it will return null (because of the default value configuration)
* Lag function works exactly similar, you just need to set source column, offset, and default value.
* Lag and Lead both are working with OVER Statement. OVER statement structure is similar to row\_number() function. OVER(Partition By …. Order By …)
* Below shows an very simple example of using Lead and Lag functions to find previous and next order dates of each customer, and the duration between previous and current order, and also between current and next order in weeks.

select CustomerID,OrderDate as [Current Order Date]

,lag(orderdate,1,null) over(partition by customerid order by orderdate) [Previous Order Date]

,datediff(week,lag(orderdate,1,null) over(partition by customerid order by orderdate),OrderDate) [Duration from Previous

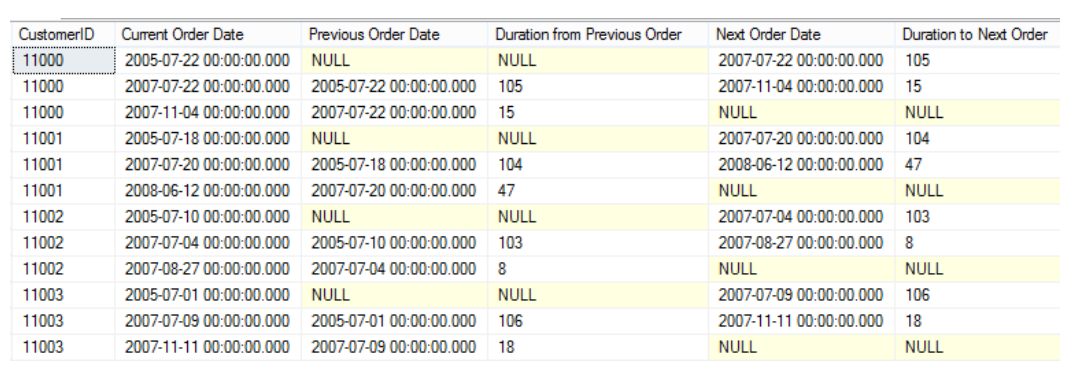
Order]

,lead(orderdate,1,null) over(partition by customerid order by orderdate) [Next Order Date]

,datediff(week,orderdate,lead(orderdate,1,null) over(partition by customerid order by orderdate)) [Duration to Next Order]

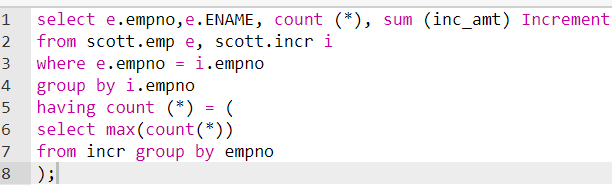
from sales.SalesOrderHeader

order by CustomerID

This is a sample result set of this script: 

8. List employee number, name, Total number of Increments and total increment amount for the employee who has got maximum number of increments

* Since the table **increment** table is missing in livesql.com we have just provided the query that can return the employee details with the highest number of increments.



9. Display Salary Grades for each employee from EMP table. Details of SALGRADE table are as below –

Table: SALGRADE

Name Datatype

------- ------------------

GRADE NUMBER

LOSAL NUMBER

HISAL NUMBER

SQL> select \* from salgrade;

GRADE LOSAL HISAL

---------- ---------- ----------

1 700 1200

2 1201 1400

3 1401 2000

4 2001 3000

5 3001 9999

select e.ENAME, e.SAL,

CASE

when e.SAL > 3000 Then 'GRADE 5'

when e.SAL BETWEEN 2001 and 3000 then 'Grade 4'

when e.SAL BETWEEN 1401 and 2000 then 'Grade 3'

when e.SAL BETWEEN 1201 and 1400 then 'Grade 2'

else 'Grade 1'

END AS COMMENTS from scott.emp e

|  |  |  |
| --- | --- | --- |
| **ENAME** | **SAL** | **COMMENTS** |
| KING | 5000 | GRADE 5 |
| BLAKE | 2850 | Grade 4 |
| CLARK | 2450 | Grade 4 |
| JONES | 2975 | Grade 4 |
| SCOTT | 3000 | Grade 4 |
| FORD | 3000 | Grade 4 |
| SMITH | 800 | Grade 1 |
| ALLEN | 1600 | Grade 3 |
| WARD | 1250 | Grade 2 |
| MARTIN | 1250 | Grade 2 |
| TURNER | 1500 | Grade 3 |
| ADAMS | 1100 | Grade 1 |
| JAMES | 950 | Grade 1 |
| MILLER | 1300 | Grade 2 |

[Download CSV](https://livesql.oracle.com/apex/f?p=590:1:101660989614262::NO:RP::)

10. Write a query to display employee names right aligned. The output should look as below(I am showing only few rows as example) -

ENAME    NEW\_ENAME\_FORMAT

---------- ----------------

SMITH SMITH

JAZBONES JAZBONES

MARTIN   MARTIN

BLAKE        BLAKE

CLARK        CLARK

* SELECT ename,

LPAD(ename,30,' ') New\_NAME

FROM scott.emp

