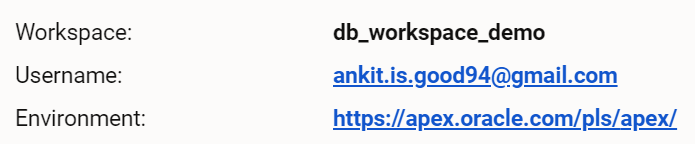
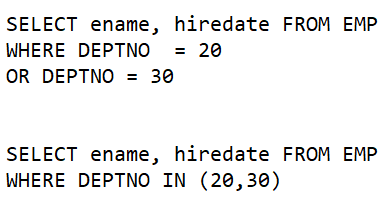
1. Database Basics
   1. A database is a place where you store data.
   2. A database is collections of disks
   3. SQL (Structural Query Language) is a set of commands that can be sent to the database.
   4. Databases interpret what these commands are and if they’re formatted correctly, if the syntax is correct, its going to perform some operations
   5. So, these commands are instructions to the database on what to do.
   6. Access database online <https://apex.oracle.com/>

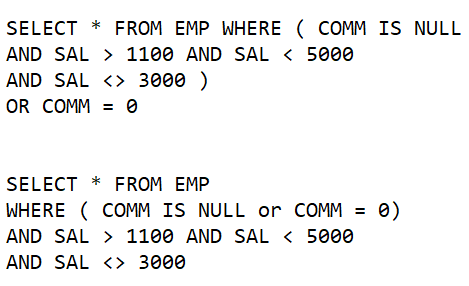


* 1. Install datasets

1. Single Table Queries
   1. SQL is not case sensitive
   2. Semicolon at the end of every statement
   3. SELECT and FROM are keywords in SQL
   4. SQL is not case sensitive, when it comes to comparing data.
   5. Select Clause
      1. SELECT \* FROM emp; (select all columns)
      2. SELECT <column> FROM <table>; (select specific columns)
      3. SELECT DISTINCT job FROM emp; (select distinct job column from emp table)
   6. Where Clause
      1. SELECT \* FROM EMP WHERE job = 'MANAGER'
      2. SELECT \* FROM EMP WHERE JOB = 'SALESMAN' AND SAL = 1600 AND COMM = 300 AND DEPTNO = 30
   7. Using Operators in the WHERE clause (<, >, =, !=, <=, >=, <>)
      1. SELECT \* FROM EMP WHERE JOB != 'SALESMAN' (Not Equal operator)
      2. SELECT \* FROM EMP WHERE JOB = 'SALESMAN' AND SAL = 1500 (AND operator)
      3. SELECT \* FROM EMP WHERE JOB != 'SALESMAN' AND SAL < 2500 (less than operator)
      4. SELECT \* FROM EMP WHERE JOB != 'SALESMAN' AND SAL <= 2500 (less than equal operator)
      5. SELECT \* FROM EMP WHERE JOB != 'SALESMAN' AND SAL > 2500 (greater than operator)
      6. SELECT \* FROM EMP WHERE COMM > SAL (commission is greater than salary)
      7. SELECT \* FROM EMP WHERE JOB != 'MANAGER' AND SAL > 2500 AND DEPTNO = 20
   8. Combining WHERE, AND & OR with operators
      1. SELECT \* FROM EMP WHERE JOB = 'CLERK' OR JOB = 'SALESMAN' (OR operator)
      2. SELECT ENAME FROM EMP WHERE JOB != 'MANAGER' AND JOB != 'SALESMAN' AND SAL >= 2000 (AND operator)
   9. Query Filtering with BETWEEN, IN, NOT and NULL
      1. SELECT ename, hiredate FROM EMP WHERE DEPTNO IN (20,30) {**IN operator**}
      2. Both below queries return same data. So, IN operator can be used in place of multiple OR operator

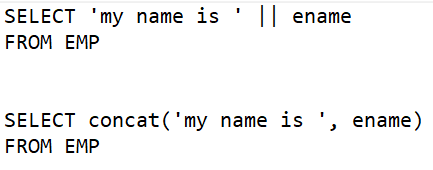


* + 1. SELECT ename, hiredate FROM EMP WHERE ENAME IN ('FORD', 'SMITH', 'ALLEN', 'WARD', 'MARTIN')
    2. SELECT ename, hiredate FROM EMP WHERE ENAME NOT IN ('FORD', 'SMITH', 'ALLEN', 'WARD', 'MARTIN') (**NOT IN operator**)
    3. SELECT \* from emp where hiredate between '05/01/1981' and '12/09/1982' (**BETWEEN operator**)
    4. SELECT \* from emp where SAL NOT BETWEEN 950 AND 1600 (**NOT BETWEEN operator**)
    5. SELECT \* FROM EMP WHERE COMM IS NOT NULL; (**NULL operator**)
    6. SELECT \* FROM EMP WHERE MGR IS NULL

