CS-GY 6083 - B, FALL 2023

Principles of Database Systems

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.

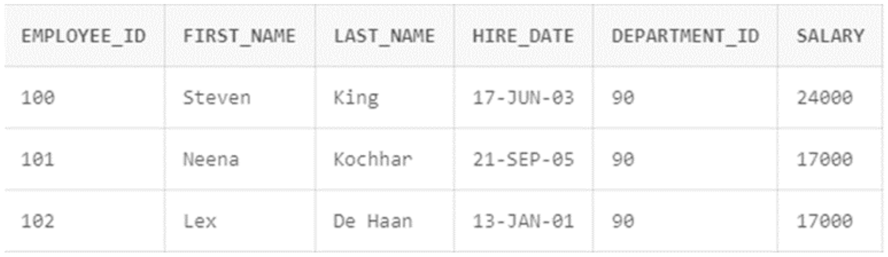
Submit:

a) Procedure code (Oracle or MySQL)

```
CREATE OR REPLACE PROCEDURE AP_raise_sal (
  p_deptno IN NUMBER,
   p_base_increment_percentage IN NUMBER
) AS
   v_avg_salary NUMBER;
   ref_date DATE := TO_DATE('2022-12-31', 'YYYY-MM-DD');
BEGIN

   SELECT AVG(salary) INTO v_avg_salary
   FROM AP_EMPLOYEE
   WHERE DEPARTMENT_ID = p_deptno;

   UPDATE AP_EMPLOYEE b
   SET salary = ROUND(salary + (v_avg_salary * p_base_increment_percentage / 100)
               + (salary * SQRT(EXTRACT(YEAR FROM ref_date) - EXTRACT(YEAR FROM
hire_date)) / 100))
   WHERE department_id = p_deptno;

   COMMIT;
EXCEPTION
   WHEN OTHERS THEN
      ROLLBACK;
      DBMS_OUTPUT.PUT_LINE('An error occurred: ' || SQLERRM);
END;
/
```

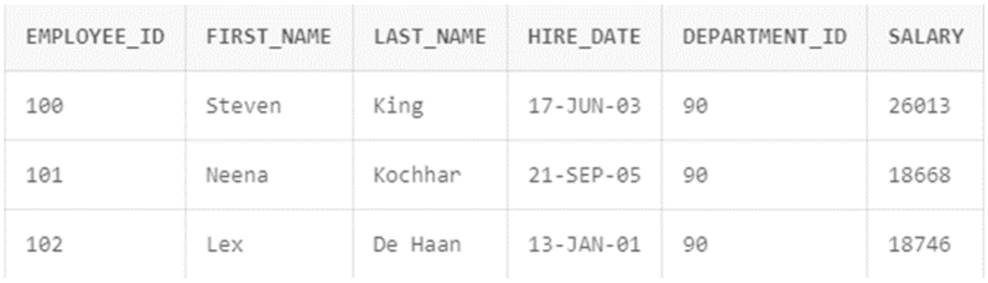
b) If you are using Oracle, provide result of following,

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



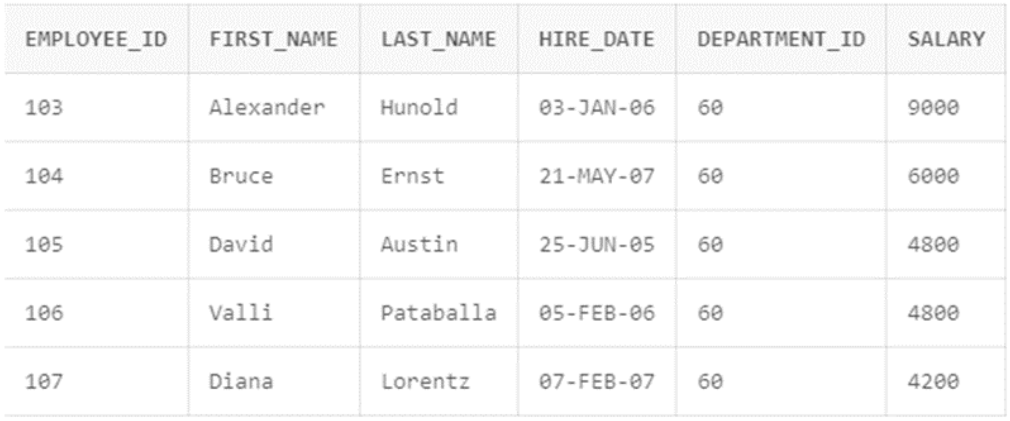
EMPLOYEE_ID	FIRST_NAME	LAST_NAME	HIRE_DATE	DEPARTMENT_ID	SALARY
100	Steven	King	17-JUN-03	90	24000
101	Neena	Kochhar	21-SEP-05	90	17000
102	Lex	De Haan	13-JAN-01	90	17000

