LAB-1

Date __/__/__
Page _____

(Q) Illustrate an example to create a table student which contains the following attributes stdid, stdname, dob, doj, fees, gender.

I/P

```
CREATE DATABASE temp;
USE temp;

CREATE TABLE Student(
        Stdid INT PRIMARY KEY,
        Stdname VARCHAR(50),
        dob DATE,
        doj DATE,
        fees INT,
        gender VARCHAR(10));

INSERT INTO Student VALUES
(1, "Bheem", "2000-03-01", "2022-08-01",
 1250000, "male"),
(2, "Bhupender", "1999-09-12", "2022-09-03",
  250000, "male"),
(3, "Dogesh", "2001-12-23", "2021-07-15",
  90000, "Female");

SELECT * FROM Student;
```

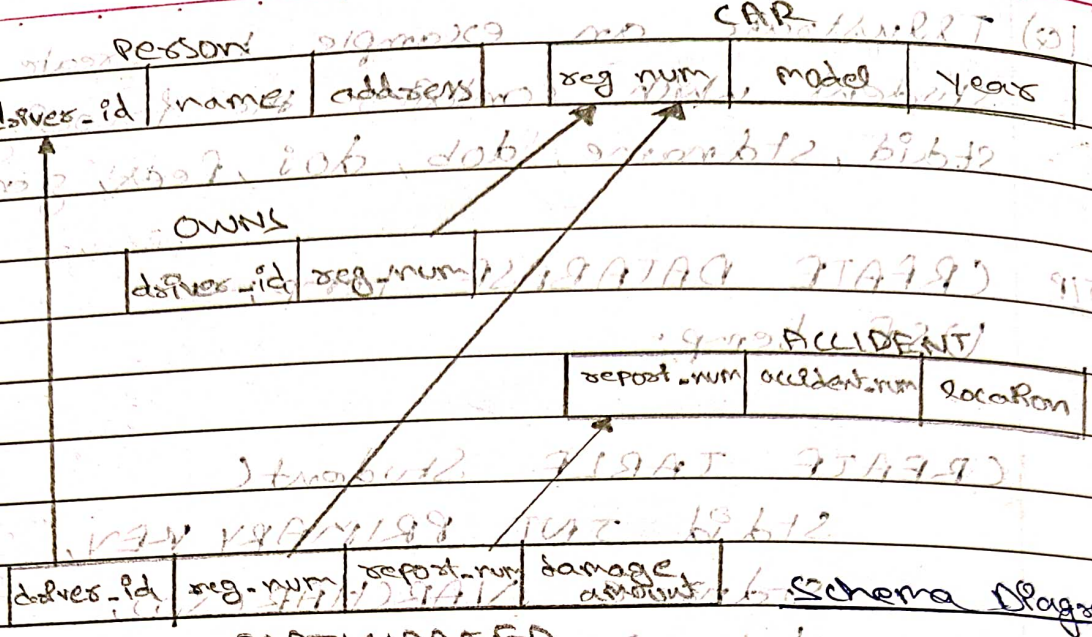
O/P

Stdid	Stdname	dob	doj	fees	gender
1	Bheem	2002-03-01	2022-08-01	250000	male
2	Bhupender	1999-09-12	2022-09-03	250000	male
3	Dogesh	2001-12-23	2021-07-15	90000	female

LAB-3

Date __/__/__
Page

Q



Schema Diagram

(a) update the damage amount to 25000 for the car with a specific reg_num ('KA05 3408') for which the accident report number was

(b) UPDATE participated
SET damage amount = 25000
WHERE reg_num = 'KA05 3408' AND
report_num = 12;

(c) Add a new accident to the database

(d) INSERT INTO accident VALUES
(16, "2008-03-15", "Domlur");

(e) Display Accident date and location.

(f) SELECT accident_date, location
FROM accident;

(g) Display driver id who did the accident damage greater than or equal to Rs 25000

Date __/__/__
Page _____

-> SELECT driver_id
FROM participated
WHERE damage_amount >= 25000;

- Find the average damage amount.