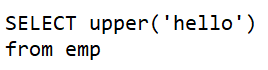
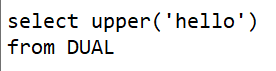


* 1. Query Filtering Conditions and Operator Precedence
     1. SELECT \* FROM EMP WHERE JOB = 'SALESMAN' AND (COMM = 300 OR COMM > 1100)
     2. SELECT \* FROM EMP WHERE JOB LIKE 'S%' (job start with ‘S’) (**LIKE operator**)
     3. SELECT \* FROM EMP WHERE JOB LIKE '%GER' (job end with ‘GER’)
  2. Ordering, Concatenating, & Aliasing Query Results
     1. SELECT ENAME "EMPLOYEE NAME", SAL SALARY, COMM COMMISSION FROM EMP; (aliasing column names)
     2. SELECT 'hello my name is ' || ename as "concatenated value" FROM EMP WHERE job = 'MANAGER' (Concatenating text with column)
     3. SELECT 'Salary: ' || sal as "Salary" FROM EMP WHERE job = 'MANAGER'
     4. SELECT ename || ' makes $' || sal || ' per month' as "employee income" FROM EMP (King makes $5000 per month)
     5. SELECT \* FROM emp ORDER BY ENAME (order by column)
     6. SELECT \* FROM emp ORDER BY SAL
     7. SELECT \* FROM emp ORDER BY SAL DESC (ORDER BY descending order)
     8. SELECT \* FROM emp ORDER BY SAL ASC (ORDER BY ascending order)
     9. SELECT DEPTNO, SAL, ENAME FROM EMP ORDER BY DEPTNO, SAL
     10. SELECT \* FROM EMP ORDER BY DEPTNO, SAL

1. Single Row Functions
   1. All rows convert into same number of rows
   2. Single Row Functions and Using The Dual Table
      1. A function is a predefined program that performs some task.
      2. The concat() function

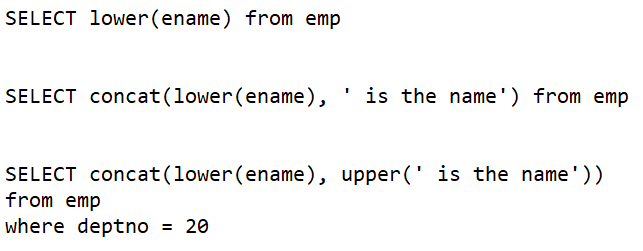


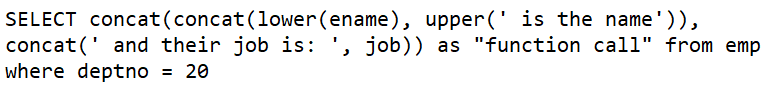
* + 1. The upper() function

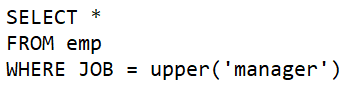
* + 1. The dual table is special table provided by Oracle.
    2. The lower() function



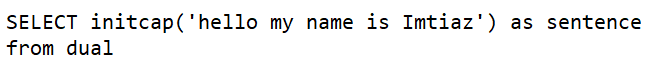




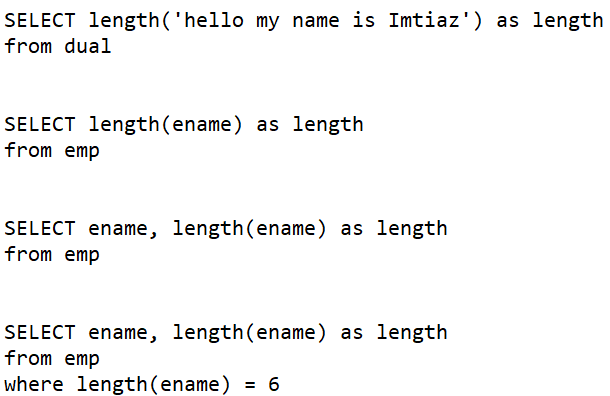
* 1. Using Functions in WHERE and Character Based SRFs
     1. Single role functions can not only be used in the select clause, you can also use single row functions in filtering conditions of the where clause.



