NetID: jc12300
Name: Jie Cheng
Course: CSGY-6083-Principles of Database Systems
Section: B
Date of submission: Dec 5, 2023

NYU TANDON SCHOOL OF ENGINEERING

CS-GY 6083 - B, FALL 2023
<u>Principles of Database Systems</u>
Assignment: 4 [100 points]

Please submit your assignment on NYU Brightspace course site with a single PDF document attachment. Please mention Student ID, Name, Course, Section Number, and date of submission on first page of your submission. Each table in your submission of SQLs and their results should have your initial as prefix, e.g., AP_EMPLOYEE etc. You can use either Oracle or MySQL for this assignment.

Q1) To write a database procedure (Oracle or MySQL) [60 points]

The HR department intend to give salary increment to employees of specific department when requested by their department director. Different directors have different criteria about salary increment. For an example, some directors ask for base increment as 5% of average salary of their department, and some may ask for base increment as 7% or 10% of average salary. So, the base increment percent of avg. salary is determined by the department director. However, following criteria remains same for all departments.

The new salary is calculated by the formula,

New Salary = S + N% of A+ S*Y%

S= original salary
N%= base increment percent of department's avg. salary (e.g 5, 7, 10 etc.)
A= average salary of the department
Y=Square root of number of years employee's working as of Dec. 31st, 2022.

Write a database procedure that takes two input variables department number and base N percentage of avg salary. Apply salary increment criteria as detailed above. Your procedure name should have your initial as prefix, e.g. AP_RAISE_SAL.

Use the table and its data attached to the assignment.

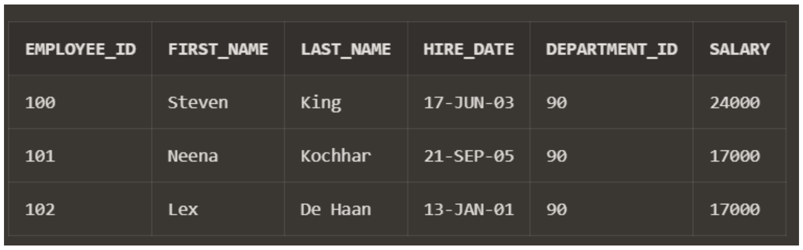
Submit:

a) Procedure code (Oracle or MySQL)

```
CREATE OR REPLACE PROCEDURE JC_RAISE_SAL (
   p_department_id JC_EMPLOYEE.DEPARTMENT_ID%TYPE,
   p_base_increment_percent NUMBER
) AS
   v_avg_salary NUMBER;
   v_increment_amount NUMBER;
   v_years_worked NUMBER;
   v_new_salary NUMBER;
BEGIN
   -- Calculate the average salary for the specified department
   SELECT AVG(SALARY)
   INTO v_avg_salary
   FROM JC_EMPLOYEE
   WHERE DEPARTMENT_ID = p_department_id;

   -- Loop through employees in the specified department
   FOR emp_rec IN (SELECT EMPLOYEE_ID, SALARY, HIRE_DATE FROM JC_EMPLOYEE WHERE
DEPARTMENT_ID = p_department_id)
   LOOP
      -- Calculate the number of years worked as of Dec. 31st, 2022
      v_years_worked := TRUNC(MONTHS_BETWEEN(TO_DATE('31-DEC-2022', 'DD-MON-YYYY'),
emp_rec.HIRE_DATE) / 12);

      -- Calculate the increment amount based on the given criteria
      v_increment_amount := (p_base_increment_percent / 100) * v_avg_salary +
emp_rec.SALARY * SQRT(v_years_worked) / 100;

      -- Calculate the new salary
      v_new_salary := emp_rec.SALARY + v_increment_amount;

      -- Update the employee's salary in the database
      UPDATE JC_EMPLOYEE
      SET SALARY = v_new_salary
      WHERE EMPLOYEE_ID = emp_rec.EMPLOYEE_ID;
   END LOOP;

   -- Commit the changes
   COMMIT;

   DBMS_OUTPUT.PUT_LINE('Salary increment applied successfully.');
EXCEPTION
   WHEN NO_DATA_FOUND THEN
      DBMS_OUTPUT.PUT_LINE('No employees found for the specified department.');
   WHEN OTHERS THEN
      DBMS_OUTPUT.PUT_LINE('An error occurred: ' || SQLERRM);
END JC_RAISE_SAL;
/
```