-> SELECT avg(damage_amount)
FROM participated;

Date __/__/__
Page

LAB-3

- List the entire participated relation in the Descending order of damage Amount.

→ SELECT * FROM participated
ORDER BY damage_amount DESC;

- find the average damage amount

→ SELECT AVG(damage_amount)
FROM participated;

- find the max damage amount

→ SELECT MAX(damage_amount)
FROM participated;

- List the name of drivers whose damage is greater than Avg damage Amount.

→ SELECT DISTINCT name, damage_amount
FROM person p, participated pp
WHERE pp.driver_id = p.driver_id
AND pp.damage_amount > (SELECT AVG(damage_amount
FROM participated);

- Display the entire CAR relation in the order of manufacturing year.

→ SELECT * FROM car
ORDER BY year ASC;

- Find the number of accidents in which cars belonging to a specific model were involved.

Date: __/__/____
Page: _____

→ SELECT (count (report_num)
FROM car c, participated p
WHERE c.reg_num = p.reg_num and
c.model = "Lancer";

- Find the total numbers of people who owned cars that were involved in accidents in 2008.

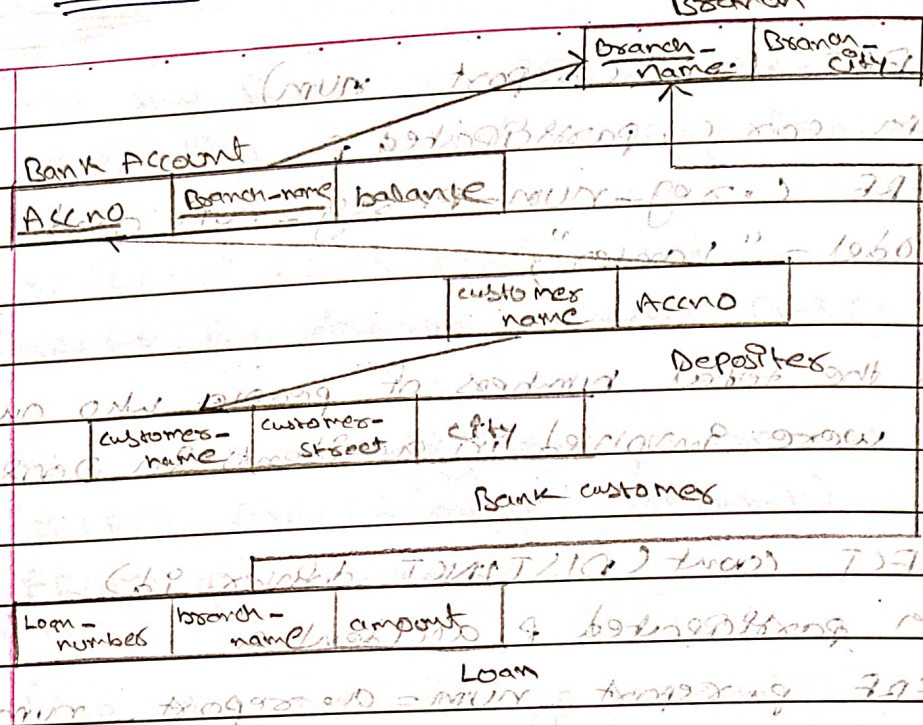
→ SELECT count (DISTINCT driver_id)
FROM participated p, accident a
WHERE p.report_num = a.report_num
AND a.accident_date LIKE "2008%";

- Delete the tuple where damage amount is below the avg damage amount.

→ DELETE FROM participated
WHERE damage_amount <= 23000;

LAB-4

Date __/__/__
Page



(Q) Display the branch name and assets from all branches in lakhs of rupees and rename the assets column to "assets in lakhs".

I/P
```
SELECT branch_name, assets as 'assets in lakh'
FROM branch;
```

O/P

branch_name	assets in lakh
SBI_jantarmantar	20000
SBI_parliamentRoad	10000
SBI_ResidencyRoad	10000
SBI_shivajiRoad	20000
SBI_chamarajpet	50000

(Q) Find all customers who have at least two account at the same branch (ex: SBI_ResidencyRoad)

I/P
```
SELECT acc no, branch_name
FROM branch_account
GROUP BY branch_name
HAVING (count(branch_name) > 1;
```

Date __/__/__
Page _____

O/P

acc no	branch_name
2	SBI_ResidencyRoad
3	SBI_Shivaji Road
4	SBI_ParliamentRoad
5	SBI_Jantarmantar.