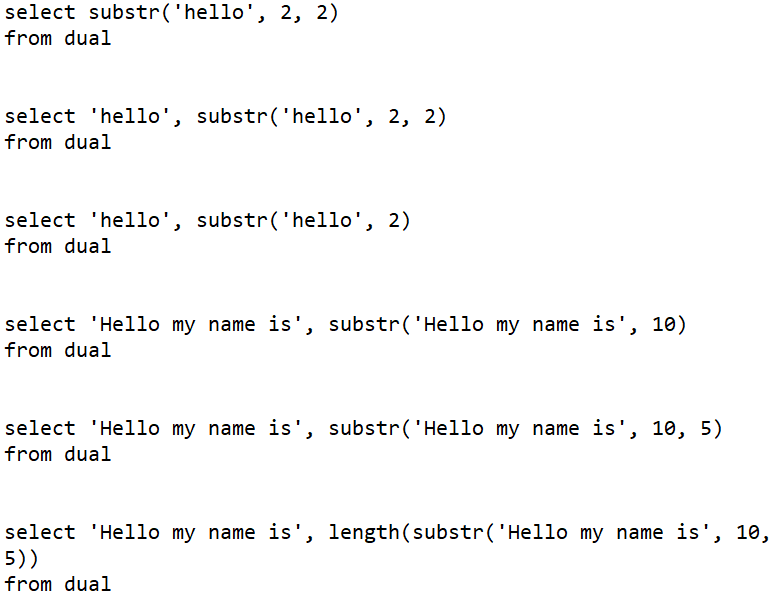
* + 1. The initcap() function: make first char capital in each word



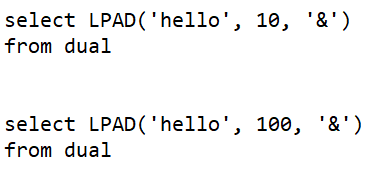
* + 1. The length() function



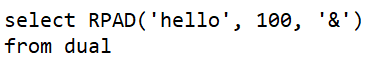
* + 1. The substr() function



* + 1. The LPAD() function: add ‘&’ 10 times before ‘hello’



* + 1. The RPAD() function: add ‘&’ 100 times after ‘hello’



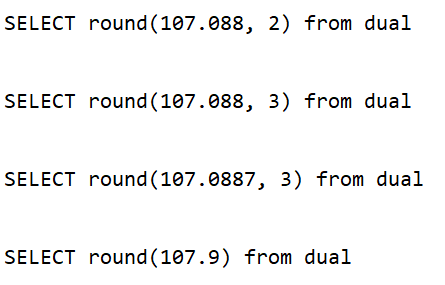
* + 1. The LTRIM() function: remove all ‘h’ from start



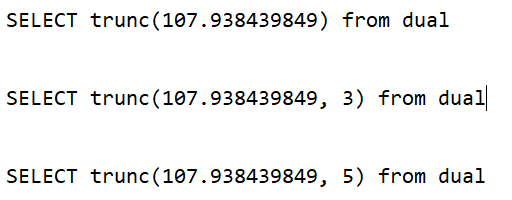
* + 1. The RTRIM() function: remove all ‘h’ from end



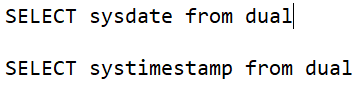
* + 1. <https://www.techonthenet.com/oracle/index.php>
  1. Numeric and Date Data types SRFs
     1. The round() function: rounding decimal place



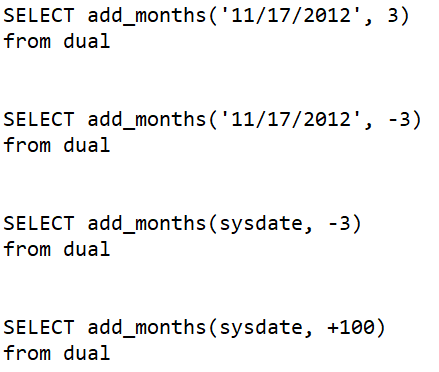
* + 1. The trunc() method: remove decimal place



