Division (DID, dname, managerID)

Employee (empID, name, salary, *DID*)

Project (PID, pname, budget, *DID*)

Workon *(PID, EmpID*, hours)

Formulate the following queries

1. **List the name and salary of employees who don't work for division 5.**

```
select name,
salary from employee
where did not in (5);
```

NAME	SALARY
kevin	32000
joan	42000
brian	37000
harry	92000
peter	45000
peter	68000
smith	39000
chen	71000
smith	46000
joan	48000
kim	49000
austin	46000
Justin	62000
Nacy	52000
Kristie	52000
John	52000
Alex	69000
Phil	72000
Steve	74000
Jenna	69000
Alan	62000
Julia	69000
Sandra	72000
Joe	74000
grace	62000

25 rows returned in 0.00 seconds

2. **List the name of project from division 1 whose budget is between 6000-7000**

```
select pname from Project
Where budget between 6000 and 7000
And
DID=1
```

PNAME
system development

1 row returned in 0.00
seconds

3. **List the total number of employee whose name has 'a' as the second letter from right (hint, using keyword LIKE and wildcard character)**

```
select count(empid) as NumberOfEmployees from employee
where name like '%a_';
```

NUMBEROFEMPLOYEES
5

1 row returned in 0.00
seconds

4. **List the total number of employee whose initial of name is NOT 's' for each division, including division ID**

```
select DID, count(empid) as NumberOfEmployees from employee
where name not like 's%'
Group by DID
```

DID	NUMBEROFEMPLOYEES
1	8
2	6
5	4
4	6
3	3

5 rows returned in 0.00
seconds

NOTES:- Question mentioned the small 's', while in database there are names starts with S. Some of are stored with Capital 'S' and some of are stored with small 's'. As question mentioned small 's'. I have retrieved the data about the names start with small 's'. I can retrieve the data with small and big 'S' too.

5. **List the total project budget for each division, including division ID.**

```
select did, sum(budget) as Totalprojectbudget from project
group by did
order by did
```

DID	TOTALPROJECTBUDGET
1	12000
2	14000
3	5000
4	6000

4 rows returned in 0.00 seconds

6. **List the ID of the division that has two or more projects with budget over $6000.**

```
select did, count(pid) as Totalprojects from project where budget >= 6000
having count(pid)>=2
group by did
```

DID	TOTALPROJECTS
2	2

1 row returned in 0.00 seconds

7. **List the ID of division that sponsors project "Web development", List the project budget too. (hint: use function Upper () or Lower() to convert case of letters to facilitate the comparison.**

```
Select did, pname, budget from project
where upper(pname) = 'WEB DEVELOPMENT'
```

DID	PNAME	BUDGET
3	Web development	5000

1 row returned in 0.00 seconds

8. **List the total number of employees whose salary is above $40000 for each division, list division ID.**

```
Select DID, count(empID) from Employee
where salary > 40000
Group by DID
```

DID	NUMBEROFEMPLOYEES
1	9
2	5
5	5
4	6
3	2

5 rows returned in 0.00 seconds

9. **List the total number of project and total budget for each division, show division ID**

```
Select DID, count(PID) as NumberOfProjects, sum(budget) as TotalBudget from
Project
Group by DID
Order by DID
```

DID	NUMBEROFPROJECTS	TOTALBUDGET
1	2	12000
2	2	14000
3	1	5000
4	1	6000

4 rows returned in 0.01 seconds

10. List the ID of employee that worked on more than three projects.

```
Select EMPID, count(PID) from Workon
having count(PID) > 3
Group by EMPID
order by EMPID
```

EMPID	COUNT(PID)
3	4
6	4
8	4
9	6
26	4

5 rows returned in 0.01 seconds

11. List the ID of each division with its highest salary.

```
Select DID, max(salary) as HighestSalary from Employee
group by did
order by did
```

DID	HIGHESTSALARY
1	71000
2	72000
3	68000
4	92000
5	82000

5 rows returned in 0.00 seconds

12. List the total number of projects each employee works on, including employee's ID and total hours an employee spent on project.

```
Select EMPID, count(PID) as TotalNoProjects, sum(hours) as Totalhours from
Workon
Group by EMPID
Order by EMPID
```

EMPID	TOTALNOPROJECTS	TOTALHOURS
1	1	30
3	4	300
4	3	105
5	1	30
6	4	165
7	3	90
8	4	120
9	6	180
12	1	30
13	1	30
14	1	20
15	1	40
16	2	60
17	1	30
18	1	30
19	2	60
21	2	60
22	2	70
23	1	30
24	1	20
25	2	70
26	4	110
27	2	70
28	2	60
29	1	30
30	3	100

26 rows returned in 0.00 seconds

13. List the total number of employees who work on project 1.

```
Select PID, count(EMPID) as NumberOfEMployees from Workon
where PID = 1
group by PID
```

PID	NUMBEROFEMPLOYEES
1	8

1 rows returned in 0.00
seconds

14. List names that are shared by more than one employee and list the number of employees who share that name.

```
Select Name, count(empid) as NumberOfEmployees from Employee
having count (EMPID) > 1
Group by Name
```

NAME	NUMBEROFEMPLOYEES
peter	2
smith	2
joan	2
kim	2

4 rows returned in 0.00
seconds

15. List the ID of project that accumulated more than 250 man-hours.

```
Select PID, sum(HOURS) as TotalManHours From Workon
Having sum(Hours) > 250
Group by PID
```

PID	TOTALMANHOURS
2	415
3	390
4	385
5	260

4 rows returned in 0.02
seconds

16. Bonus question (1 point) List the total number of employee and total salary for each division, including division name (hint: use JOIN operation, read the text for join operation)

```
SELECT count(EmpID) as TotalNoOfEmployees, Sum(Salary) as TotalSalary,
Division.DName
FROM Employee
INNER JOIN Division
ON employee.Did=division.did
group by dname
```

TOTALNOOFEMPLOYEES	TOTALSALARY	DNAME
7	482000	Research and development
5	301000	accounting
9	495000	engineering
6	322000	marketing
3	157000	human resource

5 rows returned in 0.00 seconds

(Query on Multiple Tables) (15 Queries)

1. **List the total number of projects that employees who work on it make less than $50k. (*Hint: You need to use key word DISTINCT in this query. Can you tell me why?).***

   SELECT count(distinct PID) as TotalNoOfProjects
   FROM employee e, project p
   WHERE e.salary < 50000;

   I used distinct here to remove duplicates from the result set of select statement. If I don't use distinct here then I find 16 counts in result which is duplicity because we don't even have more than 6 projects in table.