(Q) Create a view which gives each branch the sum of amount of all loans at the branch.

I/P

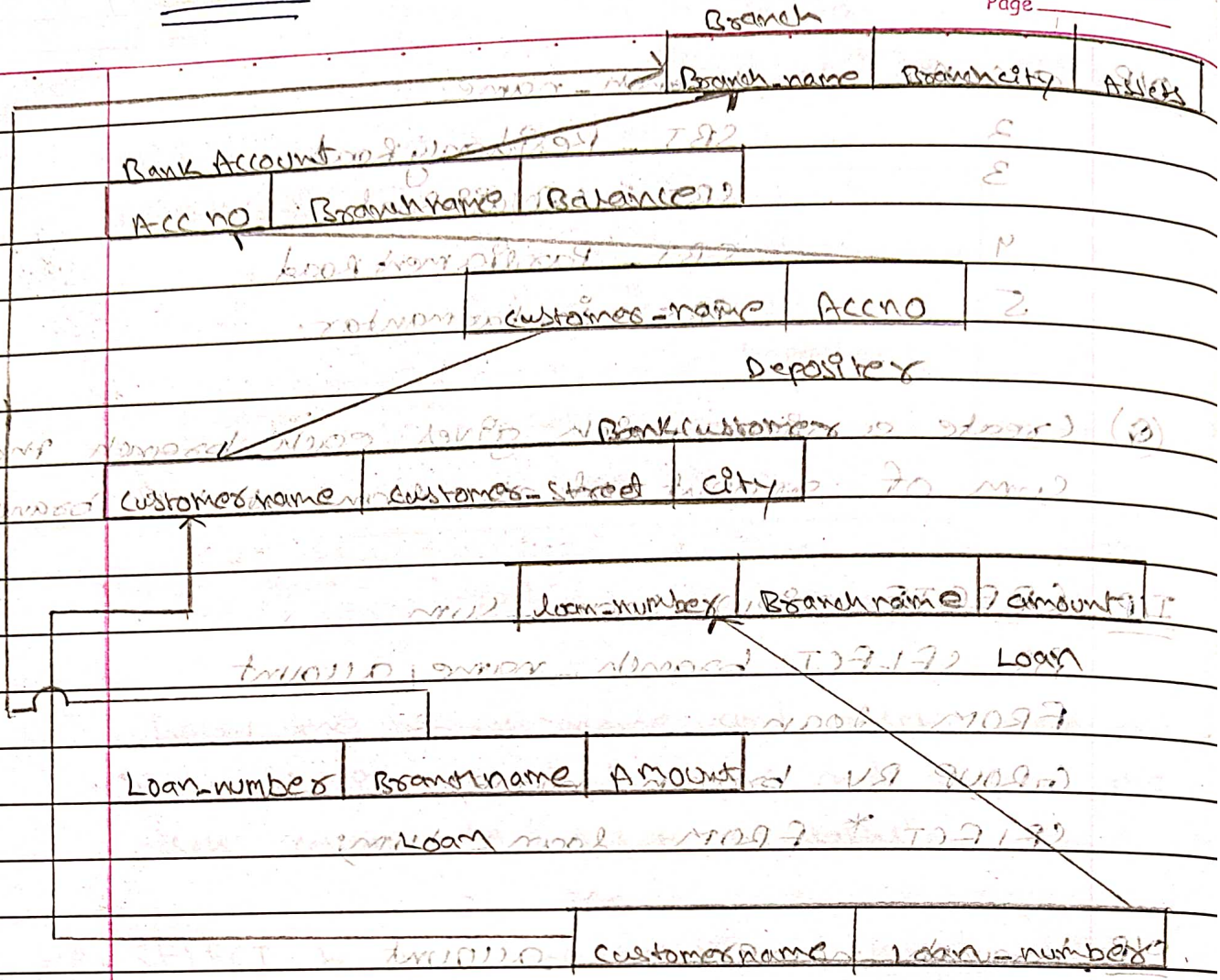
```
CREATE view loan_sum
AS SELECT branch_name, amount
FROM loan
GROUP BY branch_name;
SELECT * FROM loan_sum;
```

