Displays employees who are not IT programmers and whose salary is less than that of any IT programmer. The maximum salary that a programmer earns is $9,000.

< ANY means less than the maximum. >ANY means more than the minimum. =ANY is equivalent to IN.

Using the ALL Operator in Multiple-Row Subqueries

```
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';
```

Displays employees whose salary is less than the salary of all employees with a job ID of IT_PROG and whose job is not IT_PROG.

- ALL means more than the maximum, and <ALL means less than the minimum.

The NOT operator can be used with IN, ANY, and ALL operators.

Null Values in a Subquery

```
SELECT emp.last_name FROM employees emp
WHERE emp.employee_id NOT IN (SELECT mgr.manager_id FROM employees mgr);
```

Notice that the null value as part of the results set of a subquery is not a problem if you use the IN operator. The IN operator is equivalent to =ANY. For example, to display the employees who have subordinates, use the following SQL statement:

```
SELECT emp.last_name
FROM employees emp
WHERE emp.employee_id IN (SELECT mgr.manager_id FROM employees mgr);
```

Display all employees who do not have any subordinates:

```
SELECT last_name FROM employees
WHERE employee_id NOT IN (SELECT manager_id FROM employees WHERE manager_id IS
NOT NULL);
```

Find the Solution for the following:

1. The HR department needs a query that prompts the user for an employee last name. The query then displays the last name and hire date of any employee in the same department as the employee whose name they supply (excluding that employee). For example, if the user enters Zlotkey, find all employees who work with Zlotkey (excluding Zlotkey).

```
select e.lastname, e.hiredate from employees as e join select distinct
department_id from employees where last_name = last_name and
first_name = first_name) as targetdept ON e.department_ID = targetdept.
depart where e.lastname <> last_name OR e.firstname <> first_name;
```

2. Create a report that displays the employee number, last name, and salary of all employees who earn more than the average salary. Sort the results in order of ascending salary.

```
select employee_number, last_name, salary from
employees where salary > ( select avg (salary) from
employees ) order by salary ASC;
```

3. Write a query that displays the employee number and last name of all employees who work in a department with any employee whose last name contains a *u*.

```
select e.employee_number, e.last_name from employees
where exists ( select 1 from employees e2 where
e2.department_id = e.department_id AND lower(e2.last_name)
LIKE '%u%' );
```

4. The HR department needs a report that displays the last name, department number, and job ID of all employees whose department location ID is 1700.

```
select last_name, department_id as department_number,
job_id from employees where location_id = 1700;
```

5. Create a report for HR that displays the last name and salary of every employee who reports to King.

```
select e.lastname, e.salary from employes e where
e.manager_id = < king_employee_id >;
```

6. Create a report for HR that displays the department number, last name, and job ID for every employee in the Executive department.

```
select department_id as department_number, last_name,
job_id from employes where department_id =
exec_dept_id;
```

7. Modify the query 3 to display the employee number, last name, and salary of all employees who earn more than the average salary and who work in a department with any employee whose last name contains a *u*.

```
select e.employee_number, e.last_name, e.salary from
employee where e.salary > ( select avg( salary ) from
employees ) and exists ( select 1 from employes
e2 where e2.department_id = e.department_id
and lower ( e2.last_name) like '%u%' );
```