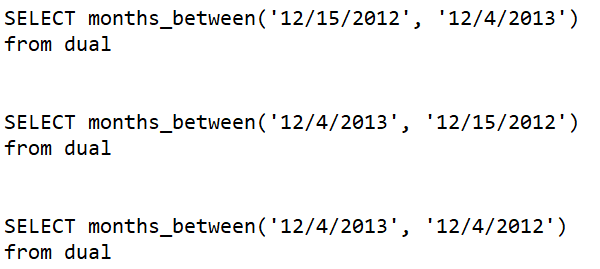
* + 1. Get current sysdate or systimestamp



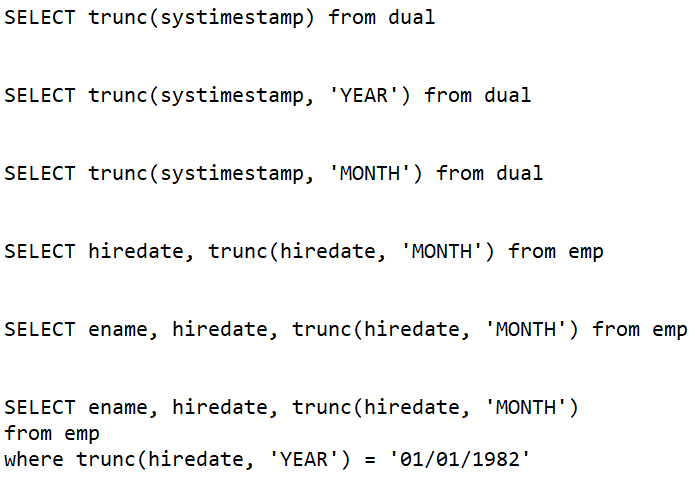
* + 1. The add\_months() function: add month in date



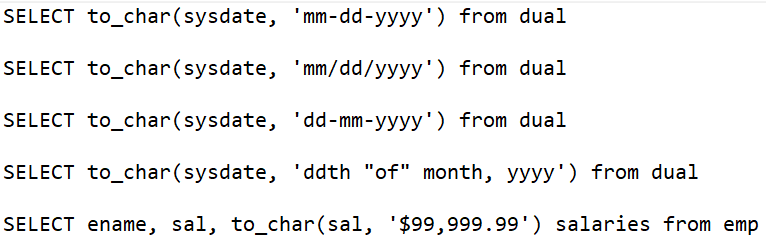
* + 1. The month\_between() function: month between two dates



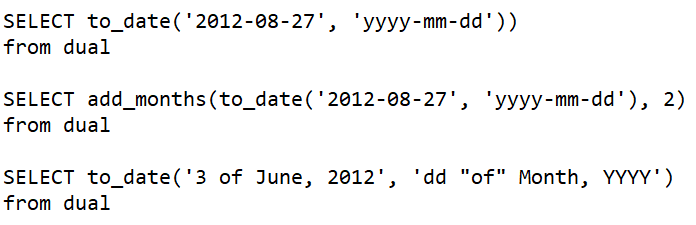
* + 1. The trunc() function with date



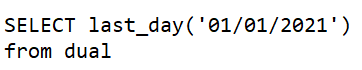
* 1. Conversion SRFs and Date Formatting
     1. The to\_char() function: this function is used to convert a date or a number to a character string and format date or number.



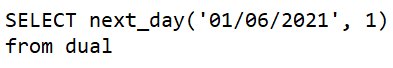
* + 1. The to\_date() function: used to format date



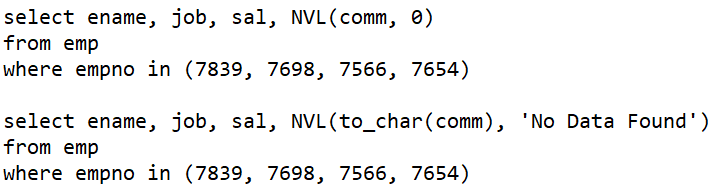
* + 1. The last\_day() function: get last day of month



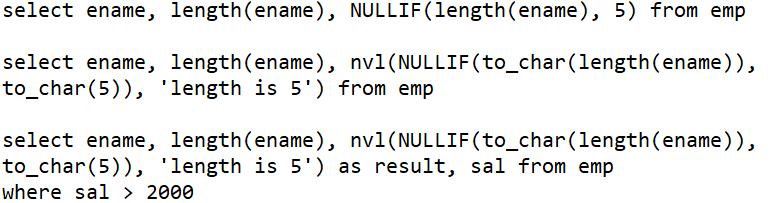
* + 1. The next\_day() function: The first argument is the date and the second argument is a text reference to a day of the week.



