if null

if else

trim

update

select, insert

create table table\_name (column name datatype constraints, ....)

Sub Query

SET Operator

Sub Query

What is a sub Query ?

Why we need sub queries?

Reat time examples?

Main Query --- > Which employees have salaries greater

than Abel’s salary?

Sub Query ----- > What is Abel’s salary?

Syntax

Main query ( sub queryy - single or multiple rows)

Output of the subquery is an input to the main query.

SELECT select\_list FROM table WHERE expr operator

(SELECT select\_list FROM table);

Sub query example

SELECT last\_name, salary

FROM employees

WHERE salary >......

(SELECT salary

FROM employees

WHERE last\_name = 'Abel');

nesting ---- ( statement / query (query (...)))

Types of Subqueries

* Single row subquery  : a sub query which returns only one result
* Multiple row subquery : a sub query which returns multiple results

Single-Row Subqueries

– Return only one row

– Use single-row comparison operators  =, >, >=, <, <=, <>

SELECT last\_name, job\_id, salary

FROM employees

WHERE job\_id =

(SELECT job\_id

FROM employees

WHERE employee\_id = 141)

AND salary >

(SELECT salary

FROM employees

WHERE employee\_id = 143);

Different ways of using  single- row subqueries

>>>>>>>>>>>>>>>   Using Group Functions in a Subquery

SELECT last\_name, job\_id, salary

FROM employees

WHERE salary =

(SELECT MIN(salary)

FROM employees);

>>>>>>>>>>>>>>  The HAVING Clause with Subqueries

fetch all records in employees which have min salary greater than the minimum of dept 50

select min(sal) from employees where deptid = 50

SELECT department\_id, MIN(salary) FROM employees GROUP BY department\_id

HAVING MIN(salary) > (SELECT MIN(salary) FROM employees WHERE department\_id = 50);

Different ways of using  multiple - row subqueries with IN, ANY, ALL

...... IN (a, b, c)

.......any ( any of the conditions matching)

condition1  and condition2    if both return true only then it is true

and  truth table  1 = true   0 = false

   condition1       condition2   and           OR

      1                      1                1               1

       1                      0                0               1

       0                       1                0               1

       0                       0                0               0

IN, ANY, ALL checking for matching values

in ( values )

SELECT employee\_id, last\_name, job\_id, salary

FROM employees

WHERE salary < ANY

(SELECT salary

FROM employees

WHERE job\_id = 'IT\_PROG')

AND job\_id <> 'IT\_PROG';

SELECT employee\_id, last\_name, job\_id, salary

FROM employees

WHERE salary < ALL

(SELECT salary

FROM employees

WHERE job\_id = 'IT\_PROG')

AND job\_id <> 'IT\_PROG';

Few examples for subqueries

# Fetch all the departments based on location = ‘ california’

Select \* from locations;

Select location\_id from locations where state\_province = ‘california’;

Select \* from departments where location-id =

(Select location\_id from locations where state\_province = ‘california’);

# Fetch all employees who has job\_id same as employee with id 141 and salary sames

As employee with id 143

select \* from employees where job\_id=(select job\_id from employees where employee\_id=141) and

  salary = ( select salary from employees where employee\_id = 143);

# fetch all employees who have min salary

Select all the min salary and then employees who have min salary

select min(salary) from employee;

select \* from employee salary = (select min(salary) from employees);

#fetch employee who have same salary as the minimum sal if department with id 50

select min(salary) from employees where department\_id=50;

select first\_name, department\_id, salary from employees where salary =

( select min(salary) from employees where department\_id=50) ;

Overview

-Place subqueries on the right side of the comparison condition.

-Use single-row operators with single-row subqueries, and use multiple-row

operators with multiple-row subqueries.

-multiple-row subqueries cannot be used with single row queries using = operator

SELECT employee\_id, last\_name FROM employees WHERE salary =

(SELECT MIN(salary) FROM employees GROUP BY department\_id);

- null value subqueries. Sometimes subquery will return null. The other type is the main subquery may not produce result

SELECT emp.last\_name FROM employees emp

WHERE emp.employee\_id NOT IN (SELECT mgr.manager\_id FROM employees mgr);

USING THE SET OPERATORS

UNION/UNION ALL

INTERSECT

MINUS

UNION OPERATOR

The UNION operator returns results from both

queries after eliminating duplications.

Display the current and previous job details of all employees. Display each employee only once.

SELECT employee\_id, job\_id

FROM employees

UNION

SELECT employee\_id, job\_id

FROM job\_history;

UNION ALL OPERATOR

The UNION ALL operator returns results from both

queries, including all duplications.

#Display the current and previous departments of all employees.

SELECT employee\_id, job\_id, department\_id

FROM employees

UNION ALL

SELECT employee\_id, job\_id, department\_id

FROM job\_history

ORDER BY employee\_id;

INTERSECT OPERATOR

The INTERSECT operator returns rows that are

common to both queries.

#Display the employee IDs and job IDs of those employees who currently have a job title that is the same as their job title when they were initially hired (that is, they changed jobs but have now gone back to doing their original job).