TOTALNOOFPROJECTS
6

1 row returned in 0.01 seconds

2. **List the total number of project and total budget for each division, show division name. (*Note Some division may not have project, you still need to show this division in result.)***

   SELECT dname, count(PID) as TotalNoOfProjects, Sum(Budget) as TotalBudget
   FROM Division d, Project p
   WHERE d.did = p.did (+)
   Group by dname;

DNAME	TOTALNOOFPROJECTS	TOTALBUDGET
Research and development	1	6000
accounting	0	-
engineering	2	12000
marketing	2	14000
human resource	1	5000

```
5 rows returned in 0.01 seconds
```

3. For each project, list its name and total number of employees who work on that project.

```
SELECT pname, count(empid) as TotalNoOfEmployees
FROM Project p, Workon w
WHERE p.pid = w.pid
Group by pname;
```

PNAME	TOTALNOOFEMPLOYEES
DB development	8
security system	6
network development	13
Wireless development	12
system development	6
Web development	11

```
6 rows returned in 0.00 seconds
```

4. List the name and ID of employees that worked on more than one projects. (*Note : there are some employees who have same names*).

```
SELECT e.empid, name
FROM Employee e, Workon w
WHERE e.empid = w.empid
group by e.empid, name
having count(pid)>1
Order by empid;
```

EMPID	NAME
3	brian
4	larry
6	peter
7	peter
8	smith
9	chen
16	Justin

19	Kristie
21	Alex
22	Phil
25	Alan
26	Julia
27	Sandra
28	Joe
30	grace

15 rows returned in 0.00 seconds

5. **List the total number of project each employee works on, including employee's name (*note : there are some employees who have same names and some employee may not work on any project*).**

```
SELECT e.name, count(pid) as TotalNoOfProjects
FROM Employee e, Project p
WHERE e.did = p.did (+)
Group by e.name, empid
Order by e.empid;
```

NAME	TOTALNOOFPROJECTS
kevin	2
joan	2
brian	1
larry	0
harry	1
peter	2
peter	1
smith	1
chen	2
kim	0
smith	2
joan	2
kim	2
austin	2
sam	0
Justin	2
Nacy	2
Marilyn	0

Kristie	2
John	1
Alex	2
Phil	2
Steve	1
Jenna	2
Alan	2
Julia	1
Sandra	1
Joe	1
karl	0
grace	1

30 rows returned in 0.00 seconds

6. **List the total number of employees who work on project 'Web development'. Also list the total working hours for this project.**

```
SELECT count(empid) as TotalNoOfEMployees, sum(hours) as TotalHours
FROM Project p, Workon w
WHERE p.pid = w.pid AND LOWER(pname) = 'web development'
group by pname;
```

TOTALNOOFEMPLOYEES	TOTALHOURS
11	390

1 rows returned in 0.01 seconds

7. **List the name of project that 'chen' works on. (*Hint: join 3 tables*)**

```
SELECT pname
FROM employee e, project p, workon w
WHERE e.empid = w.empid and p.pid = w.pid and lower(e.name) = 'chen';
```

PNAME
security system
system development
DB development
Web development
Wireless development

network development

6 rows returned in 0.00 seconds

8. **List the name of manager who work on some projects. (Note, managerID is empid of an employee who is a manager)**

```
SELECT Distinct e.name
FROM Employee e, Division d, Workon w
WHERE d.managerid = e.empid and w.empid = d.managerid;
```

NAME
larry
harry
kevin
brian

4 rows returned in 0.00 seconds

9. **List the name of employee and his/her salary, who work on any project and salary is below $45000. (*Note: don't duplicate an employee in the list)***

```
SELECT DISTINCT name, salary
FROM Employee e, Workon w
WHERE e.empid = w.empid AND Salary < 45000;
```

NAME	SALARY
smith	39000
kevin	32000
brian	37000

3 rows returned in 0.00 seconds

10. **List the name of the employee who work on one or more projects with budget over $5000. *(Note: don't duplicate an employee in the list)***

```
SELECT DISTINCT e.name
FROM Employee e, Workon w, Project p
WHERE e.empid = w.empid AND w.pid = p.pid and Budget>5000
group by e.empid, e.name
having count (*)>=1
order by e.name;
```

NAME
Alan

Julia
Justin
Kristie
Marilyn
Phil
Sandra
brian
chen
grace
harry
joan
karl
larry
peter
smith

```
16 rows returned in 0.01
seconds
```

11. **List the names that are shared among employees. (*Note, This questoin was in assignment 3 but you must use different approach than the one used in assignment 3.Hint: Use different aiases for the same table.)***

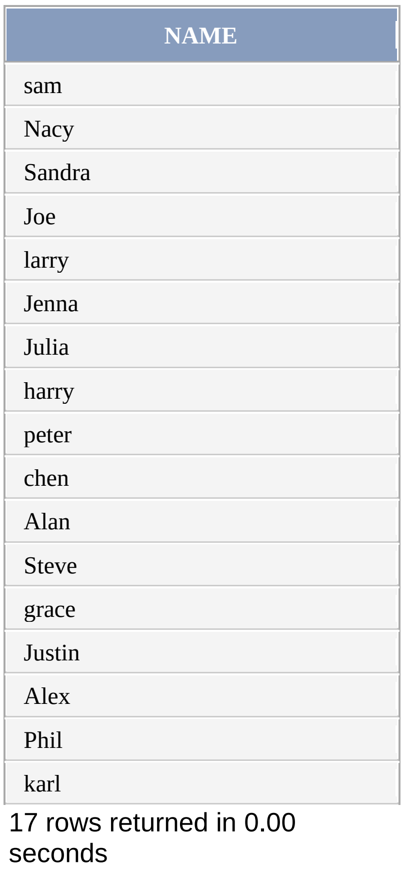
```
SELECT DISTINCT ea.name
FROM employee ea, employee eb
WHERE ea.name = eb.name AND ea.empid != eb.empid;
```

NAME
peter
smith
joan
kim

```
4 rows returned in 0.00
seconds
```

12. **List the name of employeeis he/she work on a project with the budget that is less than 10% of his/her salary. (Hint: join 3 tables)**

```
SELECT name
FROM Employee e, Workon w, Project p
Where e.empid = w.empid And p.pid = w.pid And budget<((10/100)*Salary)
Group by name
Having count (p.pid) >=1;
```