O/P

branch_name	amount
SBI - chamarajpet	1000
SBI - ResidencyRoad	2000
SBI - Shivaji Road	3000
SBI - ParliamentRoad	4000
SBI - Jantarmantar	5000

LAB-5

Date __/__/__
Page



(Q) Find all the customers who have an account at all the branches located in a specific city (Ex. Delhi

I/P
```
SELECT customer_name FROM bankcustomer
WHERE customer_city = 'Delhi';
```

O/P
```
Nikhil
Ravi
```

(Q) Find all customers who have a loan at the bank but do not have an account.

I/P
```
SELECT B.customer_name
FROM borrower B
WHERE B.customer_name NOT IN (
```

Date __/__/__
Page _____

SELECT D.customer_name
FROM depositer D
);

O/P customer_name
Moham

(Q) Find the names of all branches that have greater assets than all branches located in Bangalore.

I/P SELECT branch_name FROM branch WHERE assets > some (SELECT assets FROM branch WHERE branch_city = 'Bangalore');

O/P branch_name
SRI-Chamrajpet
SRI-Jantarmantar
SRI-Sivaji Road

(Q) Demonstrate how you delete all account tuples at every branch located in a specific city (Ex. Bombay)

I/P DELETE FROM bankcustomer
WHERE customer_city = 'Bangalore';

(Q) Update the balance of all accounts by 5%

I/P UPDATE bankAccount
SET balance = balance * 1.05;
SELECT * FROM bankaccount;

Output:-

Date __/__/__
Page

acc_no	branch_name	balance
1	SBI_chamrajpeet	22050
2	SBI_Residency Road	5512.5
3	SBI_Shivaji Road	6615
4	SBI_Parliament Road	9922.5
5	SBI_Jantarmantar	8820
6	SBI_Shivaji Road	4410
8	SBI_Residency Road	4410
9	SBI_Parliament Road	3307.5
10	SBI_Residency Road	5512.5
11	SBI_Jantarmantar	2205

(B) find all customers who have both an account and a loan at the Bangalore branch.