b) If you are using Oracle, provide result of following,

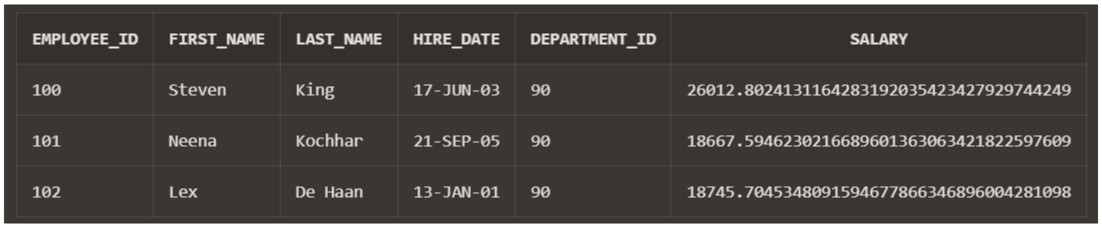
```
SELECT employee_id, first_name,last_name,hire_date,department_id, salary
FROM ap_employee
WHERE department_id=90;
```



EMPLOYEE_ID	FIRST_NAME	LAST_NAME	HIRE_DATE	DEPARTMENT_ID	SALARY
100	Steven	King	17-JUN-03	90	24000
101	Neena	Kochhar	21-SEP-05	90	17000
102	Lex	De Haan	13-JAN-01	90	17000

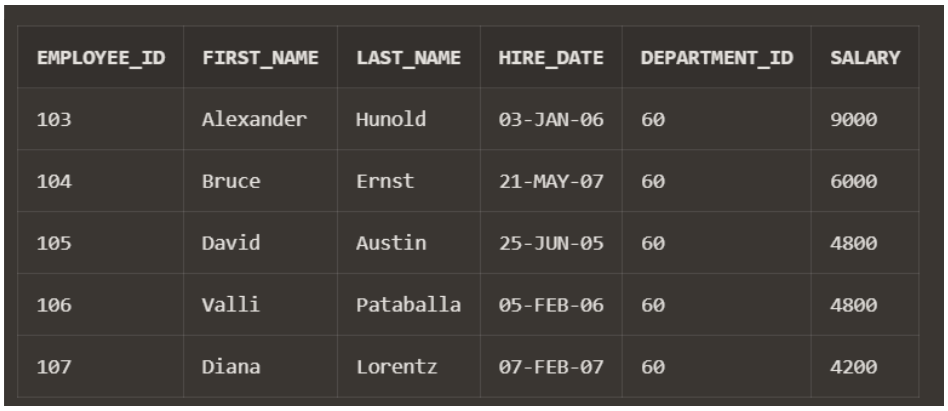
```
execute ap_raise_sal (90, 5); -- 90 is the department_id and 5 is base increment of avg. salary
```

```
SELECT employee_id, first_name,last_name,hire_date,department_id, salary
FROM ap_employee
WHERE department_id=90;
```



EMPLOYEE_ID	FIRST_NAME	LAST_NAME	HIRE_DATE	DEPARTMENT_ID	SALARY
100	Steven	King	17-JUN-03	90	26012.8024131164283192035423427929744249
101	Neena	Kochhar	21-SEP-05	90	18667.5946230216689601363063421822597609
102	Lex	De Haan	13-JAN-01	90	18745.7045348091594677866346896004281098

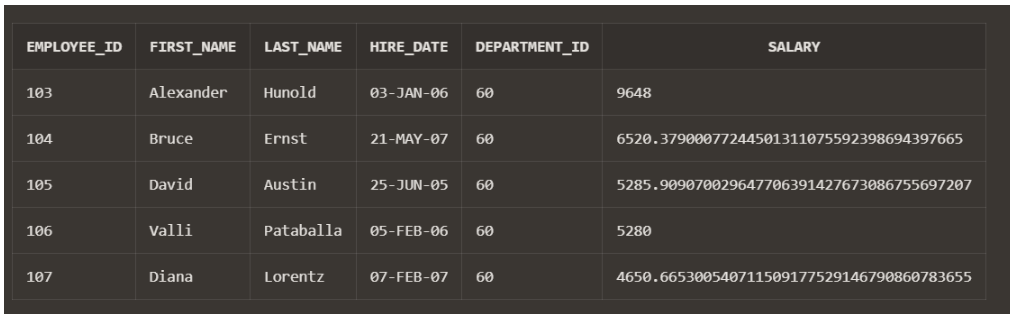
```
SELECT employee_id, first_name,last_name,hire_date,department_id, salary
FROM ap_employee
WHERE department_id=60;
```