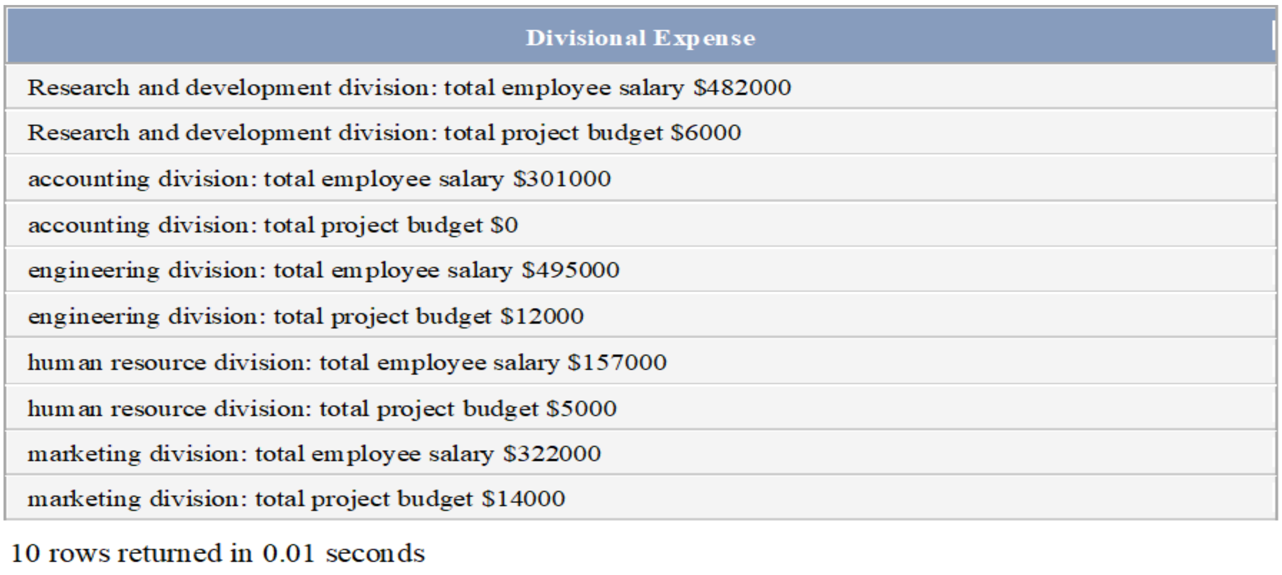
NAME
sam
Nacy
Sandra
Joe
larry
Jenna
Julia
harry
peter
chen
Alan
Steve
grace
Justin
Alex
Phil
karl

17 rows returned in 0.00 seconds

13. Run the following code to learn the use of concatenator ||

```
select 'employee ' || name || ' of Division '|| DID || ' earns $' || salary AS "Employee
Report"
From employee e
order by did
```

Use what you learned from above code to create result show as the follows (*Hint use UNION operation)*



Divisional Expense
Research and development division: total employee salary $482000
Research and development division: total project budget $6000
accounting division: total employee salary $301000
accounting division: total project budget $0
engineering division: total employee salary $495000
engineering division: total project budget $12000
human resource division: total employee salary $157000
human resource division: total project budget $5000
marketing division: total employee salary $322000
marketing division: total project budget $14000

10 rows returned in 0.01 seconds

```
SELECT dname ||' division: Total Employee Total '||sum(salary) as "Divisional Expense"
FROM Employee INNER JOIN Division ON Division.DID = Employee.DID
Group by dname
UNION
SELECT dname ||' division: Total Project Budget '||sum(budget)
FROM Division LEFT OUTER JOIN Project on Division.DID = Project.DID
Group by dname;
```

Divisional Expense
Research and developmentdivision : Total Employee Total482000
Research and developmentdivision : Total Project Budget6000
accountingdivision : Total Employee Total301000
accountingdivision : Total Project Budget
engineeringdivision : Total Employee Total495000
engineeringdivision : Total Project Budget12000
human resourcedivision : Total Employee Total157000
human resourcedivision : Total Project Budget5000
marketingdivision : Total Employee Total322000
marketingdivision : Total Project Budget14000

10 rows returned in 0.00 seconds

14. **Use INTERSECT opertion to list the name of project chen and larry both work on.**

```
SELECT pname
FROM Project p, Employee e, Workon w
Where p.pid = w.pid AND w.EmpID = e.empID AND upper(name)='CHEN'
INTERSECT
SELECT pname
FROM Project p, Employee e, Workon w
Where p.pid = w.pid AND w.empID = e.empID AND upper(name)='LARRY';
```

PNAME
Wireless development
network development
security system

3 rows returned in 0.01 seconds

15. Use MINUS operation to list the name of project chen works on but larry does not.

```
SELECT pname
FROM Project p, Employee e, Workon w
Where p.pid = w.pid AND w.EmpID = e.empID AND upper(name)='CHEN'
MINUS
SELECT pname
FROM Project p, Employee e, Workon w
Where p.pid = w.pid AND w.empID = e.empID AND upper(name)='LARRY';
```

PNAME
DB development
Web development
system development

3 rows returned in 0.00 seconds

1. **Create a view called "high_cost_Division" that shows the ID and the name of divisions that have 5 or more employees and total salary is over $350,000. The view also include the each division's headcount and total salary.**

   Show the code of view create statement and use select statement to show the contents of the view.

```
Create or replace View high_cost_Division
As
SELECT d.did, dname, sum(salary) as TotalSalary, count(e.empid) as TotalEmployee
From division d, employee e
WHERE d.did = e.did and d.did in (select did from employee group by did
                                  having sum(salary)> 350000 and count(empid)>=5)
GROUP BY d.did, dname;
```

```
View created.

    0.00 seconds
```

```
SELECT * From high_cost_Division
```

DID	DNAME	TOTALSALARY	TOTALEMPLOYEE
1	engineering	495000	9
4	Research and development	482000	7

2 rows returned in 0.00 seconds

2. **Create a view named as 'letgo_list' that has the name of employee and his/her salary, who don't work on any project but salary is over company's average. Also show employee's division name. Run select statement to show the contents of this view.**

   Show the code of view create statement and use select statement to show the contents of the view.

```
Create or replace View letgo_list
As
SELECT name, salary, dname
FROM Employee e, Division d
Where e.did = d.did And salary> (select avg(salary) from employee) And empid not in (select
empid from workon);
```

```
 View created.
```

0.01 seconds

```
SELECT * From letgo_list;
```

```
no data found
```

3. **List the name of employee whose salary is higher than ‘Justin ‘. (hint; use subquery)**

```
SELECT empid, name
FROM Employee
Where salary> (Select salary From Employee Where lower(name)='justin');
```

EMPID	NAME
4	larry
5	harry
7	peter
9	chen
21	Alex
22	Phil
23	Steve
24	Jenna
26	Julia
27	Sandra
28	Joe
29	karl

```
12 rows returned in 0.01
seconds
```

4. **List the name of EACH employee in marketing division and the total number of projects the employee works on, as well as the total hours he/she spent on the project(s). Note some employees may have same names.**

   Select e.empid, name, count(pid) as TotalProjects, sum(hours) as TotalHours

   From workon w, employee e, division d

   Where w.empid = e.empid And d.did = e.did And dname in (select dname from division where lower(dname) = 'marketing')

   Group by e.empid, name

   order by e.empid;

EMPID	NAME	TOTALPROJECTS	TOTALHOURS
1	kevin	1	30
6	peter	4	165
13	kim	1	30
16	Justin	2	60
22	Phil	2	70
25	Alan	2	70

```
6 rows returned in 0.00
seconds
```

5. **List the name of project that ‘Chen’ works on but 'Sam' does not work on. (hint: need two inner select statements /sub-queries, use key word 'IN' and 'NOT IN')**