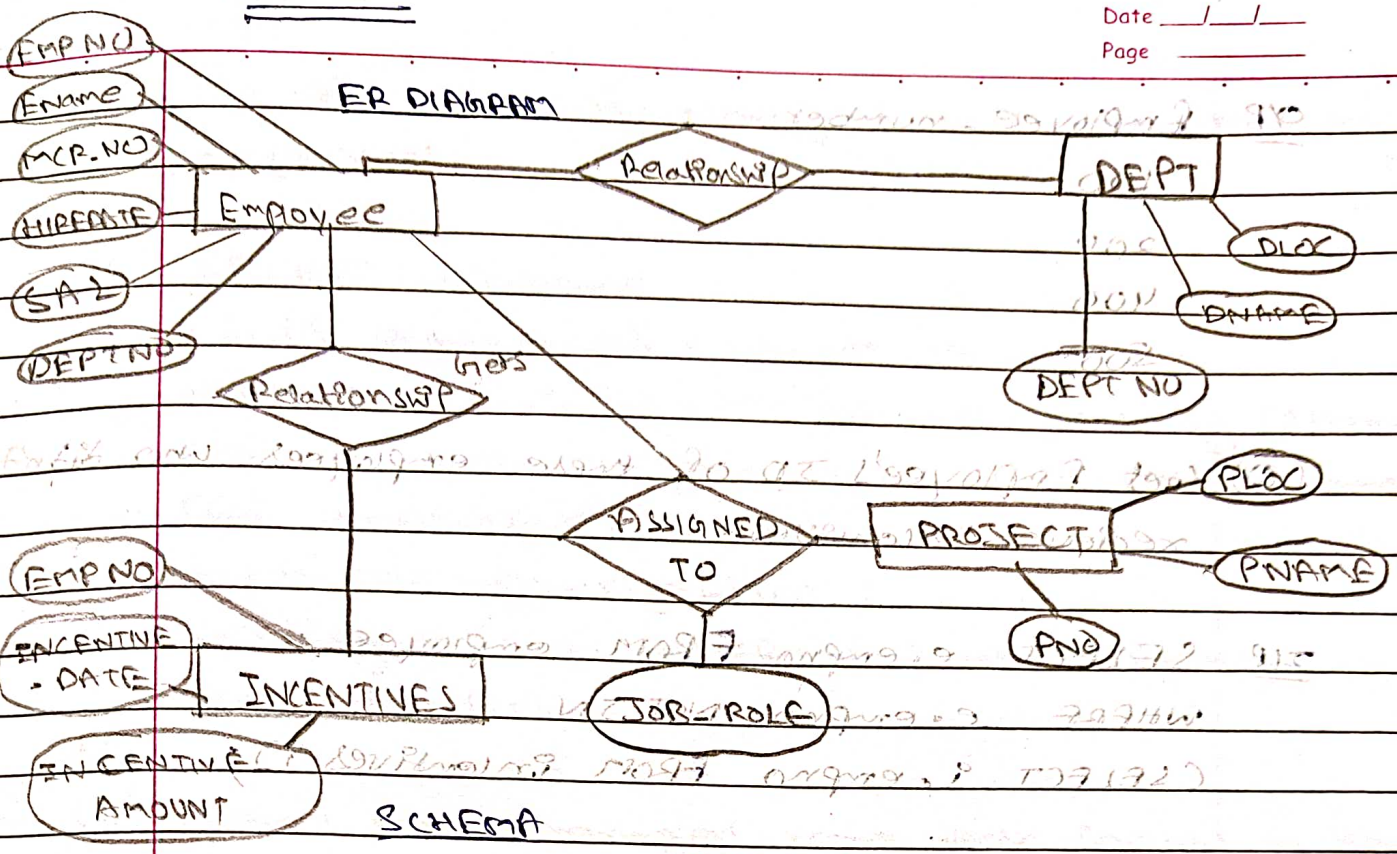
I/P
```
SELECT distinct D.customername FROM
Depositer D WHERE D.customername IN(
    SELECT D.customername FROM branch br,
    Depositer D, bankaccount ba WHERE
    br.Branch_city = 'Bangalore' and
    br.Branch_name = ba.Branch_name and
    ba.accno = D.accno and Customername IN(
        SELECT customername FROM Borrower)
);
```

O/P
SBI_Mantrimarg.