EMPLOYEE_ID	FIRST_NAME	LAST_NAME	HIRE_DATE	DEPARTMENT_ID	SALARY
103	Alexander	Hunold	03-JAN-06	60	9000
104	Bruce	Ernst	21-MAY-07	60	6000
105	David	Austin	25-JUN-05	60	4800
106	Valli	Pataballa	05-FEB-06	60	4800
107	Diana	Lorentz	07-FEB-07	60	4200

```
execute ap_raise_sal (60, 5); -- 60 is the department_id and 5 is base increment of avg. salary
```

```
SELECT employee_id, first_name,last_name,hire_date,department_id, salary
FROM ap_employee
WHERE department_id=60;
```



EMPLOYEE_ID	FIRST_NAME	LAST_NAME	HIRE_DATE	DEPARTMENT_ID	SALARY
103	Alexander	Hunold	03-JAN-06	60	9648
104	Bruce	Ernst	21-MAY-07	60	6520.37900077244501311075592398694397665
105	David	Austin	25-JUN-05	60	5285.909070029647706391427673086755697207
106	Valli	Pataballa	05-FEB-06	60	5280
107	Diana	Lorentz	07-FEB-07	60	4650.665300540711509177529146790860783655

If you are using MySQL, provide result of following,

SELECT employee_id, first_name,last_name,hire_date,department_id, salary
FROM ap_employee
WHERE department_id=90;

call ap_raise_sal (90, 5); -- 90 is the department_id and 5 is base increment of avg. salary

SELECT employee_id, first_name,last_name,hire_date,department_id, salary
FROM ap_employee
WHERE department_id=90;

SELECT employee_id, first_name,last_name,hire_date,department_id, salary
FROM ap_employee
WHERE department_id=60;

call ap_raise_sal (60, 5); -- 60 is the department_id and 5 is base increment of avg. salary

SELECT employee_id, first_name,last_name,hire_date,department_id, salary
FROM ap_employee
WHERE department_id=60;

Q2) Indexes [40 points]

Consider following queries to the same employee table that used in Q1.

```
  select *  from ap_employee
  where substr(last_name,1,1)='A'and JOB_ID='SA_REP'
  order by last_name;


select upper(first_name), upper(last_name), department_id, salary
from ap_employee a where a.salary>(select avg(salary) from ap_employee b
                                              where b.department_id=department_id);
```

For each of the above query do following

a) **Suggest which column(s) are suitable for indexes and what type of index should be created.**
b) **Create index(es) as suggested in step a**
c) **Create the query execution plan.**

Submit

i) **Suggested column(s) for index(es) and type of the index(es)**

For query 1
An index on the last_name column would be suitable for optimizing the WHERE clause and the ORDER BY clause. The type of the index is function based index.

For query 2
An index on the department_id column would be suitable for optimizing the subquery. The type of the index is bitmap index.

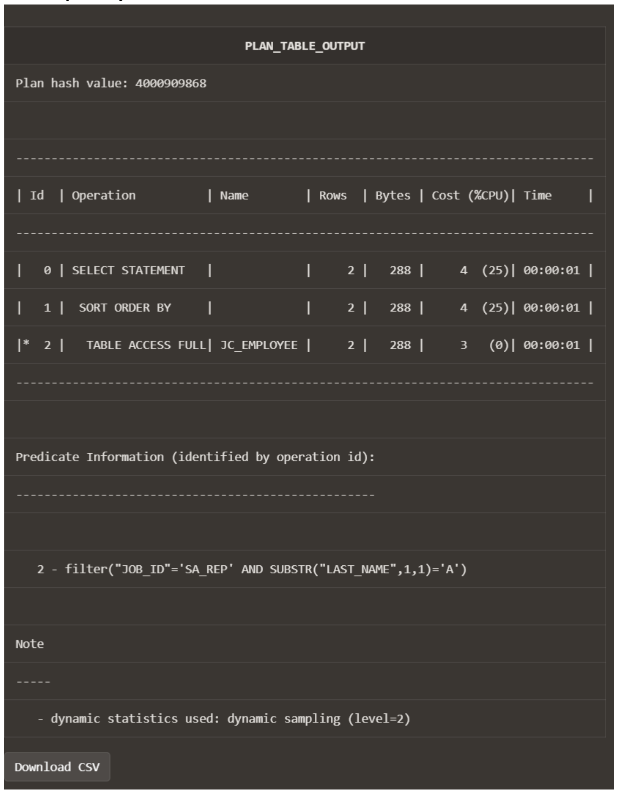
ii) **DDL code of the index(es) created.**

For query 1
CREATE INDEX idx_last_name ON ap_employee(last_name);