   SELECT pname
   FROM Project
   WHERE pid IN (select pid from workon w, employee e where w.empid=e.empid and lower(name)='chen')
   AND pid NOT IN (select pid from workon w, employee e where w.empid=e.empid and lower(name)='sam');

PNAME
DB development
system development
network development
security system
Web development

5 rows returned in 0.01
seconds

6. **List the name and budget of the project if its budget is over average budget, together with the average budget in query result.**

```
SELECT pname, budget
FROM Project
Where budget > (select avg(budget) from project);
```

PNAME	BUDGET
DB development	8000
system development	7000

2 rows returned in 0.01
seconds

7. **List the total number of project 'larry' does not works on (must use subquery for this query)**

```
Select count(pname) as "TotalProjects"
From Project
Where pid not in (select pid from workon w, employee e where e.empid=w.empid AND
lower(name)='larry');
```

TotalProjects
3

1 rows returned in 0.01
seconds

8. **List the name of employee who work on a project sponsored by accounting division but does not work on a project sponsored by engineering division**

```
SELECT Distinct e.empid, name
FROM Employee e, Workon w
WHERE e.empid = w.empid AND pid IN (select pid From Project p, Division d
                            WHERE p.did = d.did AND lower(dname) = 'accounting'
                            group by pid)
                   AND pid NOT IN (select pid From Project p, Division d
                              WHERE p.did = d.did AND lower(dname) = 'engineering'
                              group by pid)
order by e.empid;
```

output=> no data found

9. **List the name of the project that has most people working in (hint use a subquery at HAVING clause).**

```
SELECT pname, count(*) as Employees
FROM Project p, Workon w
WHERE p.pid = w.pid
GROUP BY p.pid, p.pname
HAVING count(*) = (select max(count(*))
FROM Project p, Workon w Where p.pid=w.pid
GROUP BY p.pid);
```

PNAME	EMPLOYEES
network development	13

1 rows returned in 0.01 seconds

10. **List the total number of employee whose salary is over company's average salary (subquery) for each division, including division name.**

```
SELECT d.did, dname, count(empid) as TotalNoOFEmployee
FROM Employee e, Division d
WHERE e.did = d.did AND salary > (select avg(salary) from Employee)
Group by dname, d.did
order by d.did;
```

DID	DNAME	TOTALNOOFEMPLOYEE
1	engineering	3
2	marketing	3
3	human resource	1
4	Research and development	6
5	accounting	2

5 rows returned in 0.00 seconds

11. List the name of manager (note a manager is an employee whose empid is in division table as managerID) who work on project with 'brian' (use subquery)

```
SELECT DISTINCT name

FROM division d, employee e, workon w

WHERE e.empid = d.managerid and w.empid=e.empid

AND w.pid in (select w.pid

        from employee e,  workon w

        where w.empid=e.empid and lower(name)='brian') and lower(name)!='brian';
```

NAME
larry
kevin

2 rows returned in 0.01 seconds

12. List the name of employees who don't work on project "web development" (Must use subquery. Also you need to write an explanation on why the JOIN operation does not work for this query).

```
SELECT empid, name
From Employee
Where empid not in (select empid from workon w, project p where w.pid = p.pid and
lower(pname) = 'web development');
```

EMPID	NAME
2	joan
4	larry
5	harry
6	peter
7	peter
8	smith
10	kim
11	smith
12	joan
14	austin
15	sam
18	Marilyn

19	Kristie
20	John
22	Phil
24	Jenna
25	Alan
27	Sandra
29	karl

19 rows returned in 0.01 seconds

A SELECT, UPDATE, or DELETE query's WHERE clause is used to limit the number of records affected. The WHERE condition in SQL can be used with logical operators like AND and OR, as well as comparison operators like =. A JOIN clause is used to join rows from two or more tables together based on a common column.

13. List the name of employee who work on more projects than Larry.

```
select e.empid, name
from employee e, workon w
where e.empid = w.empid
group by e.empid, name
having count(w.pid) > (select count(w.pid)

                              from employee e, workon w

                   where e.empid = w.empid and name = 'larry')

order by e.empid;
```

EMPID	NAME
3	brian
6	peter
8	smith
9	chen
26	Julia

5 rows returned in 0.00 seconds

14. List name of project whose budget is SECOND lowest, list budget too. (hint, refer to the video lecture).

```
SELECT pname, budget
FROM project
```

```
WHERE budget = (select min(budget)
                from project p
                where budget >(select min(budget) from project p));
```

PNAME	BUDGET
network development	6000
security system	6000

2 rows returned in 0.01 seconds

15. List the name of project that 'chen' works on but not from chen's division. (hint/pseudocode: find ID of proj. that is IN (ID of proj chen works on returned by a subquery) AND proj's DID NOT IN (DID of chen's returned by a subquery)

```
SELECT pname
FROM project p
WHERE pid IN (select pid

              from workon w, employee e

              where e.empid = w.empid and lower(name)='chen')

             AND did NOT IN (select d.did

                             from employee e, division d

                             where e.did = d.did and lower(name) = 'chen');
```

PNAME
security system
Web development
network development
DB development

4 rows returned in 0.01 seconds

1. **List the name of employee who work on a project sponsored by his/her own division. (Try to use correlated subquery)**

```
SELECT Distinct e.empid, name
FROM Employee e, Workon w
Where e.empid = w.empid and pid IN (select pid from Project p where p.did =
e.did)
order by e.empid;
```

EMPID	NAME
3	brian
6	peter
8	smith
9	chen
14	austin
16	Justin
21	Alex
22	Phil
24	Jenna
26	Julia

10 rows returned in 0.01 seconds

2. **List the name of employee who is working on a project with the budget that is below this project's divisional average project budget (use correlated subquery).**

```
SELECT Distinct e.empid, name
From Employee e, Workon w, Project p
WHERE e.empid = w.empid AND w.pid = p.pid AND Budget < (Select avg(budget)
                                           from Project pp  Where pp.did = p.did)

Order by e.empid;
```

EMPID	NAME
3	brian
4	larry
6	peter

7	peter
8	smith
9	chen
12	joan
14	austin
15	sam
16	Justin
19	Kristie
21	Alex
22	Phil
24	Jenna
25	Alan
26	Julia
27	Sandra
28	Joe
29	karl
30	grace

20 rows returned in 0.02 seconds

3. **List the name of project that some employee(s) who is/are working on it make less than his/her divisional average salary (use correlated subquery).**

```
SELECT Distinct pname
FROM Project p, workon w, Employee e
WHERE p.pid = w.pid AND w.empid = e.empid AND Salary < (select avg(salary)

                                                        from employee ee
                                                        where ee.did = e.did);
```

PNAME
DB development
security system
network development
Wireless development
system development
Web development

