109-1 Midterm solution

- **Database system**: database + DBMS
- **Three-Schema Architecture**
  - **Internal level**
    - Describes **physical storage structure** of the database.
  - **Conceptual level**
    - Describes **structure of the whole database** for a community of users.
  - **External** or **view level**
    - Describes part of the database that a **particular user group** is interested in.

Transaction

- Logical unit of database processing that includes **one or more access operations** (read -retrieval, write - insert or update, delete).
- Desirable Properties of Transactions **ACID**
  - Atomicity: A transaction is an **atomic** unit of processing; it is either performed in its **entirety or not** performed at all.
  - Consistency preservation: A correct execution of the transaction **must** take the database from one consistent state to another.
  - Isolation: A transaction should not make its updates visible to other transactions until it is **committed**; this property, when enforced strictly, solves the temporary update problem and makes cascading rollbacks of transactions.
  - Durability or permanency: Once a transaction changes the database and the changes are **committed**, these changes must never be lost because of subsequent failure.

Retrieve the name, email and salary of all employees who work for the **'IT'** department.

```
SELECT last_name, email, salary
FROM EMPLOYEES
WHERE department_id =
     (SELECT department_id
      FROM    DEPARTMNETS
      WHERE department_name = 'IT';
```

```
SELECT last_name, job_id, EMAIL, salary
FROM   employees E JOIN DEPARTMENTS D
ON   E.DEPARTMENT_ID = D.DEPARTMENT_ID
WHERE D.DEPARTMENT_NAME = 'Marketing';
```

$$TEMP \leftarrow DEPARTMENTS \bowtie_{DEPARTMENTS.DEPARTMENT\_ID = EMPLOYEES.DEPARTMENT\_ID} EMPLOYEES$$

$$RESULT \leftarrow \sigma_{DEPARTMENT\_NAME = \text{‘IT'}} (TEMP)$$

Display the average, highest, lowest and sum of the monthly salaries and the number of employees for each department.

```
SELECT   department_id, AVG(salary), MAX(salary),
MIN(salary),SUM(salary), COUNT(*)
FROM     employees
GROUP BY department_id ;
```

department_id $\mathcal{F}$ AVG salary, MAX salary, MIN salary, SUM salary, COUNT employees_id, (EMPLOYEES)

Following question (2) display each **department name**.

```
SELECT   department_name, AVG(salary), MAX(salary),
MIN(salary),SUM(salary), COUNT(*)

FROM     employees E JOIN DEPARTMENTS D ON
E.DEPARTMENT_ID = D.DEPARTMENT_ID

GROUP BY department name ;
```

TEMP ← DEPARTMENTS $\bowtie_{\text{DEPARTMENTS.DEPARTMENT\_ID = EMPLOYEES.DEPARTMENT\_ID}}$ EMPLOYEES

RESULT ← department_name $\mathcal{F}$ AVG salary, MAX salary, MIN salary, SUM salary, COUNT employees_id (EMPLOYEES)

Find the names of employees and their respective managers.

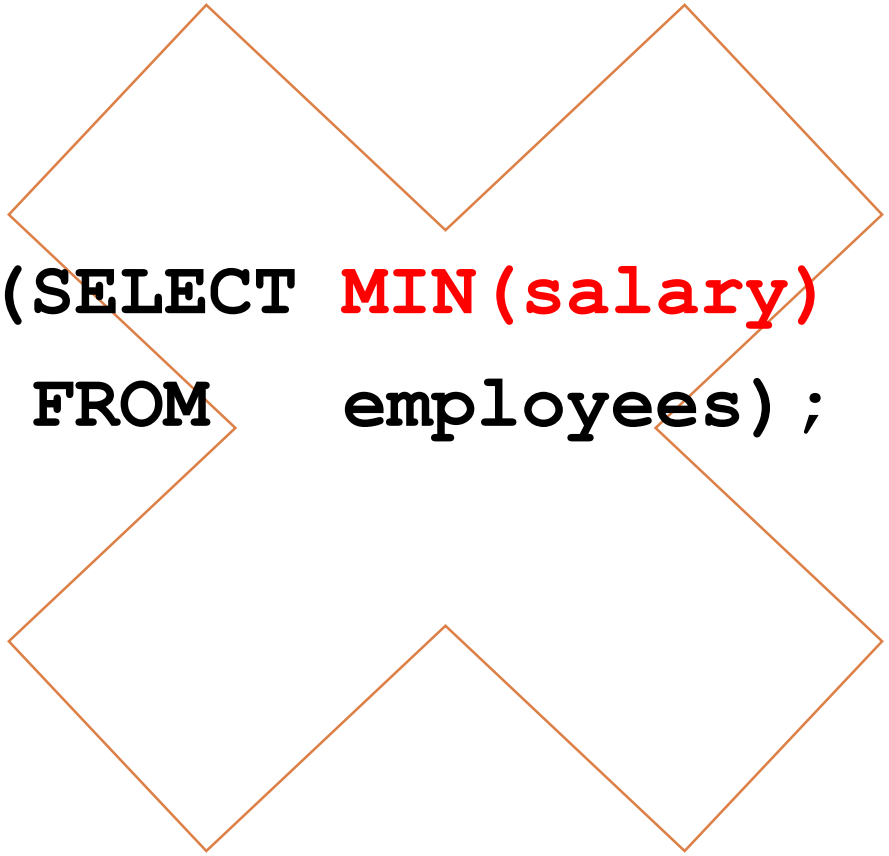
```
SELECT E.last_name, M.last_name
FROM   employees E JOIN employees M
ON     E.manager_id = M.employee_ID;
```