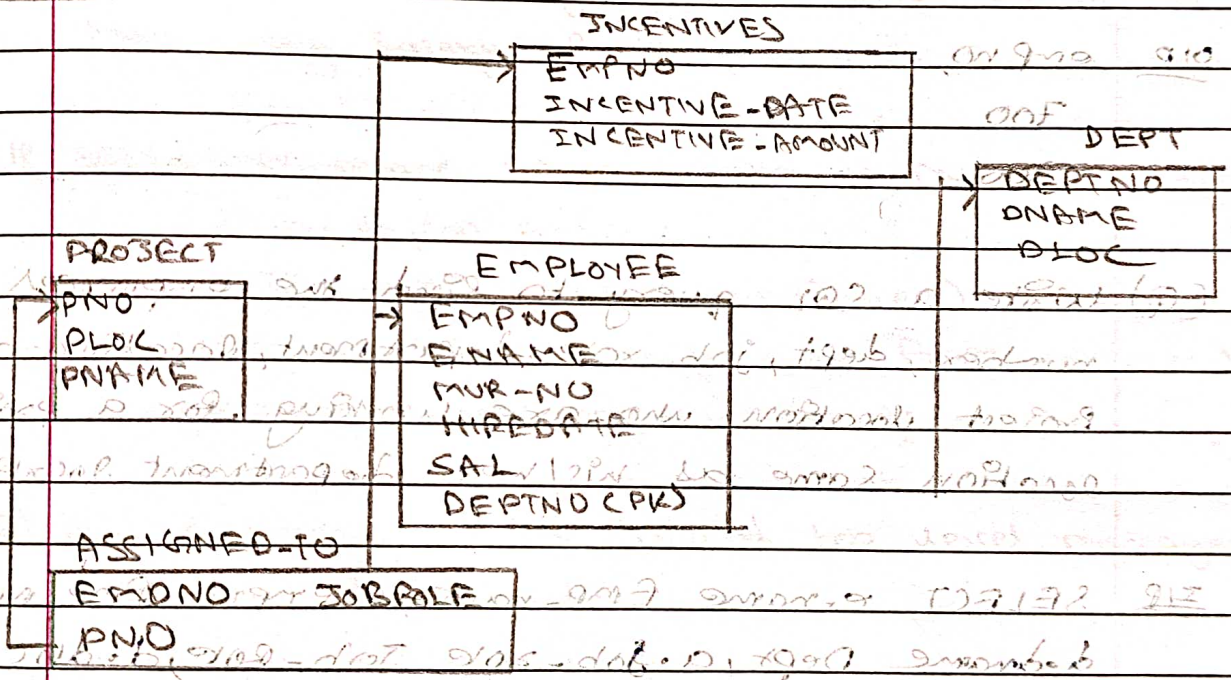
LAB-6

Date __/__/__
Page _____

ER DIAGRAM



SCHEMA



(Q) Retrieve the employee numbers of all employees who work on project located in Bangalore, Hyderabad or Mysore.

I/P SELECT a.empno EMPLOYEE_number FROM Project P, assigned_to a
WHERE p.pno = a.pno and p.ploc in
('Hyderabad', 'Bangalore', 'Mysuru');

Date / /
Page

O/P

Employee_number
100
200
400
500

(Q) Get Employee's ID of these employees who didn't recieve incentives.

I/P
```
SELECT e.empno FROM employee e
WHERE e.empno NOT IN
(SELECT i.empno FROM incentives i);
```

O/P

empno
700
300

(Q) Write a SQL query to find the employees name, number, dept, job_role, department location and project location who are working for a project location same as his/her department location.

I/P
```
SELECT e.name Emp_name, e.empno Emp_number,
d.dname Dept, a.job_role Job_Role, d.dloc Department_Location, p.ploc Project Location FROM
Project p, dept d, employee e, assigned_to a
WHERE e.empno = a.empno and p.no = a.pno
and e.deptno = d.deptno and p.ploc = d.dloc;
```

O/P

Emp_Name	Emp_Number	Dept	Job_Role	Department Location	Project Location
Praveen	100	Sales	Project manager	Bangalore	Bangalore

LAB-7

Date __/__/__
Page _____

(Q) List the name of managers with the maximum employees.

I/P
```
SELECT e1.name
FROM employee e1, employee e2
WHERE e1.empno = e2.mgr_no group by e1.name
HAVING count(e1.mgr=no) = (SELECT count(e1.name)
FROM employee e1, employee e2 WHERE
e1.empno = e2.mgr_no
group by e1.name ORDER BY count(e1.name)
desc( limit 1);
```

(Q) Display those managers name whose salary is more than avg salary of his employee.

I/P
```
SELECT m.ename FROM employee m
WHERE m.empno in(
    SELECT mgr_no FROM employee)
    and m.sal > (SELECT avg(n.sal) FROM employee n
    WHERE n.mgr_no = m.empno);
```

(Q) Find the name of the second top level managers of each department.

I/P
```
SELECT ename FROM employee WHERE empno in
(SELECT distinct mgr_no)
FROM employee WHERE emp_no in
(SELECT distinct mgr_no FROM employee WHERE
empno in (SELECT distinct mgr_no FROM employee)));
```

(Q) Find the employee details who got second maximum incentive in January 2019.