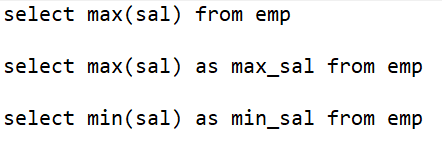
* 1. NVL and NULLIF Functions
     1. In Oracle, NULL and empty is same.
     2. The NVL() function: first argument is the column that we want to replace with a more meaningful value rather than just an empty cell.

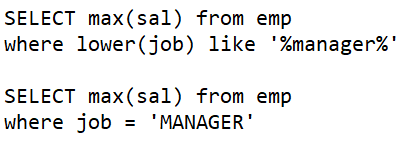


* + 1. The NULLIF() function: if argument one, whatever it is, if that's equal to argument two, then the output of this function call is actually going to be null.



1. Grouping Functions
   1. All rows from table convert into one row
   2. Grouping funtions
      1. The max() and min() function





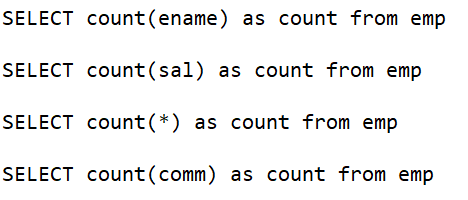
* + 1. The sum() function

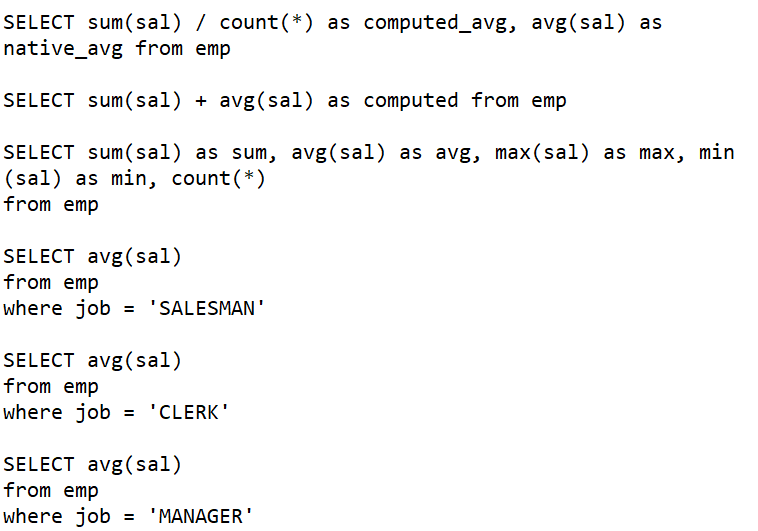


* + 1. The avg() function

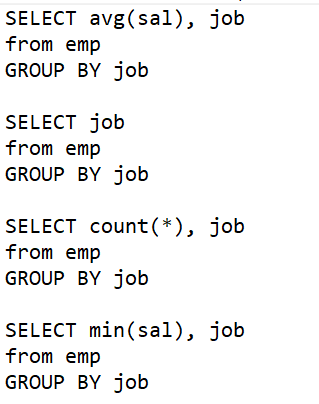


* + 1. The count() function

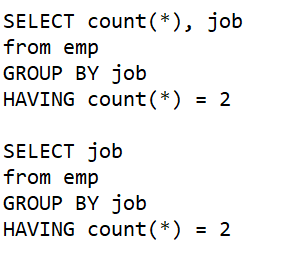


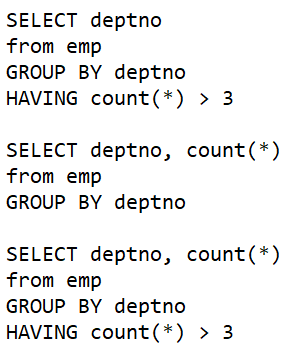


* 1. GROUP BY Clause & HAVING Clause
     1. GROUP BY Clause: group row based on duplicate value