$$TEMP \leftarrow \rho_{E}(EMPLOYEES) \bowtie_{E.\ manager\_id = M.employee\_ID} \rho_{M}(EMPLOYEES)$$

$$RESULT \leftarrow \pi_{E.last\_name,\ M.last\_name}(TEMP)$$

Retrieve the name of employees who have minimum salary in their department.

```
SELECT last_name, job_id, salary
FROM   employees
WHERE  salary =
                (SELECT MIN(salary)
                 FROM   employees);
```



Retrieve the name of employees who have maximum salary in their department.

```
SELECT last_name, job_id, salary
FROM   employees
WHERE  (department_id,salary) IN
           (SELECT department_id, MAX(salary)
            FROM   employees
            GROUP BY department_id);
```

Retrieve the name of employees who earn more than the average salary department in their department.

```
SELECT last_name
FROM EMPLOYEES OUT
WHERE OUT.salary >
     (SELECT AVG(salary)
      FROM    EMPLOYEE INNER
      WHERE   INNER.department_id =
              OUT.department_id);
```

Show the last name, job, salary, and department name of those employees who earn commission. Sort the data by salary in descending order.

```
SELECT last_name, job_id, salary,
department_name
FROM   employees E JOIN DEPARTMENTS D
ON    E.DEPARTMENT_ID = D.DEPARTMENT_ID
WHERE E.COMMISSION_PCT IS NOT NULL
ORDER BY salary DESC;
```

Show the department number, department name, and number of employees working in each department that: **Includes fewer than 3 employees.**

```
SELECT D.department_id, department_name, COUNT(*) NOofDept
FROM   employees E join departments D
on E.department_id = D.department_id
HAVING 2 >= COUNT(*)
GROUP BY D.department_id, department_name;
```

Has the highest number of employees.

```
SELECT D.department_id DEPT_ID, department_name, COUNT(*) NOofDept
FROM    employees E join departments D
on E.department_id = D.department_id
GROUP BY D.department_id, department_name
HAVING  COUNT(*)=
        (SELECT MAX(COUNT(*))
        FROM employees
        group by department_id);
```

Write a query to display the top 3 earners in the EMPLOYEES table. Display their last names, salaries and department name.

```
SELECT rownum RANK, last_name, salary, department_name
FROM
        (SELECT department_name, last_name, salary
        from employees E join departments D
        on E.department_id = D.department_id
        ORDER BY salary desc)
where rownum < 3;
```

Display the details of the last name, department name, and salary of those employees who live in cities whose name begins with T.

```
SELECT employee_id, last_name, department_id
FROM   employees e
JOIN   departments d
ON     d.department_id = e.department_id
JOIN   locations l
ON     d.location_id = l.location_id
WHERE  l.CITY like  'T%' ;
```

Find all departments that do not have any employees.

```
SELECT      department_name
FROM        DEPARTMENTS D
WHERE NOT EXIST
            (SELECT 'X'
             FROM       EMPLOYEES E
             WHERE      E.department_id = D.department_id);
```

```
SELECT E.last_name
FROM   EMPLOYEES E
WHERE NOT EXIST
          (SELECT 'X'
              FROM EMPLOYEES M
              E.manager_id = M.employee_id );
```

```
SELECT   E.last_name
FROM     EMPLOYEES E
WHERE    E.manager_id IS NULL;
```

Write a query to display the last names of employees who have one or more coworkers in their departments with later hire date but higher salaries.

```
SELECT last_name
FROM    employees outer
WHERE EXISTS
      (SELECT 'X'
      FROM     employees inner
      WHERE inner.department_id = outer.department_id
      AND  inner.hire_date > outer.hire_date
      AND  inner.salary > outer.salary);
```

Increase 10 % salary of all employees who belong to 'Finance' department.

```
UPDATE employees
SET    salary = (1+0.1) * salary
WHERE department_id =
           (SELECT department_id
           FROM   departments d
           WHERE  department_name  = 'Finance');
```

Insert < 70, 'Public Relations', 100, 1700> into departments.

```
INSERT INTO departments(department_id,
        department_name, manager_id, location_id)
VALUES (70, 'Public Relations', 100, 1700);
```

Delete the employee who belong to ‘IT’ department.

```
DELETE employees
WHERE  department_id =
        (SELECT department_id
        FROM   departments d
        WHERE  department_name  = ‘Finance');
```

Update employee 114's job and department to match that of employee 205.

```
UPDATE employees
SET    department_id=
                (SELECT department_id
                FROM employee
                WHERE        employee_id = '205'),
       job_id=
                (SELECT job_id
                FROM employee
                WHERE        employee_id = '205'),
WHERE employee_id= 114;
```