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

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



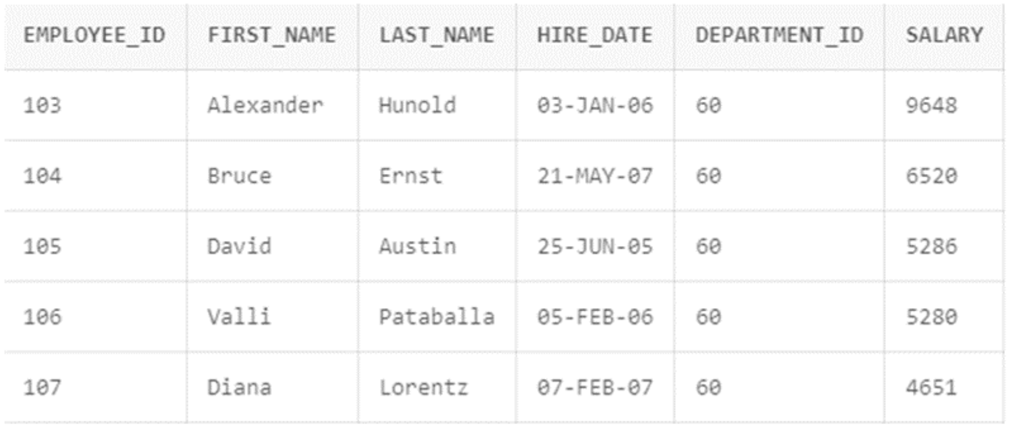
EMPLOYEE_ID	FIRST_NAME	LAST_NAME	HIRE_DATE	DEPARTMENT_ID	SALARY
100	Steven	King	17-JUN-03	90	26013
101	Neena	Kochhar	21-SEP-05	90	18668
102	Lex	De Haan	13-JAN-01	90	18746

```
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