For query 2
CREATE INDEX idx_department_id ON ap_employee(department_id);

iii) **Screenshot of execution plan**

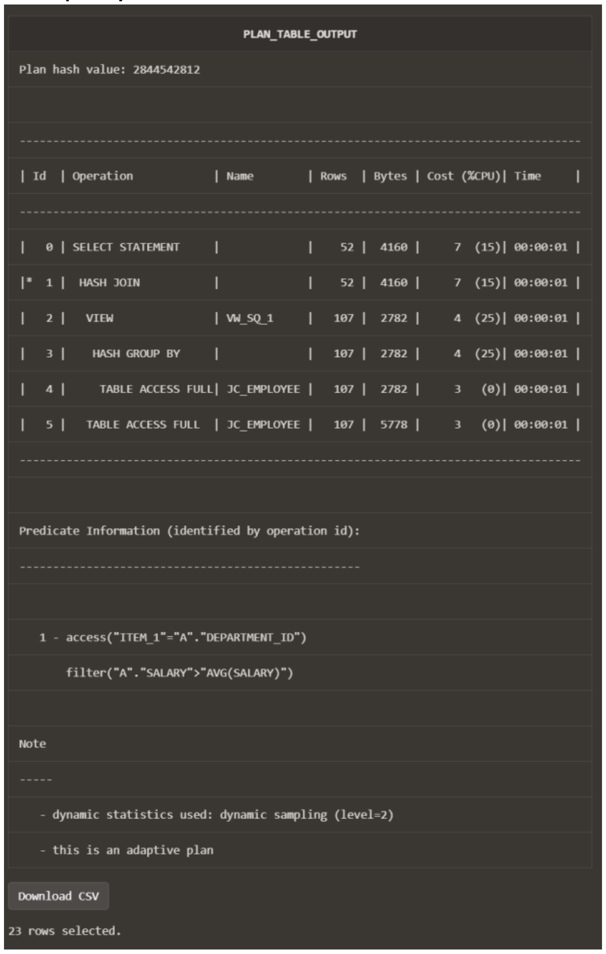
For query 1



```
PLAN_TABLE_OUTPUT
Plan hash value: 4000909868

-------------------------------------------------------------------------------
| Id  | Operation          | Name        | Rows  | Bytes | Cost (%CPU)| Time     |
-------------------------------------------------------------------------------
|   0 | SELECT STATEMENT   |             |     2 |   288 |     4  (25)| 00:00:01 |
|   1 |  SORT ORDER BY     |             |     2 |   288 |     4  (25)| 00:00:01 |
|*  2 |   TABLE ACCESS FULL| JC_EMPLOYEE |     2 |   288 |     3   (0)| 00:00:01 |
-------------------------------------------------------------------------------

Predicate Information (identified by operation id):
---------------------------------------------------

   2 - filter("JOB_ID"='SA_REP' AND SUBSTR("LAST_NAME",1,1)='A')

Note
-----
   - dynamic statistics used: dynamic sampling (level=2)
```

Download CSV

For query 2



```
PLAN_TABLE_OUTPUT
Plan hash value: 2844542812

--------------------------------------------------------------------------------
| Id  | Operation            | Name        | Rows  | Bytes | Cost (%CPU)| Time     |
--------------------------------------------------------------------------------
|   0 | SELECT STATEMENT     |             |    52 |  4160 |     7  (15)| 00:00:01 |
|*  1 |  HASH JOIN           |             |    52 |  4160 |     7  (15)| 00:00:01 |
|   2 |   VIEW               | VW_SQ_1     |   107 |  2782 |     4  (25)| 00:00:01 |
|   3 |    HASH GROUP BY     |             |   107 |  2782 |     4  (25)| 00:00:01 |
|   4 |     TABLE ACCESS FULL| JC_EMPLOYEE |   107 |  2782 |     3   (0)| 00:00:01 |
|   5 |   TABLE ACCESS FULL  | JC_EMPLOYEE |   107 |  5778 |     3   (0)| 00:00:01 |
--------------------------------------------------------------------------------

Predicate Information (identified by operation id):
---------------------------------------------------

   1 - access("ITEM_1"="A"."DEPARTMENT_ID")
       filter("A"."SALARY">"AVG(SALARY)")

Note
-----
   - dynamic statistics used: dynamic sampling (level=2)
   - this is an adaptive plan
```

Download CSV

```
23 rows selected.
```

iv) Explanation about which index(es) are used and which are not, and reason for it

For query 1
last name is used because it is used in ORDER BY clause.

For query 2
department id is used because it is used frequently in WHERE clause. Salary is not used because it might be frequently updated.