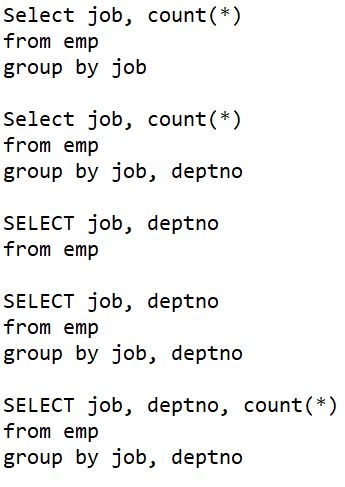


* + 1. HAVING Clause

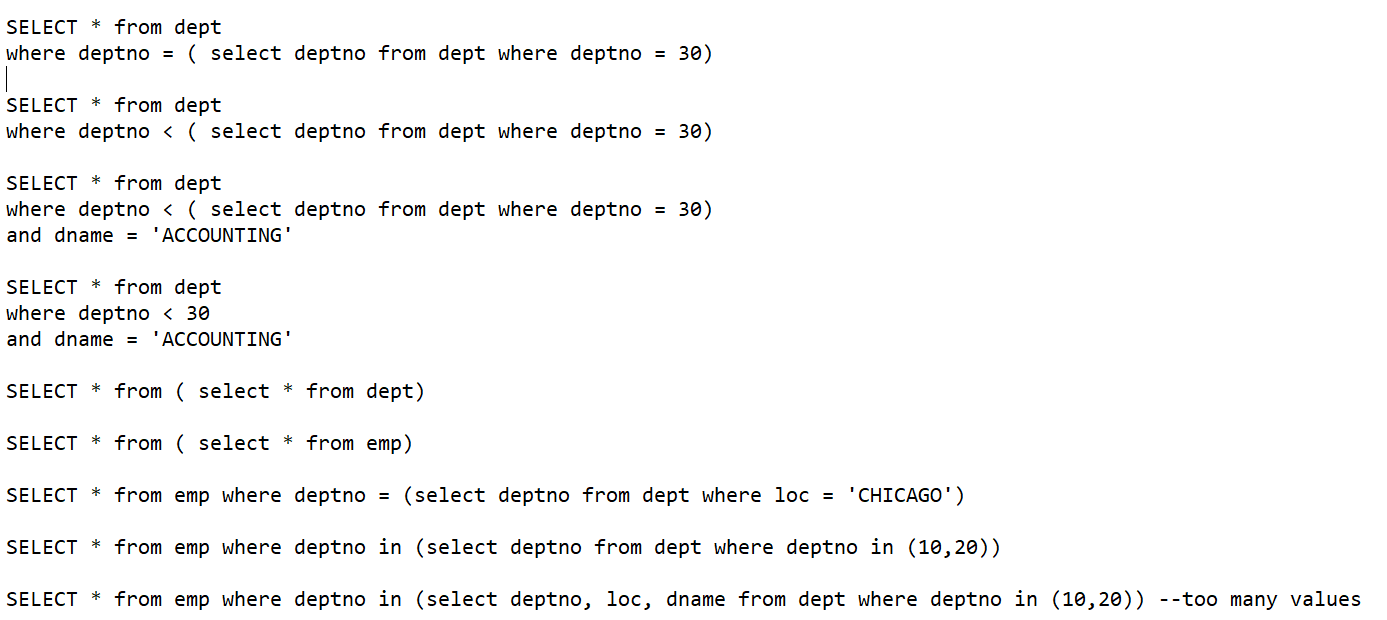


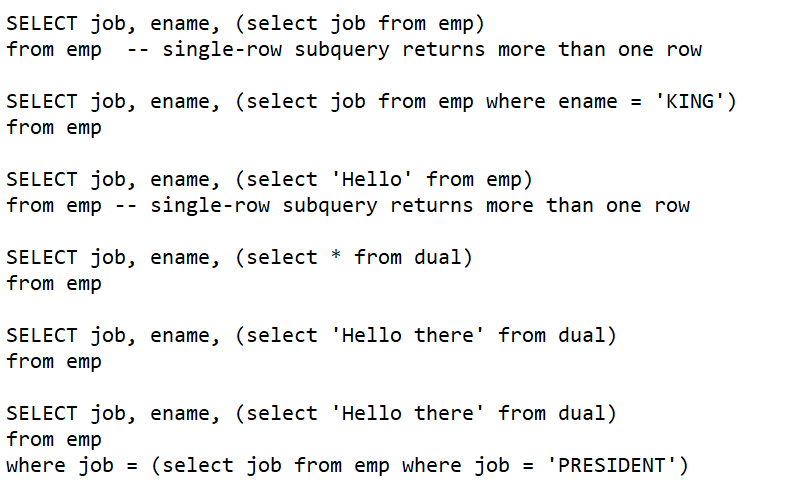


* + 1. GROUP BY with multiple column