```
SELECT employee_id, first_name,last_name,hire_date,department_id, salary
FROM ap_employee
WHERE department_id=60;
execute ap_raise_sal (60, 5);  -- 60 is the department_id  and 5 is base increment of avg. salary
```



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
105	David	Austin	25-JUN-05	60	5286
106	Valli	Pataballa	05-FEB-06	60	5280
107	Diana	Lorentz	07-FEB-07	60	4651

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)**
ii) **DDL code of the index(es) created.**
iii) **Screenshot of execution plan**
iv) **Explanation about which index(es) are used and which are not, and reason for it**

a)

i) Indexes should be created on the following columns

last_name, job_id

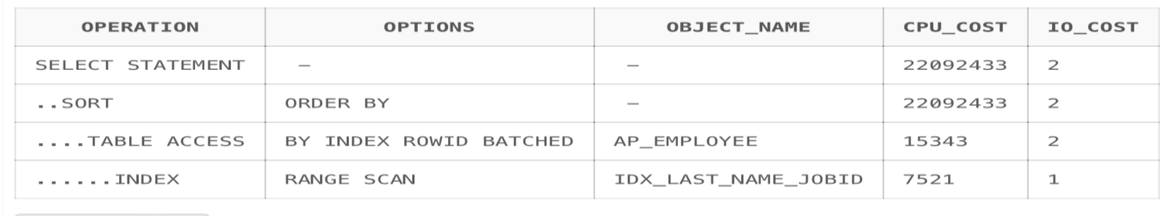
This should be a composite index containing both columns and a functional index on last_name.

Student may have suggested an additional index on last_name, since it is in order by clause.

ii)

CREATE INDEX idx_last_name_jobid ON ap_employee (SUBSTR(last_name, 1, 1), job_id);

iii)



OPERATION	OPTIONS	OBJECT_NAME	CPU_COST	IO_COST
SELECT STATEMENT	–	–	22092433	2
..SORT	ORDER BY	–	22092433	2
....TABLE ACCESS	BY INDEX ROWID BATCHED	AP_EMPLOYEE	15343	2
......INDEX	RANGE SCAN	IDX_LAST_NAME_JOBID	7521	1

iv)

We will use a composite and functional index, because the where clause filters based on 2 columns, both of which should be included in the index for optimal performance. Also, it also contains a function (substr on last_name), hence it should also be a functional index.

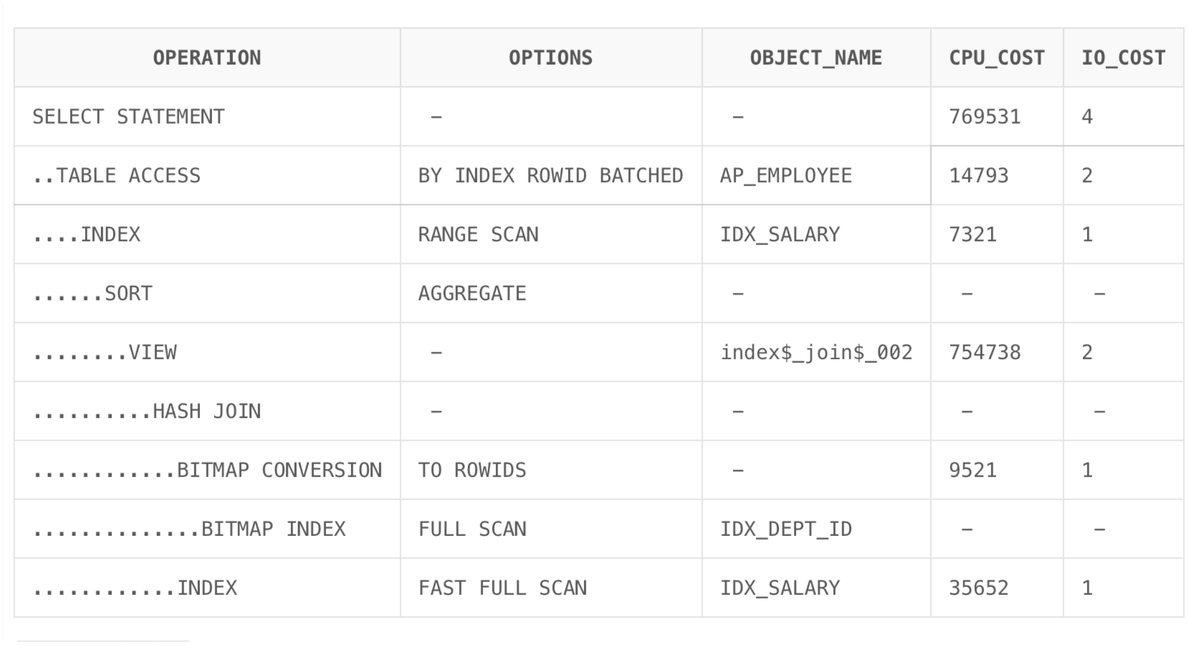
We cannot use a bitmap index as it requires columns with low cardinality which is not the case for last_name.

b)

i)
We will create index on the following columns
salary - B- tree index
department_id - bitmap index

ii)
CREATE INDEX idx_salary ON ap_employee (salary);
CREATE BITMAP INDEX idx_dept_id on ap_employee(department_id);

iii)



OPERATION	OPTIONS	OBJECT_NAME	CPU_COST	IO_COST
SELECT STATEMENT	-	-	769531	4
..TABLE ACCESS	BY INDEX ROWID BATCHED	AP_EMPLOYEE	14793	2
....INDEX	RANGE SCAN	IDX_SALARY	7321	1
......SORT	AGGREGATE	-	-	-
........VIEW	-	index$_join$_002	754738	2
..........HASH JOIN	-	-	-	-
............BITMAP CONVERSION	TO ROWIDS	-	9521	1
..............BITMAP INDEX	FULL SCAN	IDX_DEPT_ID	-	-
............INDEX	FAST FULL SCAN	IDX_SALARY	35652	1

iv)
We will create a B- tree index on salary to improve query performance. We cannot use a bitmap index on salary column as it does not have a low cardinality.
However, we can create a bitmap index on department_id column which has low cardinality (ie, the number of distinct values of department_id would be much less than the total number of records in the employee table).