6 rows returned in 0.00 seconds

4. **List the total number of division that has 3 or more employees working on projects. For this query I built a framework of the code, you just need to fill in the right code in ____ , and then run the code to get the result.**

```
select count(did)
from _____
where did in
   (select ____
    from ___ , _____
    where _______
    group by ___
    having ________ )
```

```
SELECT count(did)  as TotalNoOfDivsion
FROM Division
Where did IN (select did from project p, workon w where p.pid = w.pid
group by did
having count(empid)>=3);
```

TOTALNOOFDIVSION
4

1 rows returned in 0.01 seconds

5. **List the total number of projects 'accounting' division manager does not works on.**

```
SELECT count(pid) as TotalNoOfProject
FROM Division d, Workon w
WHERE d.managerid = w.empid and lower(dname) != 'accounting';
```

TOTALNOOFPROJECT
6

1 rows returned in 0.01 seconds

6. **List the name of the employees (and his/her DID) who work on more projects than his/her *divisional* colleagues. (hint: correlated subquery, also use having , compare count() to count, use " … having count (pid) >=ALL (select count (pid) …..)**

```
SELECT did,name
FROM employee e, workon w
WHERE e.empid  = w.empid
group by did, name
having count(pid) > = all (select count(pid)
                              from workon ww, employee ee
             where ee.empid  = ww.empid and e.did= ee.did group by ww.empid)

order by did;
```

DID	NAME
1	chen
2	peter
3	brian
4	Julia
4	smith
5	larry

6 rows returned in 0.00 seconds

7. **List the name of the division that has more than three employees whose salary is greater than his/her divisional average salary.**

```
 SELECT Distinct dname
From division d, employee e
Where d.did = e.did and salary > (select avg(Salary) from employee ee
                                  where ee.did = d.did)
                                  Group by dname
                                  Having count(empid) > 3;
```

DNAME
Research and development

1 rows returned in 0.00 seconds

8. **List the name of the project that 'chen' spend more hours than Sam. Also list the working hours of both employees on this project. (correlated subquery)**

```
SELECT pname, wa.hours, wb.hours
FROM Project p, Workon wa, Workon wb
WHERE p.pid = wa.pid AND wa.pid = wb.pid
AND wa.hours > wb.hours AND wa.Empid = (select empid from employee
                               where lower(name)='chen')
AND wb.empid = (select empid from employee where lower(name)='sam');
```

ANS:- no data found

9. **List the name of the employee that has the lowest salary in his division and list the total number of projects this employee is work on (use corelated subquery, refer to the example in the lecture video)**

```
SELECT name, did, count(pid)
FROM employee e, Workon w
Where e.empid = w.empid(+) AND salary = (select min(salary) from employee ee
Where e.did = ee.DID)
Group By name, did
Order By did
```

NAME	DID	COUNT(PID)
joan	1	0
kevin	2	1
brian	3	4
smith	4	4
kim	5	0

5 rows returned in 0.02 seconds

10. List the name of employee in Sam's division who works on a project that Sam does NOT work on.

```
SELECT distinct empid, name
FROM Employee
WHERE did in (select did from employee e where lower(name) = 'sam')
AND empid in (select empid from workon where pid not in (select pid
                                           from employee ee, workon w
                                           where ee.empid = w.empid and lower(name)= 'sam'))
order by empid;
```

EMPID	NAME
4	larry
18	Marilyn
29	karl

3 rows returned in 0.01 seconds

11. List the name of divisions that sponsors project(s) Chen works on.

```
SELECT dname
FROM Division d
Where did in (select p.did from workon w, Employee e, Project p
         where w.empid = e.empid and w.pid = p.pid
         and name = 'chen');
```

DNAME
engineering
marketing
human resource
Research and development

4 rows returned in 0.00 seconds

12. **List the name of division (d) that has employee who work on a project (p) not sponsored by this division. (hint in a corelated subquery where d.did <> p.did)**

```
SELECT DISTINCT dname
FROM Division d, Employee e
WHERE d.did = e.did AND e.empid IN (select empid
                                    from workon w, project p
                                     where w.pid = p.pid and p.did != d.did);
```

DNAME
Research and development
accounting
engineering
marketing
human resource

5 rows returned in 0.01 seconds

13. **List the name of employee who work with Chen on some project(s). (Find an employee who works on a project that chen also works on, don't show same employee multiple times in result)**

```
SELECT Distinct empid, name
FROM Employee e
WHERE empid IN (select empid from workon
                where pid in (select pid
                              from employee ee, workon w
                              where ee.empid = w.empid and lower(name)= 'chen')
                              and lower(name) != 'chen')

ORDER BY empid;
```

EMPID	NAME
1	kevin
3	brian
4	larry
5	harry
6	peter
7	peter
8	smith

12	joan
13	kim
14	austin
15	sam
16	Justin
17	Nacy
18	Marilyn
19	Kristie
21	Alex
22	Phil
23	Steve
24	Jenna
25	Alan
26	Julia
27	Sandra
28	Joe
29	karl
30	grace

25 rows returned in 0.01 seconds

14. **Increase the salary of employees in engineering division by 10% if they work on more than 1 project.**

```
UPDATE Employee set Salary = salary*1.1
WHERE empid in (select e.empid from employee e, division d
              where e.did = d.did and lower(dname) = 'engineering')
              AND empid IN (select empid from workon
             group by empid
             having count(pid) > 1);
```

```
3 row(s) updated.
```

0.00 seconds

15. **Increase the budget of a project by 10% if it has more than two employees working on it.**

```
UPDATE project set budget = budget*1.1
WHERE pid in (select pid from workon
                    group by pid
                     having count(empid) >2);
```

```
6 row(s) updated.
```

0.00 seconds

a1. Use CREATE statement to create a table Client (AcctNo, ClientName, phone). Note AcctNo is primary key and you must define a constraint for this primary key in CREATE statement. Use Char for the data type of phone. Show the statement.

```
CREATE TABLE Client
(AcctNo integer,
ClientName varchar(30),
Phone char(10),
Constraint Client_AcctNo_pk Primary Key (AcctNo));
```

```
Table created.
```

0.14 seconds

a2. Use INSERT statement to add two client records (insert two client with different account number, make up your own data for clients). Show the INSERT statements and use select statement to show the table contents.

```
INSERT into Client
Values (81433639, 'Bunty', 5512637035);
INSERT into Client
Values (36032901, 'Rocky', 8369071177);
```

```
SELECT * From Client;
```