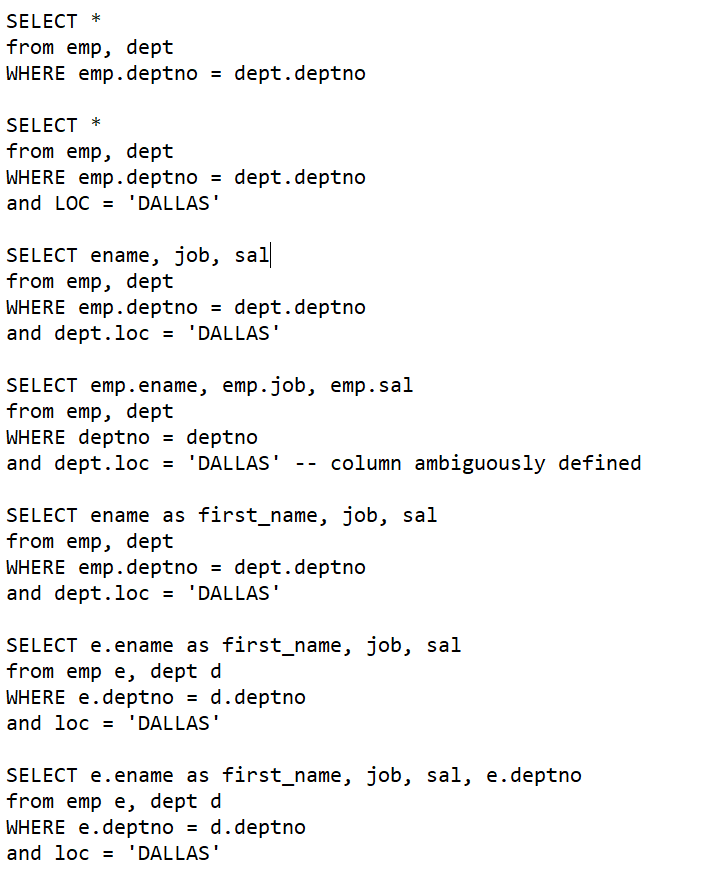


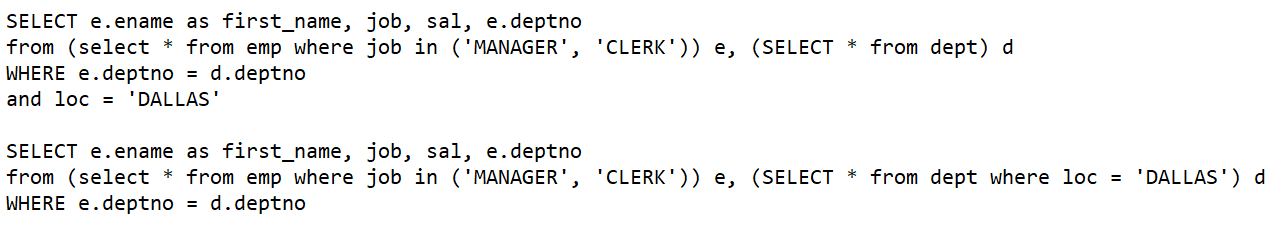
1. Multi-Table Queries and Joins
   1. Inner Queries and Sub Queries: Inner queries execute first after that outer executes.