Date / /
Page

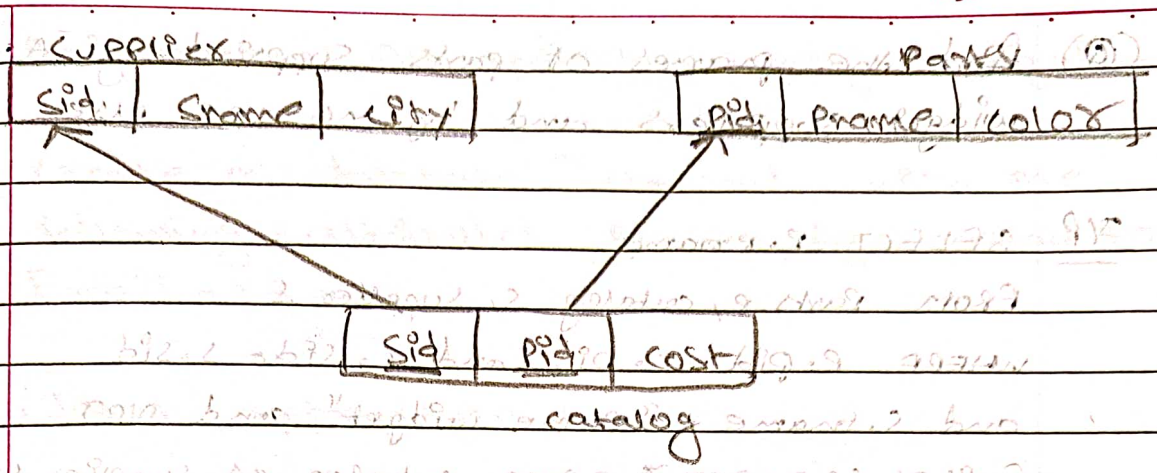
S/P SELECT * FROM employee WHERE empno = (SELECT i.empno FROM incentives i WHERE i.incentive_amount = (SELECT max (in.incentive_amount) FROM incentives in WHERE in.incentive_amount < (SELECT max (inc.incentive_amount) FROM incentives inc WHERE inc.incentive_data between '2019-01-01' and '2013-12-31') and incentive_data between '2019-01-01' and '2019-12-31'));

(d) Display those employees who are working in the same department where his manager is working

S/P SELECT e2.ename
FROM employee e1, employee e2
WHERE e1.empno = e2.mgr_no and
e1.deptno = e2.deptno;

LAB-8

Date __/__/__
Page _____



(Q) Find the pnames of parts for which there is some supplier.

I/P
```
SELECT distinct p.pname
FROM parts p, catalog c
WHERE p.pid = c.pid;
```

(Q) Find the snames of suppliers who supply every part.

I/P
```
SELECT distinct s.sname
FROM catalog c, supplier s WHERE c.sid = s.sid
and NOT EXISTS (SELECT p.pid FROM parts p)
WHERE NOT EXISTS (SELECT c1.sid FROM catalog c1
                  WHERE (c1.sid = c.sid and
                  c1.pid = p.pid));
```

(Q) Find the snames of suppliers who supply every red part.

I/P
```
SELECT distinct s.name
FROM catalog c, supplier s WHERE c.sid = c.sid and
not exists (SELECT p.pid FROM parts p WHERE p.color = 'Red'
and not exists (SELECT c1.sid FROM catalog c1
WHERE c1.sid = c.sid and c1.pid = p.pid and p.color = 'Red'))
```

Date __/__/__
Page

(Q) find the pnames of parts supplied by Acme widget suppliers and by no one else.

I/P
```
SELECT P.Pname
FROM Parts P, catalog C, Supplier S
WHERE P.pid = C.pid and C.Sid = S.Sid
and S.Sname = "Acme widget" and NOT
EXISTS (SELECT * FROM catalog C1, Supplier S1
WHERE P.pid = C1.pid and C1.Sid = S1.Sid and
S1.Sname != "Acme widget");
```

(Q) find the sids of suppliers who charge more for some part than the average cost of that part.

I/P
```
SELECT distinct C.Sid FROM Catalog C
WHERE (C.cost) (SELECT AVG(C1.cost)
FROM Catalog C1 WHERE C1.pid = C.pid);
```

(Q) for each part, find the Sname of supplier who charges the most for that part.