ACCTNO	CLIENTNAME	PHONE
81433639	Bunty	5512637035
36032901	Rocky	8369071177

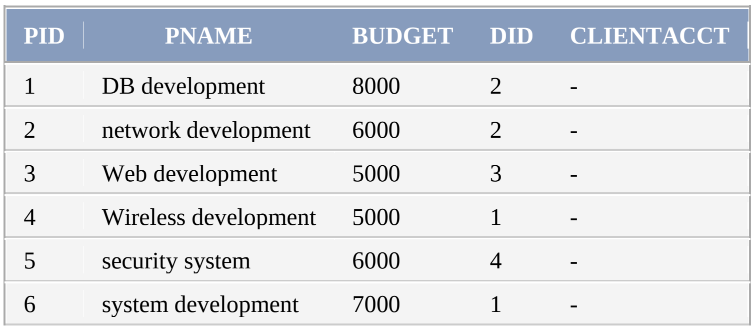
2 rows returned in 0.00 seconds

a3. Use ALTER statements to add a foreign key *ClientAcct,* into the Project table, which has values of the AcctNo of Client . So that table Client has a 1:M relationship with table Project. Note, you need to use TWO ALTER statements, one for adding ClientAcct into Project table; one for adding foreign key constraint named as project_ClientAcct_fk into Project table. Show the ALTER statements.

```
ALTER Table Project
Add ClientAcct integer;

ALTER Table Project
Add Constraint Project_ClientAcct_fk Foreign Key (ClientAcct) REFERENCES Client(AcctNo);

Select * From Project;
```



PID	PNAME	BUDGET	DID	CLIENTACCT
1	DB development	8000	2	-
2	network development	6000	2	-
3	Web development	5000	3	-
4	Wireless development	5000	1	-
5	security system	6000	4	-
6	system development	7000	1	-

6 rows returned in 0.01 seconds

a4 use update statement to assign account number to the project(s) with the name that has word 'system' in it. Write another update statement to assign another account number to the rest of the projects.

```
UPDATE Project
SET ClientAcct = 81433639
WHERE pname like '%system%';

UPDATE Project
SET ClientAcct = 36032901
WHERE ClientAcct is null;

Select * From Project;
```

PID	PNAME	BUDGET	DID	CLIENTACCT
1	DB development	8000	2	36032901
2	network development	6000	2	36032901
3	Web development	5000	3	36032901
4	Wireless development	5000	1	36032901
5	security system	6000	4	81433639
6	system development	7000	1	81433639

6 rows returned in 0.00
seconds

a5. Test the the foreign key constraint by inserting a new project with wrong client account number. Show the result.

INSERT into Project
Values (7, 'Cloud Computing', 47000, 4, 03823412);

```
ORA-02291: integrity constraint (BUNTY.PROJECT_CLIENTACCT_FK) violated - parent key not
found
```

0.00 seconds

a6. User ALTER statement to Add an attribute Project_Count into Employee table (data type to be integer, *refer to the data type used for Workon table (hours) in loadDB file*).

ALTER Table Employee
Add Project_Count integer

```
Table altered.
```

0.02 seconds

a7. Use UPDATE statement to fill the value of Project_count of each employee record in Employee table. Namely, add the count of total number of projects an employee works on into Project_count in Employee table for each employee. Hint: you need a subquery in Update statement as follows

Update __________

Set ______ = (select count (pid) ......)

Show the contents of Employee after update.

UPDATE Employee e

```
SET Project_Count = (Select Count(pid) From Workon w Where w.empid = e.empid
Group by e.empid);

UPDATE Employee
SET Project_Count = '0'
WHERE Project_Count is NULL;

Select * From Employee;
```

EMPID	NAME	SALARY	DID	PROJECT_COUNT
1	kevin	32000	2	1
2	joan	42000	1	0
3	brian	44770	3	4
4	larry	99220	5	3
5	harry	92000	4	1
6	peter	54450	2	4
7	peter	82280	3	3
8	smith	47190	4	4
9	chen	103951.1	1	6
10	kim	46000	5	0
11	smith	46000	1	0
12	joan	48000	1	1
13	kim	49000	2	1
14	austin	46000	1	1
15	sam	52000	5	1
16	Justin	75020	2	2
17	Nacy	52000	1	1
18	Marilyn	52000	5	1
19	Kristie	76133.2	1	2
20	John	52000	3	0
21	Alex	101022.9	1	2
22	Phil	87120	2	2
23	Steve	74000	4	1
24	Jenna	69000	1	1
25	Alan	75020	2	2
26	Julia	83490	4	4
27	Sandra	87120	4	2
28	Joe	89540	4	2
29	karl	69000	5	1
30	grace	75020	4	3

30 rows returned in 0.00
seconds

a8. Create a table Promotion_list (EMPID, Name, Salary, DivisionName).

```
CREATE Table Promotion_list
(EMPI_ID integer,
Name VarChar(30),
Salary float,
Dname varchar(25));
```

```
Table created.
```

0.01 seconds

a9. Load Promotion_list with the information of employees who make less than company average and work on at least 2 projects. (Hint use INSERT INTO SELECT statement). Show the code and result.

```
INSERT into promotion_list
SELECT empid, name, salary, did
FROM employee
WHERE empid IN (select e.empid

          from employee e, workon w
          where e.empid = w.empid and salary < all (select avg(salary)
                                                    from employee ee)
                                                    group by e.empid
                                                    having count(pid) >= 2)
```

```
Select * From promotion_list
```

EMPI_ID	NAME	SALARY	DNAME
8	smith	39000	4
6	peter	45000	2
3	brian	37000	3
19	Kristie	52000	1

4 rows returned in 0.00 seconds

Part B

b1. Increase the budget of a project by 5% if there is a manager working on it.

```
UPDATE Project
SET budget = budget*1.05
WHERE pid in (select pid from workon where empid in (select managerid from division))

select * from project
```

PID	PNAME	BUDGET	DID	CLIENTACCT
1	DB development	8400	2	36032901
2	network development	6300	2	36032901
3	Web development	5250	3	36032901
4	Wireless development	5250	1	36032901
5	security system	6300	4	81433639
6	system development	7000	1	81433639

6 rows returned in 0.00 seconds

b2. List the name of employee who does not work on a project sponsored by his/her own division. (correlated subquery with NOT EXISTS)

```
SELECT empid, name
FROM employee e
WHERE NOT EXISTS (select * from workon w, project p
          where e.did = p.did and w.pid = p.pid and e.empid = w.empid)
```

EMPID	NAME
1	kevin
2	joan
4	larry
5	harry
7	peter
10	kim
11	smith

12	joan
13	kim
15	sam
17	Nacy
18	Marilyn
19	Kristie
20	John
23	Steve
25	Alan
27	Sandra
28	Joe
29	karl
30	grace

20 rows returned in 0.01 seconds

b3. List the name of project that has budget that is higher than ALL projects from 'marketing' division.

```
SELECT pname
FROM Project p
WHERE budget > (select max(budget) from project p, division d
                        where p.did = d.did and lower(dname) = 'marketing')
```

no data found

b4. For each division, list its name and its divisional total salary and its total project the division has (Show three columns in result. Hint; use correlated subquery in top SELECT clause. Note, this query cannot be done by simply join three tables as follows, can you tell me why ? (a bonus point)

```
select dname, sum(salary), count(pid)
from division d, employee e, project p
where d.did = e.did and d.did = p.did
group by dname
```

```
SELECT dname, (select sum(salary) from Employee e where e.did = d.did) as TotalSalary,
(select count(pid) from project p where p.did = d.did)  as ProjectCount
From Division d;
```