* 1. Relating Tables Together Using JOINs





* 1. Inner and Outer Join

SELECT \*

from emp, dept

WHERE emp.deptno = dept.deptno

SELECT \*

from emp INNER JOIN dept

ON emp.deptno = dept.deptno

SELECT \*

from emp RIGHT JOIN dept

ON emp.deptno = dept.deptno

SELECT \*

from emp LEFT JOIN dept

ON emp.deptno = dept.deptno

SELECT \*

from dept LEFT JOIN emp

ON emp.deptno = dept.deptno

SELECT \*

from dept RIGHT JOIN emp

ON emp.deptno = dept.deptno

SELECT \*

from emp RIGHT OUTER JOIN dept

ON emp.deptno = dept.deptno

SELECT \*

from emp LEFT OUTER JOIN dept

ON emp.deptno = dept.deptno

SELECT \*

from emp, dept

WHERE emp.deptno(+) = dept.deptno --RIGHT OUTER JOIN

SELECT \*

from emp, dept

WHERE emp.deptno = dept.deptno(+) --LEFT OUTER JOIN

SELECT \*

from emp FULL OUTER JOIN dept

ON emp.deptno = dept.deptno

SELECT \*

from (select \* from emp) emp FULL OUTER JOIN dept

ON emp.deptno = dept.deptno

SELECT \*

from (select \* from emp) e FULL OUTER JOIN dept

ON e.deptno = dept.deptno

SELECT \*

from (select \* from emp) e FULL OUTER JOIN (select \* from dept) d

ON e.deptno = d.deptno

SELECT \*

from (select \* from emp) e, (select \* from dept) d

WHERE e.deptno = d.deptno

SELECT \*

from (select \* from emp where job = 'SALESMAN') e FULL OUTER JOIN (select \* from dept) d

ON e.deptno = d.deptno

* 1. Correlated sub queries with exists and not exists operators

-- Solution to Assignment from previous video

SELECT empno, ename, job, mgr, hiredate, sal, comm, e.deptno as deptno, d.deptno as deptno, dname, loc

from (select \* from dept) d LEFT OUTER JOIN (select \* from emp where job = 'SALESMAN') e

ON e.deptno = d.deptno

SELECT e.\*, d.\*

from (select \* from dept) d LEFT OUTER JOIN (select \* from emp where job = 'SALESMAN') e

ON e.deptno = d.deptno -- Further Simplified Version

SELECT \*

from emp

WHERE EXISTS (select 'random' from dual)

SELECT \*

from emp

WHERE NOT EXISTS (select 'random' from dual)

SELECT \*

from emp

WHERE EXISTS (select null from dual)

SELECT \*

from emp

WHERE NOT EXISTS (select \* from emp where job = 'PROGRAMMER')

SELECT d.\*

FROM dept d

WHERE EXISTS (SELECT \* FROM emp WHERE d.deptno = emp.deptno)

SELECT d.\*

FROM dept d

WHERE NOT EXISTS (SELECT \* FROM emp WHERE d.deptno = emp.deptno)

AND LOC = 'CHICAGO' -- no data found

SELECT d.\*

FROM dept d

WHERE NOT EXISTS (SELECT \* FROM emp WHERE d.deptno = emp.deptno)

AND LOC = 'BOSTON' -- no data found

SELECT d.\*

FROM dept d

WHERE NOT EXISTS (SELECT \* FROM emp WHERE d.deptno = emp.deptno)

OR LOC = 'CHICAGO'

* 1. Self join

select e.ename, m.ename

from emp e, emp m;

select e.ename, m.ename

from emp e, emp m

where e.empno = m.empno;

select e.ename, m.ename

from emp e, emp m

where e.mgr = m.empno;

select e.ename as employee, m.ename as manager

from emp e LEFT JOIN emp m

on e.mgr = m.empno;

* 1. Cross join, natural join and the using clause

select \*

from emp, dept;

select \*

from emp CROSS JOIN dept;

select ename, mgr, loc

from emp CROSS JOIN dept;

select ename, mgr, loc

from emp NATURAL JOIN dept;

select ename, mgr, loc

from emp JOIN dept using (deptno);

select ename, mgr, loc

from emp LEFT JOIN dept using (deptno);

select ename, mgr, loc

from emp RIGHT JOIN dept using (deptno);

* 1. EQUI-Join and Non EQUI-Joins

select e.ename, e.sal, j.grade\_level

from emp e join job\_grade j

on e.sal between j.lowest\_sal and j.highest\_sal;

* 1. Case statement

SELECT ename, job, (CASE job

WHEN 'PRESIDENT' THEN 'big shot'

WHEN 'MANAGER' THEN 'decides the pay'

WHEN 'ANALYST' THEN 'good at pay'

WHEN 'CLERK' THEN 'hard working'

ELSE 'no comment'

END) as 'COMMENT'

FROM emp;