I/P
```
SELECT P.pid, S.Sname
FROM Parts P, Supplier S, Catalog C
WHERE C.pid = P.pid and C.Sid = S.Sid and
C.cost = (SELECT max(C1.cost)
FROM Catalog C1
WHERE C1.pid = P.pid);
```

LAB-9

Date __/__/__
Page _____

Perform the DB operations using mongoDB.

(1) Create a database "student" with the following attributes RollNo, Age, contact NO, Email-id.

(2) Insert appropriate values.

(3) Write query to update Email-Id of a student with roll no 10.

(4) Replace the student name from "ABC" to "FGH" of roll no. 11.

(5) Export the created table into local file System

(6) Drop the table.

(7) Import a given CSV dataset from local file system into mongodb collection.

I/P 1) db.createCollection("Student");

2) db.Student.insert({RollNO:2, Age:21, cont:9876, email:"antara.de.9@gmail.com"});

db.Student.insert({RollNO:2, Age:22, cont:9876, email:"antara9@gmail.com"});

db.Student.insert({RollNO:3, Age:23, cont:5540, email:"anubrava.de8@gmail.com"});

db.Student.insert({RollNO:4, Age:20, cont:44, email:"pant.deg@gmail.com"});

db.Student.insert({RollNO:10, Age:23, cont:22, email:"repvag.deg@gmail.com"});

db.Student.find() //to view data

3) db.Student.update({RollNO:10},{$set:{email:"abhinav@gmail.com"}});

4) db.Student.update({RollNO:11, Name:"ABC"}, {$set:{Name:"FEM"}});

5) Execute at command prompt:

mongoexport mongodb+srv://antarar:"Hi"@clusters0.mbnkeys.mongodb.net/MyDB-collection Student-Out C:\Users\BMSCE CSE\Downloads\output.json

6) FROM "mongo: show dbs
db.Student.drop();

(7) At Command prompt:

mongoimport mongodb+srv://antarar:Test1234@clusters0.mbnkeys.mongodb.net/MyDB-collection=new_student-type json-file C:\Users\BMSCE\Downloads\output.json.

LAB-10

Date __/__/__
Page .

Q-1) Create a collection by name customers with the following attributes => cust_id, Acc_Bal, Acc_type.

Q-2) Insert at least 5 values in a table.

Q-3) Write a query to display those records whose total account balance is greater than 1200 of account type 'z' for each customer_id.

Q-4) Determine minimum and maximum account balance for each customer_id.

Q-5) Export the created collection into local file system

Q-6) Drop the table.

Q-7) Import a given csv dataset from local file system into mongodb collection.

I/P 1) db.createCollection("customer");

2) db.customers.insertMany([
{cust_id:"1", Acc_Bal:1500, Acc_Type:"z"},
{cust_id:"2", Acc_Bal:1000, Acc_Type:"z"},
{cust_id:"3", Acc_Bal:1300, Acc_Type:"z"},
{cust_id:"4", Acc_Bal:800, Acc_Type:"z"},
{cust_id:"5", Acc_Bal:2000, Acc_Type:"z"}]);

3) db.customers.aggregate([
{$match:{Acc_Type:"z"}},
{$group:{_id:"$cust_id", totalBalance:

{$sum: "$ACC-Bal"}}},
{$match: {total Balance: {$gt: 1200}}}])

4) db.customers.aggregate([
{$group: {_id: "$cust-id", minBalance:
{$min: "$ACC-Bal"}, maxBalance:
{$max: "$ACC-Bal"}}});

5) At command prompt:

mongoexport mongodb+srv://antaraare"MH:"
@cluster0.mfnfeys.mongodb.net/myDB-
collection=customer-out c:/users/BMSCE(CSE)
/Downloads/output.json

6) From mongo shell:
db.customer.drop();

7) At command prompt:

mongoimport mongodb+srv://antaraa:c:Test
1234@cluster0.mfnfeys.mongodb.net/
myDB-collection=NewCustomer type json-
file c:/users/BMSCE(CSE/Downloads/output