DNAME	TOTALSALARY	PROJECTCOUNT

engineering	495000	2
marketing	322000	2
human resource	157000	1
Research and development	482000	1
accounting	301000	0

5 rows returned in 0.00 seconds

Because of the additional join between Employee and Project, this query cannot be done by just joining three tables. Aggregate functions produce invalid data.

b5. List the name of employee who spend more time working on projects than employee 'Grace'', Show three columns in result: name of employee, project-hours spent by the employee, grace's project-working hours.

```
SELECT empid, name, total_hours , (select sum(hours)
            from workon w, employee e where w.empid = e.empid
            and lower(name) = 'grace'
            group by e.empid  , name) AS grace_hours



FROM (select e.empid  , name , sum(hours) as total_hours
  from workon w, employee e where w.empid = e.empid
 group by e.empid  , name)

WHERE total_hours > (select sum(hours) as total_hours
from workon w, employee e where w.empid = e.empid
and lower(name) = 'grace'
group by e.empid  , name)
order by empid
```

EMPID	NAME	TOTAL_HOURS	GRACE_HOURS
3	brian	300	100
4	larry	105	100
6	peter	165	100
8	smith	120	100
9	chen	180	100
26	Julia	110	100

6 rows returned in 0.00 seconds

b6. List The name of division that has employee(s) who work on other division's project . (correlated subquery)

```
SELECT Distinct dname
FROM division d, employee e
WHERE d.did = e.did AND e.empid IN (select w.empid from workon w , project p
 where p.pid = w.pid and p.did != e.did);
```

DNAME
Research and development
accounting
engineering
marketing
human resource

5 rows returned in 0.01 seconds

b7. List the name of employee who works ONLY with his/her divisional colleagues on project(s). (Hint, namely, the employee (e) firstly works on project(s) , and secondly, there NOT EXISTS a project that (e) works on and another employee (ee) also works on but they are from different divisions.)

```
SELECT Distinct e.empid, name
from employee e, workon w, project p, division d
WHERE  e.empID = w.empID AND d.did = p.did AND w.pid = p.pid
        AND e.did = p.did AND e.name NOT IN (select distinct name
              FROM workon, project, employee
              WHERE workon.pid = project.pid AND employee.empID = workon.empID AND
employee.did != project.did);
```

EMPID	NAME
14	austin
24	Jenna

2 rows returned in 0.01 seconds

You may need to watch the lecture video to review the subject on Trigger.

A1. Briefly describe the roles of triggers. (20 words)

Triggers are automatically performed in response to specified table events. To enforce data constraints and referential integrity by updating data that is impacted by the triggering event.

A2. Create a trigger that when a 'work-on' record is added (namely, an employee works on a new project so an new row is INSERTED into Workon table) the trigger will increase the salary of THIS employee who works on THIS project by 1% if this employees' salary is less than 85000. (for example, when your add a row (1, 30, 20) into workon table (Using INSERT statement), assuming employee 30 exists and does not work on project 1 currently, your trigger code needs to find out (use Where clause) if the employee's salary is less than 85000, if so, increase his salary.(*Hint: use After trigger, and use* ***:new*** *to refer the workon table after the change*).

Show me:

(1).trigger code.

```
Create or replace trigger trigger_workon
After insert on workon
For each row
Begin
Update employee
Set salary = salary + salary*0.01
Where salary<85000 and employee.empid = :new.empid;
End;
```

```
Trigger created.
```

0.00 seconds

(2) the table contents of employee and workon before triggering event (insert).

select * from employee

EMPID	NAME	SALARY	DID	PROJECT_COUNT
1	kevin	32000	2	1
2	joan	42000	1	0
3	brian	37000	3	4
4	larry	82000	5	3
5	harry	92000	4	1
6	peter	45000	2	4
7	peter	68000	3	3
8	smith	39000	4	4
9	chen	71000	1	6
10	kim	46000	5	0
11	smith	46000	1	0

12	joan	48000	1	1
13	kim	49000	2	1
14	austin	46000	1	1
15	sam	52000	5	1
16	Justin	62000	2	2
17	Nacy	52000	1	1
18	Marilyn	52000	5	1
19	Kristie	52000	1	2
20	John	52000	3	0
21	Alex	69000	1	2
22	Phil	72000	2	2
23	Steve	74000	4	1
24	Jenna	69000	1	1
25	Alan	62000	2	2
26	Julia	69000	4	4
27	Sandra	72000	4	2
28	Joe	74000	4	2
29	karl	69000	5	1
30	grace	62000	4	3

30 rows returned in 0.00 seconds

select * from workon

PID	EMPID	HOURS
3	1	30
2	3	40
5	4	30
6	6	60
4	3	70
2	4	45
5	3	90
3	3	100
6	8	30
4	4	30
5	8	30
6	7	30
6	9	40
5	9	50
4	6	45
2	7	30

1	8	30
2	9	30
1	9	30
2	8	30
1	7	30
1	5	30
1	6	30
2	6	30
2	12	30
3	13	30
4	14	20
4	15	40
2	19	30
1	19	30
5	18	30
3	17	30
4	25	30
3	16	30
2	16	30
2	22	30
3	23	30
4	24	20
6	25	40
3	21	40
4	26	20
4	27	40
2	27	30
1	26	30
5	26	30
3	26	30
4	28	30
3	28	30
2	29	30
2	30	30
3	30	30
4	21	20
6	22	40
1	30	40
3	9	10
4	9	20

56 rows returned in 0.01 seconds

(3) table contents after triggering event. This way I can see if the trigger works or not. Also show your INSERT statement (insert a row into workon table to test your trigger).

Insert into workon values (5,1,32);

```
1 row(s) inserted.
```

0.01 seconds

EMPID	NAME	SALARY	DID	PROJECT_COUNT
1	kevin	32643.2	2	1
2	joan	42000	1	0
3	brian	37000	3	4
4	larry	82000	5	3
5	harry	92000	4	1
6	peter	45000	2	4
7	peter	68000	3	3
8	smith	39000	4	4
9	chen	71000	1	6
10	kim	46000	5	0
11	smith	46000	1	0
12	joan	48000	1	1
13	kim	49000	2	1
14	austin	46000	1	1
15	sam	52000	5	1
16	Justin	62000	2	2
17	Nacy	52000	1	1
18	Marilyn	52000	5	1
19	Kristie	52000	1	2
20	John	52000	3	0
21	Alex	69000	1	2
22	Phil	72000	2	2
23	Steve	74000	4	1
24	Jenna	69000	1	1
25	Alan	62000	2	2
26	Julia	69000	4	4
27	Sandra	72000	4	2
28	Joe	74000	4	2
29	karl	69000	5	1
30	grace	62000	4	3

```
30 rows returned in 0.00
seconds
```

Write your trigger code in notepad and paste into SQL window. If you see "trigger created " then you can test the trigger by *inserting a row into workon table*. Then use select * from employee to see if or not the corresponding salary of qualified employee is changed). Make sure that your trigger does not have error. Note, if you have tried very hard and your code is still not working, do not worry, just submit your code. You may still get most of the credits for this work if the logic is correct.