-- intersect in Oracle is EXISTS in my sql

SELECT employee\_id, job\_id

FROM employees

INTERSECT

SELECT employee\_id, job\_id

FROM job\_history;

SELECT  employee\_id, first\_name, job\_id FROM employees

 WHERE EXISTS (SELECT \* FROM job\_history

               WHERE employees.employee\_id = job\_history.employee\_id);

MINUS OPERATOR

The MINUS operator returns rows in the first query

that are not present in the second query.

# Display the employee IDs of those employees who have not changed their jobs even once.

SELECT employee\_id

FROM employees

MINUS

SELECT employee\_id

FROM job\_history;

-- in my sql MINUS operator is not available instead we could use left join

Select \* from employees left join job\_history on employees.employee\_id = job\_history.employee\_id where job\_history.employee\_id is NULL;

OVERVIEW

The expressions in the SELECT lists must match in number and data

type.

– Parentheses can be used to alter the sequence of execution.

– The ORDER BY clause:

• Can appear only at the very end of the statement

• Will accept the column name, aliases from the first SELECT statement, or

the positional notation

INSERT INTO departments

(department\_id, department\_name, location\_id)

VALUES (&department\_id, '&department\_name',&location);

Updating Two Columns with a

Subquery

Update employee 114 job and salary to match that of employee 205

UPDATE employees

SET job\_id = (SELECT job\_id

FROM employees

WHERE employee\_id = 205),

salary = (SELECT salary

FROM employees

WHERE employee\_id = 205)

training@whitebox-learning.com

-----------------------QUERIES DONE IN MY SQL ------------------------------

describe employees;

select \* from employees;

select \* from job\_history;

select \* from employees where employee\_id = 106;

SELECT last\_name, salary

FROM employees

WHERE salary >

(SELECT salary

FROM employees

WHERE last\_name = 'Abel');

SELECT last\_name, employee\_id, job\_id, salary FROM employees

WHERE job\_id = (SELECT job\_id FROM employees WHERE employee\_id = 141)

AND salary > (SELECT salary FROM employees WHERE employee\_id = 143);

SELECT last\_name, job\_id, salary

FROM employees

WHERE salary =

(SELECT MIN(salary)

FROM employees);

SELECT MIN(salary)

FROM employees

WHERE department\_id = 50;

SELECT department\_id, MIN(salary)

FROM employees

GROUP BY department\_id

HAVING min(salary) >

(SELECT MIN(salary)

FROM employees

WHERE department\_id = 50);

SELECT employee\_id, last\_name

FROM employees

WHERE salary =

(SELECT MIN(salary)

FROM employees

GROUP BY department\_id);

SELECT salary FROM employees WHERE job\_id = 'IT\_PROG'; -- 9000, 6000, 4800, 4200

SELECT employee\_id, last\_name, job\_id, salary

FROM employees

WHERE salary IN (SELECT salary FROM employees WHERE job\_id = 'IT\_PROG')

AND job\_id <> 'IT\_PROG';

SELECT employee\_id, last\_name, job\_id, salary

FROM employees

WHERE salary < ALL

(SELECT salary

FROM employees

WHERE job\_id = 'IT\_PROG')

AND job\_id <> 'IT\_PROG';

-- Fetch all the departments based on location = ‘ california’

select \* from locations;

SELECT \* FROM departments WHERE

location\_id =(select location\_id from locations where upper(state\_province) = 'CALIFORNIA');

select \* from employees where

  job\_id=(select job\_id from employees where employee\_id=141) and

  salary = ( select salary from employees where employee\_id = 143);

select \* from employees WHERE salary = (select min(salary) from employees);

select first\_name, department\_id, salary from employees where salary =

( select min(salary) from employees where department\_id=50) ;

SELECT MIN(salary)

FROM employees

GROUP BY department\_id;

SELECT emp.last\_name FROM employees emp

WHERE emp.employee\_id NOT IN (SELECT mgr.manager\_id FROM employees mgr);

SELECT employee\_id, job\_id

FROM employees

UNION

SELECT employee\_id, job\_id

FROM job\_history;

SELECT employee\_id, job\_id, department\_id

FROM employees

UNION

SELECT employee\_id, job\_id, department\_id

FROM job\_history

ORDER BY employee\_id;

SELECT employee\_id, job\_id, department\_id

FROM employees WHERE EMPLOYEE\_ID =

(SELECT employee\_id

FROM job\_history

ORDER BY employee\_id);

(SELECT employee\_id, job\_id

FROM employees)

INTERSECT (SELECT employee\_id, job\_id FROM job\_history);

SELECT employee\_id, first\_name, job\_id FROM employees

  WHERE EXISTS (SELECT \* FROM job\_history

  WHERE employees.employee\_id = job\_history.employee\_id);

select \* from job\_history;

(SELECT employee\_id

FROM employees)

MINUS (SELECT employee\_id FROM job\_history);

Select \* from employees left join job\_history on

employees.employee\_id = job\_history.employee\_id

where job\_history.employee\_id is NULL;