(B) DB transactions and Recovery

1. Based on the lecture videos fill the blank in following paragraph,

In a concurrent transaction processing environment, **concurrency** problem may occur, which is while a transaction's results were overwritten by the results from another transaction. To solve this problem, database systems use **resource locking** approach. But this solution may cause another problem called as **deadlock**, which can be dealt with by approaches such as (a) **deadlock prevention** or (b) **deadlock detection**.

2. What is serializability and describe the way to achieve it. (at least 40 words)

The classic concurrency scheme is serializability. It ensures that a schedule for processing concurrent transactions is comparable to one in which the transactions are executed sequentially in some sequence. It is assumed that read and write operations are used to access the database.

We must use the 2 Phase Locking technique to achieve serializability. This system encrypts and decrypts data in two stages:

I. Growing phase: New locks on data items may be acquired but none can be released. The system locks all resources that may be required to complete a transaction and does not release them until the transaction is finished.

II. Shrinking phase: Existing locks may be released but no new locks can be acquired. During this time, the system releases all previously locked resources for a transaction; no new resources can be locked for the same transaction.

3. Briefly describe the ACID features of transaction in DB systems. (at least 60 words)

The abbreviation ACID refers to the four features of a transaction in transaction processing: atomicity, consistency, isolation, and durability.

Atomicity -All data changes are treated as if they were a single operation. That is, either all of the modifications are made, or none of them are made. The atomicity feature assures that if a debit is successfully made from one account, the matching credit is made to the other account in an application that transfers cash from one account to another.

Consistency - When a transaction begins and ends, the data is in a consistent state. The consistency attribute, for example, ensures that the total value of funds in both accounts is the same at the start and end of each transaction in an application that transfers funds from one account to another.

Isolation - Other transactions cannot see a transaction's intermediate state. As a result, concurrently running transactions appear to be serialized. For example, in an application that transfers funds from one account to another, the isolation property assures that the transferred funds are seen by another transaction in either one of the accounts, but not both, nor neither.

Durability - Changes to data persist when a transaction completes successfully and are not undone, even if the system fails. The durability attribute, for example, assures that modifications made to each account will not be reversed in an application that transfers cash from one account to another.

3. Considering the following situation for the process of database recovery:

There are 3 transactions (T1, T2, T3) since the last backup (SAVE).
T1 finished successfully before the latest checkpoint p1.
T2 finished successfully after p1.
T3 was aborted after p1 due to a main memory crash.

Describe/explain

(a). the database recovery processes regarding T1/T2/T3 if the memory crashes?

T1 is successfully saved before the checkpoint, and T2 is successfully saved after the checkpoint, therefore T1 is saved in any circumstance, and T2 is even after the check point, but the user has committed on this. From the RAM, it will be moved to the datafiles that are contained in the database that is present on the hard disk.

However, because the T3 transaction was not moved to the database, the RAM was corrupted, and the checkpoint was triggered. The T3 will then be unrecoverable, but the others will be.

(b). the database recovery processes regarding T1/T2 /T3 if the hard drive crashes (in stead of memory crashes) after p1?

Everything on the hard disk will be lost if it crashes. T1 and T2 can be retrieved if there is a backup on the actual data.

You need to answer questions (a) and (b) as detailed and specific as possible. Refer to the example of using checkpoint in recovery processes described in the lecture video. ***Describe the recovery process for each transaction. Indicate what files or images are used and why.***

(C) Queries

1. List the name of division that has more employees whose salary is above the divisional average salary than any other divisions.

SELECT dname, count(empid) as TotalEmployee

FROM Division d, Employee e

Where d.did = e.did AND salary > (select avg(salary) from employee ee

where ee.did = e.did)

Group by dname

having count(e.empid) = (select max(count(ee.empid)) from employee ee

where ee.salary>(select avg(eee.salary) from employee eee where eee.did = ee.did) group by ee.did)

DNAME	TOTALEMPLOYEE
Research and development	5

1 rows returned in 0.02 seconds

2. List the total number of employees from Chen's division who work with Chen on projects. (note if a Chen's divisional colleague works on more than one projects with Chen, this colleague should be only count once.

Select count (distinct e.empid) as TotalEmployee
From employee e, workon w where e.empid = w.empid
AND did in (select did from employee where lower(name) = 'chen') And pid in (select pid from employee e, workon w where e.empid = w.empid and lower(name) = 'chen') And lower(name) != 'chen'

TOTALEMPLOYEE
6

1 rows returned in 0.06 seconds

3. For each project, list the name of project, and (1) the total number of employee working on it, (2)the total number of employees who work on it but from other division (not the division that sponsors this project) (3) the total number of employee who work on it and are from the same division that sponsors this project. Hint, the structure of your code is as follows

Select pname ,

(a select statement that returns data in (1)) total,

(a select statement that returns data in (2)) outsiders,

(a select statement that returns data in (3)) insiders

From Project

```
Select pp.pname , total , outsiders , insiders

From (select pname,count(distinct w.empid) as total

    from employee e, workon w, project p where e.empid=w.empid and w.pid=p.pid

    group by pname

    order by pname)  pp


Left join (select pname, count(distinct w.empid) as outsiders

            from employee e, workon w, project p where w.empid=e.empid and w.pid=p.pid

            and e.empid in (select   distinct e.empid

                        from employee e, workon w, project p where e.empid = w.empid and p.pid = w.pid

                        and e.did != p.did)

            group by pname

            order by pname


            )     p3

On pp.pname = p3.pname

Left join (select pname,count(distinct w.empid) as insiders

        from employee e left join workon w on w.empid=e.empid join project p on w.pid=p.pid

            where e.empid in (select   distinct e.empid

                        from employee e left join workon w on e.empid = w.empid left join project p on p.pid = w.pid

                        where e.did = p.did)
```

group by pname

order by pname) p4

on pp.pname = p4.pname

PNAME	TOTAL	OUTSIDERS	INSIDERS
DB development	8	8	4
Web development	11	11	5
Wireless development	12	10	7
network development	13	13	6
security system	6	6	4
system development	6	6	4

6 rows returned in 0.00 seconds

4. (Bonus) List the name of division that has 2nd highest number of employees working on company's projects.

select dname

from division d, employee e

where d.did = e.did

group by dname

having count(e.empid) = (select max(count(distinct ee.empid)) from employee ee, workon ww where ww.empid=ee.empid group by ee.did

having count(distinct ee.empid)< (select max(count(distinct ee.empid)) from employee ee, workon ww where ww.empid=ee.empid group by ee.did))

DNAME
marketing

```
1 rows returned in 0.01
seconds
```